UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA DE SISTEMAS INFORMÁTICOS

CURSO DE ESPECIALIZACIÓN INGENIERÍA DE CALIDAD

IMPLEMENTACIÓN DE PRUEBAS DE CALIDAD Y EJECUCIÓN DE PRUEBAS AUTOMATIZADAS DEL SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONTRATOS DE PERSONAL DOCENTE DE LA FACULTAD DE INGENIERÍA Y ARQUITECTURA DE LA UNIVERSIDAD DE EL SALVADOR

PRESENTADO POR:

GRISELDA ROSIBEL FERNÁNDEZ BÉNITEZ

JOSÉ RENÉ LUCERO BARRERA

RONALD ANTONIO SERMEÑO AGUILAR

PARA OPTAR AL TÍTULO DE:

INGENIERO(A) DE SISTEMAS INFORMÁTICOS

CIUDAD UNIVERSITARIA, JULIO DE 2025

UNIVERSIDAD DE EL SALVADOR

RECTOR :

Msc. JUAN ROSA QUINTANILLA

SECRETARIA GENERAL :

LIC. PEDRO ROSALIO ESCOBAR CASTANEDA

FACULTAD DE INGENIERIA Y ARQUITECTURA

DECANO :

ING. LUIS SALVADOR BARRERA MANCIA

SECRETARIO :

ARQ. RAUL ALEXANDER FABIAN ORELLANA

ESCUELA DE SISTEMAS INFORMATICOS

DIRECTOR :

ING. CESAR AUGUSTO GONZALES

UNIVERSIDAD DE EL SALVADOR

FACULTAD DE INGENIERIA Y ARQUITECTURA

ESCUELA DE

Curso de Especialización previo a la opción al Grado de:

INGENIERO(A) DE SISTEMAS INFORMATICOS

Título :

CURSO DE ESPECIALIZACION DE INGENIERIA DE CALIDAD

IMPLEMENTACIÓN DE PRUEBAS DE CALIDAD Y EJECUCIÓN DE PRUEBAS AUTOMATIZADAS DEL SISTEMA INFORMÁTICO PARA LA GESTIÓN DE CONTRATOS DE PERSONAL DOCENTE DE LA FACULTAD DE INGENIERÍA Y ARQUITECTURA DE LA UNIVERSIDAD DE EL SALVADOR

Presentado por :

GRISELDA ROSIBEL FERNANDEZ BENITEZ

JOSE RENE LUCERO BARRERA

RONALD ANTONIO SERMEÑO AGUILAR

Curso de Especialización Aprobado por:

INGA. SANDRA GUADALUPE ROMERO

San Salvador, diciembre de 2025

Contenido

Resumen

El proyecto tiene como propósito implementar un proceso integral de pruebas de calidad, tanto manuales como automatizadas, para el Sistema Informático de Gestión de Contratos del Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador. Este sistema fue desarrollado para optimizar procesos administrativos como el registro de candidatos, la validación de documentos, la asignación de actividades docentes y la generación de contratos. Sin embargo, durante su desarrollo se identificó una limitación importante: no existía un proceso formal, consistente y continuo de pruebas que garantizara la confiabilidad y el correcto funcionamiento del software.

Para resolver esta problemática, el proyecto establece un enfoque estructurado que inicia con un diagnóstico del sistema, donde se analizan aspectos funcionales, arquitectónicos, riesgos y procesos críticos. Se aplican herramientas como la matriz FODA, el diagrama causa-efecto y el enfoque de sistemas para comprender las causas de posibles fallos y definir estrategias que fortalezcan la calidad del software. Posteriormente, se realiza un levantamiento completo de requerimientos, se diseña una matriz de casos de prueba y se definen los criterios de aceptación para cada funcionalidad, alineándolos con los principios de calidad establecidos por estándares internacionales.

El proyecto desarrolla pruebas manuales para validar flujos que requieren criterio humano, como la revisión visual de documentos, la evaluación de usabilidad y la interpretación de reglas de negocio. Estas actividades se complementan con un conjunto de pruebas automatizadas construidas con herramientas Open Source como Selenium WebDriver y Postman. Dichas automatizaciones permiten validar flujos repetitivos como login, validación de permisos, creación de ciclos, gestión de grupos y procesos de contratación de forma rápida, precisa y altamente repetible, reduciendo tiempos de regresión y minimizando errores humanos. Asimismo, se documentan los scripts, se generan reportes de ejecución y se establecen métricas que permiten medir la calidad del sistema de forma objetiva.

Todo este proceso se encuentra respaldado por normas internacionales como la ISO/IEC 29119, que establece lineamientos para la planificación, diseño, ejecución y documentación de pruebas; la ISO/IEC 25010, que define los atributos de calidad del software (funcionalidad, confiabilidad, usabilidad, seguridad, mantenibilidad y eficiencia); y la ISO/IEC 12207, que orienta la gestión del ciclo de vida del software. Su aplicación garantiza que el proyecto no solo cumpla con las necesidades funcionales de la Facultad, sino que también siga prácticas reconocidas mundialmente para asegurar la calidad, trazabilidad y evaluabilidad del sistema informático.

Finalmente, el proyecto presenta un análisis comparativo entre pruebas manuales y automatizadas, evidenciando mejoras significativas en cobertura, eficiencia y detección temprana de fallas. Este trabajo garantiza que el sistema opere de forma confiable, segura y eficiente, fortaleciendo la modernización administrativa de la Facultad y estableciendo bases sólidas para futuras mejoras, ampliaciones y auditorías de calidad.

Índice de Figuras

Índice de Tablas

1. Introducción

La transformación digital se ha convertido en una prioridad para las universidades, que buscan optimizar sus procesos administrativos y académicos a través de sistemas informáticos eficientes. En este contexto, la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador ha desarrollado un Sistema Informático para la Gestión de Contratos de Personal Docente, con el propósito de facilitar tareas esenciales como el registro de candidatos, la revisión de documentos, la asignación de clases y la generación de contratos.

La implementación de un proceso de pruebas de calidad de software tiene como objetivo mejorar de forma eficiente la verificación de los requerimientos funcionales y no funcionales del sistema. Este proceso permitirá identificar errores y fallas dentro de dicho sistema, contribuyendo a la prevención de defectos y la reducción de riesgos, lo que aumentará la confianza de los usuarios, como autoridades, personal administrativo, recursos humanos y docentes, en el sistema de contratación.

Además, con esta iniciativa se busca garantizar la funcionalidad del sistema en diversas condiciones operativas, brindando información clave para la toma de decisiones y facilitando un enfoque más eficiente para el mantenimiento y soporte del sistema. Esto, a su vez, sentará una base para futuros desarrollos de los sistemas informáticos dentro de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador (FIA-UES).

Este proyecto presenta un proceso exhaustivo de pruebas de calidad de software que servirá como guía para organizar y ejecutar las pruebas automatizadas. El objetivo es verificar el comportamiento de los principales módulos del sistema y comparar sus resultados con los obtenidos mediante pruebas manuales. Esto permitirá analizar el impacto de la automatización en la mejora de la calidad institucional, definiendo los alcances, las decisiones tomadas para desarrollar las pruebas, y la documentación a entregar.

Se identifican las principales historias de usuario, se diseñan los casos de prueba y se presentan los resultados de las ejecuciones, junto con los defectos encontrados. Finalmente, se proporcionarán recomendaciones relevantes relacionadas con el sistema probado, así como las conclusiones del trabajo desarrollado.

2. Objetivos

2.1 Objetivo General

Diseñar e implementar una matriz de pruebas automatizadas funcionales para validar los módulos del Sistema Informático para la Gestión de Contratos de Personal Docente, asegurando su eficiencia, fiabilidad y cumplimiento de los requisitos establecidos, con el fin de optimizar el proceso de validación y mejorar la calidad del sistema.

2.2 Objetivos Específicos

1. Realizar un análisis exhaustivo del sistema para identificar los módulos críticos y los procesos claves relacionados con la gestión de contratos del personal docente, con el fin de priorizar las pruebas más relevantes.
2. Diseñar una matriz de pruebas de calidad que cubra los casos de uso más frecuentes del sistema, validando tanto las funcionalidades principales como las interacciones entre diferentes módulos.
3. Implementar las pruebas automatizadas utilizando herramientas como Selenium, para verificar la correcta ejecución de los procesos del sistema, asegurando que las funcionalidades operen conforme a los requisitos especificados.
4. Realizar un análisis comparativo entre los resultados obtenidos de las pruebas manuales y las automatizadas, evaluando la eficiencia, cobertura y tiempos de ejecución, con el fin de demostrar las ventajas de la automatización.
5. Generar documentación de forma detallada de los casos de pruebas automatizadas, resultados obtenidos y recomendaciones de mejora, para proporcionar una base sostenible para futuras validaciones del sistema.
6. Elaborar guías manuales de uso para la ejecución y mantenimiento de las pruebas automatizadas, con el fin de garantizar la transferencia de conocimiento con el equipo de calidad.

3. Limitaciones y Alcances

3.1 Limitaciones

Para llevar a cabo la evaluación y automatización de pruebas de calidad de software sobre el Sistema Informático para la Gestión de Contratos de Personal Docente, es importante reconocer una serie de limitaciones que pueden influir en el alcance, profundidad y efectividad del trabajo realizado. Estas limitaciones responden a factores técnicos, organizativos y operativos, propios del entorno institucional y académico del proyecto.

Infraestructura Tecnológica

1. Limitaciones de hardware y red: Las pruebas se ejecutan en equipos personales y entornos locales, lo que impide realizar pruebas de alto consumo de recursos como las de carga de estrés.

Resistencia al cambio

1. Adopción de recomendaciones: Si bien se presentarán resultados y propuestas de mejora, la adopción de las recomendaciones dependerá de la disposición de los responsables institucionales del sistema.
2. Procesos administrativos tradicionales: Algunos procesos aún dependen de prácticas manuales consolidadas que podrían afectar la integración de algunos enfoques automatizados a corto plazo, por ejemplo: La gestión de registros y aprobaciones manuales aún requiere intervención humana, lo que representa un desafío para una transición completa hacia soluciones automatizadas.

3.2 Alcances

Alcances del Proyecto

Este proyecto se enfoca en la evaluación funcional y automatización de pruebas del Sistema Informático para la Gestión de Contratos de Personal Docente de la (FIA-UES). El trabajo estará limitado a los siguientes puntos claves:

Se priorizarán las pruebas sobre los siguientes módulos:

1. **Identificación de procesos críticos.** Se ejecutarán pruebas funcionales orientadas a los módulos más sensibles y de mayor uso durante los ciclos académicos, tales como registro de candidatos, validación documentos, generación de contratos y asignación de carga académica, ya que son los módulos que concentran las funciones más utilizadas en cada ciclo, ya que generan mayor riesgo en caso de fallos por esa razón se llevarán a cabo pruebas exploratorias de cada función que hace el sistema con el fin de detectar posibles bugs.
2. **Documento técnico con casos de pruebas.** Se elaborará una matriz de pruebas funcionales que describa escenarios representativos de uso del sistema. Esta matriz incluirá entradas, salidas esperadas, tipos de pruebas, precondiciones, resultados esperados y evidencias. Con el objetivo para asegurar la trazabilidad entre los

requerimientos del sistema y pruebas, garantizando que se cubran todas las funcionalidades críticas.

3. **Scripts automatizados con documentación técnica**. A partir de la matriz de pruebas, se seleccionarán los casos más relevantes para ser automatizados mediante herramientas como Selenium WebDriver, los casos de prueba que son repetitivos y de alta frecuencia (por ejemplo: Login, generación de contratos, validación de ciclos). Automatizarlos reducirá tiempo y errores humanos.
4. **Evidencias de ejecución de pruebas.** Las pruebas serán ejecutadas en un entorno controlado (no productivo) y se documentarán los resultados. Se generarán evidencias, informes de defectos detectados y métricas de desempeño, con el objetivo de confirmar si el sistema responde correctamente en condiciones reales simuladas y así asegurar confiabilidad.
5. **Informes comparativos de eficiencia (manual vs automatizado).** Se compararán los resultados obtenidos mediante pruebas manuales y automatizadas, evaluando tiempos de ejecución, esfuerzo requerido y cobertura funcional.
6. **Manual técnico del proceso de automatización.** El proyecto contemplará la entrega de documentación técnica que incluya: la matriz de pruebas, evidencias de ejecución, informe de resultados y recomendaciones para su uso futuro. Es decir, se entregará un plan de pruebas que documente los procedimientos, herramientas que se usaron para identificar si el sistema cumple con los estándares de calidad.
7. **Dashboard**. Se presentará de forma gráfica y resumida datos claves para el seguimiento y control de pruebas automatizadas del sistema.
8. **Exclusiones**. El proyecto contempla pruebas de seguridad no muy avanzadas, así como pruebas de carga o estrés, por el motivo de que se cuenta con el sistema de forma local. Tampoco se realizarán modificaciones al código fuente del sistema

CAPÍTULO I. Fundamentos de Calidad y Pruebas de Software en el Sistema de Gestión de Contratos de Personal Docente de la FIA-UES

1.1 Calidad de Software en Sistemas Web Institucionales

1.1.1 Fundamentos de la Calidad del Software

La calidad del software es un atributo esencial que determina el grado en que un sistema o aplicación cumple con los requerimientos funcionales y no funcionales, satisfaciendo las necesidades de los usuarios y garantizando su correcto desempeño dentro del entorno operativo para el que fue diseñado. La calidad de software se evalúa a partir de características como fiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad, las cuales deben ser consideradas desde las etapas iniciales del ciclo de vida del desarrollo de un sistema.

En el marco de este sistema, los fundamentos de la calidad del software se aplican directamente a:

- La eficiencia y seguridad del sistema, asegurando un desempeño óptimo y protección de la información.
- La usabilidad, garantizando que los usuarios finales puedan interactuar con el sistema de manera intuitiva.
- La mantenibilidad, facilitando mejoras futuras sin afectar la operatividad.

El cumplimiento de estos fundamentos asegura que el sistema desarrollado sea confiable, seguro y adaptable, alineándose con los objetivos y necesidades de la Organización.

1.1.2 Definición e Importancia de la Calidad del Software

La calidad del software puede definirse como el grado en que un sistema o aplicación satisface los requerimientos especificados y las expectativas de los usuarios, garantizando su correcto funcionamiento en un entorno determinado. Según la norma ISO/IEC 25010, esta calidad se evalúa a través de atributos como funcionalidad, confiabilidad, usabilidad, eficiencia, seguridad, mantenibilidad y portabilidad, los cuales permiten determinar si un producto cumple su propósito y aporta valor real a quienes lo utilizan.

La importancia de la calidad del software trasciende el ámbito técnico y se refleja directamente en la confianza que los usuarios depositan en un sistema. Un software de calidad asegura que los procesos institucionales o empresariales que dependen de él se ejecuten de manera confiable, reduciendo riesgos de errores, pérdidas de información o interrupciones que puedan afectar la continuidad operativa. En el caso de sistemas administrativos y académicos, como el de la Gestión de Contratos Docentes de la FIA-UES, garantizar la calidad significa brindar transparencia, eficiencia y seguridad en procesos que tienen implicaciones legales y organizativas.

Asimismo, la calidad del software se vincula estrechamente con la mejora continua y con la sostenibilidad tecnológica a mediano y largo plazo. Integrar prácticas de aseguramiento de calidad desde las primeras etapas del ciclo de vida del software permite prevenir defectos en lugar de solo corregirlos, reduciendo costos de mantenimiento y facilitando futuras actualizaciones.

Otro aspecto fundamental radica en que la calidad no solo se mide por la ausencia de fallos, sino también por la capacidad del software de adaptarse al cambio, responder a escenarios diversos y ofrecer una experiencia de uso intuitiva y confiable. En palabras de Sommerville (2011), un sistema de calidad debe ser útil, seguro y mantenible, más allá de que funcione sin errores visibles.

1.1.3 Modelos y Estándares de Calidad

Los modelos de calidad de software evalúan el ciclo de vida en todas las etapas para culminar con el producto informático final, estos modelos se basan en los estándares que han sido estipulados por organizaciones internacionales, un ejemplo es la conocida Organización Internacional de Normalización también conocida por sus siglas en ingles ISO (International Organization for Standardization).

Gracias a las normas establecidas por las ISO se asegura que los productos y servicios cumplan con las expectativas del cliente consumidor final, esto también potencia el desarrollo de prácticas competitivas dentro de la industria del software.

De acuerdo con ISO existen siete principios que son pueden ser medidos en la gestión de calidad, los cuales son los siguientes:

1. Enfoque al cliente: se basa en comprender las necesidades de los clientes con el fin de lograr satisfacerlas o superarlas.
2. Liderazgo: los lideres dentro de un proyecto informático forman un pilar importante ya que establecen la unidad de propósito y dirección así mismo, crean un entorno para que las personas logren cumplir con los objetivos de la organización.
3. Compromiso por las personas: para crear y entregar a la organización valor, es necesario que cada persona participante en cualquier nivel debe encontrarse empoderada, competente y comprometida con los objetivos establecidos.
4. Enfoque basado en procesos: gestionar los recursos y actividades de manera interrelacionada como un proceso permitirá obtener resultados consistentes y más eficientes.
5. Mejora Continua: la mejora continua del desempeño global de la organización debe ser un objetivo que permanezca.
6. Toma de decisiones basadas en evidencias: la toma de decisiones e realizan a partir de análisis de datos e información.
7. Gestión de relaciones: la gestión de las relaciones entre los interesados aumenta la probabilidad de éxito sostenido.

Por lo cual la disciplina de ingeniería de software ha desarrollados diversos modelos y estándares para permitir medir, evaluar y garantizar la calidad durante el ciclo de vida del software.

1.1.3.1 Modelos de Calidad de Software.

1. **Modelo de McCall.**

El modelo de evaluación de McCall fue uno de los primero creado para medir la calidad de software, este fue creado en el año 1,977, fue propuesto por James A. McCall, Paul K. Richards y Gene F. Walter. Este es un modelo que está considerado desde la percepción del usuario y propone tres factores los cuales son conocidos como factores McCall.

Estos tres factores son la operación, transición y revisión. Cada una de estas capacidades tienen consigo un conjunto de factores que definen ciertas métricas que permiten evaluar el producto. Cada métrica cuenta con criterios propios o medidas que posibilitan la medición de la calidad.

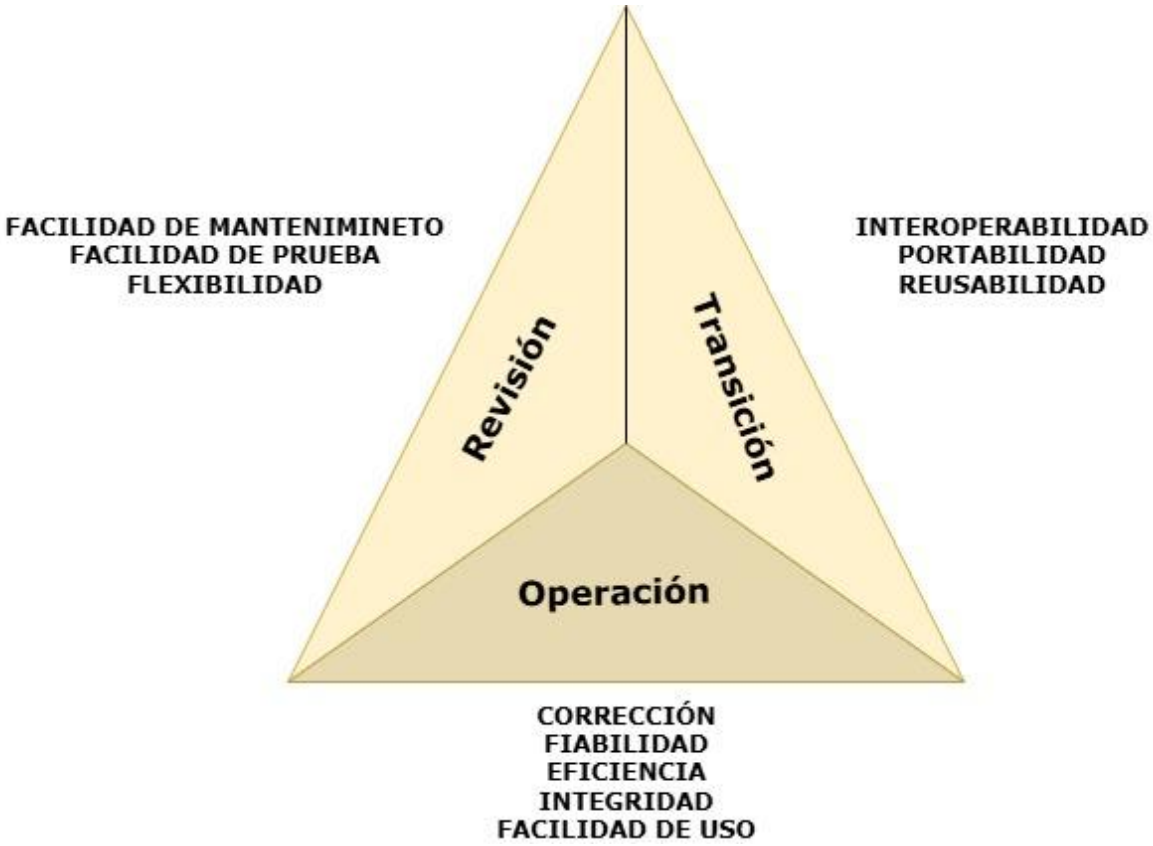


Ilustración 1 Triangulo de Métricas de Medición de la Calidad de Software

Como podemos observar en la imagen podemos comentar que las métricas de McCall con su modelo facilitan un poco más la evaluación, permitiendo evaluar de una manera menos compleja la calidad del producto. El modelo de McCall provee un sistema de calidad ya establecido a quien desee hacer uso de este.

2. M**odelo de Boehm.**

Según nos indica Velazco “Es un modelo incremental, divido en regiones de tareas y estas a su vez en conjunto de tareas, las cuales se ajustan a la cantidad de iteraciones que el equipo defina, y cada iteración se divide en cuatro sectores los cuales son planeación, análisis de riesgos, ingeniería y evaluación”.

Para el año 1,978, Barry Boehm propuso un modelo de calidad en términos de atributos cualitativos y métricas para realizar medidas. Este modelo se basa en que el software debe hacer lo que el usuario quiere que haga:

- El uso de los recursos de la computadora debe realizarse de manera correcta y eficiente.
- Para los diferentes usuarios debe percibir el sistema de fácil uso y entendimiento.
- El diseño debe ser bien codificado y probado, para mantenerse fácilmente.

Entre los niveles y características del modelo Boehm tenemos los siguientes:

Nivel	Características
Nivel Primario	Independencia, completitud, exactitud, consistencia, eficiencia, accesibilidad, comunicación, estructuración, autodescripción, concisión, legibilidad y expansión.
Nivel Intermedio	Portabilidad, fiabilidad, eficiencia, usabilidad, capacidad de prueba, compresibilidad y flexibilidad.
Nivel Alto	Utilidad, mantenimiento y portabilidad.

Tabla 1 Niveles y Características Modelo Boehm.

El modelo de Boehm es considerado como visionario, debido que incluyo atributos de calidad interna.

3. Modelo de Calidad FURPS.

Este modelo fue propuesto en 1987 por Robert Grady y Hewlett Packard. Este modelo desarrolla 5 factores y a los cuales les debe su nombre por sus siglas en inglés: Functionality (Funcionalidad), Usability (Usabilidad), Reliabity (Confiabilidad), Performance (Presentación) y Supportability (Soporte). A continuación, se muestra una gráfica para validar la relación de los factores:

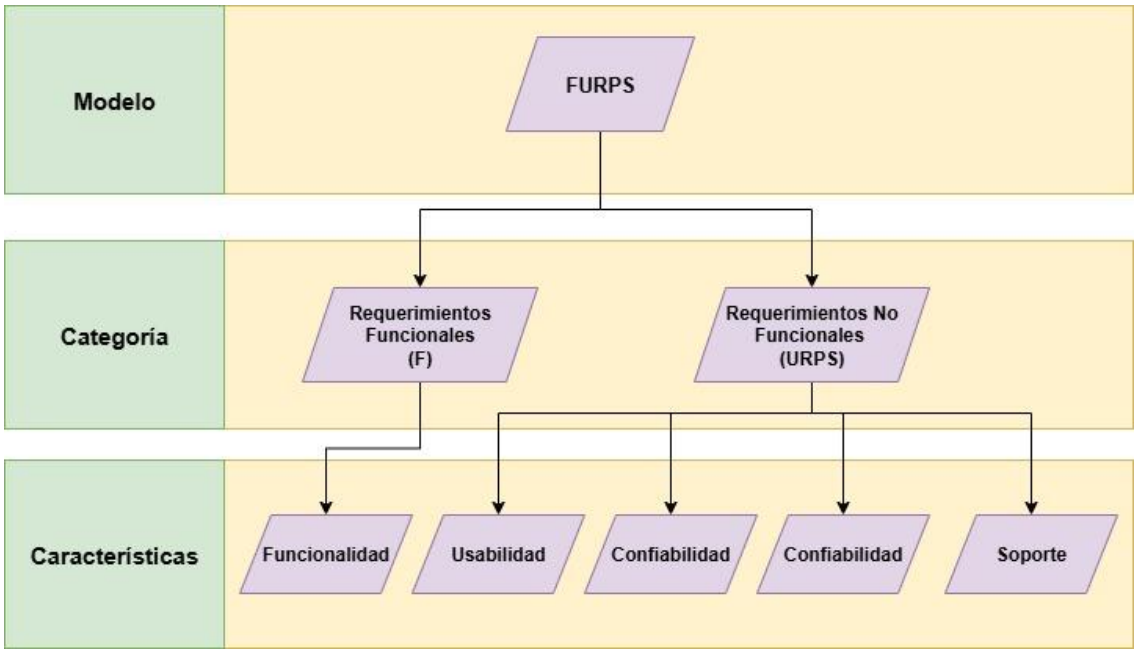


Ilustración 2 Relación de Factores Modelo de Calidad FURPS.

También puede incluir subcategorías tales como: Factores humanos, estética, consistencia, ayuda en línea, asistentes, documentación, material de capacitación.

Este modelo es ampliamente utilizado en el ámbito del desarrollo de software empresarial, especialmente en la fase de recolección y análisis de requisitos. El modelo de FURPS es un modelo de calidad fijo y para realizar la evaluación de la calidad de un producto, primero se asignan prioridades y después se definen los atributos de calidad que pueden ser medidos.

4. **Modelo ISO/IEC 9126**

A pesar de que este es clasificado oficialmente como un estándar internacional, pero dentro de este estándar se propone un modelo de calidad de software el cual cuenta con 6 características: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Esta norma es una de las normas ISO que goza con más reconocimiento dentro de la comunidad y tiene como fundamento modelos de calidad por diversas investigaciones.

El modelo de calidad que propone la norma puede aplicarse a cualquier tipo de software incluido el desarrollo para el ámbito educativo.

La ISO 9126 propone dos modelos de calidad: un modelo para la calidad interna y externa, a continuación, definimos en un diagrama las características y sub características.

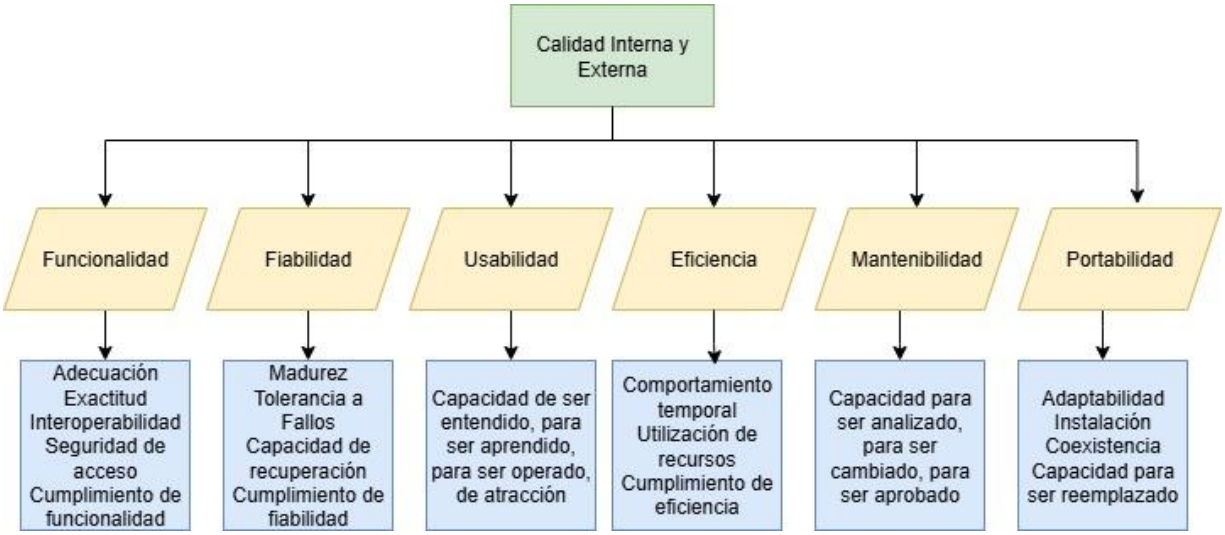


Ilustración 3 Características y Subcaracterísticas de la ISO 9126.

5. Modelo ISO/IEC 25010

Este modelo de calidad es la evolución del ISO 9126, y es uno de los modelos más aceptados en la actualidad. En este modelo se determinan características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto de software.

La calidad del software podemos interpretarlo en qué grado el producto final cubre las necesidades del usuario final aportando valor. Estos requisitos que ofrecen valor son los que se encuentran estipulados para representar este modelo.

El modelo de calidad basado en la ISO/IEC 25010 se divide en nueve características que se muestran a continuación:

Calidad del Software							
Adecuación Funcional	Eficiencia de Desempeño	Compatibilidad	Fiabilidad	Seguridad	Mantenibilidad	Flexibilidad	Protección
1. Completitud Funcional. 2. Corrección Funcional. 3. Pertenencia Funcional	1. Comportamiento Temporal. 2. Utilización de Recursos. 3. Capacidad	1. Coexistencia 2. Interoperabilidad	1. Reconocibilidad de adecuación. 2. Apredizabilidad. 3. Operabilidad. 4. Protección frente a errores de usuario. 5. Involucración del usuario. 6. Inclusivilidad. 7. Asistencia al usuario. 8. Auto descriptividad	1. Confidencialidad. 2. Integridad. 3. No Repudio. 4. Responsabilidad. 5. Autenticidad. 6. Resistencia.	1. Modularidad. 2. Reusabilidad. 3. Analizabilidad. 4. Capacidad de ser modificado. 5. Capacidad de ser probado.	1. Adaptabilidad. 2. Escalabilidad. 3. Instalabilidad. 4. Reemplazabilidad.	1. Restricción Operativa. 2. Indentificación de riesgos. 3. Protección de fallos. 4. Advertencia de peligro. 5. Integración segura.

Ilustración 4 Modelo de Calidad Basado en la ISO/IEC 25010

1.1.3.2 Estándares de Calidad de Software

1. ISO 9001.

   La norma ISO 9001 es un estándar internacional que ha sido adoptado por empresas de todo tipo y tamaño alrededor del mundo. El estándar especifica los requerimientos para la implantación de un Sistema de Gestión de la Calidad y así mismo recoge las mejores prácticas para la aplicación ya sea interna, para certificación o fines contractuales.

   Adicional a ello esta norma es compatible con otro tipo de ISO (Como por ejemplo la ISO 14001 de gestión ambiental). La importancia de la ISO 9001 radica en tres puntos importantes: la confianza de los clientes y la diferenciación de la marca, el aumento de la estabilidad en el desarrollo y el fomento de la participación y liderazgo de los trabajadores de la empresa y organización.

   Las características claves suelen incluir:

   - Control de documentos.
   - Acciones correctivas y preventivas.
   - Seguimiento de la formación y competencias.
   - Auditorías internas.
   - Gestión de riesgos.
   - Gestión de proveedores.

   También la ISO 9001 proporciona a las organizaciones cuatro capacidades clave que mejoran los resultados:

   - Planificación de la calidad.
   - Control de calidad.
   - Garantía de calidad.
   - Mejora de la calidad.

2. ISO/IEC 12207

   La norma ISO/IEC 12207 proporciona un marco de referencia para establecer y mejorar los procesos de ingeniería de software de una organización. En esta norma se establece un conjunto de procesos que cubren todo el ciclo de vida del producto, desde el inicio y conceptualización hasta operatividad, mantenimiento y desactivación de este. En resumen, estable un conjunto de actividades y tareas que deben ser realizadas en cada fase de desarrollo y operación del software.

   Los principales procesos que abarca la normativa ISO/IEC 12207 son los siguientes:

   - Procesos primarios: adquisición, suministro, desarrollo, operación y mantenimiento.
   - Procesos de apoyo: documentación, control de calidad y validación.
   - Procesos organizacionales: gestión de la configuración, gestión de proyectos y revisión y auditoria.

3. ISO/IEC 15504 (SPICE)

   El estándar internacional ISO/IEC 15504 denominado Software Process Improvement Capability Determination (“Determinación de la Capacidad de Mejora del Proceso de Software”), también conocido por sus siglas en ingles SPICE, nos propone

un modelo para la evaluación de la capacidad en los procesos de desarrollo de productos de Software. La ISO/IEC 15504 cuenta con los siguientes objetivos a comprender:

- Proponer y evaluar un estándar de evaluación de procesos de software.
- Mediante la experimentación en la industria de desarrollo de software evaluar el desempeño.
- Promover la transferencia de tecnología de evaluación de procesos de software a la industria del software.

La norma SPICE estable requisitos para una evaluación de procesos y los modelos de evaluación pretendiendo que estos requisitos puedan ser aplicados en cualquier modelo de evaluación en una organización, la evaluación de los requisitos comprende: evaluación de procesos, mejora de procesos y evaluación de la capacidad y madurez de los procesos.

4. CMMI (Capability Maturity Model Integration)

Es un modelo que nos mencione qué buenas prácticas implementar organizadas por capacidades críticas del negocio con el objetivo de mejorar su rendimiento. Estas capacidades abordan los principales desafíos a los cuales una organización se enfrenta. Actualmente quien gestiona CMMI es el CMMI Institute, que es una empresa subsidiaria de ISACA.

CMMI define cinco niveles de madurez: inicial, gestionado, definido, cuantitativamente gestionado y optimizado.

1.2 Pruebas de Software en el Sistema de Gestión de Contratos

1.2.1 El Rol de las Pruebas en el Aseguramiento de Calidad

El Aseguramiento de la Calidad del Software (SQA, por sus siglas en inglés) es un proceso sistemático que busca garantizar que los productos de software cumplan con estándares definidos de calidad, mediante actividades de verificación, validación y mejora continua. Dentro de estas actividades, las pruebas de software son fundamentales, ya que permiten identificar defectos, comprobar que los requerimientos han sido implementados correctamente y asegurar la confiabilidad del sistema antes de su puesta en producción (Pressman, 2014).

Función de las Pruebas en la Calidad

Las pruebas cumplen con dos objetivos principales:

- **Verificación:** Confirmar que el software se desarrolla de acuerdo con las especificaciones establecidas.
- **Validación:** Confirmar que el software final satisface las necesidades del usuario y responde al propósito para el que fue creado (Sommerville, 2011).

Tipos de Pruebas.

- **Pruebas de seguridad:** Validar que el sistema rechace accesos no autorizados y proteja adecuadamente la información sensible de la Organización.
- **Pruebas de rendimiento:** Simular múltiples usuarios accediendo al sistema para comprobar que los tiempos de respuesta se mantengan dentro de parámetros aceptables.

- **Pruebas de integración:** Verificar que los distintos módulos del sistema (gestión de usuarios, consultas, reportes, etc.) trabajen de manera conjunta sin errores.
- **Pruebas de aceptación:** Permitir que los usuarios finales interactúen con el sistema en un entorno de prueba, validando que la solución cumpla con las necesidades reales de la Organización.
- **Pruebas de usabilidad:** Evaluar que la interfaz sea intuitiva y fácil de aprender, reduciendo la curva de adaptación de los administradores y usuarios finales.

Más allá de la detección de errores, las pruebas aportan beneficios estratégicos reduciendo riesgos de interrupciones en los procesos críticos e Incrementando la confianza de los usuarios en la solución tecnológica, respaldando todo en el cumplimiento de estándares internacionales de calidad, como la ISO/IEC 25010 y las normas del IEEE en el ámbito de pruebas de software.

1.2.2 Pruebas de Software: Conceptos Esenciales

Las pruebas de software representan una actividad esencial dentro del desarrollo y aseguramiento de la calidad, cuyo propósito principal es verificar y validar que un sistema cumple con los requisitos establecidos, funciona de manera correcta en su entorno operativo y satisface las expectativas de sus usuarios. En términos prácticos, las pruebas permiten identificar errores, inconsistencias o desviaciones antes de la puesta en producción, minimizando riesgos y asegurando la confiabilidad del producto final.

De acuerdo con la norma ISO/IEC 29119, las pruebas deben planificarse, diseñarse, ejecutarse y documentarse de forma sistemática, entendidas no como una fase aislada, sino como un proceso transversal que acompaña todo el ciclo de vida del software. Este enfoque garantiza no solo la detección temprana de defectos, sino también la mejora continua del producto.

Los conceptos esenciales pueden resumirse en tres pilares fundamentales:

Verificación y Validación:

- La verificación busca confirmar que el software ha sido construido correctamente conforme a especificaciones técnicas y de diseño.
- La validación confirma que el sistema satisface las necesidades reales del usuario final y cumple con sus expectativas funcionales y de rendimiento.

Tipos de Pruebas:

- Existen múltiples categorías que permiten evaluar distintos aspectos del sistema. Entre las más importantes se destacan:
- Pruebas funcionales, que verifican que las funcionalidades se comporten según los requerimientos.
- Pruebas de integración, que revisan la correcta interacción entre módulos.
- Pruebas no funcionales, enfocadas en aspectos como seguridad, usabilidad, rendimiento y confiabilidad.

Ciclo de Vida de las Pruebas:

Las pruebas deben integrarse desde etapas tempranas del desarrollo, incluyendo planificación, diseño, ejecución, análisis de resultados y documentación. Este ciclo iterativo permite no solo anticipar defectos, sino también fortalecer la sostenibilidad y mantenibilidad del software a largo plazo.

En el contexto del Sistema de Gestión de Contratos Docentes de la Facultad de Ingeniería y Arquitectura de la UES, aplicar un enfoque riguroso en las pruebas resulta indispensable. Esto garantiza que procesos administrativos críticos como la contratación docente, la validación de documentos y la generación de contratos se realicen de manera eficiente, transparente y segura.

1.2.3 Definición, Objetivos y Beneficios de las Pruebas de Software

Definición de Pruebas de Software.

Para lograr una definición amplia tomaremos de referencia definiciones que nos proporcionan diferentes autores, estándares y organismos reconocidos:

Myers: “La prueba de software es el proceso de ejecutar un programa con el fin de demostrar que contiene errores”. (Myers, 1979, p. 6).

Pressman: “Probar un programa consiste en ejecutarlo con la intención de encontrar errores”. (Pressman, 2010, p. 449). .

Sommerville: “La prueba de software es el proceso de ejecutar un sistema con el fin de encontrar defectos y asegurar que cumpla con los requisitos del usuario”. (Sommerville, 2011, p. 328).

IEEE 829: “La prueba de software es un proceso controlado que implica la planificación, preparación y ejecución de pruebas, así como la evaluación de resultados, para verificar que un sistema cumpla con las especificaciones”. (IEEE, 1998, p. 12).

ISTQB: “Es el proceso de evaluar un producto de software para determinar si satisface los requisitos especificados y descubrir diferencias entre los resultados esperados y los observados”. (ISTQB, 2018, p. 45).

ISO/ IEC/ IEEE 29119 – 1: “La prueba de software es un proceso que comprende todas las actividades del ciclo de vida, tanto estáticas como dinámicas, relacionadas con las planificación, preparación y evaluación de productos de software, con el fin de verificar que cumplen con los requisitos”. (ISO/IEC/IEEE, 2013, p. 10).

En resumen, podemos identificar que cada definición proporcionada tiene relación entre sí y por lo cual podemos afirmar que la prueba de software es un proceso sistemático que conlleva la planificación, diseño, ejecución y evaluación de actividades estas con el fin de verificar que el sistema cumpla con los requisitos, detectar defectos en distintas fases con el fin de validar la utilidad para los usuarios finales y generar confianza en la calidad del software.

Objetivos de las Pruebas de Software.

Recordemos que las pruebas de software no se limitan únicamente a la detección de fallos, si no que estas abarcan un conjunto más amplio de finalidades orientadas a garantizar la calidad del producto y la satisfacción de los interesados. Entre los objetivos típicos y alineados al ISO/IEC/IEEE 29119, ISQTB y Pressman podemos mencionar:

- Evaluación de productos de trabajo como requisitos, historias de usuario, diseños y código.
- Detectar defectos y desencadenar fallos, permitiendo que los errores sean encontrados en etapas antes de producción.
- Asegurar la cobertura de las pruebas necesarias, permitiendo validar los flujos críticos para el negocio y la operación.
- Verificar el cumplimiento de los requisitos tanto funcionales como no funcionales.
- Confirmar conformidad con requisitos legales, contractuales y reglamentarios.
- Reducir el nivel de riesgo de calidad inadecuada del software.
- Proporcionar información objetiva a las partes interesadas.
- Generar confianza en la calidad del producto.
- Validar la completitud y funcionamiento del sistema.

Beneficios de las Pruebas de Software.

Las pruebas de software son una actividad importante durante el ciclo de vida del desarrollo de un software, por lo cual es necesario realizar pruebas exhaustivas que garanticen que los sistemas de software funcionen correctamente. Entre los beneficios más relevantes de la implementación de pruebas de software tenemos:

- Reducción de costos y tiempos asociados a correcciones, ya que si se logra detectar un error en etapas tempranas de desarrollo resulta significativamente menos costoso al contrario de cuando es identificado en producción.
- Disminución de riesgos, identificar defectos en etapas tempranas reduce riesgos del proyecto que pueden impactar en la calidad, la seguridad y el rendimiento.
- Mejora integral de la calidad de producto, ya que el sistema debe reflejar confiabilidad, usabilidad, eficiencia y seguridad.
- Optimización, las pruebas proporcionan información vital que puede utilizarse para mejorar continuamente la calidad del software, la experiencia del usuario, la seguridad, el rendimiento y otros atributos del producto.
- Satisfacción del usuario, las pruebas rigurosas desde la perspectiva del usuario permiten verificar la usabilidad, la funcionalidad y la compatibilidad.

1.3 Proceso General de Pruebas de Software

1.3.1 Niveles de Pruebas

El proceso de pruebas de software se organiza en distintos niveles de prueba, que representan etapas secuenciales y complementarias en la validación del sistema. Cada nivel tiene un objetivo particular y se enfoca en diferentes aspectos del producto, permitiendo detectar errores de forma progresiva y asegurar que el software cumpla con los requisitos establecidos.

1.3.2 Enfoque de Ejecución de Pruebas

Cuando nos referimos al enfoque de pruebas nos referimos a la estrategia y metodología adoptada para llevar a cabo la verificación y validación del software. La importancia radica en que estable el cómo, cuándo y con qué recursos se realizarán las pruebas, permitiendo garantizar la consistencia en los resultados y una cobertura adecuada de los requisitos del sistema.

En un plan de pruebas, la ejecución no se limita a la simple ejecución de los casos, si no también incluye con las actividades como la preparación del entorno de pruebas, la carga de datos de pruebas, la automatización de scripts cuando es necesario, y la documentación de los resultados obtenidos para su posterior análisis.

El tipo de enfoque a seleccionar e implementar dependerá de los objetivos del proyecto y del modelo de desarrollo.

El enfoque de ejecución de pruebas puede clasificarse según el momento de ciclo de vida, según la naturaleza de ejecución, según la estrategia de cobertura y según la automatización del ciclo de vida.

Enfoque de pruebas según el momento de ciclo de vida.

- Enfoque secuencial o tradicional (Waterfall / V-Model): es una metodología secuencial para la gestión de proyectos que se divide en fases. Cada fase comienza cuando ha terminado la anterior. Este enfoque para la gestión de proyectos surgió a partir de los sectores de fabricación y construcción en los que cada hito debe ser finalizado para continuar con el proceso de producción. Se le llama Waterfall o Cascada porque cada tarea cae en cascada sobre el paso siguiente.
- Enfoque iterativo o incremental: es una metodología en donde el proyecto se planifica en diversos bloques temporales llamados iteraciones. Las iteraciones se pueden entender como proyectos pequeños en todas las iteraciones se repiten un proceso de trabajo similar para proporcionar un resultado completo sobre producto final, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental.
  En cada iteración el equipo evoluciona el producto a partir de los resultados completados en las iteraciones anteriores, añadiendo nuevos objetivos o requisitos o mejorando de los que ya fueron completados.
- Enfoque ágil o continuo: la metodología agile es una metodología iterativa, es decir, se realizan entregas iterativas y en cada entrega se realizan todas las fases del ciclo: toma de requerimientos, diseño, verificación y entrega. La mayor diferencia de las metodologías agiles es que en todos los procesos se entrega valor y se recibe retroalimentación constantemente durante todo el proyecto.

En 2001 un grupo de 17 desarrolladores de software se reunieron en un retiro en Utah para debatir sobre los problemas de gestión de proyectos a los que se enfrentaban e intentar soluciones a los mismos. Este grupo de desarrolladores detectó que uno de los principales problemas a los que se enfrentaban la industria eran las entregas prolongadas, poco ágiles y flexibles del software.

1.4 Niveles de prueba

1.4.1 Pruebas Unitarias

Las pruebas unitarias son el primer nivel de prueba, centrado en verificar el funcionamiento correcto de los componentes individuales del sistema, tales como funciones, clases o módulos.

- **Objetivo:** Detectar errores de programación a nivel local.
- **Ejemplo:** Verificar que el formulario de creación de usuario valide correctamente los campos obligatorios (nombre, email, rol).

1.4.2 Pruebas de Integración

En este nivel se validan las interacciones entre los distintos módulos o componentes del sistema.

- **Objetivo:** Identificar errores en la comunicación y flujo de datos entre módulos.
- **Ejemplo:** Probar que, al registrar un nuevo ciclo académico, este quede disponible en el módulo de cargas académicas y en el de gestión de contratos

1.4.3 Pruebas de Sistema

Este nivel evalúa el comportamiento del sistema completo en relación con los requerimientos funcionales y no funcionales.

- **Objetivo:** Asegurar que el software implementa todas las funcionalidades esperadas y que cumple con criterios de rendimiento, seguridad y usabilidad.
- **Ejemplo:** Verificar que el flujo completo de contratación (registro de candidato, validación de documentos, generación de contrato, aprobación por decano) funcione sin interrupciones.

1.4.4. Pruebas de Aceptación

Son el último nivel de pruebas y están orientadas al usuario final o cliente. Permiten validar si el sistema realmente satisface las necesidades para las cuales fue desarrollado.

- **Objetivo:** Determinar si el software es apto para su despliegue en producción.
- **Ejemplo:** Un candidato debe poder ingresar su información personal, adjuntar documentos y verificar que recursos humanos valide y genere el contrato correspondiente.

1.5 Tipos de Pruebas

El proceso de pruebas de software comprende distintos enfoques que permiten evaluar el sistema desde múltiples perspectivas, asegurando no solo su correcto funcionamiento, sino también su confiabilidad y capacidad de adaptación en escenarios reales. La selección de los tipos de pruebas a aplicar depende de la naturaleza del proyecto, de los riesgos identificados y de los requisitos funcionales y no funcionales definidos para el sistema.

De acuerdo con la norma ISO/IEC 29119 y las mejores prácticas propuestas por el ISTQB, los tipos de pruebas pueden clasificarse en las siguientes categorías:

1.5.1 Pruebas Funcionales

Evalúan si el software cumple con las funciones especificadas en los requerimientos. En el caso del Sistema de Gestión de Contratos Docentes de la FIA-UES, estas pruebas comprenden procesos críticos como:

- Login con credenciales correctas e incorrectas.
- Registro y validación de usuarios.
- Generación y edición de contratos docentes.
- Validación de formularios con campos obligatorios y restricciones. Estas pruebas se implementan principalmente con técnicas de caja negra y partición equivalente, como se refleja en la matriz de pruebas.

1.5.2 Pruebas No Funcionales

Se orientan a verificar características de calidad que no están ligadas directamente a una funcionalidad específica, pero sí impactan en la experiencia de uso y confiabilidad del sistema. Entre ellas destacan:

- **Pruebas de seguridad:** validan que solo usuarios autorizados accedan a determinados módulos, como el menú de ciclos o la creación de contratos.
- **Pruebas de rendimiento:** orientadas a medir tiempos de respuesta y estabilidad bajo escenarios de carga (aunque en este proyecto se limitan por infraestructura local).
- **Pruebas de usabilidad:** aseguran que las interfaces sean claras, intuitivas y accesibles para el personal administrativo.

1.5.3 Pruebas Automatizadas y Manuales

Las pruebas manuales se aplican en escenarios exploratorios, de usabilidad o cuando la validación requiere criterio humano.

Las pruebas automatizadas, registradas en la matriz, se proponen para flujos repetitivos y críticos como el Login, la validación de roles y la generación de contratos, ya que su repetibilidad y frecuencia justifican la automatización.

Pruebas Manuales

Consisten en la ejecución de casos de prueba de manera directa por parte de los evaluadores o usuarios, sin la intervención de herramientas automatizadas. Este tipo de pruebas permite identificar errores funcionales, problemas de usabilidad y fallos en el comportamiento del sistema desde la perspectiva del usuario final.

1. **Características de las Pruebas Manuales**
   - **Interacción directa:** El evaluador sigue un conjunto de pasos definidos en los casos de prueba, reproduciendo escenarios de uso real.
   - **Flexibilidad:** Permiten la exploración de comportamientos inesperados que las pruebas automatizadas podrían no detectar.
   - **Orientación al usuario:** Se centran en validar que el sistema cumple con los requisitos funcionales y satisface las necesidades del usuario final.
   - **Mayor esfuerzo humano:** Requieren tiempo, dedicación y experiencia por parte de los evaluadores.

2. **Ventajas y Limitaciones**

   Ventajas:

   - Adecuadas para probar funcionalidades nuevas o cambios pequeños en el sistema.
   - Permiten evaluar aspectos subjetivos como la usabilidad y la experiencia del usuario.
   - No requieren de herramientas especializadas ni conocimientos técnicos avanzados.

   Limitaciones:

   - Son más lentas y costosas que las pruebas automatizadas.
   - Existe un mayor riesgo de errores humanos durante la ejecución.
   - Resultan poco escalables para pruebas de rendimiento o de regresión en sistemas grandes.

3. **Tipos de Pruebas Manuales**
   - **Pruebas funcionales:** Validan que las funciones del sistema operen conforme a los requerimientos.
   - **Pruebas de regresión:** Se ejecutan manualmente tras una actualización para comprobar que no se introdujeron errores en funcionalidades previas.
   - **Pruebas exploratorias:** No siguen un guion estricto, sino que el evaluador explora libremente el sistema buscando comportamientos inesperados.
   - **Pruebas de aceptación del usuario (UAT):** Permiten que el usuario final valide que el sistema cumple con sus necesidades reales.

Pruebas Automatizadas

Las pruebas automatizadas son aquellas que se ejecutan mediante herramientas y scripts de software, sin requerir la intervención constante de un tester. Su propósito principal es aumentar la eficiencia, repetibilidad y precisión del proceso de validación, reduciendo errores humanos y tiempos de ejecución.

En el marco de este proyecto, las pruebas automatizadas resultan fundamentales debido a que varios procesos del Sistema de Gestión de Contratos Docentes de la FIA-UES son altamente repetitivos y críticos para la operatividad institucional. La automatización permite validar estos flujos de manera continua, garantizando la estabilidad del sistema y facilitando su mantenimiento a futuro.

Beneficios de la automatización en este proyecto

- **Rapidez y eficiencia:** procesos que manualmente tomarían horas (como validar múltiples roles de usuario o generar contratos en diferentes ciclos) se ejecutan en minutos de forma automática.
- **Reducción de errores humanos:** los scripts siguen siempre la misma lógica, eliminando la variabilidad de la ejecución manual.
- **Reutilización:** los scripts pueden aplicarse en diferentes versiones del sistema, asegurando consistencia en las validaciones.
- **Escalabilidad**: a medida que se agreguen nuevas funcionalidades, los casos automatizados pueden extenderse con facilidad.

Trazabilidad: la ejecución automatizada genera evidencias claras y comparables, facilitando la documentación y auditoría de resultados.

1.5.4 Comparativa de Pruebas Manuales y Automatizadas

Las pruebas de software implican diversas metodologías para garantizar la calidad y la fiabilidad. Estas abarcan enfoques manuales y automatizados, cada uno con ventajas y desventajas específicas para la detección de defectos y la verificación de la funcionalidad.

En el ámbito de las pruebas de calidad de software ambos tipos de pruebas son importantes de realizar ya que los dos enfoques son complementarios para la validación de aplicaciones. Ambas buscan garantizar que el sistema cumpla con los requisitos funcionales y no funcionales, pero difieren en su metodología, alcance, costo y velocidad de ejecución.

Mientras que las pruebas manuales se apoyan de la intervención humana para diseñar, ejecutar y evaluar los casos de pruebas, las automatizadas se apoyan de herramientas y scripts que permiten repetir pruebas de forma rápida y sistema.

La elección de una o de la otra o la combinación de ambas pruebas depende de factores como la complejidad del proyecto, los recursos que se encuentran disponibles y los objetivos de calidad a perseguir. La siguiente tabla muestra una comparativa entre ambos enfoques, valorando sus atributos y características principales.

Aspecto	Pruebas Manuales	Pruebas Automatizadas
Proceso de ejecución	Los testers ejecutan casos de prueba paso a paso sin herramientas de automatización. Requiere intervención humana constante.	Los casos de prueba se ejecutan mediante scripts y herramientas de automatización con mínima intervención humana.
Velocidad y eficiencia	Más lentas, especialmente en pruebas repetitivas o de gran volumen. La eficiencia depende de la experiencia del tester.	Muy rápidas para tareas repetitivas y de regresión. Permiten ejecutar grandes volúmenes de pruebas en menor tiempo.
Reutilización	Los casos de prueba suelen ser poco reutilizables; deben reescribirse o adaptarse en cada ciclo.	Los scripts son fácilmente reutilizables en múltiples ciclos y entornos de prueba, favoreciendo la consistencia.
Cobertura	Limitada, ya que el tiempo y esfuerzo humano restringen el alcance. Útiles para escenarios exploratorios.	Amplia cobertura, los scripts pueden ejecutarse en diferentes plataformas, navegadores, dispositivos, y bajo múltiples configuraciones.
Exploración e intuición	Ideal para pruebas exploratorias, de usabilidad y validaciones visuales, donde la intuición y criterio humano son esenciales.	Poco efectivas en pruebas exploratorias, ya que carecen de intuición humana; se centran en validaciones predefinidas.
Mantenibilidad	Casos de prueba pueden ser fáciles de actualizar, pero la documentación puede ser demasiado extensa y dispersa	Los scripts requieren mantenimiento continuo si cambian las interfaces o funcionalidades del sistema.
Confiabilidad	Susceptibles a errores humanos durante la ejecución o registro de resultados.	Altamente confiables y consistentes en la ejecución siempre que los scripts estén bien diseñados.

Tabla 2 Tabla Comparación Pruebas Manuales y Pruebas Automatizadas Parte Uno

Aspecto	Pruebas Manuales	Pruebas Automatizadas
Escalabilidad	Poco escalables, el aumento de pruebas implica más testers y más tiempo.	Altamente escalables, se pueden ejecutar cientos de pruebas en paralelo en diferentes entornos.
Tipos de pruebas comunes	Exploratorias, pruebas de aceptación del usuario (UAT), validaciones visuales, pruebas ad hoc.	Regresión, carga, rendimiento, seguridad, pruebas en múltiples dispositivos/navegadores.
Aplicación ideal	Útiles en proyecto pequeños, en fases tempranas o cuando se necesita criterio humano directo.	Indispensables en proyectos grandes, de larga duración, con múltiples
Inversión Inicial	Baja inversión en herramientas, pero mayor costo operativo a largo plazo	Alta inversión inicial en herramientas, configuración y capacitación, pero menor costo operativo a largo plazo.

Tabla 3 Tabla Comparación Pruebas Manuales y Pruebas Automatizadas Parte Dos

1.6 Testing Outline: Pruebas Estáticas y Dinámicas

De acuerdo con la norma ISO/IEC/IEEE 29119-1, el proceso de pruebas de software comprende todas las actividades del ciclo de vida, tanto estáticas como dinámicas, orientadas a la planificación, preparación y evaluación de productos de software para verificar que cumplen con los requisitos establecidos. En el contexto del Sistema Informático para la Gestión de Contratos de Personal Docente de la FIA-UES, se ha definido un esquema de pruebas que combina ambos enfoques para maximizar la detección temprana de defectos y aumentar la confiabilidad del sistema.

1- Pruebas estáticas

Las pruebas estáticas son aquellas que se realizan sin ejecutar el software, enfocándose en la revisión y análisis de los productos de trabajo generados durante el ciclo de vida (requerimientos, historias de usuario, diseños, código, casos de prueba, etc.). En este proyecto, las pruebas estáticas se aplican principalmente a través de:

- **Revisión de requerimientos y procesos críticos:** se analizan los requerimientos funcionales y no funcionales, así como los procesos críticos identificados para la gestión de contratos, con el fin de detectar ambigüedades, inconsistencias o falta de información antes de diseñar los casos de prueba.
- **Revisión de la matriz de pruebas:** se verifica que los casos de prueba cubran los módulos priorizados (registro de candidatos, validación de documentos, generación de contratos, asignación de carga académica) y que exista trazabilidad entre requerimientos, casos y resultados esperados.
- **Revisión de diseños técnicos y flujos de proceso:** a partir de los diagramas de arquitectura y flujos de procesos documentados en el diagnóstico del sistema, se validan las entradas, salidas y reglas de negocio antes de ejecutar las pruebas dinámicas, reduciendo el riesgo de omitir escenarios importantes.

El objetivo de estas actividades estáticas es prevenir defectos antes de la ejecución, mejorar la calidad de la documentación de prueba y asegurar que el esfuerzo de pruebas dinámicas se enfoque en los flujos realmente relevantes para el negocio.

2- Pruebas dinámicas

Las pruebas dinámicas son aquellas en las que se ejecuta el sistema o sus componentes, observando el comportamiento frente a datos de entrada para comparar los resultados obtenidos con los resultados esperados. En este proyecto, las pruebas dinámicas se organizan de acuerdo con los niveles y tipos de prueba definidos en el plan de pruebas: pruebas funcionales, de integración y de sistema, tanto manuales como automatizadas.

Las principales actividades de pruebas dinámicas incluyen:

- Ejecución de pruebas funcionales manuales sobre los módulos críticos, utilizando la matriz de pruebas como guía para validar que cada caso de uso se comporta conforme a los requerimientos funcionales definidos.
- Ejecución de pruebas automatizadas en los flujos repetitivos y de alta frecuencia (por ejemplo, inicio de sesión, gestión de usuarios, generación de contratos y validación de ciclos), mediante scripts desarrollados con herramientas open source seleccionadas en este proyecto. Esto permite reducir tiempos de ejecución, aumentar la cobertura de regresión y disminuir errores humanos.
- Pruebas no funcionales seleccionadas (como rendimiento básico y validaciones de seguridad a nivel de roles y permisos), en la medida que lo permite la infraestructura disponible, con el objetivo de verificar que el sistema se comporte de forma estable en condiciones operativas representativas.

Las pruebas dinámicas se apoyan en los criterios de entrada y salida definidos en el Plan de Pruebas, así como en las métricas de prueba establecidas para evaluar la calidad del sistema y la efectividad del proceso de prueba.

Relación entre pruebas estáticas y dinámicas en el proyecto

El esquema de pruebas propuesto integra pruebas estáticas y dinámicas de forma complementaria:

- Las pruebas estáticas permiten depurar y mejorar la calidad de los requerimientos, procesos críticos y casos de prueba antes de la ejecución.
- Las pruebas dinámicas validan, mediante la ejecución del sistema, que los flujos de negocio priorizados funcionan correctamente y cumplen con los requerimientos funcionales y no funcionales definidos.

De esta manera, el Testing outline definido para el Sistema de Gestión de Contratos Docentes asegura un enfoque estructurado y alineado con la norma ISO/IEC/IEEE 29119, contribuyendo a la reducción de riesgos, la trazabilidad de resultados y la mejora continua de la calidad del software institucional.

1.7 Norma ISO/IEC 29119 Aplicada al Sistema de Gestión de Contratos de Docentes

1.7.1 Justificación de la Norma Seleccionada

La norma ISO/IEC 29119 es un estándar internacional desarrollado por la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica Internacional (IEC). Su objetivo principal es establecer un marco coherente y sistemático para la planificación, ejecución y documentación de pruebas de software, aplicable a cualquier tipo de proyecto o entorno tecnológico. Fue seleccionada para este proyecto por las siguientes razones:

- Se adapta tanto a sistemas grandes como a implementaciones locales, como el sistema de la FIA.
- Establece buenas prácticas en la planificación, diseño, ejecución y documentación de pruebas.
- Promueve la mejora continua de la calidad del software mediante procesos definidos y repetibles.
- Ofrece una estructura modular que facilita la aplicación parcial de sus componentes según el alcance del sistema y los recursos disponibles.

Al adoptar esta norma, el proyecto garantiza que las pruebas no sean vistas como una fase aislada, sino como una actividad transversal e integrada al ciclo de vida del software, alineada con los objetivos institucionales y con una visión de calidad sostenible.

1.7.2 Componentes Relevantes Aplicados

La ISO/IEC 29119 está compuesta por cinco partes. Las siguientes se consideran clave para este proyecto:

Parte 1: Conceptos y definiciones

Define términos y conceptos fundamentales relacionados con el proceso de pruebas, tales como "caso de prueba", "plan de pruebas", "criterio de aceptación", entre otros. Esto garantiza que todos los involucrados en el proyecto compartan un lenguaje común y una comprensión clara de las actividades que se ejecutarán.

Parte 2: Proceso de pruebas

Proporciona una guía detallada para gestionar el ciclo de vida de las pruebas, que incluye: planificación de pruebas, monitoreo y control, análisis, diseño, implementación, ejecución, evaluación de criterios de salida y cierre de pruebas. Esta parte asegura que cada fase esté debidamente documentada, controlada y alineada con los objetivos de calidad del proyecto.

Parte 3: Documentación de pruebas

Establece un conjunto estándar de documentos que deben producirse a lo largo del proceso de pruebas, tales como el plan maestro de pruebas, especificaciones de diseño de pruebas, especificaciones de casos de prueba y reportes de resultados. Esto permite garantizar la trazabilidad, facilitar auditorías y ofrecer evidencia objetiva del cumplimiento de los requisitos.

Describe diversas técnicas de diseño de pruebas agrupadas en tres categorías: pruebas basadas en la caja negra (orientadas al comportamiento externo del sistema), caja blanca (orientadas a la lógica interna del código) y pruebas basadas en la experiencia (como revisiones exploratorias o pruebas basadas en errores comunes). Esta diversidad metodológica permite elegir las técnicas más adecuadas para cada tipo de módulo, funcionalidad o restricción del entorno local.

Parte 5: Pruebas de conformidad

Esta sección proporciona directrices para verificar que un sistema cumple con los estándares, regulaciones y requisitos contractuales aplicables. En el contexto del sistema de gestión de contratos docentes, esta parte se considera útil como referencia para futuras auditorías internas o externas que puedan requerir demostrar que el software cumple con criterios institucionales o normativos.

1.7.3 Aplicación de la Norma al Proyecto

La implementación de pruebas en el sistema de gestión de contratos docentes se desarrolla considerando los lineamientos de la norma, con énfasis en:

- Diseño estructurado de casos de prueba: Se están elaborando casos de prueba detallados que especifican entradas, acciones esperadas, resultados esperados y criterios de éxito o fallo. Esto garantiza consistencia en las validaciones y facilita la futura automatización de pruebas.
- Priorización de pruebas en módulos críticos: Se ha identificado que módulos como la gestión de contratos, ciclos académicos, escalafones y usuarios representan procesos clave para el funcionamiento institucional, por lo que las pruebas se concentran inicialmente en estos componentes.
- Uso de documentación formalizada: Se está generando un conjunto de documentos que respaldan el proceso de pruebas, como el plan de pruebas funcionales, reportes de ejecución y evidencia visual de los resultados, siguiendo las plantillas sugeridas por la norma.
- Automatización parcial de pruebas: Dada la frecuencia y criticidad de algunas funciones, se ha optado por automatizar pruebas funcionales mediante herramientas como PHPUnit (para pruebas en Laravel) y Postman (para validaciones de APIs), lo que mejora la repetibilidad, reduce el esfuerzo manual y mejora los tiempos de respuesta en validación.

1.7.4 Beneficios Esperados

La aplicación de la norma ISO/IEC 29119 en este proyecto permite:

1. Incrementar la confiabilidad y mantenibilidad del sistema, ya que los errores se detectan y corrigen de manera estructurada.
2. Establecer una cultura de calidad dentro del ciclo de pruebas, lo cual es replicable en futuros sistemas o actualizaciones institucionales.
3. Proporcionar evidencia clara de que el sistema cumple con los requerimientos funcionales y responde a las necesidades del usuario.

4. Facilitar futuras auditorías técnicas o revisiones del sistema por parte de terceros, gracias a la documentación y trazabilidad generada.

CAPÍTULO II. Plan de Pruebas del Sistema de Gestión de Contratos de Personal Docente de la FIA-UES

2.1 Definición del Plan de Pruebas

El plan de pruebas del Sistema de Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador es el documento que establece, de manera formal y estructurada, el enfoque, el alcance y las actividades necesarias para verificar la calidad del sistema. Su propósito es definir cómo se evaluará el correcto funcionamiento del software, asegurar el cumplimiento de los requisitos funcionales y no funcionales, y garantizar que los procesos asociados a la contratación docente operen de forma confiable y segura.

Este plan describe los elementos fundamentales del proceso de pruebas, incluyendo los tipos y niveles que serán aplicados, los riesgos identificados, los recursos necesarios, los ambientes de ejecución, los criterios de aprobación y fallo, así como los entregables que se generarán durante el ciclo de pruebas. Además, detalla las tareas que deberá ejecutar el equipo de pruebas y establece lineamientos para la gestión de incidencias, la suspensión temporal de actividades y la reanudación de pruebas cuando sea necesario.

En conjunto, el plan de pruebas sirve como guía para la planificación, organización, ejecución y seguimiento de las actividades de aseguramiento de calidad, proporcionando una visión clara y ordenada del proceso que permitirá validar la estabilidad, funcionalidad y desempeño del Sistema de Gestión de Contratos de Personal Docente.

2.2 Estándar IEEE-829 e ISO/IEC 29119 Aplicados al Proyecto

El plan de pruebas del Sistema de Gestión de Contratos de Personal Docente se apoya en las buenas prácticas establecidas por los estándares internacionales de pruebas de software. En particular, se consideran dos referencias principales: el estándar IEEE-829 y la norma ISO/IEC 29119.

El estándar IEEE-829, Standard for Software and System Test Documentation, define un conjunto de documentos y contenidos mínimos recomendados para la planificación, diseño, ejecución y reporte de pruebas. Entre ellos se encuentran el plan de pruebas, las especificaciones de diseño de prueba, los procedimientos de prueba, los informes de resultados y los reportes de incidentes. Tomar este estándar como referencia permite estructurar el presente plan de forma coherente, asegurando que incluya elementos esenciales como el identificador del plan, el alcance, los ítems de prueba, los criterios de entrada y salida, los riesgos, los recursos y las aprobaciones.

Por otra parte, la norma ISO/IEC 29119 se adopta como marco principal para el proceso de pruebas, tal como se describió en el Capítulo I. Esta norma proporciona lineamientos para la organización de las actividades de prueba, la gestión del ciclo de vida, la documentación y las técnicas a emplear. En este proyecto, la ISO/IEC 29119 orienta la definición del proceso y de las

actividades a realizar, mientras que IEEE-829 sirve como guía para la estructura y el contenido del conjunto de documentos que conforman la documentación de pruebas.

De esta manera, el plan de pruebas integra ambos estándares: utiliza la ISO/IEC 29119 para asegurar que el proceso de pruebas siga un enfoque sistemático y basado en riesgos, y emplea IEEE-829 para garantizar que la documentación generada sea completa, consistente y trazable a lo largo de todo el proyecto.

2.3 Identificador del Plan de Pruebas

Para garantizar la trazabilidad y el control de versiones de la documentación de pruebas, en el proyecto se definieron formalmente los planes que rigen la verificación y validación del Sistema Informático para la Gestión de Contratos de Personal Docente. Dado que se contemplan tanto pruebas manuales como automatizadas, se trabaja con un plan de pruebas que abarque pruebas de la API y la interfaz web, y enfocado en las suites desarrolladas con Postman, JMeter y Selenium.

El plan de pruebas cuenta con un identificador propio (nombre, código interno, versión y fecha de emisión) que permite referirse de manera unívoca al documento dentro del proyecto y facilita su seguimiento durante las revisiones y actualizaciones.

De forma consistente con este plan, los casos de prueba manuales y automatizados se documentan utilizando formatos estandarizados, en los que se registran, como mínimo, el identificador del caso, el módulo, el tipo de prueba, la versión del sistema, las precondiciones, los datos de entrada, los pasos de ejecución, el resultado esperado, el resultado obtenido y las observaciones relevantes. Estos formatos sirven de base para la elaboración de las evidencias de prueba, la gestión de defectos y la consolidación de resultados que se presenta en los apartados posteriores.

2.4 Referencias del Plan de Pruebas

El plan de pruebas se apoya en un conjunto de documentos y fuentes de información que sirven como base para definir el alcance, la estrategia y los casos de prueba del Sistema Informático para la Gestión de Contratos de Personal Docente. Estas referencias permiten asegurar que las pruebas estén alineadas con los requerimientos funcionales y no funcionales del sistema, con los procesos del negocio y con los estándares de calidad adoptados en el proyecto.

Entre las principales referencias consideradas para la elaboración del plan de pruebas se encuentran:

- **Documentación del sistema bajo prueba (SUT)**: se utilizó únicamente el manual de usuarios compartido por la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador para explorar y familiarizarse con el sistema de Contratación de Personal Docente.
- **Documento de diagnóstico y diseño del proyecto**: análisis de la situación actual, problemas identificados, objetivos de mejora, alcance del proyecto y matriz de requisitos priorizados para el sistema de Gestión de Contratos.

- **Norma ISO/IEC/IEEE 29119 de pruebas de software:** lineamientos utilizados para estructurar el proceso de pruebas, los artefactos documentales y los criterios de seguimiento y cierre de las actividades de prueba.
- **Políticas y lineamientos institucionales de contratación docente:** normativa interna y procedimientos administrativos de la Facultad de Ingeniería y Arquitectura que establecen las reglas de negocio que el sistema debe cumplir, investigación del proceso general que conlleva la contratación de candidatos en la FIA UES.
- **Documentación de herramientas de prueba:** guías y especificaciones técnicas de Postman, JMeter y Selenium, utilizadas para diseñar y ejecutar las pruebas funcionales de API, las pruebas de rendimiento y las pruebas automatizadas de interfaz de usuario.

Estas referencias conforman el marco de información sobre el cual se construye y mantiene el plan de pruebas, y constituyen la base para la trazabilidad entre requisitos, casos de prueba, resultados obtenidos y decisiones relacionadas con la calidad del sistema.

2.5 Elementos de las Pruebas en el Proyecto

Las actividades de prueba definidas para el Sistema Informático para la Gestión de Contratos de Personal Docente se estructuran a partir de un conjunto de elementos que permiten organizar y ejecutar el proceso de manera sistemática. Estos elementos incluyen el sistema bajo prueba, los módulos y funcionalidades priorizadas, los tipos y niveles de prueba seleccionados, los artefactos documentales que los respaldan, las herramientas de apoyo utilizadas y los roles involucrados en la ejecución de las pruebas.

En primer lugar, el sistema bajo prueba (SUT) está constituido por el Sistema de Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura, el cual integra módulos como autenticación y control de acceso, gestión de usuarios y docentes, registro y administración de contratos, actividades de docentes y bitácora de acciones. Estos módulos se consideran la base funcional sobre la que se diseñan y ejecutan las pruebas, priorizando aquellos procesos que tienen mayor impacto en la operación institucional y en la correcta formalización de los contratos docentes.

En segundo lugar, el plan contempla la aplicación de distintos tipos y niveles de prueba, entre los que se destacan las pruebas funcionales, las pruebas de integración y sistema, las pruebas de aceptación y, en el ámbito no funcional, las pruebas de rendimiento sobre endpoints clave. Estas pruebas se ejecutan tanto de forma manual como automatizada, siguiendo los lineamientos establecidos en el plan de pruebas y tomando como referencia las necesidades del negocio y los riesgos identificados en el diagnóstico del proyecto.

Un tercer elemento lo constituyen los artefactos de prueba, dentro de los cuales se encuentran el plan de pruebas, la matriz de casos de prueba, los formatos de registro de ejecución, los reportes de defectos y las evidencias recopiladas (capturas, logs y reportes generados por las herramientas). La matriz de pruebas permite vincular los casos de prueba con los módulos y requisitos a los que responden, mientras que los registros de ejecución y los reportes de defectos documentan de manera detallada los resultados obtenidos y las incidencias detectadas durante el proceso de prueba.

Finalmente, las pruebas se apoyan en un conjunto de herramientas de apoyo y en la participación de roles específicos. Entre las herramientas se incluyen Postman, para las pruebas funcionales de la API; JMeter, para las pruebas de rendimiento; y Selenium, para las pruebas automatizadas de interfaz de usuario. Estas herramientas se utilizan en entornos de prueba controlados, definidos dentro del proyecto. En cuanto a los roles, el equipo de trabajo asume las funciones de diseño de pruebas, ejecución, registro de resultados y reporte de defectos, en coordinación con la persona docente guía y con las partes interesadas del sistema.

En conjunto, estos elementos conforman la estructura operativa del proceso de pruebas en el proyecto, permitiendo que las actividades de verificación y validación se realicen de forma organizada, trazable y alineada con los objetivos de calidad planteados para el Sistema de Gestión de Contratos de Personal Docente.

2.6 Riesgos Asociados a las Pruebas

El proceso de pruebas del Sistema Informático para la Gestión de Contratos de Personal Docente está expuesto a diversos riesgos que pueden afectar el cumplimiento del alcance, el cronograma y los objetivos de calidad definidos en el plan. Identificar estos riesgos de manera explícita permite anticipar problemas potenciales, definir acciones de mitigación y establecer prioridades en la ejecución de las actividades de prueba. Los riesgos asociados a las pruebas se relacionan tanto con aspectos técnicos del sistema y de la infraestructura, como con factores organizacionales, de gestión y de disponibilidad de recursos.

Uno de los principales riesgos corresponde a la disponibilidad limitada de entornos y datos de prueba adecuados. Si el entorno de pruebas no refleja de forma razonable las condiciones de producción (configuración de servidores, versiones de software, volumen de datos, usuarios y permisos), los resultados obtenidos podrían no ser representativos del comportamiento real del sistema. De manera similar, el uso de datos de prueba insuficientes o poco realistas puede ocultar defectos que solo se manifiestan con cargas mayores o con combinaciones específicas de información. Este riesgo se mitiga mediante la definición anticipada del ambiente de pruebas, la carga controlada de datos representativos y la coordinación con las personas responsables de la infraestructura.

Otro riesgo relevante está relacionado con la disponibilidad de tiempo y de personal para la ejecución de las pruebas. Dado que el proyecto se desarrolla en paralelo con actividades académicas y otros compromisos, existe la posibilidad de que no se logren ejecutar todos los casos de prueba planificados o que ciertas pruebas como las de rendimiento o las de regresión automatizada se vean reducidas en alcance. Esto podría derivar en una menor cobertura de pruebas sobre módulos críticos, incrementando la probabilidad de que defectos importantes permanezcan sin detectar. Para mitigar este riesgo, el plan prioriza los casos de prueba asociados a funcionalidades esenciales del proceso de contratación docente y define una matriz que indica cuáles pruebas son obligatorias y cuáles pueden posponerse si existen limitaciones de tiempo.

También se identifican riesgos vinculados a cambios en los requisitos o en el sistema durante la etapa de pruebas. Ajustes de última hora en la lógica de negocio, en la estructura de la base de datos o en la interfaz de usuario pueden dejar obsoletos algunos casos de prueba,

afectar los scripts de automatización o invalidar evidencias ya generadas. Esto impacta directamente en la trazabilidad entre requerimientos, pruebas y resultados. Para reducir este riesgo, se plantea mantener una comunicación constante con las personas responsables del desarrollo, registrar los cambios relevantes y actualizar de forma controlada la matriz de pruebas y los scripts automatizados.

Finalmente, existe el riesgo de subestimar la criticidad de determinados defectos o de no registrar adecuadamente las incidencias detectadas. Una gestión deficiente de los defectos puede provocar que problemas de alta severidad no sean corregidos a tiempo o que se pierda información importante para el análisis posterior. Este riesgo se atiende mediante el uso de un formato estándar de reporte de defectos, la clasificación de las incidencias por severidad y prioridad, y el seguimiento sistemático de su estado en la herramienta de apoyo seleccionada.

En conjunto, el reconocimiento de estos riesgos asociados a las pruebas y la definición de acciones de mitigación permiten fortalecer el proceso de aseguramiento de calidad del sistema, contribuyendo a que los resultados obtenidos sean más confiables y útiles para la toma de decisiones sobre la puesta en operación del Sistema de Gestión de Contratos de Personal Docente.

2.7 Características que se Probarán

Las pruebas definidas para el Sistema Informático para la Gestión de Contratos de Personal Docente se centran en las características funcionales y no funcionales asociadas a los módulos críticos del sistema, identificados en el plan de pruebas como aquellos que concentran mayor riesgo para el proceso de contratación docente. Estos módulos fueron priorizados a partir del diagnóstico del proyecto, de los requisitos definidos y de la importancia que tienen en la operación diaria de la Facultad.

En el ámbito funcional, se probarán principalmente las siguientes características:

- **Autenticación y control de acceso (login)**

  Verificación del inicio y cierre de sesión, manejo de credenciales válidas e inválidas y restricción de funcionalidades según el rol del usuario. Este módulo es crítico porque controla el acceso al resto del sistema y porque forma parte de los casos seleccionados para automatización.

- **Registro de candidatos**

  Creación y actualización de registros de candidatos a docencia, validación de campos obligatorios, formatos de datos y mensajes de error. Se comprobará que los candidatos queden correctamente asociados a la información necesaria para los procesos posteriores de evaluación y contratación.

- **Validación de documentos**

  Registro y validación de los documentos requeridos para la contratación (títulos, diplomas, constancias, entre otros), así como la visualización de su estado. Se verificará que el sistema permita marcar documentos como completos o pendientes y que refleje de forma consistente la situación de cada candidato.

- **Generación de contratos de personal docente**

  Creación y edición de contratos, asociación con el docente correspondiente, el período académico y las unidades de aprendizaje asignadas. Se probarán reglas de negocio relacionadas con fechas, estados del contrato y coherencia de la información utilizada para la generación de los documentos contractuales.

- **Asignación de carga académica y validación de ciclos**

  Registro y actualización de la carga académica asignada a cada docente, comprobando que las horas y asignaturas se correspondan con el ciclo académico y con las restricciones definidas. Se validará además que los ciclos utilizados en los contratos y en la carga académica sean consistentes y estén debidamente habilitados en el sistema.

- **Gestión de actividades de docentes y bitácora de acciones**

  Registro y consulta de actividades de docentes, así como el registro de acciones relevantes en la bitácora del sistema. Se verificará que los listados reflejen información coherente, que los filtros funcionen correctamente y que la bitácora muestre los datos necesarios (usuario, acción, fecha y módulo afectado) solo para los perfiles autorizados.

En cuanto a las características no funcionales incluidas en el alcance del plan de pruebas, se evaluarán principalmente:

- **Rendimiento básico de endpoints y listados clave**

  Medición de tiempos de respuesta en operaciones frecuentes y críticas, como listados de candidatos, contratos, carga académica, actividades de docentes y registros de bitácora, bajo una carga moderada definida para el entorno de pruebas.

- **Comportamiento de seguridad relacionado con permisos**

  Verificación de que las operaciones protegidas solo estén disponibles para usuarios autenticados con los roles adecuados y que el sistema responda correctamente ante intentos de acceso no autorizado a módulos o acciones restringidas.
- **Consistencia de datos en flujos completos de negocio**

  Comprobación de que los flujos principales desde el registro del candidato y la validación de documentos hasta la generación del contrato y el registro en la bitácora mantengan la integridad y coherencia de los datos a lo largo de las diferentes vistas y módulos del sistema.

Estas características corresponden al conjunto de funcionalidades y comportamientos considerados prioritarios en el plan de pruebas y en la matriz de casos de prueba. Su verificación, mediante pruebas manuales y automatizadas, busca aportar evidencia suficiente sobre la capacidad del sistema para soportar de forma confiable el proceso de contratación docente de la Facultad de Ingeniería y Arquitectura.

2.8 Características que No se Probarán

Además de las funcionalidades y comportamientos incluidos en el alcance del plan de pruebas, existen características del sistema y aspectos de calidad que no serán cubiertos en este proyecto, ya sea por limitaciones de tiempo, de infraestructura o porque no forman parte de los objetivos definidos para la tesina. Dejar explícitas estas exclusiones permite delimitar el alcance real de las pruebas y evitar interpretaciones erróneas sobre los resultados obtenidos.

En cuanto a funcionalidades, no se probarán de forma exhaustiva las siguientes características:

- **Módulos o pantallas secundarias no priorizadas en el diagnóstico inicial,** tales como secciones de administración general, configuración avanzada del sistema o mantenimiento de catálogos que no intervienen de manera directa en el proceso de contratación docente.
- **Reportes complementarios o consultas especiales** que no están directamente relacionados con los flujos críticos definidos (registro de candidatos, validación de documentos, generación de contratos, asignación de carga académica, actividades de docentes y bitácora).
- **Integraciones externas con otros sistemas institucionales o con servicios de terceros** (por ejemplo, sistemas financieros o académicos externos), en caso de existir, ya que el foco del proyecto se centra en el comportamiento interno del Sistema de Gestión de Contratos.
- **Envío de notificaciones vía correo electrónico:** debido a que el servicio que utiliza para el envío de notificaciones por medio de correo electrónico es un recurso externo que no ha sido configurado en el entorno de pruebas.

Desde la perspectiva de calidad no funcional, quedan fuera del alcance de este plan las siguientes pruebas:

- **Pruebas de rendimiento de alto volumen y estrés extremo**, como pruebas de carga masiva, pruebas de picos de concurrencia elevados o estudios de escalabilidad a gran escala, debido a las limitaciones del entorno de pruebas disponible y a la naturaleza académica del proyecto.
- **Pruebas de seguridad avanzadas**, tales como pruebas de penetración (pentesting) especializadas, explotación de vulnerabilidades complejas o evaluaciones formales de cumplimiento normativo en materia de seguridad de la información. Solo se abordarán verificaciones básicas relacionadas con permisos y control de acceso y así como pruebas de humo.
- **Pruebas formales de usabilidad y accesibilidad, que involucren técnicas como estudios de campo**, pruebas con grupos de usuarios representativos, medición de tiempos de tarea o cumplimiento de estándares de accesibilidad web. En esta tesina la evaluación de la interfaz se limita a verificar que los flujos críticos puedan ejecutarse de manera correcta.
- **Pruebas de recuperación ante desastres y alta disponibilidad,** incluyendo escenarios de caída de servidores, conmutación por error (failover) o restauración de

copias de respaldo en ambientes productivos, ya que estos aspectos dependen de decisiones de infraestructura fuera del alcance del equipo de tesina.

El plan de pruebas se concentra en las funcionalidades y comportamientos considerados críticos para la gestión de contratos de personal docente y en un conjunto acotado de aspectos no funcionales. Las características aquí enumeradas no serán objeto de prueba detallada en este proyecto y, en caso de requerirse su evaluación, deberán abordarse en iniciativas posteriores o en planes de pruebas complementarios definidos por la institución.

2.9 Alcance y Estrategia General de Pruebas

El plan de pruebas del Sistema Informático para la Gestión de Contratos de Personal Docente (FIA-UES) abarca la verificación de todos los módulos funcionales, considerando lo siguiente:

1. **Módulos prioritarios para probar**
   - Registro de candidatos.
   - Validación de documentos.
   - Generación de contratos.
   - Asignación de carga académica.
   - Gestión de Usuarios (login, logout, creación de usuarios).
   - Catálogos que intervienen en el proceso de los flujos críticos.

   Estos son los más críticos porque concentran el mayor riesgo en caso de fallos.
2. **Documentación de casos de prueba**
   - Se elaborará una matriz de pruebas con entradas, salidas esperadas, precondiciones, resultados y evidencias.
   - Se asegura la trazabilidad entre requerimientos y pruebas.
3. **Automatización de pruebas seleccionadas**
   - Se automatizarán los casos de pruebas siguiendo, priorizándolos con la técnica de MoSCoW y adicional se deberán de considerar por cada caso de pruebas los siguientes atributos:
     - Frecuencia de ejecución.
     - Criticidad del proceso.
     - Estabilidad del proceso.
     - Resolución determinística.
     - Compatibilidad técnica.
4. **Ejecución y validación**
   - Se ejecutarán en un entorno controlado (no productivo), el cual ha sido desplegado en Railway.
   - Se generarán evidencias, métricas e informes de defectos.
5. **Análisis comparativo**
   - Se compararán los resultados entre pruebas manuales y automatizadas para evaluar tiempos, cobertura y esfuerzo.
6. **Entregables**
   - Plan de pruebas.
   - Matriz de pruebas.

- Evidencias de ejecución.
- Informe de resultados y recomendaciones.
- Manual técnico de pruebas automatizadas y pruebas manuales.

7. **Exclusiones (fuera del alcance)**
   - No se harán pruebas de carga o estrés avanzadas (por limitación de hardware).
   - No se harán modificaciones al código fuente.
   - Se realizarán pruebas de seguridad, pero no en nivel avanzado.

2.9.1 Enfoque Metodológico de Ejecución de Pruebas (Scrum)

Para la ejecución de las pruebas se desarrollará bajo el enfoque de metodología Ágil, adaptado a la etapa de pruebas de calidad, ya que el sistema se encuentra terminado. Las pruebas las organizaremos en ciclos cortos por módulos, priorizando funcionalidades críticas y aplicando la matriz de riesgos definida.

Con este enfoque aseguraremos la retroalimentación rápida y la mejora continua en la calidad del producto.

Para este proyecto se eligió la metodología Ágil Scrum porque permite trabajar de forma iterativa, organizada y flexible.

Razones de la elección

- **Entregas rápidas y continuas**: cada Sprint produce avances visibles (matriz de pruebas, scripts, reportes).
- **Flexibilidad ante cambios**: se pueden ajustar los casos de prueba según nuevas necesidades.
- **Colaboración constante**: promueve comunicación diaria y revisión conjunta de resultados como grupo de trabajo.
- **Priorización**: se enfocan primero los módulos más críticos (registro de candidatos, validación de documentos, contratos).

Scrum fue preferido frente a cascada (muy rígido y pruebas al final) y Kanban (útil para flujo visual, pero sin estructura de entregables periódicos).

Beneficios para el proyecto

- Retroalimentación temprana y mejora continua.
- Menor riesgo de fallos no detectados.
- Entrega de valor constante y documentada.
- Mayor compromiso del equipo y facilidad para escalar en el futuro.

En conclusión: Scrum se escogió porque combina flexibilidad, control y valor incremental, lo que asegura un proceso de pruebas más eficiente y de calidad.

2.10 Criterios de Aprobación y Fallo

2.10.1 Criterios de Entrada

En el contexto del proyecto de Implementación del Sistema Informático para la Gestión de Contratos de Personal Docente, los criterios de entrada son condiciones esenciales que deben cumplirse antes de iniciar las pruebas de calidad. Estos criterios aseguran que el entorno y los recursos estén listos para garantizar que las pruebas de integración del sistema se realicen correctamente, validando que el sistema cumpla con los requisitos definidos.

Los criterios específicos son los siguientes:

1. **Historias de Usuario Aprobadas y Validadas**: Las historias de usuario deben estar definidas con detalle, basadas en los requisitos funcionales y no funcionales. Estas historias deben reflejar de manera precisa las funcionalidades del sistema como la gestión de contratos docentes, los escalafones, la validación de datos, y la gestión de acceso.
2. **Requerimientos del Usuario Documentados**: Los requerimientos funcionales, como la creación, edición y validación de contratos docentes, y los requerimientos no funcionales, como el rendimiento y la seguridad del sistema, deben estar documentados, aprobados y accesibles para las pruebas de calidad.
3. **Ambiente de Pruebas Configurado y Validado**: En este proyecto, el entorno de pruebas se configuró utilizando un repositorio proporcionado en GitHub, desde donde se descargó el código fuente del sistema. El proceso de configuración y validación del ambiente de pruebas incluyó las siguientes actividades:
    - **Descarga y Configuración del Sistema**: El sistema fue descargado directamente desde el repositorio de GitHub proporcionado, asegurando que la versión más reciente y estable estuviera disponible para las pruebas. Se utilizó un entorno local de desarrollo para realizar la configuración inicial del sistema, replicando las condiciones del entorno de producción en la medida de lo posible.
    - **Configuración de la Base de Datos**: La base de datos necesaria para las pruebas fue configurada localmente, asegurando que todos los datos de prueba relevantes estuvieran disponibles y que el sistema pudiera acceder a ellos sin problemas. Se validó que las conexiones a la base de datos fueran correctas y que las consultas de la aplicación funcionaran según lo esperado.
    - **Validación de Acceso al Sistema:** Se verificó que el sistema fuera accesible a través del entorno local levantado, asegurando que los usuarios pudieran interactuar con las funcionalidades claves, como la gestión de contratos docentes y escalafones. Esta validación incluyó el acceso a las interfaces de usuario y la comprobación de la correcta ejecución de las funciones básicas.
    - **Herramientas de Pruebas Configuradas y Funcionando**: Las herramientas utilizadas para la ejecución de las pruebas fueron configuradas y validadas, incluyendo la integración con Azure DevOps para la gestión de defectos y la ejecución de pruebas automatizadas. A pesar de ser un entorno local, se garantizó que las pruebas de integración y del sistema pudieran ejecutarse de manera controlada y repetible.

4. **Casos de Prueba y Plan de Pruebas Revisados y Aprobados**: Todos los casos de prueba, diseñados con base en las historias de usuario y los requerimientos del sistema, deben ser documentados y revisados. El plan de pruebas debe estar alineado con los criterios de aceptación, y debe ser aprobado por el equipo de QA. Los casos de prueba deben incluir todas las funcionalidades del sistema, como la gestión de escalafones docentes, el control de duplicados, y la validación de datos de contratos docentes.
5. **Datos de Prueba Creados y Validados**: Los datos necesarios para realizar las pruebas deben estar creadas y validadas para asegurar que cubren todos los escenarios posibles, como la creación de un contrato, validación de escalafones docentes, y la gestión de accesos.

2.10.2 Criterios de Salida

El proceso de pruebas se considerará concluido cuando se cumplan las siguientes condiciones:

1. **Cobertura de pruebas alcanzada:**
    - Se deben ejecutar al menos el 90% de los casos de prueba definidos en la matriz de pruebas funcionales.
2. **Defectos gestionados:**
    - Todos los defectos de alta prioridad (que afecten procesos críticos: registro de candidatos, validación documental, generación de contratos, asignación de carga académica) deben estar corregidos y validados.
3. **Defectos críticos resueltos:**
    - Todos los defectos de alta prioridad (que afecten procesos críticos: registro de candidatos, validación documental, generación de contratos, asignación de carga académica) deben estar corregidos y validados.
4. **Documentación completada:**
    - Toda la documentación deberá estar finalizada:
        - ✓ Matriz de casos de prueba.
        - ✓ Scripts automatizados.
        - ✓ Evidencias de ejecución (capturas, logs, reportes).
        - ✓ Informe de resultados (casos exitosos, fallidos, pendientes).
        - ✓ Análisis comparativo entre pruebas manuales y automatizadas.
5. **Comparativo de eficiencia entregado:**
    - Se debe entregar el análisis comparativo entre pruebas manuales y automatizadas (tiempos, esfuerzo y cobertura).
6. **Métricas de calidad cumplidas:**
    - Al menos un 95% de los casos ejecutados deben ser exitosos.
    - Cero defectos críticos abiertos al cierre.
7. **Aprobación del equipo responsable:**
    - Los entregables deben ser revisados y aprobados por el equipo de QA y el asesor académico, validando que el sistema cumple con los requisitos funcionales definidos.

2.11 Criterios de Suspensión y Requisitos de Reanudación

Los criterios de suspensión y los requisitos de reanudación permiten establecer de antemano en qué situaciones debe detenerse temporalmente la ejecución de las pruebas y en qué condiciones es posible retomarlas de manera segura. Esto evita continuar con pruebas en un contexto inestable o poco confiable y ayuda a utilizar de forma más eficiente el tiempo y los recursos del proyecto.

2.11.1 Criterios de suspensión de las pruebas

La ejecución de las pruebas se podrá suspender total o parcialmente cuando se presente alguna de las siguientes situaciones:

- **Defectos críticos en módulos esenciales:** Se detecte uno o más defectos de severidad Crítica o Alta en módulos clave (autenticación, registro de candidatos, validación de documentos, generación de contratos, validación de ciclos, actividades de docentes o bitácora) que impidan continuar con la ejecución de otros casos de prueba dependientes de dichos módulos.

- **Inestabilidad del entorno de pruebas:** El entorno donde se ejecutan las pruebas (servidor, base de datos, aplicación web o servicios asociados) presente fallos recurrentes, caídas frecuentes, errores de configuración o tiempos de respuesta excesivos que no permitan obtener resultados confiables.

- **Inconsistencias graves en los datos de prueba:** Se identifiquen problemas importantes en la carga o calidad de los datos de prueba (registros incompletos, relaciones rotas, datos incoherentes) que afecten múltiples casos de prueba y que requieran un ajuste significativo del entorno o de la base de datos antes de continuar.

- **Fallas en las herramientas de prueba:** Las herramientas utilizadas (Postman, JMeter, Selenium o la plataforma de seguimiento como Azure DevOps) presenten errores que imposibiliten ejecutar los scripts, registrar resultados o documentar defectos de forma adecuada.

- **Cambios no planificados en el sistema bajo prueba:** Se apliquen modificaciones relevantes en el código, la estructura de la base de datos o la configuración del sistema durante el ciclo de pruebas, sin que se haya actualizado la matriz de casos de prueba o las suites automatizadas, generando resultados inconsistentes o no comparables.

Ante cualquiera de estos escenarios, el responsable de pruebas deberá documentar la situación, registrar los incidentes correspondientes y notificar al equipo técnico y a la persona docente guía para la toma de decisiones.

2.11.2 Requisitos de Reanudación de las Pruebas

La reanudación de las pruebas estará sujeta al cumplimiento de los siguientes requisitos:

- **Corrección o control de los defectos críticos:** Los defectos de severidad Crítica o Alta que motivaron la suspensión deberán estar corregidos y verificados mediante la ejecución de los casos de prueba asociados (incluyendo pruebas de regresión cuando sea necesario), o bien controlados mediante decisiones explícitas documentadas en el plan de trabajo.

- **Restablecimiento del entorno de pruebas:** El entorno deberá encontrarse nuevamente estable y operativo, con la aplicación desplegada en una versión definida, servicios en funcionamiento y tiempos de respuesta aceptables para ejecutar los casos de prueba planificados.

- **Normalización de los datos de prueba:** Los datos de prueba deberán haber sido revisados y ajustados para garantizar su coherencia, completitud y adecuación a los escenarios definidos en la matriz de casos de prueba, evitando que las inconsistencias anteriores vuelvan a afectar la ejecución.

- **Disponibilidad de las herramientas de apoyo:** Las herramientas de prueba y de gestión (Postman, JMeter, Selenium, Azure DevOps) deberán estar nuevamente disponibles y funcionando correctamente, permitiendo la ejecución de scripts, la recolección de métricas y el registro de resultados y defectos.

- **Actualización de artefactos de prueba afectados:** En caso de que la suspensión haya estado relacionada con cambios en el sistema, deberá haberse actualizado la matriz de casos de prueba, los scripts automatizados y, en su caso, la planificación del ciclo de pruebas, dejando constancia de los ajustes realizados.

Una vez verificado el cumplimiento de estos requisitos, el responsable de pruebas podrá autorizar la reanudación del ciclo de pruebas, priorizando primero la ejecución de los casos críticos y de regresión, para confirmar la estabilidad del sistema antes de continuar con el resto de los casos planificados.

2.12 Ambiente de Pruebas

2.12.1 Aspectos Técnicos del Sistema

El Sistema de Gestión de Contratos de Personal Docente de la FIA-UES está desarrollado bajo una arquitectura cliente-servidor moderna, que separa la lógica de negocio en el backend (API en Laravel) y la interfaz gráfica en el frontend (React). Este diseño modular garantiza mayor escalabilidad, portabilidad y facilidad de mantenimiento.

1. **Backend**

   El backend se implementó con Laravel 10 sobre PHP 8.x, lo que permite organizar la lógica de negocio de forma estructurada y aprovechar funcionalidades avanzadas como controladores, middleware y servicios REST. Se expone una API que gestiona operaciones CRUD relacionadas con los contratos, la autenticación y otros procesos administrativos.

- **Base de datos:** se utiliza PostgreSQL, asegurando integridad, transacciones seguras y manejo eficiente de grandes volúmenes de información.
- **ORM:** la comunicación con la base de datos se realiza a través de Eloquent, simplificando la persistencia de datos.
- **Seguridad:** la autenticación se gestiona mediante Laravel Sanctum y los roles/permisos a través de Spatie Roles & Permissions. Se incluye cifrado de contraseñas y bitácoras de auditoría para trazabilidad.
- **Configuración:** el sistema utiliza archivos. env para parametrizar conexiones y credenciales, lo que facilita la adaptación a distintos entornos.
- **Contenedores**: el despliegue se gestiona con Docker y Docker Composer, permitiendo levantar entornos de desarrollo y producción de manera aislada y replicable. En producción, el sistema se ejecuta sobre NGINX en el puerto 8000.

2. Frontend

La interfaz gráfica fue desarrollada con React (Node.js v16+, npm v8+) y TypeScript, brindando una experiencia de usuario dinámica y responsiva, adaptable a distintos navegadores y dispositivos.

- **Ejecución en desarrollo**: la aplicación corre en http://localhost:3000 y permite recarga automática de cambios.
- **Ejecución en producción:** se genera un build optimizado en la carpeta build, que se despliega en un servidor NGINX en el puerto 80.
- **Configuración:** el frontend utiliza un archivo. env para definir la URL base de la API (REACT_APP_BASE_URL) y el puerto de ejecución.
- **Contenedores:** al igual que el backend, el frontend se puede ejecutar en contenedores Docker tanto en entornos de desarrollo como de producción, lo que asegura portabilidad e independencia de la infraestructura local.

3. Herramientas de soporte

- **Composer**: para la gestión de dependencias en PHP.
- **npm**: para la instalación y actualización de paquetes de frontend.
- **Docker y Docker Compose:** para la gestión de entornos portables.
- **Azure DevOps y GitHub**: para control de versiones, despliegues y gestión de pruebas.

En conjunto, esta infraestructura técnica ofrece un sistema confiable, modular y seguro, capaz de responder a las necesidades administrativas de la Facultad de Ingeniería y Arquitectura, garantizando eficiencia en la gestión de los contratos docentes.

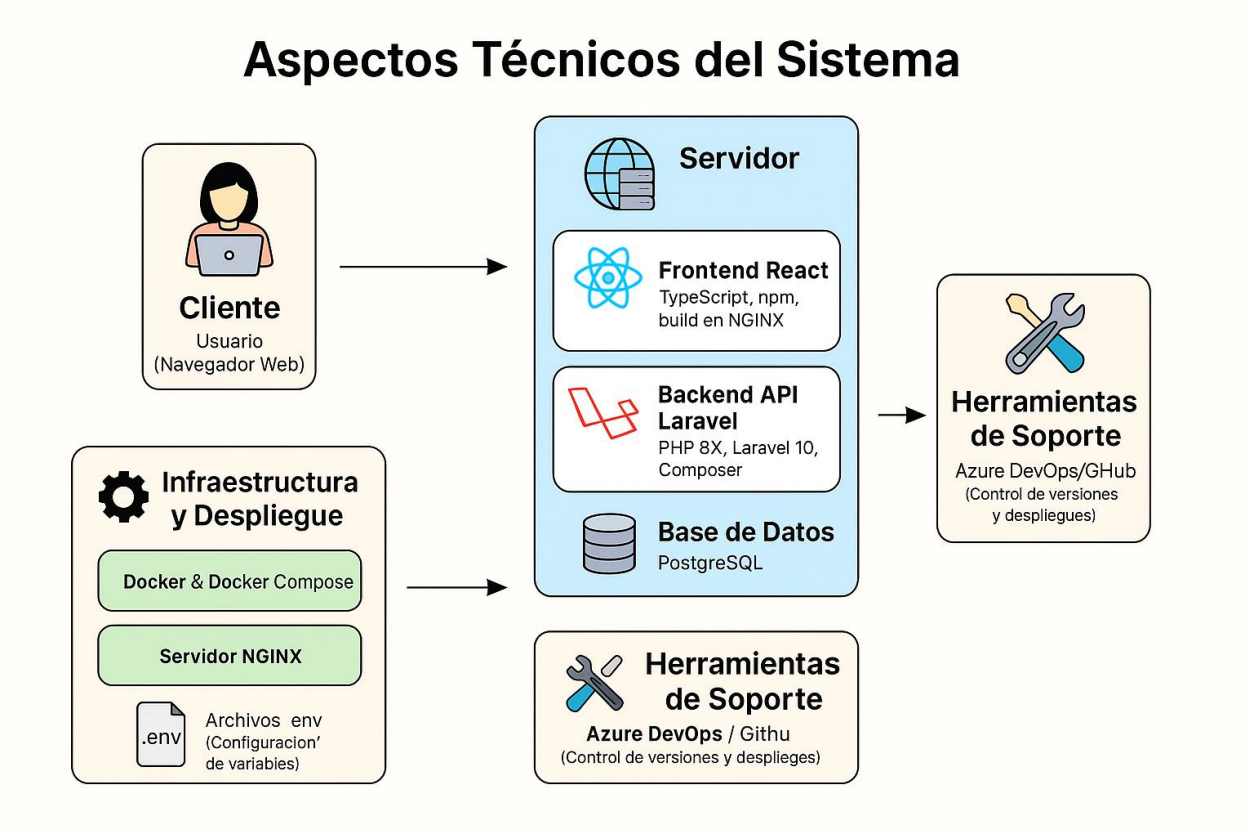


Ilustración 5 Aspectos Técnicos del Sistema.

2.12.2 Preparación de los Recursos para las Pruebas

La preparación de recursos es una etapa fundamental para garantizar que el proceso de pruebas del Sistema de Gestión de Contratos de Personal Docente de la FIA-UES se desarrolle de forma organizada, eficiente y con resultados confiables. Para ello, se contemplaron los siguientes recursos:

1- **Recursos Humanos**

- Equipo de pruebas conformado por estudiantes responsables del proyecto y supervisión de un asesor académico.
- Usuarios clave de la Facultad de Ingeniería y Arquitectura (personal de Recursos Humanos, Dirección de Escuela y Decanato) que aportan retroalimentación en la validación de funcionalidades.
- Roles definidos en el equipo de pruebas, tales como responsable de ejecución, encargado de documentar resultados y coordinador de incidencias.

2. **Recursos Técnicos**

- Equipos de cómputo con especificaciones adecuadas para ejecutar pruebas automatizadas y de carga.

- Ambientes de prueba locales configurados mediante Docker y Docker Compose, que permiten replicar la arquitectura del sistema sin afectar el entorno de producción.
- Conexión de red estable para la interacción con la base de datos y los servicios de la aplicación.

3**. Herramientas de Software**

- Selenium WebDriver, para pruebas funcionales y de interfaz.
- Postman, para pruebas de servicios REST expuestos por el backend.
- JMeter, para pruebas de rendimiento y carga controlada.
- OWASP ZAP, para validación de seguridad básica..

4. **Recursos de Documentación**

- Matriz de pruebas, que permite trazar los casos con los requerimientos del sistema.
- Casos de prueba detallados, incluyendo entradas, pasos a seguir, resultados esperados y resultados obtenidos.
- Formatos de reporte, diseñados para documentar hallazgos, incidencias y recomendaciones.

2.12.3 Recurso de Hardware

Se hará uso de tres equipos que cuenta con diferentes características, las cuales listamos a continuación:

Hardware	Equipo 1	Equipo 2	Equipo 3
Sistema Operativo	Windows 11	Windows 11 Home Single Language 64 bits	Windows 11 Home Single Language
Marca	HP	ACER	ASUS
Procesador	11th Gen Intel(R) Core (TM) i5-1135G7	Intel® Core™ i5-1035G1 CPU @ 1.00GHz (~1.2GHz, 4 núcleos)	12th Gen Intel(R) Core (TM) i7-12700H (2.30 GHz)
Memoria RAM	8.00 GB	8 GB	16.0 GB
Almacenamiento	500 GB	1.36 TB	1 TB
GPU	Intel(R) UHD Graphics	Intel UHD Graphics (integrada)	Intel(R) UHD Graphics

Tabla 4 Hardware a Utilizar en el Proyecto.

2.12.4 Recursos para las Pruebas

Para la realización de nuestro plan de pruebas listamos a continuación los recursos necesarios para su desarrollo:

- Historias de usuario, escenarios de pruebas y casos de pruebas diseñados.
- Plantilla seleccionada para reportar defectos.
- Sistema instalado de manera local.

- Servidor web local.
- Base de datos funcionando de manera correcta.

2.12.5 Recursos de Software

- El sistema se encuentra configurado lo más parecido al entorno productivo, bajo las siguientes características:
    - Versión PHP mayor a 8.0
    - Composer
    - PostgreSQL
    - Node JS v 16
    - NPM v 8
- Uso de Office (Excel, Word y OneDrive).
- Lector de PDF para la visualización de los documentos de pruebas (Documento de Identidad, Título, Currículum Vitae, entre otros).
- Azure DevOps: herramienta a utilizar para registrar y coordinar el backlog de nuestro proyecto.
- Power BI: se utiliza para llevar una actualización visible de los avances del proyecto y los hallazgos.
- Google Meet: se utilizará para la comunicación del equipo y para realizar documentación de pruebas en caso de que sea necesario.
- Docker: garantizar el entorno consistente y portable. Optimiza recursos y se integra con flujos CI/CD, mejorando la eficiencia y confiabilidad del servidor.
- Selenium WebDriver: permitirá simular la interacción real de un usuario con la aplicación. Facilitará la validación de funcionalidades clave y mejorará la cobertura de pruebas.
- OSWAP ZAP: herramienta open source, fácil de usar, actualizada y muy enfocada a vulnerabilidades web comunes.

2.12.5.1 Criterios de selección de herramientas

La selección de herramientas para la realización de pruebas en el Sistema de Gestión de Contratos de Personal Docente de la FIA-UES se basó en criterios como facilidad de uso, compatibilidad tecnológica, soporte comunitario, costo, escalabilidad y alineación con los objetivos del proyecto. Dado que se trata de un entorno académico con limitaciones presupuestarias, se priorizaron herramientas gratuitas y de código abierto (Open Source) que garantizan confiabilidad y sostenibilidad en el tiempo.

Criterios de evaluación.

Utilizamos diferentes criterios para la selección de las herramientas y con ello para cumplir con el desarrollo de las pruebas de calidad a ejecutar.

No	Criterio	Evaluación
1	Facilidad de uso	¿Es fácil su instalación y configuración?
2	Compatibilidad	¿Puede funcionar en cualquier navegador, plataforma o entorno?
3	Escalabilidad	¿La herramienta puede soportar un crecimiento en la cantidad de pruebas a ejecutar?
4	Soporte y Comunidad	¿Cuenta con documentación la herramienta? y ¿Cuenta con una comunidad activa?
5	Costo	¿La herramienta es gratis o de código abierto?
6	Mantenimiento y Actualización	¿La herramienta cuenta con actualizaciones periódicas?
7	Funcionalidades	¿La herramienta se ajusta a las necesidades del proyecto?

Tabla 5 Criterios de evaluación herramientas.

Puntuación de herramientas.

Para facilitar la selección de la herramienta a utilizar manejaremos una escala de puntuación, la cual detallamos a continuación:

Puntuación	Descripción
3	Excelente: la herramienta cumple con los criterios establecidos y se ajusta a los requisitos establecidos.
2	Buena: Cumple con el criterio de manera parcial, presenta áreas que pueden mejorarse.
1	Insuficiente: la herramienta no cumple con ninguno de los criterios establecidos para su selección.

Tabla 6 Puntuación de herramientas.

2.12.5.2 Evaluación comparativa de herramientas de prueba

Dados los parámetros anteriormente listados y clasificados, procedemos a realizar la evaluación de las herramientas propuestas a utilizar:

Evaluación para herramientas de pruebas funcionales.

Herramienta	Facilidad de uso	Compatibilidad	Escalabilidad	Soporte y Comunidad	Costo	Mantenimiento y Actualización	Funcionalidades	Puntuación
Selenium WebDriver	3	3	2	3	3	3	3	20
Plawright	2	3	2	3	3	1	3	17
Cypress	3	3	2	3	2	3	3	19

Tabla 7 Evaluación Herramientas para Pruebas Funcionales.

Evaluación para herramientas de pruebas de seguridad.

Herramienta	Facilidad de uso	Compatibilidad	Escalabilidad	Soporte y Comunidad	Costo	Mantenimiento y Actualización	Funcionalidades	Puntuación
OWASP ZAP	3	3	2	3	3	3	3	20
Burp Suite	2	3	3	3	2	3	3	19
Nikto	2	2	2	2	3	2	2	15
SonarQube	2	3	3	3	2	3	3	19

Tabla 8 Evaluación Herramientas para Pruebas de Seguridad.

Evaluación para herramientas de pruebas de API.

Herramienta	Facilidad de uso	Compatibilidad	Escalabilidad	Soporte y Comunidad	Costo	Mantenimiento y Actualización	Funcionalidades	Puntuación
Postman	3	3	3	3	2	3	3	20
SoapUI	2	2	2	2	2	2	2	14
Jmeter	2	2	2	2	3	2	1	14
Insomnia	2	2	2	2	2	2	1	13

Tabla 9 Evaluación de Herramientas para Pruebas API.

Evaluación para herramientas de pruebas de rendimiento y carga.

Herramienta	Facilidad de uso	Compatibilidad	Escalabilidad	Soporte y Comunidad	Costo	Mantenimiento y Actualización	Funcionalidades	Puntuación
JMeter	2	3	3	3	3	3	3	20
Gatling	2	2	3	2	3	2	2	16
LoadRunner	2	3	3	3	1	3	3	18
K6	2	2	3	2	3	2	2	16

Tabla 10 Evaluación Herramientas Pruebas de Rendimiento y Carga.

Evaluación para herramientas de pruebas estáticas de código.

Herramienta	Facilidad de uso	Compatibilidad	Escalabilidad	Soporte y Comunidad	Costo	Mantenimiento y Actualización	Funcionalidades	Puntuación
SonarQube	3	3	3	3	2	3	3	20
PMD	2	2	2	2	3	2	2	15
CheckStyle	2	2	2	2	3	2	2	15
ESLint (JS)	2	3	2	2	3	2	2	16

Tabla 11 Evaluación Herramientas para Pruebas Estáticas de Código.

2.13 Equipo de Trabajo y Capacitación

El proceso de pruebas del Sistema Informático para la Gestión de Contratos de Personal Docente es realizado por el equipo de tesina de la Especialización en Ingeniería de la Calidad, bajo la guía de la persona docente responsable del proyecto. Este equipo asume las actividades de planificación, diseño, ejecución y documentación de las pruebas, tanto en su modalidad manual como automatizada, de acuerdo con lo establecido en el plan de pruebas.

En términos generales, el equipo se organiza para cubrir las siguientes funciones:

- **Planificación y coordinación de pruebas**, que incluye la elaboración y actualización del plan de pruebas, la definición del alcance por iteración y la priorización de módulos y casos de prueba.
- **Diseño de casos de prueba,** a partir de los requisitos funcionales y no funcionales del sistema, de las historias de usuario y de los flujos de negocio identificados en el diagnóstico del proyecto.
- **Ejecución de pruebas manuales,** mediante la aplicación de la matriz de casos de prueba sobre los módulos priorizados, el registro de resultados en los formatos establecidos y la documentación de incidentes encontrados.
- **Desarrollo y ejecución de pruebas automatizadas**, utilizando las herramientas seleccionadas en el proyecto (Postman, JMeter y Selenium) para la automatización de pruebas de API, de rendimiento y de interfaz de usuario.
- **Gestión de defectos y métricas**, a través del registro estructurado de incidencias, la clasificación por severidad y prioridad, el seguimiento de su estado y la consolidación de indicadores de prueba que sirvan de base para el informe de resultados.

Debido a la naturaleza del proyecto y al uso de herramientas especializadas de pruebas, se identifican necesidades de capacitación específicas para el equipo de trabajo, entre las que destacan:

- **Uso de herramientas de prueba:** formación práctica en el manejo de Postman para pruebas de APIs, JMeter para pruebas de rendimiento y Selenium para la automatización de flujos de interfaz, incluyendo la configuración de entornos, el diseño de casos automatizados y la interpretación de reportes.
- **Estándares y buenas prácticas de pruebas:** refuerzo de conceptos relacionados con la norma ISO/IEC/IEEE 29119, el diseño sistemático de casos de prueba, la clasificación de tipos y niveles de prueba, y la aplicación de criterios de aprobación, fallo, suspensión y reanudación definidos en el plan.
- **Gestión de defectos y trazabilidad:** capacitación en el uso de la herramienta de seguimiento de incidencias (por ejemplo, Azure DevOps u otra plataforma similar), para asegurar el registro adecuado de defectos y la trazabilidad entre requisitos, casos de prueba y resultados de ejecución.

La atención a estas necesidades de capacitación contribuye a que el equipo ejecute las pruebas de manera más rigurosa y eficiente, y a que los resultados obtenidos sean confiables y útiles para la evaluación de la calidad del Sistema de Gestión de Contratos de Personal Docente.

2.13.1 Recurso Humano

Para la realización del proyecto se ha determinado que el recurso humano se distribuirá de la siguiente manera.

Recurso	Nombre	Contacto	Rol
QA1	Ronald Antonio Sermeño	sa16007@ues.edu.sv	Líder QA y analista QA
QA2	Griselda Rosibel Fernández	fb19012@ues.edu.sv	Analista QA
QA3	José René Lucero	lb19005@ues.edu.sv	Analista QA

Tabla 12 Recurso Humano para el Proyecto.

2.14 Responsabilidades del Equipo de Pruebas

Para la asignación de actividades al recurso humano utilizamos la matriz RACI (R: Responsable, A: Apoyo, C: Consultado, I: Informado).

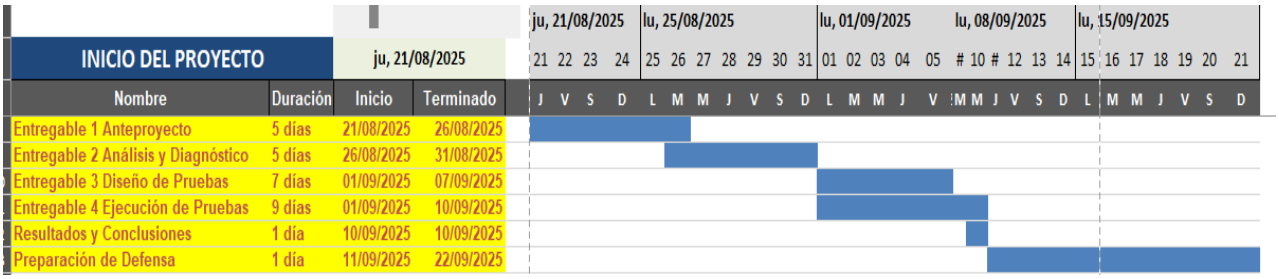
No.	Actividad/Tarea	Ronald Sermeño	Griselda Benítez	José Lucero
1	Definir la estrategia y el plan de pruebas	R	AI	AI
2	Coordinar y supervisar las actividades del equipo de pruebas	R	AI	AI
3	Asegurar que se cumplan los estándares y métricas de calidad	R	AI	AI
4	Revisar y aprobar los casos de pruebas, y los informes de defectos.	RI	RI	RI
5	Diseñar y documentar casos de prueba.	RI	RI	RI
6	Ejecutar pruebas manuales y automatizadas	RI	RI	RI
7	Reportar y documentar incidencias.	RI	RI	RI
8	Validar la corrección de defectos.	AI	AI	RI
9	Garantizar la trazabilidad entre requerimientos y casos de prueba.	RI	RI	RI
10	Validar criterios de aceptación.	RI	RI	RI
11	Apoyar en la priorización de pruebas críticas.	RI	RI	RI
12	Elaborar métricas de calidad.	RI	AI	AI
13	Configurar y mantener entornos de prueba.	R	AI	AI
14	Gestionar bases de datos de prueba.	R	AI	AI
15	Asegurar disponibilidad de recursos tecnológicos.	R	AI	AI

Tabla 13 Asignación de Actividades RACI.

2.15 Calendario de Pruebas

Por motivos de espacio, en este documento se presenta únicamente la versión resumida del cronograma con las fases principales del proyecto.

Para consultar el cronograma completo y detallado (con todas las actividades y entregables), se comparte el siguiente enlace: Calendario de Actividades.xlsx



INICIO DEL PROYECTO		ju, 21/08/2025	
Nombre	Duración	Inicio	Terminado
Entregable 1 Anteproyecto	5 días	21/08/2025	26/08/2025
Entregable 2 Análisis y Diagnóstico	5 días	26/08/2025	31/08/2025
Entregable 3 Diseño de Pruebas	7 días	01/09/2025	07/09/2025
Entregable 4 Ejecución de Pruebas	9 días	01/09/2025	10/09/2025
Resultados y Conclusiones	1 día	10/09/2025	10/09/2025
Preparación de Defensa	1 día	11/09/2025	22/09/2025

2.16 Plan de Riesgos y Contingencias

El proceso de pruebas del Sistema Informático para la Gestión de Contratos de Personal Docente está expuesto a diversos riesgos que pueden afectar el cumplimiento del alcance, el cronograma y los objetivos de calidad definidos en el plan de pruebas. Por ello, además de identificar los riesgos asociados a las pruebas, se establece un plan de riesgos y contingencias que define acciones preventivas, medidas de mitigación y respuestas ante la eventual materialización de dichos riesgos.

Este plan se enfoca en los riesgos más relevantes para la ejecución de las pruebas manuales y automatizadas, considerando aspectos técnicos, organizacionales y de gestión, e integra sus acciones con el calendario general de trabajo del proyecto.

2.16.1 Riesgos de Calidad Detectados

Para la revisión de los riesgos de calidad que el proyecto pueda enfrentar los clasificaremos en dos tipos de riesgos:

- Riesgo de Proyecto: son aquellos riesgos que afectan a la gestión del proyecto y su éxito, en relación con el cumplimiento de plazos, presupuesto y la disponibilidad de recursos.
- Riesgos de Producto: están relacionados con la calidad del producto, donde impacta en funcionalidades defectuosas o problemas con la usabilidad del sistema.

Para lograr evaluar los riesgos de calidad nos auxiliaremos de puntajes para la probabilidad con la que puede suceder el riesgo y el impacto que esta pueda tener.

En la medición del riesgo de calidad para el impacto utilizaremos una escala del 1 al 5 siendo 5 el más alto impacto.

Para el caso de la probabilidad se medirá en función de la ocurrencia que se pueda materializar el riesgo.

Donde obtenemos la siguiente tabla resumen para la medición del riesgo según la tipología.

Riesgo Cuantitativo		Impacto				
		1	2	3	4	5
Probabilidad		Muy baja	Baja	Moderada	Alta	Muy alta
1	Muy baja	1	2	3	4	5
2	Baja	2	4	6	8	10
3	Moderada	3	6	9	12	15
4	Alta	4	8	12	16	20
5	Muy alta	5	10	15	20	25

Tabla 14 Tabla de puntajes de riesgos Probabilidad vs Impacto.

Con la matriz de riesgo podemos determinar el menor riesgo asociado a nuestro proyecto tendrá la puntuación de 1, mientras que para el mayor riesgo tendrá una puntuación de 25. A continuación definiremos los criterios para cada tipología.

Criterios de Probabilidad.

Para poder cuantificar y medir la probabilidad para la aplicación y en base a los datos presentados, tendremos los siguientes criterios para evaluar la ocurrencia de probabilidad de la siguiente forma:

Criterios de Probabilidad		
Nivel	**Probabilidad**	**Ocurrencia**
1	Muy Baja	El riesgo es posible que ocurra entre 0 y 2 veces de cada 10 usos.
2	Baja	El riesgo es posible que ocurra entre 3 y 4 veces de cada 10 usos.
3	Moderada	El riesgo es posible que ocurra entre 5 y 6 veces de cada 10 usos.
4	Alta	El riesgo es posible que ocurra entre 7 y 8 veces de cada 10 usos.
5	Muy Alta	El riesgo es posible que ocurra entre 9 y 10 veces de cada 10 usos.

Tabla 15 Criterios de Probabilidad.

Criterios de Impacto.

Para evaluar los criterios de impacto tendremos la siguiente matriz que nos permitirá cuantificar el impacto de cada riesgo.

Criterios de Impacto		
Nivel	**Impacto**	**Descripción**
1	Muy Baja	Si el caso de prueba fallara, el impacto no afectaría la experiencia del usuario ni sus funcionalidades
2	Baja	Si el caso de prueba fallara, la funcionalidad cumplirá parcialmente el objetivo, con afectación menor y sin ningún impacto en funcionalidades relacionadas
3	Moderada	Si el caso de prueba fallara, la funcionalidad presentaría fallos significativos que afectan el uso, pero el sistema sigue operativo.
4	Alta	Si el caso de prueba fallara, la funcionalidad quedaría inoperante, pero sin afectar y bloquear funcionalidades críticas.
5	Muy Alta	Si el caso de prueba fallara, el sistema o funcionalidad quedaría completamente inoperante con errores graves y bloqueantes afectando de manera crítica al sistema.

Tabla 16 Criterios de Impacto.

A continuación, detallamos los riesgos identificados para la realización exitosa de nuestro proceso de pruebas de calidad para el sistema.

Riesgo	Probabilidad	Impacto	Severidad	Estrategia de Mitigación
Automatización mal implementada	4	5	20	Revisar los scripts de manera periódica, aplicar buenas prácticas de automatización y utilizar un versionador de código.
Ambientes de pruebas inestables	4	4	16	Se deberá realizar monitoreos periódicos que permitan revisar el estado actual del sistema y ejecutar pruebas críticas.
Cobertura de las pruebas incompletas	3	5	15	Elaborar matriz de casos de prueba y que permita priorizar la ejecución de módulos críticos.
Defectos críticos detectados tarde	3	5	15	Realizar smoke testing y uso de las pruebas automatizadas para los módulos críticos del sistema.

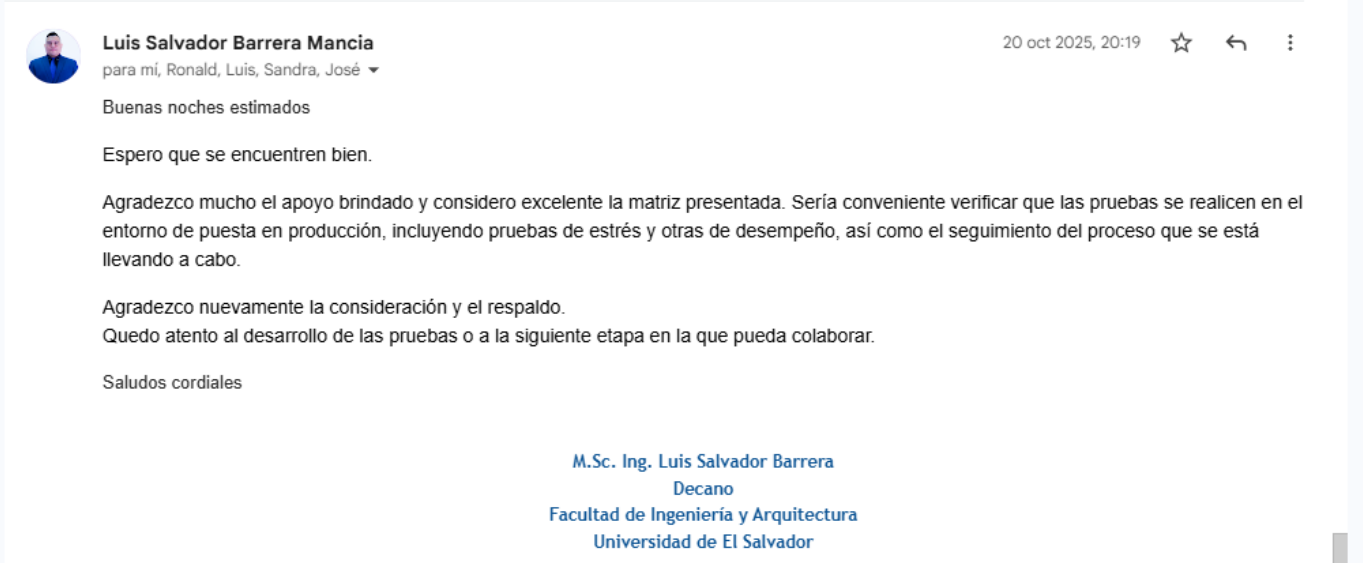
Tabla 17 Tabla de Riesgos de Calidad Parte 1.

Riesgo	Probabilidad	Impacto	Severidad	Estrategia de Mitigación
Vulnerabilidades de seguridad no detectadas	3	4	12	Uso de herramientas que nos permitan realizar análisis de seguridad y pruebas de penetración para identificar vulnerabilidades.
Problemas de compatibilidad	3	3	9	Realizar una investigación previa para la verificación de los navegadores web en tendencia y con mayor uso.
Retrasos en presentación de entregables	3	3	9	Asegurar el cumplimiento de la metodología Agile en nuestro proyecto realizando entregas graduales y estableciendo un cronograma del plan de trabajo.
La información utilizada para la ejecución de las pruebas es repetitiva y no presenta variaciones	2	3	6	Diseño y elaboración de un set de datos variados para cubrir casos positivos, negativos, límites y excepciones. Revisar que el set de datos es realista y presenten las condiciones del negocio.
Problemas de mantenibilidad a largo plazo	2	3	6	Fomentar la documentación adecuada para facilitar futuras actualizaciones.
Costo inicial de las herramientas	2	2	4	Se deberá de implementar el uso de herramientas de open-source, es decir que se trabajará con herramientas gratis y no de paga durante la ejecución del proyecto.
Falta de retroalimentación durante las pruebas	2	2	4	Verificar la participación del equipo, asegurando que cada miembro del equipo comparta sus análisis y observaciones.

Tabla 18 Tabla de Riesgos de Calidad Parte 2.

2.17 Aprobaciones del Plan de Pruebas

El presente plan de pruebas cuenta con el aval del M.Sc. Ing. Luis Salvador Barrera Mancía, Decano de la FIA-UES, quien manifestó su conformidad con la matriz de pruebas y recomendó la continuación del proceso en entornos controlados de preproducción.



Luis Salvador Barrera Mancia 20 oct 2025, 20:19

para mí, Ronald, Luis, Sandra, José

Buenas noches estimados

Espero que se encuentren bien.

Agradezco mucho el apoyo brindado y considero excelente la matriz presentada. Sería conveniente verificar que las pruebas se realicen en el entorno de puesta en producción, incluyendo pruebas de estrés y otras de desempeño, así como el seguimiento del proceso que se está llevando a cabo.

Agradezco nuevamente la consideración y el respaldo.
Quedo atento al desarrollo de las pruebas o a la siguiente etapa en la que pueda colaborar.

Saludos cordiales

M.Sc. Ing. Luis Salvador Barrera
Decano
Facultad de Ingeniería y Arquitectura
Universidad de El Salvador

Ilustración 6: Visto Bueno Aprobación de Plan de Pruebas.

CAPÍTULO III. Caso de Estudio: Plan de Pruebas Aplicado al Sistema de Gestión de Contratos de Personal Docente de la FIA-UES

3.1 Antecedentes del Sistema y del Proyecto de Pruebas

3.1.1 Análisis de la Situación Actual.

Descripción del Sistema

El Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador (FIA-UES) es una herramienta tecnológica diseñada para optimizar y automatizar el proceso administrativo de contratación docente, reduciendo la carga de trabajo manual y minimizando errores en la gestión.

El sistema integra diferentes módulos que permiten a los usuarios registrar, validar, gestionar y dar seguimiento a los contratos de forma centralizada. Está estructurado para atender a ocho tipos de usuarios con funciones específicas: Administrador, Super Administrador, Candidato, Recursos Humanos, Asistente Financiero, Director de Escuela, Asistente Administrativo y Decano, cada uno con accesos y permisos definidos según su rol en el proceso de contratación.

Ilustración 7 Diagrama de Funcionalidades del Sistema.

Entre sus principales funcionalidades se encuentran:

1. **Gestión de usuarios y roles:** creación, edición, control de accesos y registro en bitácoras.
2. **Gestión académica y administrativa:** registro de ciclos, planes de estudio, materias, escalafones, cargos y actividades docentes.
3. **Procesos de contratación**: desde el registro de datos y documentos por parte de candidatos, validación de solicitudes por Recursos Humanos, hasta la generación automatizada de contratos con base en plantillas predefinidas.
4. **Soporte a la toma de decisiones**: a través de dashboards gráficos y reportes que permiten visualizar contrataciones por escuela, montos totales y estadísticas generales.

Gracias a esta estructura, el sistema no solo mejora la eficiencia y transparencia en el manejo de la información, sino que también garantiza mayor trazabilidad y control en cada etapa del proceso. Con ello, se convierte en un recurso estratégico para la FIA-UES, permitiendo mantener un flujo administrativo ágil y confiable en la contratación de personal docente.

3.1.2 Arquitectura Tecnológica

El sistema de Gestión de Contratos de Docentes se soporta por una arquitectura tecnológica multicapa, orientada a la web, que separa responsabilidades y facilita la escalabilidad, el mantenimiento y la seguridad. A continuación, se describe su composición general, componentes tecnológicos, servicios transversales.

3.1.3 Descripción General por Capas.

La arquitectura se organiza en tres capas principales:

Capa	Descripción	Elementos Principales
Presentación (Clientes/SPA).	Interfaz web responsiva que permite a candidatos, docentes, personal administrativo y autoridades interactuar con el sistema.	Navegador web, aplicación SPA, formularios, vistas de catálogos, vistas de contratación, vistas de contratos y reportes.
Lógica de Negocio (Servicios /API).	Capa intermedia que implementa las reglas del negocio, valida la información y expone servicios REST al cliente.	API REST, control de autenticación/autorización, orquestación de procesos.
Datos (Persistencia y Almacenamiento).	Gestiona la información transaccional y documental, garantizando disponibilidad y respaldo.	Base de datos relacional, almacenamiento de archivos, sistema de cache y colas de trabajo.

Tabla 19: Arquitectura Tres Capas del Sistema.

3.1.4 Componentes Tecnológicos Principales

En la siguiente tabla se detallan los principales tecnológicos utilizadas para su implementación dentro del proyecto.

Componente	Funcionalidad	Tecnología utilizada
Frontend (SPA)	Provee la interfaz gráfica web responsiva para los diferentes tipos de usuario. Incluyendo formularios, validaciones, vistas de reportes y control de permisos en la interfaz.	React 17 + TypeScript + Ant Design 4; React Router DOM v5; React Query v3; Axios; CASL (Control de Permisos en UI).
Backend (API REST)	Implementa reglas de negocio gestiona contratos, candidatos, escalafones, ciclos. Expone endpoints REST para comunicación con el frontend. Incluye autenticación, autorización, validación y bitácoras.	Laravel 10 con PHP 8.1. Sanctum. Spatie Roles & Permissions.
Base de datos	Gestiona información estructurada, garantizando integridad y persistencia de datos críticos.	PostgreSQL16 (Motor relacional en Amazon RDS, con ORM Eloquent de Laravel)
Almacenamiento de archivos	Conserva documentos digitalizados asociados a los procesos de contratos, completitud del expediente de candidatos. Permite acceso controlado y visualización por roles autorizados.	Acceso mediante presigned URLs y políticas de permisos.
Caché/Colas	Acelera la consulta de datos frecuentes y ejecuta tareas pesadas en segundo plano.	Laravel Horizon como monitor de colas.
Reportes	Generación de documentos administrativos y contratos en distintos formatos. Soporta descargas individuales y masivas.	Maatwebsite Laravel Excel Laravel dompdf PhpOffice/PhpWord

Tabla 20 Componentes Tecnológicos Principales Parte 1.

Componente	Funcionalidad	Tecnología utilizada
Notificaciones	Comunicación automática con usuarios a través de correo electrónico para altas, aprobaciones, rechazos y recordatorios.	Laravel Mailer.
Tareas Programadas	Automatiza la ejecución de procesos recurrentes como envío de recordatorios, depuración de registros y actualización de estados.	Laravel Scheduler (Artisan + Cron Jobs).

Tabla 21 Componentes Tecnológicos Principales Parte 2.

3.1.5 Servicios Transversales

Los servicios transversales se organizan principalmente en dos ejes: seguridad y observabilidad, los cuales se describen a continuación.

Seguridad:

- Autenticación y autorización basadas en roles, que regulan el acceso a funcionalidades críticas.
- Políticas de validación y control de archivos adjuntos, asegurando el cumplimiento de formatos y tamaños permitidos.
- Registro de actividades en bitácoras, que permiten la trazabilidad de acciones de los usuarios y facilitan auditorias posteriores.

Observabilidad:

- Generación de logs, que consolidan la información de eventos relevantes para análisis y depuración.
- Definición de métricas básicas de desempeño que permiten evaluar el comportamiento del sistema en distintos escenarios de operación.

3.1.6 Contexto del Desarrollo y Alcance del Proyecto

Contexto del Desarrollo

El Sistema de Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador (FIA-UES) surge como respuesta a la necesidad de optimizar los procesos administrativos relacionados con la contratación académica. El desarrollo se enmarca en un entorno académico, con recursos limitados en cuanto a presupuesto e infraestructura tecnológica, lo que ha llevado a priorizar el uso de herramientas de código abierto y metodologías de trabajo adaptadas a la realidad institucional.

El proyecto se ha construido bajo una arquitectura cliente-servidor, con un backend en Laravel (PHP 8.x) y una interfaz en React (TypeScript), integrados con una base de datos PostgreSQL. Además, se utiliza Docker para garantizar portabilidad y Azure DevOps/GitHub para

la gestión de versiones y control de despliegues. La fase de validación contempla la ejecución de pruebas manuales y automatizadas, con el fin de garantizar que el sistema cumpla con los requisitos funcionales y no funcionales establecidos.

El contexto también refleja limitaciones propias de un entorno universitario, como la falta de presupuesto para licencias comerciales, la infraestructura limitada para pruebas de alto consumo (carga y estrés) y la resistencia al cambio por parte de algunos usuarios acostumbrados a procesos manuales. Aun así, el proyecto busca sentar un precedente institucional en la aplicación de metodologías de calidad de software.

Alcance del Proyecto

El alcance de este proyecto está definido en función de las metas trazadas y los recursos disponibles, comprendiendo los siguientes puntos principales:

- Evaluación y validación funcional del sistema mediante la ejecución de pruebas orientadas a los módulos críticos: registro de candidatos, validación de documentos, generación de contratos y asignación de carga académica.
- Automatización de pruebas en flujos repetitivos y de alta criticidad (ejemplo: inicio de sesión, validación de roles, generación de contratos).
- Implementación de una matriz de pruebas que asegure trazabilidad entre los requerimientos y los casos validados.
- Documentación técnica del proceso de pruebas, incluyendo evidencias de ejecución, resultados comparativos y recomendaciones de mejora.
- Generación de reportes y dashboards que permitan a los responsables administrativos visualizar métricas de calidad y confiabilidad del sistema.
- Transferencia de conocimiento mediante manuales técnicos y de usuario, que faciliten la continuidad del proceso de pruebas y el mantenimiento futuro del sistema.

3.2 Contexto del Problema y Justificación

3.2.1 Planteamiento del Problema

Actualmente, la Universidad de El Salvador, por medio de la Facultad de Ingeniería y Arquitectura, está impulsando varios proyectos enfocados en la creación de sistemas informáticos, buscando así lograr la excelencia operativa en las diferentes áreas en las que se ve involucrada. Dentro de estos proyectos, se vio la necesidad de crear un sistema informático que permitiera la gestión de contratos para el personal docente de la Facultad de Ingeniería y Arquitectura, proyecto que se encuentra en su etapa final de desarrollo.

Sin embargo, durante este proceso se ha identificado una deficiencia importante: el sistema carece de un proceso estructurado y continuo de pruebas de calidad del software. Este problema representa un riesgo, ya que pueden existir deficiencias no detectadas en el sistema y que podrían impedir su correcto funcionamiento, lo que podría derivar a problemas técnicos que

afecten directamente al proceso de gestión de contratación de personal docente, impactando así la eficiencia y confiabilidad del proceso administrativo.

Por ello, es indispensable realizar pruebas de caja negra, centradas en las entradas y salidas del sistema, sin necesidad de conocer su lógica interna. Estas pruebas permitirán:

- Validar que el sistema responda correctamente ante distintas entradas de datos ingresados por los diferentes tipos de usuario (administrador, candidatos, recursos humanos, director, decanos) y también fechas de contratos, tipos de contrato.
- Detectar errores en la generación de resultados, reportes o confirmaciones que el sistema debe entregar al usuario.
- Garantizar que los requisitos funcionales se cumplan y que la experiencia del usuario no se vea afectada por fallos o interrupciones.

El problema radica en que, sin pruebas de calidad adecuadas, las entradas podrían no ser validadas correctamente, el proceso interno podría generar inconsistencias, y las salidas esperadas podrían verse afectadas con errores que impacten la eficiencia del sistema y la confianza en el proceso de contratación.

Por lo tanto, se hace necesario establecer un proceso de pruebas manuales y automatizadas, con el fin de verificar que las entradas sean procesadas correctamente y que las salidas coincidan con los requisitos funcionales definidos, garantizando así que el sistema opere de forma confiable y sin interrupciones.

3.2.2 Matriz FODA.

F: FORTALEZAS	D: DEBILIDADES
F1. El sistema ya se encuentra en su etapa final de desarrollo, por lo tanto, se encuentra apto para validar su funcionamiento bajo diferentes escenarios. **F2.** Para el desarrollo de la aplicación se han utilizado herramientas de software tendencia en la actualidad. **F3.** Disponibilidad de API que puede facilitar la integración con otros sistemas. **F4.** El sistema está diseñado y desarrollado exclusivamente por la Universidad de El Salvador lo que facilita el control del alcance de las pruebas y reduce dependencias externas. **F5.** La automatización de pruebas nos permitirá poder identificar de forma continua errores del sistema. **F6.** El sistema facilitará la gestión de contratos de docentes, donde está podía ser demasiado manual y propensa a errores.	**D1**. La documentación técnica se encuentra incompleta, lo que puede impactar en la ejecución de las pruebas. **D2**. A pesar de que el sistema se encuentra culminado durante su ciclo de vida no se ha realizado pruebas de calidad de software. **D3**. Las pruebas unitarias no han sido implementadas por los desarrolladores del sistema. **D4**. No existe un presupuesto para poder adoptar herramientas de pago o certificaciones avanzadas. **D5.** No se han establecido métricas para analizar el estado de calidad del sistema. **D6**. No todas las pruebas pueden automatizarse, es necesario que exista criterio humano para validar.
O: OPORTUNIDADES	**A: AMENAZAS**
O1. Uso de herramientas Open Source para reducir los costos y fortalecer la automatización de las pruebas de calidad de software. **O2.** Proporcionar a los usuarios finales satisfacción garantizando que el sistema sea confiable, eficiente y que cumpla con los objetivos establecidos. **O3.** La certificación de las pruebas de calidad puede aplicarse bajo una normativa internacional, incrementando la credibilidad de la certificación de calidad al software. **O4.** Competitividad del sistema al aplicar pruebas de calidad al software. **O5**. Las pruebas automatizadas pueden escalar cuando sea necesario agregar nuevas funcionalidades al sistema. **O6**. El sistema reduce carga laboral liberando tiempo que puede ser asignado a actividades de mayor valor.	**A1**. Dependencias de recursos asignados de TI al sistema, lo cual puede afectar la ejecución de pruebas. **A2**. En la actualidad los avances continuos en tecnologías de QA pueden dejar fuera de uso u obsoletas las herramientas a utilizar. **A3.** El sistema puede sobrecargarse cuando no se realiza una buena planificación con respecto a su crecimiento impactando en el rendimiento. **A4.** Falsos positivos en la ejecución de las pruebas, generando una falsa sensación de seguridad. **A5**. Fallas en el entorno de pruebas, impactando en los resultados ya que no se estaría mostrando la realidad. **A6.** Si no se planifica de forma correcta, las pruebas automatizadas pueden tardar demasiado presentando cuellos de botellas. **A7.** Si no se actualizan constantemente los scripts, las pruebas automatizadas pueden resultar inservibles. **A8**. Uso de datos reales del personal docente puede comprometer la seguridad del sistema si estos no se enmascaran. **A9:** Resistencia al cambio por parte de los usuarios finales del sistema

Tabla 22 Matriz FODA del Sistema Gestión de Contratos de Docentes de la FIA

Identificando las diferentes tipologías de nuestra matriz FODA, podemos definir las siguientes estrategias:

Estrategias FO (Fortalezas y Oportunidades).

- Aprovecharemos el uso de herramientas Open Source (O1) de la mano con la experiencia de tecnologías de software tendencia en la actualidad (F2) para contribuir en la automatización de las pruebas sin incrementar los costos.
- Potenciaremos la credibilidad de la Universidad de El Salvador (O3) con el uso de pruebas automatizadas (F5) para respaldar la confiabilidad del sistema frente a posibles auditorías internas o externas.
- Mediante el uso de APIS (F3) pueden escalar nuevas funcionalidades (O5) y con ello asegurar que las pruebas de calidad continúen siendo aptas a pesar de estos cambios aplicados.

Estrategias DO (Debilidades y Oportunidades).

- Implementar métricas que nos ayuden a medir la calidad del sistema (D5) aplicando certificaciones avaladas internacionalmente (O3) para obligar a establecer indicadores de desempeño y estándares de validación.
- Reducir la ausencia de las pruebas unitarias (D3) a través de la documentación del uso de herramientas Open Source (O1) que no requiera coste para su uso.
- Enfrentar la falta de presupuesto (D4) optando por la selección y uso de herramientas gratuitas y comunitarias que nos permitan cumplir con la escalabilidad de las pruebas automatizas (O5).

Estrategias FA (Fortalezas y Amenazas).

- Aprovechar que el sistema no depende de externos únicamente por el equipo de desarrollo de la Universidad de El Salvador (F4) para poder responder oportunamente a la obsolescencia de las herramientas QA (A2), aplicando nuevas versiones o frameworks.
- Auxiliarnos de las pruebas de rendimiento del sistema (F5) para combatir los riesgos de sobrecarga que pueda sufrir el sistema, validando la estabilidad de los flujos críticos del sistema.
- Beneficiarnos de la implementación de APIS (F3) para fortalecer los mecanismos de seguridad y así reducir la amenaza del uso incorrecto de los datos (A8).

Estrategias DA (Debilidades y Amenazas).

- Mitigar la falta de documentación técnica del sistema (D1) para evitar que los scripts automatizados se vuelvan inservibles (A7), documentando de manera sólida garantizando la mantenibilidad.
- Reducir la ausencia de pruebas de calidad durante el ciclo de desarrollo del sistema (D2) utilizan un plan mixto de pruebas automatizadas y manuales, lo que nos ayudará a disminuir la posibilidad de falsos positivos (A4).

- Responder a la falta de presupuesto (D4) con procesos de documentación de los scripts y las pruebas realizadas para confrontar el riesgo de fallas en el entorno de pruebas (A5) y cuellos de botella en la automatización (A6).

3.2.3 Diagrama Causa y Efecto

Con el fin de identificar las principales causas que afectan la calidad del sistema, se elaboró un Diagrama de Causa y Efecto (Ishikawa). Este tipo de diagrama permite representar de manera estructurada los factores que influyen en un problema central, en este caso, la deficiencia en la calidad de las pruebas del Sistema de Gestión de Contratos de Docentes. Las causas se agrupan en seis categorías principales:

- Herramientas,
- Personas
- Procesos y Tecnología
- Recursos
- Entorno
- Medición

Al clasificarlas en estas categorías nos facilita visualizar de forma integral los aspectos que deben ser atendidos para mejorar la calidad del software.

Diagrama Causa y Efecto

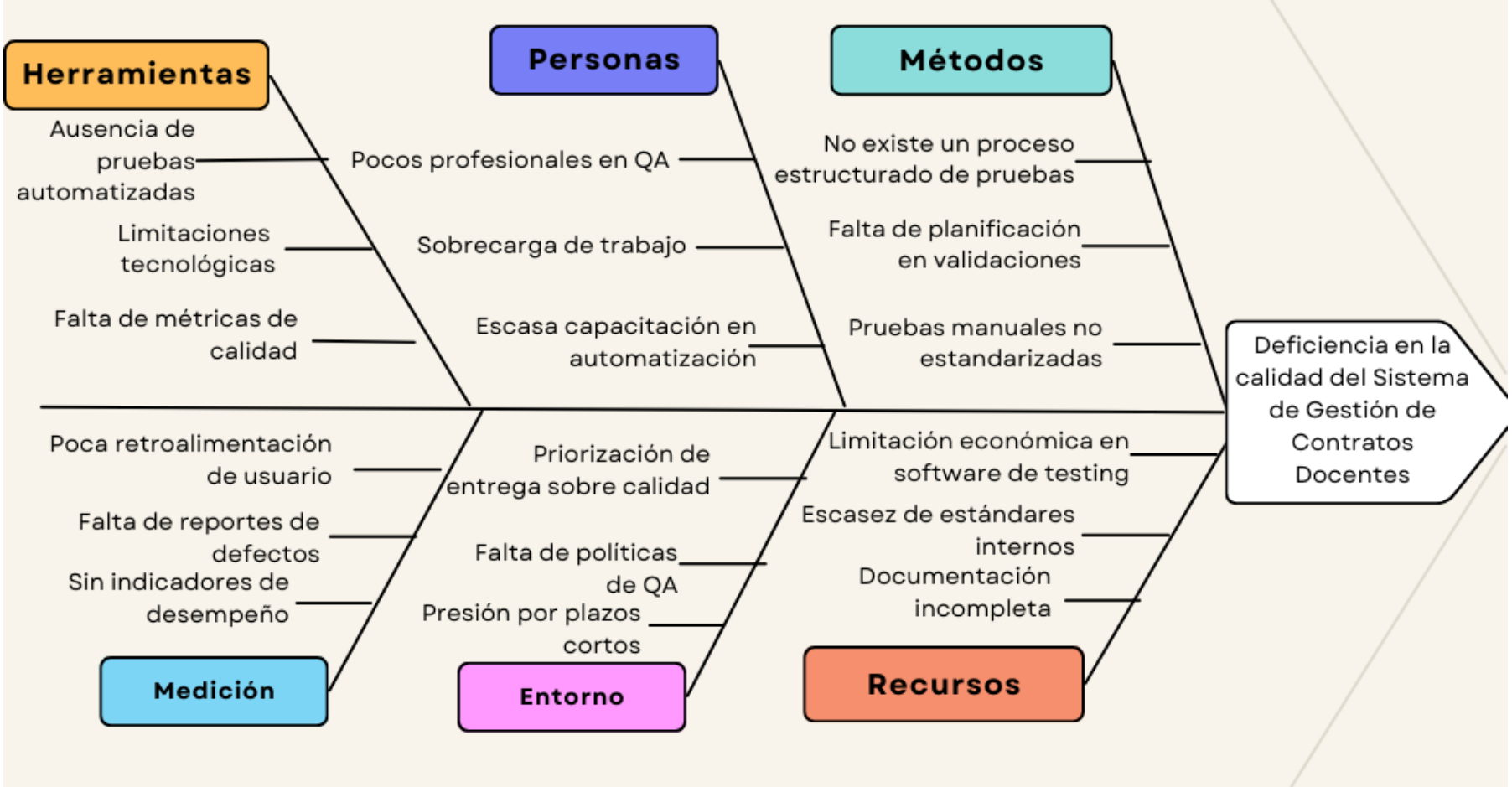


Ilustración 8 Diagrama Causa y Efecto Sistema de Contratos de Docentes de la FIA.

3.2.4 Solución Propuesta.

Mediante el uso de las diferentes herramientas de análisis que se utilizaron, como la matriz FODA, el diagrama causa y efecto y el enfoque de sistemas, podemos observar diferentes temas críticos que impactan en la calidad del software y la entrega satisfactoria del producto final.

Por lo que se propone el análisis, diseño, implementación y ejecución de un plan integral de pruebas de calidad y pruebas automatizadas orientado al sistema de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de El Salvador, con el fin de que se asegure y proporcione una adecuación funcional, eficiencia del desempeño en el uso del sistema y que el sistema ofrezca a los usuarios finales compatibilidad, usabilidad y fiabilidad.

Para el plan propuesto nos centraremos en los siguientes puntos importantes:

1. Elaborar un análisis de la situación del sistema, en donde buscaremos documentar la información necesaria para entender, mantener y adoptar las mejores alternativas para la ejecución de las pruebas de calidad tanto para las pruebas manuales como para las pruebas automatizadas.
2. Levantamiento de un documento formal que cuente con los casos técnicos con entradas, salidas esperadas, reglas del negocio asociadas y escenarios de pruebas.
3. Codificación de scripts automatizados con herramientas especializadas que ayuden de manera eficiente y rápida. Se estructura el código con un patrón modular que permita asegurar la mantenibilidad de los scripts.
4. Implementación de las pruebas automatizadas utilizando herramientas Open Source. La automatización de las pruebas nos permitirá poder validar, detectar errores de manera continua y proporcionará escalabilidad al sistema, si es necesario realizar futuras modificaciones.
5. Se buscará afrontar las limitaciones de presupuesto aprovechando el uso de herramientas gratuitas, implementando una metodología de trabajo que nos permita la entrega de valor continua, asegurando la actualización de los scripts de automatización y garantizando la sostenibilidad del sistema.
6. Entregar al equipo de calidad toda la documentación generada durante el proyecto, incluyendo scripts de prueba, reportes de ejecución, defectos detectados y manual de uso.
7. Comparación de tiempos y errores antes y después de la automatización. Se recopilarán observaciones del equipo de calidad sobre facilidad de uso, esfuerzo técnico requerido y reusabilidad de los scripts.

Con la ejecución exitosa del plan de integral de pruebas de calidad y pruebas automatizadas estamos buscando ofrecer diferentes beneficios, como lo son:

1. Con la reducción de errores y el aseguramiento del desempeño estable contribuirá con la confiabilidad del sistema.
2. La implementación de pruebas automatizadas se optimiza el tiempo invertido de trabajo del equipo de calidad.
3. Facilitar la repetición y el mantenimiento de las pruebas al contar con los scripts y manuales documentados.

4. Satisfacción de los usuarios finales al contar con un sistema funcional, confiable y ajustado a sus necesidades.

3.2.5 Definición del Problema

El Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador aún no cuenta con un proceso formal de pruebas de calidad. Esta ausencia hace que existan riesgos de errores no detectados que pueden afectar la confiabilidad y eficiencia del proceso de contratación docente. Por ello, el problema central consiste en la necesidad de implementar pruebas manuales y automatizadas que aseguren el buen funcionamiento, la estabilidad y la calidad del sistema.

Ilustración 9 Diagrama de definición del problema

3.3 Modelado de Negocio del Proceso de Contratación de Personal Docente

3.3.1 Enfoque del Sistema

El Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador puede comprenderse desde la perspectiva de la Teoría General de Sistemas (TGS), la cual establece que todo sistema está conformado por un conjunto de elementos interrelacionados que reciben entradas, las procesan y generan salidas, manteniendo un proceso de retroalimentación que asegura su estabilidad y mejora continua (Bertalanffy, 1968).

Aplicado a este caso, el sistema se estructura de la siguiente manera:

- **Entradas (Inputs):** información proporcionada por los usuarios del sistema, tales como datos personales de los candidatos, documentos académicos, escalafones, cargas horarias y requerimientos administrativos.
- **Procesos:** validación de la información ingresada, revisión de documentos, generación de contratos, asignación de carga académica y validación de los flujos administrativos.
- **Salidas (Outputs):** contratos generados correctamente, reportes administrativos, asignación efectiva de docentes a ciclos académicos y trazabilidad de la información.
- **Retroalimentación:** Identificación de errores mediante pruebas manuales y automatizadas, métricas de calidad y sugerencias de mejora que alimentan futuros ajustes del sistema.

De esta manera, el sistema no se limita a una herramienta tecnológica, sino que se concibe como un proceso integral orientado a garantizar la eficiencia y transparencia en la contratación docente. Además, bajo los lineamientos de la ingeniería de software, la calidad del sistema no debe evaluarse únicamente en términos de ausencia de errores, sino también en su capacidad para ser confiable, seguro, usable y mantenible en el tiempo (Sommerville, 2011)

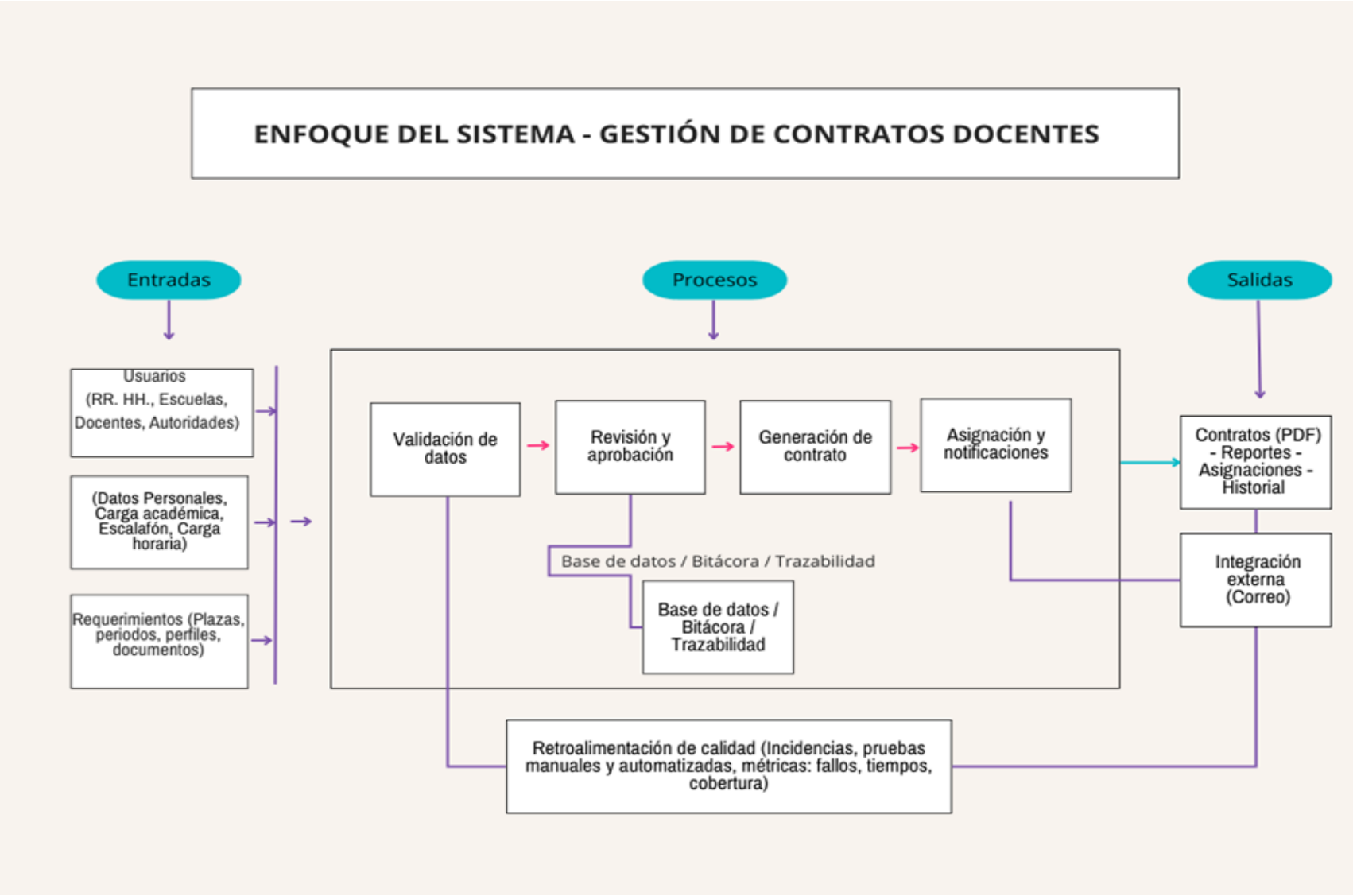


Ilustración 10 Diagrama Enfoque de Sistemas para la Gestión de Contratos de Docentes de la FIA

3.3.2 Proceso y Funcionalidades del Sistema.

El Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura (FIA-UES) fue diseñado con el objetivo de digitalizar y optimizar el proceso de contratación de personal docente, garantizando trazabilidad, seguridad y eficiencia administrativa.

El sistema involucra a diferentes actores, cada uno con un conjunto de funcionalidades específicas. A continuación, se describen los procesos principales y las funcionalidades por tipo de usuario:

Acceso y Funciones Generales

- **Inicio de sesión y autenticación de usuarios:** Se lleva a cabo mediante credenciales asignadas.
- **Cambio de contraseña y cierre de sesión**. Se lleva a cabo mediante un usuario creado.
- **Acceso al sistema:** Se lleva a cabo vía navegador web (recomendado Chrome/Firefox).

Tipos de Usuario y Funcionalidades

a) Administrador

- Gestión de usuarios (crear, editar, restablecer contraseñas).
- Consulta de bitácoras de uso del sistema.
- Gestión de ciclos académicos (crear, editar, eliminar).
- Administración de catálogos: bancos, escalafones, facultades, escuelas, planes de estudio, materias, cargos y actividades docentes.
- Gestión de autoridades en distintos niveles (central, facultad y escuela).
- Configuración de formatos de contrato.

b) Super Administrador

- Funciona como cuenta de soporte para casos críticos.
- Respaldo y gestión de usuarios cuando el administrador no puede intervenir.

c) Candidato (Docente postulante)

- Registro y edición de información personal.
- Carga de documentos requeridos.
- Gestión de horarios y actividades de permanencia.
- Consulta de solicitudes de contratación vinculadas a su perfil.

d) Recursos Humanos

- Validación de información y documentos de candidatos.
- Revisión y aprobación de solicitudes de contratación.
- Generación y validación de contratos.
- Gestión de plantillas de contrato en formato Word.

e) Asistente Financiero

- Revisión de solicitudes de contratación finalizadas.
- Generación de reportes financieros y administrativos.

f) Director de Escuela

- Gestión de planes de estudio y materias.
- Creación de solicitudes de contratación.
- Asignación de carga académica a docentes.

g) Asistente Administrativo

- Recepción y registro de solicitudes de contratación.
- Registro de acuerdos de Junta Directiva.
- Consulta de catálogos y registros del sistema.

h) Decano

- Acceso general a reportes y solicitudes de contratación.
- Visualización de dashboard con datos estadísticos del proceso de contratación.

Flujo del Sistema

Se desarrolla en varias etapas que integran la interacción de diferentes actores institucionales. A continuación, se describe de manera detallada:

1. **Creación de usuario candidato (Administrador)**
    a. El Administrador del sistema crea las cuentas de los candidatos/docentes en la plataforma, asignándoles credenciales de acceso.
    b. Opcionalmente, también gestiona roles, facultades y escuelas correspondientes al perfil del candidato.
2. **Registro y actualización de información (Candidato)**
    a. El Candidato accede al sistema con las credenciales proporcionadas.
    b. Completa su perfil con datos personales, documentación académica y profesional requerida (títulos, escalafón, dui entre otros documentos que solicita el sistema).
    c. Registra su disponibilidad horaria y actividades de permanencia.
3. **Validación de documentos y perfil (Recursos Humanos)**
    a. El área de Recursos Humanos (RRHH) revisa la información ingresada por el candidato.
    b. Se validan documentos (DUI, NIT, títulos, constancias) y se confirma la elegibilidad del docente.
    c. En caso de inconsistencias, RRHH puede devolver observaciones al candidato para correcciones.
4. **Creación de solicitudes de contratación (Director de Escuela)**
    a. El Director de Escuela genera solicitudes de contratación de acuerdo con las necesidades académicas del ciclo (materias asignadas y cantidad de horas a trabajar).
    b. Se asignan candidatos a dichas solicitudes según la validación realizada por RRHH.

5. **Recepción y registro de acuerdos (Asistente Administrativo)**
    a. El Asistente Administrativo recibe las solicitudes de contratación elaboradas por los Directores.
    b. Registra los acuerdos de Junta Directiva (JD) relacionados con dichas solicitudes.
    c. Consolida la información para enviarla a Recursos Humanos.
6. **Generación de contratos (Recursos Humanos)**
    a. Con base en las solicitudes validadas y los acuerdos, RRHH genera los contratos en el sistema utilizando los formatos oficiales predefinidos.
    b. Los contratos pueden exportarse en formato digital (Word/PDF) para su firma y archivo institucional.
7. **Validación y aprobación institucional (Autoridades y JD)**
    a. Las solicitudes y contratos pasan a revisión de las autoridades académicas (Decano, Junta Directiva y otras instancias).
    b. Una vez aprobados, los contratos quedan formalmente registrados en el sistema.
8. **Control, seguimiento y reportes (Decano y Autoridades)**
    a. El sistema ofrece un dashboard de control, accesible al Decano y autoridades, donde se consolidan indicadores clave:
        i. Número de solicitudes por ciclo.
        ii. Contratos aprobados y pendientes.
        iii. Distribución de carga académica.
    b. Se generan reportes administrativos y estadísticos que apoyan la toma de decisiones.

A continuación, se muestran los diagramas de flujos representados de forma gráfica para mayor entendimiento:

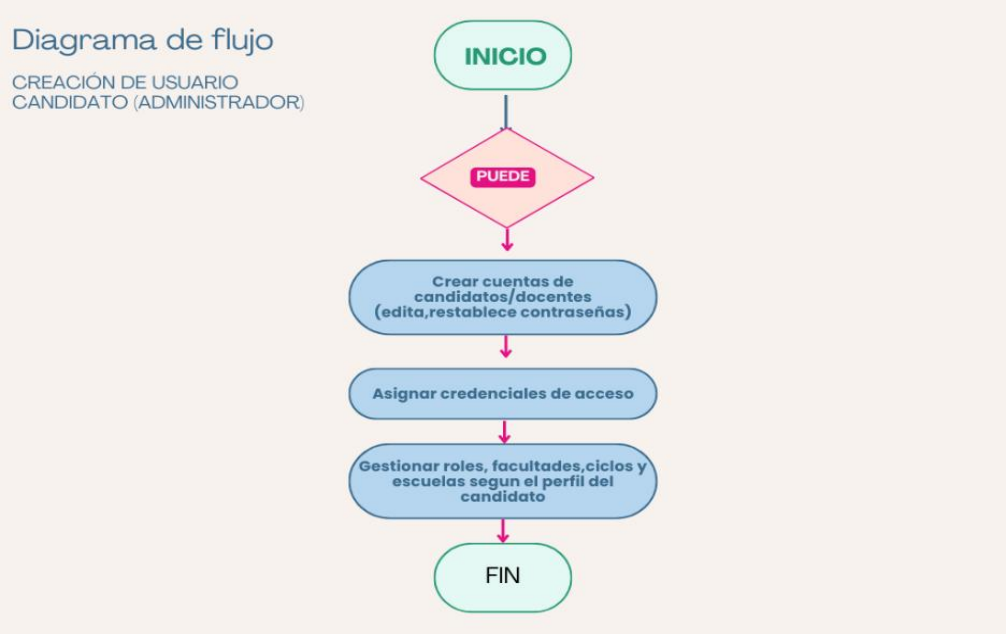


Ilustración 11: Diagrama de Flujo Creación de Usuario Candidato.

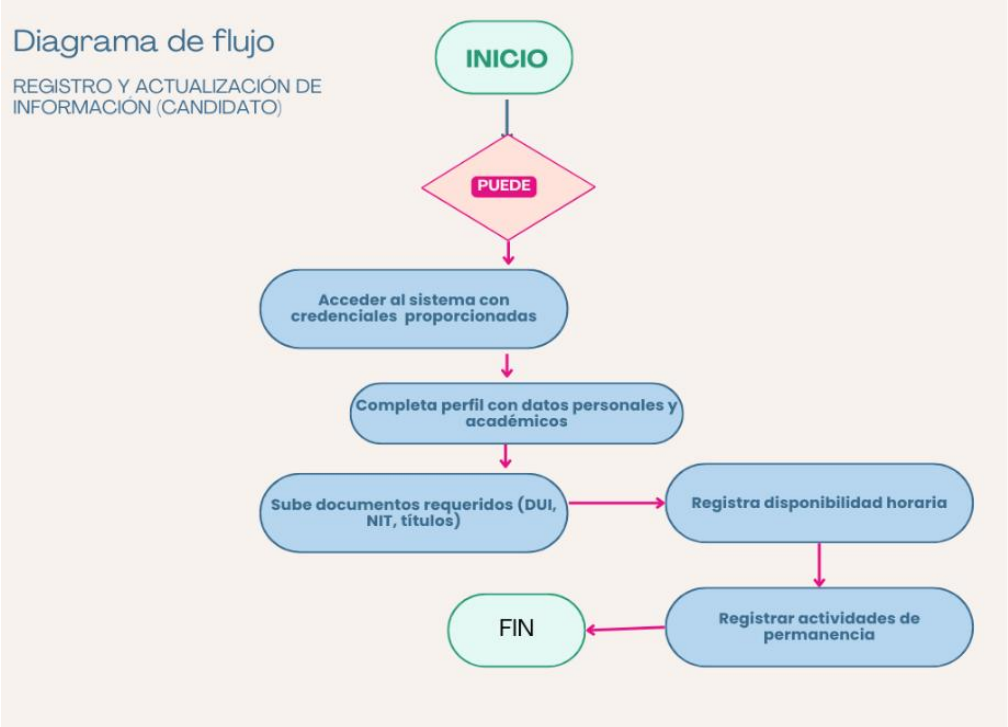


Ilustración 12: Diagrama Registro y Actualización de Información de Candidato.

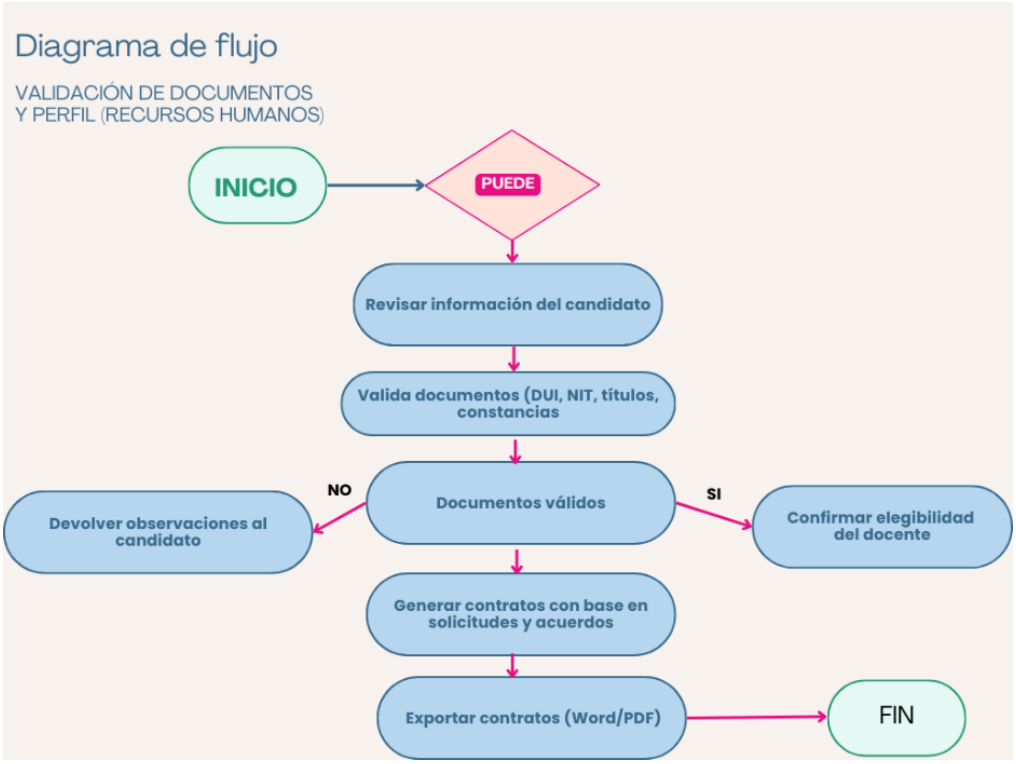


Ilustración 13: Diagrama de Flujo Validación de Documentos y Perfil.

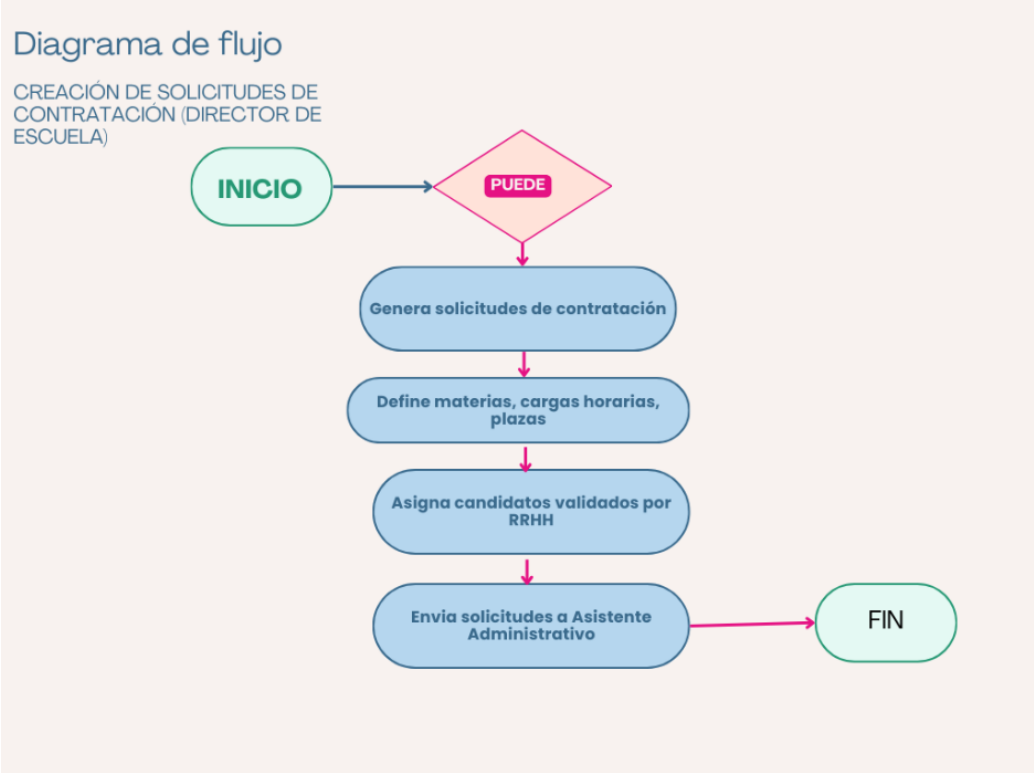


Ilustración 14: Diagrama de Flujo Creación de Solicitudes de Contratación.

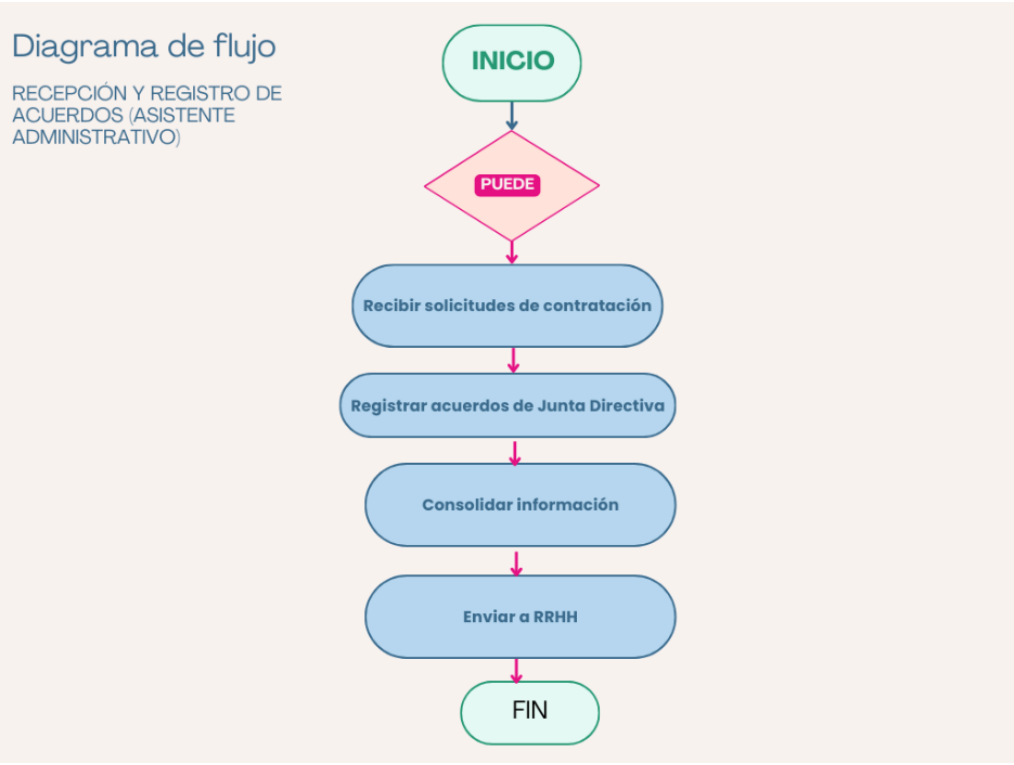


Ilustración 15: Diagrama de Flujo Recepción de Acuerdos de Junta Directiva.

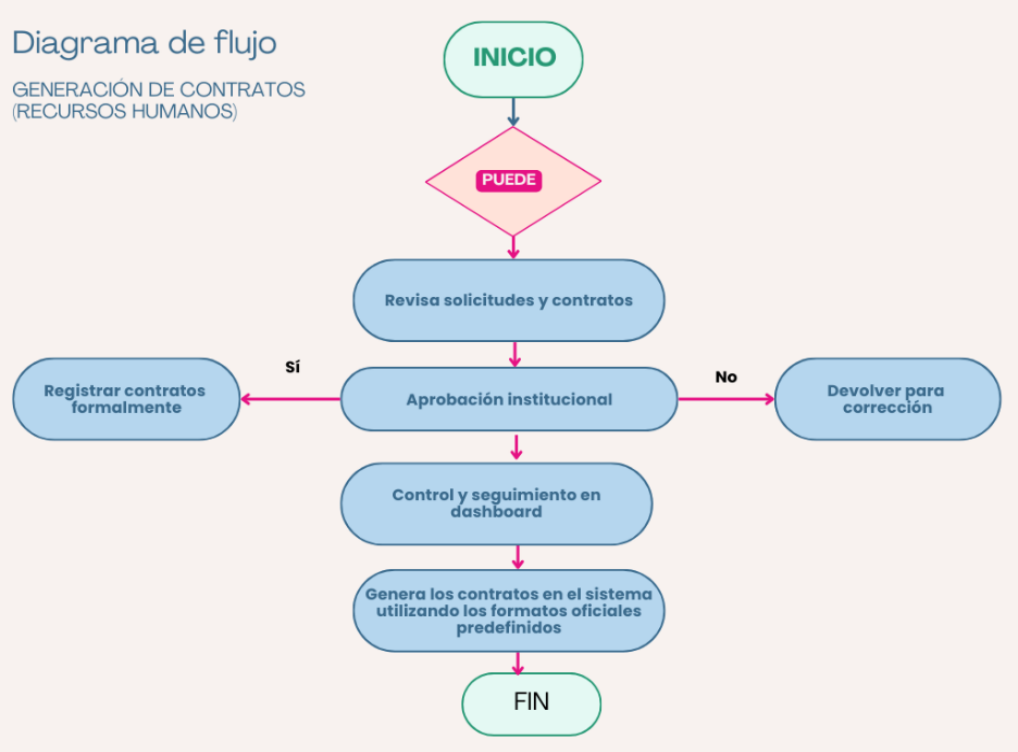


Ilustración 16:Diagrama de Flujo Generación de Contratos.

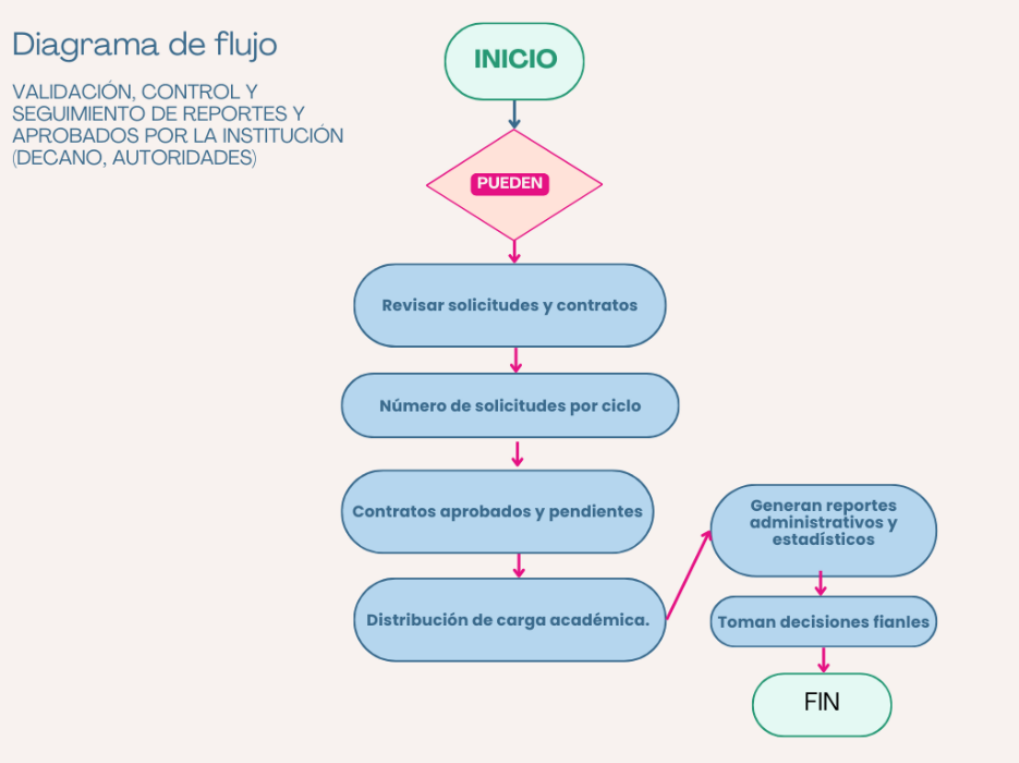


Ilustración 17: Diagrama de Flujo Validación, Control y Seguimiento de Reportes y Aprobaciones.

3.3.3 Fortalezas y Debilidad del Sistema

F: FORTALEZAS	D: DEBILIDADES
F1: Separación entre frontend, backend y datos, lo que facilita la mantenibilidad y escalabilidad. **F2:** Base de datos relacional estructurada que permite la gestión de información crítica. **F3**: Autenticación por roles, cifrado de tránsito, validación de archivos y bitácoras de auditoría. **F4**: Tratamiento de documentos en carga y descarga de documentos. **F5**: Automatización manual de procesos que antes eran manuales, reduciendo errores humanos y agilizando tramites.	**D1**. Escasa documentación técnica y manuales de usuario, lo que limita la transferencia de conocimiento y la formación de nuevos equipos de soporte. **D2**: Dependencia de confirmación de resguardo de la documentación de los procesos. **D3**: Riesgo de dependencia tecnológica, ya que no existen planes de contingencia para fallas o costos futuros.
O: OPORTUNIDADES	**A: AMENAZAS**
O1: Adopción de institucional de prácticas de aseguramiento de calidad. **O2**: Escalabilidad hacia entornos cloud como AWS, Azure o GCP que permitirá soportar mayores números de usuarios. **O3**: Integración con otros sistemas universitarios generando un ecosistema más eficiente y centralizado. **O4**: Potencial de innovación al incorporar analítica de datos sobre contratación docente.	**A1**: Ciberataques y vulnerabilidades comunes en aplicaciones web (SQL Injection, XSS, CRSF). **A2**: Inestabilidad en los servicios de la gestión de documentos. **A3**: Resistencia al cambio por parte de los usuarios. **A4**: Obsolescencia tecnológica si no se actualizan versiones de frameworks, librerías y dependencias de seguridad.

Tabla 23 FODA del Sistema de Contratación de Docentes.

3.3.4 Diagrama de Arquitectura

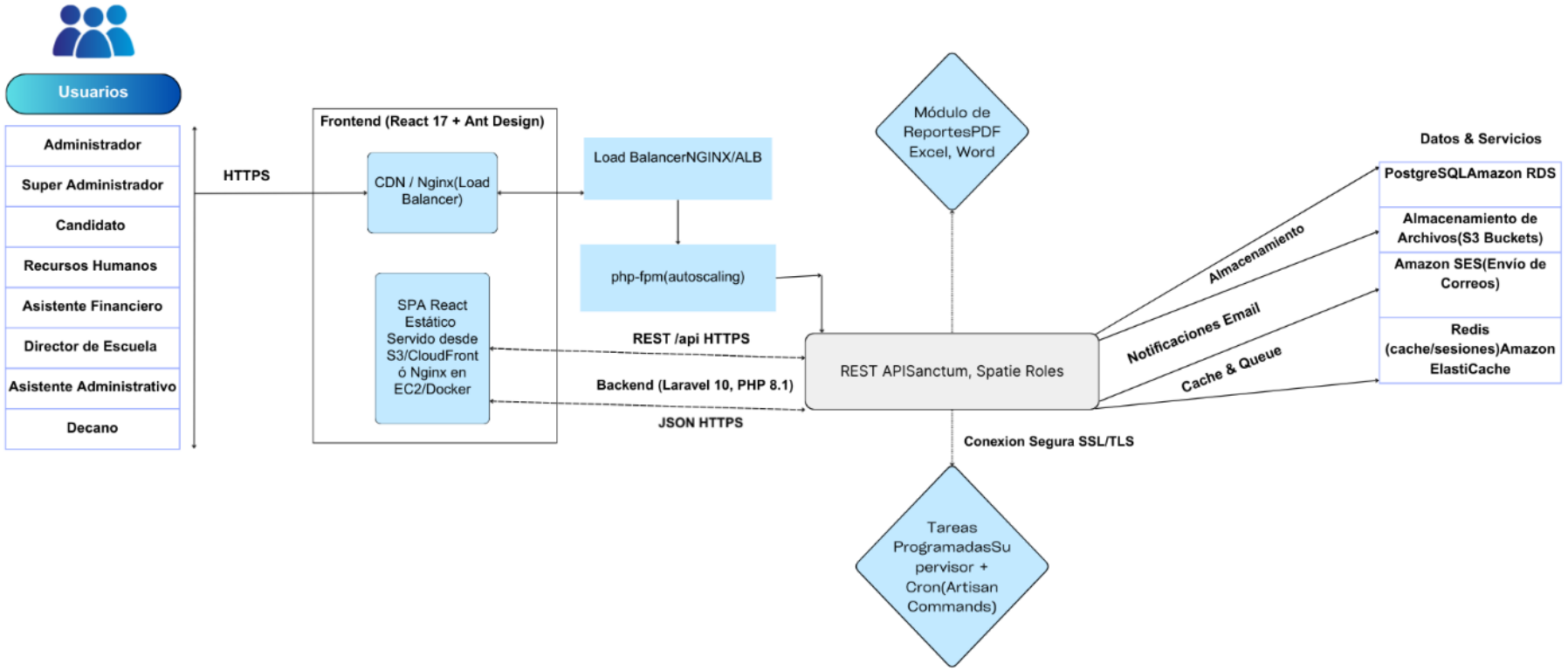


Ilustración 18 Diagrama de Arquitectura del Sistema de Contratación de Docentes FIA UES.

Diagrama de Enfoque de Sistemas

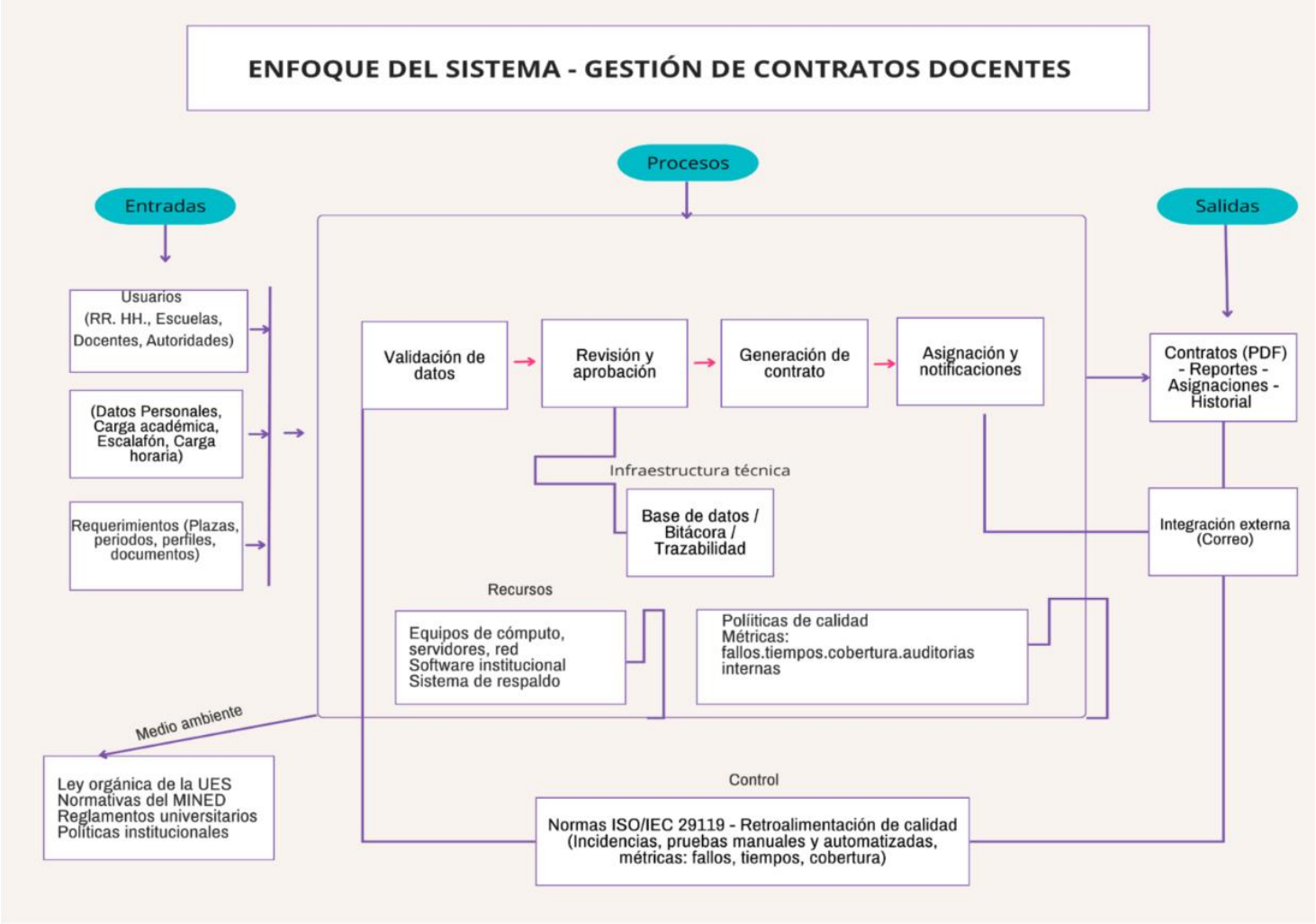


Ilustración 19 Diagrama Enfoque de Sistemas - Sistema Contratación de Docentes FIA UES..

Diagrama de Flujos de Procesos

Gestionar Bancos

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

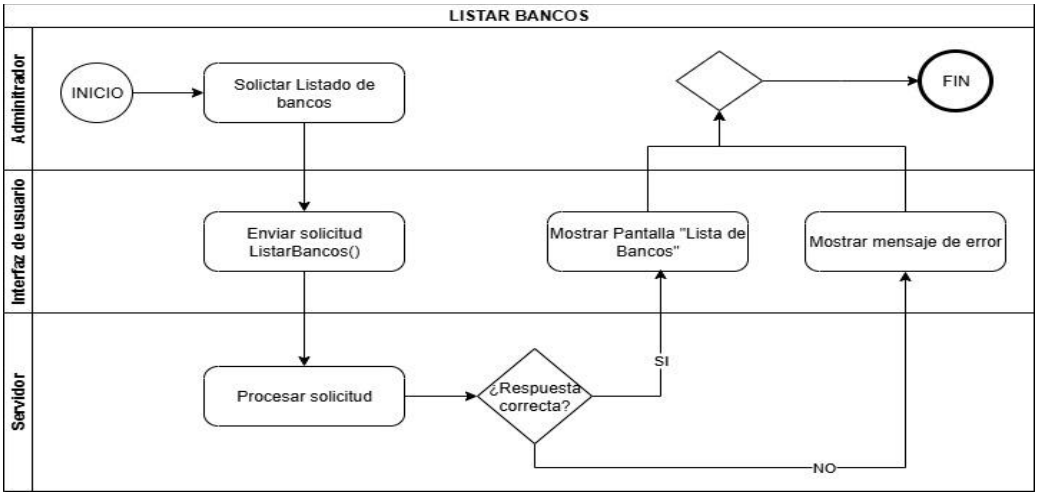


Ilustración 20 Flujo de listar Bancos.

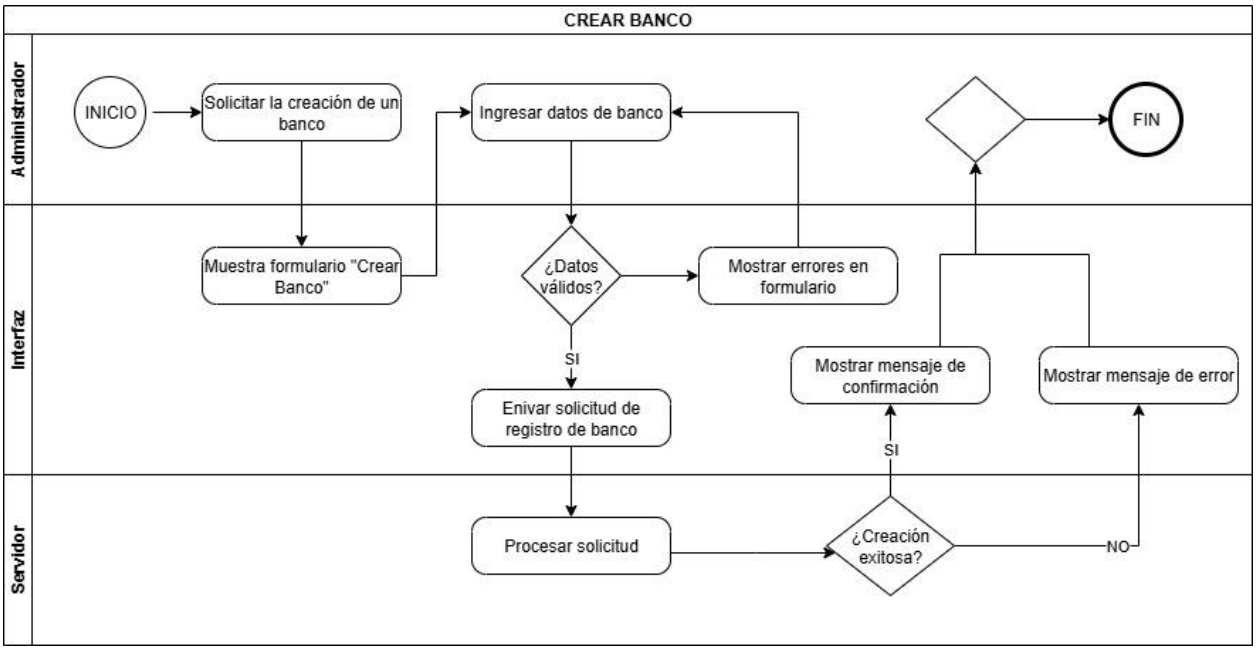


Ilustración 21 Flujo Crear Bancos.

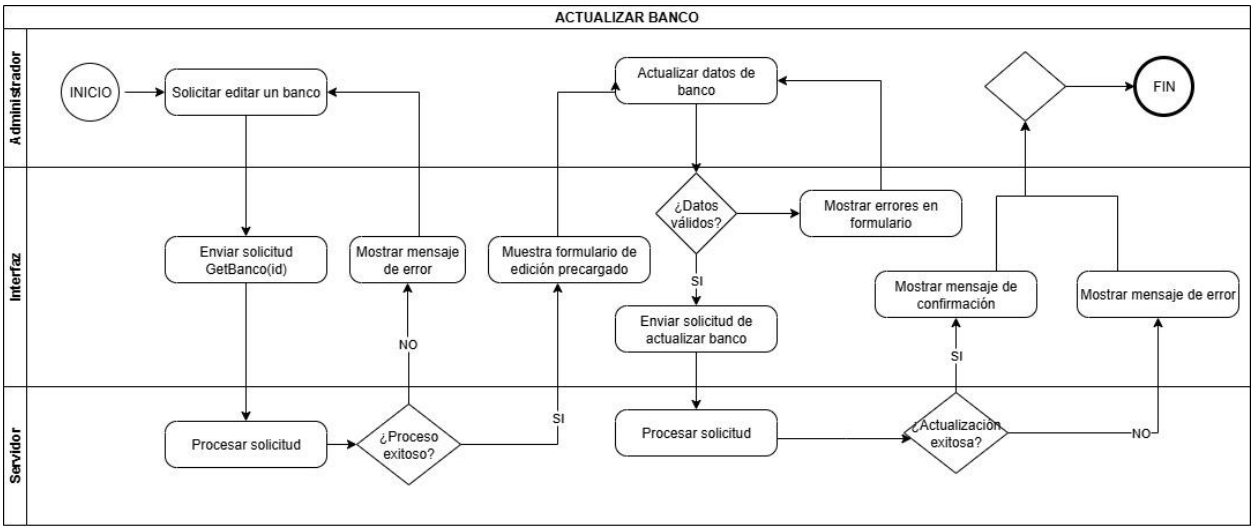


Ilustración 22 Flujo Actualizar Bancos.

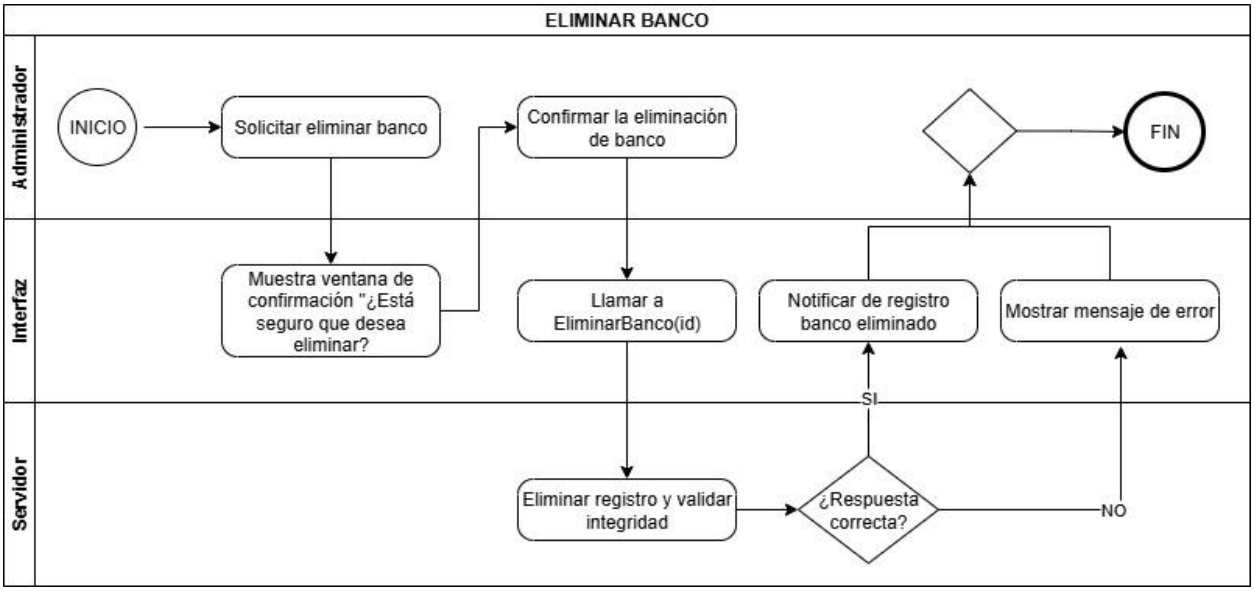


Ilustración 23 Flujo Eliminar Bancos.

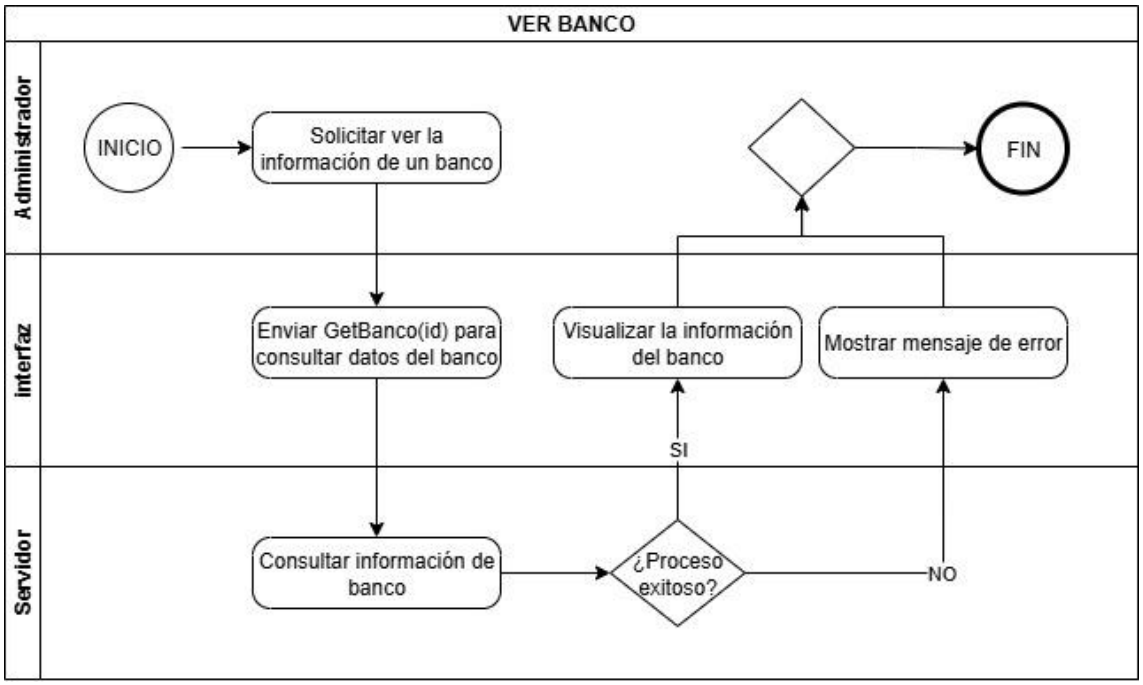


Ilustración 24 Flujo Ver Banco.

Gestionar Usuarios

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

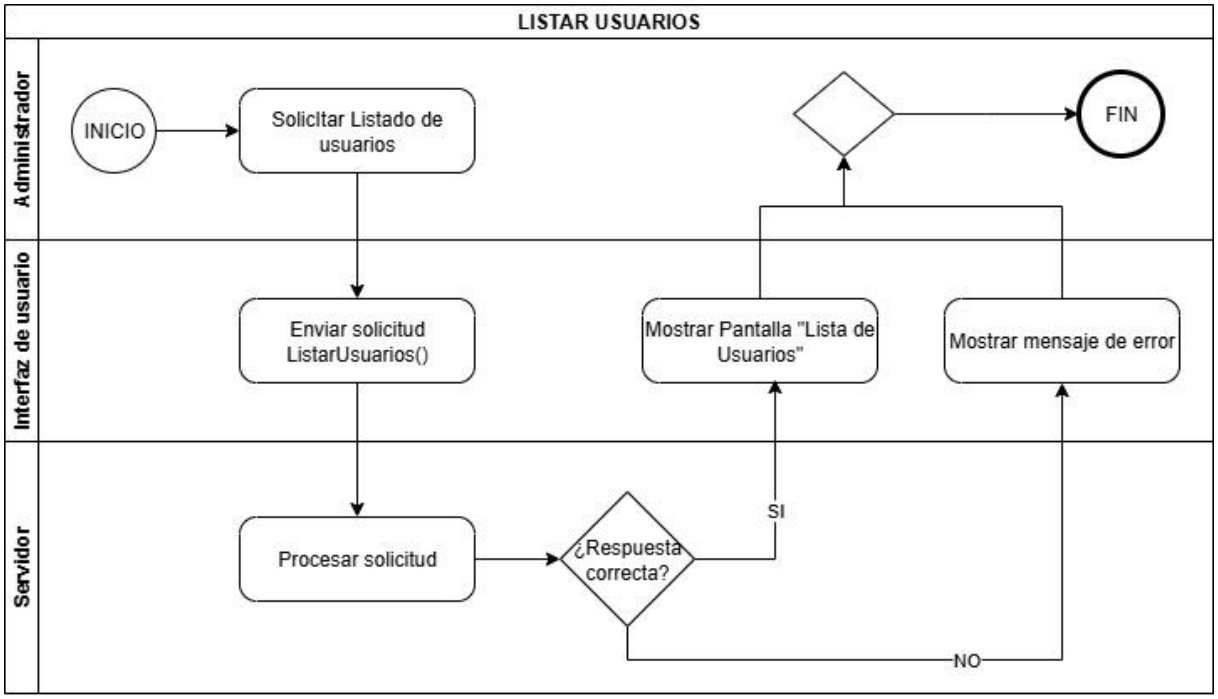


Ilustración 25 Flujo Listar Usuarios.

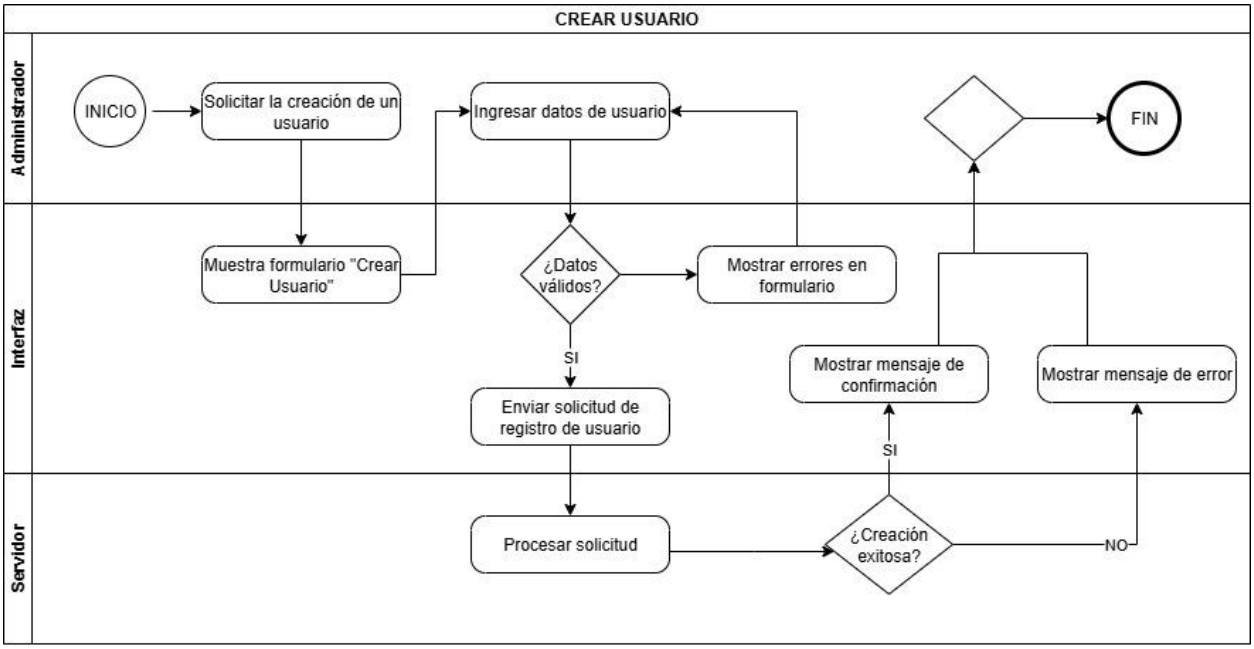


Ilustración 26 Flujo Crear Usuario.

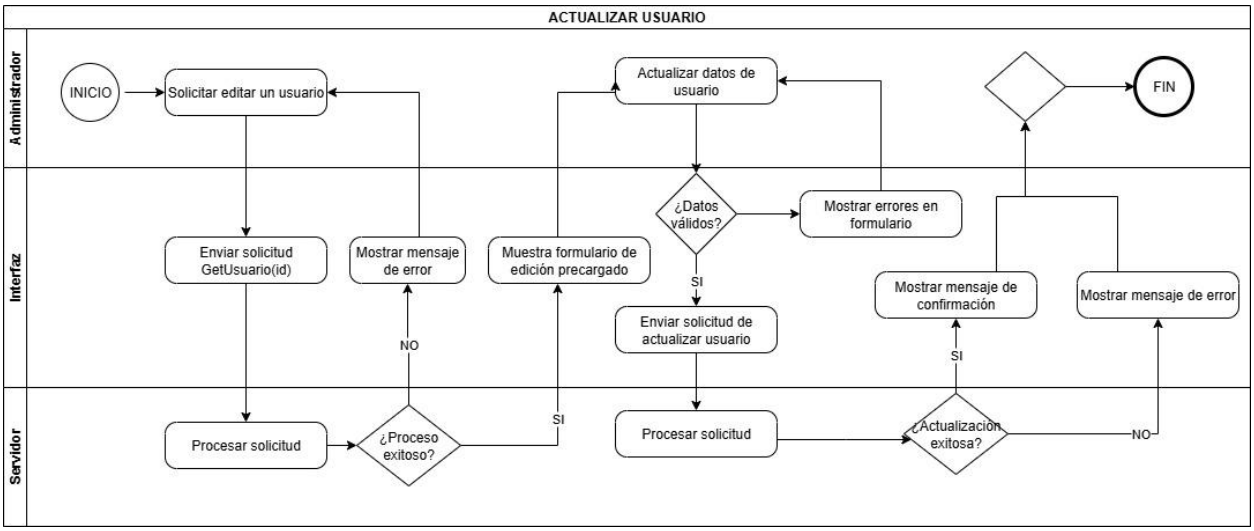


Ilustración 27 Flujo Actualizar Usuario.

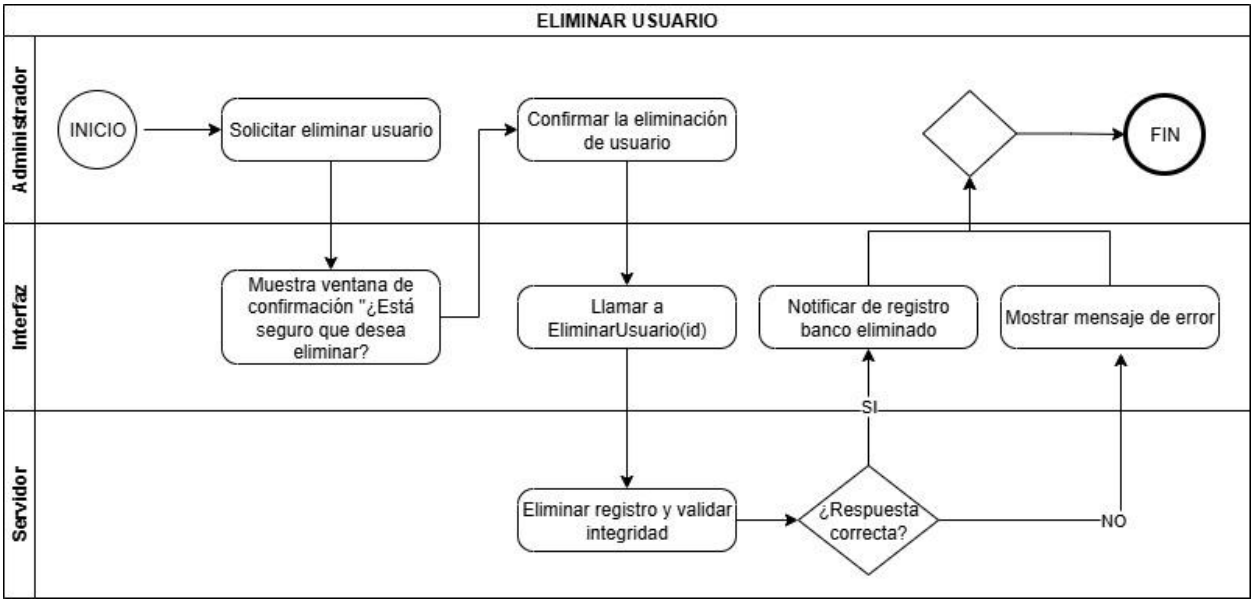


Ilustración 28 Flujo Eliminar Usuario.

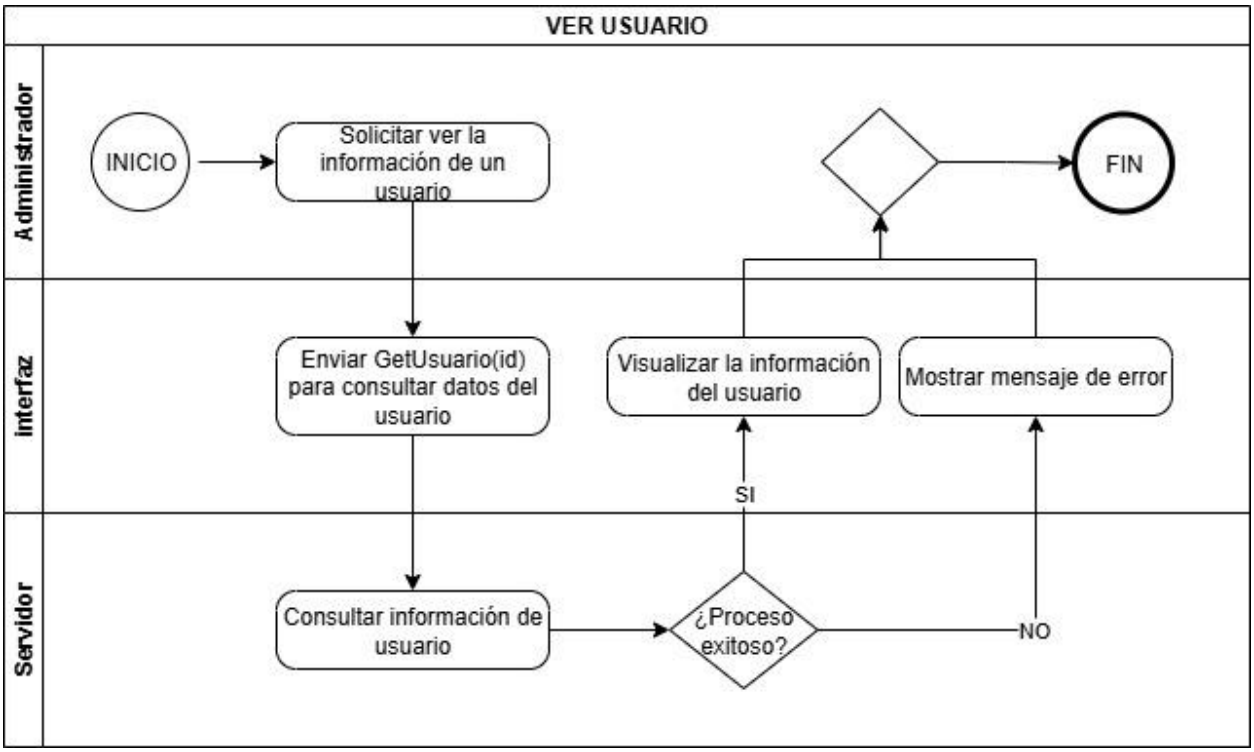


Ilustración 29 Flujo Ver Usuario.

Gestión Ciclos

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

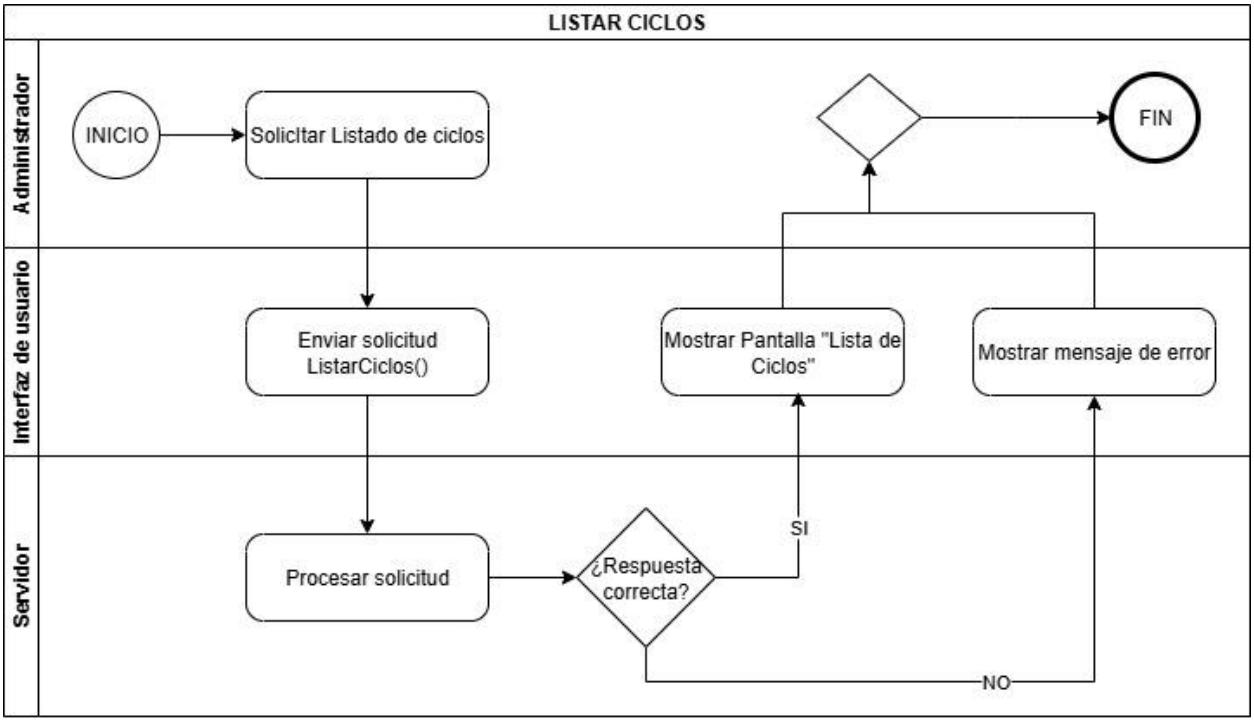


Ilustración 30 Flujo Listar Ciclos.

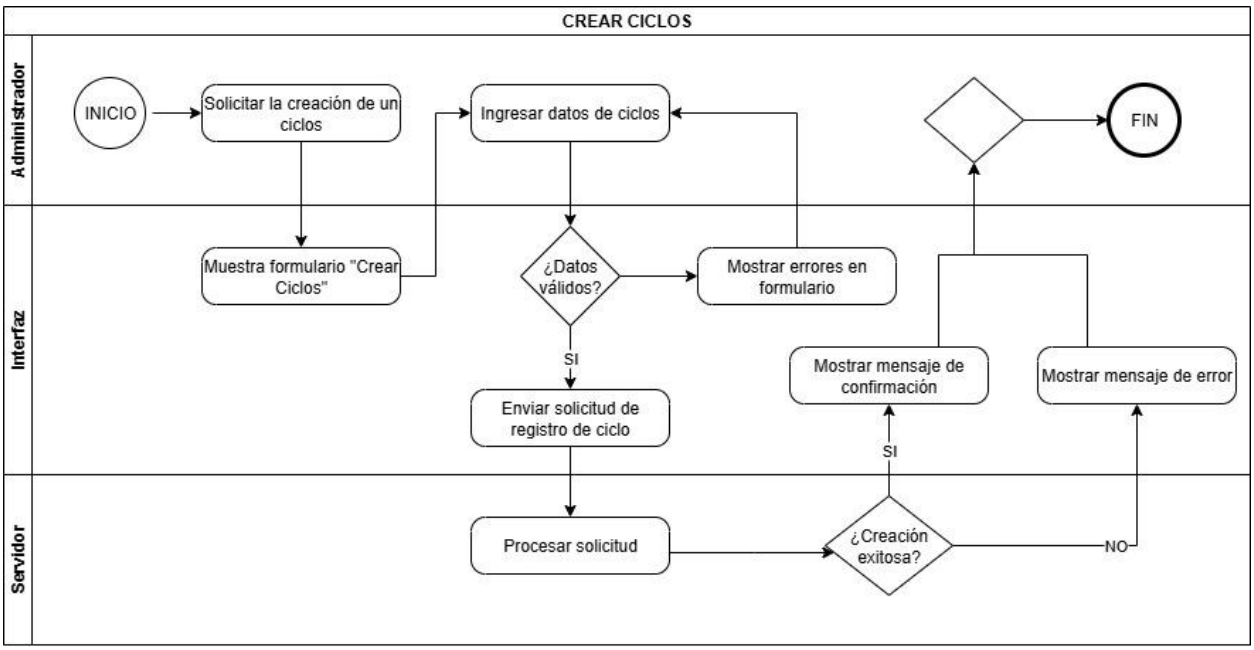


Ilustración 31 Flujo Crear Ciclos.

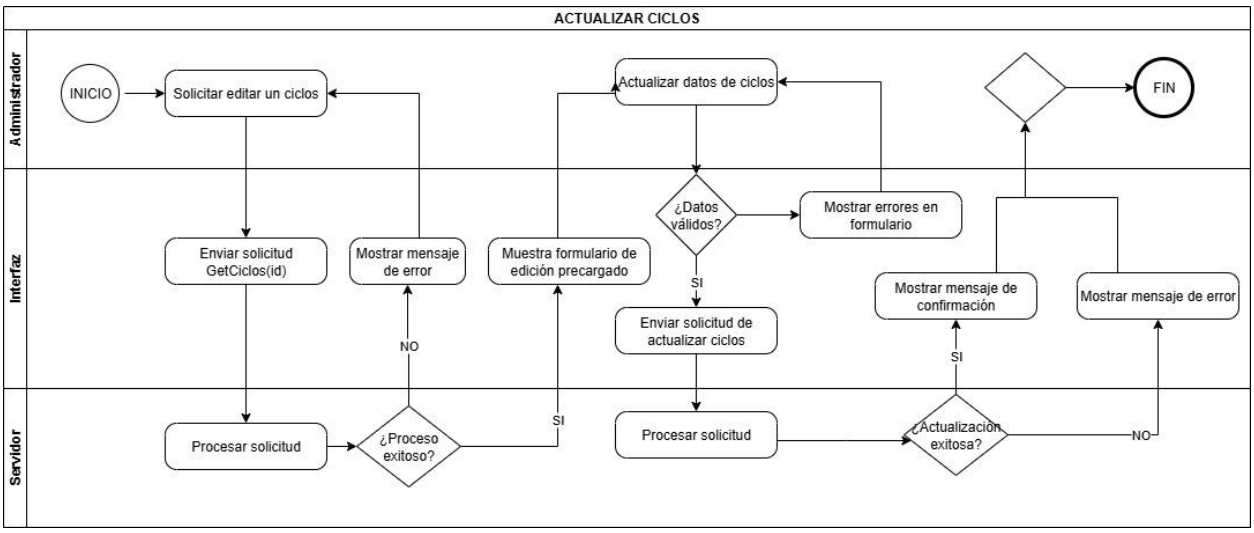


Ilustración 32 Flujo Actualización Ciclos.

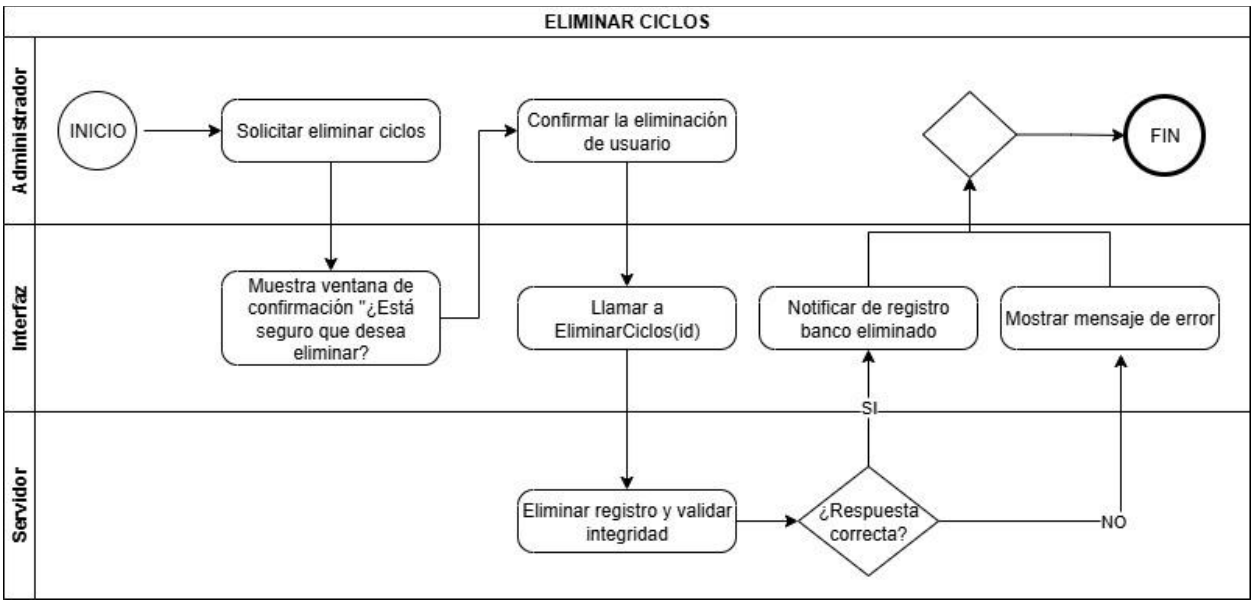


Ilustración 33 Flujo Eliminar Ciclos.

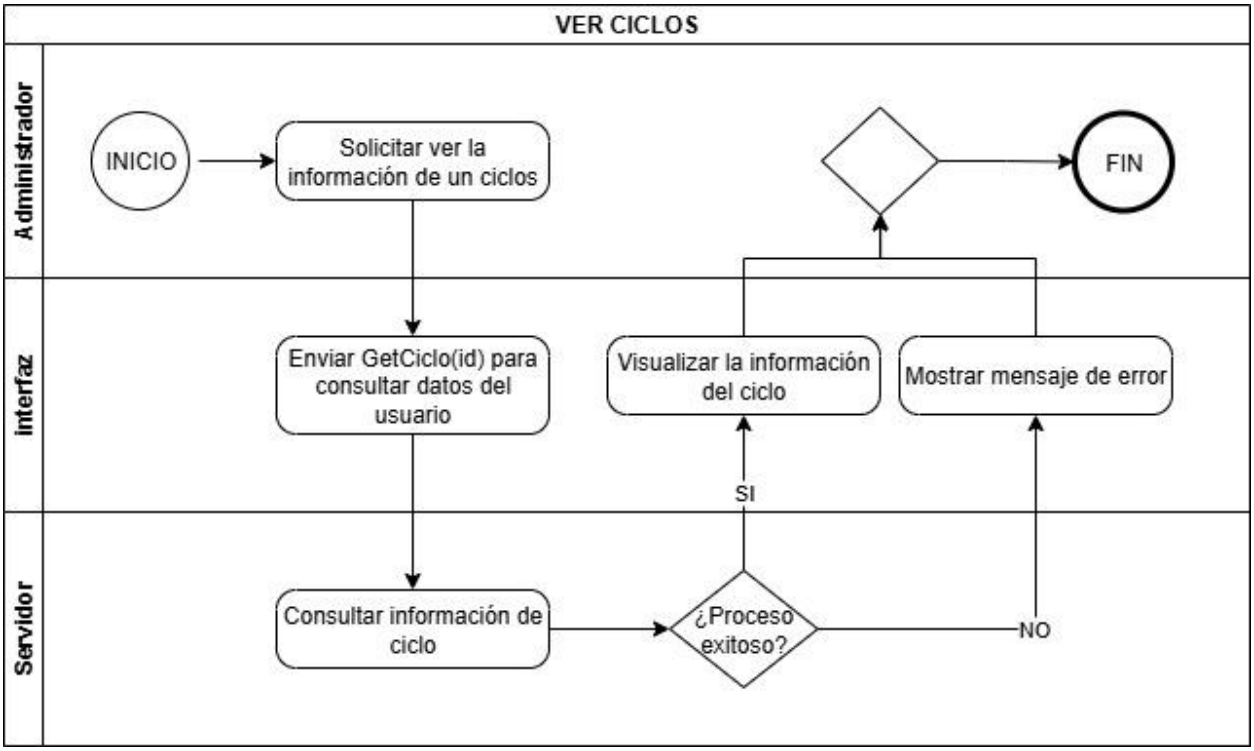


Ilustración 34 Flujo Ver Ciclo.

Gestión Escalafones

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

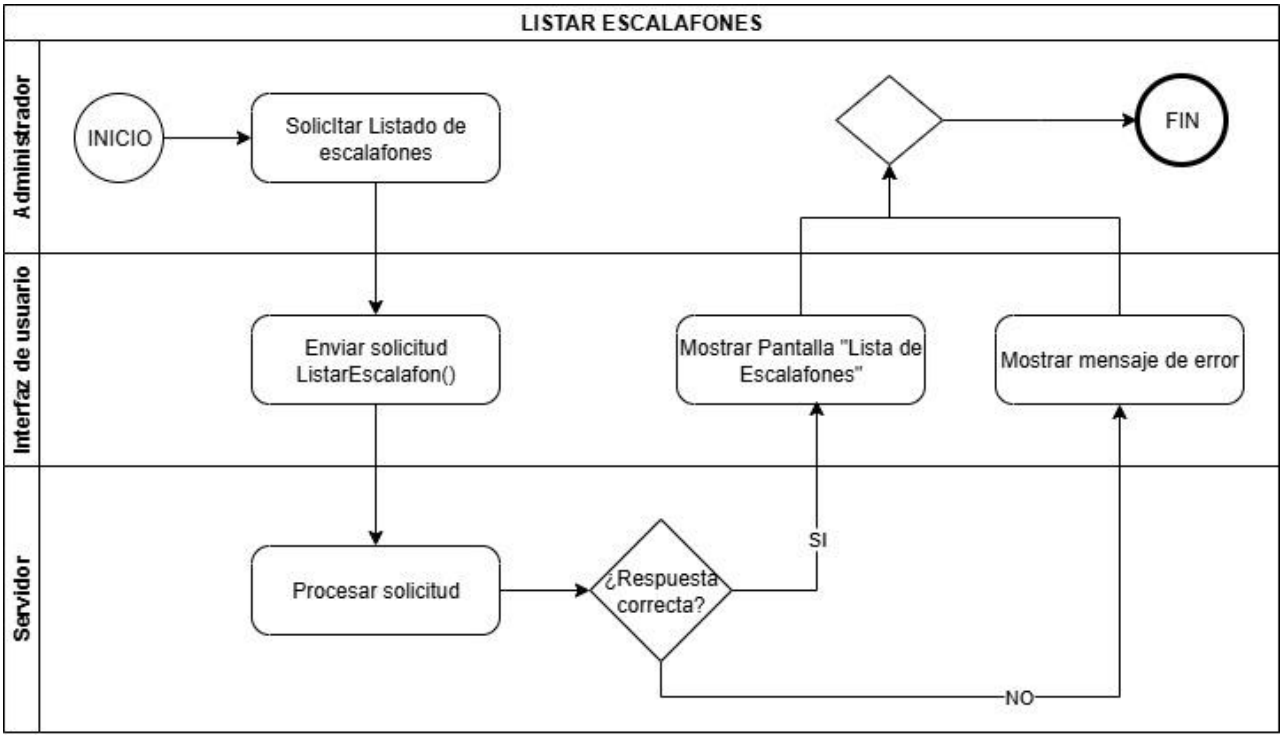


Ilustración 35 Flujo Listar Escalafones.

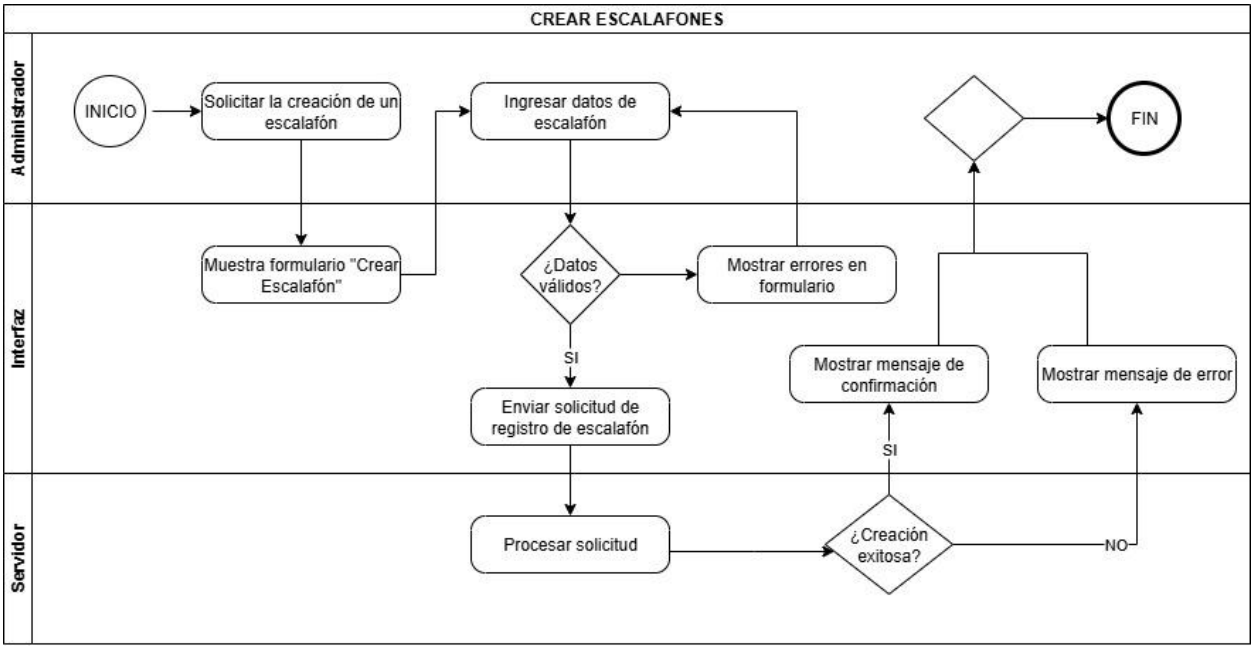


Ilustración 36 Flujo Crear Escalafones.

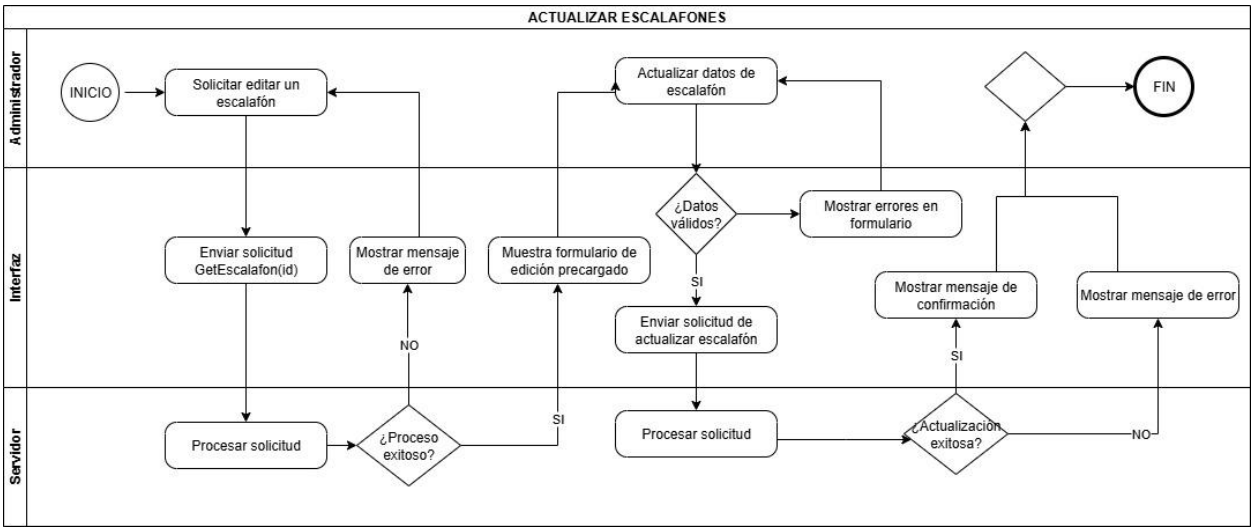


Ilustración 37 Flujo Actualizar Escalafones.

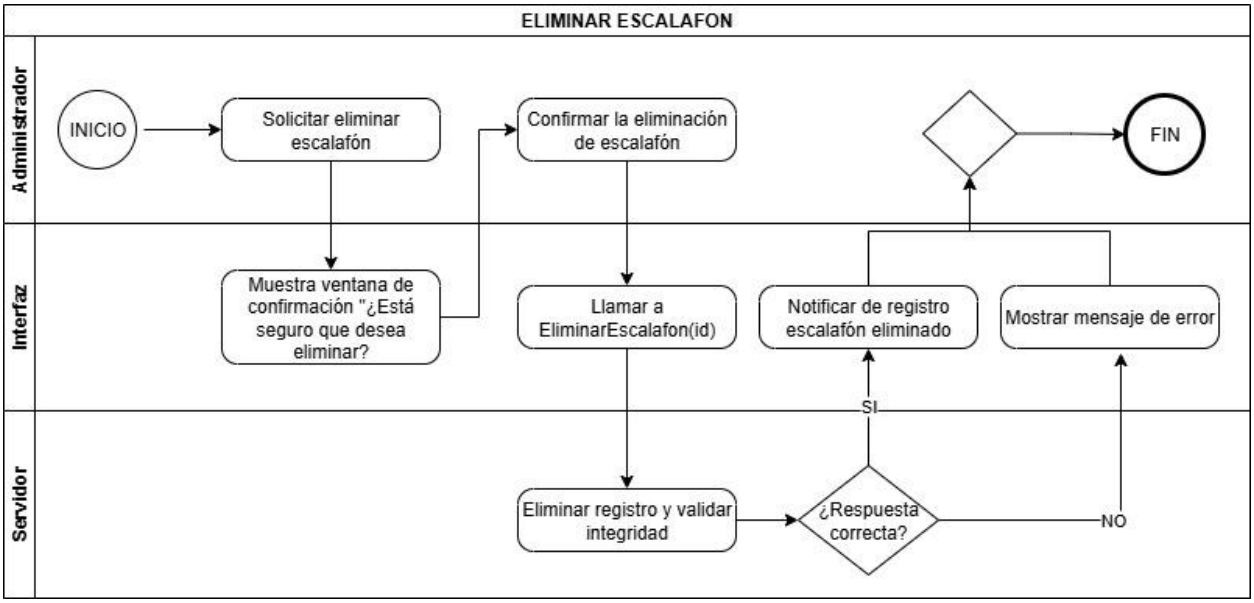


Ilustración 38 Flujo Eliminar Escalafón.

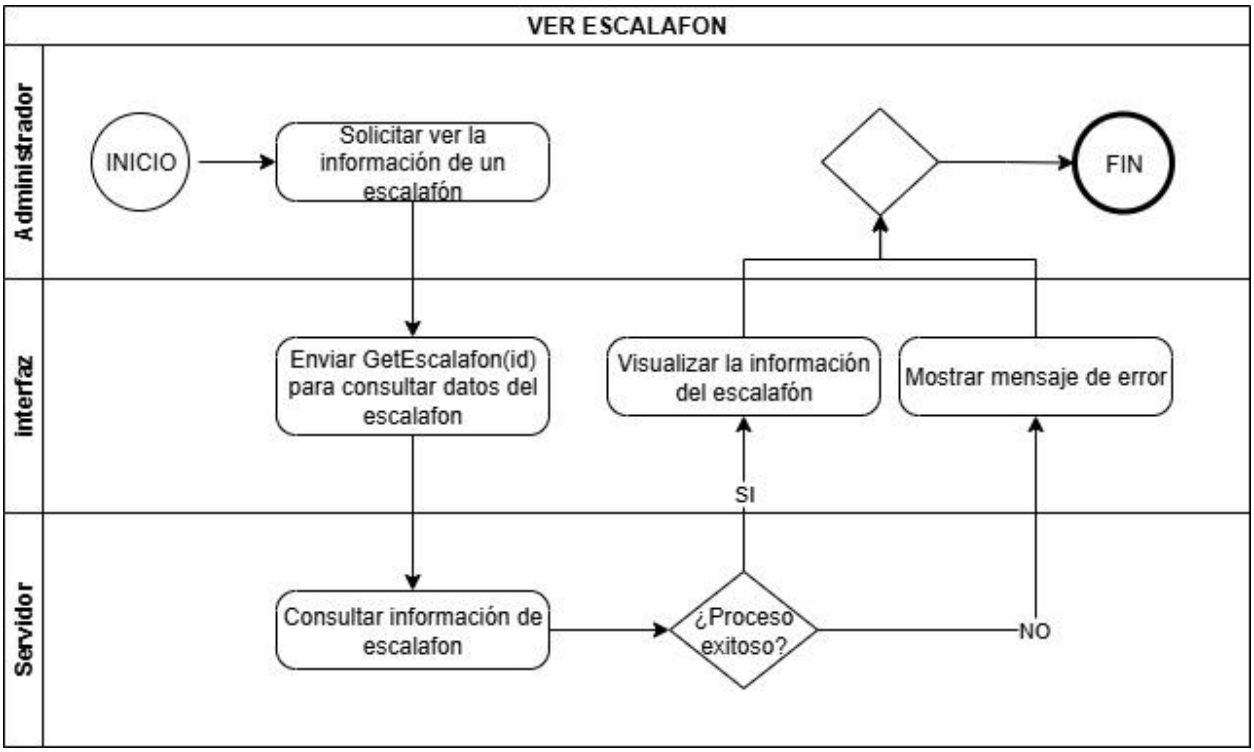


Ilustración 39 Flujo Ver Escalafón.

Gestión Facultades

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

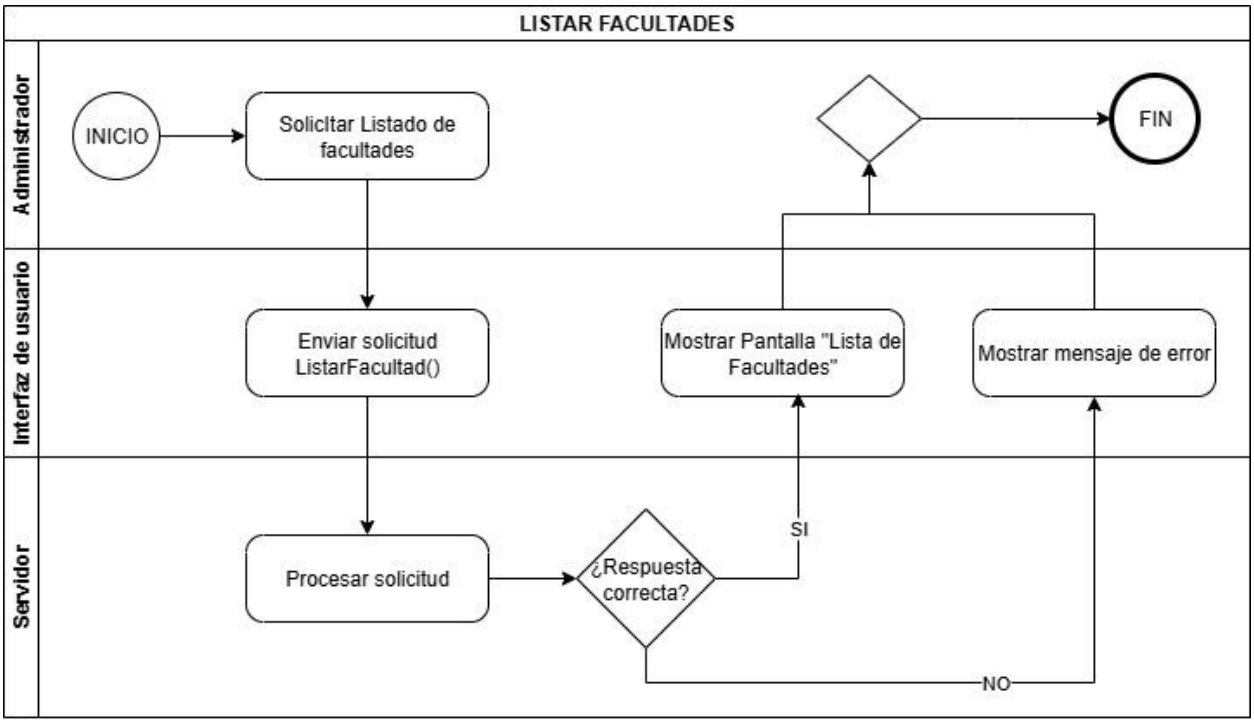


Ilustración 40 Flujo Listar Facultades

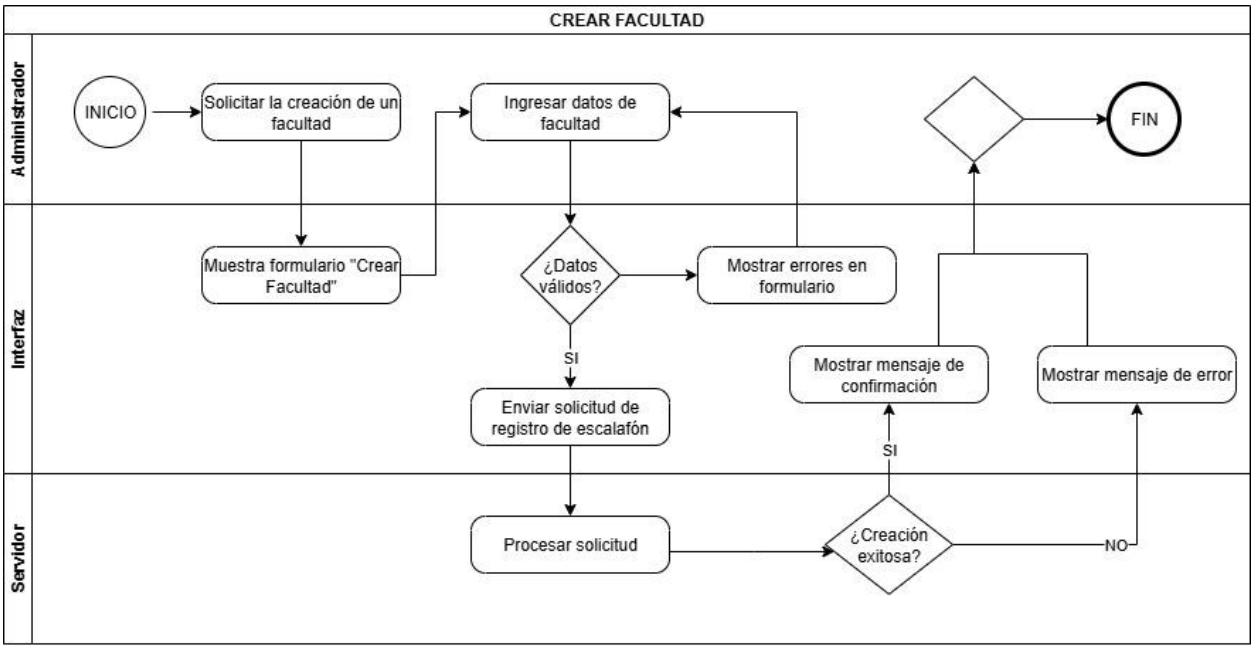


Ilustración 41 Flujo Crear Facultad.

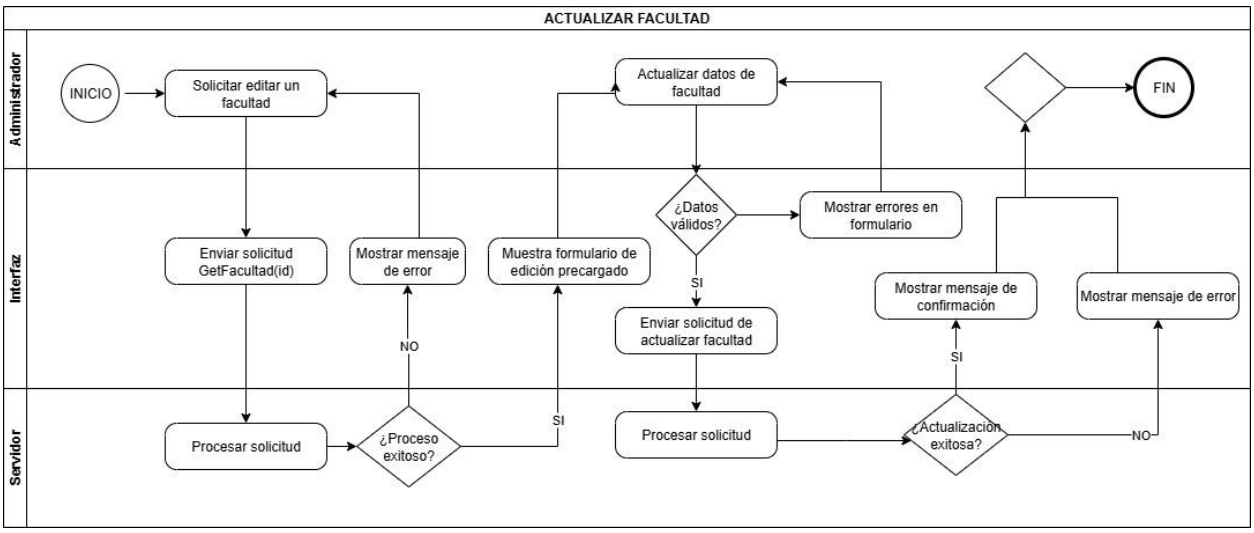


Ilustración 42 Flujo Actualizar Facultad.

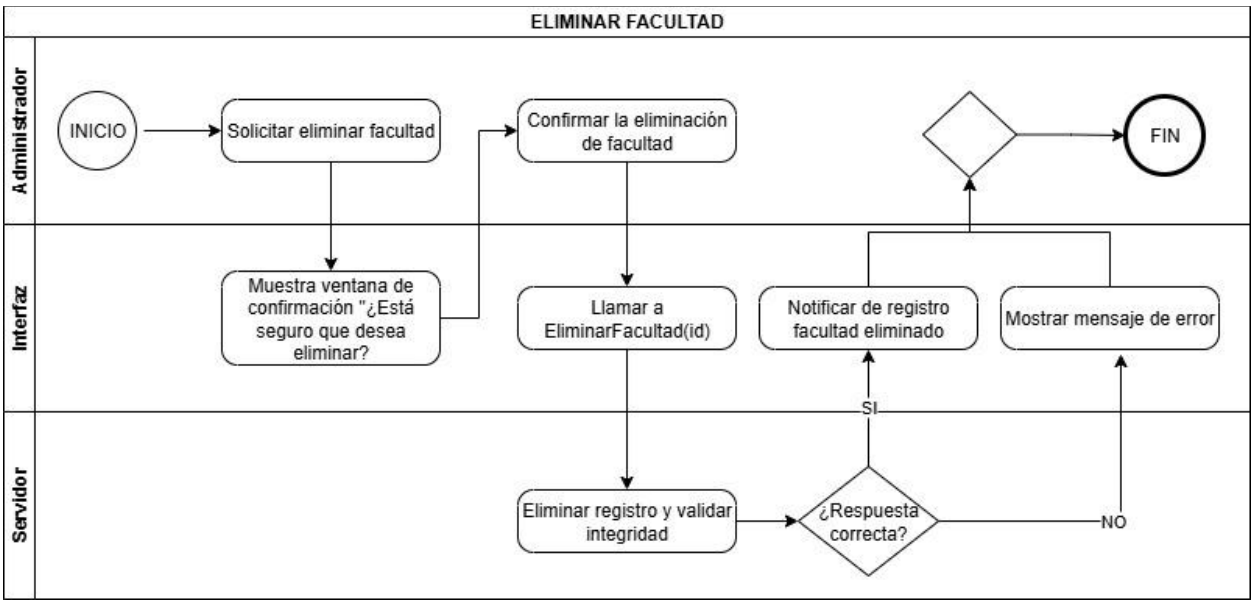


Ilustración 43 Flujo Eliminar Facultad.

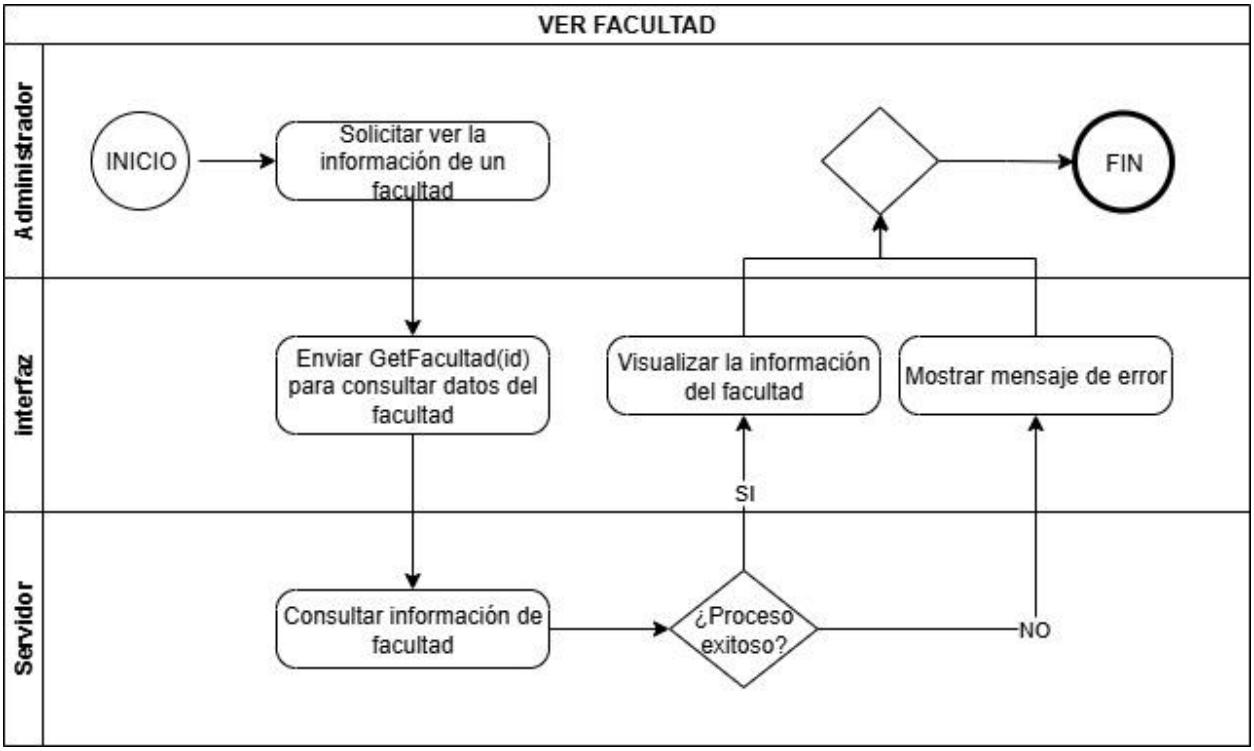


Ilustración 44 Flujo Ver Facultad.

Gestión Escuelas

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

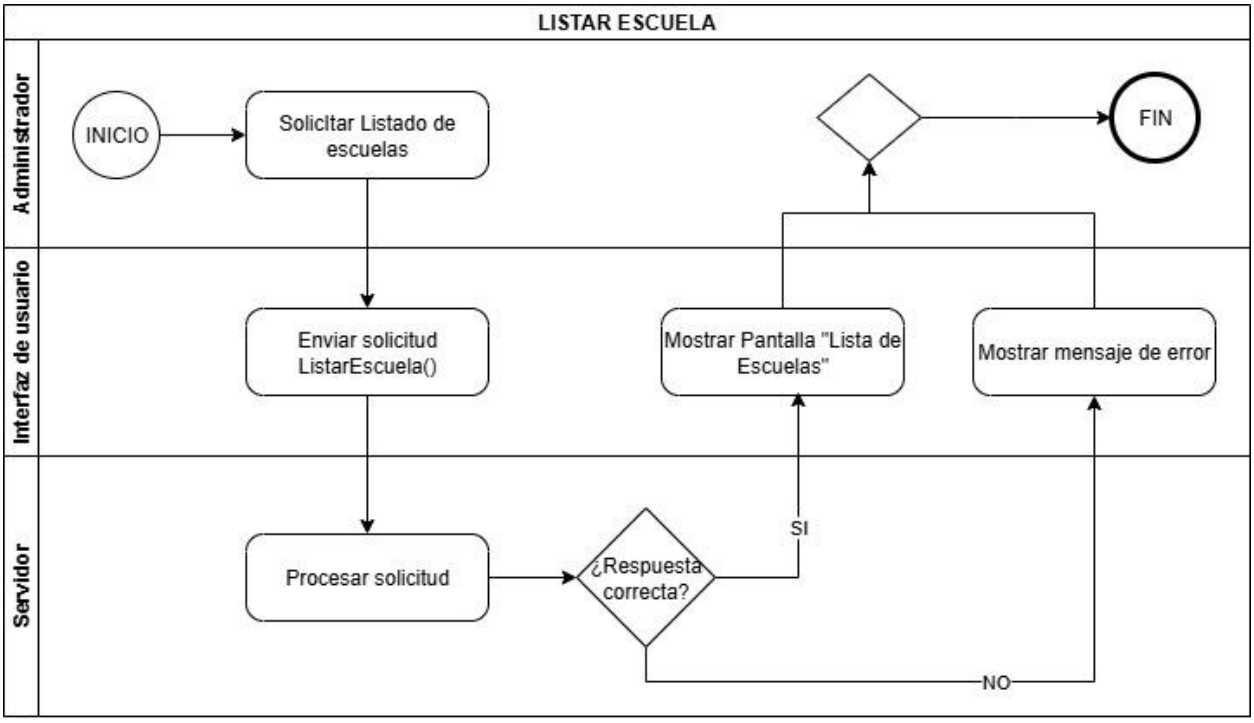


Ilustración 45 Flujo Listar Escuela.

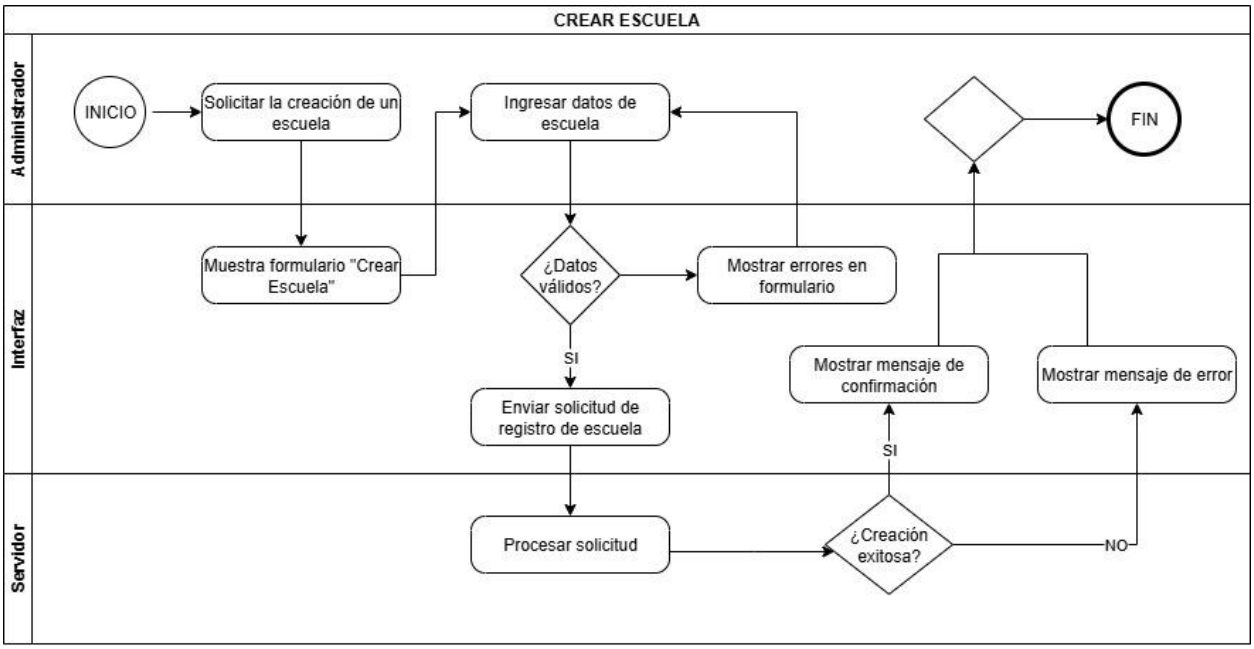


Ilustración 46 Flujo Crear Escuela.

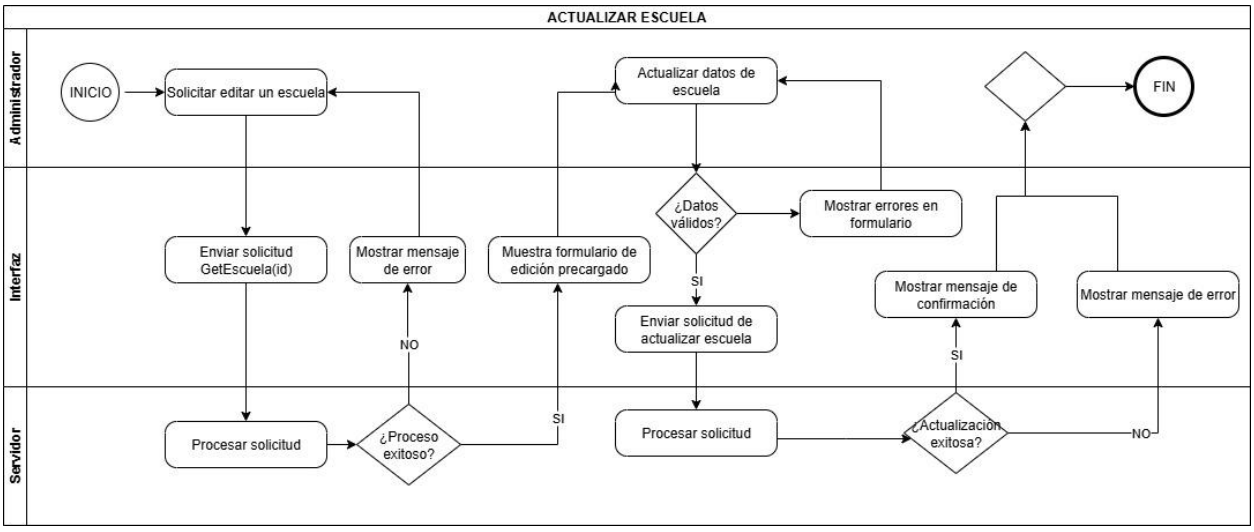


Ilustración 47 Flujo Actualizar Escuela.

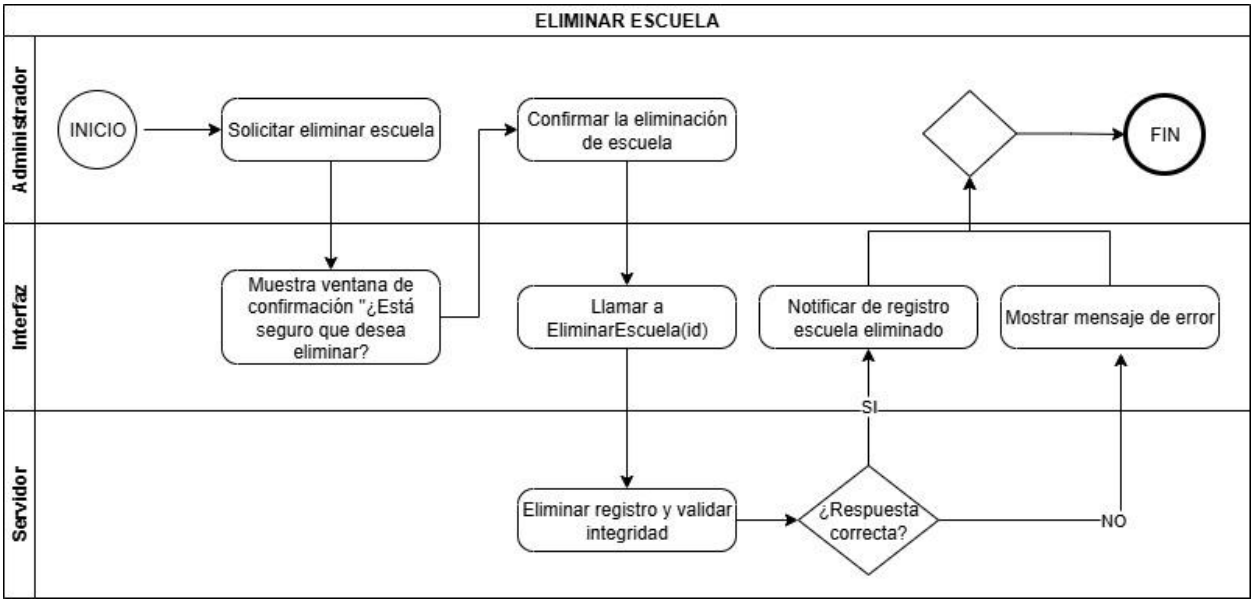


Ilustración 48 Flujo Eliminar Escuela.

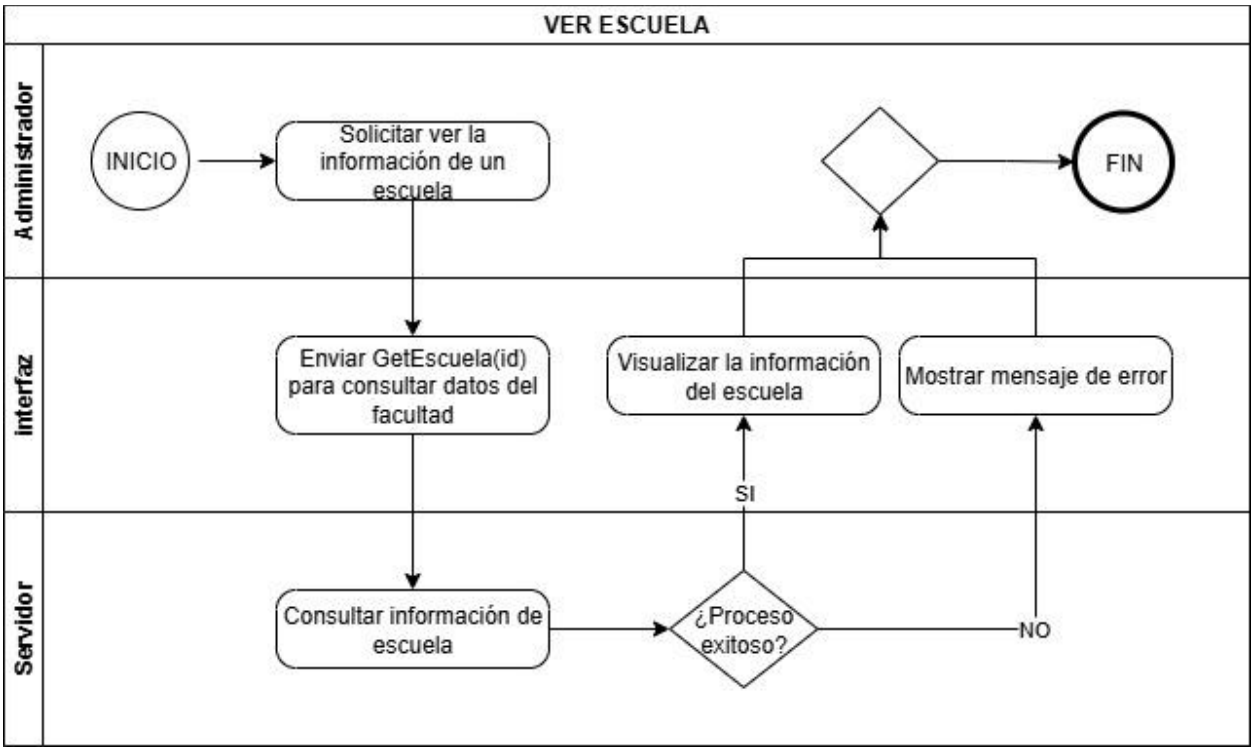


Ilustración 49 Flujo Ver Escuela.

Gestión Materias

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

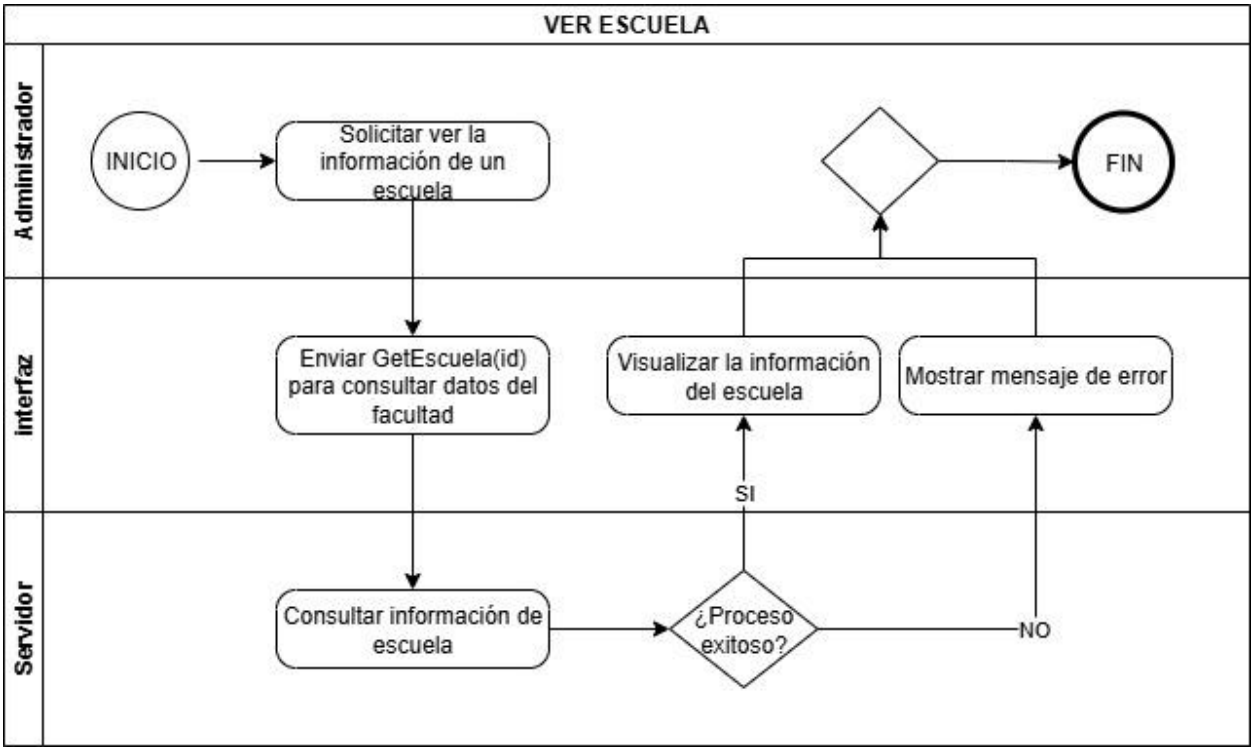


Ilustración 50 : Flujo Ver Escuela.

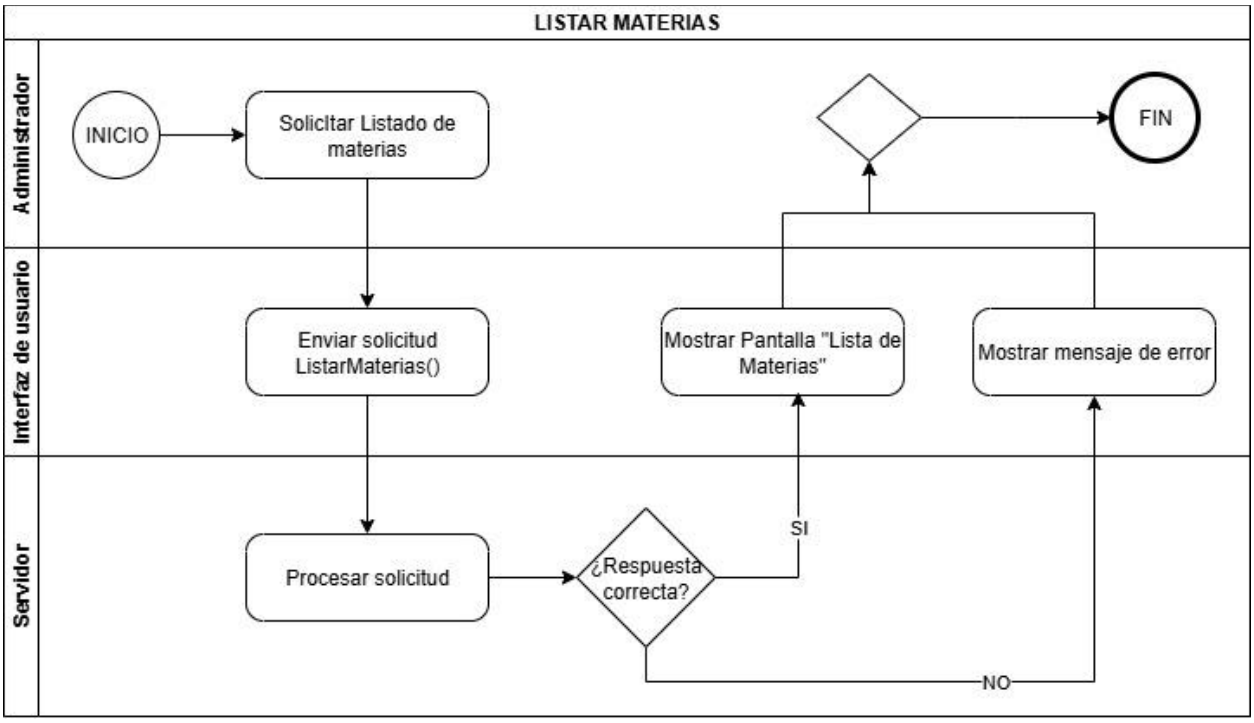


Ilustración 51 Flujo Listar Materias.

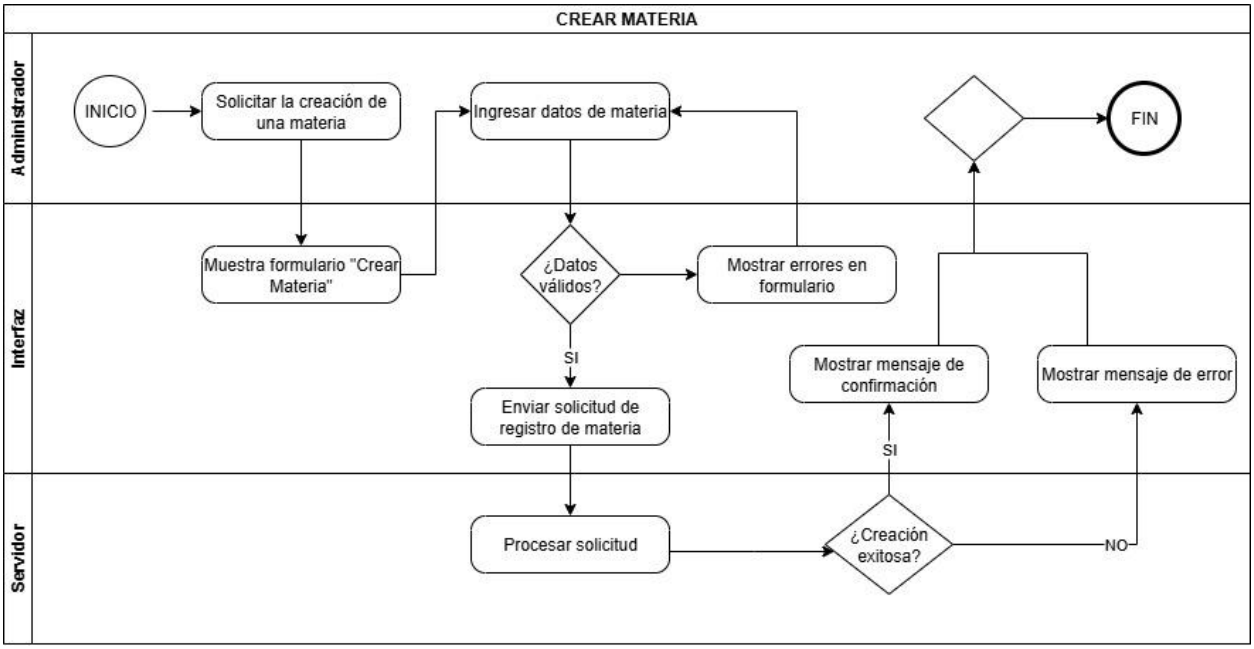


Ilustración 52 Flujo Crear Materia.

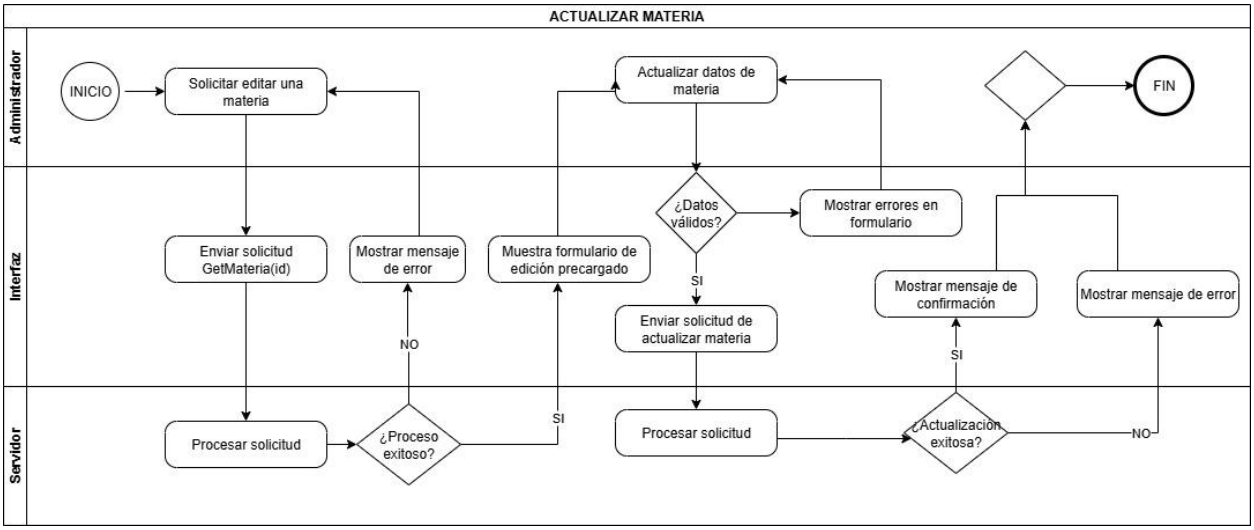


Ilustración 53 Flujo Actualizar Materia.

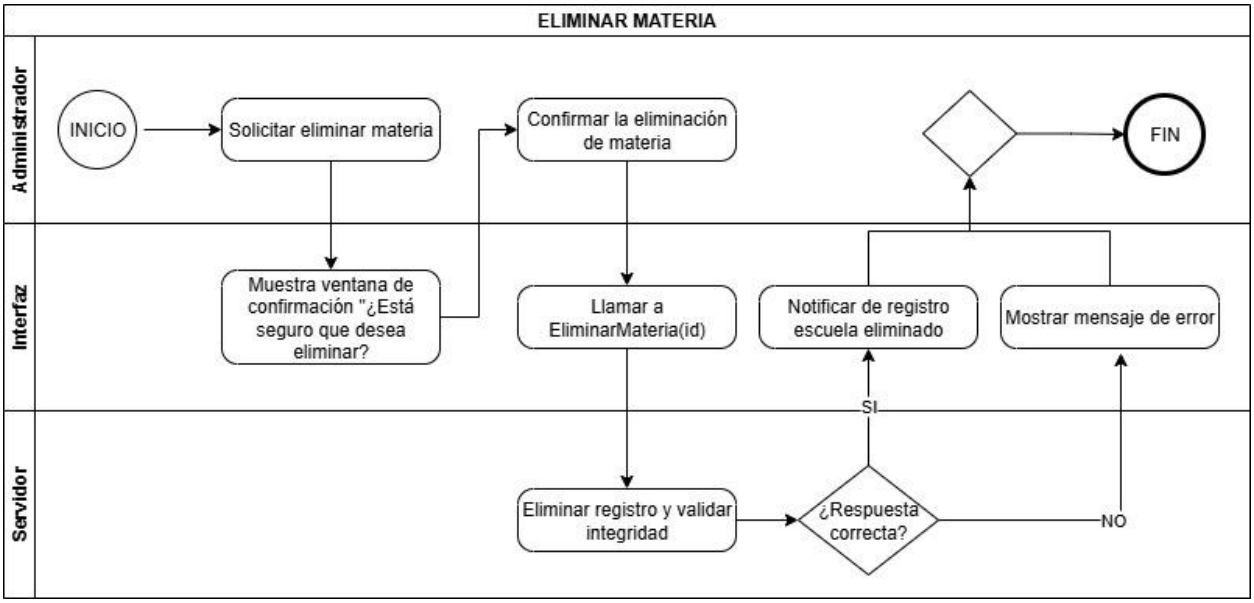


Ilustración 54 Flujo Eliminar Materia.

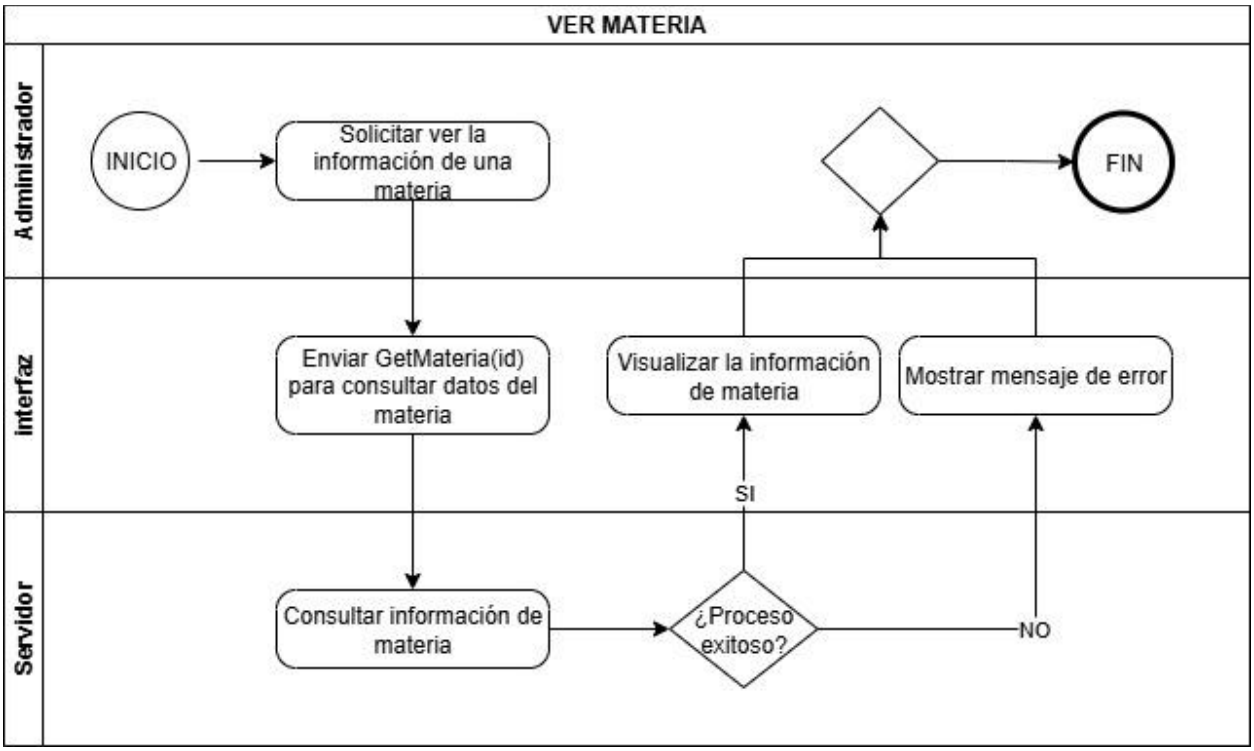


Ilustración 55 : Flujo Ver Materia.

Gestión Actividades Docentes.

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

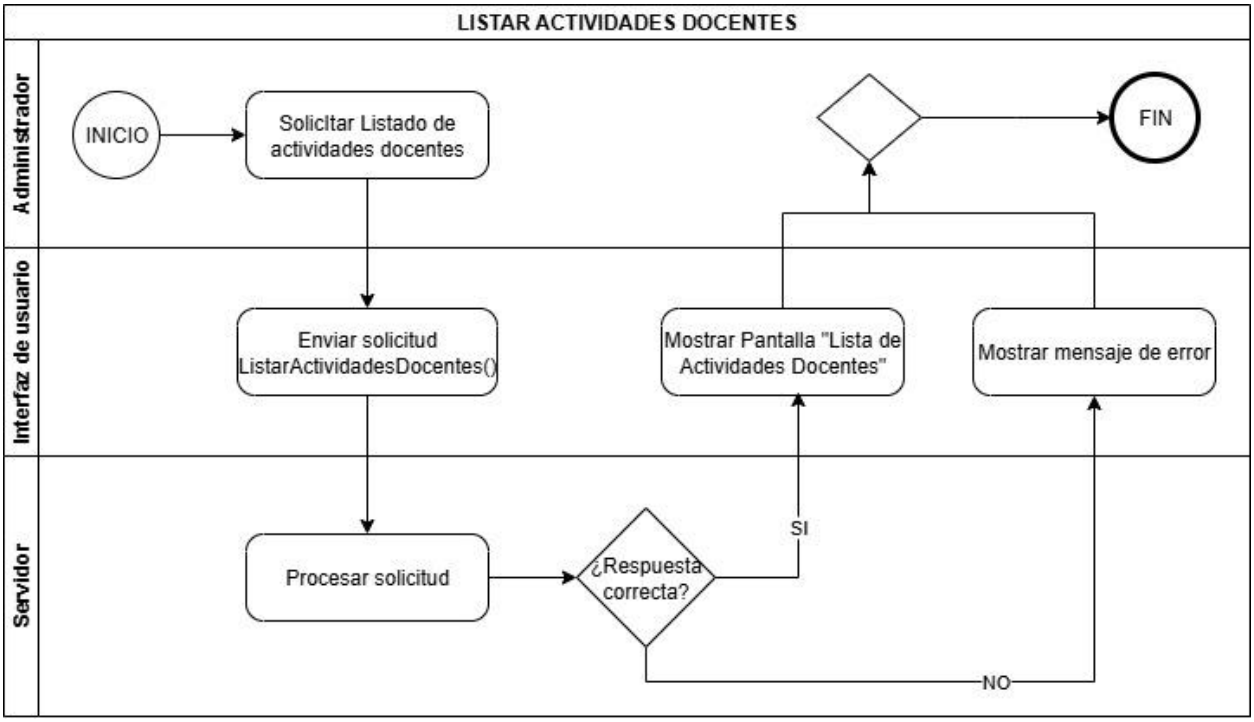


Ilustración 56 Flujo Listar Actividades Docentes.

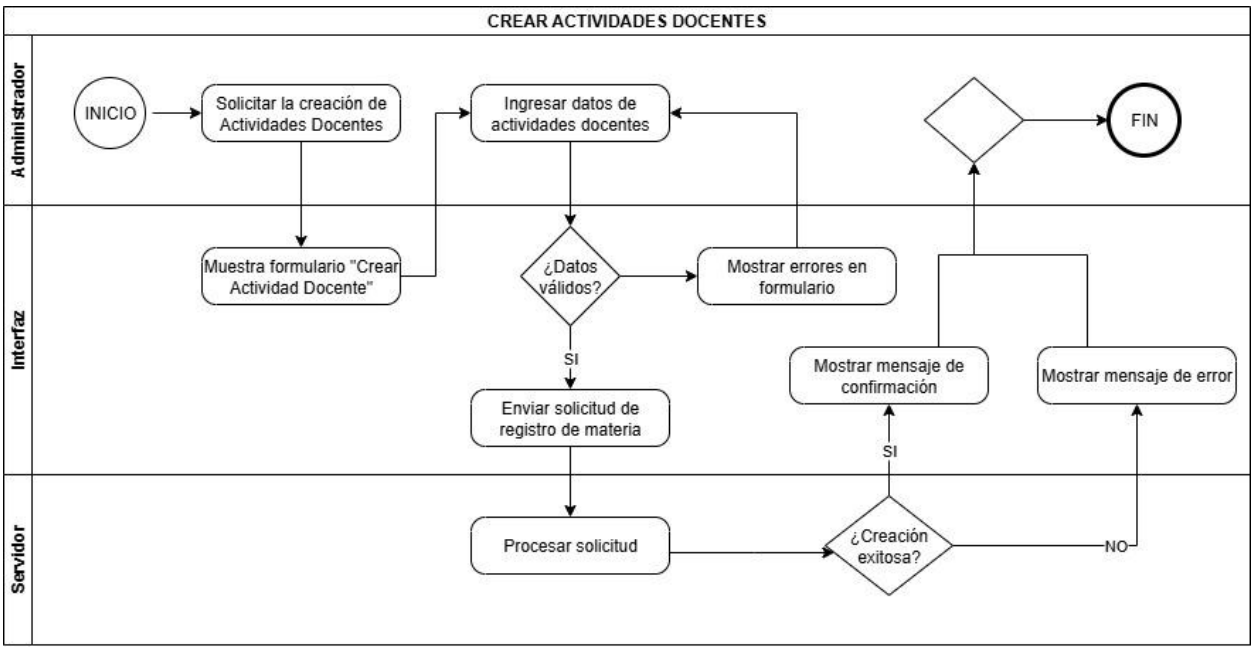


Ilustración 57 Flujo Crear Actividades Docentes.

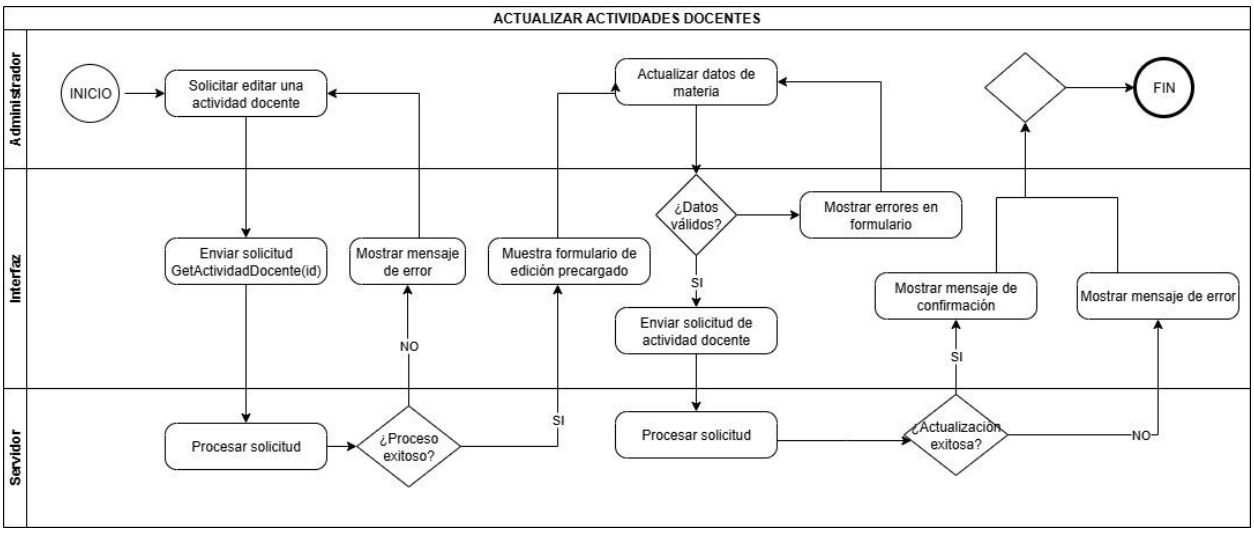


Ilustración 58 Flujo Actualizar Actividades Docentes.

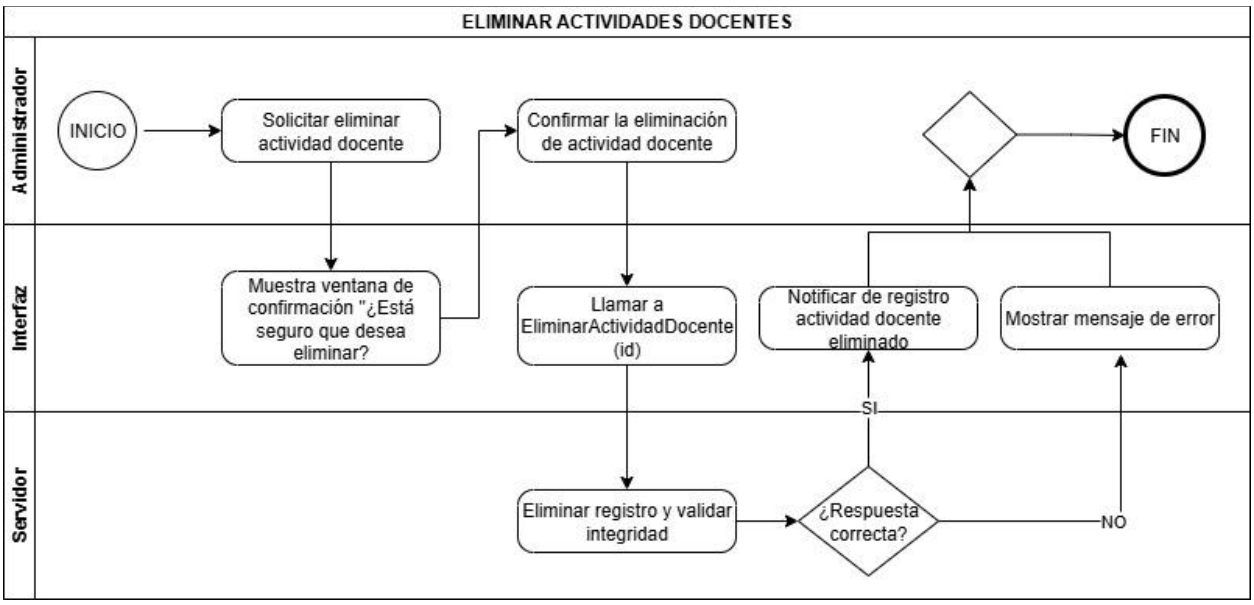


Ilustración 59 Flujo Eliminar Actividades Docentes.

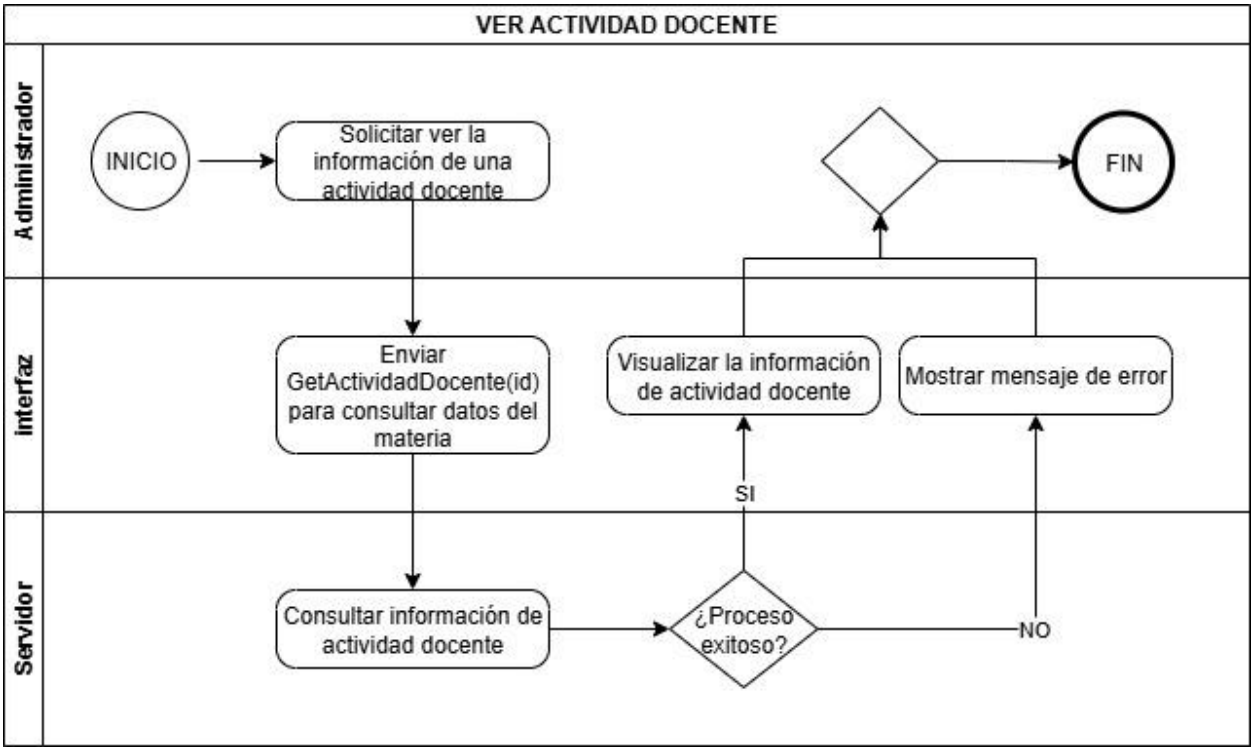


Ilustración 60 Flujo Ver Actividad Docente.

Gestión Cargos

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

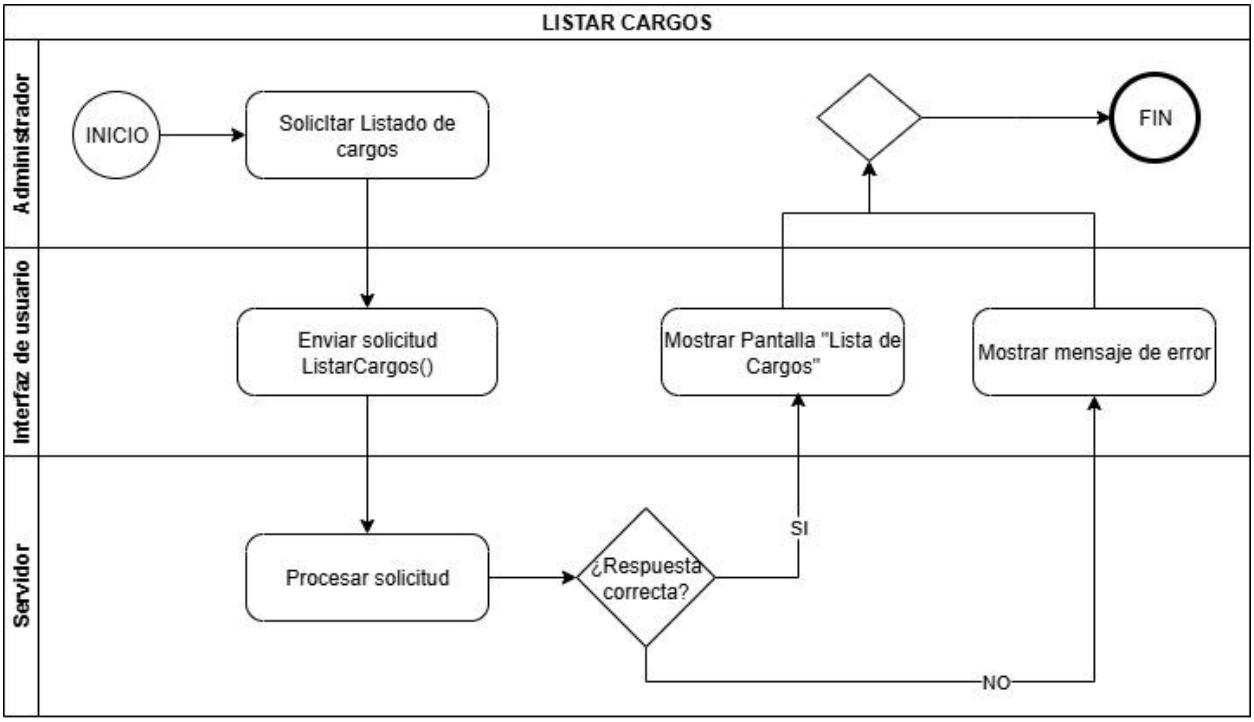


Ilustración 61 Flujo Listar Cargos.

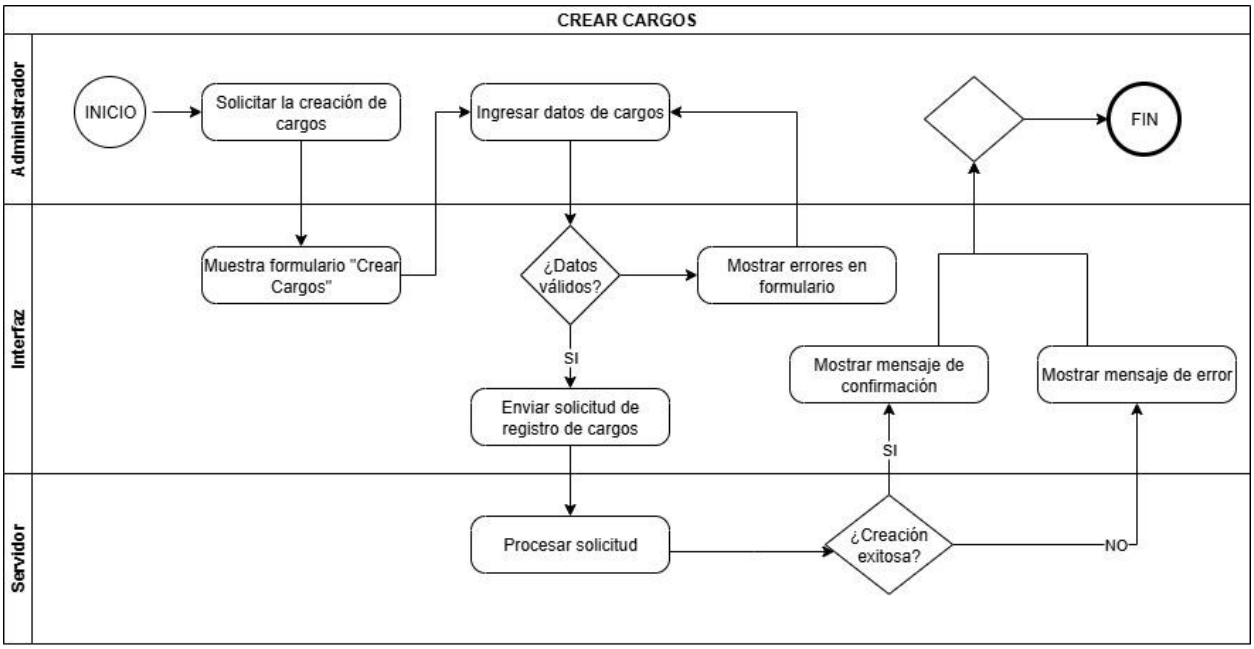


Ilustración 62 Flujo Crear Cargos.

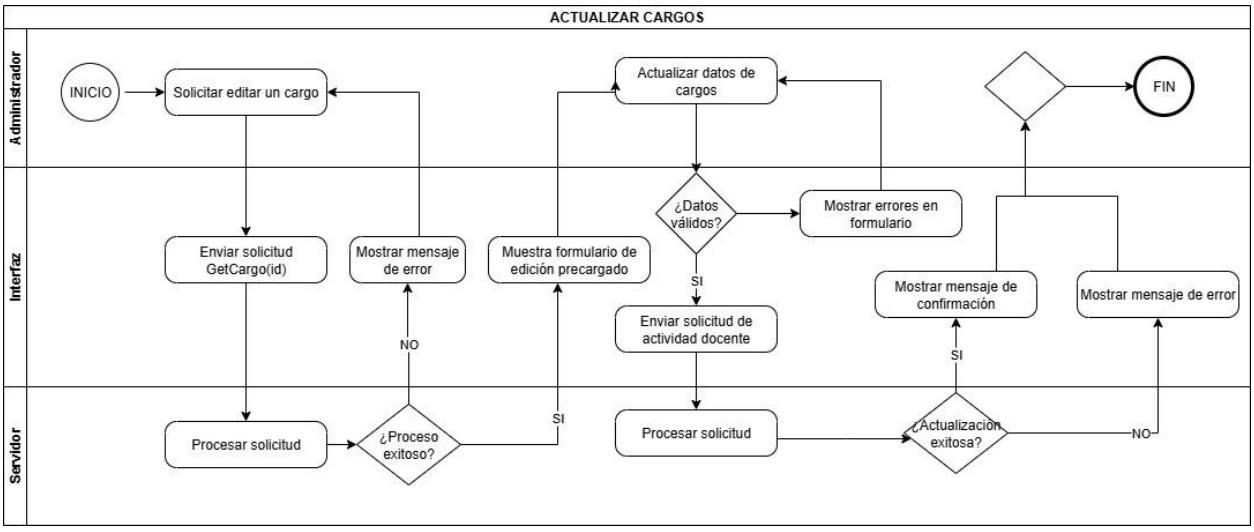


Ilustración 63 Flujo Actualizar Cargos.

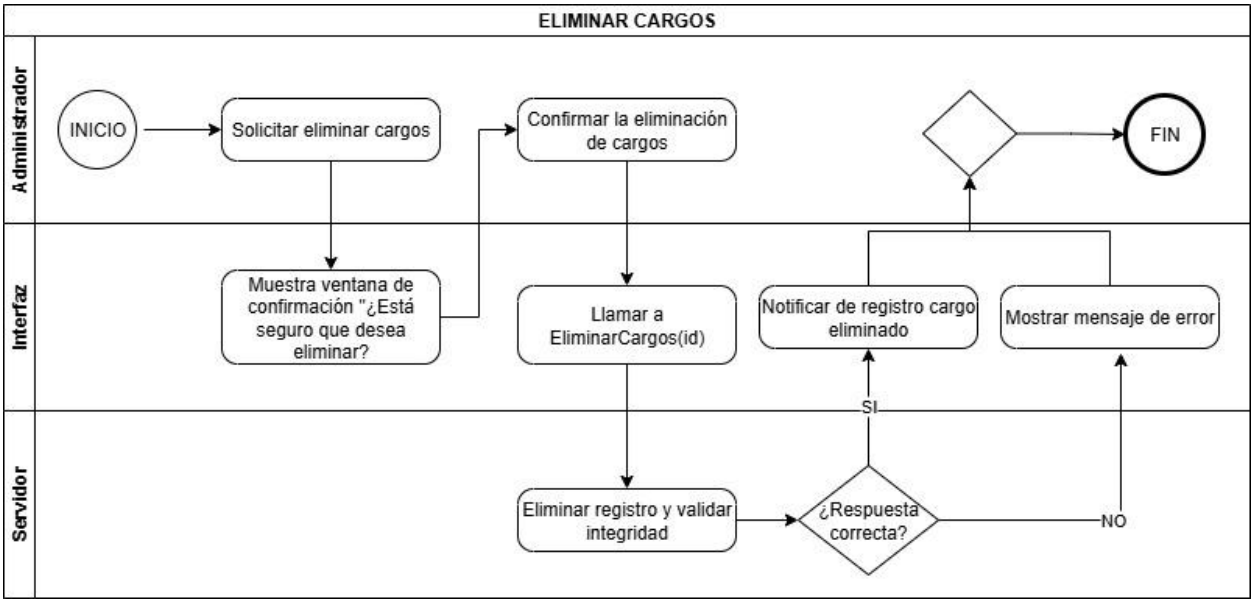


Ilustración 64 Flujo Eliminar Cargo.

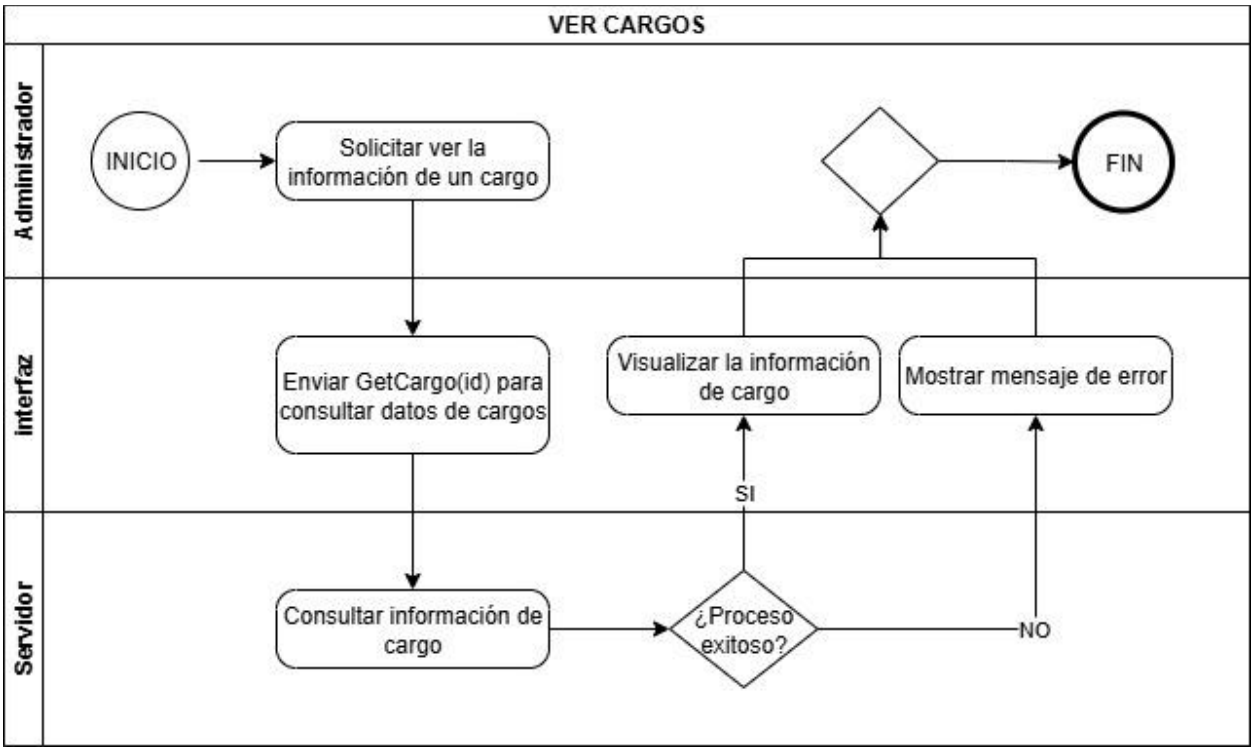


Ilustración 65 Flujos Ver Cargo.

Gestión Grupos.

Precondiciones: el usuario ha iniciado sesión con rol Administrador.

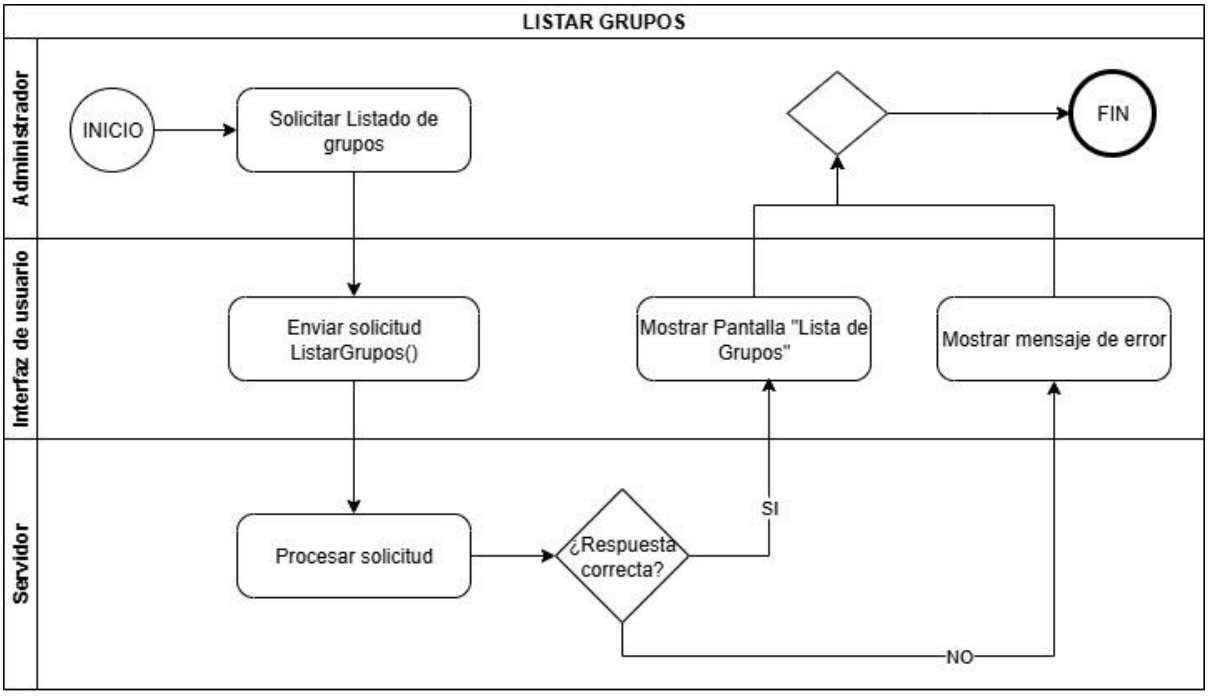


Ilustración 66 Flujo Listar Grupos.

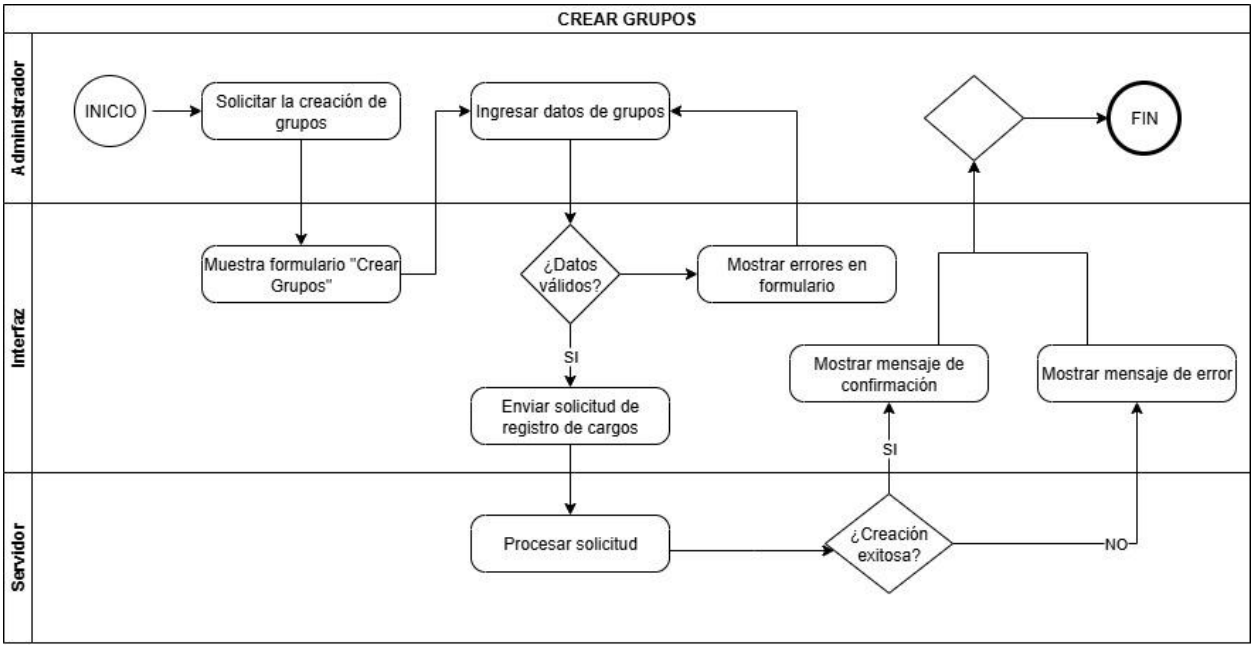


Ilustración 67 Flujo Crear Grupos.

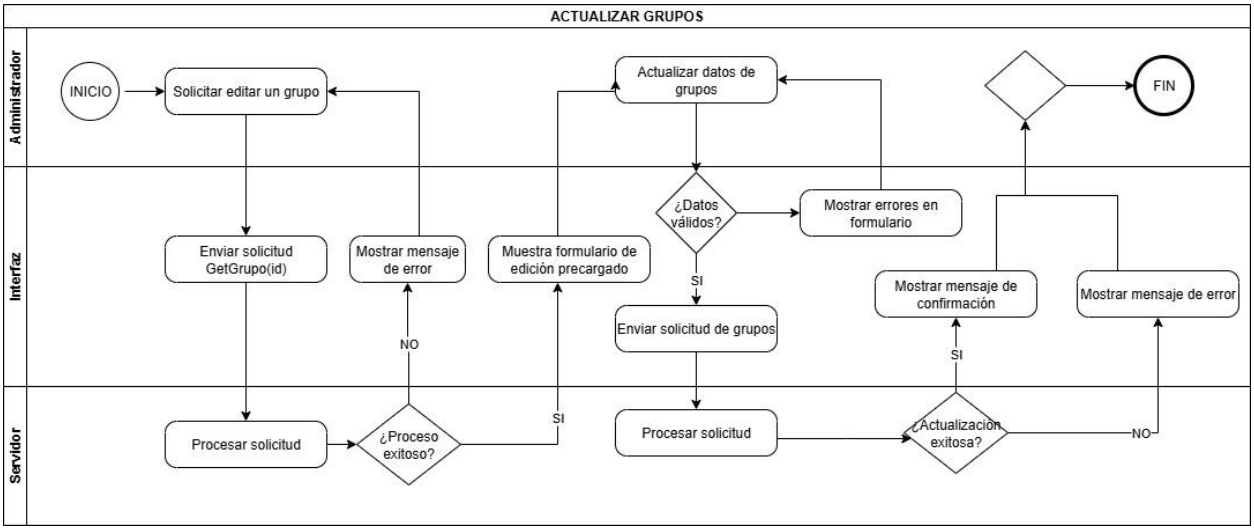


Ilustración 68 : Flujo Actualizar Grupos.

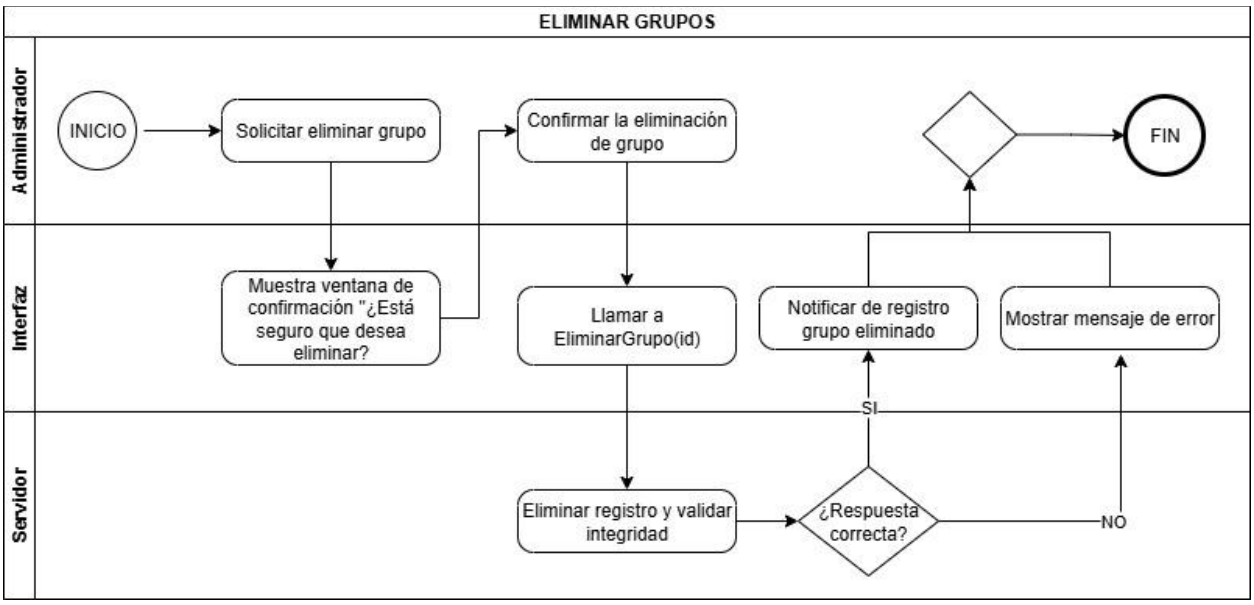


Ilustración 69 Flujo Eliminar Grupo.

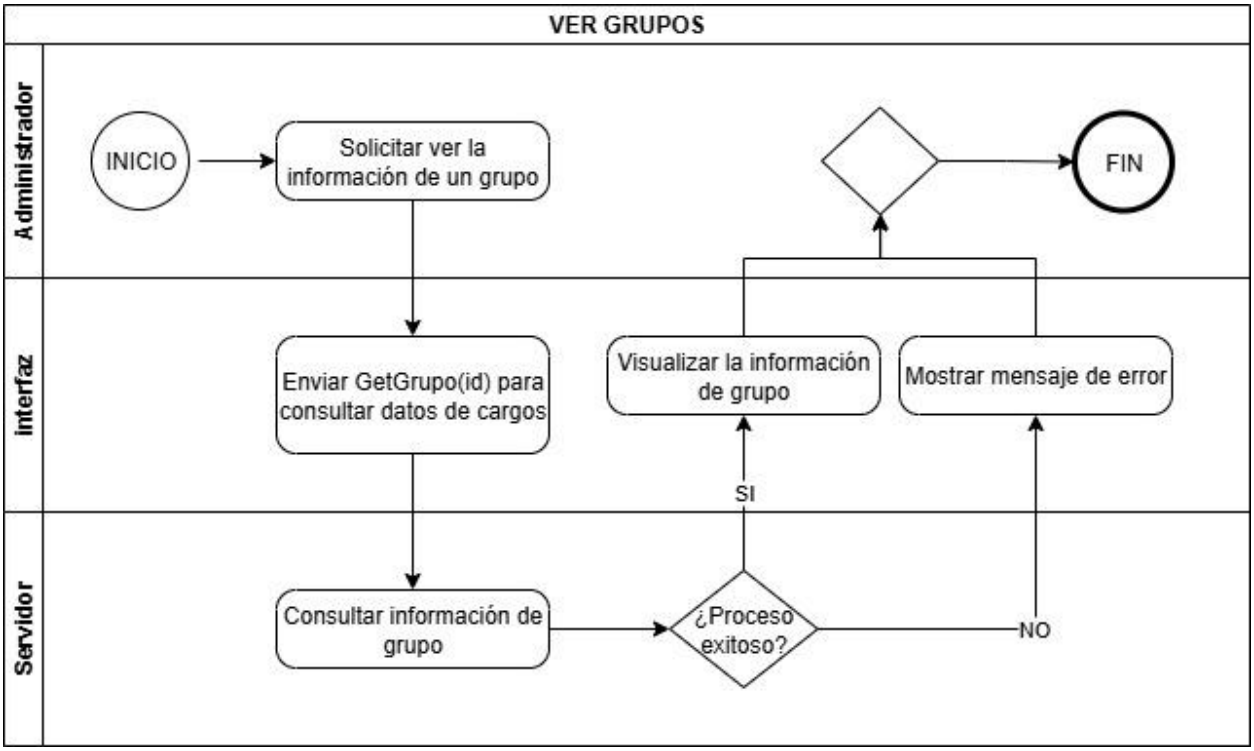


Ilustración 70 Flujo Ver Grupo.

Gestión de Contratación

Precondiciones: el usuario ha iniciado sesión con rol Director, Recursos Humanos o Asistente Administrativo.

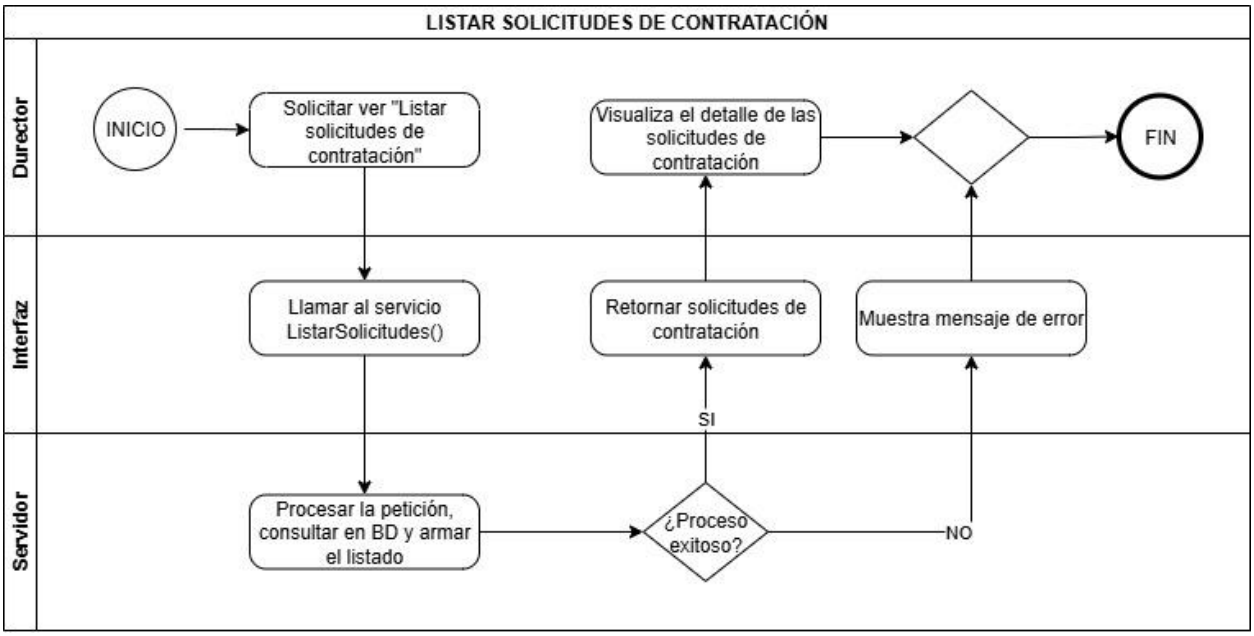


Ilustración 71 Flujo Listar Solicitudes de Contratación.

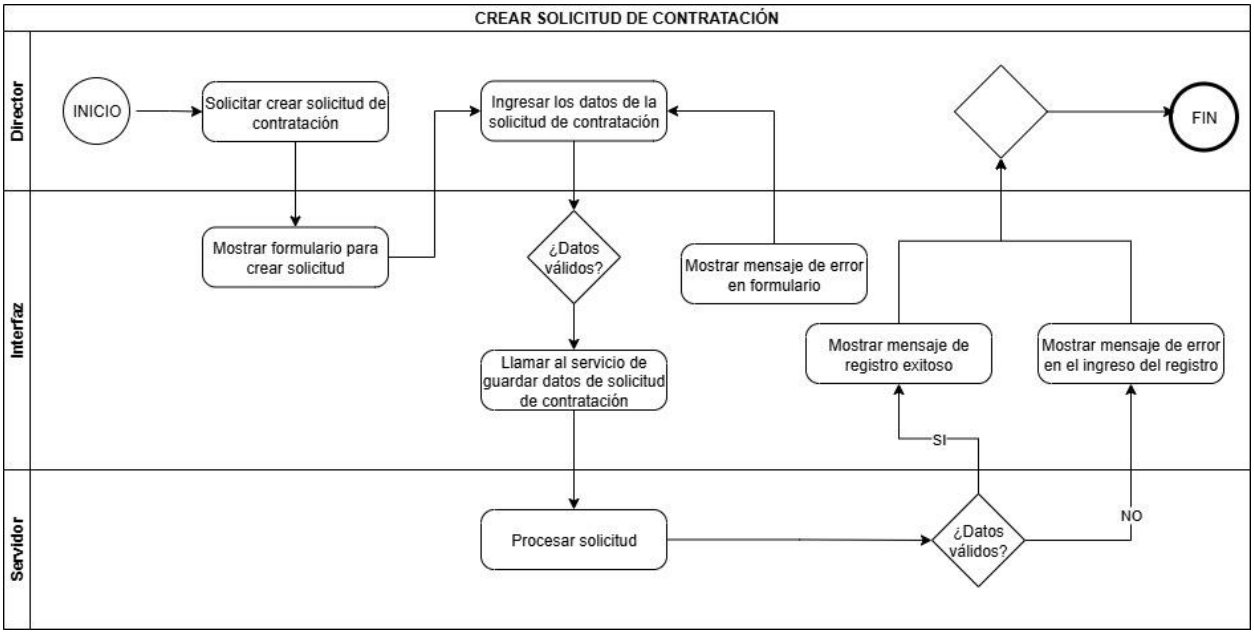


Ilustración 72 Flujo Crear Solicitud de Contratación.

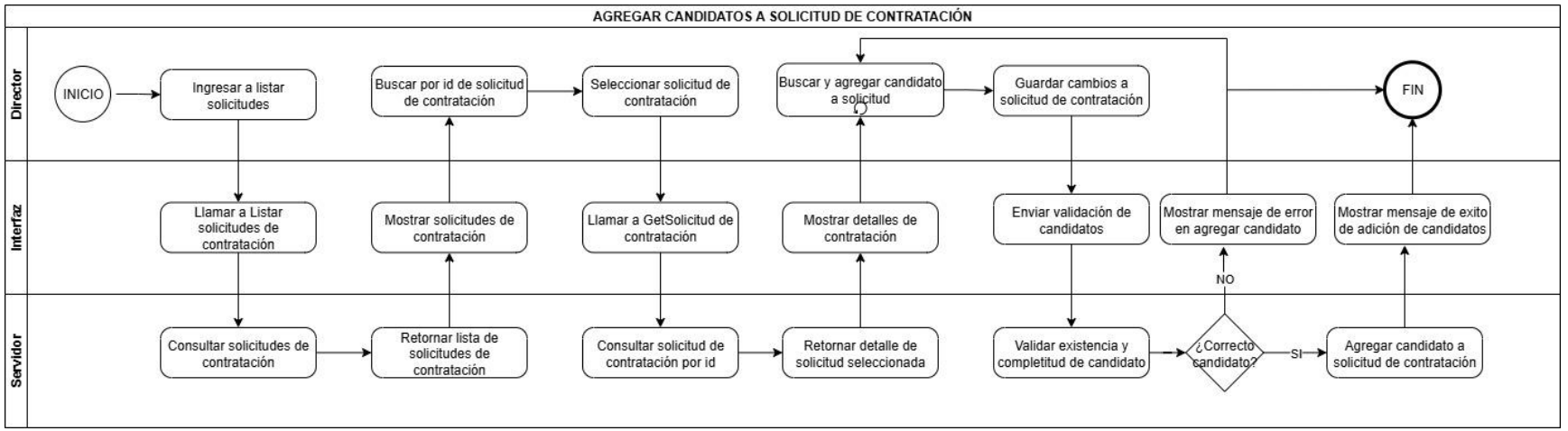


Ilustración 73 Flujo Agregar Candidatos en Solicitud de Contratación.

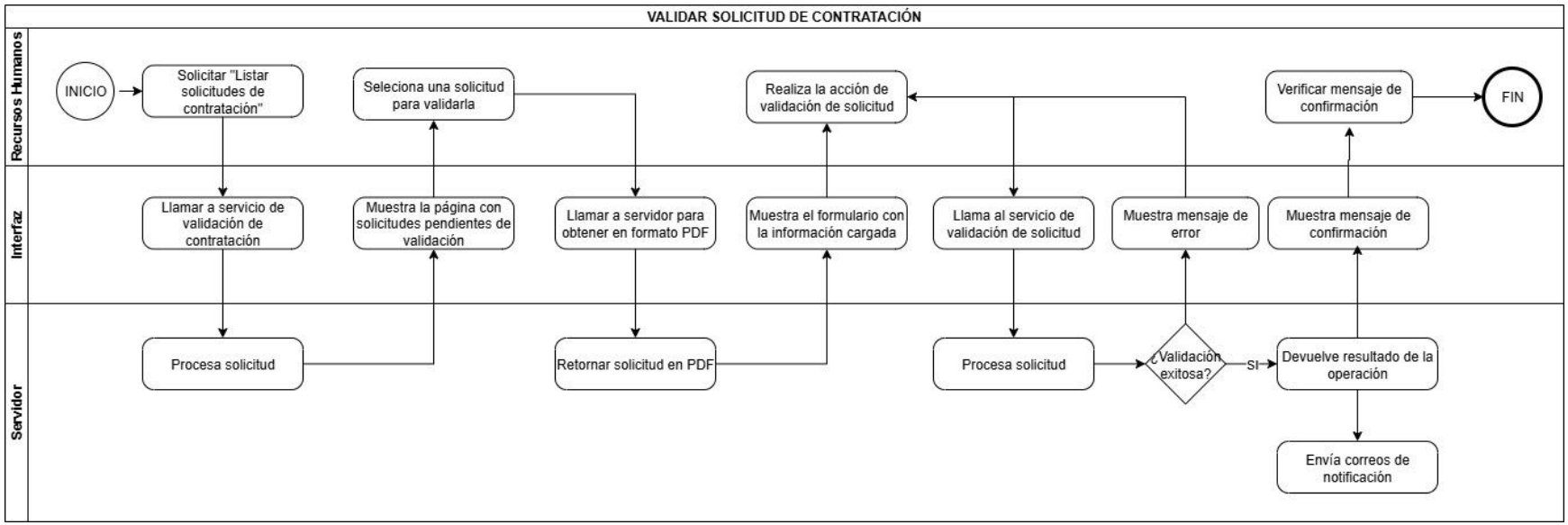


Ilustración 74 Flujo Validar Solicitud de Contratación.

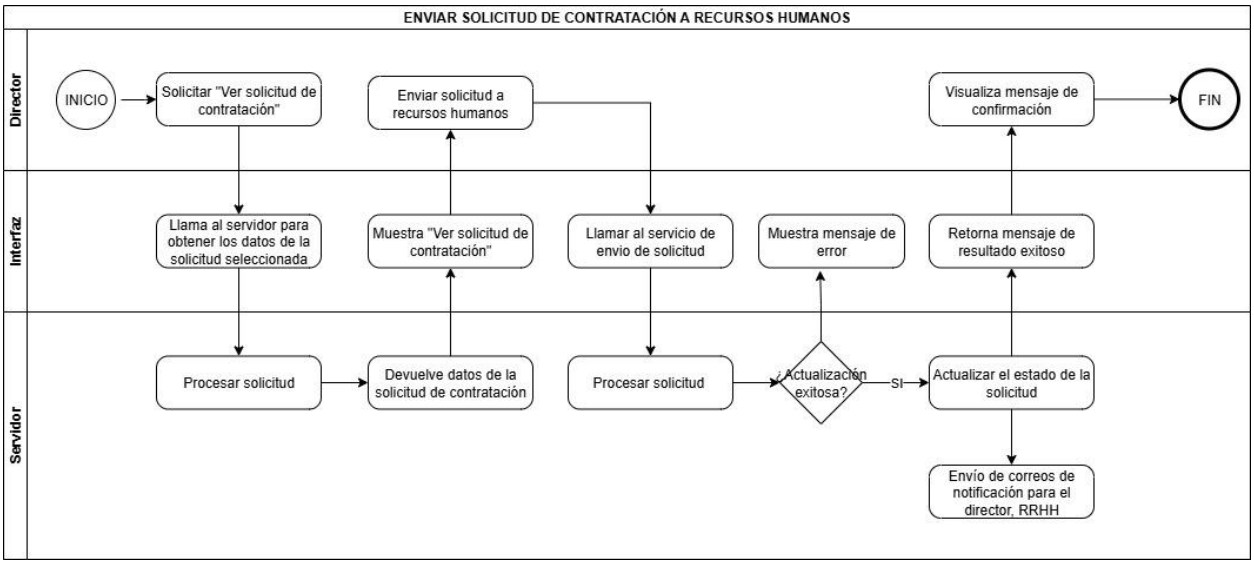


Ilustración 75 Flujo Enviar Solicitud de Contratación a Recursos Humanos.

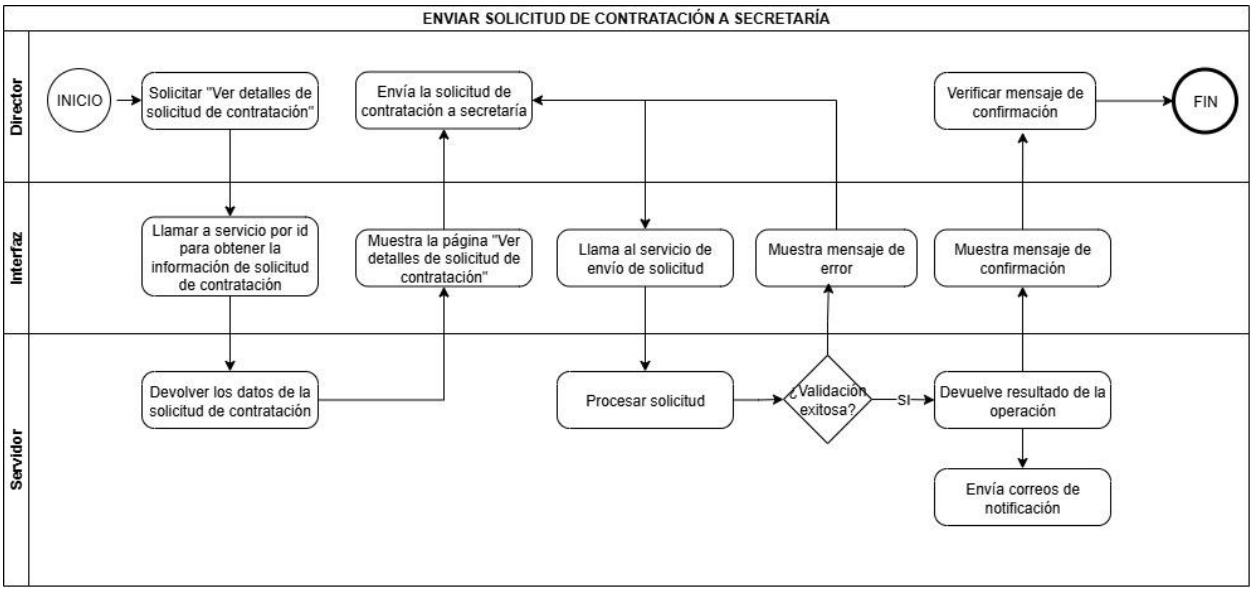


Ilustración 76 Flujo Enviar Solicitud de Contratación a Secretaría.

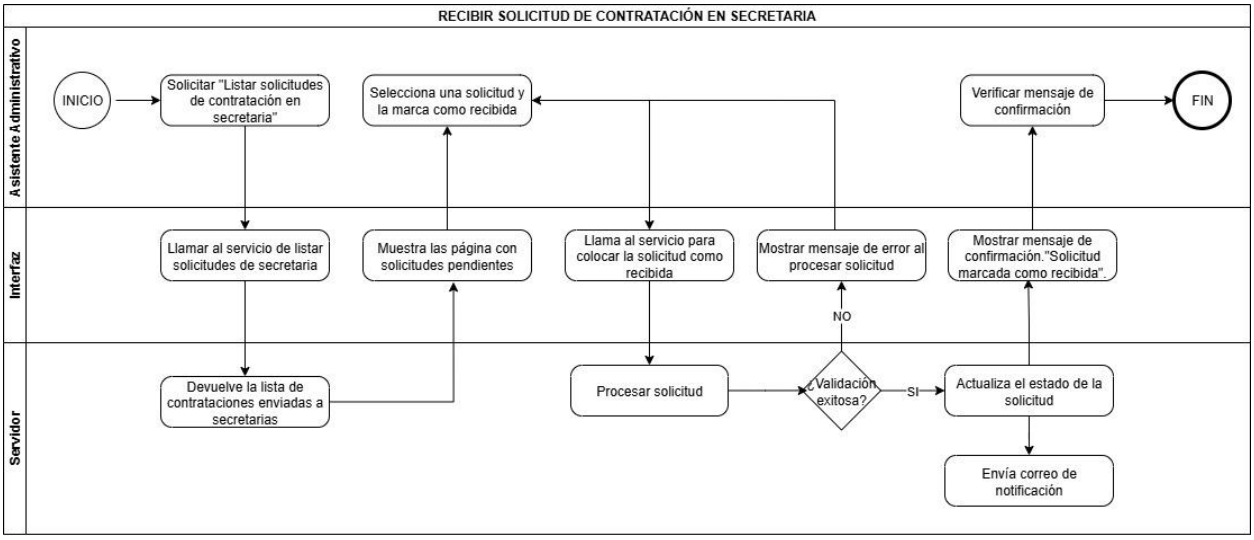


Ilustración 77 Flujo Recibir Solicitud de Contratación en la Secretaría.

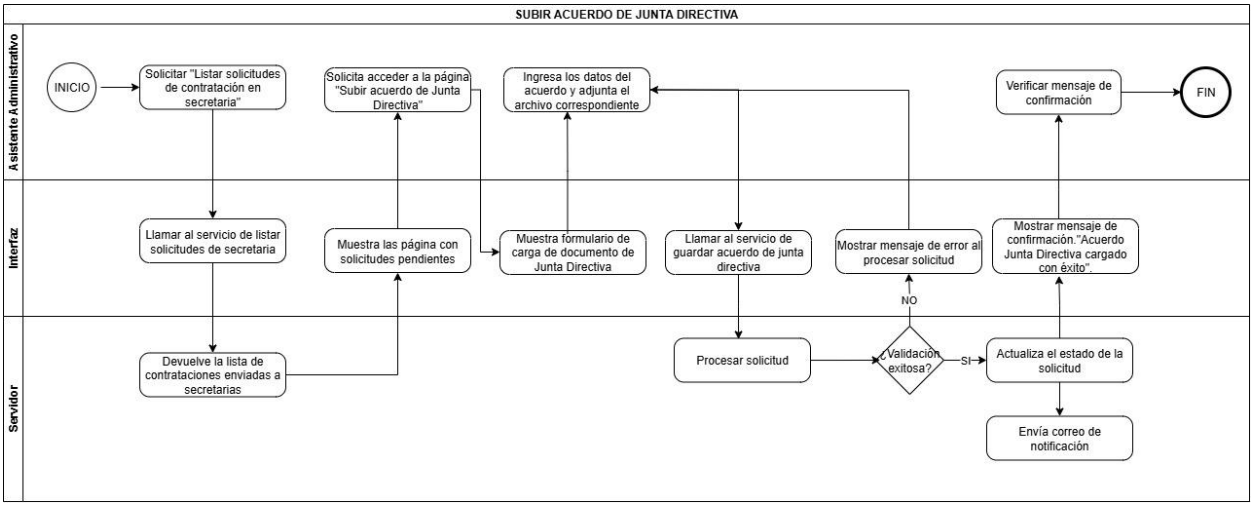


Ilustración 78 Flujo Subir Acuerdo de Junta Directiva.

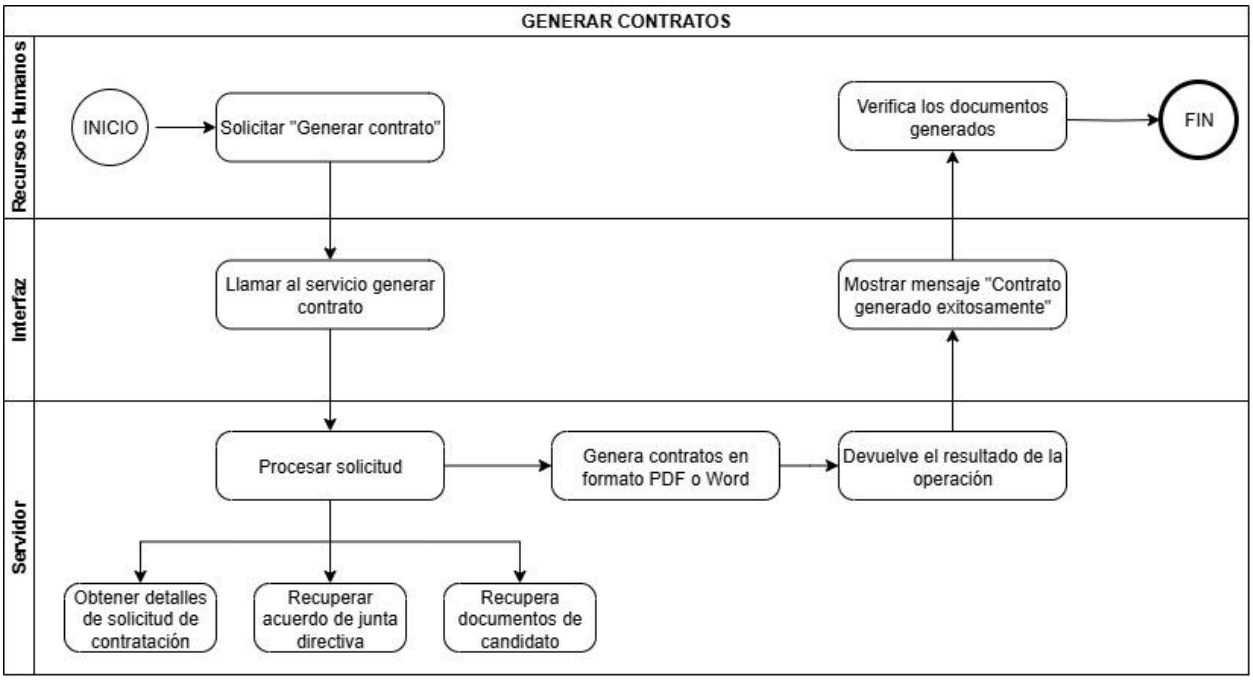


Ilustración 79 : Flujo Generar Contratos.

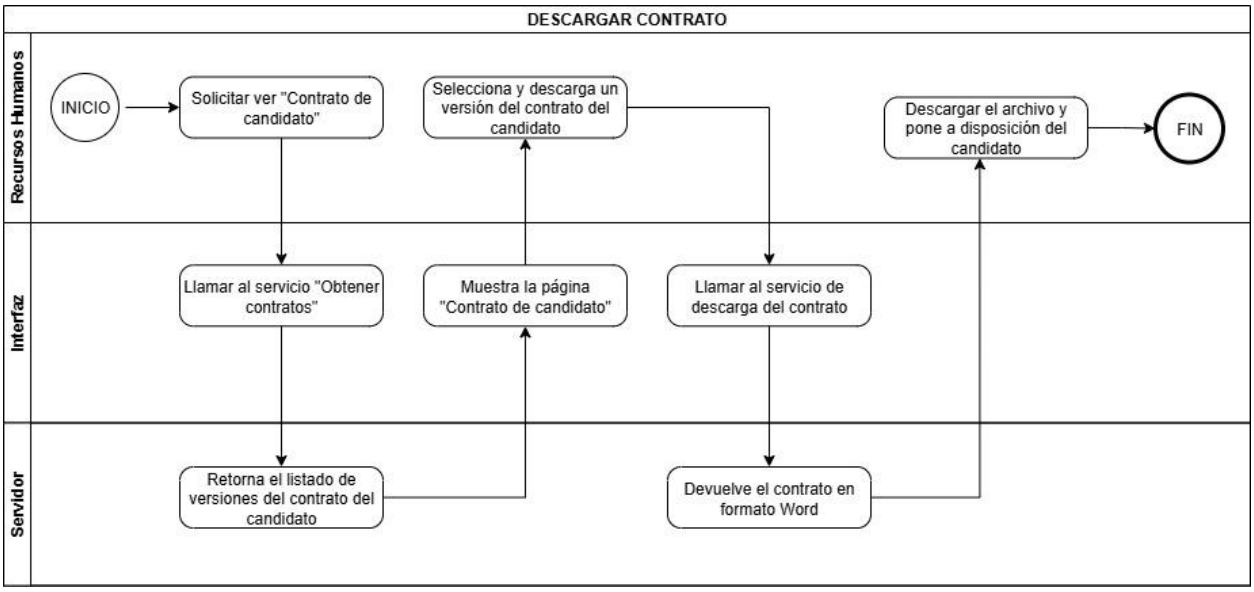


Ilustración 80 Flujo Descargar Contrato.

Iniciar Sesión

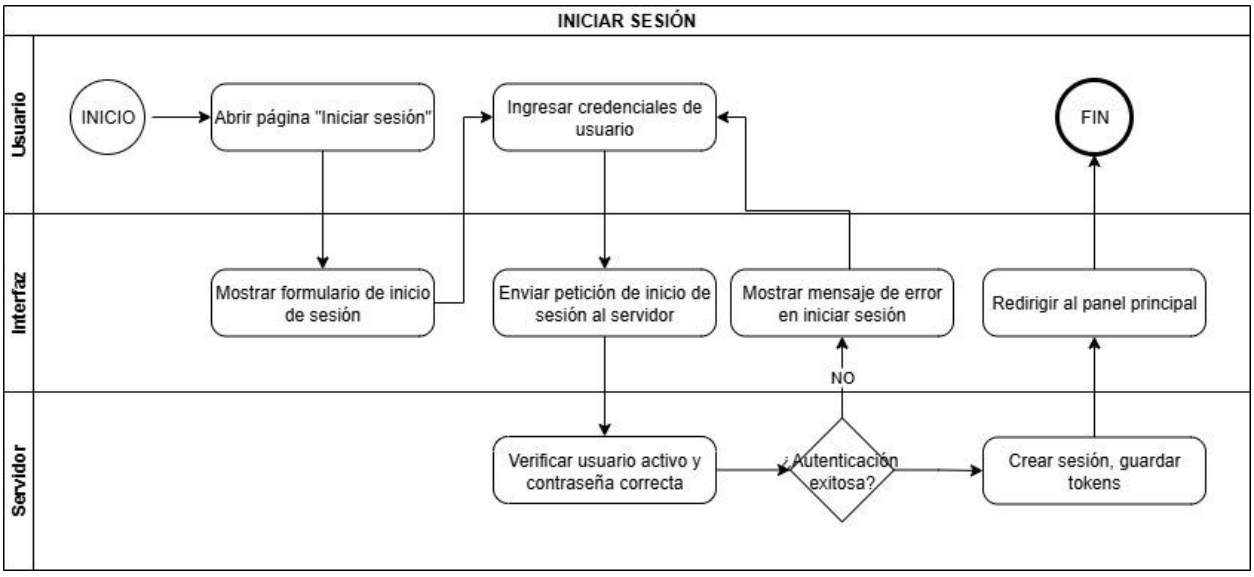


Ilustración 81 Flujo Iniciar Sesión.

Gestionar Información de Candidato

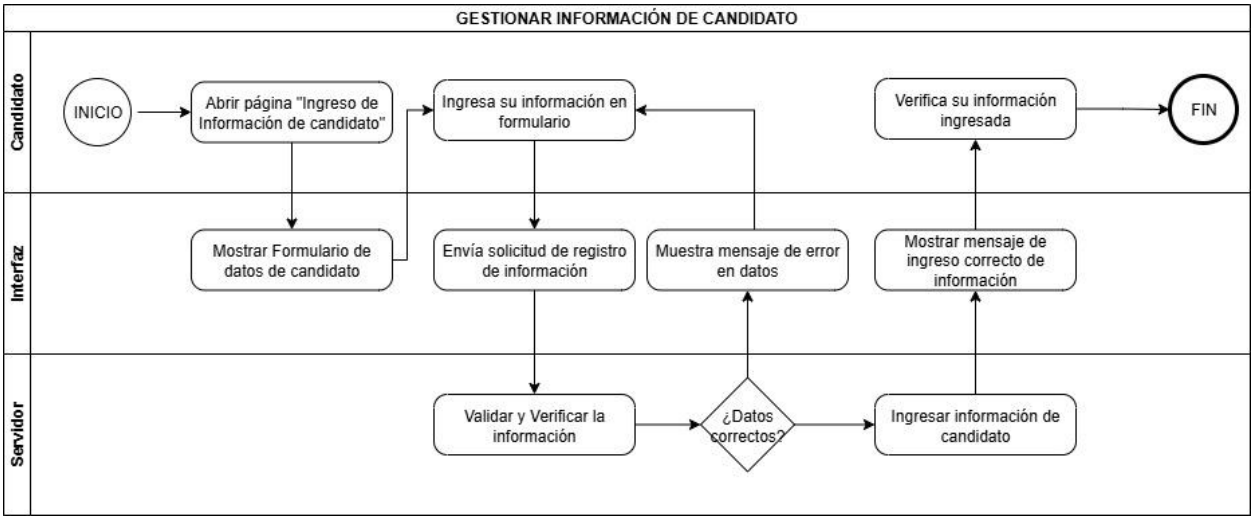


Ilustración 82 Gestionar Información de Candidato.

3.4 Necesidades del Negocio y Objetivos de Calidad

3.4.1 Necesidades del Negocio

Propósito General

El propósito general de la solución planteada es que mediante el plan integral de pruebas de calidad y pruebas automatizadas se pueda garantizar la calidad, confiabilidad, usabilidad y eficiencia del Sistema para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador. Así mismo, con la culminación del proyecto aseguramos que el usuario final obtenga el desempeño esperado del sistema cubriendo pilares importantes como compatibilidad, usabilidad, seguridad y fiabilidad.

Marco Estratégico

La propuesta de solución se fundamenta en un enfoque estratégico que se compone de los siguientes puntos:

1. El uso de herramientas para el análisis del problema como la matriz FODA, el diagrama causa efecto y el enfoque de sistemas nos permitieron tener una visión general y detalla de los temas críticos que impactan en la calidad del software.
2. En la actualidad con los avances tecnológicos a grandes escalas, es necesario que la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador priorice la modernización los procesos administrativos que muchas veces son demasiado largos y complejos, permitiendo facilitarlos y centralizarlos para garantizar la eficiencia operativa del equipo.
3. Para la planificación y ejecución de las pruebas se adoptarán metodologías de calidad reconocidas (ISO /IEC29119, ISTQB).
4. Se garantizará la sostenibilidad y escalabilidad buscando el aprovechamiento del tiempo con el uso de herramientas gratuitas y de software libre.
5. Incorporación de un ciclo de mejora continua, donde los resultados de las pruebas que se realizan permitan dar retroalimentación al equipo de desarrollo para que el sistema se mantenga actualizado.

Ruta de Implementación

Partiendo de los puntos a los que se realizarán para dar una solución oportuna a la problemática identificada, la implementación del plan se organizará en diferentes fases con el fin que se realice en un despliegue estructurado y efectivo.

1. **Diagnóstico inicial del proyecto:** se elaborará un análisis de la situación actual del sistema para poder documentar la información necesaria y con ello poder adoptar las mejores alternativas y prácticas en la ejecución de las pruebas de calidad tanto manuales como automatizadas.
2. **Planificación y documentación:** levantamiento de un plan de pruebas formal en donde se detalle casos técnicos, entradas, salidas esperadas, reglas del negocio y una matriz detalla de casos de prueba.
3. **Diseño y codificación de scripts:** creación de scripts automatizados con herramientas que permitan eficiencia y mantenibilidad.

4. **Automatización de pruebas**: implementación de pruebas automatizadas con herramientas de software Libre para que pueda validarse el sistema de forma continua y con ellos se logrará detectar errores y garantizar la escalabilidad.
5. **Creación de Documentación**: se entregará al equipo de calidad toda la documentación, scripts de prueba, reportes de ejecución, defectos detectados y manual de uso.
6. **Evaluación comparativa**: comparación de tiempos y errores antes y después de la automatización, recopilando observaciones para mejorar la reutilización de los scripts.

Importancia

La implementación de un proceso sistemático de evaluación y automatización de pruebas de la calidad de software en el Sistema Informático para la Gestión de Contratos de Personal Docente (FIA-UES) representa un paso fundamental para fortalecer la calidad del software institucional con el fin de determinar si el sistema cumple con sus objetivos operativos. Este proceso estará enfocado en detectar posibles errores, evaluar el rendimiento del sistema y proponer recomendaciones de mejora, sin realizar cambios directamente en el software. El objetivo es garantizar que la herramienta cumpla con los estándares de calidad establecidos, ofreciendo un soporte confiable para su funcionamiento diario. La importancia de este proyecto radica no solo en sus beneficios técnicos, sino también en su impacto institucional, social y económico.

Impacto Institucional

1. Mejora en la calidad del sistema: Mediante la automatización de pruebas funcionales se podrá identificar oportunamente errores en procesos claves, contribuyendo a la estabilidad y confiabilidad del sistema, especialmente en períodos de alta carga como la planificación académica.
2. Apoyo a los equipos internos: Al reducir la carga operativa de validaciones manuales, el personal encargado del aseguramiento de calidad puede concentrarse en actividades más analíticas y estratégicas, optimizando así el uso de los recursos humanos.
3. Fortalecimiento de la gestión académica y administrativa: Un sistema validado y con procesos bien definidos mejora el cumplimiento de los calendarios académicos y administrativos, evitando retrasos en contrataciones docentes y asegurando una planificación académica más efectiva.
4. Modelo replicable dentro de la universidad: Este proyecto establece una base que puede ser utilizada como referencia para aplicar buenas prácticas de QA en otros sistemas informáticos desarrollados por la UES, fomentando una cultura institucional de calidad.

Impacto Social

1. Confiabilidad en los servicios brindados: Un sistema que opera correctamente genera confianza en la comunidad universitaria, garantizando que los procesos de contratación docente sean transparentes, ágiles y confiables para todos los involucrados.
2. Modernización de prácticas institucionales: Al incorporar herramientas de automatización en procesos críticos, la universidad se posiciona como una institución comprometida con la innovación tecnológica y la mejora continua, lo que contribuye a fortalecer su imagen pública y académica.

Impacto Económico

1. Reducción de costos operativos: La automatización de pruebas permite disminuir el tiempo dedicado a validaciones repetitivas, lo cual se traduce en ahorro de recursos institucionales a mediano y largo plazo.
2. Sostenibilidad técnica del sistema: Detectar y corregir fallos antes de que impacten el entorno productivo reduce costos de mantenimiento y mejora la vida útil del sistema. Esto permite que los recursos económicos puedan ser redirigidos a otras áreas estratégicas dentro de la facultad.

3.4.2 Objetivos de Calidad del Sistema

Justificación

El Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador es una herramienta fundamental para la administración académica, ya que facilita procesos críticos como la contratación de docentes, la asignación de cargos, la validación de documentos y la gestión de ciclos académicos. Debido a la complejidad y la importancia de estos procesos, es esencial garantizar que el sistema funcione de manera eficiente, confiable y sin errores.

Actualmente, los procesos de validación y pruebas del sistema se realizan de manera manual, lo que implica un alto costo en términos de tiempo y esfuerzo, además de un mayor riesgo de pasar por alto errores que pueden afectar la operatividad de la facultad. La creciente demanda de eficiencia y la necesidad de asegurar la calidad del software han puesto la necesidad de adoptar un enfoque más automatizado y sistemático en las pruebas de software.

La implementación de pruebas automatizadas en este proyecto se justifica por varias razones claves, entre ellas tenemos:

1. **Eficiencia en el proceso de validación**. Las pruebas manuales son intensivas en tiempo y requieren una gran cantidad de recursos humanos. La automatización permite realizar pruebas de manera más rápida y eficiente, lo que reduce considerablemente los tiempos de validación y permite realizar un mayor número de pruebas en menor tiempo. Esto es especialmente importante cuando se realizan actualizaciones al sistema o cuando se debe validar un alto volumen de datos.
2. **Mejora en la precisión y fiabilidad**. Las pruebas automatizadas minimizan el riesgo de errores humanos, garantizando una mayor precisión en los resultados. Los scripts de prueba automatizados pueden ejecutar exactamente las mismas acciones de manera consistente, lo que aumenta la fiabilidad del sistema y permite detectar fallos o inconsistencias que podrían pasar desapercibidos durante las pruebas manuales.
3. **Sostenibilidad y escalabilidad del proceso de pruebas**. La automatización de pruebas proporciona una solución escalable que puede ser reutilizada en futuros ciclos del sistema o en nuevas versiones del software. Además, permite realizar pruebas regulares sin necesidad de intervención manual constante, lo que facilita el mantenimiento del sistema a largo plazo.
4. **Optimización de recursos humanos**. Al liberar al equipo de calidad de la necesidad de ejecutar pruebas repetitivas, estos pueden centrarse en tareas más analíticas, como la

evaluación de resultados o la mejora de otros aspectos del sistema. Esto no solo mejora la productividad, sino que también contribuye a un ambiente de trabajo más eficiente y satisfactorio para los miembros del equipo.

5. **Mejora continua en el proceso de calidad del software**. Implementar pruebas automatizadas no solo optimiza la validación del sistema actual, sino que también contribuye a mejorar las prácticas de aseguramiento de calidad dentro de la institución. La automatización fomenta la adopción de metodologías modernas de pruebas, que son esenciales para garantizar la calidad en un entorno de desarrollo de software dinámico y en constante evolución.
6. **Cumplimiento de estándares institucionales**. La adopción de pruebas automatizadas en el sistema también responde a un enfoque institucional de mejora continua en la calidad del software, alineándose con los objetivos estratégicos de la universidad de optimizar sus procesos y adoptar tecnologías más eficientes.

Resultados Esperados

Tras el desarrollo del proyecto en donde se busca implementar un plan integral de pruebas de calidad en el Sistema Informático para la Gestión de Contratos de Personal Docente de la FIA-UES y su respectiva automatización, permitirá que logremos los siguientes resultados:

1. Marco de pruebas estandarizado: con la aplicación de la norma ISO/IEC 29119 se implementará una matriz de pruebas formalmente documentada, esta matriz nos permitirá evaluar los módulos críticos del sistema, pero también nos generará la trazabilidad entre requisitos, casos de prueba y los resultados. Al contar con una matriz documentada de manera técnica esta puede ser utilizada para futuros proyectos informáticos de la Universidad de El Salvador o para pruebas de regresión según sea conveniente.
2. Evidencias de la ejecución de las pruebas: se deberá de identificar, clasificar y documentar la ejecución de las evidencias de las pruebas realizadas, así mismo si durante las pruebas se encuentran defectos, errores funcionales, fallos en la integración del sistema o deficiencias en la usabilidad. Los hallazgos deberán ser analizados mediante el uso de métricas para poder medir el impacto en los procesos de la contratación del personal docente.
3. Métricas de calidad: se entregarán un conjunto de métricas que nos permita validar en un futuro poder medir si el sistema mejora o empeora con respecto a la calidad con nuevas versiones.
4. Comparativa de pruebas manuales y automatizadas: al ejecutar ambos enfoques de pruebas en paralelo para el proyecto, obtendremos resultados que nos permitirán comparar la eficiencia, cobertura y confiabilidad de cada uno. Se espera que podamos demostrar a través de variables cuantitativas la reducción de tiempos de ejecución y disminuir los errores humanos en tareas repetitivas, validando así los beneficios o contras de ambos enfoques.
5. Documentación técnica: se elaborarán un conjunto de entregables que incluyen scripts automatizados, reportes de las ejecuciones, defectos clasificados, métricas y manuales técnicos. Estos productos servirán como una guía de referencia para la reutilización y

validación general del sistema o también para que sea tomada de base para la verificación de un nuevo sistema.

6. Sostenibilidad tecnológica: con la implementación del plan de pruebas se espera aportar a la facultad un mecanismo que asegure la calidad en el proyecto. Los resultados ayudaran a verificar que el sistema cumple con los atributos como fiabilidad, usabilidad y mantenibilidad.

3.4.3 Requerimientos Funcionales

En el marco del Sistema Informático para la Gestión de Contratos de Personal Docente de la Facultad de Ingeniería y Arquitectura (FIA-UES), se identifican los siguientes requerimientos funcionales que garantizan la correcta ejecución de los procesos administrativos y académicos asociados a la contratación de docentes, con el fin de asegurar la confiabilidad, eficiencia y trazabilidad del sistema, además de servir como base para las pruebas manuales y automatizadas.

Para la recopilación de los requerimientos funcionales del sistema, se utilizó como técnica principal la entrevista, aplicada a través de una encuesta estructurada dirigida a las autoridades de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador, quienes participan directamente en el proceso de contratación de personal docente

El cuestionario aplicado fue diseñado previamente y se presenta en el Anexo 1 como referencia**,** el cual sirvió como instrumento de recolección de datos. Dicho formato incluyó preguntas abiertas y cerradas que permitieron obtener tanto información cualitativa como cuantitativa sobre las necesidades, problemas y expectativas relacionadas con la gestión de contratos.

.

ID	Requerimiento Funcional	Descripción	Actor(es)	Prioridad	Criterio de Aceptación
RF-01	Autenticación y control de accesos (Login de usuario)	Permitir inicio de sesión en el sistema y que sea capaz de autenticar a un usuario previamente registrado y le conceda el acceso al sistema una vez se validen sus credenciales, reconozca e identifique los permisos asignados y el tipo de usuario para que conforme a esto el sistema muestre la información de acuerdo a los permisos concedidos.	Usuarios, Administradores	Alta	Escenario 1 1. Ingresar el usuario y contraseña. 2. Validar los datos ingresados del usuario 3. El usuario inicia sesión exitosamente. 4. Se muestra el contenido en pantalla según su rol. Escenario 2 1. Ingresa el usuario y contraseña. 2. Se validan los datos ingresados del usuario. 3. El usuario no inicia sesión porque las credenciales ingresadas no coinciden con los registros de la base de datos. 4. Se muestra mensaje de fallo en inicio de sesión

ID	Requerimiento Funcional	Descripción	Actor(es)	Prioridad	Criterio de Aceptación
RF-02	Registro de candidatos	Ingresar datos personales, académicos y profesionales de docentes; cargar documentos digitalizados.	RRHH	Alta	Se puede registrar un candidato y almacenar documentos en el sistema con confirmación.
RF-03	Validación de documentos	Validar la autenticidad de documentos cargados (títulos, escalafones, documentos personales.) y asignar estado (aprobado/rechazado/pendiente).	RRHH, Autoridades	Alta	Los documentos cambian de estado correctamente y se refleja en el perfil del candidato.
RF-04	Generación de contratos	Generar automáticamente contratos digitales con datos del docente, asignaturas y condiciones administrativas.	RRHH, Autoridades	Alta	El contrato se genera en PDF/Word con datos correctos y descargable.
RF-05	Asignación de carga académica	Asignar materias y horarios a los docentes contratados, vinculados a ciclos académicos.	Coordinadores, Autoridades	Alta	El sistema registra correctamente la asignación y muestra la carga en el perfil del docente.
RF-06	Gestión de ciclos académicos	Crear, editar y cerrar ciclos académicos, vinculando contratos y asignaciones.	Administradores, Autoridades	Media	Se pueden abrir/cerrar ciclos y asociar contratos sin errores.
RF-07	Notificaciones y trazabilidad	Enviar notificaciones automáticas de cambios en estado de contratos y guardar historial de acciones.	Todos los usuarios	Media	Los usuarios reciben notificaciones (email/sistema) y el historial se conserva en bitácora.

ID	Requerimiento Funcional	Descripción	Actor(es)	Prioridad	Criterio de Aceptación
RF-08	Generación de reportes	Emitir reportes sobre docentes contratados, cargas académicas y validaciones.	Administradores, Autoridades	Media	Se generan reportes en PDF/Excel con filtros configurables.
RF-09	Seguridad de datos	Proteger información personal y académica mediante cifrado, control de accesos y enmascaramiento de datos.	Administradores, RRHH	Alta	Los datos sensibles no se muestran en claro y se restringe el acceso a roles no autorizados.
RF-10	Auditoría interna	Registrar bitácora de operaciones (Historial de procesos realizados).	Administradores	Media	El sistema almacena logs consultables con fecha, hora, usuario y acción ejecutada.

Tabla 24: Tabla Requisitos Funcionales

3.4.4 Requerimientos No Funcionales

En el contexto del Sistema de Gestión de Contratos de Personal Docente de la Universidad de El Salvador, los RNF garantizan que la solución no solo cumpla con los procesos administrativos requeridos, sino que además lo haga de manera eficiente, segura, confiable y alineada con la normativa vigente. Por lo cual hemos listado los siguientes RNF:

Requerimiento No Funcional	**NRF01 – Encriptación de contraseñas de usuario.**
Tipo	Seguridad
Prioridad	Alta
Descripción	Las contraseñas al ser un dato sensible para los usuarios deben almacenarse en la base de datos con algoritmo de encriptación SHA-256.
Criterios de aceptación	• Ninguna contraseña debe almacenarse en texto plano. • Se valida que los datos de contraseña se encuentran cifrados en la base.

Tabla 25: NRF01 - Encriptación de contraseñas de usuario.

Requerimiento No Funcional	**NRF02 – Tiempo de sesión máximo por inactividad**
Tipo	Seguridad
Prioridad	Alta
Descripción	El sistema debe cerrar automáticamente la sesión del usuario después de 120 minutos de inactividad.
Criterios de aceptación	• Toda sesión inactiva por más de 120 minutos debe cerrarse de manera automática. • El sistema redirige al formulario de inicio de sesión al intentar una acción posterior al cierre.

Tabla 26: NRF02 - Tiempo de sesión máximo por inactividad

Requerimiento No Funcional	**NRF03 – Bitácora de registro de acciones del sistema**
Tipo	Seguridad
Prioridad	Alta
Descripción	El sistema debe registrar en una bitácora en una bitácora las acciones realizadas por los usuarios y catalogarlas según nivel de criticidad.
Criterios de aceptación	• Cada acción queda registrada. • Los registros no pueden ser modificados ni eliminados por usuarios finales. • La bitácora está protegida y solo puede acceder usuarios con los perfiles autorizados.

Tabla 27: NRF03 - Bitácora de registro de acciones del sistema

Requerimiento No Funcional	**NRF04 – Control de accesos basados en roles (RBAC)**
Tipo	Seguridad
Prioridad	Alta
Descripción	El sistema debe registrar en una bitácora en una bitácora las acciones realizadas por los usuarios y catalogarlas según nivel de criticidad.
Criterios de aceptación	• Cada acción queda registrada. • Los registros no pueden ser modificados ni eliminados por usuarios finales. • La bitácora está protegida y solo puede acceder usuarios con los perfiles autorizados.

Tabla 28: NRF04 - Control de accesos basados en roles (RBAC)

Requerimiento No Funcional	**NRF05 – Complejidad mínima de contraseñas**
Tipo	Seguridad
Prioridad	Alta
Descripción	El sistema debe exigir que toda contraseña ingresada tenga una longitud de 8 caracteres e incluya al menos una mayúscula, una minúscula, un digito y un carácter especial.
Criterios de aceptación	• El sistema rechaza contraseñas que no cumplan con la política. • Al intentar registrar un usuario con una contraseña insegura, se muestra un mensaje de error indicando los requisitos.

Tabla 29: NRF05 - Complejidad mínima de contraseñas

Requerimiento No Funcional	**NRF06 – Cifrado y resguardo de documentos**
Tipo	Seguridad
Prioridad	Alta
Descripción	Todos los documentos guardados y almacenados en el sistema deben guardarse en un directorio privado y cuando corresponda deberán estar cifrados.
Criterios de aceptación	• Los archivos no son accesibles a través de URL directa desde el navegador. • Los documentos solo pueden descargarse mediante un endpoint con autenticación y autorización. • Los archivos no se pueden leer desde el servidor sin pasar por el sistema.

Tabla 30: NRF06 - Cifrado y resguardo de documentos

Requerimiento No Funcional	NRF07 – Expiración de contraseñas
Tipo	Seguridad
Prioridad	Media
Descripción	Las contraseñas deben caducar cada 90 días, obligando al usuario a actualizarlas.
Criterios de aceptación	• Al vencer la contraseña, el Login es rechazado con mensaje de caducidad. • El sistema obliga a definir una nueva contraseña antes de permitir el acceso.

Tabla 31: NRF07 - Expiración de contraseñas

Requerimiento No Funcional	NRF08 – Transporte seguro (HTTPS-HSTS)
Tipo	Seguridad
Prioridad	Alta
Descripción	Todo el tráfico entre clientes y el sistema debe transmitirse de manera cifrada usando HTTPS (TLS 1.2 o superior). El servidor debe forzar el uso de conexiones seguras a través de políticas HSTS (HTTP Strict Transport Security), asegurándose que solo se conecte a través de HTTPS y evitando ataques de degradación o de tipo man-in-the-middle.
Criterios de aceptación	• Todo acceso HTTP debe redirigir permanentemente a HTTPS. • Todas las respuestas contienen el header Strict-Transport-Security. • El sitio solo acepta STL 1.2/1.3 • El sistema no responde con protocolos inseguros.

Tabla 32: NRF08 – Transporte seguro (HTTPS-HSTS)

Requerimiento No Funcional	NRF09 – Disponibilidad de la API
Tipo	Funcionalidad
Prioridad	Alta
Descripción	La API debe garantizar un nivel de servicio para usuarios administrativos, docentes, asegurando que el sistema este accesible y funcional durante el horario laboral (8:00 am a 5:00 pm). Mantener un nivel de disponibilidad mensual de al menos 99.5%
Criterios de aceptación	• Los reportes de monitoreo muestran un uptime mensual mayor o igual a 99.5%. • Se dispone de evidencias que demuestran cumplimiento.

Tabla 33: NRF09 – Disponibilidad de la API

Requerimiento No Funcional	**NRF10 – Paginación por defecto.**
Tipo	Funcionalidad
Prioridad	Media
Descripción	Los listados de registros que entrega la API deben implementar paginación controlada para garantizar el desempeño y evitar sobrecargas para el servidor.
Criterios de aceptación	• Al consultar un endpoint de listado sin per_page, devuelve 50 registros por página. • Los parámetros validos devuelven exactamente la cantidad solicitada. • El frontend refleja correctamente la paginación que devuelve la API.

Tabla 34 NRF10 – Paginación por defecto.

Requerimiento No Funcional	**NRF11 – Mensajes de validación.**
Tipo	Funcionalidad
Prioridad	Alta
Descripción	El sistema debe mostrar los mensajes de validación claros e inmediatos cuando el usuario ingrese datos incorrectos o incompletos en formularios.
Criterios de aceptación	• Los mensajes se muestran junto al campo afectado. • El mensaje indica la causa y la forma de corregir el valor. • No se permite enviar formularios hasta resolver la validación.

Tabla 35 NRF11 – Mensajes de validación.

Requerimiento No Funcional	**NRF12 – Mensajes de alerta.**
Tipo	Funcionalidad
Prioridad	Media
Descripción	El sistema debe mostrar alertas en situaciones críticas que requieran confirmación por parte del usuario.
Criterios de aceptación	• La alerta se muestra como ventana modal o banner destacado. • Requiere acción explicita de confirmación o cancelación del usuario. • La acción se ejecuta solo tras la confirmación.

Tabla 36 NRF12 – Mensajes de alerta.

Requerimiento No Funcional	**NRF13 – Mensajes de advertencia.**
Tipo	Funcionalidad
Prioridad	Media
Descripción	El sistema debe mostrar alertas cuando una acción pueda generar consecuencias no críticas, pero relevantes.
Criterios de aceptación	• El mensaje advierte sin bloquear la acción obligatoriamente. • Incluye recomendación para prevenir errores futuros. • El usuario puede decidir continuar o cancelar

Tabla 37 NRF13 – Mensajes de advertencia.

Requerimiento No Funcional	**NRF14 – Mensajes de Notificación vía correo electrónico.**
Tipo	Funcionalidad
Prioridad	Media
Descripción	El sistema debe enviar notificaciones automáticas por correo electrónico para eventos relevantes (creación de contrato, cambio de estado de solicitud, restablecimiento de contraseña).
Criterios de aceptación	• El correo debe enviarse inmediatamente luego de realizado el evento. • El correo posee información básica del evento reportado. • El diseño del correo es adaptable y legible para los diferentes dominios de correo.

Tabla 38 NRF14 – Mensajes de Notificación vía correo electrónico.

Requerimiento No Funcional	**NRF15 – Descarga de archivos.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	El sistema debe permitir descargar al usuario archivos generados de manera sencilla y desde la interfaz.
Criterios de aceptación	• Los botones de descarga son fáciles de identificar y los archivos se descargan con un solo clic. • Los archivos son descargados en el formato esperado y con un nombre representativo. • Los archivos que pueden ser descargados, solo los pueden descargar los usuarios que poseen permisos autorizados.

Tabla 39 NRF15 – Descarga de archivos.

Requerimiento No Funcional	**NRF16 – Menú dinámico.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	El sistema debe de mostrar un menú dinámico y que se adapte según el rol y los permisos que posee el usuario.
Criterios de aceptación	• Los usuarios ven solo las opciones que les corresponden según el rol que tienen asignado. • El menú se actualiza automáticamente al modificar o actualizar el rol a un usuario. • La navegación es clara, fácil de entender ya accesible desde cualquier pantalla.

Tabla 40 NRF16 – Menú dinámico.

Requerimiento No Funcional	**NRF17 – Adaptabilidad dinámica en uso.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	El sistema debe adaptarse dinámicamente a distintos dispositivos, resoluciones y contextos de uso, garantizando una visualización correcta y una experiencia fluida en computadoras.
Criterios de aceptación	• La interfaz debe ser responsive, ajustando automáticamente el diseño a diferentes resoluciones de pantalla. • Los formularios y menús deben mantener su funcionalidad completa sin importar el dispositivo. • Las pruebas de compatibilidad deben cubrir al menos los navegadores más usados.

Tabla 41 NRF17 – Adaptabilidad dinámica en uso.

Requerimiento No Funcional	**NRF18 – Menor uso de campos de registro manuales.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	El sistema debe reducir el tipeo manual en formularios mediante pre poblado de datos, autocompletado desde catálogos/maestros, valores por defecto
Criterios de aceptación	• Autocompletado de los campos a partir de catálogos disponibles. • Precargar datos personales ya registrados y documentos cargados. • Mostrar solo campos relevantes según selección previa. • Validación de campos en tiempo real.

Tabla 42 NRF18 – Menor uso de campos de registro manuales.

Requerimiento No Funcional	**NRF19 – Facilidad de uso.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	El sistema debe ofrecer una interfaz amigable que facilite el aprendizaje y uso por parte de los usuarios, reduciendo la necesidad de capacitación extensa.
Criterios de aceptación	• El usuario puede ejecutar las operaciones principales en menos de 3 pasos. • La curva de aprendizaje no debe superar los 30 minutos de uso guiado. • Los términos utilizados son compresibles de usuario final.

Tabla 43 NRF19 – Facilidad de uso.

Requerimiento No Funcional	**NRF20 – Usabilidad Intuitiva.**
Tipo	Usabilidad
Prioridad	Alta
Descripción	La navegación del sistema debe ser intuitiva, con menús claros, botones reconocibles y rutas de acceso consistentes en todas las pantallas.
Criterios de aceptación	• El 90% de los usuarios debe completar tareas básicas (registro, consulta, edición) sin necesidad de asistencia externa. • Los íconos y botones deben tener etiquetas o tooltips que indiquen su función. • Las rutas de navegación deben ser consistentes en todas las secciones.

Tabla 44 NRF20 – Usabilidad Intuitiva.

Requerimiento No Funcional	**NRF21 – Claridad de datos mostrados.**
Tipo	Usabilidad
Prioridad	Media
Descripción	La información desplegada en el sistema debe mostrarse de forma clara, organizada y fácilmente legible, evitando la sobrecarga visual.
Criterios de aceptación	• Los listados deben contar con filtros, paginación y ordenamiento. • Los mensajes de error y confirmación deben mostrarse en un lenguaje comprensible para el usuario. • Se deben usar tipografías legibles y contraste adecuado de colores para garantizar accesibilidad.

Tabla 45 NRF21 – Claridad de datos mostrados.

Requerimiento No Funcional	**NRF22 – Cumplimiento con Reglamento General de la Ley Orgánica de la Universidad de El Salvador.**
Tipo	Legal
Prioridad	Alta
Descripción	El sistema debe garantizar que los procesos y registros gestionados se apeguen al Reglamento General de la Ley Orgánica de la Universidad de El Salvador, asegurando que la información y documentación generada respete lo establecido en dicha normativa.
Criterios de aceptación	• Los flujos del sistema deben alinearse con los procedimientos definidos por la normativa interna de la UES. • Todo documento o registro generado por el sistema debe poder ser auditado frente al reglamento correspondiente.

Tabla 46 NRF22 – Cumplimiento con Reglamento General de la Ley Orgánica de la UES.

Requerimiento No Funcional	**NRF23 – Cumplimiento con Sistema de Escalafón Docente y Ley de Salarios UES.**
Tipo	Legal
Prioridad	Alta
Descripción	El sistema debe contemplar la normativa del Escalafón Docente y la Ley de Salarios de la Universidad de El Salvador, garantizando la correcta asignación y control de categorías, niveles y remuneraciones.
Criterios de aceptación	• La información de salarios y categorías docentes debe generarse en concordancia con las tablas vigentes del escalafón. • Los reportes de nómina deben poder ser contrastados con la normativa de la Ley de Salarios.

Tabla 47 : NRF23 – Cumplimiento con Sistema de Escalafón Docente y Ley de Salarios UES.

Requerimiento No Funcional	**NRF24 – Cumplimiento con Ley de Acceso a la Información Pública (LAIP).**
Tipo	Legal
Prioridad	Alta
Descripción	El sistema debe cumplir con la Ley de Acceso a la Información Pública (LAIP), garantizando transparencia en el acceso a información pública y protección de información reservada o confidencial.
Criterios de aceptación	• Los datos públicos deben estar disponibles mediante reportes o consultas según lo establece la ley. • La información clasificada como reservada o confidencial debe estar protegida mediante roles y permisos de acceso. • El sistema debe poder generar evidencia de cumplimiento.

Tabla 48 NRF24 – Cumplimiento con Ley de Acceso a la Información Pública (LAIP).

3.4.5 Procesos Críticos del Sistema

Dentro del Sistema de Gestión de Contratos de la FIA-UES, se identifican como procesos críticos aquellos que resultan indispensables para garantizar la continuidad y confiabilidad del flujo de contratación docente. Estos procesos requieren especial atención en las fases de validación, ya que cualquier error o interrupción puede generar retrasos significativos o inconsistencias en la gestión administrativa.

Los principales procesos críticos son:

1- **Autenticación y Control de Acceso**
   - Validación de credenciales de usuarios y asignación de roles según su perfil (Administrador, Candidato, Recursos Humanos, etc.).
   - Su criticidad radica en que asegura la confidencialidad de la información y previene accesos indebidos al sistema.

2- **Registro y Postulación de Candidatos**
   - Ingreso de datos personales, académicos y carga de documentos de respaldo.
   - Constituye el punto de partida del flujo de contratación, por lo que un fallo en este proceso impediría la postulación de nuevos docentes.

3- **Validación de Solicitudes por Recursos Humanos**
   - Revisión de la información registrada por el candidato y validación de la documentación cargada.
   - Es crítico porque garantiza que únicamente candidatos que cumplan los requisitos avancen en el proceso.

4- **Revisión Académica por el Director de Escuela**
   - Validación de la necesidad académica y asignación de carga docente.
   - Es esencial ya que define si el contrato propuesto corresponde con las necesidades de la facultad.

5- **Generación de Contratos**
   - Creación automática de contratos a partir de plantillas predefinidas, integrando datos de candidatos y validaciones previas.
   - Proceso altamente crítico, ya que constituye el producto final del sistema y su exactitud es fundamental para la relación laboral.

6- **Aprobación Final por el Decano**
   - Revisión y autorización definitiva del contrato docente.
   - Su criticidad radica en que habilita legalmente la incorporación del docente al sistema institucional.

7- **Gestión Financiera**
   - Validación de disponibilidad presupuestaria por parte del asistente financiero.
   - Es crítico porque asegura la sostenibilidad económica de los contratos autorizados.

Estos procesos constituyen la columna vertebral del sistema, ya que integran aspectos técnicos, administrativos, académicos y financieros. La correcta ejecución de los mismos es indispensable para mantener la eficiencia, transparencia y confiabilidad en la gestión de contratos de personal docente en la FIA-UES.

3.4.6 Criterios de Selección de Pruebas

Para garantizar que el proceso de pruebas se oriente a los aspectos más relevantes del Sistema de Gestión de Contratos de la FIA-UES, se definieron criterios de selección que permiten priorizar los casos a validar. Estos criterios se establecieron con base en la criticidad de los procesos, los riesgos asociados, la frecuencia de uso y la relevancia institucional de cada módulo.

Los principales criterios son los siguientes:

1- **Criticidad del proceso**
   - Se priorizaron los procesos identificados como críticos (autenticación, registro de candidatos, validación de solicitudes, generación de contratos y aprobación final).
   - Esto asegura que los componentes esenciales del sistema mantengan su estabilidad y confiabilidad.

2- **Frecuencia de uso**
   - Los módulos y funciones que se utilizan de manera recurrente durante el ciclo académico (como inicio de sesión, registro de datos y validación de documentos) fueron seleccionados para pruebas exhaustivas.
   - Este criterio se basa en el impacto directo en la operatividad diaria de la facultad.

3- **Impacto institucional**
   - Se seleccionaron pruebas en procesos que tienen incidencia en decisiones administrativas y académicas de alto nivel, como la validación de carga docente y la aprobación de contratos.
   - Esto garantiza que el sistema respalde de manera confiable las funciones estratégicas de la FIA-UES.

4- **Riesgos asociados**
   - Se incluyeron pruebas específicas en áreas vulnerables a errores o ataques, como autenticación de usuarios y seguridad de la información.
   - Este criterio busca mitigar riesgos de accesos indebidos o inconsistencias en los datos.

5- **Automatización de flujos repetitivos**
   - Se priorizó la automatización en aquellos procesos que presentan alta repetitividad (login, generación de contratos, validaciones), con el fin de mejorar la eficiencia y reducir el riesgo de error humano.

6- **Cobertura de funcionalidades**
   - La selección de pruebas se realizó considerando la necesidad de cubrir las funcionalidades principales del sistema, garantizando una validación integral y no únicamente puntual.

La selección de pruebas se fundamenta en la combinación de factores técnicos, operativos e institucionales, con el fin de asegurar que las validaciones se enfoquen en los procesos de mayor valor y riesgo para la FIA-UES. Este enfoque asegura que los resultados del proyecto no solo evidencien la funcionalidad del sistema, sino también su confiabilidad, seguridad y sostenibilidad.

3.5 Matriz de pruebas del Sistema de Gestión de Contratos

Para la creación de la matriz de prueba seguimos los lineamientos estipulados en la ISO/IEC 29119 agregando los campos que nos permitan poder realizar la ejecución de la prueba, tener trazabilidad y evidencia de los casos. A continuación, se listan los campos utilizados en la matriz:

a. **ID_CASO:** Identificador único de la prueba.
b. **Modulo:** Módulo del sistema al que pertenece la prueba.
c. **Nombre del caso de prueba:** Nombre identificado del caso de prueba.
d. **Nivel de prueba:** Nivel de la prueba a evaluar.
e. **Tipo de prueba**: Tipo de la prueba a evaluar.
f. **Descripción:** objetivo de la ejecución de la prueba.
g. **Pasos:** Pasos a seguir para la ejecución de la prueba.
h. **Precondiciones:** condiciones que deben encontrarse para ejecutar la prueba con sin impedimentos.
i. **Datos de entrada:** Datos a ingresar para la ejecución de la prueba
j. **Resultado esperado**: Resultado que se espera al culminar la ejecución de la prueba.
k. **Nivel de prioridad:** Prioridad de importancia del caso de prueba en el sistema.
l. **Estado**: Estado de ejecución del caso de prueba.
m. **Evidencias:** Evidencias del resultado obtenido de las pruebas.
n. **Fecha inicio de prueba**: Fecha en que se inició la ejecución de la prueba.
o. **Fecha finalización de prueba:** Fecha en que se terminó la ejecución de la prueba.
p. **Duración de la prueba:** tiempo invertido en la ejecución de la prueba.

A partir de estos campos nos permitirá tener un control general de gestión que se realizará para cada caso de prueba.

Debido a la proporción de la matriz de pruebas se colocará un resumen con su identificador único y su nombre correspondiente sin embargo para mayor detalle se comparte enlace al archivo Excel que concentra los casos de pruebas:

Matriz de pruebas: MATRIZ DE PRUEBAS.xlsx

Ver en Anexos tabla de casos de pruebas.

3.5.1 Casos Seleccionados para Automatizar

A partir de la matriz de pruebas funcionales y no funcionales elaborada para el Sistema de Gestión de Contratos de Personal Docente, se seleccionaron los casos de prueba que serán automatizados en esta primera fase. La selección se realizó considerando principalmente los siguientes criterios:

- **Criticidad del proceso de negocio:** se priorizaron los casos asociados a funcionalidades clave para la FIA-UES, como la autenticación de usuarios, la gestión de contratos, la administración de ciclos académicos y la bitácora de acciones, ya que cualquier falla en estos módulos impacta directamente en la operación del sistema.

- **Frecuencia de ejecución y regresión:** se eligieron casos que se ejecutan de forma recurrente en cada ciclo de pruebas (por ejemplo, creación, edición y eliminación de registros, listados y filtros), de forma que la automatización reduzca el esfuerzo manual en futuras iteraciones de mantenimiento y liberación.
- **Estabilidad de los requisitos y de la interfaz:** se seleccionaron casos cuyos requisitos funcionales ya se encuentran consolidados y cuya interfaz o endpoints presentan pocos cambios, de manera que los scripts automatizados tengan una vida útil más larga y no requieran ajustes constantes.
- **Resolución Determinística:** se priorizaron escenarios con pasos bien definidos, datos de entrada controlables y resultados esperados claros (códigos de estado HTTP, mensajes de éxito, estructuras de respuesta JSON, redirecciones esperadas, etc.), lo que facilita la automatización y la verificación objetiva de resultados.
- **Compatibilidad Técnica:** se consideró la posibilidad de implementar los scripts utilizando las herramientas seleccionadas en el proyecto (por ejemplo, Postman/Newman para pruebas de API y Selenium + Pytest para pruebas de interfaz web), así como la facilidad de integración con el entorno de pruebas existente.

Con base en estos criterios, se definió como alcance de la automatización los casos de prueba relacionados con:

- Inicio de sesión y manejo de autenticación/autorización de usuarios.
- Operaciones CRUD sobre entidades críticas (usuarios, personas, docentes, contratos, ciclos académicos, etc.).
- Listados principales con paginación y filtros, que se usan de forma frecuente en el trabajo diario del personal administrativo.
- Registro y consulta de la bitácora de trazabilidad de acciones, para asegurar que las operaciones realizadas por los usuarios queden correctamente registradas.

En la Matriz de Pruebas se marcaron estos casos con el indicador correspondiente (por ejemplo, una columna “Automatizar: Sí”), de forma que quede claramente documentado qué identificadores de caso forman parte de la suite automatizada y sobre qué módulos del sistema se enfocará esta primera etapa de automatización.

Cuadro de pruebas a automatizar: Pruebas a automatizar

3.5.2 Casos que No se Automatizarán

Aunque la matriz de pruebas contempla un conjunto amplio de casos sobre los diferentes módulos del sistema, no todos son candidatos adecuados para automatización en esta fase del proyecto. Existen casos que, por su naturaleza o por restricciones de tiempo y recursos, se mantendrán como pruebas manuales. Entre las principales razones para no automatizarlos se encuentran:

- **Baja frecuencia de ejecución**: algunos escenarios se ejecutan únicamente en situaciones excepcionales (por ejemplo, configuraciones iniciales, tareas administrativas poco frecuentes o procesos especiales de cierre), por lo que el costo de automatizarlos no se justifica frente al beneficio obtenido.

- **Alta variabilidad de datos o del flujo**: ciertos casos dependen de decisiones discrecionales de las autoridades o de datos que cambian constantemente (por ejemplo, criterios particulares de contratación, observaciones cualitativas o flujos que cambian según cada periodo académico), lo que dificulta definir scripts estables.
- **Dependencia de validaciones manuales o visuales:** pruebas relacionadas con la experiencia de usuario, revisión de formatos de documentos, apariencia de interfaces o lectura de archivos generados (por ejemplo, revisión estética de reportes o comprobantes) requieren juicio humano, por lo que se consideran más adecuados para ejecución manual.
- **Entornos o integraciones no controladas:** algunos escenarios implican interacción con sistemas externos o servicios que no están disponibles en un ambiente de pruebas controlado, lo que limita su automatización confiable en esta etapa.
- **Limitaciones de tiempo y alcance del proyecto:** dado que el objetivo de la tesina es demostrar la implementación de automatización sobre los flujos más críticos, se decidió dejar fuera de esta fase ciertos casos de baja prioridad o alto esfuerzo técnico, los cuales podrán considerarse para futuras iteraciones de mejora continua.

Estos casos se encuentran igualmente documentados en la Matriz de Pruebas, pero se identifican como “solo manuales” (por ejemplo, con un indicador “Automatizar: No”). De esta manera, se asegura que el plan de pruebas mantenga un equilibrio entre cobertura mediante automatización en los flujos críticos y flexibilidad mediante pruebas manuales donde la intervención humana sigue siendo necesaria o más eficiente.

3.6 ID y Objetivos de las Pruebas

Cada caso de prueba se identifica mediante un código único, que permite una adecuada gestión, trazabilidad y reporte durante el ciclo de pruebas. Este identificador está compuesto por un prefijo relacionado con el módulo o funcionalidad bajo prueba, seguido por un número correlativo. Por ejemplo, IDs como CP_106 o CP_64 reflejan casos específicos dentro de módulos como Usuarios, Contratos, o Carga Académica. Esta identificación estandarizada facilita la referencia precisa, la ejecución controlada y el seguimiento de resultados.

Objetivos específicos de las pruebas

Los objetivos específicos de las pruebas son los siguientes:

- Validar la correcta funcionalidad de cada módulo y proceso crítico, asegurando que responden acorde con los requerimientos definidos, tales como creación y gestión de usuarios, generación de contratos, y administración de carga académica.
- Garantizar la seguridad del sistema, verificando controles de acceso, autenticación de usuarios, y protecciones contra vulnerabilidades comunes (inyecciones, XSS, enumeración de cuentas).
- Confirmar la integridad y confidencialidad de la información transmitida y almacenada mediante el uso de protocolos seguros (HTTPS) y configuración adecuada de cabeceras y políticas de seguridad.
- Evaluar la eficiencia y estabilidad del sistema en funciones clave, a través de casos que revisan la respuesta del sistema bajo condiciones esperadas y excepcionales.

- Controlar la calidad y estabilidad de la interfaz de usuario, validando la correcta presentación de datos, mensajes de error, y restricciones de acceso a información sensible.
- Asegurar la correcta integración entre módulos, validando los flujos de trabajo y procesos automatizados para minimizar errores y facilitar el cumplimiento de objetivos del sistema.

Estos objetivos están alineados con las mejores prácticas internacionales para la gestión y aseguramiento de la calidad del software, tales como la norma ISO/IEC 29119, lo que asegura un enfoque formal y riguroso durante la ejecución de las pruebas.

3.7 Alcance de Pruebas

El alcance de las pruebas del Sistema de Gestión de Contratos de Personal Docente se estructura en torno a los módulos críticos identificados en el plan de pruebas: registro de candidatos, validación de documentos, generación de contratos y asignación de carga académica; además de incluir módulos de soporte como autenticación, validación de ciclos, actividades de docentes y bitácora. En cada módulo se combinan pruebas manuales y automatizadas, de acuerdo con la priorización establecida y con los recursos disponibles para el proyecto.

3.7.1 Módulo de Autenticación y Control de Acceso (login)

En este módulo se valida el proceso de inicio y cierre de sesión, así como la correcta aplicación de permisos según el rol del usuario. El alcance incluye pruebas funcionales manuales para verificar credenciales válidas e inválidas, manejo de mensajes de error, recuperación de acceso cuando aplique y restricciones de acceso a módulos protegidos.

Adicionalmente, se automatizan casos de prueba de login considerados críticos y repetitivos utilizando Selenium WebDriver, con el objetivo de contar con un conjunto de pruebas de regresión que permita verificar rápidamente el acceso al sistema cada vez que se despliega una nueva versión. Estas pruebas se complementan con verificaciones básicas de seguridad relacionadas con el uso de sesiones y la protección de rutas restringidas.

3.7.2 Módulo de Registro de Candidatos

En el módulo de registro de candidatos se prueba la capacidad del sistema para almacenar de forma correcta y consistente la información personal, académica y laboral de los candidatos. El alcance comprende pruebas funcionales manuales sobre: creación de nuevos registros, edición de datos existentes, validación de campos obligatorios, formatos de correo electrónico y documentos, así como manejo de mensajes de error cuando la información es incompleta o incorrecta.

En la API correspondiente se realizan pruebas con Postman para verificar que los endpoints de creación, actualización y consulta de candidatos respondan con el código de estado adecuado, apliquen las validaciones definidas y devuelvan la estructura de datos esperada. Se incluyen también pruebas negativas (datos inválidos, falta de autenticación, permisos insuficientes) para comprobar el manejo correcto de errores en el backend.

3.7.3 Módulo de Validación de Documentos

En este módulo se evalúa el proceso mediante el cual el área de Recursos Humanos revisa y valida los documentos adjuntos por el candidato. El alcance contempla pruebas manuales para verificar: carga y visualización de archivos, cambio de estados de los documentos (aprobado, rechazado, pendiente), registro de observaciones y actualización de la situación general del candidato.

A nivel de API se diseñan casos con Postman para comprobar que los servicios relacionados con la validación de documentos permiten actualizar estados de forma coherente, registran correctamente quién realizó la validación y mantienen la integridad entre la información del candidato y los documentos asociados. Se realizan también pruebas de permisos para asegurar que solo usuarios con rol adecuado puedan modificar estados de documentos.

3.7.4 Módulo de Generación de Contratos

El módulo de generación de contratos se considera uno de los más críticos del sistema. El alcance de pruebas incluye la verificación de la creación de contratos a partir de solicitudes aprobadas, la incorporación correcta de los datos del docente, ciclo académico, carga asignada, escalafón y demás parámetros necesarios, así como la generación del documento en el formato previsto (por ejemplo, PDF).

Se ejecutan pruebas funcionales manuales sobre la interfaz y pruebas automatizadas con Selenium para los flujos más repetitivos, como la generación de contratos a partir de solicitudes con diferentes combinaciones de carga académica. Además, se utilizan pruebas de API en Postman para comprobar que los endpoints de generación y consulta de contratos devuelvan información completa y consistente. Este módulo también es objeto de pruebas de rendimiento básicas, midiendo el tiempo de respuesta al generar contratos en serie.

3.7.5 Módulo de Asignación de Carga Académica y Validación de Ciclos

En este módulo se valida la asignación de asignaturas, grupos y horas a los docentes, así como la coherencia de dicha asignación con los ciclos académicos definidos. El alcance incluye pruebas manuales para: creación y modificación de registros de carga académica, verificación de restricciones básicas (por ejemplo, horas máximas por docente o por materia, cuando correspondan) y consulta de la carga asignada por ciclo.

Se automatizan casos específicos relacionados con la validación de ciclos, considerados críticos en el plan de pruebas, verificando que solo se utilicen ciclos habilitados y que la información asociada a cada ciclo se refleje de forma correcta en contratos y reportes. Estas pruebas se apoyan tanto en Selenium como en Postman, según corresponda a la interfaz o a la API. Para endpoints que devuelven grandes listados de carga académica, se incluyen pruebas de rendimiento con JMeter, enfocadas en tiempos de respuesta bajo una carga moderada.

3.7.6 Módulo de Actividades de Docentes y Bitácora de Acciones

El módulo de actividades de docentes y la bitácora de acciones dan soporte a la trazabilidad del sistema. El alcance de pruebas comprende: registro de actividades vinculadas a contratos y ciclos, consulta de listados con filtros por docente, ciclo y tipo de actividad, además

de la verificación de que la bitácora almacena información relevante sobre las operaciones realizadas (usuario, fecha, acción y módulo afectado).

Se ejecutan pruebas funcionales manuales sobre la interfaz, verificando filtros, paginación y ordenamiento, y pruebas de API con Postman para los servicios de consulta y registro. En el caso de la bitácora se enfatiza el control de acceso, comprobando que solo perfiles autorizados (por ejemplo, Administrador o roles equivalentes) puedan acceder al listado completo. Cuando la cantidad de registros es grande, se aplican pruebas de rendimiento focalizadas para evaluar el comportamiento de los listados y la respuesta de la API.

3.8 Estrategia de Pruebas (Manuales, Automatizadas, Iteraciones)

3.8.1 Independencia de la Prueba

El principio de independencia de las pruebas establece que quienes diseñan y ejecutan los casos de prueba deben ser, en la medida de lo posible, diferentes de quienes desarrollan el sistema. Esto tiene como finalidad evitar sesgos, mejorar la objetividad en la evaluación y aumentar la tasa de detección de errores.

En este proyecto, aunque el equipo es reducido, se han establecido medidas para promover dicha independencia. Por ejemplo:

- Los casos de prueba serán elaborados por miembros del equipo que no codificaron directamente el módulo a evaluar.
- La validación de resultados será realizada por un tercer integrante, utilizando una matriz de trazabilidad que permite comprobar la cobertura de requerimientos.
- Se documentará cualquier defecto identificado por parte del equipo de pruebas, sin que el desarrollador sea el encargado de validar su corrección.

Estas medidas contribuyen a la transparencia del proceso de calidad, reducen la posibilidad de omitir errores por familiaridad con el código, y cumplen con las recomendaciones de la norma ISO/IEC 29119 y del ISTQB respecto a los niveles de independencia sugeridos en proyectos de software.

3.8.2 Enfoque del Proceso de Prueba

Para la selección del navegador se realizó una investigación considerando diferentes fuentes de información sobre el consumo y tráfico web por tipo de navegador. Según el reporte actualizado de W3Counter (31/07/2025), el navegador con mayor uso a nivel mundial es Google Chrome, con una participación del 63.10%. Por lo tanto, se determinó que este será el navegador principal en el que se ejecutarán las pruebas del sistema, asegurando así mayor representatividad y realismo en la validación.

El enfoque de pruebas se definió tomando como referencia:

- Las historias de usuario del sistema.
- Los objetivos establecidos en el proyecto.
- Las mejores prácticas recomendadas por ISTQB para asegurar calidad y trazabilidad.

Estrategias de Prueba Adoptadas

1- **Estrategia de Prueba Analítica (Analytical Test Strategy)**
   - Basada en el análisis detallado de la base de prueba (historias de usuario, criterios de aceptación y flujos de negocio).
   - Permite diseñar casos de prueba con cobertura integral de requisitos.
   - Facilita la priorización de los casos más relevantes y de mayor impacto para la universidad.

2- **Estrategia Basada en Modelos (Model-based Test Strategy)**
   - Utiliza modelos formales y conceptuales para representar aspectos críticos del sistema, principalmente en pruebas no funcionales.
   - Permite derivar casos de prueba directamente de los modelos, asegurando una cobertura estructurada y sistemática.
   - Contribuye a garantizar la calidad en áreas sensibles como rendimiento, seguridad y comportamiento del sistema.

3.9 Niveles de Pruebas y Fases del Proyecto

3.9.1 Niveles de Pruebas

Para garantizar la calidad del Sistema Informático para la Gestión de Contratos de Personal Docente (SIGC-FIA), se han definido los siguientes niveles de prueba que se llevaran a cabo en este proyecto, abarcando desde la validación técnica de componentes hasta la aceptación final por parte de los usuarios.

Nivel de Prueba	Objetivo	Ejemplo en el SIGC-FIA
Prueba de Componentes	Validar el correcto funcionamiento de cada módulo individual del sistema.	Verificar que el módulo de registro de candidatos almacene correctamente la información.
Pruebas de Integración	Evaluar la comunicación y coherencia de datos entre los diferentes módulos.	Comprobar que los datos de un candidato registrado se reflejen en la asignación de clases.
Pruebas de Sistema	Validar el comportamiento del sistema completo frente a requerimientos.	Ejecutar todo el proceso de contratación docente desde el registro hasta la firma.
Pruebas de Seguridad	Verificar la protección de información sensible y resistencia a ataques.	Confirmar que solo usuarios autorizados accedan a escalafones y que contraseñas estén cifradas.
Pruebas de Usabilidad	Evaluar la facilidad de uso, accesibilidad y experiencia del usuario.	Comprobar que los formularios sean intuitivos y que los mensajes de error sean claros.
Pruebas de Aceptación	Confirmar que el sistema cumple las expectativas y está listo para producción.	Validación por autoridades y personal administrativo sobre los procesos de contratación.

Tabla 46: Niveles de Pruebas a Implementar.

3.9.2 Tipos de Pruebas

Para lograr una validación integral del sistema, se emplearán diversos tipos de pruebas, seleccionados en función del contexto operativo, los riesgos identificados y la frecuencia de uso de cada funcionalidad. A continuación, se describen los tipos de pruebas incluidos en el plan:

- **Pruebas funcionales**: Verifican que cada módulo cumpla con los requerimientos definidos en la especificación funcional. Se basan en criterios de caja negra y comprueban que el sistema actúe conforme a lo esperado.
- **Pruebas de integración**: Evalúan que los módulos del sistema se comuniquen correctamente entre sí. Por ejemplo, se probará que la selección de un ciclo académico influya correctamente en la generación de contratos y en la asignación de docentes.
- **Pruebas de regresión**: Se ejecutarán para confirmar que las actualizaciones o correcciones no afecten negativamente otras funcionalidades ya validadas previamente. Estas pruebas serán esenciales conforme el sistema se modifique o se realicen mejoras.
- **Pruebas automatizadas**: Se utilizarán en funcionalidades repetitivas o críticas (como validaciones de login, generación de contratos o verificación de ciclos), usando herramientas como PHPUnit y Postman. Estas pruebas permiten acelerar el proceso y mejorar la precisión de las validaciones en futuras iteraciones.
- **Pruebas exploratorias**: Permiten descubrir errores no cubiertos por los casos de prueba formales, simulando el uso libre e intuitivo del sistema por parte de un usuario final. Serán especialmente útiles para validar la usabilidad y robustez del sistema en condiciones no previstas.

Cada tipo de prueba estará respaldado por una especificación técnica y evidencias de su ejecución.

3.9.3 Fases del Proyecto

En el marco del proyecto de tesina, los niveles de prueba definidos se aplicaron de manera progresiva a lo largo de las distintas fases establecidas en el calendario de pruebas (sección 2.18).

En una fase de planificación y diseño, se definieron los casos de prueba a partir de los requisitos funcionales y no funcionales del sistema, asignando a cada uno el nivel de prueba correspondiente (componente, integración, sistema, seguridad o aceptación). Esta fase permitió estructurar la matriz de pruebas y priorizar los escenarios críticos del proceso de contratación docente.

Posteriormente, en la fase de preparación del entorno, se configuraron los ambientes de prueba, tanto a nivel local como en el servidor, y se cargaron los datos necesarios para la ejecución de los casos de prueba, asegurando condiciones controladas y reproducibles.

La fase de ejecución manual se centró en la aplicación de pruebas funcionales, de integración y de sistema sobre los módulos más relevantes del proceso de contratación. Durante esta etapa se registraron los resultados obtenidos y se documentaron los defectos detectados, generando la base para los análisis posteriores.

En la fase de ejecución automatizada y de rendimiento, se llevaron a cabo los casos de prueba seleccionados para automatización, así como las pruebas de desempeño sobre los listados y procesos más utilizados. Esta fase permitió validar de manera más eficiente el comportamiento del sistema ante múltiples ejecuciones y escenarios de carga.

Finalmente, en la fase de análisis y cierre, se revisaron de forma consolidada los resultados de todos los niveles de prueba, se calcularon métricas, se actualizaron los riesgos asociados y se documentaron los hallazgos en el acta de cierre de pruebas, proporcionando una visión global del estado de calidad del sistema.

3.9.4 Logística de Pruebas de Integración

Las pruebas de integración tuvieron como objetivo verificar la correcta interacción entre los distintos módulos del Sistema de Gestión de Contratos de Personal Docente, asegurando que los datos fluyeran de forma coherente a lo largo del proceso de contratación. En el plan de pruebas se definió que estas pruebas abarcarían escenarios donde intervienen múltiples entidades, tales como usuarios, candidatos, ciclos académicos, escalafones, actividades y contratos.

Para su ejecución se utilizaron los ambientes descritos en el plan de pruebas: un entorno local de desarrollo y un servidor de pruebas en la nube, configurados con versiones equivalentes del sistema y bases de datos preparadas específicamente para este proyecto.

La logística de las pruebas de integración se organizó de la siguiente manera:

- Primero se verificó que los módulos individuales funcionaran de forma aceptable mediante pruebas funcionales básicas, de manera que los defectos de componente no interfirieran en la evaluación de la integración.
- Posteriormente, se ejecutaron casos de prueba que involucraban el flujo entre módulos, por ejemplo: registro de un candidato, asignación de su información académica, asociación de actividades docentes y generación del contrato correspondiente.
- Se utilizaron datos de prueba consistentes entre los distintos módulos, de modo que las entidades creadas en un proceso (por ejemplo, un ciclo académico o un grupo) se reutilizaran en otros casos de prueba de integración.

La ejecución de estas pruebas se llevó a cabo en ciclos, siguiendo el calendario de pruebas definido en el proyecto. En cada ciclo se priorizaron los escenarios de integración asociados a procesos críticos del negocio, como la gestión de contratos por ciclo académico, la asignación de carga docente y la actualización de datos de candidatos. Los resultados y defectos detectados se documentaron utilizando el procedimiento de administración de defectos descrito en el plan (sección 14.8), lo que permitió dar seguimiento a las incidencias hasta su corrección o cierre.

3.9.5 Logística de Pruebas de Aceptación de Usuario (UAT)

Además de la descripción cualitativa de los flujos de prueba y de las actividades de aceptación de usuario, fue necesario contar con indicadores cuantitativos que permitieran evaluar de manera objetiva el desempeño del sistema y la efectividad del proceso de pruebas. En este contexto, se definió un conjunto de métricas específicas para el proyecto, orientadas a medir la

calidad del software desde distintas dimensiones: la severidad de los defectos encontrados, el grado de éxito de los casos ejecutados, la cobertura alcanzada sobre la funcionalidad del sistema y la capacidad del equipo de pruebas para detectar incidencias de forma oportuna.

Estas métricas constituyen un complemento fundamental a los resultados observados durante la ejecución de los flujos end-to-end y de las pruebas de aceptación, ya que proporcionan una base numérica para valorar el estado del sistema y respaldar las decisiones sobre su aceptación académica y las recomendaciones de mejora futura.

3.9.5.1 Métricas de Prueba

Las métricas las obtendremos una vez ejecutadas las actividades de las pruebas, estas métricas nos permitirán medir el nivel actual del sistema conforme a las funciones que cuenta. Las métricas se han definido basados en los estándares y practicas recomendadas por la certificación ISTQB.

1. **Defectos por severidad**.
   Definición: Distribución de defectos según su impacto en el negocio (Critico, Alto, Medio o Bajo).
   Fórmula: $\frac{Numero\ de\ defectos\ por\ severidad}{Total\ de\ defectos} \times 100$
   Objetivo: cero defectos críticos al cierre.
2. **Porcentaje de casos exitosos**.
   Definición: proporción de casos ejecutados que cumplen con el resultado esperados.
   Fórmula: $\frac{Casos\ exitosos}{Total\ de\ casos\ ejecutados} \times 100$
   Objetivo: se espera un umbral ≥ 95% en casos.
3. **Cobertura de ejecución de pruebas**.
   Definición: permite tener una idea del número total de casos de prueba ejecutados en comparación con el número de casos de prueba pendientes.
   Fórmula: $\frac{Número\ total\ de\ casos\ de\ prueba\ ejecutados}{Numero\ total\ de\ casos\ de\ prueba} \times 100$
   Objetivo: se espera completar en un 100% los casos de pruebas.
4. **Cobertura Funcional**.
   Definición: porcentaje de funcionalidades planificadas que han sido probadas.
   Fórmula: $\frac{Funcionalidades\ probadas}{Total\ de\ funcionalidades} \times 100$
   Objetivo: se espera un umbral ≥ 90% antes de la fase UAT.
5. **Eficiencia de detección**.
   Definición: porcentaje de defectos detectados antes del pase a producción.
   Fórmula: $\frac{Defectos\ QA}{Total\ de\ defectos\ detectos} \times 100$
   Objetivo: umbral ≥ 85% de defectos detectados antes de producción.

Esta logística permitió comprobar no solo que cada módulo funcionara de forma aislada, sino que el sistema, como conjunto, mantuviera la consistencia de la información y soportara correctamente el flujo completo de contratación docente desde un punto de vista operativo.

3.10 Áreas de Enfoque de Prueba

Las actividades de prueba del Sistema de Gestión de Contratos de Personal Docente se concentraron en tres grandes áreas de enfoque, definidas a partir del diagnóstico del proyecto, de los módulos priorizados y de las herramientas seleccionadas en el plan de pruebas:

- **Pruebas funcionales por módulo,** orientadas a verificar el cumplimiento de los requisitos de negocio en los flujos críticos de contratación.
- **Pruebas estructurales**, apoyadas de forma acotada en análisis estático de código y en la revisión de la organización de la API y de los datos.
- **Pruebas no funcionales,** con énfasis en rendimiento y en seguridad básica, alineadas con los casos de prueba definidos en la matriz (incluyendo CP_45 y CP_97).

Estas áreas de enfoque se corresponden con los objetivos del proyecto de tesina: validar el funcionamiento de los módulos clave del sistema, aprovechar herramientas seleccionadas como Selenium WebDriver, Postman, JMeter, OWASP ZAP y SonarQube, y generar evidencia suficiente para valorar la calidad del sistema en el contexto de la FIA-UES.

3.10.1 Pruebas Funcionales por Módulo (Contratación, Actividades de Docentes, Bitácora, etc.)

Las pruebas funcionales constituyen el núcleo del trabajo realizado. Se diseñaron y ejecutaron casos de prueba manuales y automatizados sobre los módulos que soportan directamente el proceso de contratación docente, de acuerdo con el alcance definido en la matriz de casos de prueba y en el plan de pruebas.

Los módulos funcionales principales fueron:

1- **Módulo de Contratación**

Agrupa los flujos centrales del sistema:

- Registro y actualización de candidatos,
- Carga y validación de documentos,
- Creación de solicitudes de contratación,
- Asignación de carga académica por ciclo,
- Generación de contratos a partir de solicitudes y acuerdos de Junta Directiva.

Sobre este módulo se ejecutaron pruebas funcionales manuales en la interfaz web (validación de formularios, estados, mensajes de error) y en la API, utilizando Postman para verificar códigos de estado, estructura de respuestas y reglas de validación. Los flujos considerados más críticos y repetitivos, como login, validación de ciclos y generación de contratos, fueron seleccionados para automatización con Selenium WebDriver, con el fin de contar con una base de regresión automatizada.

2. **Módulo de Actividades de Docentes**

Se verificó el registro, actualización y consulta de actividades asociadas a contratos y ciclos académicos. Las pruebas se centraron en:

- Creación de actividades,

- Uso de filtros por docente, ciclo y tipo de actividad,
- Coherencia entre la información mostrada y los datos almacenados.

Estas pruebas se ejecutaron principalmente de manera manual, registrando resultados y evidencias en la matriz de casos de prueba.

3. **Módulo de Bitácora de acciones**

   Este módulo es clave para la trazabilidad del sistema. Se probaron:

   - El registro automático de acciones relevantes (creación, actualización, eliminación, generación de contratos),
   - La visualización de la bitácora con filtros y paginación,
   - Control de acceso, verificando que solo perfiles autorizados (por ejemplo, Administrador o roles equivalentes) puedan consultar el historial completo.

   Se combinaron pruebas manuales desde la interfaz con pruebas de API en Postman, comprobando la estructura y contenido de las respuestas JSON.

4. **Módulo de Autenticación y control de acceso**

   Aunque no se presenta como módulo de negocio independiente, el login y la gestión de roles constituyen un prerequisito para la ejecución del resto de pruebas. Se validó el inicio y cierre de sesión, el manejo de credenciales inválidas y la restricción de acceso a funcionalidades según el rol, combinando casos manuales y automatizados con Selenium.

   En conjunto, las pruebas funcionales por módulo aseguraron que los flujos de negocio definidos en el modelado de negocio se ejecutaran de forma correcta y coherente con los requisitos documentados en el proyecto.

3.11 Criterios de Entrada y Salida de las Pruebas

Los criterios de entrada y salida permiten controlar de forma objetiva el avance de las fases de prueba del Sistema de Gestión de Contratos de Personal Docente, asegurando que cada etapa se inicie con las condiciones mínimas necesarias y se considere finalizada únicamente cuando se han cumplido los objetivos definidos en el plan de pruebas.

En la práctica, y de acuerdo con el alcance real del proyecto de tesina, las actividades de prueba se organizaron en las siguientes fases principales:

- **Preparación y diseño de pruebas.**
  - Pruebas de sistema, que incluyeron tanto la validación funcional como la verificación de la interacción entre módulos (en lugar de una fase separada de pruebas de integración).
- Pruebas de aceptación de usuario (UAT).
- Pruebas de rendimiento y seguridad.

3.11.1 Preparación y Diseño de Pruebas

Criterios de entrada

La fase de preparación y diseño de pruebas se inicia cuando:

- Se cuenta con una versión funcional del Sistema de Gestión de Contratos de Personal Docente suficientemente estable para ser utilizada como base del plan de pruebas.

- Están disponibles los documentos de referencia: diagnóstico del proyecto, requisitos funcionales y no funcionales, descripción de módulos y modelo de negocio.
- Se ha definido el alcance general de las pruebas (módulos críticos, tipos de prueba, herramientas a utilizar), de acuerdo con lo planteado en el proyecto.

Criterios de salida

La fase se considera completada cuando:

- El plan de pruebas está documentado y validado académicamente por la docente guía, incluyendo objetivos, alcance, niveles de prueba considerados, riesgos y entregables.
- Se dispone de una matriz inicial de casos de prueba que cubre los módulos y flujos críticos (contratación, actividades de docentes, bitácora, autenticación, etc.).
- Se han seleccionado y configurado las herramientas principales (Postman, Selenium, JMeter y, en el alcance previsto, SonarQube/OWASP ZAP).
- Se han identificado las tareas de prueba planificadas y su integración con el calendario general del proyecto

3.11.2 Fase de Pruebas de Sistema

En este proyecto no se ejecutó una fase independiente de pruebas de integración. Las verificaciones sobre la interacción entre módulos (por ejemplo, entre candidatos, documentos, contratos, carga académica y bitácora) se incorporaron dentro de las pruebas de sistema, mediante flujos end-to-end que recorren varias funcionalidades encadenadas.

Criterios de entrada

Las pruebas de sistema se inician cuando:

- La versión del sistema desplegada en el entorno de pruebas es estable y permite ejecutar los flujos completos de negocio sin fallos bloqueantes conocidos.
- Se han definido los flujos end-to-end representativos del proceso de contratación docente, tales como:
    - Registro de candidato → carga de documentos → validación por Recursos Humanos,
    - Solicitud de contratación → acuerdos de Junta Directiva → generación de contrato,
    - Asignación de carga académica → generación de contratos → registro en bitácora.
- Están disponibles los usuarios y roles necesarios (Administrador, Recursos Humanos, Director de Escuela, Asistente Administrativo, etc.) para ejecutar dichos flujos.

Criterios de salida

La fase de pruebas de sistema se considera completada cuando:

- Se han ejecutado los flujos de negocio definidos y sus casos de prueba asociados (manuales y, cuando aplica, automatizados con Selenium y Postman).

- El porcentaje de casos de prueba de sistema aprobados alcanza el umbral establecido en el plan (por ejemplo, ≥ 90 % de los casos ejecutados).
- No se mantienen defectos de severidad Crítica o Alta que afecten los flujos completos de contratación docente o la integridad de los datos.
- Se cuenta con evidencias documentadas (capturas, reportes, registros de ejecución) que respalden los resultados de la fase.

3.11.3 Fase de Pruebas de Aceptación de Usuario (UAT)

Criterios de entrada

Las pruebas de aceptación de usuario se realizan cuando:

- La fase de pruebas de sistema ha alcanzado un grado de estabilidad suficiente y no existen defectos críticos abiertos en los flujos que se presentarán a los usuarios clave.
- Se han definido los criterios de aceptación para cada flujo que se evaluará, alineados con los requisitos del proyecto y con las necesidades del proceso de contratación en la FIA-UES.

Criterios de salida

La fase de UAT se considera finalizada cuando:

- Se han revisado los flujos definidos y confirmado que el comportamiento del sistema satisface los criterios de aceptación acordados.
- Las observaciones realizadas han sido documentadas; si corresponden a defectos, se han registrado como incidencias con su severidad y prioridad.
- No existen observaciones de tipo crítico que impidan el uso del sistema en el contexto de la contratación docente.

3.11.4 Fase de Pruebas de Rendimiento y Seguridad

Criterios de entrada

Las pruebas de rendimiento y seguridad se inician cuando:

- El sistema se encuentra en un estado estable, apto para ser sometido a cargas de uso mayores que las de una ejecución manual aislada.
- Se han identificado los endpoints y listados críticos para las pruebas de rendimiento (por ejemplo, listados de contratos, candidatos, carga académica y bitácora).
- Se han definido los casos de prueba de seguridad incluidos en la matriz (como CP_45 y CP_97), detallando los escenarios a verificar (inyección, exposición de datos, control de acceso).
- Herramientas como JMeter y, cuando corresponde, OWASP ZAP, están configuradas para apuntar al entorno de pruebas.

Criterios de salida

La fase se considera finalizada cuando:

- Se han ejecutado los planes de prueba de rendimiento definidos, obteniendo métricas de tiempos de respuesta y tasa de errores sobre los endpoints seleccionados.
- Los resultados muestran tiempos de respuesta y niveles de error aceptables para el contexto de uso previsto, o bien se han documentado las limitaciones encontradas como hallazgos del proyecto.
- Se han ejecutado las verificaciones de seguridad planificadas (control de acceso, pruebas negativas, casos CP_45 y CP_97), sin identificar vulnerabilidades críticas en los flujos principales.

Los hallazgos de rendimiento y seguridad se han incorporado al informe de resultados, junto con recomendaciones para mejoras futuras

3.12 Definición de Defectos y Tiempos de Respuesta

Para gestionar de forma ordenada los resultados de las pruebas del Sistema de Gestión de Contratos de Personal Docente, se definió un esquema común para el registro y tratamiento de defectos, que incluye la clasificación por severidad, prioridad y la estimación de tiempos de respuesta esperados. Esto permite dar trazabilidad a los hallazgos, facilitar su análisis y apoyar la toma de decisiones sobre qué corregir primero dentro de las limitaciones de tiempo del proyecto de tesina.

3.12.1 Definición de Defecto

En el contexto de este proyecto, se entiende por defecto (o incidencia) cualquier comportamiento del sistema que:

- Se desvíe del resultado esperado definido en un caso de prueba,
- Incumpla un requisito funcional o no funcional documentado,
- Afecte negativamente la ejecución de un flujo de negocio del proceso de contratación docente.

Cada defecto registrado debe incluir, como mínimo, los siguientes datos: identificador, fecha de detección, módulo afectado, descripción, pasos para reproducirlo, resultado esperado, resultado obtenido, severidad, prioridad, estado (nuevo, en análisis, corregido, cerrado) y responsable del seguimiento.

3.12.2 Clasificación por Severidad

La severidad describe el impacto técnico y funcional del defecto sobre el sistema y sobre el proceso de contratación docente. Para este proyecto se definieron cuatro niveles:

- **Severidad Crítica**
  Defectos que impiden el funcionamiento del sistema o de un flujo de negocio completo, sin posibilidad de solución temporal. Incluye, por ejemplo, caídas del sistema, imposibilidad total de iniciar sesión o fallos que bloquean la generación de contratos o la validación de documentos.
- **Severidad Alta**
  Defectos que afectan de manera importante módulos o funciones críticas (contratación, carga académica, actividades, bitácora), pero que pueden tener alguna

solución temporal o alternativa manual. Aunque no detienen por completo el proceso, lo vuelven ineficiente o arriesgado.

- **Severidad Media**
  Defectos que afectan funcionalidades secundarias o casos específicos, sin bloquear los flujos principales de contratación. Su impacto es moderado: pueden generar errores de presentación, validaciones incompletas o inconsistencias menores que requieren corrección, pero que permiten seguir utilizando el sistema.
- **Severidad Baja**
  Defectos de tipo estético o de usabilidad menor (por ejemplo, textos, alineaciones, mensajes poco claros, detalles visuales), que no afectan el resultado funcional ni el proceso de contratación, pero cuya corrección mejora la experiencia de uso.

3.12.3 Clasificación por Prioridad

La prioridad indica la urgencia con la que debe atenderse un defecto, considerando el contexto del proyecto, los plazos de la tesina y la importancia del módulo afectado. Se definieron tres niveles:

- **Prioridad Alta**
  Defectos que deben analizarse y, en la medida de lo posible, corregirse de forma inmediata, por su relación con módulos o flujos críticos (login, contratos, validación de documentos, ciclos). Su tratamiento se considera indispensable antes de cerrar la fase de pruebas correspondiente.
- **Prioridad Media**
  Defectos que deben atenderse dentro del ciclo de trabajo actual, pero que no bloquean de forma inmediata la ejecución de pruebas ni el uso del sistema. Se programan para corrección durante la iteración en curso, siempre que los recursos lo permitan.
- **Prioridad Baja**
  Defectos cuya corrección puede posponerse para fases futuras o considerarse como mejora evolutiva. Su atención depende de la disponibilidad de tiempo, dado que no comprometen la operación principal del sistema ni los objetivos de la tesina.

Aunque severidad y prioridad suelen estar relacionadas, no son equivalentes: un defecto de severidad alta puede tener prioridad media si existe un “workaround” temporal, y un defecto de severidad media puede elevarse a prioridad alta si afecta una demostración o entrega clave del proyecto.

3.12.4 Tiempos de Respuesta

Los tiempos en los que deben ser solventadas las brechas identificadas están regidas bajo la siguiente tabla en donde el objetivo planteado está orientado a poder mantener el acuerdo de nivel de servicio, como el sistema tendrá su primer despliegue es necesario que se logren cerrar todas las brechas que imposibilitan su puesta en producción:

Severidad	Descripción	Tiempo de respuesta esperado
Crítica	Tipo de problema que bloquea el uso del sistema o proceso que forma parte clave.	Deben atenderse en menos de 5 horas y corregirse en lapsos no mayores a 24 horas.
Alta	Afecta funciones importantes, hay alternativas para el manejo de estos problemas.	Deben atenderse en menos de 1 día y corregirse en lapsos no mayores a 3 días.
Media	Problema funcional no crítico y no impacta a los flujos importantes.	Deben atenderse en menos de 3 días y corregirse en lapsos no mayores a 6 días.
Baja	Cosas que no impactan (Por ejemplo: cosas cosméticas o mejoras)	Deben atenderse en menos de 6 días y corregirse en siguientes versiones a desplegar.

Tabla 49:Tiempos de Respuesta para Resolución de Bugs

3.13 Análisis de Riesgos de Pruebas

El proceso de pruebas del Sistema de Gestión de Contratos de Personal Docente se desarrolló en un contexto con restricciones de tiempo, recursos y acceso al sistema, por lo que fue necesario identificar los riesgos específicos asociados a las pruebas y valorar su impacto sobre el alcance y la calidad de los resultados obtenidos.

Este análisis complementa el plan de riesgos definido en el capítulo II, pero se centra en cómo dichos riesgos se manifestaron en la práctica durante la ejecución de las pruebas manuales y automatizadas sobre el sistema.

3.13.1 Identificación de Riesgos de Pruebas

Durante el proyecto se identificaron los siguientes riesgos principales vinculados al proceso de pruebas:

- **R1. Cobertura limitada de pruebas por restricciones de tiempo**
  La combinación de carga académica, plazos del curso de especialización y alcance del sistema obligó a priorizar los módulos y casos de prueba.
  - ✓ **Impacto:** la cobertura se concentró en flujos críticos (contratación, contratos, carga académica, actividades de docentes y bitácora), mientras que funcionalidades menos prioritarias recibieron una verificación más superficial.
  - ✓ **Línea de acción:** priorización explícita en la matriz de pruebas (casos obligatorios vs deseables) y enfoque en escenarios end-to-end representativos del proceso de contratación.
- **R2. Alcance acotado en pruebas de rendimiento y seguridad**
  Aunque el plan incluía pruebas con JMeter y casos específicos de seguridad (como los CP_45 y CP_97), estas se aplicaron sobre un conjunto limitado de endpoints y escenarios.
  - ✓ **Impacto:** los resultados permiten una valoración inicial del comportamiento del sistema bajo carga moderada y frente a vulnerabilidades comunes, pero no constituyen un estudio exhaustivo de desempeño ni de seguridad.

- ✓ **Línea de acción:** ejecución focalizada en listados y operaciones críticas, documentación explícita del alcance y recomendación de ampliar estas pruebas en fases posteriores.

- **R3. Dependencia de un entorno de pruebas limitado**
  Las pruebas se realizaron en entornos de desarrollo/pruebas (locales o en contenedores), con volúmenes de datos y recursos de infraestructura inferiores a un entorno institucional de producción.
  - ✓ **Impacto:** ciertos problemas de rendimiento, concurrencia o integración con otros sistemas podrían no haberse manifestado en el entorno de pruebas o hacerlo de forma distinta en producción.
  - ✓ **Línea de acción:** uso de datos de prueba representativos, simulación de escenarios de uso moderado y aclaración en el informe de que las conclusiones se basan en este entorno específico.
- **R4. Cambios en el sistema durante el ciclo de pruebas**
  Cualquier ajuste aplicado al sistema durante la tesina (corrección de errores, mejoras puntuales) puede invalidar evidencias obtenidas previamente o requerir la actualización de casos de prueba y scripts automatizados.
  - ✓ **Impacto:** necesidad de reejecutar algunos flujos, riesgo de inconsistencias entre versiones probadas y resultados documentados.
  - ✓ **Línea de acción:** registro de cambios relevantes, actualización selectiva de scripts y casos cuando fue necesario y reejecución priorizada de pruebas en módulos críticos.
- **R5. Uso limitado de herramientas de análisis estático y seguridad**
  Herramientas como SonarQube y OWASP ZAP se consideraron en el plan, pero su utilización fue acotada y de carácter ilustrativo, no como un barrido exhaustivo del código y de la superficie de ataque.
  - ✓ **Impacto:** posibles issues internos de calidad de código o vulnerabilidades no detectadas en zonas no cubiertas por las pruebas.
  - ✓ **Línea de acción:** uso puntual de estas herramientas para ejemplificar su aporte, y recomendación de integrarlas de forma sistemática en ciclos futuros de mantenimiento del sistema.
- **R6. Grado de subjetividad en la clasificación de defectos**
  La severidad y prioridad de los defectos fueron asignadas por el equipo de tesina, en un contexto académico.
  - ✓ **Impacto:** algunos defectos podrían haber recibido una clasificación distinta en un entorno productivo con políticas formales de gestión de incidentes.
  - ✓ **Línea de acción:** definición clara de criterios de severidad y prioridad (sección 3.12) y aplicación consistente de estos criterios en el registro y análisis de defectos.

3.13.2 Evaluación y Seguimiento de Riesgos de Pruebas

La evaluación de los riesgos se realizó de forma cualitativa, considerando para cada uno su probabilidad de ocurrencia y su impacto potencial sobre el cumplimiento del plan de pruebas. A partir de esta valoración, se priorizaron como riesgos más relevantes aquellos ligados a la

cobertura de pruebas (R1), al entorno de ejecución (R3) y al alcance de rendimiento y seguridad (R2).

El seguimiento de los riesgos se integró a las actividades regulares del proyecto mediante:

- La revisión periódica del avance frente al plan de pruebas (casos ejecutados, módulos cubiertos, defectos detectados),
- La actualización de decisiones de priorización cuando aparecían nuevas restricciones de tiempo o cambios en el sistema,
- La incorporación de los efectos de estos riesgos en el análisis de resultados y en las recomendaciones finales del proyecto.

De esta forma, el análisis de riesgos no se limitó a una lista teórica, sino que sirvió como marco de referencia para interpretar el alcance real de las pruebas realizadas y para justificar las decisiones de priorización que guiaron la ejecución del caso de estudio sobre el Sistema de Gestión de Contratos de Personal Docente.

3.14 Supuestos del proyecto

La planificación, ejecución y análisis de las pruebas del Sistema de Gestión de Contratos de Personal Docente se realizaron bajo una serie de supuestos explícitos, necesarios para delimitar el alcance del proyecto de tesina y para interpretar adecuadamente los resultados obtenidos. Estos supuestos se derivan tanto de las características propias del sistema como de las condiciones académicas, técnicas y organizativas en las que se desarrolló el trabajo.

A continuación, se detallan los principales supuestos considerados:

1- **Estabilidad de la versión del sistema bajo prueba**
Se asumió que la versión del Sistema de Gestión de Contratos utilizada durante la tesina representaba de manera adecuada el comportamiento funcional previsto para su uso institucional y que no se introducirían cambios funcionales profundos en medio de las fases clave de prueba. Cualquier ajuste realizado se consideró puntual y fue gestionado dentro del ciclo de pruebas.

2- **Separación entre entornos de prueba y entornos productivos**
Se asumió que todas las pruebas se ejecutarían en entornos de desarrollo o de prueba (locales o en contenedores), separados de cualquier entorno productivo de la FIA-UES. Bajo este supuesto, las actividades de prueba no afectarían datos reales ni procesos formales de contratación docente.

3- **Uso de datos de prueba representativos, pero no reales**
Se trabajó con datos de prueba diseñados para simular escenarios habituales de contratación (docentes, candidatos, ciclos académicos, contratos, actividades), asumiendo que estos datos eran representativos del uso real del sistema, aunque no correspondieran a información verdadera de la Facultad ni cubrieran todos los casos posibles.

4- **Disponibilidad de credenciales y roles necesarios**

Se asumió que el equipo de tesina dispondría de cuentas de usuario con los roles requeridos (Administrador, Recursos Humanos, Director de Escuela, Asistente Administrativo, etc.) para ejecutar los flujos de negocio definidos en la matriz de pruebas, sin restricciones adicionales de acceso distintas a las propias del sistema.

5- **Configuración adecuada de las herramientas de prueba**
Se asumió que las herramientas seleccionadas Postman para pruebas de API, Selenium WebDriver para automatización de interfaz, JMeter para rendimiento y las herramientas de apoyo para análisis estático y seguridad estaban correctamente instaladas y configuradas, de manera que no introdujeran errores ajenos al sistema bajo prueba. Ante cualquier anomalía, se revisó primero la configuración de la herramienta antes de atribuir el problema al sistema.

6- **Alcance acotado de las pruebas de rendimiento y seguridad**
Se asumió que, para los fines del proyecto de tesina, era suficiente realizar una evaluación básica de rendimiento y seguridad sobre endpoints y flujos críticos, dejando explícito que estudios más profundos de carga, estrés, escalabilidad y vulnerabilidades avanzadas corresponden a fases posteriores de fortalecimiento del sistema.

7- **Evaluación en un contexto académico y no productivo**
Se asumió que la interpretación de los resultados de prueba se realizaría dentro del marco de un proyecto académico, y no como un proceso formal de certificación de software para despliegue productivo. En consecuencia, los hallazgos y recomendaciones se entienden como insumos para la mejora del sistema y de su proceso de pruebas, más que como un informe de auditoría con carácter obligatorio para la institución.

3.15 Tareas del Equipo de Pruebas

El equipo de tesina asumió la responsabilidad de planificar, diseñar, ejecutar y documentar las pruebas del Sistema de Gestión de Contratos de Personal Docente, de acuerdo con el plan de pruebas definido en el capítulo II. En este contexto, las tareas del equipo de pruebas se organizaron de manera que cubrieran todo el ciclo de aseguramiento de calidad, desde la preparación de la matriz de pruebas hasta la consolidación de resultados y la elaboración de conclusiones.

A continuación, se describen las tareas principales realizadas por el equipo de pruebas durante el proyecto:

1- **Análisis del sistema y del modelo de negocio**
- Revisión del diagnóstico, de la descripción del sistema, de los módulos funcionales y del flujo de contratación docente.
- Identificación de los módulos y flujos críticos que debían priorizarse en las actividades de prueba (contratación, contratos, actividades de docentes, bitácora, autenticación).
- **Diseño y mantenimiento de la matriz de pruebas.**

    - Definición de los casos de prueba manuales, incluyendo: identificador, módulo, descripción, precondiciones, datos de entrada, pasos de ejecución y resultados esperados.
- Clasificación de los casos según su tipo (funcionales, no funcionales, seguridad, rendimiento) y su prioridad dentro del proyecto.
- Actualización de la matriz conforme se incorporaban nuevos casos o se refinaban los flujos a partir de la experiencia de prueba.
- **Configuración de entornos y herramientas de prueba**
    - Preparación del entorno de pruebas del sistema (instancias locales y/o contenedores, base de datos de prueba, usuarios y roles).
- Configuración de las herramientas seleccionadas: colecciones y entornos en Postman, proyectos y suites en Selenium, planes de prueba en JMeter y, de forma exploratoria, configuración básica de herramientas de análisis estático y seguridad.

2- Ejecución de pruebas manuales

- Ejecución sistemática de los casos de prueba manuales sobre la interfaz web y la API, siguiendo la matriz de pruebas.
- Registro de resultados (aprobado/no aprobado) y documentación de evidencias visuales (capturas de pantalla, respuestas de API, observaciones relevantes).
- Verificación de flujos end-to-end que recorren varios módulos (por ejemplo, candidato a documentos a solicitud a contrato a bitácora).

3- Diseño, implementación y ejecución de pruebas automatizadas

- Selección de los casos de prueba a automatizar (login, generación de contratos, validación de ciclos, entre otros), de acuerdo con el alcance establecido en el plan.
- Desarrollo de scripts de automatización en Postman (para API), Selenium (para interfaz) y JMeter (para rendimiento).
- Ejecución de los scripts, análisis de resultados y ajuste de estos, cuando se detectaban errores de configuración o de sincronización.

4- Gestión de defectos e incidencias

- Registro de los defectos detectados durante las pruebas manuales y automatizadas, utilizando los criterios de severidad y prioridad definidos en la sección 3.12.
- Documentación de pasos para reproducir, resultados esperados y obtenidos, módulo afectado y evidencias asociadas.
- Seguimiento del estado de los defectos (nuevo, en análisis, corregido, cerrado) y actualización de la matriz y reportes de resultados.
- **Análisis de resultados y elaboración de métricas**
    - Consolidación de la información sobre casos ejecutados, casos aprobados/no aprobados y defectos detectados por módulo y por tipo de prueba.
- Cálculo de métricas básicas de pruebas (por ejemplo, porcentaje de aprobación por módulo, densidad de defectos, distribución por severidad) para apoyar el análisis de la efectividad de las pruebas.

- Interpretación de los resultados en función de los objetivos de calidad definidos para el sistema y de las necesidades del negocio.
- **Documentación y reportes de avance.**
    - Elaboración de reportes parciales de avance de pruebas, integrando observaciones, hallazgos y riesgos detectados.
- Preparación de insumos para las sesiones de retroalimentación con la docente guía, ajustando el plan de trabajo cuando era necesario.
- Integración de todos los artefactos de pruebas (matriz, evidencias, scripts, reportes) en los entregables formales de la tesina.

En conjunto, estas tareas reflejan que el equipo de pruebas no solo se limitó a “ejecutar casos”, sino que asumió un rol activo en la planificación, diseño, ejecución, análisis y documentación del proceso de pruebas del Sistema de Gestión de Contratos de Personal Docente, en coherencia con los lineamientos metodológicos de la Especialización en Ingeniería de la Calidad y con el alcance definido para el proyecto.

3.16 Roles y Responsabilidades en la Ejecución de Pruebas

La ejecución del plan de pruebas del Sistema de Gestión de Contratos de Personal Docente se desarrolló con la participación de distintos roles, tanto académicos como técnicos, cada uno con responsabilidades específicas orientadas a garantizar la calidad del proceso y la correcta documentación de los resultados. A continuación, se describen los principales roles involucrados y sus responsabilidades dentro del proyecto de pruebas.

Equipo de tesina (equipo de pruebas)

El equipo de tesina, conformado por los tres integrantes del proyecto, asumió el rol de equipo de pruebas (QA) responsable de la planificación, diseño y ejecución de las pruebas. Entre sus responsabilidades se encuentran:

- Elaborar la matriz de casos de prueba a partir de los requerimientos funcionales y no funcionales del sistema.
- Seleccionar los casos de prueba a automatizar y definir los criterios de priorización.
- Configurar los entornos de prueba y las herramientas necesarias para la ejecución (Selenium, Postman, JMeter, entre otras).
- Ejecutar las pruebas manuales y automatizadas, registrando resultados y evidencias.
- Identificar, documentar y dar seguimiento a los defectos detectados durante las pruebas.
- Calcular y analizar las métricas de prueba definidas para el proyecto.
- Elaborar los informes de resultados y el acta de cierre de pruebas.

Tutora del proyecto

La tutora del proyecto cumplió un rol de supervisión académica y técnica del proceso de pruebas. Sus responsabilidades incluyeron:

- Revisar y retroalimentar el plan de pruebas, la matriz de casos y los entregables asociados

- Validar la coherencia entre los objetivos del proyecto, el alcance de las pruebas y los resultados obtenidos.

Orientar al equipo de tesina en la aplicación de normas, buenas prácticas y enfoques de aseguramiento de calidad.

Avalar los resultados finales del proceso de pruebas y el acta de cierre correspondiente.

Equipo técnico del sistema

El equipo técnico responsable del desarrollo y mantenimiento del Sistema de Gestión de Contratos de Personal Docente tuvo un rol clave en el soporte a las actividades de prueba. Entre sus responsabilidades se encuentran

- Proveer las versiones del sistema a ser evaluadas en los entornos de prueba.
- Apoyar en la configuración de los ambientes (servidor, base de datos, accesos) necesarios para la ejecución de las pruebas.
- Analizar los defectos reportados por el equipo de pruebas y aplicar las correcciones correspondientes.
- Comunicar cambios relevantes que pudieran afectar el alcance o los resultados de las pruebas.

Usuarios clave y personal administrativo

Los usuarios clave, especialmente personal de Recursos Humanos y autoridades académicas, participaron en las actividades de aceptación de usuario (UAT). Sus responsabilidades incluyeron:

- Ejecutar, con acompañamiento del equipo de tesina, los flujos representativos del proceso de contratación docente.
- Evaluar la usabilidad, claridad de la información y adecuación del sistema a las necesidades del proceso.
- Emitir observaciones y comentarios que alimentaron la mejora del sistema y la interpretación de los resultados de prueba.

Autoridades académicas

Finalmente, las autoridades académicas de la Facultad tuvieron un rol de validación y aprobación institucional del trabajo realizado. Su principal responsabilidad fue:

- Revisar los resultados globales del proceso de pruebas y emitir el aval académico correspondiente sobre el uso del sistema en el contexto del proyecto de tesina.

La definición de estos roles y responsabilidades permitió estructurar el proceso de pruebas de manera clara, facilitando la coordinación entre los distintos actores involucrados y asegurando que las actividades de planificación, ejecución, seguimiento y cierre se desarrollaran con un marco de trabajo definido.

3.17 Principales Hitos del Proyecto de Pruebas

El proyecto de pruebas del Sistema de Gestión de Contratos de Personal Docente se desarrolló siguiendo una secuencia de actividades planificadas que marcaron el avance desde la etapa de análisis inicial hasta el cierre formal del proceso de pruebas. A continuación, se describen los principales hitos que estructuraron el proyecto y que permitieron evidenciar su progreso.

Definición del alcance y objetivos del proyecto de pruebas

Como punto de partida, se definieron el alcance, los objetivos generales y específicos del proyecto de pruebas, alineados con las necesidades de la Facultad de Ingeniería y Arquitectura y con las funcionalidades del sistema. En esta etapa se estableció que el foco principal sería la evaluación funcional y de rendimiento del sistema, priorizando los módulos y procesos críticos del flujo de contratación docente.

Levantamiento de requerimientos de prueba y elaboración de la matriz de casos

Posteriormente, se llevó a cabo el levantamiento de requerimientos de prueba a partir de los requisitos funcionales y no funcionales del sistema. Con base en esta información se elaboró la matriz de casos de prueba, en la que se documentaron los escenarios a evaluar, sus datos de entrada, pasos de ejecución y resultados esperados. Este hito fue fundamental para estructurar de manera sistemática la ejecución de pruebas.

Selección de herramientas y definición de la estrategia de pruebas

Un siguiente hito consistió en la selección y justificación de las herramientas de apoyo a las pruebas (por ejemplo, Selenium, Postman y JMeter), así como en la definición de la estrategia de pruebas manuales y automatizadas. En esta etapa se determinó qué casos se ejecutarían manualmente, cuáles serían candidatos a automatización y de qué forma se incorporarían pruebas de rendimiento sobre los listados y procesos más utilizados.

Ejecución de pruebas funcionales sobre módulos críticos

Con la matriz de casos y los entornos preparados, se procedió a la ejecución de las pruebas funcionales sobre los módulos críticos del sistema. Este hito permitió validar el correcto funcionamiento de los flujos de contratación docente, identificar defectos funcionales y registrar evidencias de los resultados obtenidos.

Desarrollo y ejecución de pruebas automatizadas y de rendimiento

A continuación, se desarrollaron y ejecutaron los casos de prueba seleccionados para automatización, así como los escenarios definidos para pruebas de rendimiento. Este hito consolidó el uso de herramientas especializadas para validar de forma más eficiente la estabilidad del sistema en flujos repetitivos y bajo condiciones de carga moderada.

Registro y análisis de defectos y métricas de prueba

Otro hito relevante fue la consolidación de los defectos detectados y de las métricas de prueba, lo que permitió evaluar cuantitativamente el comportamiento del sistema. El análisis de

defectos por severidad, porcentaje de casos exitosos, cobertura de ejecución y cobertura funcional proporcionó una base objetiva para valorar el grado de calidad alcanzado.

Ejecución de pruebas de aceptación de usuario (UAT)

En una etapa posterior, se llevaron a cabo las pruebas de aceptación de usuario con la participación de personal representativo de los roles funcionales del sistema. Este hito permitió validar el sistema desde la perspectiva del usuario final, recoger observaciones cualitativas y verificar el cumplimiento de los criterios de aceptación establecidos.

Cierre del proceso de pruebas y emisión del acta de cierre

Finalmente, el proyecto de pruebas alcanzó su hito de cierre con la elaboración del informe de resultados y la emisión del acta de cierre de pruebas, en la que se documentaron las conclusiones generales, el estado de los defectos y las recomendaciones de mejora futura. Este hito se complementó con el aval académico de la tutora y de las autoridades correspondientes, formalizando la aceptación del trabajo realizado en el contexto de la tesina.

3.18 Recursos Necesarios para las Pruebas

Para la ejecución del plan de pruebas del Sistema de Gestión de Contratos de Personal Docente fue necesario identificar y planificar los recursos requeridos, tanto a nivel de infraestructura técnica como de personal y herramientas de apoyo. Una adecuada gestión de estos recursos permitió asegurar que las actividades de prueba se desarrollaran en entornos controlados, con las capacidades mínimas necesarias y con el acompañamiento del equipo responsable del proyecto.

3.18.1 Ambientes de Ejecución (Local, Servidor)

Para el proyecto de pruebas se definieron dos tipos principales de ambientes de ejecución, con el objetivo de reproducir condiciones cercanas al uso real del sistema y, al mismo tiempo, disponer de un entorno controlado para la experimentación y el análisis de resultados.

En primer lugar, se contó con un entorno local de pruebas, instalado en los equipos de los integrantes del equipo de tesina. En este ambiente se configuró el sistema con su arquitectura base (backend en Laravel y frontend en React), una base de datos de pruebas y las herramientas necesarias para la ejecución de casos funcionales y automatizados. Este entorno permitió realizar ajustes iniciales, validar la correcta instalación del sistema y depurar posibles incidencias antes de ejecutar pruebas en un ambiente compartido.

En segundo lugar, se utilizó un servidor remoto de pruebas, contratado específicamente para el proyecto por un período limitado, el cual permitió desplegar el sistema en un entorno más cercano a su operación real. En este servidor se configuraron los servicios requeridos (base de datos, servidor de aplicaciones, acceso remoto) y se utilizó como plataforma principal para la ejecución de pruebas de rendimiento, de concurrencia moderada y para las pruebas de aceptación de usuario (UAT). El uso de este ambiente facilitó evaluar el comportamiento del sistema bajo condiciones de acceso más semejantes a las de un entorno institucional.

Ambos ambientes compartían una configuración funcional equivalente del sistema, de manera que los casos de prueba definidos en la matriz pudieran ejecutarse indistintamente en uno u otro, según la naturaleza de la prueba (validación funcional, automatización, rendimiento o aceptación de usuario).

3.18.2 Planeación de Recursos (Humanos, Técnicos, Herramientas)

La planificación de recursos para el proyecto de pruebas abarcó tres dimensiones principales: recursos humanos, recursos técnicos e infraestructura de herramientas de apoyo.

Recursos humanos

El equipo de tesina estuvo conformado por tres integrantes, quienes asumieron de manera conjunta el rol de equipo de pruebas (QA). Sus actividades incluyeron el diseño de la matriz de casos de prueba, la preparación de datos, la ejecución de pruebas manuales y automatizadas, el registro de defectos, el análisis de métricas y la elaboración de informes.

Adicionalmente, se contó con la participación de la tutora del proyecto, quien brindó acompañamiento académico y técnico, revisó los artefactos generados y avaló los resultados del proceso de pruebas. En las etapas de aceptación de usuario intervinieron también usuarios clave del proceso de contratación (personal administrativo y académico), quienes aportaron su visión sobre la adecuación del sistema a las necesidades reales de la Facultad.

Recursos técnicos

En cuanto a infraestructura, se utilizaron equipos de cómputo personales de los integrantes del proyecto para el trabajo local, con las capacidades mínimas necesarias para ejecutar el entorno de desarrollo y las herramientas de pruebas. Además, se dispuso del servidor remoto de pruebas, cuya contratación implicó un costo mensual y permitió desplegar el sistema en un ambiente accesible vía internet para la ejecución de pruebas de rendimiento y UAT.

Se consideraron también como recursos técnicos la conectividad a internet, el almacenamiento requerido para evidencias de prueba (capturas de pantalla, reportes, registros) y el espacio en repositorios para respaldar la configuración del sistema y de las herramientas.

Herramientas de apoyo

Finalmente, se planificó el uso de un conjunto de herramientas de apoyo al proceso de pruebas, seleccionadas con base en criterios de facilidad de uso, compatibilidad tecnológica, costo y soporte comunitario. Entre ellas destacan:

- Herramientas para pruebas funcionales y automatizadas de interfaz, como Selenium WebDriver.
- Herramientas para pruebas de API, como Postman, utilizadas para validar los endpoints del backend.
- Herramientas para pruebas de rendimiento y carga, como JMeter, empleadas para evaluar tiempos de respuesta y comportamiento bajo carga moderada.
- Herramientas de gestión y registro de defectos, y de organización del trabajo del equipo.

La integración de estos recursos humanos, técnicos y de software, planificados desde el inicio del proyecto, permitió llevar a cabo el proceso de pruebas de forma estructurada, documentada y alineada con los objetivos de calidad establecidos para el sistema.

3.19 Estrategia de Datos de Prueba y Herramientas

La ejecución de pruebas dentro del proyecto Sistema de Gestión de Contratos del Personal Docente FIA requirió el diseño de una estrategia integral de datos que asegurara la cobertura de escenarios reales, negativos y extremos. Para ello se aplicaron tres enfoques principales: uso de datos reales anonimizados, generación de datos sintéticos y construcción de datos basados en escenarios de prueba. Esta estrategia permitió validar la funcionalidad del sistema en condiciones controladas, reproducibles y representativas del entorno operativo.

3.19.1 Estrategia de Datos de Prueba (Anonimización, Datos Ficticios, Restauración)

Datos Reales Anonimizados

Se utilizaron datos reales del sistema, pero sometidos a un proceso de anonimización para proteger información sensible. Las acciones principales fueron:

- Sustitución de nombres por equivalentes ficticios.
- Reemplazo de DUI, NIT, teléfonos y correos por valores aleatorios conservando el formato original.
- Eliminación de direcciones reales y datos de contacto privados.
- Enmascaramiento parcial de documentos PDF y fotografías.

Este tipo de datos permitió ejecutar pruebas funcionales con estructuras altamente coherentes y representativas del sistema real.

Datos Sintéticos

Se generaron datos artificiales para validar casos límite, pruebas negativas y escenarios no presentes en la base real. Entre los datos sintéticos generados se incluyen:

- Candidatos con información incompleta o con validaciones incorrectas.
- Contratos duplicados para verificar restricciones de unicidad.
- Registros masivos para pruebas de paginación y cargas de trabajo.
- Semestres fuera del rango permitido para evaluar respuestas 422.
- Usuarios con roles incorrectos o sin permisos para validar errores 403.

El uso de datos sintéticos facilitó explorar comportamientos del sistema ante situaciones poco comunes o extremas.

Datos Basados en Escenarios de Prueba

Los datos se construyeron directamente desde los casos de prueba (CP_XX), permitiendo reproducir condiciones específicas requeridas por pruebas funcionales y automatizadas.

Ejemplos:

- Candidato sin documentos (CP_16).

- Carga académica incompleta para validación de contratos (CP_68).
- Pruebas de paginación y volumen con más de 10 registros (CP_79).
- Tokens inválidos o expirados para validación de autenticación (CP_92 y Postman).

Este enfoque permitió comprobar reglas de negocio críticas en rutas específicas.

3.19.2 Herramientas de Apoyo a las Pruebas (Selenium, Postman, JMeter, SonarQube, etc.)

Postman

Se empleó para pruebas de API, validación de respuestas, autenticación (token), errores controlados (422, 401, 403) y ejecución masiva mediante Collection Runner. Permite crear entornos con variables para automatizar flujos de:

- Semestres
- Facultades
- Escuelas
- Candidatos
- Contratos

Postman fue fundamental para validar endpoints y verificar la integridad de los datos usados en las pruebas.

Selenium con Python

Utilizado para la automatización de flujos repetitivos del sistema, especialmente:

- Login
- Registro y carga de candidatos
- Flujo de contratos y carga académica
- Validación de formularios y restricciones
- Flujo completo para pruebas de regresión

Los scripts automatizados permitieron reducir significativamente el tiempo de ejecución respecto a pruebas manuales.

Generadores de Datos

Se usaron herramientas para producir datos de prueba de manera controlada:

a) **Python + Faker**
   Para generar nombres, correos, DUI ficticios, teléfonos y fechas.
b) **Laravel Seeders y Factories**
   Para poblar entornos de desarrollo y QA con datos estructurados de:
   - Facultades
   - Escuelas
   - Usuarios
   - Candidatos sintéticos
   - Grupos académicos

Herramientas de Soporte

Se utilizaron herramientas específicas para gestionar y verificar la base de datos en PostgreSQL:

- **PgAdmin:** consulta y verificación de integridad de los datos (FK, PK, restricciones).
- **JMeter**: para pruebas de rendimiento en endpoints críticos.

3.20 Entregables del Proyecto de Pruebas

En el marco de este proyecto, el proceso de pruebas generará una serie de entregables documentales y técnicos que permitirán evidenciar, rastrear y auditar cada una de las actividades ejecutadas durante la validación del sistema de gestión de contratos docentes. Estos entregables son fundamentales para asegurar la transparencia del proceso, facilitar futuras mejoras y garantizar la trazabilidad entre los requerimientos definidos y los resultados obtenidos.

Todos los entregables estarán alineados con lo estipulado en la norma ISO/IEC 29119-3: Documentación de Pruebas, la cual define los tipos de documentos mínimos requeridos para garantizar la correcta implementación de un proceso de calidad en entornos de software.

A continuación, se detallan los principales entregables del proceso de pruebas:

- **Plan de Pruebas Funcionales**: Documento base que describe el objetivo, alcance, criterios de entrada y salida, roles involucrados, riesgos, tipo de pruebas a realizar y recursos necesarios. Este plan orienta todo el proceso de validación y su estructura sigue las recomendaciones de la ISO/IEC 29119.
- **Casos de Prueba (Test Cases)**: Documento que detalla paso a paso las acciones a realizar, los datos de entrada, las condiciones iniciales, los resultados esperados y los criterios de aceptación. Cada caso de prueba está vinculado a uno o varios requerimientos funcionales del sistema.
- **Matriz de Trazabilidad de Requisitos y Pruebas**: Documento que permite verificar que cada requerimiento funcional ha sido cubierto por uno o más casos de prueba, garantizando la integridad del proceso de validación.
- **Evidencias de Ejecución de Pruebas**: Incluye capturas de pantalla, respuestas del sistema, registros de validación (logs) y resultados de herramientas como PHPUnit. Estas evidencias respaldan la correcta ejecución de las pruebas y facilitan auditorías posteriores.
- **Informe de Resultados de Prueba**: Documento resumen que presenta los resultados obtenidos durante la ejecución de los casos de prueba. Se clasifican como casos exitosos, fallidos o pendientes, e incluyen observaciones y recomendaciones.
- **Reporte de Defectos**: Registro que detalla cada defecto identificado, incluyendo su descripción, prioridad, estado actual, módulo afectado y responsable de resolución. Este documento se actualiza conforme los errores son corregidos y validados nuevamente.
- **Plan de Pruebas Automatizadas (cuando aplique)**: Documento complementario que describe los scripts implementados, herramientas utilizadas, lógica de automatización y criterios de ejecución. Esto aplica especialmente en módulos como login, gestión de usuarios y generación de contratos.

Todos estos entregables serán almacenados de forma organizada y respaldados en una carpeta del proyecto, permitiendo su consulta posterior por parte de responsables institucionales o auditores internos. Asimismo, se incluirán como anexos seleccionados en la versión final de la tesina, a fin de evidenciar el cumplimiento del proceso de pruebas según los estándares internacionales adoptados.

3.20.1 Matriz de Pruebas Manuales

La matriz de pruebas manuales constituye uno de los entregables centrales del proyecto de pruebas, ya que en ella se documentan de manera sistemática los casos diseñados para evaluar el comportamiento funcional del sistema. Cada caso de prueba incluye su identificador (CP_XX), el módulo al que pertenece, una descripción breve, las precondiciones necesarias, los pasos de ejecución, los datos de entrada, el resultado esperado, el resultado obtenido y las evidencias asociadas.

Esta matriz permitió organizar y controlar la ejecución de las pruebas manuales sobre los módulos críticos del Sistema de Gestión de Contratos de Personal Docente, y facilitó el registro de incidencias y el cálculo de métricas relacionadas con la cobertura y el porcentaje de casos exitosos.

3.20.2 Matriz de Pruebas Automatizadas

Además de la matriz de pruebas manuales, el proyecto contempló la definición y construcción de un conjunto de pruebas automatizadas agrupadas en suites, orientadas a la validación recurrente de flujos críticos del sistema y a la ejecución de escenarios de rendimiento. Estas suites se diseñaron a partir de la selección de casos de prueba.

Las pruebas automatizadas se implementaron utilizando herramientas como Selenium WebDriver para la automatización de interfaz gráfica, Postman para la validación de servicios API y JMeter para la simulación de carga y la medición de tiempos de respuesta. Cada suite agrupa un conjunto de casos de prueba relacionados, permitiendo su ejecución de manera repetible y eficiente cuando se requiera validar una nueva versión del sistema o realizar pruebas de regresión.

Si bien la estructura de estas suites no se presenta en forma de tabla tan detallada como la matriz de pruebas manuales, su organización se documenta en los manuales técnicos de automatización y en los proyectos de pruebas correspondientes, donde se especifican los casos cubiertos, los pasos automatizados, los datos de entrada gestionados y la forma de interpretar los resultados generados por las herramientas.

En conjunto, la matriz de pruebas manuales y las suites automatizadas constituyen la base operativa del proceso de pruebas del sistema, proporcionando tanto una cobertura funcional amplia como mecanismos eficientes para la reevaluación de los escenarios más relevantes del proceso de contratación docente.

3.20.3 Test Readiness Review (TRR)

El Test Readiness Review (TRR) tiene como propósito verificar que todos los elementos necesarios para iniciar la fase de pruebas del sistema se encuentran completos, operativos y en

condiciones adecuadas. Este proceso asegura que las pruebas se ejecuten de manera estable, controlada y sin riesgos críticos que afecten la calidad del software.

1. **Preparación del Entorno de Pruebas**
   El sistema se encuentra desplegado y en funcionamiento en un servidor en la nube mediante Railway.app, accesible desde la URL: https://helpful-youth-production-1d3f.up.railway.app/
   El entorno es estable, accesible y reúne todas las condiciones necesarias para realizar pruebas funcionales y automatizadas. El backend (Laravel), el frontend (React) y la base de datos PostgreSQL están integrados correctamente, con migraciones, servicios y rutas API operativas.
   Las variables de entorno, autenticación, permisos, carga de archivos y generación de documentos funcionan sin errores. Se verificó la disponibilidad del entorno, los tiempos de respuesta y la estabilidad general del servidor, confirmando que se encuentra apto para iniciar la fase de QA.
2. **Disponibilidad de Datos de Prueba**
   El proyecto cuenta con datos de prueba completos y listos para su uso:
   - **Datos reales anonimizados** de candidatos, contratos y cargas académicas para pruebas funcionales.
   - **Datos sintéticos** para validar escenarios negativos, extremos y casos poco frecuentes (DUI duplicados, contratos inválidos, cargas excedidas, semestres fuera de rango, tokens expirados).
   - **Datos organizados por módulo** para semestres, facultades, escuelas, candidatos, grupos, carga académica y contratos.

   Estos datos garantizan la ejecución controlada y reproducible de todos los flujos del sistema.

3. **Matriz de Casos de Prueba**
   Se dispone de una matriz completa de casos de prueba manuales que cubre:
   - Validaciones (422), permisos (403, 401) y flujos CRUD.
   - Procesos de autenticación, registro de candidatos, carga académica y gestión de contratos.
   - Pruebas de regresión, carga de documentos y estados del contrato.
   - La cobertura funcional supera el 95% del sistema, abarcando escenarios reales y negativos necesarios para asegurar la calidad del software.
4. **Preparación de Pruebas Automatizadas**
   Existen herramientas y scripts listos para su ejecución:
   Selenium + Python:
   - Automatización de login, registro de candidatos, flujo de contratos, validaciones y pruebas repetitivas de regresión.
   - Scripts configurados con waits, selectores limpios, logs y capturas.

   Postman:

- Colección completa para semestres, facultades, escuelas, candidatos, carga académica, contratos y autenticación.
- Variables de entorno, manejo de tokens y ejecución masiva con Collection Runner.

Estas herramientas reducen tiempos, aumentan la precisión y permiten repetir pruebas sin intervención manual.

5. **Revisión de Requerimientos y Criterios de Aceptación**
   Todos los requerimientos funcionales del sistema se encuentran documentados, validados y alineados con las necesidades del sistema. Cada requerimiento posee criterios de aceptación claros y casos de prueba asociados, lo que asegura trazabilidad directa entre desarrollo, pruebas y resultados esperados.
6. **Control de Defectos y Versionamiento**
   Los defectos detectados en las fases de pruebas se dan posibles soluciones o recomendaciones al equipo de desarrolladores para solventar fallas en sistema. Los repositorios de frontend y backend se encuentran en GitHub, y la versión desplegada en Railwaypara hacer las pruebas.

3.21 Estimaciones de Esfuerzo de Pruebas

3.21.1 Estimación de Esfuerzo en Tiempo

La estimación se realizó con base en:

- **106 casos de prueba documentados** en la matriz de pruebas y el informe de evidencias entregado.
- **Tiempos reales de ejecución** registrados por cada Tester en el documento de evidencias de las pruebas del sistema por modulo.

Resumen general de pruebas manuales

Indicador	Valor
Total, de casos de prueba	106
Casos exitosos	83
Casos fallidos	18
Casos con bloqueo	4
Tiempo total invertido	1,744.2 minutos
Horas totales	29.07 horas
Iteración de pruebas cubierta	100 % de cobertura funcional

Tabla 50 Resumen general de pruebas manuales

3.22 Defectos

3.22.1 Administración de Defectos

Durante la fase de pruebas, todos los defectos detectados serán documentados en la plataforma Azure DevOps, con el fin de garantizar un control organizado, y transparente de cada hallazgo.

Además, la administración de defectos se alineará con el formato TCS (Test Case Specification) de la norma IEEE 829, asegurando que la información registrada cumpla con los estándares internacionales para documentación de pruebas de software.

Para cada defecto se registrará la siguiente información:

- ID del incidente: Código único generado por la herramienta para identificar y dar seguimiento al defecto durante todo el ciclo de pruebas del proyecto.
- Título del incidente: Breve descripción clara del problema encontrado.
- Reportado Por: Miembro del equipo de QA encargado de la ejecución de pruebas.
- Versión del software: Identificación de la versión donde se detectó el defecto.
- Referencia: Relación directa con la historia de usuario y el caso de prueba donde se presentó la incidencia.
- Severidad y Prioridad: Grado de impacto y urgencia de resolución.
- URL Azure: Enlace directo al registro en la plataforma.
- Tipo de error: Clasificación y frecuencia de aparición.
- Resumen del incidente: Descripción del comportamiento incorrecto observado.
- Impacto: Consecuencias que el defecto provoca en el sistema o en el usuario.
- Precondiciones: Estado previo necesario para reproducir el defecto.
- Datos utilizados: Información de entrada empleada en la prueba.
- Pasos de reproducción: Secuencia detallada para replicar el problema.
- Resultado esperado y Resultado obtenido: Comparación entre el comportamiento previsto y el real.
- Evidencia: Archivos de apoyo como capturas, registros o videos.

Reporte de Defectos – (Relacionado con IEEE829)

ID del incidente		**Fecha del reporte**	
Título del incidente		**Reportado por**	
Versión del software		**Caso de prueba asociado**	
Severidad		**Prioridad**	
URL Azure			
Tipo de Error			
Resumen del Incidente			
Impacto			
Entorno de pruebas			
Precondiciones			
Datos utilizados			
Resultado esperado y Resultado Obtenido			
Evidencia			

Ilustración 83 Formulario de Reporte de Defectos

3.22.2 Seguimiento de Defectos

Una vez registrado un defecto en el formulario de reporte de incidentes, se estableció un proceso de seguimiento para garantizar que cada incidencia fuera atendida de manera oportuna y transparente. Este seguimiento permitió conocer en todo momento el estado de cada defecto, su responsable y las acciones realizadas para su resolución.

El flujo de vida de los defectos contempló, de forma general, las siguientes etapas:

- **Registro inicial**: el integrante del equipo de pruebas que detectaba el problema completaba el formulario de reporte de defectos, asignando un identificador único al incidente e indicando la severidad, el caso de prueba asociado, el entorno donde se había presentado y la evidencia correspondiente.
- **Revisión y clasificación**: los defectos reportados eran revisados para confirmar su validez, completar información en caso necesario y ajustar la severidad o prioridad de acuerdo con su impacto en el proceso de contratación docente.
- **Asignación para corrección:** una vez confirmados, los defectos se ponían a disposición del equipo técnico responsable del sistema, quien asumía la tarea de análisis de causa y corrección en la versión correspondiente.
- **Verificación de corrección:** tras aplicar los cambios, el equipo de pruebas volvía a ejecutar el caso de prueba asociado (y, cuando correspondía, pruebas de regresión)

para confirmar que el defecto había sido resuelto y que no se habían introducido nuevos problemas.

- **Cierre o re-apertura:** si el comportamiento del sistema coincidía con el resultado esperado, el defecto se marcaba como cerrado. En caso contrario, el incidente se reabría, actualizando la descripción y la evidencia para una nueva iteración de corrección.

Este proceso de seguimiento permitió mantener la trazabilidad entre los casos de prueba ejecutados, los defectos detectados y las decisiones tomadas sobre cada incidencia. Además, facilitó la elaboración de métricas relacionadas con la cantidad de defectos abiertos y cerrados, su distribución por severidad y la eficacia del proceso de pruebas, aportando información relevante para el análisis de resultados y para la elaboración del acta de cierre de pruebas.

3.22.3 Revisión de Defectos

Además del registro y seguimiento individual de cada incidencia, se realizaron revisiones periódicas de los defectos detectados con el fin de analizar su comportamiento de forma global y tomar decisiones informadas sobre las acciones de mejora necesarias. Estas revisiones permitieron identificar patrones, evaluar el impacto de los defectos en el proceso de contratación docente y verificar el avance del equipo técnico en la atención de los problemas reportados.

Durante estas sesiones de revisión se ponía especial atención a los siguientes aspectos:

- **Defectos críticos y altos:** se verificaba que todos los defectos clasificados con severidad crítica o alta estuvieran siendo atendidos con prioridad, dado su potencial impacto en la operación del sistema y en la continuidad del proceso de contratación.
- **Tendencias y recurrencia:** se analizaban los defectos agrupados por módulo, tipo de error o causa probable, con el propósito de identificar áreas del sistema particularmente problemáticas o patrones de fallos que requirieran acciones correctivas más amplias.
- **Estado general del ciclo de pruebas:** se revisaba el número de defectos abiertos, en corrección y cerrados, relacionándolo con el avance en la ejecución de casos de prueba y con las métricas de calidad definidas para el proyecto.
- **Impacto en el cronograma:** se evaluaba si la cantidad y severidad de los defectos detectados podía afectar las fechas previstas para las fases de pruebas, la UAT o el cierre del proyecto, y se discutían ajustes necesarios en la planificación.

Los resultados de estas revisiones se documentaban en notas de seguimiento y se incorporaban tanto al análisis de métricas como a las conclusiones del informe de resultados de pruebas. De esta manera, la revisión periódica de defectos no solo sirvió para controlar el estado de las incidencias reportadas, sino también como una herramienta de gestión para orientar las decisiones del proyecto y apoyar la mejora continua del sistema.

3.23 Proceso para el Reporte de Avance de Pruebas

El proceso para elaborar el Reporte de Avance de Pruebas tiene como propósito documentar el estado actual de las actividades de QA, el porcentaje ejecutado, los resultados obtenidos, los defectos identificados y los riesgos que pueden afectar la calidad del sistema. Este

proceso asegura trazabilidad, control, seguimiento y comunicación clara con el equipo de desarrollo y autoridades. Como equipo QA se llevó el seguimiento de cada proceso en los siguientes enlaces: Calendario de Actividades.xlsx

1. Planificación del Avance del Ciclo de Pruebas

Antes de iniciar el seguimiento, se definen:

- El alcance del ciclo de pruebas (módulos incluidos).
- Los casos de prueba priorizados (CP alta, media y baja).
- El tipo de pruebas ejecutadas: funcionales, API, automatizadas.
- Las herramientas por utilizar (Excel, Selenium, Postman).
- El entorno de ejecución:

  Railway.app https://helpful-youth-production-1d3f.up.railway.app/

Esta planificación permite establecer una línea base para medir el avance real.

2. Ejecución y Registro del Progreso

Cada caso de prueba se ejecuta y se registra su estado.

Para ello se documenta:

- Caso ejecutado (ID CP_XX)
- Resultado: Aprobado / Fallado / Bloqueado / No Ejecutado
- Tiempo empleado
- Datos utilizados (reales anonimizados o sintéticos)
- Evidencias: capturas, videos, logs o resultados Postman
- Observaciones relevantes

Los resultados se consolidan en la matriz de pruebas para generar métricas precisas. En el siguiente enlace: MATRIZ DE PRUEBAS.xlsx

3. Cálculo de Métricas de Avance

El avance del ciclo se calcula a partir del estado de todos los casos de prueba:

3.1. Métricas principales

- Casos ejecutados vs. totales
- Porcentaje de aprobación
- Cantidad de fallas por módulo
- Defectos críticos detectados
- Pruebas automatizadas ejecutadas
- Tiempo real vs. tiempo estimado

Estas métricas permiten conocer el estado de calidad del sistema en tiempo real.

4. Consolidación de Defectos Detectados

Los defectos detectados se registran con la siguiente información:

- ID del defecto
- Módulo afectado
- Severidad (Crítico, Alto, Medio, Bajo)
- Descripción detallada
- Pasos para reproducir
- Evidencia adjunta (captura/log/Postman)
- Estado: Abierto / En corrección / Solucionado / Validado

Esto permite medir la estabilidad y madurez del sistema.

5. Integración de Resultados de Pruebas Automatizadas

Las ejecuciones automatizadas con Selenium y Postman se integran al reporte describiendo:

Selenium

- Scripts ejecutados (login, candidatos, contratos, formularios).
- Tiempo total de ejecución.
- Incidencias detectadas.

Postman

- Colecciones ejecutadas (API semestres, escuelas, candidatos, contratos).

- Validaciones automáticas 200/422/401/403.
- Tiempo de respuesta promedio.

La automatización reduce el tiempo de regresión y mejora la confiabilidad del reporte.

6. Evaluación del Progreso por Módulo

El reporte detalla el avance por cada módulo del sistema:

- Semestres
- Facultades
- Escuelas
- Candidatos
- Grupos
- Carga Académica
- Contratos
- Autenticación

Para cada módulo se indica:

- Casos ejecutados
- Casos aprobados
- Casos fallados
- Riesgos o bloqueos
- Recomendaciones de seguimiento

Esto permite enfocar la atención en áreas críticas.

3.24 Documentación Técnica de Pruebas Automatizadas

Como parte del proyecto de pruebas del Sistema de Gestión de Contratos de Personal Docente, se desarrolló un Manual Técnico de Pruebas Automatizadas, el cual documenta de manera detallada la implementación, configuración y ejecución de las suites desarrolladas con herramientas como Selenium WebDriver, Postman y JMeter.

Dado que este manual constituye un artefacto extenso y altamente técnico, y con el fin de optimizar el espacio dentro de este documento de tesina, su contenido completo se encuentra disponible en un archivo independiente alojado en un repositorio digital.

El manual incluye, entre otros, los siguientes elementos:

- Estructura de los proyectos de automatización (carpetas, archivos y convenciones utilizadas).
- Configuración del entorno de pruebas automatizadas, incluyendo dependencias, instalación de paquetes, entornos virtuales y configuración de navegadores.
- Descripción técnica de los casos de prueba automatizados, indicando el CP de referencia, el flujo representado y los criterios de validación implementados.
- Procedimiento para la ejecución de las suites, tanto en modo individual como en ejecución completa.
- Evidencias generadas automáticamente, tales como reportes HTML, capturas de pantalla, logs, resultados de ejecución y análisis de fallas.
- Pruebas de rendimiento, con la configuración de JMeter, los elementos del test plan y la forma de interpretar los reportes (Aggregate Report, Summary Report, Response Time Graph).
- Buenas prácticas aplicadas, convenciones de codificación, manejo de excepciones, organización de bibliotecas y recomendaciones para su mantenimiento.

El documento completo puede consultarse en el siguiente enlace: Manual Tecnico de Pruebas de Calidad al Sistema de Contratos de Personal Docente FIA UES.pdf

3.25 Carta de Salida

Como cierre formal del proceso de pruebas del Sistema de Gestión de Contratos de Personal Docente, se elaboró la Carta de Salida o Acta de Cierre de Pruebas, documento que consolida los resultados obtenidos durante la ejecución de las pruebas funcionales, automatizadas y de rendimiento, así como el estado de los defectos identificados y las métricas finales del proyecto.

La Carta de Salida constituye el aval académico que certifica que las actividades de pruebas fueron ejecutadas conforme al plan establecido, que los módulos críticos del sistema fueron validados y que no permanecen defectos de severidad crítica o alta que comprometan la operación del proceso de contratación docente. Asimismo, documenta las observaciones generales, las conclusiones obtenidas y las recomendaciones para futuras iteraciones del sistema.

Debido a su extensión y a la naturaleza formal del documento, el acta completa se incluye como un archivo independiente y puede consultarse en el siguiente enlace: Acta de Cierre.pdf

4. Análisis de Datos y Métricas Obtenidos

4.1 Resultados Obtenidos.

4.1.1 Pruebas Manuales.

Las pruebas de calidad realizadas en el proyecto fueron un total de 106 pruebas de las cuales se tiene la siguiente clasificación por módulo:

Módulos	Estado de Prueba				Total, General
	Bloqueo	Ejecución	Exitosa	Fallida	
ACTIVIDADES DE DOCENTES	0	0	3	0	3
BANCOS	0	0	5	0	5
BITÁCORA	0	0	5	0	5
CANDIDATO	0	0	7	0	7
CARGA ACADEMICA	0	0	2	1	3
CARGOS	0	0	1	1	2
CICLO	0	0	5	1	6
CONTRATACIÓN	0	0	14	1	15
EMPLEADOS	0	0	2	0	2
ESCALAFÓN	0	0	5	1	6
ESCUELAS	0	0	4	1	5
FACULTADES	0	0	5	0	5
FORMATOS	0	0	3	0	3
GRUPOS	0	0	2	0	2
RENDIMIENTO	0	0	1	3	4
REVISION DE DOCUMENTOS	0	0	7	0	7
SEGURIDAD	3	0	4	4	11
TIPOS DE GRUPO	0	0	2	0	2
USABILIDAD	0	0	0	2	2
USUARIOS	1	0	7	3	11
Total, general	**4**	**0**	**83**	**18**	**106**

Tabla 51:Resumen de Estado de Pruebas Manuales.

Según los hallazgos durante la ejecución de las pruebas manuales se encontraron un total de 22 pruebas que presentaron errores o bloqueos para lograr evaluarlas (18 pruebas de calidad fallidas y 4 pruebas presentaron bloqueos para realizarlas con éxito).

4.1.2 Reporte de Pruebas Fallidas o con Bloqueo.

ID_CASO	MÓDULO	NOMBRE DE CASO DE PRUEBA	TITULO INCIDENTE	ESTADO	SEVERIDAD
CP_03	USUARIOS	Creación de usuario	Error en Creación de Usuario con Rol Asistente de Recursos Humanos	Fallida	Critica
CP_04	USUARIOS	Actualizar Usuario	Nombre del usuario permite ingreso de datos numéricos.	Fallida	Critica
CP_04	USUARIOS	Actualizar Usuario	La operación de actualización responde 201 Created en lugar de 200 OK	Fallida	Baja
CP_07	USUARIOS	Validación de roles y permisos	Usuarios con rol restringido pueden visualizar rutas y datos no autorizados.	Fallida	Alta
CP_29	ESCALAFÓN	Rutas protegidas de módulo escalafón	Acceso no autorizado de candidato a rutas protegidas	Fallida	Alta
CP_43	CONTRATACIÓN	Permisos por rol en solicitudes de contratación	Permisos de rol en solicitudes de contratación	Fallida	Alta
CP_45	SEGURIDAD	Seguridad: inyección SQL / XSS	Formularios aceptan payloads maliciosos (SQLi /XSS)	Fallida	Alta
CP_47	RENDIMIENTO	Tiempo de respuesta en listados principales	Tiempo de respuesta en listados principales	Fallida	Alta

Tabla 52: Reporte de Pruebas Fallidas o Con Bloqueo

ID_CASO	MÓDULO	NOMBRE DE CASO DE PRUEBA	TITULO INCIDENTE	ESTADO	SEVERIDAD
CP_49	RENDIMIENTO	Inicio de sesión con muchos usuarios simultáneos	Inicio de sesión con muchos usuarios simultáneos	Fallida	Alta
CP_50	RENDIMIENTO	Tiempo de respuesta del Dashboard (Carga de indicadores)	Tiempo de respuesta del Dashboard (Carga de indicadores)	Fallida	Alta
CP_55	CICLO	Permisos de lectura en ciclos	Permisos de lectura en ciclos	Fallida	Alta
CP_58	ESCUELAS	Creación de escuela	Creación de escuela bajo una facultad existente	Fallida	Alta
CP_63	CARGOS	Acceso restringido a Cargos para rol no Administrador	Acceso restringido a Cargos para rol no Administrador	Fallida	Alta
CP_68	CARGA ACADEMICA	Consultar cargas por escuela	El sistema no permite validar filtrado de cargas por escuela	Fallida	Media
CP_97	SEGURIDAD	Acceso no autorizado a datos de otro usuario	Acceso no autorizado a los datos de otro usuario.	Fallida	Alta
CP_101	SEGURIDAD	CORS restrictivo	CORS no bloquea correctamente orígenes no permitidos	Fallida	Alta
CP_102	SEGURIDAD	Cabeceras de seguridad	Faltan cabeceras de seguridad y hay riesgo de contenido mixto.	Fallida	Media
CP_107	USABILIDAD	Compatibilidad: de navegadores	Problemas en identificadores en la usabilidad del sistema en navegadores.	Fallida	Media

Tabla 53: Reporte de Pruebas Fallidas o Con Bloqueo

ID_CASO	MÓDULO	NOMBRE DE CASO DE PRUEBA	TITULO INCIDENTE	ESTADO	SEVERIDAD
CP_108	USABILIDAD	Accesibilidad básica	Problemas de visualización al aplicar Zoom 200%	Fallida	Baja

Tabla 54: Reporte de Pruebas Fallidas o Con Bloqueo

A continuación, detallamos las pruebas a las cuales no se le logro realizar proceso de revisión de calidad debido a impedimentos.

ID_CASO	MODULO	NOMBRE DEL CASO DE PRUEBA	ESTADO	MOTIVO DE BLOQUEO
CP_08	USUARIOS	Generación y validación de JWT/refresh token	Bloqueo	El proyecto no cuenta con proceso de expiración, rotación y renovación de tokens, por lo cual este es un riesgo potencial en el proyecto
CP_96	SEGURIDAD	Contador de intentos fallidos y lockout (intentos incorrectos de inicio de sesión)	Bloqueo	El proyecto no cuenta con una política de lockout por intentos fallido. No hay forma de verificar número máximo de intentos ni tiempos de bloqueo.
CP_105	SEGURIDAD	Detección de enumeración de cuentas	Bloqueo	El ambiente de pruebas no cuenta con la capacidad del volumen de pruebas que puede disparar.
CP_106	SEGURIDAD	Validar seguridad de “Recordarme” y persistencia de sesión	Bloqueo	No se dispone de especificación técnica para validar vencimiento, cierre remoto o reutilización de cookies.

Tabla 55: Reporte de Pruebas Fallidas o Con Bloqueo

Para mayor detalle de las pruebas que se encontraron fallidas y de las propuestas para los escenarios de las pruebas con bloqueo pueden referenciarse del documento Evidencias Pruebas Manuales en el apartado de Reporte de Defectos.

1.4.3 Pruebas de Seguridad.

Alcance.

ZAP se ejecutó con una versión 2.16.1 y analizo los siguientes dominios para nuestra aplicación:

- Frontend: https://helpful-youth-production-1d3f.up.railway.app
- Backend: https://gestioncontratos-production.up.railway.app
- Servicios de terceros: cdnjs.cloudflare.com y servicios de Google.

En el archivo resultado del informe únicamente tomaremos los hallazgos relacionados a nuestros dominios no a los de terceros.

Resumen de Riesgos.

Según la ejecución de realizada se encontraron según clasificación de riesgo un total de alertas:

Riesgo	Frontend	Backend	Total
Alta	1	0	1
Media	2	0	2
Baja	3	3	6
Informativas	3	3	6

Tabla 56: Resumen de Riesgo de Seguridad OWASP ZAP

La superficie de ataque principal como podemos revisar en el cuadro resumen se concentra en el frontend ya que para el backend solo se presentaron hallazgos de bajo e informativo impacto.

- La mayoría de los hallazgos se relacionan con:
- Uso de librería JavaScript con vulnerabilidades conocidas en el frontend.
- Cabeceras de seguridad ausentes o mal configuradas, como Anti-Clicjacking, HSTSS y X-Content-Type-Options.
- Configuración de caché para contenido potencialmente sensible.

A continuación, detallaremos los principales tipos de hallazgos:

Nivel de Riesgo	Tipo	Apunta al archivo	Descripción	Riesgo	Solución
Alto	Librería JS Vulnerable	/static/js/ main.180add7d.js	Se detecta una librería JavaScript incluida en el bundle del frontend con vulnerabilidades conocidas (por ejemplo, versión vulnerable de una dependencia como axios).	Un atacante podría explotar vulnerabilidades públicas de esa librería para ejecutar código malicioso, robar datos o comprometer la aplicación.	Actualizar la librería afectada a una versión corregida, regenerar el build del frontend y establecer un proceso de revisión periódica de dependencias (npm audit, Dependabot, etc.).
Medio	Cabecera Content Security Policy (CSP) no configurada	Ruta Principal	No se ha definido una cabecera Content-Security-Policy, que controla desde qué orígenes se permiten scripts, estilos, iframes, imágenes, etc.	Facilita ataques como XSS o carga de recursos maliciosos, ya que el navegador no tiene restricciones claras sobre qué contenido ejecutar.	Definir una política CSP adecuada (ej. default-src 'self'; script-src 'self' ...) y aplicarla en las respuestas HTML principales.
Medio	Falta de cabecera Anti-Clickjacking	Ruta Principal	La respuesta no incluye cabeceras de protección contra clickjacking (X-Frame-Options o directiva frame-ancestors en CSP).	La aplicación puede ser embebida en un iframe malicioso y engañar al usuario para que haga clic en botones o enlaces sin darse cuenta (clickjacking).	Configurar X-Frame-Options: DENY o SAMEORIGIN, o bien usar Content-Security-Policy: frame-ancestors 'self'; u orígenes estrictamente permitidos.
Bajo	Falta encabezado X-Content-Type-Options	Recursos HTML/JS	No se envía la cabecera X-Content-Type-Options: nosniff, que indica al navegador que no debe por el mismo investigar el tipo de contenido.	Algunos navegadores podrían interpretar recursos como un tipo distinto al declarado, lo que abre la puerta a ciertos vectores de inyección o ejecución no deseada.	Añadir la cabecera X-Content-Type-Options: nosniff en las respuestas HTML/JS para todos los recursos servidos por la app.

Tabla 57: Riesgos de Seguridad OWASP ZAP

Nivel de Riesgo	Tipo	Apunta al archivo	Descripción	Riesgo	Solución
Bajo	Divulgación de Marcas de Tiempo – Unix	/static/js/ 469.bcf10ceb. chunk.js	En el contenido se exponen marcas de tiempo en formato Unix u otras referencias temporales internas.	Puede ayudar a un atacante a entender ciclos de despliegue, versiones o comportamiento interno, útil para reconocimiento.	Evitar exponer timestamps internos innecesarios en respuestas públicas o minimizar detalles que no aporten valor funcional al usuario.
Informativo	Aplicación Web Moderna	SPA REACT	ZAP detecta que el sitio se comporta como aplicación web moderna.	No es una vulnerabilidad en sí; indica que deben considerarse controles específicos para SPAs (protección de APIs, CORS, tokens.	Usar este hallazgo como recordatorio para complementar con pruebas a la API, controles de autenticación/autorización y protección de tokens en el frontend.
Informativo	Divulgación de información - Comentarios sospechosos	/static/js/ main.180add7d.js	En el código servido al cliente hay comentarios que podrían contener información interna (nombres de funciones, mensajes, pistas de lógica o configuraciones).	Un atacante puede usar esos comentarios como guía para comprender la lógica interna, endpoints o debilidades potenciales.	Limpiar comentarios sensibles del código que se despliega a producción y evitar mensajes que revelen detalles internos del sistema.
Informativo	Reexaminar las Directivas de Control de Caché	Página principal y rutas secundarias	La cabecera Cache-Control está ausente o no es suficientemente restrictiva para ciertos recursos que podrían considerarse sensibles.	Información potencialmente sensible podría quedar almacenada en caché de navegadores o proxies.	Para contenido sensible, usar Cache-Control: no-cache, no-store, must-revalidate. Para recursos estáticos “seguros”, definir políticas claras de cacheo.

Tabla 58: Riesgos de Seguridad OWASP ZAP

API.

Nivel de Riesgo	Tipo	Apunta al archivo	Descripción	Riesgo	Solución
Bajo	Cookie Sin Flag HttpOnly	/api/auth/login	Se establece una cookie sin el flag HttpOnly, por lo que puede ser accesible desde JavaScript en el navegador.	Si existiera un XSS, el atacante podría leer la cookie (por ejemplo, de sesión) y robar la sesión del usuario.	Configurar las cookies sensibles con flag HttpOnly para que no puedan ser leídas ni manipuladas desde JavaScript.
Bajo	Cookie Sin Flag de Seguridad (Secure)	/api/auth/login	La cookie no incluye el flag Secure, que obliga a enviarla solo por conexiones HTTPS.	En escenarios donde se exponga HTTP, la cookie podría viajar sin cifrado y ser interceptada por un atacante en la red (MITM).	Establecer el flag Secure en cookies de sesión y cualquier cookie sensible, asegurando que solo se envíen sobre HTTPS.
Bajo	El servidor divulga información mediante encabezado HTTP X-Powered-By	/api/auth/login	La respuesta incluye cabeceras como X-Powered-By con información sobre la tecnología/plataforma utilizada.	Facilita el fingerprinting de la tecnología usada, ayudando a un atacante a buscar exploits específicos para esa versión o stack.	Eliminar o anonimizar cabeceras como X-Powered-By y similares en el servidor o framework.
Informativo	Petición de Autenticación Identificada	/api/auth/login	ZAP identifica que esta petición corresponde a un flujo de autenticación (login), por el tipo de parámetros y respuesta.	No es una vulnerabilidad directa, pero señala que este endpoint es crítico y debe protegerse bien.	Usar este hallazgo como recordatorio para endurecer el endpoint: protección contra fuerza bruta, validaciones de entrada, login y monitoreo de intentos fallidos.

Tabla 59: Riesgos de Seguridad OWASP ZAP

Nivel de Riesgo	Tipo	Apunta al archivo	Descripción	Riesgo	Solución
Informativo	Respuesta de Gestión de Sesión Identificada	/api/auth/login	La respuesta contiene un token o identificador de sesión, cabeceras o cookies que gestionan la sesión del usuario.	Indica que la gestión de sesión se realiza en este punto; si estos tokens no se protegen adecuadamente podrían ser abusados.	Hay que asegurar que tokens/cookies tengan expiración adecuada, flags Secure y HttpOnly, y que se transmitan únicamente por HTTPS.
Informativo	Loosely Scoped Cookie	/api/auth/login	La cookie está definida con un ámbito más amplio de lo necesario, por ejemplo, aplicable a demasiadas rutas dentro del dominio.	Aumenta la superficie de ataque: otras rutas o sub-aplicaciones que no necesitan la cookie podrían acceder o verse afectadas por ella.	Restringir el ámbito de las cookies al dominio y ruta mínimos necesarios (FQDN y path específico) para la funcionalidad que las requiere.

Tabla 60: Riesgos de Seguridad OWASP ZAP

4.1.4 Pruebas de Rendimiento.

Resultados de la prueba CP_47: Prueba de rendimiento del listado de usuarios (GET /api/users)

Análisis del Summary Report

Posteriormente se revisa el listener Summary Report para obtener las métricas cuantitativas de la prueba:

Para el sampler POST – Login (obtener token) se registran:

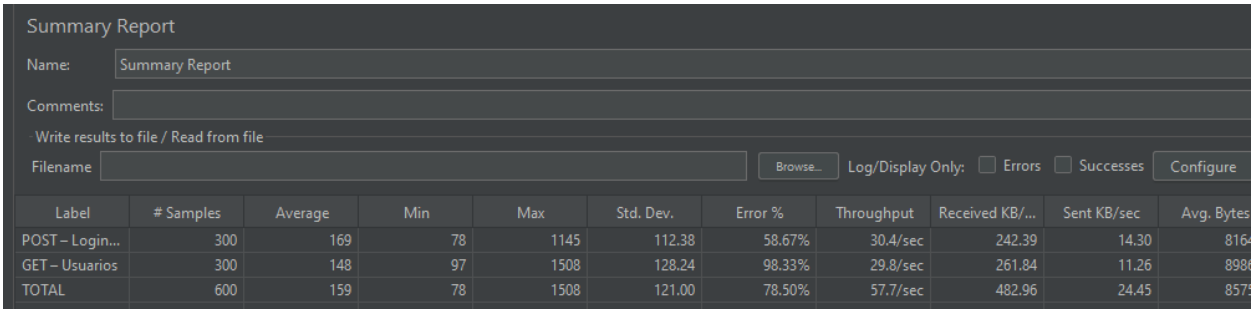


Tabla 61: Pruebas de Rendimiento Login

Nombre	**Respuesta**
# Samples	300 solicitudes.
Average:	169 ms.
Min	78 ms.
Max:	1145 ms.
Error %	58.67 %.
Throughput.	30.4 solicitudes/segundo.

Tabla 62: Tabla Resumen Peticiones Login

Para el sampler GET – Usuarios se registran:

Nombre	**Respuesta**
# Samples	300 solicitudes.
Average	148 ms.
Min	97 ms.
Max	1508 ms.
Error %	98.33 %.
Throughput	29.8 solicitudes/segundo

Tabla 63: Tabla Resumen Peticiones Login(GET)

En la fila TOTAL se observa:

- 600 solicitudes en total.

- Tiempo promedio 159 ms.
- Error % global: 78.50 %.
- Throughput acumulado: 57.7 solicitudes/segundo.

Al comparar estos resultados con los criterios de aceptación del caso CP_47 (tiempo promedio < 500 ms y Error % = 0 %), se obtiene lo siguiente:

Los tiempos promedios (169 ms para login y 148 ms para el listado de usuarios) se encuentran muy por debajo del límite de 500 ms, por lo que desde el punto de vista de rendimiento puro el sistema responde de forma rápida.

Sin embargo, el porcentaje de errores es elevado (58.67 % en el login y 98.33 % en el listado de usuarios), lo que implica que la mayoría de las solicitudes fallan durante la prueba concurrente. Estos errores corresponden principalmente a respuestas 401 Unauthenticated, detectadas previamente en el View Results Tree.

Conclusión de la prueba CP_47

- Con base en la información proporcionada por View Results Tree y Summary Report, se concluye que:
- El endpoint GET /api/users presenta tiempos de respuesta aceptables bajo carga concurrente (promedios inferiores a 200 ms).
- No obstante, la prueba CP_47 se clasifica como no satisfactoria, debido a que la tasa de error supera ampliamente el 0 % establecido en los criterios de aceptación.
- La principal causa de fallo es la gestión inadecuada del token de autenticación en escenarios concurrentes, lo que genera respuestas 401 y afecta la estabilidad global de la prueba.

Ejecución del caso CP_48 – Consultas relacionadas

Análisis del Summary Report

En el listener Summary Report se obtuvieron las métricas de rendimiento que se detallan a continuación:

POST – Login (obtener token)

Nombre	**Respuesta**
# Samples	2.
Average	438 ms
Min	272 ms
Max	605 ms
Error %	0.00 %
Throughput	52.5 solicitudes/minuto

Tabla 64: Rendimiento Consultas Relacionadas

GET – Users 50 p1

Nombre	Respuesta
# Samples	2.
Average	544 ms
Min	445 ms
Max	643 ms
Error %	0.00 %
Throughput	56.5 solicitudes/minuto

Tabla 65: Rendimiento Consultas Relacionadas (50)

GET – Users 100 p1

Nombre	Respuesta
# Samples	2.
Average	446 ms
Min	377 ms
Max	516 ms
Error %	0.00 %
Throughput	1.0 solicitud/segundo

Tabla 66: Rendimiento Consultas Relacionadas (100)

GET – Formats 50 p1

Nombre	Respuesta
# Samples	2.
Average	169 ms
Min	145 ms
Max	194 ms
Error %	0.00 %
Throughput	1.1 solicitud/segundo

Tabla 67: Rendimiento Consultas Relacionadas (50) - Fomatos

GET – Formats 100 p1

Nombre	Respuesta
# Samples	2.
Average	171 ms
Min	151 ms
Max	191 ms
Error %	0.00 %
Throughput	1.2 solicitudes/segundo

Tabla 68: Rendimiento Consultas Relacionadas (100) - Formatos

En la fila TOTAL se reportan:

- Samples: 10
- Average general: 353 ms
- Error % global: 0.00 %
- Throughput global: 2.8 solicitudes/segundo

Estos resultados muestran que todos los endpoints respondieron en menos de 600 ms en promedio y, lo más importante, no se registraron errores durante la ejecución.

Interpretación y conclusión de la CP_48

Al comparar los valores obtenidos con el objetivo de la prueba CP_48 evitar consultas excesivas y degradación de rendimiento al aumentar el tamaño de paginación y trabajar con relaciones se observa que:

El Error % = 0.00 % para todas las peticiones indica que no hubo fallos de autenticación ni errores de servidor.

El paso de per_page=50 a per_page=100 no provoca una degradación crítica del tiempo de respuesta:

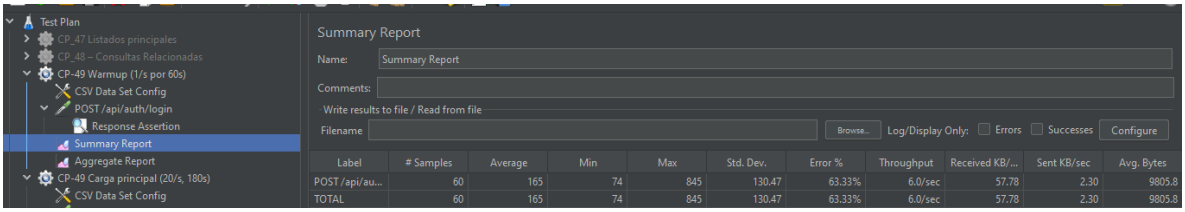
- GET – Users 50 p1 promedia 544 ms.
- GET – Users 100 p1 mejora a 446 ms, manteniéndose dentro de un rango aceptable.

Los módulos de formatos presentan tiempos aún más bajos (alrededor de 170 ms), confirmando que las consultas se resuelven de manera eficiente.

Por lo anterior, la prueba CP_48 – Consultas relacionadas se considera satisfactoria. El sistema demuestra que es capaz de gestionar listados con relaciones y distintos tamaños de paginación sin incrementar el número de consultas ni afectar negativamente el rendimiento.

Resultados de la prueba CP-49 Warmup (1/s por 60s)

En la primera fase de la prueba CP-49 se ejecutó el Thread Group “CP-49 Warmup (1/s por 60s)”, cuyo objetivo es “calentar” el servicio de autenticación con una carga moderada antes de aplicar la carga principal.



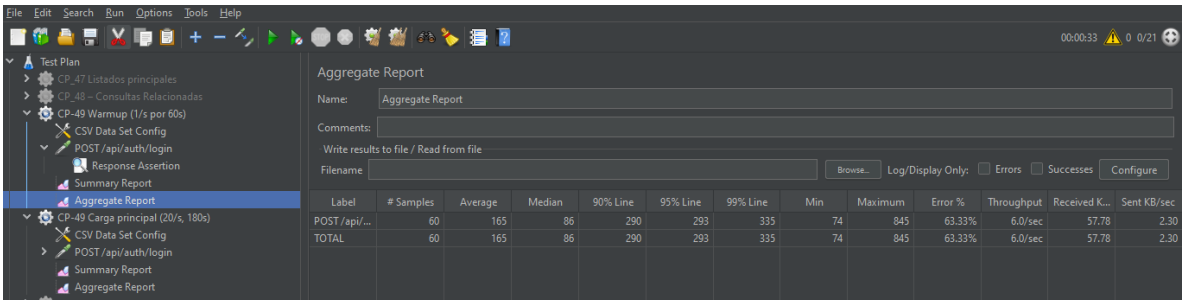


Tabla 69: Configuraciones JMeter

A partir del Summary Report y del Aggregate Report se obtuvieron las siguientes métricas para el sampler POST /api/auth/login:

A partir del Summary Report y del Aggregate Report se obtuvieron las siguientes métricas para el sampler POST /api/auth/login:

Nombre	Respuesta
# Samples	60 solicitudes de inicio de sesión.
Average (promedio):	165 ms.
Min	74 ms
Max	845 ms.
Median	86 ms.
90% Line	290 ms.
95% Line	293 ms.
99% Line	335 ms.
Error %	63.33 %.
Throughput	6.0 solicitudes por segundo.
Median	86 ms.

Tabla 70: 60 Solicitudes de Inicio de Sesión.

Interpretación:

Durante el warmup, el tiempo de respuesta del servicio es rápido (promedio 165 ms y percentil 95 de 293 ms, muy por debajo del límite de 800 ms definido para el caso). Sin embargo, el 63.33 % de los intentos de login fallan, lo que indica que, aun con una frecuencia relativamente baja (6 logins por segundo), el módulo de autenticación presenta errores frecuentes (principalmente respuestas 401/429).

Esta fase confirma que el sistema responde rápido, pero ya muestra inestabilidad en la autenticación incluso antes de aplicar la carga máxima.

Resultados de la fase Carga principal (20/s, 180s)

En la segunda fase se ejecutó el Thread Group “CP-49 Carga principal (20/s, 180s)”, diseñado para someter al módulo de autenticación a una alta concurrencia de inicios de sesión.

4.2 Cálculo y Análisis de Métricas de Calidad.

Las métricas se calcularon una vez ejecutadas las actividades de prueba del sistema. Estos indicadores permiten conocer el nivel actual de calidad del software y verificar el cumplimiento de los objetivos definidos en el plan de pruebas. Las métricas se calcularon siguiendo los lineamientos establecidos por las buenas prácticas ISTQB

4.2.1 Defectos por severidad

Definición: distribución de los defectos según su impacto en el negocio (Crítico, Alto, Medio o Bajo).

Fórmula:

$$\frac{\boldsymbol{Número\ de\ defectos\ por\ severidad}}{\boldsymbol{Total\ de\ defectos}} X\ 100$$

Durante la ejecución se registraron 4 defectos de tipo Bloqueo, los cuales corresponden a la categoría de defectos críticos, ya que impiden continuar con la

$$\boldsymbol{Defectos\ Críticos}: \frac{2}{18} X\ 100 = \mathbf{11.11\%}$$

$$\boldsymbol{Defectos\ Altos} = \frac{11}{18} X\ 100 = \mathbf{61.11\%}$$

$$\boldsymbol{Defectos\ Medios} = \frac{4}{18} X\ 100 = 22.22\%$$

$$\boldsymbol{Defectos\ Bajos} = \frac{1}{18} X\ 100 = 5.56\%$$

Resultado:

- Defectos Severidad Crítica: 2 casos fallidos (11.11%)
- Defectos Severidad Alta: 11 casos fallidos (61.11%)
- Defectos Severidad Media: 4 casos fallidos (22.22%)
- Defectos Severidad Baja: 1 caso fallidos (5.56%)

Objetivo: Se identifican una distribución de casi el 75% de casos que deben ser solventados y que no son negociables para la correspondiente puesta en producción del sistema.

Cumplimiento: No se cumple, debido a la presencia de 18 casos fallidos.

4.2.2 Porcentaje de casos exitosos

Definición: proporción de casos que cumplen el resultado esperado.

Fórmula:

$$\frac{\boldsymbol{Casos\ exitosos}}{\boldsymbol{Total\ de\ casos\ ejecutados}} \boldsymbol{X}\ 100$$

Cálculo obtenido:

- Casos Exitosos: 84
- Total, ejecutados: 106

$$\frac{84}{106} X\ 100 = 79.24\%$$

Resultado: 79.24%

Objetivo: ≥ 95%

Cumplimiento: No se cumple, indicando que aún existen fallos funcionales que deben corregirse.

4.2.3. Cobertura de ejecución de pruebas

Definición: porcentaje del total de casos de prueba que fueron ejecutados.

Fórmula:

$$\frac{\boldsymbol{Nu´mero\ total\ de\ casos\ ejecutados}}{\boldsymbol{Nu´mero\ total\ de\ casos\ planificados}} \boldsymbol{X}\ 100$$

Cálculo obtenido:

- Total, planificado: 106
- Total, ejecutado: 106

$$\frac{106}{106} X\ 100 = 100\%$$

Resultado: 100%

Objetivo: 100%

Cumplimiento: Se cumple completamente.

4.2.4 Cobertura funcional

Definición: porcentaje de funcionalidades planificadas que fueron probadas.

Fórmula:

$$\frac{\boldsymbol{Funcionalidades\ probadas}}{\boldsymbol{Total\ de\ funcionalidades}} \boldsymbol{X}\ 100$$

Cálculo obtenido:

Todas las funcionalidades contempladas en la matriz de pruebas fueron validadas.

$$\frac{20}{20} X\ 100 = 100\%$$

Resultado: 100%

Objetivo: ≥ 90%

Cumplimiento: Se cumple, alcanzando el 100% de cobertura del alcance funcional.

4.2.5 Eficiencia de detección

Definición: porcentaje de defectos detectados antes del pase a producción.

Fórmula:

$$\frac{\boldsymbol{Defectos\ QA}}{\boldsymbol{Total\ de\ defectos\ detectados}} \boldsymbol{X}\ 100$$

Cálculo obtenido

El sistema aún no ha sido desplegado en producción, sin embargo, se evitarán la fuga de 18 defectos para el sistema.

Por lo tanto, nuestra eficiencia de detección de defectos sería del 100%.

Resultado: No aplica en esta iteración.

Objetivo: ≥ 85%

Estado: Se calcula en fases posteriores, cuando existan datos de producción.

En total se ejecutaron 106 casos de prueba, alcanzando una cobertura del 100%. El 79.24% obtuvo un resultado exitoso, mientras que se identificaron 18 fallas y 4 bloqueos críticos, lo que impidió cumplir el objetivo de cero defectos críticos. La cobertura funcional fue del 100%, validando todas las funcionalidades definidas en el alcance. La eficiencia de detección no aplica aún, dado que el sistema no se encuentra en producción.

4.3 Análisis costo-beneficio e impacto de la automatización de pruebas.

Se evaluarán los beneficios obtenidos con la implementación de pruebas automatizadas en relación con el Sistema de Gestión de Contrato de Personal Docente de la FIA UES, comparándolos con las pruebas manuales. El análisis se realizará en base a tiempos, costo y aporte de calidad al producto, tomando en cuenta las recomendaciones establecidas en las normas de procesos y documentación de pruebas (ISO/IEC/IEEE 29119) y el informe de resumen de pruebas (IEEE829).

4.3.1 Supuestos y parámetros económicos para el análisis.

Precondiciones para el análisis.

Se realizó una investigación previa para verificar los salarios actuales para los que ejercen la profesión de tester y en promedio ronda los siguientes:

Puesto	Salario
QA Manual (QA Junior)	$500 - $850
QA Analist / Ingeniero QA (QA Senior)	$900 - $1200
QA Automation	$1400 - $2000

Tabla 71: Supuestos Salario de QA Automation

Para nuestro proceso de análisis nos centraremos en la posición de QA Automation con una paga del mínimo de $1400. Cálculo de pago por hora para el QA:

$$Pago\ por\ hora = \frac{\$1400}{30\ días * 8\ horas} = \$5.83\ dólares$$

Entonces por lo tanto tenemos que por hora el QA Automation percibe $5.83 dólares.

Para nuestro proyecto las pruebas manuales se ejecutaron con el detalle siguiente:

- 3 iteraciones cada año.

Así mismo se esta considerando un tiempo para la curva de aprendizaje y que el tester logre conocer el sistema y con ello lograr ejecutar de forma exitosa los casos de pruebas donde tendremos lo siguientes datos:

- Días reservados: 6 días.
- Costo Por Día:

$$Pago\ Por\ Tiempo\ Invertido = 48\ horas\ x\ \$5.83 = \$279.84\ doláres$$

Por lo tanto, tenemos que el tester para conocer el sistema y lograr cubrir con su curva de aprendizaje será destinado un total de $279.84 dólares los cuales deberán ser considerados en el calculo total de la ejecución de las pruebas manuales y estos serán incluidos en la tabla general bajo el nombre de “Tiempo de Exploración”.

4.3.2 Situación Inicial: proceso de pruebas manual.

Costo Total de Pruebas Manuales.

Módulo	Casos ejecutados	Tiempo total manual (horas)	Tiempo por caso (Costo / Hora)	Costo
Actividades de Docentes	3 casos	0.50	$5.83	$2.92
Bancos	5 casos	1.42	$5.83	$8.26
Bitácora	5 casos	1.83	$5.83	$10.69
Candidatos	7 casos	1.67	$5.83	$9.72
Carga Académica	3 casos	0.83	$5.83	$4.86
Cargos	2 casos	0.75	$5.83	$4.37
Ciclo	6 casos	2.08	$5.83	$12.15
Contratación	15 casos	7.00	$5.83	$40.81
Empleados	2 casos	1.00	$5.83	$5.83
Escalafón	6 casos	2.08	$5.83	$12.15
Escuelas	5 casos	1.58	$5.83	$9.23
Facultades	5 casos	2.50	$5.83	$14.58
Formatos	3 casos	1.50	$5.83	$8.75
Grupos	2 casos	0.67	$5.83	$3.89
Rendimiento	4 casos	4.00	$5.83	$23.32
Revisión de Documentos	7 casos	3.92	$5.83	$22.83
Seguridad	11 casos	4.00	$5.83	$23.32
Tipos de Grupos	2 casos	1.00	$5.83	$5.83
Usabilidad	2 casos	1.00	$5.83	$5.83
Usuarios	11 casos	6.83	$5.83	$39.84
Tiempo de Exploración	N/A	48.00	$5.83	$279.84
TOTAL	**106 casos**	**94.17 horas**	**$5.83**	**$548.99**

Tabla 72: Costo Total de Pruebas Manuales

Para el QA Automation se tardó en total aproximado **94 horas y en días un total de 12 días** y dando como resultado un costo para las pruebas manuales de **$548.99 (aproximaciones)** en la ejecución de verificación de pruebas manuales por ciclo.

4.3.3 Proceso de Automatización.

Para el desarrollo del código y lograr la automatización contamos con el detalle siguiente:

Módulo	Desarrollo de Scripts automatizados	Costo por hora	Costo Total
Actividades de Docentes	2 horas	$5.83	$11.66
Bancos	1 hora	$5.83	$5.83
Bitácora	2 horas	$5.83	$11.66
Candidato	5 horas	$5.83	$29.15
Carga Académica	2 horas	$5.83	$11.66
Ciclo	2 horas	$5.83	$11.66
Contratación	24 horas	$5.83	$139.92
Escuelas	2 horas	$5.83	$11.66
Facultades	2 horas	$5.83	$11.66
Formatos	2 horas	$5.83	$11.66
Grupos	2 horas	$5.83	$11.66
Revisión de Documentos	4 horas	$5.83	$23.32
Seguridad	4 horas	$5.83	$23.32
Tipos de Grupos	2 horas	$5.83	$11.66
Usuarios	12 horas	$5.83	$69.96
Total	68 horas	$5.83	$396.44

Tabla 73: Costo Proceso de Automatización.

En resumen, para la implementación de la automatización el desarrollo de los scripts por módulo, el Automation invertirá un total de **68 horas alrededor de 8 días y medio** con un costo total de **$396.44** dólares como inversión inicial del proyecto de automatización.

4.3.4 Costo Total del Proyecto.

Módulo	Costo Pruebas Manuales	Costo Desarrollo Scripts	Total
Actividades de Docentes	$2.92	$11.66	$14.58
Bancos	$8.26	$5.83	$14.09
Bitácora	$10.69	$11.66	$22.35
Candidato	$9.72	$29.15	$38.87
Carga Académica	$4.86	$11.66	$16.52
Cargos	$4.37	$-	$4.37
Ciclo	$12.15	$11.66	$23.81
Contratación	$40.81	$139.92	$180.73
Empleados	$5.83	$-	$5.83
Escalafón	$12.15	$-	$12.15
Escuelas	$9.23	$11.66	$20.89
Facultades	$14.58	$11.66	$26.24
Formatos	$8.75	$11.66	$20.41
Grupos	$3.89	$11.66	$15.55
Rendimiento	$23.32	$-	$23.32
Revisión de Documentos	$22.83	$23.32	$46.15
Seguridad	$23.32	$23.32	$46.64
Tipos de Grupos	$5.83	$11.66	$17.49
Usabilidad	$5.83	$-	$5.83
Usuarios	$39.84	$69.96	$109.80
Tiempo de Exploración	$279.84	$-	$279.84
Total	$548.99	$396.44	$945.43

Tabla 74: Costo Total del Proyecto.

Total, de costo de desarrollo inicial del proyecto **$945.43 con 162 horas trabajadas.**

4.3.5 Situación Posterior con Pruebas Automatizadas.

Con las automatizaciones realizadas algunos de los casos de pruebas han quedado de manera parcial, es decir que hasta cierto porcentaje en la ejecución es necesaria la intervención humana, dando como resultado lo siguiente:

Ahorro Monetario.

Módulo	Porcentaje de ahorro por automatización	Costo de pruebas manuales	Ahorro total por ejecución	Horas Ahorradas
Actividades de Docentes	100%	$-	$2.92	0.5
Bancos	100%	$-	$8.26	1.4
Bitácora	100%	$-	$10.69	1.8
Candidato	50%	$4.86	$4.86	0.8
Carga Académica	100%	$-	$4.86	0.8
Cargos	0%	$4.37	$0.00	0.00
Ciclo	100%	$-	$12.15	2.10
Contratación	100%	$-	$40.81	7.00
Empleados	0%	$5.83	$-	0.00
Escalafón	0%	$12.15	$0.00	0.00
Escuelas	100%	$-	$9.23	1.60
Facultades	100%	$-	$14.58	2.50
Formatos	100%	$-	$8.75	1.50
Grupos	100%	$-	$3.89	0.70
Rendimiento	0%	$23.32	$-	0.00
Revisión de Documentos	80%	$5.06	$17.77	3.13
Seguridad	100%	$-	$23.32	4.00
Tipos de Grupos	100%	$-	$5.83	1.00
Usabilidad	0%	$5.83	$-	0.00
Usuarios	100%	$-	$39.84	6.80
Tiempo de Exploración	0%	$279.84	$-	0.00
Total		$341.26	$207.73	36

Tabla 75: Ahorro Monetario.

Comparación del Esfuerzo Sin y Con Automatización.

En la situación inicial, sin pruebas automatizadas, la ejecución completa de un ciclo de pruebas manuales requiere de aproximadamente 94.17 horas trabajadas, lo que equivale a 12 días laborales (Jornadas de 8 horas según ley).

Tras la implementación de las pruebas de las pruebas automatizadas, el esfuerzo de ejecución por ciclo se reduce a alrededor de 58.17 horas trabajadas lo que equivale a 7 días laborales. En términos prácticos, esto implica una reducción de 36 horas trabajadas por iteración, que puede interpretarse como 5 días laborales completos que el equipo deja de invertir en la ejecución repetitiva de pruebas de regresión.

Proyectando este comportamiento alrededor de un año con 3 iteraciones de pruebas, el escenario sin automatización tendrá un total de 282.51 horas de esfuerzo anual, equivalente a 35 días laborales, mientras que con la automatización el esfuerzo se reduce a alrededor de 174.51 horas, es decir a 22 días laborales. La automatización permitirá liberar 108 horas de trabajo al año, lo que se traduce a aproximadamente 14 días laborales completos que pueden reasignarse a actividades de mayor valor agregado, como el diseño de nuevos casos de pruebas, pruebas exploratorias y acciones de mejora continua.

4.3.6 Cálculo del ROI

Definimos las siguientes variables para el calculo

- **N**: número de iteraciones = 3
- **Costo**: costo manual con pruebas manuales = $548.99
- **Costo Residual**: costo por iteración luego de automatización: $341.26
- **I**: inversión inicial de automatización = $396.44

Costos anuales con y sin automatización.

Escenario sin automatización.

$$\boldsymbol{Costo\ anual\ sin\ automatización = N\ x\ C_{manual}}$$

$$\boldsymbol{Costo\ anual\ sin\ automatización = 3\ x\ \$548.99 = \$1,646.97\ dólares.}$$

Escenario con automatización (Primer año).

$$\boldsymbol{Costo\ anual\ con\ automatización = I + (N\ x\ C_{residual})}$$

$$\boldsymbol{Costo\ anual\ con\ automatización = \$396.44 + (\$341.26X3) = \$1,420.22\ dólares}$$

$$\boldsymbol{Beneficio\ Año\ 1 = \$1,646.97 - \$1,420.22 = \$226.75\ dólares}$$

Comparado con el escenario totalmente manual, cuyo costo anual sería de 1,646.97 USD, la automatización genera un ahorro neto de 226.75 USD en el primer año (considerando tres iteraciones de pruebas).

Escenario con automatización (Años Posteriores).

A partir del segundo año, la inversión inicial del desarrollo de scripts ya ha sido realizada, por lo que únicamente se incurre en el costo de ejecución manual residual de cada ciclo:

$$\boldsymbol{Costo\ Anual\ Año\ 2 = 3\ x\ \$341.26 = \$1,023.78\ dólares}$$

Para este escenario, el ahorro anual respecto al esquema manual es:

$$\boldsymbol{Beneficio\ Año\ 2 = \$1,646.97 - \$623.19 = \$1,023.78\ dólares.}$$

Es decir, a partir del segundo año la automatización nos permitirá ahorrar más de $600.00 dólares anuales

Cálculo del ROI.

El retorno de la inversión (ROI) se calcula a partir del beneficio generado respecto a la inversión inicial en la automatización.

$$ROI = \frac{Beneficio\ Neto}{Inversión}\ x\ 100$$

Para el primer año.

- Inversión inicial: $396.44 dólares.
- Beneficio neto: diferencia entre el costo anual sin automatización y el costo anual con automatización: $226.75 dólares.

Sustituyendo en la fórmula:

$$ROI = \frac{\$226.75}{\$396.44}\ x\ 100 = 57.20$$

Este resultado nos indica que, en el primer año, además de recuperar parcialmente la inversión se obtiene un retorno equivalente al 57.19% de lo invertido.

Si analizamos el periodo de recuperación (payback) considerando el ahorro por ciclo de pruebas:

$$\boldsymbol{Ahorro\ Por\ Ciclo = \$548.99 - \$341.26 = \$207.73\ dólares}$$

$$\boldsymbol{Ciclos\ para\ recuperar\ inversión = {}^{\$396.44}/_{\$207.73} = 1.91\ ciclos}$$

Dado que se supone que se efectuarán tres ciclos de pruebas por año, la inversión se recupera en la práctica durante el primer año de operación del sistema.

4.3.7 Conclusiones del análisis costo-beneficio.

La automatización de pruebas de calidad del Sistema de Gestión de Contrato de Personal Docente implica una inversión inicial de $396.44 dólares, pero permite reducir el costo de ejecución por ciclo completo de pruebas de regresión de $548.99 dólares a aproximadamente $341.26 dólares lo que representa un ahorro de 207.73 USD por ciclo.

Bajo el supuesto de tres iteraciones anuales, el costo anual sin automatización ascendería a 1,646.97 USD, mientras que con la automatización el costo del primer año se reduce a 1,420.22 USD y, a partir del segundo año, a 1,023.78 USD, generando ahorros anuales de 226.75 USD en el primer año y de 623.19 USD en los años posteriores. El ROI obtenido para el primer año es de aproximadamente 57.20 % y el período de recuperación de la inversión se sitúa alrededor de 1.9 ciclos de pruebas, lo que evidencia la viabilidad económica y operativa de la automatización implementada y respalda su adopción como estrategia de mejora del proceso de aseguramiento de la calidad..

4.4 Lecciones Aprendidas y Recomendaciones

4.4.1 Lecciones Aprendidas

1. **La automatización acelera el trabajo y mejora la calidad:** Se comprobó que los flujos repetitivos como creación de cargas, validación de permisos, revisión de grupos o

detección de duplicados son mucho más eficientes cuando se automatizan, evitando errores humanos y reduciendo tiempos de regresión.

2. **Las pruebas manuales siguen siendo necesarias:** Aunque la automatización aporta grandes beneficios, la revisión de documentos, validaciones visuales y decisiones basadas en criterio humano no pueden automatizarse completamente.

3. **El diseño de casos de prueba claros evita confusiones:** Establecer precondiciones, pasos, datos de entrada y resultados esperados ayuda a mantener orden, facilita la reproducción de errores y mejora la comunicación entre QA y desarrollo.

4. **La comunicación efectiva entre el equipo es clave:** Los cambios en reglas de negocio, validaciones o endpoints afectaron varios escenarios, demostrando lo importante que es mantener alineación continua entre QA, frontend, backend.

5. **La versión y la evidencia estructurada simplifican auditorías:** Mantener logs, capturas de fallos de pruebas manuales, scripts de pruebas automatizadas y reportes permitió rastrear problemas, justificar decisiones y documentar todo el proceso de calidad del proyecto.

6. **Considerar tiempos en procesos de aprendizaje de herramientas para el equipo:** Contemplar dentro de la planificación tiempos prudenciales para la curva de aprendizaje de los Tester en el manejo de nuevas herramientas**.**

4.4.2 Recomendaciones

1. **Completar la automatización para los flujos de candidatos.** Con esto se reduce el tiempo de regresión, mejora la estabilidad del sistema y disminuye entre un 90% el uso de horas hombre.

2. **Implementación de las pruebas automáticas.** Ejecutar pruebas en cada error o despliegue ayudaría a capturar errores de manera temprana, evitando incidentes en producción.

3. **Mantener un repositorio único y ordenado de scripts.** Esto facilita el mantenimiento, evita duplicidad y permite a nuevos integrantes comprender rápidamente los flujos.

4. **Actualizar los casos de prueba después de cada cambio funcional.** Si el sistema cambia constantemente, se recomienda que la documentación y pruebas de QA deben evolucionar también.

5. **Fortalecer la documentación técnica y funcional.** Una mejor documentación reduce retrasos, facilita el análisis de impacto y mejora la toma de decisiones para las pruebas.

6. **Asignar roles claros dentro del equipo de QA.** Definir responsables para pruebas manuales, automatizadas, revisión de documentos y análisis de incidencias optimiza el trabajo y reduce cuellos de botella.

7. **Implementar CI/CD una vez se obtenga los insumos.** Es necesario que la automatización sea incorporada dentro de los despliegues de nuevas versiones que se

realicen de la aplicación para que funcionen de manera síncrona con las nuevas versiones a implementar.

8. **Realizar capacitaciones internas sobre automatización.** Incrementar el dominio de herramientas como Selenium, Postman y monitoreo eleva significativamente la eficiencia del equipo.

4.5 Control de versiones y almacenamiento de evidencias.

Para garantizar la trazabilidad, integridad, y disponibilidad de los entregables del proyecto, se estableció una estrategia de la configuración del código fuente de la aplicación y la gestión documental de las evidencias.

4.5.1 Gestión del Código Fuente y Artefactos Técnicos.

El control de versiones del código fuente lo centralizamos a través de la herramienta Git sirviéndonos como sistema de control distribuido, alojado en el repositorio central en la plataforma GitHub.

Se implemento una metodología de flujo de trabajo basada en Feature Branch, al cual permite aislar el desarrollo de nuevas funcionalidades de la versión estable del producto. La estructura de las ramas se definió de la siguiente forma:

- Main/Master: es el que contiene únicamente el código fuente que se encuentra sin errores y estable para nuestro producto final.
- Develop: Rama de integración donde se unifican los cambios que son aprobados.
- Feature: Ramas temporales para el desarrollo de módulos específicos.

Cada “Commit” lleva una estructura que nos permita identificar el cambio que fue cargado y facilitar la auditoria del progreso técnico.

4.5.2 Almacenamiento y Organización de Evidencias.

El Almacenamiento de la documentación administrativa, manual y evidencias de las pruebas se centralizo en una carpeta en OneDrive bajo denominación “Evidencias Pruebas Manuales”. También la guía técnica del proceso de pruebas automatizadas se encuentra en la carpeta que lleva por nombre “Guía Técnica de Pruebas de Calidad”.

- Repositorio de GitHub: Repositorio de Pruebas Automatizadas
- Matriz de Pruebas: MATRIZ DE PRUEBAS.xlsx
- Evidencias de Pruebas Realizadas: Evidencias_Pruebas_Manuales.pdf
- Manual Técnico de Pruebas Automatizadas: Manual Tecnico de Pruebas de Calidad al Sistema de Contratos de Personal Docente FIA UES.pdf
- Acta de Cierre: Acta de Cierre.pdf
- Tablero PowerBI: TABLERO SEGUIMIENTO PRUEBAS DE CALIDAD GRUPO 03.pdf

4 Conclusiones

El desarrollo del proyecto nos permitió poder confirmar que la ausencia de un proceso de pruebas de calidad dentro del sistema de Gestión de Contratos del Personal Docente de la Facultad de Ingeniería y Arquitectura de la Universidad de El Salvador representa un riesgo significativo para la confiabilidad. Con la implementación de este plan estructurado de pruebas, que combina enfoques manuales y automatizados, apoyará considerablemente a la calidad del software, garantizando que los procesos críticos de contratación, asignación de ciclos y validación de documentos se ejecuten de manera estable, segura, transparente. Cumpliendo con los requerimientos funcionales y no funcionales según lo documentado.

El proyecto también evidencio la importancia de alinear los procesos de pruebas con estándares internacionales como la norma que aplicamos ISO/IEC 29119, dicha norma nos proporcionó un marco metodológico que nos ayudó con la planificación, diseño, documentación y futura ejecución de las pruebas. Al implementar esta norma aumentamos la credibilidad institucional al demostrar que los procesos de desarrollo tecnológico de la Universidad de El Salvador siguen los lineamientos reconocidos a nivel mundial. El enfoque también ayudará a que en futuras auditorias que se realicen al sistema se cuente con un proceso establecido, documentado y verificado que el sistema cumple con los criterios de calidad.

Esta fase nos permitió establecer un marco estratégico claro, una ruta de implementación para dar solución a la carencia de pruebas de calidad al sistema y los fundamentos necesarios para avanzar hacia la etapa de ejecución. El trabajo desarrollado no solo aporta un plan de aseguramiento de calidad para el sistema, sino que también sienta precedentes para futuros proyectos que la universidad desarrolle, contribuyendo al fortalecimiento de la cultura institucional de calidad.

Durante la ejecución del proceso de pruebas de calidad se nos permitió también comprobar como beneficia la automatización de pruebas, en donde para nuestro caso se logró optimizar el tiempo para el encargado de las pruebas de calidad del sistema.

Así mismo el proyecto permitirá ser adaptado, replicarse y utilizarse para nuevos sistemas que sean desarrollados dentro de la Universidad de El Salvador aprovechando la reutilización de casos de prueba, guías técnicas, plantillas de reportes y scripts automatizados, lo que reduce el esfuerzo de implementación en futuros proyectos.

5 Glosario

5.1 Glosario de Términos

Término	Definición
Ambiente de Pruebas	Entorno controlado en el que se ejecutan los casos de prueba, separado del ambiente de producción, con el fin de evitar riesgos sobre el sistema real.
Automatización de Pruebas	Proceso de diseñar y ejecutar pruebas mediante herramientas que simulan la interacción del usuario con el sistema, reduciendo esfuerzo manual y errores humanos.
Bitácora (Log)	Registro cronológico de eventos y actividades del sistema, útil para auditoría, depuración y trazabilidad de errores.
Caso de Prueba	Conjunto de condiciones, pasos y datos diseñados para comprobar que una funcionalidad del sistema se comporta de acuerdo con los requisitos establecidos.
Cobertura Funcional	Porcentaje de funcionalidades planificadas que han sido probadas, asegurando que el software ha sido evaluado en la mayoría de sus procesos.
Cobertura de Pruebas	Medida que indica cuántos casos de prueba planificados se han ejecutado en comparación con el total definido.
Criterio de Aceptación	Condiciones mínimas que debe cumplir el sistema para ser considerado válido por parte de los usuarios o clientes.
Criterio de Salida	Reglas que determinan cuándo un conjunto de pruebas puede considerarse finalizado, basadas en métricas como cobertura, porcentaje de éxito o ausencia de defectos críticos.
Defecto (Bug)	Error, falla o desviación respecto al comportamiento esperado del sistema.
Defecto Crítico	Anomalía en el sistema que impide el funcionamiento de procesos esenciales y que debe ser corregida antes de la liberación del software.
Defecto por Severidad	Clasificación de un defecto en función del nivel de impacto que genera en el sistema (crítico, alto, medio, bajo).
Eficiencia de Detección	Porcentaje de defectos descubiertos durante la fase de pruebas en comparación con el total de defectos identificados en producción.
Ejecución de Pruebas	Fase del ciclo de pruebas en la que los casos diseñados se ejecutan de manera manual o automatizada, recopilando resultados.
Especificación de Requisitos	Documento que describe las funciones, restricciones y objetivos que debe cumplir el sistema informático.

Evidencia de Prueba	Documentación que respalda la ejecución de pruebas, como capturas de pantalla, logs, reportes o grabaciones.
Gestión de Pruebas	Conjunto de actividades de planificación, organización y control del proceso de pruebas para asegurar calidad y trazabilidad.
Informe de Pruebas	Documento que consolida resultados, defectos detectados, métricas de cobertura y conclusiones sobre la calidad del sistema probado.
Informe de Resultados de Prueba	Documento que presenta un resumen de los casos ejecutados, clasificando los resultados en exitosos, fallidos o pendientes.
ISO/IEC 29119	Estándar internacional que define procesos, técnicas y documentación para la correcta ejecución de pruebas de software.
Mantenibilidad	Capacidad del software para ser actualizado, corregido o mejorado sin afectar su funcionamiento general.
Matriz de Riesgo	Herramienta que evalúa los riesgos del sistema asignando valores de probabilidad e impacto, permitiendo priorizar los más críticos.
Matriz de Trazabilidad	Herramienta que vincula requisitos, casos de prueba y resultados, asegurando cobertura total y control de cambios.
Métrica de Prueba	Indicador cuantitativo que permite medir aspectos del proceso de pruebas, como defectos encontrados, cobertura alcanzada o porcentaje de éxito.
Plan de Pruebas	Documento formal que define el enfoque, los recursos, el alcance, la estrategia y la programación de las actividades de prueba.
Prueba de Carga	Tipo de prueba no funcional que mide el desempeño del sistema bajo una cantidad específica de usuarios o transacciones simultáneas.
Prueba de Estrés	Prueba no funcional que evalúa el comportamiento del sistema en condiciones extremas de carga, superiores a las previstas en producción.
Prueba de Penetración	Evaluación de seguridad en la que se simula un ataque para descubrir vulnerabilidades en el sistema.
Prueba de Regresión	Técnica que consiste en ejecutar nuevamente pruebas ya realizadas tras un cambio en el software, para verificar que no se introdujeron errores adicionales.
Prueba de Usabilidad	Evaluación de la facilidad de uso y experiencia de usuario del sistema, verificando que sea intuitivo y accesible.
Prueba Exploratoria	Técnica en la que el evaluador navega libremente por el sistema, sin guiones predefinidos, con el fin de descubrir comportamientos inesperados.

Pruebas Funcionales	Validaciones que verifican que el sistema cumple con las funciones definidas en los requisitos.
Reporte de Defectos	Documento que describe los errores identificados, incluyendo pasos para reproducirlos, severidad, prioridad y estado de resolución.
Script de Prueba	Conjunto de instrucciones automatizadas que permiten ejecutar de forma repetible los casos de prueba.
Script Automatizado	Archivo que contiene instrucciones codificadas para ejecutar de manera repetida un conjunto de pruebas sobre el sistema.
Severidad	Grado de impacto que tiene un defecto sobre la operatividad del sistema (crítico, alto, medio, bajo).
Sistema bajo Prueba (SuT)	Aplicación, módulo o componente específico del sistema que se encuentra en evaluación durante el proceso de pruebas.
Trazabilidad de Requisitos	Relación que asegura que cada requisito definido está cubierto por uno o varios casos de prueba.
Usuario Final	Persona que interactuará directamente con el sistema en el entorno productivo, cuyas necesidades y expectativas deben ser satisfechas mediante las pruebas.
Validación	Proceso de confirmar que el sistema satisface las necesidades y expectativas de los usuarios finales.
Verificación	Proceso de confirmar que el software fue construido correctamente conforme a las especificaciones técnicas y de diseño.
Vulnerabilidad	Debilidad en el sistema que puede ser explotada para comprometer la seguridad o integridad de los datos.

5.2 Acrónimos y Abreviaturas

- **API**: Application Programming Interface – Interfaz de Programación de Aplicaciones.
- **CMMI**: Capability Maturity Model Integration – Modelo Integrado de Madurez de Capacidades.
- **FIA**: Facultad de Ingeniería y Arquitectura (Universidad de El Salvador).
- **FODA**: Fortalezas, Oportunidades, Debilidades y Amenazas.
- **IEEE**: Institute of Electrical and Electronics Engineers.
- **ISO**: International Organization for Standardization – Organización Internacional de Normalización.
- **IEC**: International Electrotechnical Commission – Comisión Electrotécnica Internacional.
- **ISTQB**: International Software Testing Qualifications Board.
- **QA**: Quality Assurance – Aseguramiento de la Calidad.
- **SQA**: Software Quality Assurance – Aseguramiento de la Calidad del Software.

- **SPICE**: Software Process Improvement and Capability Determination – Mejora de Procesos de Software y Determinación de Capacidades.
- **TGS**: Teoría General de Sistemas.
- **UAT**: User Acceptance Testing – Pruebas de Aceptación por el Usuario.
- **UES**: Universidad de El Salvador.
- **UML**: Unified Modeling Language – Lenguaje de Modelado Unificado.
- **V-Model**: Verification and Validation Model – Modelo de Verificación y Validación.

6 Bibliografía

ISO/IEC. (2011). *ISO/IEC 25010: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. International Organization for Standardization.

Pressman, R. S. (2014). *Ingeniería del software: Un enfoque práctico* (7.ª ed.). McGraw-Hill.

Sommerville, I. (2011). *Software Engineering* (9th ed.). Addison-Wesley.

IEEE. (2014). *IEEE Standard for Software Quality Assurance Processes (IEEE Std 730-201*

Constanzo, M. (2014). Comparación de modelos de calidad, factores y métricas en el ámbito de la ingeniería de software. https://dialnet.unirioja.es/descarga/articulo/5123569.pdf4). Institute of Electrical and Electronics Engineers.

IEEE. (1998). *IEEE standard for software test documentation (IEEE Std 829-1998)*. Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/IEEESTD.1998.88286

ISO/IEC/IEEE. (2013). *Software and systems engineering — Software testing — Part 1: Concepts and definitions (ISO/IEC/IEEE 29119-1:2013)* [Withdrawn standard]. ISO.

ISTQB. (2018). *Standard glossary of terms used in software testing* (Version 3.2). International Software Testing Qualifications Board. https://www.istqb.org

Myers, G. J. (1979). *The art of software testing*. John Wiley & Sons.

7 Anexos

Anexo 1- Entrevista para el Levantamiento de Requerimientos Funcionales

Proyecto: Sistema Informático para la Gestión de Contratos de Personal Docente – FIA-UES

Fecha de aplicación: 08 de septiembre de 2025

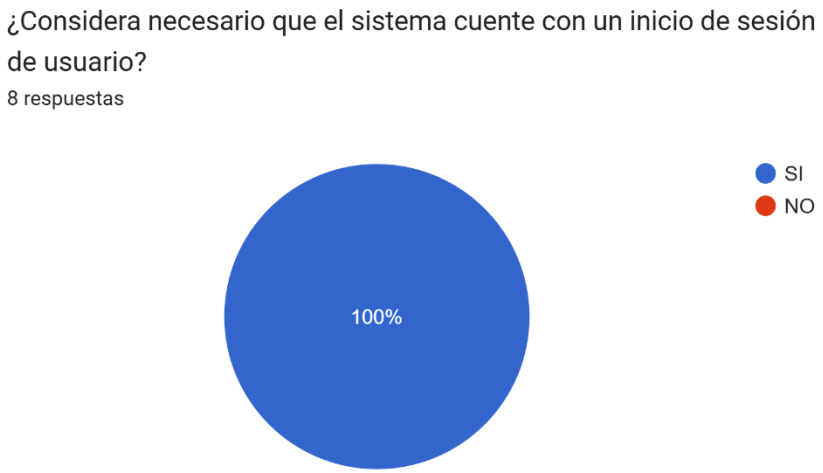
Medio: Entrevista estructurada / Encuesta en línea

Participantes: Autoridades de la Facultad, Administradores del sistema, Recursos Humanos, Coordinadores académicos

Cuestionario aplicado

Pregunta 1. ¿Considera necesario que el sistema cuente con un inicio de sesión seguro y controlado por roles de usuario?

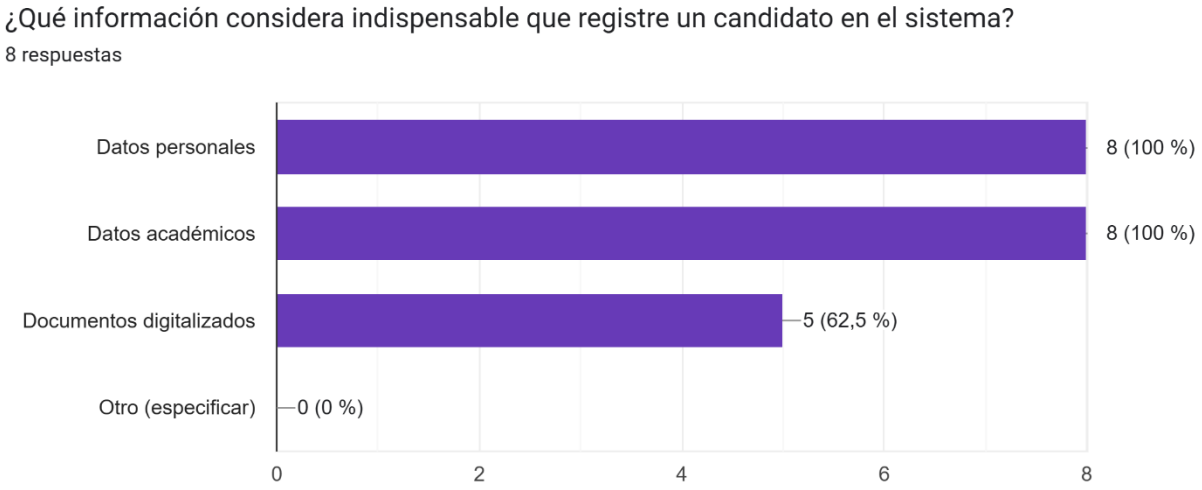
- 1- SI
- 2- NO



(Requerimiento derivado: RF-01 – Autenticación y control de accesos)

Pregunta 2. ¿Qué información considera indispensable que registre un candidato en el sistema?

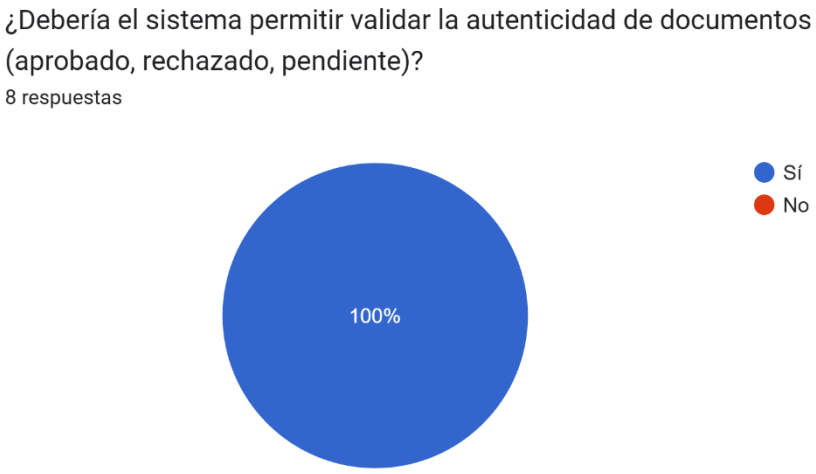
- Datos personales
- Datos académicos
- Documentos digitalizados
- Otro (especificar)



(Requerimiento derivado: RF-02 – Registro de candidatos)

Pregunta 3. ¿Debería el sistema permitir validar la autenticidad de documentos cargados y asignar estados (aprobado, rechazado, pendiente)?

- Sí
- No

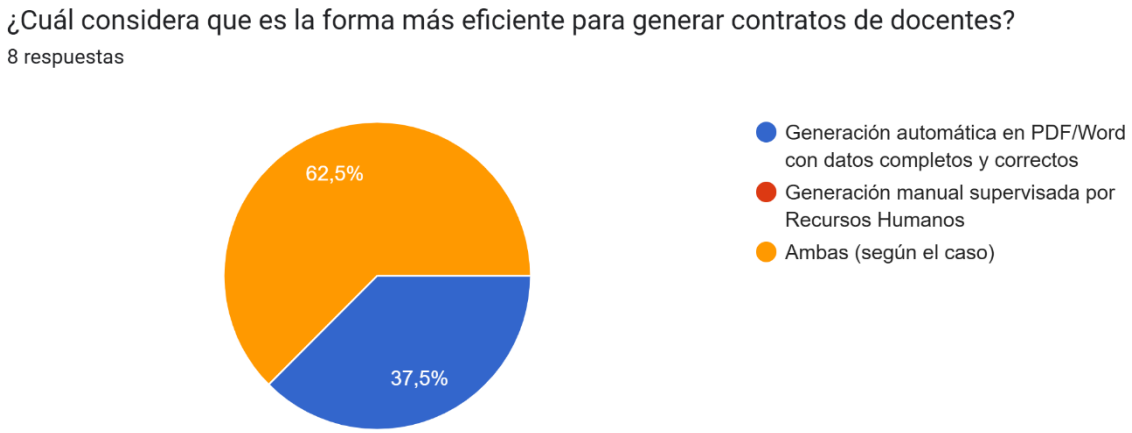


(Requerimiento derivado: RF-03 – Validación de documentos)

Pregunta 4. ¿Cuál considera que es la forma más eficiente para generar contratos de docentes?

- Generación automática en PDF/Word con datos completos y correctos
- Generación manual supervisada por Recursos Humanos

- Ambas (según el caso)



(Requerimiento derivado: RF-04 – Generación de contratos)

Pregunta 5. ¿El sistema debería permitir la asignación de materias y horarios a los docentes contratados?

- Sí
- No

(Requerimiento derivado: RF-05 – Asignación de carga académica)

Pregunta 6. En relación con los ciclos académicos, ¿qué nivel de gestión considera más importante?

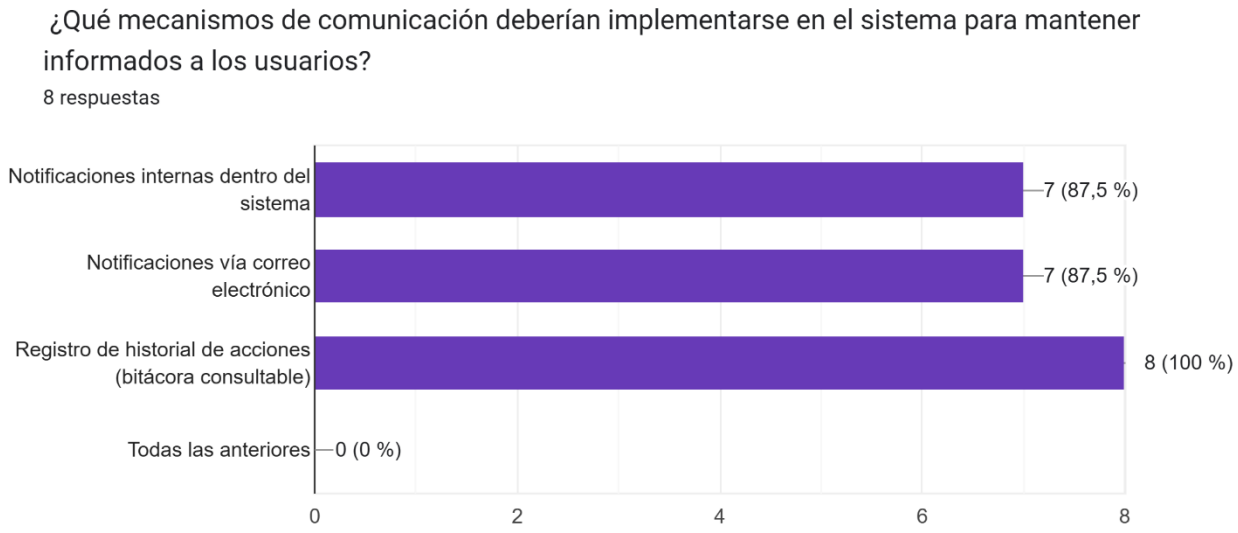
- Solo creación de ciclos académicos
- Creación y edición de ciclos en curso

- Creación, edición y cierre formal de ciclos con control de historial

(Requerimiento derivado: RF-06 – Gestión de ciclos académicos)

Pregunta 7. ¿Qué mecanismos de comunicación deberían implementarse en el sistema para mantener informados a los usuarios?

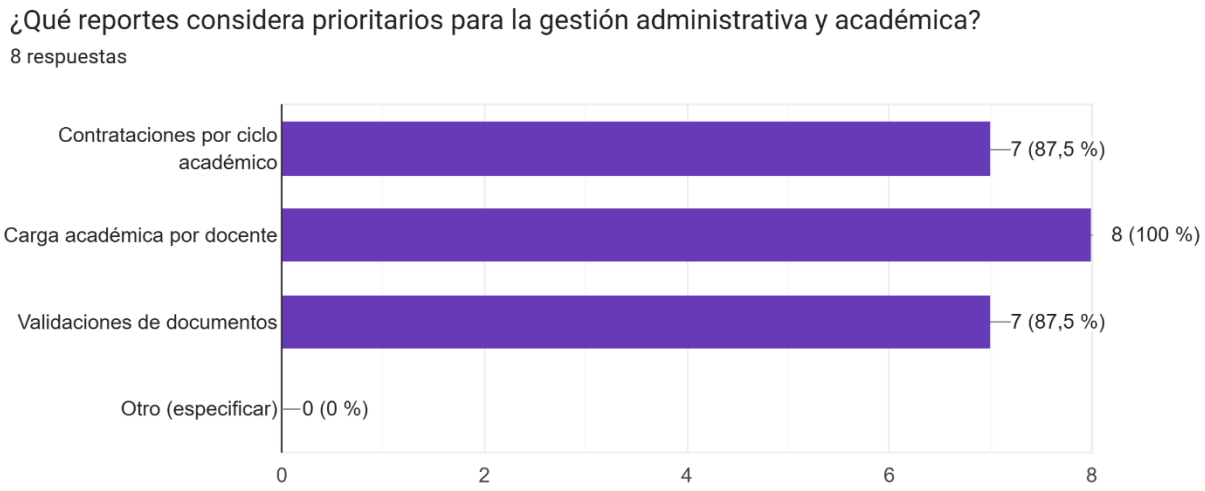
- Notificaciones internas dentro del sistema
- Notificaciones vía correo electrónico
- Registro de historial de acciones (bitácora consultable)
- Todas las anteriores



(Requerimiento derivado: RF-07 – Notificaciones y trazabilidad)

Pregunta 8. ¿Qué reportes considera prioritarios para la gestión administrativa y académica?

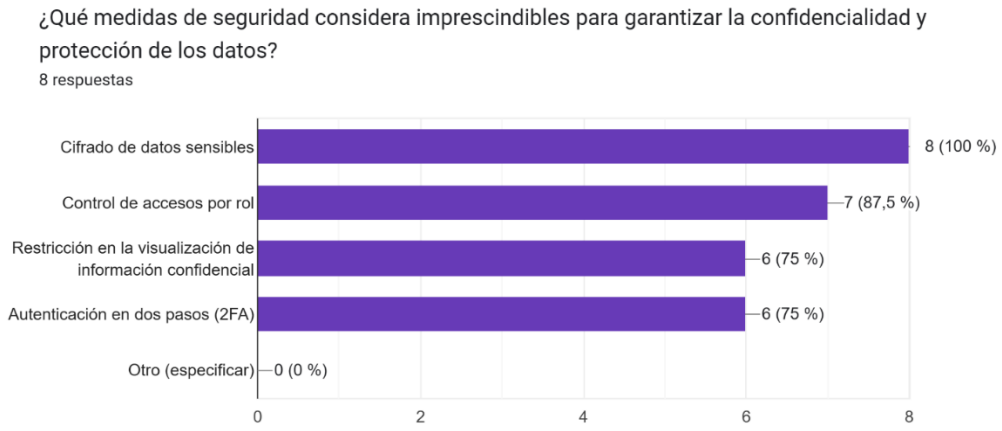
- Contrataciones por ciclo académico
- Carga académica por docente
- Validaciones de documentos
- Otro (especificar)



(Requerimiento derivado: RF-08 – Generación de reportes)

Pregunta 9. ¿Qué medidas de seguridad considera imprescindibles para garantizar la confidencialidad y protección de los datos?

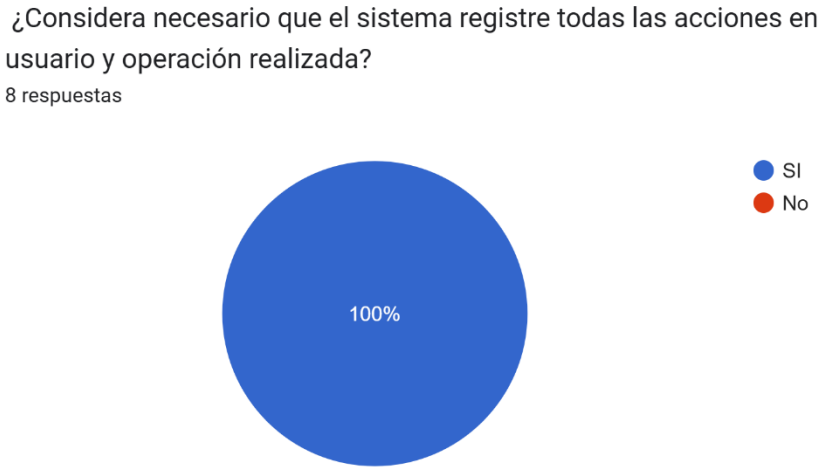
- Cifrado de datos sensibles
- Control de accesos por rol
- Restricción en la visualización de información confidencial
- Autenticación en dos pasos (2FA)
- Otro (especificar)



(Requerimiento derivado: RF-09 – Seguridad de datos)

Pregunta 10. ¿Considera necesario que el sistema registre todas las acciones en una bitácora con fecha, hora, usuario y operación realizada?

- Sí
- No



(Requerimiento derivado: RF-10 – Auditoría interna)

Anexo 2 – Tabla de Casos de Pruebas.

ID_CASO	Nombre del Caso de Prueba
CP_01	Login con credenciales correctas
CP_02	Creación de usuario en el sistema
CP_03	Validación de formulario de creación de usuario con datos vacío o inválidos
CP_04	Validar roles diferentes a Administrador y Super Administrador no puedan crear usuarios
CP_05	Validación de listado de usuarios creados en el sistema
CP_06	Cambiar contraseña a usuario existente desde módulo Usuarios
CP_07	Listado de bitácora de trazabilidad de acciones de usuarios
CP_08	Filtrar bitácora de acciones del usuario por usuario
CP_09	Filtrar bitácora de acciones del usuario por usuario inexistente
CP_10	Filtrar bitácora de acciones del usuario con una relevancia "Alta".
CP_11	Acceso a "Bitácora" con rol no autorizado
CP_12	Rendimiento de carga inicial en el módulo de "Bitácora".
CP_13	Validar el nombre del ciclo para la creación de un ciclo academico
CP_14	Validar la congruencia de las fechas en la creación de ciclo
CP_15	Validar unicidad de ciclo (nombre y periodo)

ID_CASO	Nombre del Caso de Prueba
CP_16	Estado del ciclo académico por defecto
CP_17	Listar los ciclos académicos (paginado con filtros requeridos).
CP_18	Editar ciclo creado (campos requeridos)
CP_19	Editar ciclo enviando datos faltantes.
CP_20	Eliminar ciclo académico sin dependencias
CP_21	Permisos de lectura para rol diferente de administrador a ciclos académicos
CP_22	intento de creación de ciclo sin permisos
CP_23	Listar bancos registrados.
CP_24	Crear registro de banco válido.
CP_25	Validación de obligatoriedad de name de banco
CP_26	Actualizar nombre de banco registrado
CP_27	Eliminar banco sin dependencias
CP_28	Permisos de lectura para rol diferente a administrador en Bancos
CP_29	Intento de creación de bancos sin permiso
CP_30	Eliminar registro de escalafón sin empleados asociados.
CP_31	Eliminar registros de escalafón con empleados asociados
CP_32	Acceso al menú de Escalafón con rol diferente a administrador.
CP_33	Intento de creación de escalafón sin permiso
CP_34	Validación de formulario de creación de escalafón
CP_35	Creación exitosa de escalafón desde formulario.
CP_36	Paginación visual de tabla Escalafón
CP_37	Ordenamiento de tabla por encabezados Escalafón
CP_38	Listar escuelas por facultad (sin paginado)
CP_39	Director activo asignado a escuela.
CP_40	Crear escuela válida por facultad
CP_41	Actualizar escuela con datos correctos.
CP_42	Actualizar escuela con datos faltantes
CP_43	Eliminar escuela sin dependencia
CP_44	Permisos de lectura para rol diferente de administrador en escuelas
CP_45	intento de creación de ciclo sin permisos
CP_46	Validación de formulario de creación de escuela
CP_47	Creación exitosa de escuela en formulario
CP_48	Editar escuela en formulario
CP_49	Editar con información invalida de escuela en formulario.
CP_50	Eliminar escuela con dependencia en menú Escuela
CP_51	Eliminar escuela sin dependencia en menú Escuela
CP_52	Búsqueda por nombre de Escuela en menú de Escuela
CP_53	Paginación visual de tabla en menú de Escuelas.

ID_CASO	Nombre del Caso de Prueba
CP_54	Ordenamiento por columna en tabla de Escuela
CP_55	Validación de StoreHiringRequest
CP_56	Mapeo de estados HiringRequestStatusCode
CP_57	Transición invalida de estado de solicitud
CP_58	Validación de actualización de tipo de contratación (SPNP/TI/TA)
CP_59	Conversión duiToText / nitToText
CP_60	Cálculo de edad (calculaedad)
CP_61	Generación de strings de periodo y horario
CP_62	Plantillas/Formatos Disponbiles
CP_63	Crear Solicitud SPNP
CP_64	Crear Solicitud TI
CP_65	Crear Solicitud TA
CP_66	Validación al crear solicitudes sin campos
CP_67	Actualizar detalle solicitud SPNP
CP_68	Permisos por rol en solicitudes de contratación
CP_69	Seguridad: inyección SQL / XSS
CP_70	Seguridad: exposición de datos sensibles
CP_71	Compatibilidad: navegadores
CP_72	Accesibilidad básica
CP_73	Duplicidad de persona en la misma solicitud
CP_74	Validación de StoreEmployeeRequest (campos obligatorios y formato básico)
CP_75	Normalización de identificadores (DUI/NIT)
CP_76	Unicidad por DUI/NIT
CP_77	Formato de correo y teléfono
CP_78	Validador de adjuntos (tipos/tamaño)
CP_79	Enmascaramiento de datos sensibles en Serializer
CP_80	Crear persona con datos correctos
CP_81	Crear persona con rol NO autorizado
CP_82	Crear persona con datos faltantes o incorrectos
CP_83	Listar y buscar candidatos por nombre
CP_84	Subir documento sin autenticación o token inválido
CP_85	Subir documento de persona en formato no permitido
CP_86	Crear empleado duplicado
CP_87	Privacidad en la exposición de datos en listados de empleados
CP_88	Búsqueda y filtro por nombre
CP_89	Crear información de empleado
CP_90	Editar datos de empleado y verificar bitácora
CP_91	Validación de estados de documentos

ID_CASO	Nombre del Caso de Prueba
CP_92	Conjunto de documentos requeridos por tipo de candidato
CP_93	Listar documentos por estado y candidato
CP_94	Aprobar documento de candidato
CP_95	Validación de permisos sobre verificación de documentos
CP_96	Ver validaciones de documentos por candidato
CP_97	Aprobar documento de candidato
CP_98	Rechazar documento con observación obligatoria
CP_99	Solicitar reenvío/corrección
CP_100	Hash y verificación de contraseña
CP_101	Normalización y validación de email
CP_102	Generación y validación de token de acceso y expiración/revocación
CP_103	Contador de intentos fallidos y lockout (intentos incorrectos de inicio de sesión)
CP_104	Generación de contraseña autogenerada al crear usuario
CP_105	Bloqueo por intentos fallidos de iniciar sesión API
CP_106	Cambio de contraseña de usuario
CP_107	Reglas de creación de usuario
CP_108	Actualización de rol (cambios de rol y datos obligatorios)
CP_109	Crear usuario API
CP_110	Crear usuario con correo duplicado
CP_111	Listar usuarios
CP_112	Actualizar usuario
CP_113	Login válido
CP_114	Login inválido con credenciales incorrectas
CP_115	Crear facultad (API) con permisos
CP_116	Eliminar facultad sin permisos (API)
CP_117	Editar facultad (API) con permisos
CP_118	Listar facultades (API) con permisos
CP_119	Crear facultad duplicada (API)
CP_120	Listar actividades (API) con permisos
CP_121	Crear actividad duplicada (API)
CP_122	Editar actividad (API) con permisos
CP_123	Listado de cargos
CP_124	Ver detalles de cargo
CP_125	Acceso restringido a Cargos para rol no Administrador
CP_126	Listar tipos de grupos registrados.
CP_127	Eliminar un tipo de grupo existente con permisos
CP_128	Eliminar un tipo de grupo sin permisos
CP_129	Crear carga académica válida
CP_130	Crear carga académica sin permisos

ID_CASO	Nombre del Caso de Prueba
CP_131	Listar todas las cargas académicas
CP_132	Consultar carga por ID válido
CP_133	Consultar carga por ID inexistente
CP_134	Consultar cargas por escuela
CP_135	Registrar grupo en carga académica
CP_136	Asignar profesor a grupo
CP_137	Listar todos los formatos registrados
CP_138	Consultar formato por ID válido
CP_139	Consultar formato por ID inexistente
CP_140	Descargar plantilla de materias en Excel
CP_141	Intentar crear formato sin permisos (candidato)
CP_142	Acceso no autorizado a datos de otro usuario
CP_143	Acceso no autorizado a editar recurso ajeno
CP_144	XSS almacenado en campos de texto
CP_145	Inyección en filtros
CP_146	CORS restrictivo
CP_147	Cabeceras de seguridad
CP_148	Tiempo de respuesta en listados principales
CP_149	Evitar consultas excesivas en listados con relaciones
CP_150	Inicio de sesión con muchos usuarios simultáneos
CP_151	Velocidad de procesamiento de correos y notificaciones
CP_152	Verificación de transporte seguro de credenciales
CP_153	Intento de inicio de sesión con credenciales predeterminadas, débiles o en blanco
CP_154	Detección de enumeración de cuentas
CP_155	Validar seguridad de “Recordarme” y persistencia de sesión
CP_156	Consistencia visual y flujo de navegación principal
CP_157	Diseño responsivo y adaptativo
CP_158	Formularios: validaciones y mensajes
CP_159	Resiliencia ante fallos de red
CP_160	Manejo de los mensajes de error

Anexo 3 - Normas y Herramientas Utilizadas

Normas y Estándares:

- **ISO/IEC 9126**: Norma de calidad de software basada en características internas y externas.
- **ISO/IEC 12207**: Norma para procesos del ciclo de vida del software.
- **ISO/IEC 15504 (SPICE)**: Norma para evaluación y mejora de procesos de software.
- **ISO/IEC 25010**: Modelo de calidad del producto de software.
- **ISO/IEC/IEEE 29119**: Norma internacional para pruebas de software.
- **ISO 9001**: Norma para Sistemas de Gestión de la Calidad.

Herramientas de Pruebas Seleccionadas.

- **Selenium WebDriver:** Framework para pruebas funcionales y de interfaz gráfica.
- **Postman:** Herramienta para pruebas de APIs REST.
- **Apache JMeter:** Herramienta para pruebas de rendimiento y carga.
- **OWASP ZAP:** Herramienta para pruebas de seguridad en aplicaciones web.
- **SonarQube:** Plataforma para análisis estático de código.

Herramientas de Soporte y Desarrollo.

- **Composer:** Gestor de dependencias para PHP/Laravel.
- **npm (Node Package Manager):** Gestor de paquetes para JavaScript/TypeScript en el frontend.
- **Docker:** Despliegue de entornos portables.
- **Azure DevOps:** Plataforma de integración y despliegue continuo (CI/CD), gestión de proyectos y pruebas.
- **GitHub:** Plataforma de control de versiones y repositorios Git.

Tecnologías del Sistema

- **Laravel 10 (PHP 8.1):** Framework backend para la API REST**.**
- **React 17 + TypeScript:** Framework frontend para aplicaciones SPA.
- **Ant Design 4:** Librería de componentes UI para React.
- **React Router DOM v5:** Librería para enrutamiento en React**.**
- **React Query v3:** Manejo de estados y datos en el frontend.
- **Axios:** Cliente HTTP para consumo de APIs**.**
- **CASL:** Librería de control de permisos en la interfaz de usuario.
- **Laravel Sanctum:** Autenticación y gestión de tokens en Laravel.
- **Spatie Roles & Permissions:** Paquete de gestión de roles y permisos en Laravel.
- **PostgreSQL 16 (Amazon RDS):** Motor de base de datos relacional.
- **Laravel Horizon:** Monitor de colas en Laravel**.**
- **Maatwebsite Laravel Excel:** Generación de reportes Excel.

- **Laravel dompdf:** Generación de reportes en PDF.
- **PhpOffice/PhpWord:** Generación de reportes en Word.
- **Laravel Mailer:** Gestión de notificaciones por correo electrónico.
- **Laravel Scheduler (Artisan + Cron Jobs):** Automatización de tareas programadas.
- **NGINX:** Servidor web para despliegue de frontend y backend.