

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE SOFTWARE

Tema:

SISTEMA DE GESTIÓN DE HORARIOS Y AULAS DE LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL MEDIANTE ALGORITMOS DE OPTIMIZACIÓN

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Software

ÁREA: Desarrollo de Software

LÍNEA DE INVESTIGACIÓN: Tecnologías de la información y Sistemas de Control

AUTOR: Christian Alejandro Chico López

TUTOR: Ing. Santiago David Jara Moya, MSc.

Ambato - Ecuador

julio – 2025

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE GESTIÓN DE HORARIOS Y AULAS DE LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL MEDIANTE ALGORITMOS DE OPTIMIZACIÓN desarrollado bajo la modalidad Proyecto de Investigación por el señor Christian Alejandro Chico López, estudiante de la Carrera de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, julio 2025.

Firmado electrónicamente por:
SANTIAGO DAVID JARA MOYA
Validar únicamente con FirmaEC

Ing. Santiago David Jara Moya, MSc.

TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE GESTIÓN DE HORARIOS Y AULAS DE LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL MEDIANTE ALGORITMOS DE OPTIMIZACIÓN es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, julio 2025.



Christian Alejandro Chico López

C.C. 1804309837

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, julio 2025.



Christian Alejandro Chico López

C.C. 1804309837

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Christian Alejandro Chico López, estudiante de la Carrera de Software, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE GESTIÓN DE HORARIOS Y AULAS DE LA FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL MEDIANTE ALGORITMOS DE OPTIMIZACIÓN, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, julio 2025



Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL



Ing. Hernan Naranjo Avalos, Mg.
PROFESOR CALIFICADOR



Ing. José Caiza Caizabuano, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

Dedico este logro a mis padres, quienes han sido un pilar fundamental a lo largo de mi carrera. Sin su apoyo, este camino habría sido mucho más difícil de recorrer. También me lo dedico a mí mismo, por la constancia, el esfuerzo y la dedicación que me han permitido alcanzar este objetivo.

AGRADECIMIENTO

Agradezco profundamente a mis padres por todo el apoyo incondicional que me han brindado a lo largo de estos años, así como por el esfuerzo y sacrificio que han realizado para ayudarme a alcanzar mis metas.

Agradezco también a mi tutor de tesis, por su guía, paciencia y compromiso durante el desarrollo de este trabajo.

Extiendo también mi gratitud a todas las personas que me acompañaron y aportaron en el desarrollo de mi carrera, siendo parte esencial de este camino.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA ..... **i**

APROBACIÓN DEL TUTOR ..... **ii**

AUTORÍA ..... **iii**

DERECHOS DE AUTOR ..... **iv**

APROBACIÓN DEL TRIBUNAL DE GRADO ..... **v**

DEDICATORIA ..... **vi**

AGRADECIMIENTO ..... **vii**

ÍNDICE GENERAL DE CONTENIDOS ..... **viii**

ÍNDICE DE TABLAS ..... **xii**

ÍNDICE DE FIGURAS ..... **xiii**

ÍNDICE DE ANEXOS ..... **xv**

RESUMEN EJECUTIVO ..... **xvi**

ABSTRACT ..... **xvii**

CAPÍTULO I. MARCO TEÓRICO **1**

1.1 Tema de investigación 1

1.1.1 Planteamiento del problema 1

1.2 Antecedentes investigativos 2

1.3 Fundamentación teórica 18

1.3.1 Herramientas Tecnológicas 19

1.3.2 Aplicación Web 19

1.3.3 Backend 20

1.3.4 Frontend 21

1.3.5 Base de Datos 21

1.3.6 Framework 22

1.3.7 API Rest 24

1.3.8 ORM (Object-Relational Mapping) 24

1.3.9 Control de Versiones 25

1.3.10 Control de Entornos 26

1.3.11 Servidor 26

1.3.12 Lenguaje de Programación 27

1.3.13 Encriptación 28

1.3.14 Metodología Ágil 30

1.3.15 Algoritmos de Optimización 31

1.3.16 Restricciones en la Optimización de Horarios 33

1.4 Objetivos 33

1.4.1 Objetivo general 33

1.4.2 Objetivos específicos 33

CAPÍTULO II. METODOLOGÍA 35

2.1 Materiales 35

2.2 Métodos 35

2.2.1 Modalidad de la investigación 37

2.2.2 Población y muestra 38

2.2.3 Recolección de información 39

2.2.4 Procesamiento y análisis de datos 42

CAPÍTULO III. RESULTADOS Y DISCUSIÓN 43

3.1 Análisis del proceso actual 43

3.1.1 Identificador de factores necesarios 43

3.1.2 Proceso actual 44

3.1.3 Ineficiencias en el proceso actual 48

3.1.4 Oportunidades de optimización 48

3.2 Análisis de algoritmos de optimización 49

3.2.1 Análisis y selección de algoritmo 49

3.2.2 Comparativa de algoritmos 51

3.3 Desarrollo de la Propuesta 55

3.3.1 Metodología Ágil 55

3.3.2 Selección de Herramientas 59

3.3.3 Fase 1: Planificación de requerimientos 65

3.3.4 Fase 2: Diseño de usuario 67

3.3.5 Fase 3: Construcción 84

3.3.6 Fase 4: Implementación 97

3.4 Comparativa de Tiempos: Manualmente vs IA 106

3.4.1 Objetivo 106

3.4.2 Análisis 107

3.4.3 Resultados 108

3.5 Comparativa de Horarios Generados: Manualmente vs IA 109

3.5.1 Análisis mediante ToT(Time on Task) 109

3.5.2 Resultados Comparativos 110

3.5.3 Reducción de Conflictos y Reasignaciones 111

3.5.4 Distribución Horaria y Balance de Carga 111

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES 113

4.1 Conclusiones 113

4.2 Recomendaciones 114

REFERENCIAS BIBLIOGRÁFICAS 116

ANEXOS 121

ÍNDICE DE TABLAS

Tabla 1. Tabla de algoritmos y cantidades 14

Tabla 2: Tabla de restricciones y cantidades 15

Tabla 3: Tabla de restricciones obligatorias y cantidades 17

Tabla 4. Escala de Likert para evaluación de preguntas 39

Tabla 5. Promedio de los criterios evaluados en la encuesta de Likert 40

Tabla 6. Comparativa de algoritmos 52

Tabla 7. Comparación de Metodologías Ágiles 58

Tabla 8. Análisis entre React, Next.js, Astro, Angular 59

Tabla 9. Análisis entre Django, Nest.js, Express.js, Spring Boot 61

Tabla 10. Análisis entre MySQL, PostgreSQL, MongoDB 62

Tabla 11. Análisis entre AES, RSA 63

Tabla 12. Análisis entre Git, Subversion y Mercurial 64

Tabla 13. Planificación de requerimientos del sistema por iteraciones 66

Tabla 14. Pruebas de aceptación – Iteración 1 98

Tabla 15. Pruebas de aceptación – Iteración 2 99

Tabla 16. Pruebas de aceptación – Iteración 3 99

Tabla 17: Entrevista sobre la gestión de horarios 124

Tabla 18: Entrevista sobre el proceso de gestión de horarios 129

Tabla 19: Entrevista sobre el proceso de gestión de horarios 135

ÍNDICE DE FIGURAS

Figura 1: Cantidad de diferentes algoritmos utilizados en investigaciones 14

Figura 2: Cantidad de restricciones en las investigaciones 16

Figura 3: Cantidad de restricciones obligatorias que tenían que cumplir para solucionar los problemas en las investigaciones 18

Figura 4. Esquema de la Metodología 36

Figura 5. Nube de palabras de las entrevistas aplicadas al personal de laboratorio ... 41

Figura 6. Diagrama de flujo del proceso actual 47

Figura 7. Diagrama de flujo del proceso de creación de horarios propuesto 70

Figura 8. Diagrama de flujo del proceso propuesto de generación de hojas de control 73

Figura 9. Diagrama de flujo del proceso propuesto de reservar aulas o laboratorios 75

Figura 10: Arquitectura de la aplicación 76

Figura 11. Prototipo de inicio de sesión 77

Figura 12. Prototipo de panel de navegación 78

Figura 13. Prototipo de panel de control 79

Figura 14. Prototipo de consulta de horarios por docentes 80

Figura 15. Prototipo de formulario para la generación de hojas de control individual 80

Figura 16. Prototipo de formulario para la generación de hojas de control por bloque 81

Figura 17. Prototipo de interfaz para la gestión de datos almacenados 82

Figura 18. Prototipo de formulario de registro de datos 83

Figura 19. Prototipo de reserva de aulas y laboratorios 83

Figura 20. Prototipo de generación de horarios 84

Figura 21. Diseño Conceptual de la Base de Datos 86

Figura 22. Arquitectura proyecto Backend 87

Figura 23. Creación entidades con TypeORM 88

Figura 24. Uso de JWT 94

Figura 25. Arquitectura proyecto Frontend 95

Figura 26. Interfaz configuración inicial de Docker 103

Figura 27. Confirmación de configuración inicial 104

ÍNDICE DE ANEXOS

Anexo A. Cuestionario de la entrevista sobre la comparación de proceso manual de creación de horarios y creación automatizada..... 121

Anexo B. Entrevista sobre el proceso de gestión de horarios..... 123

Anexo C. Entrevista sobre el proceso de gestión de horarios..... 128

Anexo D. Entrevista sobre el proceso de gestión de horarios..... 134

Anexo E. Entrevista de levantamiento de requerimientos..... 138

Anexo F. Levantamiento de requerimientos – IEEE 830..... 147

Anexo G. Diagrama de Base de Datos..... 161

Anexo H. Manual de Usuario..... 165

RESUMEN EJECUTIVO

La planificación manual de horarios académicos y la asignación de aulas representa un desafío operativo en instituciones educativas debido a la cantidad de restricciones a considerar, como la disponibilidad docente, la capacidad de las aulas y los requerimientos institucionales. En la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI) de la Universidad Técnica de Ambato, este proceso demandaba entre 7 y 10 días laborales, además de la utilización de múltiples herramientas y una alta carga administrativa.

Con el objetivo de optimizar dicho proceso, se desarrolló una aplicación web que automatiza la creación de horarios considerando las restricciones reales del entorno académico. La solución se implementó mediante la metodología RAD, utilizando tecnologías modernas como Next.js, Nest.js y MySQL, junto con un motor de optimización desarrollado en Python, que integra algoritmos genéticos y programación lineal.

Los resultados obtenidos evidencian una reducción del 66.67 % en el tiempo de planificación, una mejora del 27 % en la ocupación de espacios académicos y la eliminación de conflictos de asignación. La aplicación permite generar horarios en minutos, gestionar restricciones, crear hojas de control, reservar espacios y visualizar datos académicos en una sola plataforma.

Este sistema demuestra cómo el uso de algoritmos de optimización puede transformar procesos tradicionales, mejorando la eficiencia institucional, reduciendo errores humanos y facilitando una gestión más precisa, ágil y sostenible. La propuesta representa una solución escalable y alineada con las necesidades de planificación de horarios en entornos educativos complejos.

Palabras clave: Algoritmos, optimización, gestión, horario, aulas, inteligencia artificial

ABSTRACT

Manual planning of academic schedules and classroom allocation represents an operational challenge in educational institutions due to the number of restrictions to consider, such as teacher availability, classroom capacity, and institutional requirements. At the Faculty of Systems, Electronics and Industrial Engineering (FISEI) of the Technical University of Ambato, this process used to take between 7 and 10 working days, in addition to the use of multiple tools and a high administrative workload.

With the aim of optimising the aforementioned process, a web application was developed that automates the creation of timetables, taking into account the real constraints of the academic environment. The solution was implemented using the RAD methodology, employing modern technologies such as Next.js, Nest.js and MySQL, together with an optimisation engine developed in Python, which integrates genetic algorithms and linear programming.

The results obtained show a 66.67% reduction in planning time, a 27% improvement in the occupancy of academic spaces, and the elimination of allocation conflicts. The application allows schedules to be generated in minutes, restrictions to be managed, control sheets to be created, spaces to be reserved, and academic data to be viewed on a single platform.

This system demonstrates how the use of optimisation algorithms can transform traditional processes, improving institutional efficiency, reducing human error, and facilitating more accurate, agile, and sustainable management. The proposal represents a scalable solution aligned with the needs of schedule planning in complex educational environments.

Keywords: Algorithms, optimization, management, timetable, classrooms, artificial intelligence

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

Sistema de Gestión de Horarios y Aulas de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial mediante Algoritmos de Optimización.

1.1.1 Planteamiento del problema

La asignación de aulas durante la creación y planificación de horarios representa una tarea crítica para muchas instituciones educativas. El proceso manual puede volverse complejo y repetitivo, ya que debe ajustarse a la disponibilidad de espacios y docentes, considerando diversas restricciones como el uso óptimo de las aulas, los requerimientos del personal académico y las características específicas de los laboratorios. Esto convierte la planificación en un desafío considerable para el personal responsable. Ante esta situación, la incorporación de tecnologías avanzadas, como los algoritmos de optimización, se ha vuelto indispensable para mejorar la distribución de los recursos académicos y reducir el tiempo y esfuerzo requeridos en dicha labor [1][2].

En la Universidad Técnica de Ambato, particularmente en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, este proceso se realiza de forma manual, lo que representa una carga operativa significativa. Actualmente, la facultad cuenta con 83 docentes, cada uno con una jornada semanal de 40 horas, de las cuales hasta 30 horas pueden corresponder a clases asignables mediante planificación automática. Esta carga docente debe distribuirse entre 5 carreras, haciendo uso de 31 aulas y 23 laboratorios, lo que implica una alta complejidad en la gestión de espacios y tiempos. La asignación eficiente se ve limitada por la dificultad de coordinar estos recursos, lo cual incrementa el riesgo de solapamientos, espacios subutilizados o conflictos en la disponibilidad de aulas.

Además, el incremento constante de la población estudiantil y las demandas específicas del cuerpo docente han complicado aún más el proceso, reduciendo el tiempo disponible para otras tareas críticas como el mantenimiento de la infraestructura tecnológica, redes internas, equipos informáticos y la actualización del software requerido en los laboratorios. Esta metodología tradicional impacta negativamente la eficiencia en la gestión de recursos, dificultando la optimización del uso de los espacios académicos disponibles [2].

La raíz del problema radica en la ausencia de un sistema automatizado que utilice algoritmos de optimización para asignar aulas y horarios de forma eficiente. Una solución de este tipo debería contemplar tanto restricciones generales como específicas. La falta de una herramienta tecnológica adecuada no solo ralentiza el proceso, sino que incrementa la probabilidad de errores humanos en la distribución de recursos. Implementar una plataforma automatizada permitiría generar horarios de manera precisa y ágil, reduciendo significativamente el tiempo invertido y elevando la calidad de la planificación académica y administrativa [3].

1.2 Antecedentes investigativos

En [9], se utiliza el algoritmo Preference-Based Stepping Ahead Firefly Algorithm (mdFA-Step), una versión modificada del algoritmo de luciérnagas adaptado a problemas discretos. Aunque no se menciona un porcentaje de satisfacción específico, el algoritmo cumple con una serie de restricciones clave. Entre ellas, se destacan las restricciones duras (hard constraints), como evitar que dos exámenes con estudiantes en común se programen simultáneamente. Asimismo, las restricciones flexibles (soft constraints) están enfocadas en la distribución adecuada de los exámenes para minimizar la carga académica de los estudiantes. Si bien no se proporciona un número exacto de restricciones específicas, se consideran aspectos generales como la capacidad de las aulas y los horarios asignados. El principal objetivo de este algoritmo es optimizar la proximidad de los exámenes de un mismo estudiante, reduciendo conflictos de horarios. Finalmente, el algoritmo fue implementado y probado en un

caso real de programación de exámenes en la Universidad de Toronto, validando su aplicación en entornos universitarios.

En [10], se implementa un algoritmo genético (GA) como método de optimización para gestionar el cronograma de trenes. El porcentaje de satisfacción se refleja en la priorización de los pasajeros vulnerables, quienes representan el 24%, frente a los pasajeros generales que conforman el 76% del total. El modelo respeta varias restricciones, como la capacidad de carga, el tamaño de la flota y los tiempos de espera entre trenes. Entre las restricciones obligatorias se incluyen una capacidad mínima de carga y tiempos máximos de espera entre trenes. Las restricciones flexibles abarcan aspectos como el tiempo de espera y el tamaño de la flota, que se pueden ajustar según los coeficientes ponderados del modelo. También se imponen restricciones específicas, como los tiempos de caminata de los pasajeros y los tiempos de operación del tren para pasajeros vulnerables y generales. El objetivo principal de la optimización es reducir los tiempos de espera y el tamaño de la flota, equilibrando los costos operativos y la satisfacción de los pasajeros. Finalmente, este algoritmo se utiliza en el sistema de metro de Beijing, específicamente en las estaciones de transferencia.

En [11], se propone un algoritmo de programación dinámica (Dynamic Programming - DP) para resolver problemas de reprogramación de trenes en los ferrocarriles de alta velocidad. Aunque no se menciona un porcentaje de satisfacción específico, el modelo cumple con varias restricciones importantes, como los tiempos de espera y las capacidades de los trenes y andenes. Las restricciones obligatorias incluyen tiempos mínimos y máximos de llegada y salida, además de restricciones de capacidad para garantizar un flujo seguro y eficiente de pasajeros. Las restricciones flexibles permiten ajustar el número de trenes y los tiempos de espera dentro de un rango predeterminado. También existen restricciones específicas relacionadas con el flujo de pasajeros en las estaciones de transferencia y la capacidad de los trenes. El principal objetivo del algoritmo es optimizar la reprogramación de los horarios de trenes para reducir los tiempos de retraso y mejorar la eficiencia operativa del sistema ferroviario. Este enfoque se utiliza en la gestión del tráfico ferroviario en los ferrocarriles de alta velocidad de China.

En [12], se emplea el algoritmo de optimización de presa-cazador (Hunter–Prey Optimization - HPO) para ajustar los parámetros de una red neuronal convolucional de múltiples cabezas inducida por atención (AIMCNN). Aunque no se menciona un porcentaje de satisfacción específico, el modelo evalúa diversas métricas de desempeño, como el análisis de habilidades de aprendizaje y actividad de los estudiantes. No se detallan restricciones obligatorias, pero el enfoque parece adaptable para optimizar la calidad de la enseñanza en diferentes contextos. Tampoco se mencionan restricciones flexibles explícitas, pero el sistema es flexible en su implementación. El objetivo principal es optimizar la calidad de la enseñanza de piano a distancia, analizando el rendimiento y mejorando la retroalimentación para los estudiantes. Este enfoque se aplica en sistemas de enseñanza remota de piano, particularmente en un entorno de computación en la nube.

En [13], se emplea un algoritmo de programación dinámica como base para resolver problemas de asignación de aulas en la Facultad de Ingeniería Eléctrica y Electrónica. Los resultados de la encuesta muestran un nivel de satisfacción completo, lo que refleja una aceptación total del algoritmo por parte de los usuarios. El sistema cumple con varias restricciones, como la hora, día, capacidad, edificio y piso, aunque no se especifica el número exacto de restricciones. Las restricciones obligatorias incluyen estos mismos parámetros esenciales para garantizar una asignación adecuada de aulas. Aunque no se mencionan explícitamente, las restricciones flexibles pueden incluir las preferencias del usuario, porque el sistema ofrece recomendaciones basadas en criterios personalizados. El objetivo del algoritmo es optimizar el proceso de asignación de aulas, reduciendo errores y mejorando la eficiencia en la búsqueda de espacios disponibles. Este sistema es utilizado en la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional para la gestión de horarios y asignación de aulas.

En [1], se utiliza un algoritmo genético para generar horarios escolares de manera eficiente. Aunque no se menciona un porcentaje de satisfacción específico, se indica que el algoritmo cumple con la mayoría de las restricciones impuestas, lo que sugiere un alto grado de eficiencia. El algoritmo considera restricciones esenciales y no esenciales, cumpliendo con varias en ambos grupos. Entre las restricciones obligatorias se incluyen la capacidad del aula, la no duplicidad de uso de aulas, y que

un docente no imparta clase a más de un grupo simultáneamente. Las restricciones flexibles incluyen la organización de clases desde las 7:00, horarios corridos y la distribución homogénea de horas durante los días de la semana. El objetivo de este algoritmo es optimizar la generación de horarios, asignando aulas y docentes de manera eficiente y maximizando el uso de recursos. Este sistema es utilizado en la Escuela Superior de Tizayuca para el programa de Licenciatura en Tecnologías de la Información.

En [14], se utiliza el algoritmo de optimización utilizado es el Battle Royal Optimization (BRO), inspirado en el comportamiento social de "Battle Royale", el cual equilibra la exploración y explotación para obtener soluciones óptimas globales. Este algoritmo ha demostrado ser efectivo, con mejoras de precisión del 15.26%, 20.42%, y 21.27%, y tasas de precisión superiores en un 15.61%, 17.50%, y 20.24% respecto a métodos previos. Aunque no se menciona un número exacto de restricciones, el BRO está diseñado para clasificar y ajustar parámetros dentro de complejos conjuntos de datos educativos. El modelo no especifica claramente las restricciones obligatorias o flexibles, pero optimiza la precisión y eficiencia en los resultados obtenidos. La optimización principal radica en mejorar el proceso de aprendizaje basado en proyectos (PBL) para la enseñanza del chino, utilizando una red neuronal de convergencia rápida tolerante a interferencias (ITFCZNN). Este enfoque optimiza tanto la precisión como los parámetros de la red neuronal empleada. En última instancia, el BRO se aplica en contextos educativos, mejorando la enseñanza del idioma chino para estudiantes no nativos a través del PBL.

En [15], el algoritmo de optimización utilizado incluye tres enfoques de agrupamiento de estudiantes: K-means clustering para el agrupamiento dentro de un módulo, Curriculum-wide grouping para el agrupamiento a lo largo de todo el currículo, y el Reciprocal pairing que empareja estudiantes en base a conocimientos complementarios. Aunque no se menciona un porcentaje específico de satisfacción, los algoritmos se evalúan mediante métricas en un entorno simulado. No se especifica cuántas restricciones exactas se cumplen, ya que los algoritmos están diseñados para optimizar el agrupamiento en función de las habilidades y el progreso de los estudiantes. No se mencionan restricciones obligatorias ni flexibles en el documento. La optimización principal radica en la creación de grupos homogéneos o heterogéneos

para facilitar la enseñanza diferenciada y el aprendizaje colaborativo, ayudando a los estudiantes a avanzar en áreas específicas del currículo. Este enfoque se aplica en el sistema de tutoría adaptativa ALEKS, utilizado principalmente en clases de álgebra de nivel secundario.

En [16], el algoritmo de optimización utilizado se basa en algoritmos metaheurísticos, combinando enfoques como algoritmos genéticos, optimización por enjambre de partículas (PSO) y búsqueda tabú (TS), que se combinan para mejorar el rendimiento en la creación de horarios universitarios. No se menciona un porcentaje explícito de satisfacción, pero los resultados optimizados se enfocan en cumplir el mayor número de restricciones y minimizar los conflictos dentro de los horarios generados. Los algoritmos están diseñados para cumplir con las restricciones duras, como evitar conflictos de horario entre materias y asegurar la disponibilidad de recursos, mientras que las restricciones flexibles incluyen la preferencia por periodos no consecutivos para un mismo estudiante o la asignación en ciertos espacios. Se optimizan los horarios universitarios, asignando cursos o exámenes a intervalos de tiempo adecuados, mejorando la eficiencia y evitando conflictos de recursos. Este enfoque se utiliza principalmente en el timetabling universitario, tanto en la planificación de clases como de exámenes.

El algoritmo de optimización utilizado en [17] es un algoritmo genético, inspirado en la teoría de la evolución de Charles Darwin, empleando conceptos como reproducción, mutación y supervivencia para encontrar soluciones óptimas en la asignación de horarios académicos. Aunque no se menciona un porcentaje explícito de satisfacción, el enfoque busca minimizar las violaciones de restricciones y generar soluciones factibles que mejoren la calidad de los horarios. El algoritmo cumple con todas las restricciones duras, como evitar la asignación de un profesor a más de una clase simultáneamente y garantizar que los cursos del mismo semestre no se superpongan. Las restricciones flexibles incluyen preferencias de horarios, como evitar clases en horarios protegidos, entre las 12:00 y 14:10 horas. El objetivo principal es optimizar la asignación de horarios académicos para la carrera de Ingeniería Civil en Computación, minimizando conflictos de planificación y solapamientos de asignaturas. Este enfoque se implementa en la Escuela de Ingeniería Civil en

Computación de la Universidad de Talca, ayudando a evitar conflictos en la disponibilidad de profesores y aulas.

El algoritmo de optimización utilizado en [18] combinas heurísticas constructivas con un algoritmo de búsqueda local. La heurística constructiva genera una programación inicial de los exámenes, mientras que el algoritmo de búsqueda local optimiza la solución al minimizar la distancia entre las aulas asignadas a un mismo examen. Aunque no se menciona un porcentaje directo de satisfacción, se considera que la solución optimizada cumple con los requisitos de proximidad de aulas y restricciones, lo que mejora la eficiencia y satisfacción en la programación de exámenes. El sistema cumple con las restricciones duras, como evitar solapamientos de exámenes para estudiantes que compartan asignaturas y asignar aulas con capacidad adecuada para los matriculados. Las restricciones blandas incluyen minimizar la distancia entre las aulas asignadas para facilitar la supervisión y el desplazamiento de los estudiantes. La optimización se enfoca en la asignación de aulas y horarios para exámenes, priorizando la proximidad y evitando solapamientos entre exámenes de asignaturas comunes. Este sistema se implementa en la Escuela Técnica Superior de Ingeniería Industrial (ETSII) de la Universitat Politècnica de València (UPV), específicamente para la programación de exámenes.

El algoritmo de optimización utilizado en [19] combina algoritmos genéticos (GA) y Optimización por Enjambre de Partículas (PSO), ambos mencionados como métodos efectivos para resolver problemas complejos en la asignación de horarios universitarios. No se proporciona un porcentaje explícito de satisfacción, pero se sugiere que estos algoritmos mejoran la eficiencia en la generación de horarios y minimizan los conflictos. El algoritmo cumple con las restricciones duras, asegurando que no haya conflictos, como clases asignadas a la misma aula o un profesor impartiendo más de una clase al mismo tiempo. Las restricciones obligatorias incluyen la correcta asignación de aulas sin sobrecargar su capacidad y garantizar que un profesor no esté asignado a más de una clase simultáneamente. Las restricciones flexibles consideran la distribución equitativa de horarios para los profesores y la preferencia por ciertos bloques de tiempo según las solicitudes individuales. Se optimiza la asignación de horarios universitarios, enfocándose en evitar conflictos de recursos y maximizando su uso eficiente. Estos algoritmos se aplican principalmente

en el contexto de la asignación de horarios universitarios en instituciones educativas para garantizar la correcta programación de cursos.

En [20], se utiliza un algoritmo metaheurístico de Recocido Simulado (Simulated Annealing, SA) para optimizar la planificación de cursos universitarios. El enfoque busca asignar eventos, como clases, a salas y horarios de manera eficiente, minimizando la violación de restricciones. Las restricciones se dividen en dos categorías: restricciones obligatorias (duras), cuyo incumplimiento genera soluciones inviables, y restricciones opcionales (suaves), que afectan la calidad del horario, pero no su viabilidad. Entre las restricciones obligatorias se incluye que ningún estudiante puede estar asignado a más de una clase al mismo tiempo, y que las aulas asignadas deben cumplir con las características necesarias. Las restricciones suaves permiten evaluar la calidad de la solución y pueden ser violadas, afectando el costo total de la planificación. Aunque el documento no menciona un porcentaje específico de satisfacción, la calidad de las soluciones se mide en función de la minimización de estas violaciones, impactando indirectamente la satisfacción de los involucrados. Este tipo de algoritmo se aplica principalmente en la planificación de cursos universitarios, un problema recurrente en instituciones académicas.

En [21], emplea algoritmos metaheurísticos, como Simulated Annealing (SA), Hill Climbing, y Great Deluge, para resolver problemas de planificación educativa. El objetivo es optimizar la asignación de exámenes y cursos a horarios y aulas, cumpliendo con diversas restricciones obligatorias y minimizando las violaciones de restricciones suaves. Entre las restricciones duras, se incluye evitar el solapamiento de exámenes y asegurar la correcta asignación de salas. Por otro lado, las restricciones suaves abarcan aspectos como la distribución equitativa de exámenes en días separados y la asignación de horarios preferidos. Aunque no se menciona un porcentaje específico de satisfacción, el éxito del modelo se mide por su capacidad de minimizar las violaciones a estas restricciones. El objetivo es mantener las restricciones obligatorias mientras se reducen las penalizaciones por incumplimiento de restricciones flexibles. Estos algoritmos se utilizan principalmente en la planificación académica de universidades para la programación de exámenes y asignación de cursos.

En [16], aborda la planificación universitaria utilizando una combinación de algoritmos metaheurísticos, como Simulated Annealing (SA), Tabu Search (TS), Ant Colony Optimization (ACO) y Particle Swarm Optimization (PSO), para optimizar la asignación de eventos a aulas y horarios. El rendimiento de las soluciones se evalúa en función de las restricciones flexibles que afectan la calidad del horario, aunque no se menciona un porcentaje específico de satisfacción. El modelo debe cumplir con restricciones obligatorias, tales como evitar la coincidencia de clases que comparten el mismo recurso, y asignar adecuadamente las clases a los profesores. Además, también tiene en cuenta las restricciones suaves, como la preferencia de horarios para profesores y estudiantes, cuyo cumplimiento mejora la calidad final del horario. El objetivo es minimizar las violaciones a las restricciones duras y suaves para lograr una solución equilibrada. Este tipo de algoritmos se aplica principalmente en la planificación de horarios universitarios para la programación tanto de cursos como de exámenes.

En [22], utiliza un algoritmo genético (GA) para resolver el problema de la programación de clases universitarias, considerado un problema NP-HARD. El objetivo principal es optimizar la asignación de clases a profesores, horarios y aulas, teniendo en cuenta tanto restricciones duras como restricciones suaves. Las restricciones obligatorias incluyen la disponibilidad de los profesores y la no superposición de clases, mientras que las restricciones flexibles se centran en las preferencias de horarios de estudiantes y profesores, así como en evitar vacíos largos entre clases. Aunque no se menciona un porcentaje específico de satisfacción, el modelo se basa en maximizar la satisfacción de ambos grupos a partir de cuestionarios. El modelo busca minimizar las violaciones de las restricciones, mejorando así la calidad del horario generado. Este enfoque se aplica principalmente en la planificación de horarios universitarios, optimizando la asignación de cursos en instituciones educativas.

En el presente estudio [23], se utiliza un algoritmo de programación lineal entera (PLE) para optimizar la creación de horarios académicos, abordando la complejidad de asignación de recursos en el contexto educativo. Aunque no se especifica un porcentaje exacto de satisfacción en los resultados, se destaca que el algoritmo cumple con cinco conjuntos de restricciones clave, como la asignación de horas por asignatura y la

evitación de cruces entre horarios de cursos. Las restricciones obligatorias incluyen la correcta asignación de horas diarias y la prevención de conflictos en las cargas horarias de los docentes. No se mencionan restricciones flexibles explícitas, pero se permite ajustar los horarios generados en función de necesidades humanas. El principal objetivo del modelo es optimizar la asignación automática de horarios, maximizando la ocupación de los recursos disponibles y evitando conflictos. Este sistema se implementa específicamente en la Escuela "Lic. Angélica Villón Lindao", logrando una gestión eficiente y automatizada de los horarios académicos. En conclusión, el uso de PLE representa una solución eficaz para la planificación académica, integrando múltiples restricciones en su proceso de optimización.

En el presente trabajo [24] se emplean varios métodos de optimización basados en inteligencia artificial, tales como algoritmos genéticos (AG), recocido simulado (RS) y búsqueda por cardumen de peces (BCP) para resolver el problema de distribución de personal de enfermería. Aunque no se menciona un porcentaje exacto de satisfacción, la evaluación de la aptitud del sistema se realiza en función del cumplimiento de restricciones individuales y grupales, midiendo el error en cada caso. Las restricciones que el sistema debe cumplir están relacionadas con las horas de trabajo, la cantidad de personal necesario, y las normas que rigen los turnos, deteniéndose cuando se encuentra una solución óptima que cumpla con todas las restricciones obligatorias. Las restricciones obligatorias incluyen aspectos clave del reglamento laboral, tales como el número mínimo de horas trabajadas y la correcta distribución de turnos. Además, se contempla la flexibilidad en el diseño del algoritmo, permitiendo adaptaciones según la variación en la cantidad de personal disponible y ajustándose a distintos contextos. El objetivo principal es optimizar la distribución del personal de enfermería, asegurando que se cumplan las normativas establecidas y maximizando la eficiencia en la asignación de los recursos humanos. Este enfoque se aplica en instituciones de salud, particularmente en el contexto de la planificación de horarios en la ciudad de Pereira, logrando una gestión eficiente del personal.

En [25], se implementa un algoritmo genético para la asignación automática de aulas y horarios en la Universidad Politécnica Salesiana. Aunque no se especifica un porcentaje exacto de satisfacción, el documento señala mejoras significativas en cuanto al tiempo de ejecución y la optimización de los recursos. El algoritmo cumple

con diversas restricciones, tales como la cantidad de horas por materia, la disponibilidad activa de aulas, y la asignación de horarios contiguos. Las restricciones obligatorias incluyen que las horas de clase sean consecutivas, que las aulas estén activas y disponibles, y que las materias impartidas en un mismo día se asignen a la misma aula. Aunque no se mencionan restricciones flexibles explícitas, el algoritmo está diseñado para adaptarse a las variaciones en la disponibilidad de aulas y horarios. La optimización se centra en la asignación eficiente de aulas y horarios para diferentes carreras, mejorando la gestión del tiempo de los estudiantes y reduciendo su desplazamiento entre aulas. Este sistema se implementa en el Campus Sur de la Universidad Politécnica Salesiana, en Quito, con el objetivo de automatizar y mejorar la planificación académica.

En la investigación [8], tiene como objetivo analizar la influencia de los algoritmos genéticos en la generación de horarios en una unidad educativa. Los algoritmos genéticos se presentan como una opción atractiva para resolver problemas de alta complejidad matemática, permitiendo abordar dificultades mayores en este estudio. A través de los cruces genéticos, se seleccionan los individuos más aptos, utilizando las operaciones basadas en los datos proporcionados. El proceso de mutación interviene de manera gradual, aportando mayor diversidad al conjunto de posibles soluciones, lo que mejora la eficiencia y efectividad del algoritmo en la generación de horarios.

En la investigación [26], se plantea la optimización de horarios apropiados para el transporte público. Un buen horario debe satisfacer la demanda de los pasajeros manteniendo los costos operativos controlados para el operador. La satisfacción de los pasajeros depende en gran medida de la puntualidad del servicio y los tiempos de espera. Desde la perspectiva del operador, la frecuencia adecuada de los autobuses es esencial, especialmente en horas pico. Un sistema de transporte bien planificado mejora el confort de los pasajeros al reducir los tiempos de espera y aumentar la predictibilidad. La coordinación entre la programación de los vehículos y los horarios es compleja, pero fundamental para alcanzar un equilibrio entre oferta y demanda. Finalmente, un servicio confiable es un factor crucial para que los pasajeros elijan usar el transporte público.

Los algoritmos de optimización utilizados en [27], incluyen metaheurísticas, híbridos de metaheurísticas y hiperheurísticas, destacando el Recocido Simulado (Simulated Annealing) como uno de los más efectivos para resolver problemas de programación de horarios. Aunque no se menciona un porcentaje específico de satisfacción, se indica que estos algoritmos mejoran considerablemente el rendimiento en comparación con los enfoques tradicionales. Los algoritmos están diseñados para cumplir con restricciones duras, tales como evitar la programación de eventos simultáneos y asignar adecuadamente los recursos, como aulas con capacidad suficiente. Asimismo, se minimizan las violaciones de las restricciones suaves, como evitar que los estudiantes tengan eventos consecutivos o en los últimos horarios del día. El objetivo principal es optimizar la asignación de horarios en diferentes dominios, mejorando la distribución de recursos y minimizando conflictos. Estos enfoques son aplicables en áreas como educación, salud, deportes y transporte, donde la programación eficiente es fundamental.

El algoritmo de optimización utilizado en [28] es un algoritmo de optimización multiobjetivo difuso, diseñado para resolver el modelo de programación de horarios de trenes. Este algoritmo optimiza tres funciones clave: la minimización del consumo de energía, el tiempo de espera de los pasajeros y el tiempo de espera de pasajeros en transferencia. Aunque no se menciona un porcentaje de satisfacción directo, los resultados muestran una mejora significativa en la eficiencia energética y la calidad del servicio. El modelo cumple con las restricciones duras, asegurando que los trenes lleguen a tiempo a las estaciones de transferencia y que se mantenga la capacidad de transporte de pasajeros. Las restricciones obligatorias incluyen que los pasajeros lleguen a la estación de transferencia antes de la salida del tren y la correcta asignación de tiempos de marcha y permanencia en las estaciones. Las restricciones flexibles se enfocan en la optimización del tiempo de espera y el aprovechamiento de la energía regenerativa. Se optimizan los horarios del metro y el tren, reduciendo el tiempo de espera de los pasajeros y el consumo energético. Este enfoque se implementa en el sistema de metro y ferrocarril de Chengdu, mejorando la eficiencia del servicio y reduciendo los tiempos de espera en las estaciones de transferencia.

En la optimización de horarios académicos o problemas similares, se pueden emplear diversos algoritmos de optimización que ayudan a encontrar soluciones viables y

eficientes. En la investigación, se han considerado diferentes algoritmos de acuerdo a su efectividad en resolver problemas específicos y su popularidad en la literatura. A continuación, se detallan algunos de los algoritmos más destacados utilizados en la optimización:

Algoritmo Genético: Es el algoritmo más utilizado, con una frecuencia de uso de 10 en los estudios revisados. Este algoritmo es conocido por su capacidad para resolver problemas complejos a través de la simulación de procesos de evolución natural, como la selección, cruce y mutación, permitiendo explorar grandes espacios de soluciones.

Simulated Annealing: Con una frecuencia de 4, este algoritmo se inspira en el proceso de enfriamiento de materiales en metalurgia. Comienza con una solución inicial y realiza cambios aleatorios, aceptando ocasionalmente soluciones peores para evitar quedarse atrapado en óptimos locales. A medida que avanza, reduce gradualmente la probabilidad de aceptar soluciones peores, lo que le permite converger hacia una solución óptima.

Particle Swarm Optimization: Utilizado en 3 ocasiones, este algoritmo está inspirado en el comportamiento de enjambres y busca optimizar problemas ajustando iterativamente las posiciones de las partículas (soluciones). Cada partícula se mueve en el espacio de soluciones guiada tanto por su experiencia personal como por la mejor posición encontrada por el grupo. Esto permite explorar el espacio de soluciones y encontrar de manera efectiva el óptimo global.

Otros Algoritmos Específicos: También se encuentran algoritmos como el K-Means, Algoritmo de Optimización Difusa de Multiobjetivo, Dynamic Programming (DP), Battle Royal Optimization (BRO) y Preference-Based Stepping Ahead Firefly Algorithm, cada uno utilizado en una o dos ocasiones dependiendo de la naturaleza del problema de optimización y sus restricciones.

La distribución de uso de estos algoritmos se puede observar en la Tabla 1 y la Figura 1, donde se muestra que el Algoritmo Genético es el más comúnmente utilizado debido a su versatilidad y efectividad en la optimización de problemas de gran escala. Otros algoritmos, como el Hill Climbing y la Búsqueda Tabú, también son seleccionados por su capacidad para mejorar soluciones iterativamente y evitar óptimos locales.

Tabla 1. Tabla de algoritmos y cantidades

Que algoritmo de optimización utiliza	
Algoritmo	**Cantidad**
Preference-Based Stepping Ahead Firefly Algorithm	1
K-Means	2
Algoritmo genético	10
Algoritmo difuso de optimización multiobjetivo	1
Algoritmo de programación dinámica (DP)	2
Algoritmo de optimización de presa-cazador (HPO)	1
Battle Royal Optimization (BRO)	1
Particle Swarm Optimization	3
Búsqueda Tabú	1
Simulated Annealing	4
Hill Climbing	1
Great Deluge	1
Ant Colony Optimization	1
Algoritmo de programación lineal entera	1
Búsqueda por cardumen de peces	1
Algoritmo de programación dinámica	2
Algoritmo de búsqueda local	1

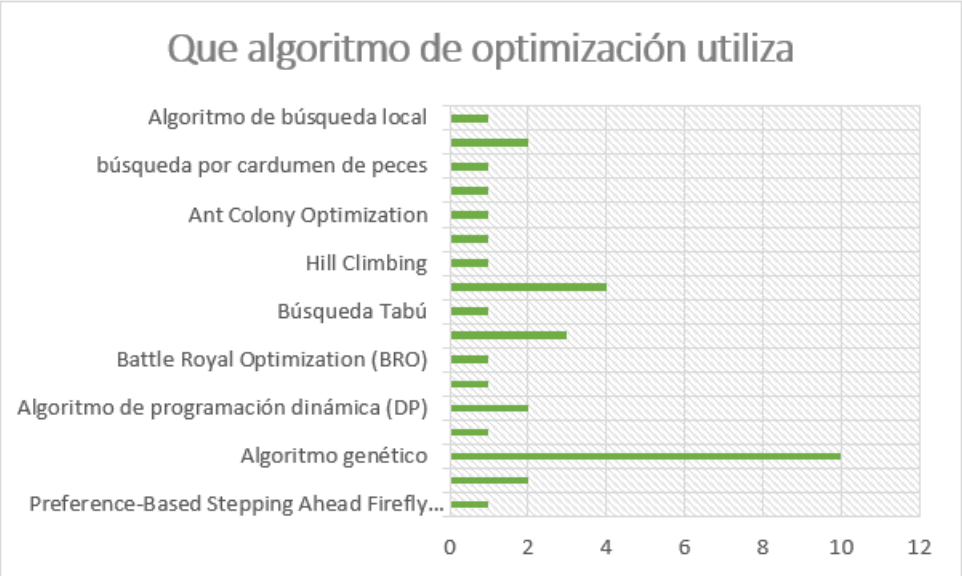


Figura 1: Cantidad de diferentes algoritmos utilizados en investigaciones

Dentro de los algoritmos de optimización, el cumplimiento de restricciones es un aspecto fundamental que influye directamente en la aplicabilidad y eficiencia del algoritmo. Los resultados obtenidos muestran una diversidad en el número de restricciones que los algoritmos deben satisfacer como se observan en la Tabla 2 y Figura 2, destacando la capacidad de adaptación de cada uno a diferentes niveles de complejidad en el problema planteado.

Cantidad de Restricciones

La primera tabla y gráfico indican la cantidad de restricciones que cada algoritmo necesita cumplir para garantizar una solución óptima o viable. Observamos que:

Tabla 2: Tabla de restricciones y cantidades

Número de restricciones encontradas en las investigaciones	
# Restricciones	**Cantidad**
9 Restricciones	2
8 Restricciones	1
7 Restricciones	2
6 Restricciones	4
5 Restricciones	7
4 Restricciones	3
3 Restricciones	0
2 Restricciones	2
1 Restricciones	0
No especificadas	2

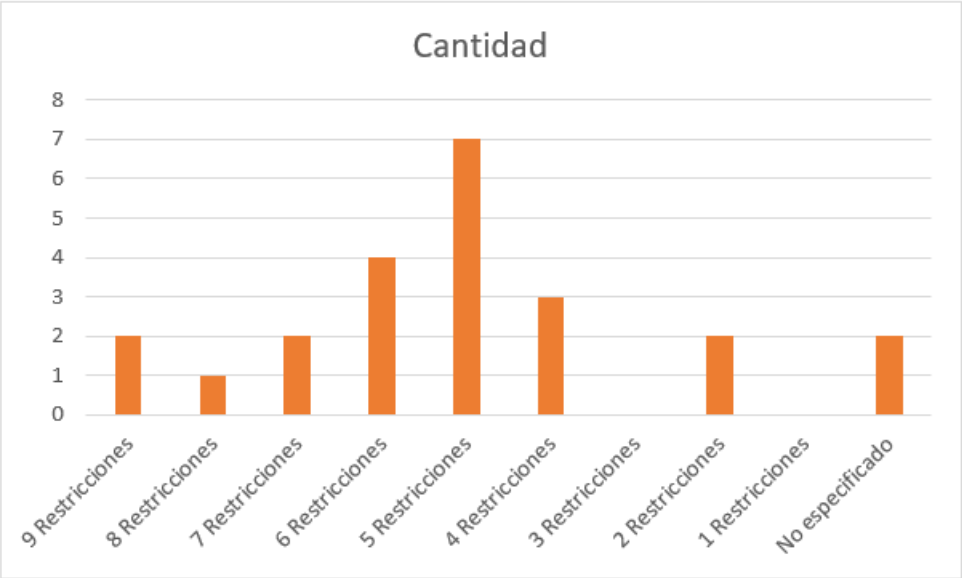


Figura 2: Cantidad de restricciones en las investigaciones

Cuatro restricciones es la cantidad más común, con un total de 7 instancias en las investigaciones revisadas. Esto sugiere que una cantidad moderada de restricciones es común en muchos problemas de optimización. Le siguen las configuraciones con cinco y dos restricciones, que aparecen cuatro veces cada una. Hay algunos casos con una o sin especificar restricciones, lo que indica que algunos algoritmos pueden funcionar con menor cantidad de limitaciones. Este análisis resalta que la mayoría de los problemas de optimización tienden a estar moderadamente restringidos, lo cual puede deberse a la necesidad de un balance entre precisión y complejidad computacional.

Restricciones Obligatorias

En la segunda tabla y gráfico, se desglosa la cantidad de restricciones obligatorias que algunos algoritmos requieren para ser efectivos. Los resultados muestran:

Una preponderancia de algoritmos que requieren dos restricciones obligatorias (9 casos), seguido de aquellos que necesitan cinco restricciones (5 casos).

También se identifican algoritmos que funcionan con solo una o tres restricciones obligatorias, mostrando una flexibilidad en los modelos de optimización para ajustarse a problemas menos estructurados como se muestra en la Tabla 3 y Figura 3.

Estos resultados sugieren que, en términos de diseño, algunos algoritmos están específicamente optimizados para operar bajo ciertas condiciones mínimas de restricción, mientras que otros pueden adaptarse a un entorno más restringido. Esta característica es crucial al seleccionar un algoritmo adecuado para un problema específico, porque permite prever la complejidad y la necesidad de cumplir con ciertas reglas o limitaciones.

Tabla 3: Tabla de restricciones obligatorias y cantidades

Cantidad de restricciones obligatorias	
Restricciones	**Cantidad**
5 Restricciones	5
4 Restricciones	1
3 Restricciones	4
2 Restricciones	9
1 Restricciones	2
No especificado	2

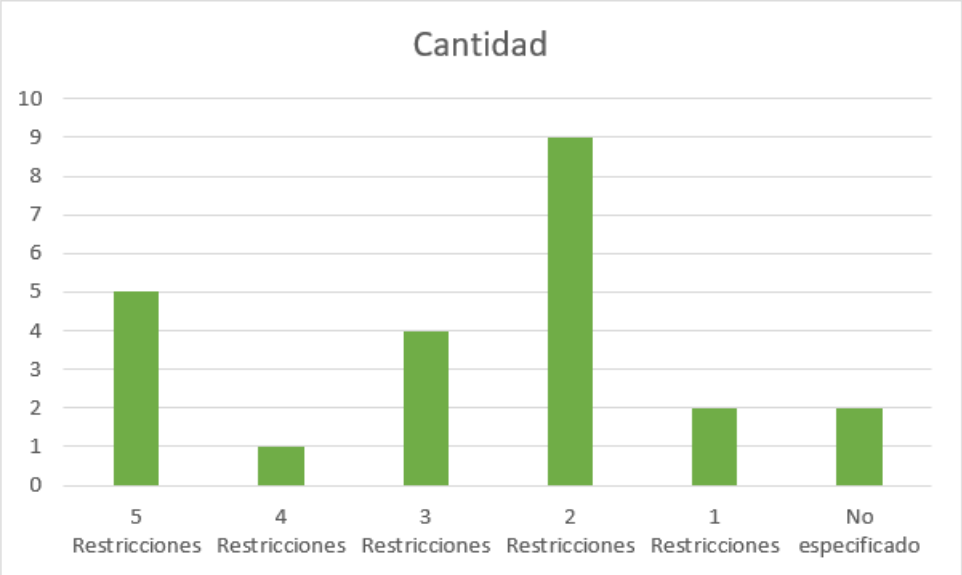


Figura 3: Cantidad de restricciones obligatorias que tenían que cumplir para solucionar los problemas en las investigaciones

1.3 Fundamentación teórica

La fundamentación teórica de esta investigación se centra en los conceptos y enfoques que sustentan el uso de algoritmos genéticos y otras técnicas de inteligencia artificial para optimizar la asignación de recursos académicos, como aulas y horarios docentes. A medida que las instituciones educativas crecen, el desafío de gestionar eficazmente estos recursos se vuelve cada vez más complejo, requiriendo soluciones avanzadas que puedan manejar múltiples restricciones y optimizar diversos objetivos. En este contexto, los algoritmos genéticos han surgido como una alternativa viable, debido a su capacidad para adaptarse y encontrar soluciones eficientes en problemas con un alto grado de complejidad y variabilidad. Esta sección explora las bases teóricas y el estado actual de la investigación en algoritmos genéticos aplicados a la gestión académica, así como otros enfoques complementarios que permiten entender mejor la implementación y los beneficios de estas tecnologías en el ámbito educativo.

1.3.1 Herramientas Tecnológicas

Las herramientas tecnológicas son recursos digitales o electrónicos diseñados para facilitar y optimizar diversas tareas en contextos profesionales, educativos o personales. Estas herramientas permiten automatizar procesos, mejorar la comunicación, gestionar información de manera eficiente y potenciar la productividad en múltiples áreas. En el ámbito académico y del desarrollo de software, su implementación resulta fundamental, ya que proporcionan entornos de trabajo colaborativos, plataformas de desarrollo, sistemas de gestión de bases de datos, control de versiones y lenguajes de programación. El uso adecuado de estas herramientas contribuye significativamente a la eficiencia, calidad y escalabilidad de los proyectos tecnológicos [45].

1.3.2 Aplicación Web

Una aplicación web es un programa de software que se ejecuta en un servidor y se accede a través de un navegador web mediante una conexión a Internet. A diferencia de las aplicaciones de escritorio tradicionales, las aplicaciones web no requieren instalación en el dispositivo del usuario, lo que facilita su acceso desde diversos dispositivos y ubicaciones [29]. Estas aplicaciones permiten a los usuarios interactuar con interfaces dinámicas para realizar diversas tareas, como completar formularios, gestionar bases de datos, colaborar en tiempo real o realizar transacciones en línea. La arquitectura generalmente sigue un modelo cliente-servidor, donde el navegador actúa como cliente que envía solicitudes al servidor, el cual procesa la lógica de negocio y responde con la información solicitada.

Entre las características clave de las aplicaciones web se incluyen:

- **Accesibilidad multiplataforma:** pueden ser utilizadas en diferentes sistemas operativos y dispositivos sin necesidad de adaptaciones específicas.

- **Actualizaciones centralizadas:** las mejoras y correcciones se implementan directamente en el servidor, sin requerir acciones por parte del usuario final.
- **Escalabilidad:** permiten ajustar recursos según la demanda de usuarios o transacciones.
- **Interactividad:** ofrecen experiencias de usuario enriquecidas mediante tecnologías como HTML5, CSS3 y JavaScript .

1.3.3 Backend

El backend, conocido como el "lado del servidor", es la parte de una aplicación web o sitio web que no es visible para el usuario final, pero que es esencial para su funcionamiento. Esta capa se encarga de gestionar la lógica de negocio, el almacenamiento y recuperación de datos, la seguridad y la integración con otros servicios[30].

Responsabilidades clave del backend:

- **Gestión de datos:** Interactúa con bases de datos para almacenar, recuperar y manipular información.
- **Lógica de negocio:** Implementa las reglas que definen el funcionamiento de la aplicación, como la autenticación de usuarios y la gestión de pedidos.
- **Seguridad:** Implementa medidas para proteger los datos y la aplicación contra amenazas, incluyendo la autenticación y autorización de usuarios.
- **Integraciones:** Facilita la comunicación con otros sistemas y servicios, como APIs de terceros y servicios de pago.
- **Rendimiento y escalabilidad:** Optimiza el uso de recursos del servidor y permite que la aplicación maneje un mayor número de usuarios y cargas de trabajo.

- **Mantenimiento:** Permite realizar actualizaciones y correcciones de errores sin afectar la experiencia del usuario final.

1.3.4 Frontend

El frontend es la parte de una aplicación o sitio web que interactúa directamente con el usuario final. Se refiere a la interfaz gráfica de usuario (GUI) que permite a los usuarios visualizar e interactuar con la aplicación, incluyendo elementos como botones, menús, formularios y gráficos [31]. Las tecnologías fundamentales utilizadas en el desarrollo frontend son:

- **HTML (HyperText Markup Language):** Define la estructura y el contenido de las páginas web.
- **CSS (Cascading Style Sheets):** Controla la presentación visual, incluyendo el diseño, colores y tipografía.
- **JavaScript:** Añade interactividad y dinamismo a las páginas, permitiendo responder a acciones del usuario sin necesidad de recargar la página.

El desarrollo frontend se centra en proporcionar una experiencia de usuario positiva, optimizando la accesibilidad, el rendimiento y la capacidad de respuesta en diferentes dispositivos y navegadores. Además, el frontend se comunica con el backend de la aplicación mediante solicitudes HTTP, enviando y recibiendo datos para actualizar la interfaz según las acciones del usuario.

1.3.5 Base de Datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Las bases de datos permiten que muchos usuarios accedan y consulten los

datos de forma rápida y segura al mismo tiempo mediante una lógica y un lenguaje muy complejos [32].

Tipos de bases de datos:

- **Bases de datos relacionales:** Organizan los datos en tablas con filas y columnas, utilizando el lenguaje SQL para la gestión y consulta de la información.
- **Bases de datos NoSQL:** Permiten almacenar y manipular datos no estructurados y semiestructurados, siendo ideales para aplicaciones web modernas que requieren flexibilidad en el manejo de datos.

Bases de datos en la nube: Son bases de datos desplegadas en plataformas de computación en la nube, ofreciendo ventajas como la automatización, escalabilidad y disponibilidad.

Bases de datos autónomas: Utilizan inteligencia artificial y aprendizaje automático para automatizar tareas de gestión rutinarias, como el ajuste de rendimiento y la seguridad.

1.3.6 Framework

Un framework, o marco de trabajo, es una estructura conceptual y tecnológica que proporciona un conjunto estandarizado de herramientas, bibliotecas y buenas prácticas para facilitar el desarrollo de software. Su objetivo principal es ofrecer una base sólida sobre la cual los desarrolladores puedan construir aplicaciones de manera más eficiente y organizada [33]. A diferencia de las bibliotecas, que ofrecen funciones específicas que el desarrollador puede llamar a voluntad, un framework impone una estructura y flujo de trabajo determinados, en los cuales el código del desarrollador se integra siguiendo las convenciones establecidas por el framework.

Ventajas de utilizar frameworks:

- **Eficiencia y productividad:** Al proporcionar componentes reutilizables y soluciones predefinidas para tareas comunes, los frameworks reducen significativamente el tiempo de desarrollo.

- **Consistencia y mantenimiento:** Fomentan una estructura de código uniforme, lo que facilita su comprensión y permite realizar modificaciones o mejoras de manera más eficiente en entornos colaborativos.

- **Seguridad y fiabilidad:** Muchos frameworks incluyen medidas de seguridad integradas y son mantenidos por comunidades activas que actualizan y corrigen vulnerabilidades regularmente.

- **Escalabilidad:** Están diseñados para manejar aplicaciones de diferentes tamaños y complejidades, permitiendo que las aplicaciones crezcan y se adapten a nuevas necesidades .

Tipos de frameworks:

- **Frontend:** Se enfocan en la interfaz de usuario y la experiencia del cliente. Ejemplos incluyen React, Angular y Vue.js.

- **Backend:** Manejan la lógica del servidor, bases de datos y autenticación. Ejemplos son Django (Python), Laravel (PHP) y Express.js (Node.js).

- **Full-stack:** Combinan funcionalidades de frontend y backend, ofreciendo una solución integral para el desarrollo de aplicaciones web.

La elección del framework adecuado depende de diversos factores, como el lenguaje de programación preferido, los requisitos específicos del proyecto y la experiencia del equipo de desarrollo. Utilizar un framework apropiado puede mejorar significativamente la calidad del software y acelerar el proceso de desarrollo.

1.3.7 API Rest

Una API REST (Representational State Transfer) es una interfaz de programación de aplicaciones que se ajusta a los principios del estilo arquitectónico REST. Este estilo fue definido por Roy Fielding en el año 2000 y se basa en el protocolo HTTP para facilitar la comunicación entre sistemas distribuidos. Las API REST permiten que diferentes aplicaciones interactúen entre sí de manera flexible y eficiente, siendo ampliamente utilizadas en arquitecturas de microservicios y aplicaciones web modernas. Las API REST operan mediante métodos HTTP estándar: **GET** para recuperar datos, **POST** para crear nuevos recursos, **PUT** para actualizar recursos existentes y **DELETE** para eliminarlos. Esta utilización de métodos HTTP proporciona una forma sencilla y coherente de realizar operaciones sobre los recursos disponibles en el servidor. Además, las API REST son independientes del lenguaje de programación, lo que permite su implementación en diversos entornos y plataformas. Una característica fundamental de las API REST es su naturaleza **sin estado** (*stateless*), lo que significa que cada solicitud del cliente al servidor debe contener toda la información necesaria para procesarla. El servidor no almacena ningún contexto de la comunicación entre peticiones, lo que mejora la escalabilidad y facilita la gestión de las sesiones. Esta propiedad es especialmente beneficiosa en aplicaciones que requieren alta disponibilidad y rendimiento [34].

1.3.8 ORM (Object-Relational Mapping)

Un ORM (Object-Relational Mapping) es una herramienta que facilita la interacción entre las aplicaciones y las bases de datos relacionales. Su función principal es mapear las estructuras de las bases de datos a objetos en el lenguaje de programación utilizado, permitiendo a los desarrolladores manipular datos mediante código orientado a objetos en lugar de escribir consultas SQL directamente. El uso de un ORM ofrece varias ventajas. Simplifica la escritura de operaciones comunes como crear, leer, actualizar y eliminar registros (CRUD), reduce el código repetitivo y mejora la mantenibilidad al separar la lógica de negocio de los detalles específicos de la base de datos. Además,

muchos ORM proporcionan medidas de seguridad integradas, como la prevención de inyecciones SQL, y facilitan la portabilidad del código entre diferentes sistemas de gestión de bases de datos [35].

Sin embargo, es importante destacar que el uso de un ORM no elimina la necesidad de comprender SQL. Aunque estas herramientas abstraen muchas operaciones, en situaciones complejas o de optimización avanzada, puede ser necesario escribir consultas SQL personalizadas. Por lo tanto, aprender SQL sigue siendo esencial para los desarrolladores backend que buscan una comprensión profunda y un control total sobre las interacciones con la base de datos.

1.3.9 Control de Versiones

El control de versiones es una práctica esencial en el desarrollo de software que permite rastrear y gestionar los cambios realizados en el código fuente a lo largo del tiempo. Esta disciplina facilita la colaboración entre múltiples desarrolladores, asegurando que cada modificación esté documentada y pueda ser revertida si es necesario. Los sistemas de control de versiones (VCS, por sus siglas en inglés) proporcionan un historial completo de las alteraciones en los archivos, incluyendo información sobre quién realizó cada cambio y cuándo. Esto es especialmente útil para identificar la introducción de errores o para comprender la evolución de una funcionalidad específica. Existen dos tipos principales de sistemas de control de versiones: centralizados y distribuidos. En los sistemas centralizados, como Subversion (SVN), existe un único repositorio central al que todos los desarrolladores acceden. Por otro lado, los sistemas distribuidos, como Git, permiten que cada desarrollador tenga una copia completa del repositorio, lo que mejora la flexibilidad y la resiliencia del desarrollo [36].

Implementar un sistema de control de versiones no solo mejora la eficiencia del equipo de desarrollo, sino que también reduce el riesgo de pérdida de información y facilita la integración continua y la entrega continua (CI/CD). En el contexto de esta investigación, el uso de un sistema de control de versiones es fundamental para

mantener la integridad y la trazabilidad del código fuente, especialmente cuando se trabaja en equipos distribuidos o en proyectos de larga duración.

1.3.10 Control de Entornos

El control de entornos en el desarrollo de software se refiere a la práctica de crear y gestionar entornos de ejecución aislados que contienen todas las dependencias necesarias para que una aplicación funcione correctamente. Estos entornos encapsulan el código de la aplicación junto con sus bibliotecas, configuraciones y otros componentes esenciales, asegurando que la aplicación se ejecute de manera consistente en diferentes sistemas y plataformas. Una de las principales ventajas de esta práctica es la portabilidad que ofrece. Al empaquetar la aplicación y sus dependencias en un entorno aislado, los desarrolladores pueden trasladar fácilmente la aplicación entre distintos entornos de desarrollo, prueba y producción sin preocuparse por problemas de compatibilidad. Esto facilita la colaboración entre equipos y mejora la eficiencia en el ciclo de vida del desarrollo de software. Además, el control de entornos contribuye a la seguridad y estabilidad de las aplicaciones. Al aislar cada entorno, se minimiza el riesgo de que cambios o errores en una parte del sistema afecten a otras. Esto es especialmente útil en arquitecturas basadas en microservicios, donde múltiples componentes independientes deben interactuar de manera confiable [37].

1.3.11 Servidor

Un servidor es un sistema informático diseñado para proporcionar servicios, recursos o datos a otros dispositivos, conocidos como clientes, a través de una red. Opera bajo el modelo cliente-servidor, en el cual el servidor atiende las solicitudes de los clientes y les proporciona las respuestas o recursos requeridos. Existen diversos tipos de servidores, cada uno especializado en funciones específicas. Por ejemplo, los servidores web gestionan y entregan contenido web a los navegadores de los usuarios;

los servidores de archivos almacenan y permiten el acceso a archivos compartidos; y los servidores de bases de datos gestionan sistemas de gestión de bases de datos, facilitando el almacenamiento y recuperación de datos estructurados.

En cuanto a su implementación, los servidores pueden ser físicos, instalados en las instalaciones de una organización (on-premise), o virtuales, alojados en la nube. La elección entre estas opciones depende de factores como el coste, la escalabilidad, el mantenimiento y las necesidades específicas de la empresa. Las pequeñas y medianas empresas suelen optar por soluciones en la nube debido a su flexibilidad y menor inversión inicial, mientras que las grandes organizaciones pueden preferir servidores locales para tener un mayor control y personalización [38].

En el contexto del desarrollo de software, los servidores desempeñan un papel crucial al alojar aplicaciones, gestionar bases de datos y facilitar la comunicación entre diferentes componentes del sistema. Su correcta configuración y administración son esenciales para garantizar el rendimiento, la seguridad y la disponibilidad de los servicios proporcionados a los usuarios finales.

1.3.12 Lenguaje de Programación

Un lenguaje de programación es un conjunto de instrucciones y reglas sintácticas que permiten a los desarrolladores comunicarse con una computadora para crear software, aplicaciones, y sistemas. A través de estos lenguajes, se definen algoritmos, estructuras de datos, y se controla el flujo de ejecución de un programa [39].

Principales características de los lenguajes de programación:

- **Sintaxis y semántica definidas:** Cada lenguaje tiene su propia gramática y significado, lo que permite expresar instrucciones de forma precisa y sin ambigüedades.
- **Abstracción:** Permiten representar procesos complejos de manera más simple, facilitando la resolución de problemas.

- **Paradigmas de programación:** Existen distintos enfoques como el imperativo, orientado a objetos, funcional o lógico, y cada lenguaje puede adoptar uno o varios de estos paradigmas.

- **Compilación o interpretación:** Los lenguajes pueden requerir ser compilados (como C++) o interpretados (como Python), dependiendo de cómo se transforma el código en instrucciones que entiende la máquina.

Ventajas de utilizar lenguajes de programación:

- **Automatización de procesos:** Permiten diseñar soluciones que ejecuten tareas de forma automática y repetible.

- **Reusabilidad del código:** El uso de funciones, clases y módulos permite construir soluciones modulares y reutilizables.

- **Versatilidad:** Existen lenguajes específicos para desarrollo web, móvil, inteligencia artificial, análisis de datos, entre otros.

En el desarrollo de software, la elección del lenguaje depende del tipo de aplicación, rendimiento requerido y familiaridad del equipo de desarrollo.

1.3.13 Encriptación

La encriptación, o cifrado, es un proceso criptográfico que transforma información legible, denominada texto plano, en un formato ilegible conocido como texto cifrado. Este procedimiento se realiza mediante un algoritmo de cifrado y una clave secreta, lo que garantiza la confidencialidad de los datos al impedir su acceso no autorizado [46]. Su objetivo principal es asegurar la información transmitida o almacenada, protegiéndola de la interceptación y el espionaje, lo que lo convierte en un pilar fundamental de la ciberseguridad moderna.

Principales características de la encriptación:

- Algoritmos y Claves: Se basa en algoritmos matemáticos complejos que definen las reglas de transformación y en una clave secreta, que es un dato vital para cifrar y descifrar la información.

- **Tipos de Cifrado:**

    - **Cifrado Simétrico:** Utiliza la misma clave tanto para encriptar como para desencriptar los datos. Ejemplos incluyen AES (Advanced Encryption Standard) y DES (Data Encryption Standard).

    - **Cifrado Asimétrico (o de Clave Pública):** Emplea un par de claves: una clave pública para encriptar y una clave privada para desencriptar. Ejemplos prominentes son RSA (Rivest-Shamir-Adleman) y los algoritmos de Curvas Elípticas (ECC).

- **Irreversibilidad (sin clave):** Sin la clave correcta, revertir el texto cifrado a su forma original es computacionalmente inviable, incluso con el poder de cómputo actual.

- **Confidencialidad:** Asegura que solo las partes autorizadas que poseen la clave secreta puedan acceder y comprender la información original.

- **Integridad y Autenticidad (con mecanismos adicionales):** Aunque su función principal es la confidencialidad, la encriptación a menudo se combina con funciones hash y firmas digitales para garantizar la integridad de los datos (que no han sido alterados) y la autenticidad (verificar la identidad del remitente).

Ventajas de utilizar la encriptación:

- Protección de la Privacidad: Resguarda la información personal y sensible de individuos y organizaciones frente a accesos no deseados.

- Cumplimiento Normativo: Facilita el cumplimiento de regulaciones de privacidad y protección de datos, como el GDPR o la HIPAA, que exigen la salvaguarda de la información.

- Confianza en las Comunicaciones: Permite establecer canales de comunicación seguros, vitales para transacciones bancarias, comercio electrónico y comunicaciones gubernamentales.

- Seguridad de Almacenamiento: Protege los datos en reposo, tanto en dispositivos locales como en servicios de almacenamiento en la nube, contra robos o filtraciones.

La encriptación es una tecnología esencial que evoluciona constantemente para enfrentar nuevas amenazas de seguridad, siendo la base para la protección de la información en el ecosistema digital global. Su correcta implementación es crucial para mantener la seguridad y la confianza en la era de la información.

1.3.14 Metodología Ágil

La metodología ágil es un enfoque de gestión de proyectos que prioriza la colaboración, la adaptabilidad y la entrega continua de valor. En lugar de seguir un plan rígido y lineal, los proyectos ágiles se dividen en ciclos cortos, o sprints, durante los cuales se completa un conjunto de tareas específicas. Al final de cada sprint, se obtiene retroalimentación de los implicados y se realizan ajustes en el plan según sea necesario. En el contexto de las organizaciones no gubernamentales (ONGs), la implementación de metodologías ágiles puede ser especialmente útil para abordar los retos que enfrentan estas organizaciones, como la complejidad de los proyectos, la escasez de recursos y la necesidad de adaptarse a los cambios constantes. Las metodologías ágiles permiten a las ONGs adaptarse a los cambios constantes en su entorno y responder de manera más efectiva a las necesidades de las comunidades a las que sirven.

Los principios clave de la metodología ágil incluyen la entrega temprana y frecuente de valor al cliente o beneficiario, la adaptación a los cambios, la colaboración entre todos los miembros del equipo y los agentes del ecosistema, la comunicación abierta y transparente, el enfoque en el trabajo realizado más que en la documentación extensa, y la búsqueda de la mejora continua en todos los aspectos del proyecto. La adopción

de metodologías ágiles en ONGs puede conducir a una mayor eficiencia y productividad, una entrega más rápida de valor, una mejora en la comunicación y la colaboración, una mayor participación de los agentes involucrados y una mayor flexibilidad y capacidad de adaptación. Estos beneficios pueden tener un impacto positivo en las comunidades a las que sirven las ONGs [40].

1.3.15 Algoritmos de Optimización

Los algoritmos de optimización son métodos que buscan la mejor solución dentro de un conjunto, minimizando o maximizando una función objetivo según ciertos criterios. Existen diversas técnicas de optimización, como los algoritmos de búsqueda, genéticos, gradiente descendente, optimización convexa y enjambre de partículas, aplicables en áreas como transporte y logística.

En el ámbito del transporte, los algoritmos de optimización se utilizan para mejorar la eficiencia en la planificación de rutas y la asignación de vehículos. Según varios autores [10],[11],[26],[28], la implementación de estos algoritmos contribuye significativamente a la reducción de costos operativos. Además, optimizan el servicio al usuario al permitir que los transportistas respeten los horarios definidos, lo que incrementa la satisfacción de los clientes [26]. Entre estos algoritmos se destaca el algoritmo difuso de optimización multiobjetivo, una técnica que integra los principios de la lógica difusa y métodos de optimización multiobjetivo para abordar problemas con múltiples criterios o metas en conflicto. Este enfoque es especialmente útil en contextos donde los objetivos no son excluyentes y las restricciones pueden ser ambiguas o inciertas, permitiendo tomar decisiones de manera más flexible y efectiva. En estos casos, el objetivo no es encontrar una única solución "óptima", sino un conjunto de soluciones conocido como frente de Pareto, en el cual cada alternativa representa un equilibrio aceptable entre los diferentes objetivos en competencia.

Por otro lado, en el ámbito académico, los algoritmos de optimización se aplican ampliamente en la planificación de recursos, asignación de horarios y organización de actividades como se indica en las investigaciones [9],[12],[14],[15]. Estos algoritmos

ayudan a distribuir eficientemente aulas, profesores y horarios, maximizando el uso de los recursos disponibles como se menciona en [20]. Además, facilitan la organización de exámenes y actividades extracurriculares mencionado en [9], reduciendo conflictos y sobrecargas en el calendario académico. En [1],[8],[17],[22],[25] los autores emplean el algoritmo genético cuya técnica de optimización es inspirada en los principios de la selección natural y la genética evolutiva, que se emplea para resolver problemas complejos en los cuales el espacio de soluciones es muy amplio, se enfoca en aproximar soluciones óptimas en situaciones donde los métodos convencionales pueden ser ineficaces debido a la complejidad o al tamaño del problema. En un algoritmo genético, se representa cada solución potencial como un individuo o cromosoma dentro de una población. A través de iteraciones o generaciones, el algoritmo evoluciona la población de soluciones, mejorando la calidad de las soluciones generadas hasta alcanzar un conjunto de individuos que representan las mejores soluciones posibles para el problema.

En [16],[19],[24], los autores combinan el uso del algoritmo genético con la optimización por enjambre de partículas (PSO, por sus siglas en inglés), una técnica inspirada en el comportamiento social de los enjambres en la naturaleza, como bandadas de aves o cardúmenes de peces. PSO, que se utiliza tanto en problemas de optimización continua como discreta, destaca por su simplicidad y su capacidad para encontrar soluciones eficientes en espacios de búsqueda complejos. En este enfoque, cada partícula representa una posible solución y ajusta dinámicamente su posición y velocidad en el espacio de búsqueda, influenciada tanto por su experiencia personal como por la del grupo. Esta interacción permite que las partículas exploren y optimicen sus posiciones, logrando un equilibrio entre la exploración y la explotación de soluciones. En el contexto de la optimización de recursos y horarios académicos, PSO es particularmente útil, debido a que permite ajustarse a múltiples restricciones y adaptarse a cambios en el entorno, como la disponibilidad de aulas o las necesidades de los estudiantes [16],[24]. Con este enfoque, se maximiza el uso eficiente de los recursos, logrando soluciones prácticas y adaptadas a las necesidades específicas de la institución educativa.

1.3.16 Restricciones en la Optimización de Horarios

En el contexto académico, la optimización de horarios es un problema complejo que debe resolver múltiples restricciones. Estas restricciones se clasifican en dos tipos:

- **Restricciones duras (obligatorias):** Son aquellas que deben cumplirse necesariamente para que el horario sea válido. El incumplimiento de estas restricciones resulta en horarios inviables y, por lo tanto, no se consideran soluciones aceptables.
- **Restricciones suaves (flexibles):** Son restricciones deseables, pero no estrictamente obligatorias. El incumplimiento de estas restricciones no invalida un horario, pero puede reducir su calidad o nivel de satisfacción de los involucrados.

Para abordar este problema de optimización con múltiples restricciones, se emplean algoritmos avanzados que permiten encontrar soluciones equilibradas entre estas restricciones duras y suaves.

1.4 Objetivos

1.4.1 Objetivo general

Implementar una aplicación web con algoritmos de optimización para la reducción de tiempo en el proceso de creación de horarios y optimización de espacios académicos.

1.4.2 Objetivos específicos

- Analizar el proceso actual de creación de horarios para detectar ineficiencias y oportunidades de optimización.
- Investigar algoritmos de optimización que permitan reducir el tiempo de creación de horarios, considerando las restricciones del sistema académico.

- Desarrollar una aplicación web que automatice el proceso de creación de horarios, utilizando los algoritmos de optimización seleccionados.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

Para el desarrollo de esta investigación se emplearon como materiales principales dos instrumentos metodológicos: entrevistas y una encuesta de validación. Las entrevistas fueron dirigidas a los actores clave involucrados en la gestión de horarios académicos dentro de la facultad, con el objetivo de identificar las limitaciones del proceso actual y levantar los requerimientos necesarios para el nuevo sistema propuesto. Previo a su aplicación, se diseñó una encuesta destinada a validar la claridad, pertinencia y relevancia de las preguntas formuladas en la entrevista. Esta validación permitió asegurar que el instrumento recoja información precisa y alineada con los objetivos de la investigación. El detalle de las preguntas utilizadas en la entrevista se encuentra documentado en el **Anexo A**. Estos materiales resultaron fundamentales para sustentar las decisiones de diseño del sistema y asegurar su adecuación a las necesidades reales del entorno académico.

2.2 Métodos

La Figura 4 representa el esquema metodológico propuesto para esta investigación, el cual abarca desde la identificación del problema hasta la implementación de algoritmos de optimización. En las primeras etapas, se realiza un análisis del problema y una revisión sistemática de literatura, con el fin de identificar enfoques previos y construir un dataset relevante. Posteriormente, se lleva a cabo la recolección de datos mediante entrevistas estructuradas con los actores del proceso, permitiendo identificar limitaciones del sistema actual y levantar los requerimientos del nuevo sistema. Con esta información, se rediseña la base de datos y se desarrollan las funcionalidades necesarias en una aplicación web. Paralelamente, se diseñan interfaces y se realizan pruebas para asegurar el cumplimiento de los requerimientos. Finalmente, se analizan y seleccionan algoritmos de optimización adecuados, los cuales son implementados y ajustados en Python, integrándolos al sistema con el objetivo de generar horarios

académicos que cumplan eficazmente con todas las restricciones y necesidades identificadas.

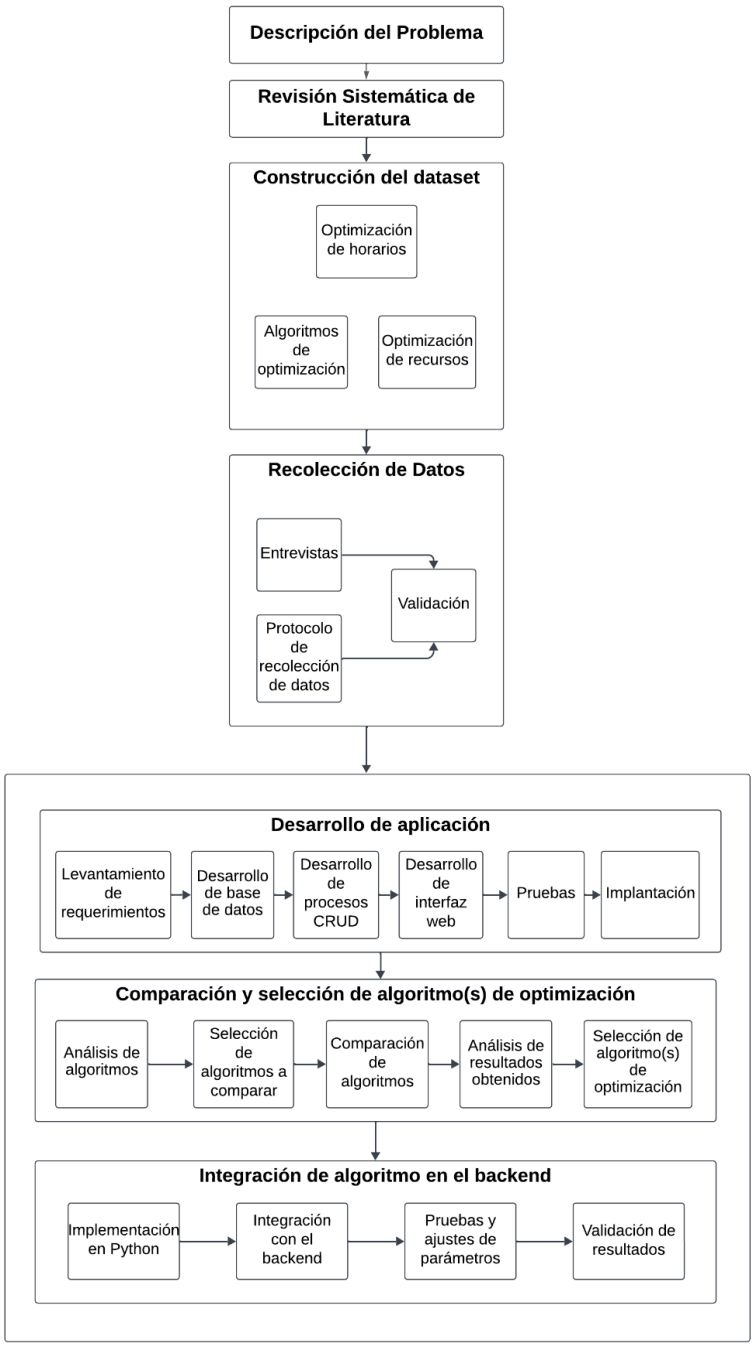


Figura 4. Esquema de la Metodología

2.2.1 Modalidad de la investigación

La presente investigación combinó múltiples enfoques metodológicos con el fin de abordar de forma integral las problemáticas asociadas a la asignación y optimización de horarios académicos en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI). A continuación, se detallan las modalidades que fueron aplicadas:

a. Investigación experimental

Se recurrió a la investigación experimental con el objetivo de comparar el rendimiento entre un algoritmo de optimización y el proceso manual que se utilizaba actualmente para la generación de horarios académicos. Esta comparación permitió medir la eficiencia, precisión y capacidad de respuesta del nuevo sistema basado en inteligencia artificial en contraste con el método tradicional.

b. Investigación de campo

La investigación también se enmarcó dentro de la modalidad de campo, ya que se recopiló información directa de los actores involucrados en la planificación de horarios, como personal administrativo y laboratoristas. Esta aproximación posibilitó un levantamiento detallado de las necesidades reales, restricciones operativas y limitaciones del sistema existente.

c. Investigación bibliográfica

Se complementó el trabajo con una modalidad bibliográfica, fundamentada en la revisión de literatura científica y técnica relevante sobre algoritmos de optimización aplicados a problemas de programación y asignación de recursos. Esta revisión teórica permitió identificar los enfoques más efectivos y adaptarlos al contexto educativo específico de la FISEI.

d. Enfoque cuantitativo

Finalmente, se adoptó un enfoque cuantitativo para evaluar objetivamente los resultados del sistema propuesto. Se midieron variables como el tiempo de procesamiento, el cumplimiento de restricciones, la eficiencia en la distribución de

recursos y la reducción de errores. Estos indicadores proporcionaron evidencia empírica sobre la efectividad del algoritmo aplicado.

2.2.2 Población y muestra

En esta investigación, la población está conformada por un grupo reducido de 7 personas directamente involucradas en la gestión y coordinación de horarios académicos en la facultad. Este grupo tiene el siguiente estrato:

- **Personal de laboratorio y encargados de horarios:** Este estrato incluye a 7 personas responsables de la generación y gestión de los horarios, así como del uso de los espacios académicos.

Dado el tamaño reducido de la población y la unicidad del estrato, se considerará la totalidad de la población como muestra.

a. Cálculo del tamaño de la muestra

Como la población total es de 7 personas, se tomará toda la población como muestra (N = 7). Por lo tanto, no es necesario aplicar un muestreo proporcional.

Resultados finales:

La muestra estará compuesta por la totalidad de la población:

- **Personal de laboratorio y encargados de horarios:** 7 personas.

Este enfoque garantiza la inclusión de todos los individuos relevantes para la recolección de datos, permitiendo obtener información completa y precisa para el diseño e implementación del sistema de optimización de horarios.

2.2.3 Recolección de información

En la presente investigación se emplearon entrevistas como herramienta principal para la recolección de información, dirigidas a técnicos y personal académico involucrados en la generación y supervisión de horarios en la facultad. La finalidad fue comprender las limitaciones del sistema actual, las necesidades específicas de los usuarios y su percepción sobre la implementación de un sistema automatizado basado en algoritmos de optimización. Las entrevistas se llevaron a cabo en un entorno académico tranquilo, asegurando la comodidad de los participantes.

a. Validación del instrumento

Para el proceso de validación, se aplicará una encuesta al personal de laboratorio, utilizando una escala de Likert construida a partir de las preguntas a utilizar en la entrevista técnica. El propósito de esta encuesta es evaluar tres aspectos: la claridad, la coherencia y la relevancia de las preguntas formuladas, en el contexto de los procesos relacionados con la gestión y generación de horarios. La información cuantitativa recopilada a través de esta herramienta permitirá ajustar y optimizar la calidad de las preguntas utilizadas en futuras evaluaciones.

Tabla 4. Escala de Likert para evaluación de preguntas

Criterios	Muy Baja	Baja	Medio	Alta	Muy Alto
Claridad	1	2	3	4	5
Coherencia	1	2	3	4	5
Relevancia	1	2	3	4	5

Para comprender a fondo las vivencias y requerimientos de los responsables de laboratorios y del coordinador de administración de redes en cuanto a la gestión y creación de horarios, se llevó a cabo una entrevista técnica. La información cualitativa recabada se analizó a través de una encuesta con escala de Likert, la cual evaluó la claridad, coherencia y relevancia de cada pregunta de la entrevista. Los resultados de esta evaluación permitieron calcular los promedios para cada una de las dimensiones mencionadas, los cuales se presentan en la tabla que sigue:

Tabla 5. Promedio de los criterios evaluados en la encuesta de Likert

Pregunta	Claridad	Coherencia	Relevancia
P1	4.33	4.7	4.7
P2	5	4.7	4.7
P3	4.33	4.7	4.33
P4	4	4.7	4.33
P5	4.33	4	4.7
P6	4	4.33	4.33
P7	4.7	4.7	5
P8	5	4.33	4.33
P9	4.33	4	4.33
P10	4	5	4.33
P11	4.33	4.7	4.33
P12	4.7	4.7	5
P13	4.7	4.33	4.7
P14	4.7	5	4.7
P15	4.33	4.33	4.33
P16	4.33	4.7	4.7

El protocolo diseñado para las entrevistas técnicas incluyó preguntas validadas mediante este método, lo que aseguró la calidad y pertinencia de la información recolectada. Los datos obtenidos fueron analizados con el propósito de identificar patrones, problemáticas recurrentes y expectativas prioritarias de los usuarios, insumos clave para el diseño e implementación del sistema de optimización. El detalle completo de las preguntas formuladas se encuentra en el **Anexo A**.

Este método se realizó basándose en [41] en el cual también se establecen que los promedios obtenidos en escala de Likert que se encuentran en un rango entre 3.51 y 4.5 corresponden a una valoración alta y los promedios que superan 4.51 se consideran muy altos. Este sistema de clasificación se emplea para interpretar la claridad, coherencia y relevancia de los ítems en procesos de validación de instrumentos de investigación cualitativa, asegurando que las valoraciones reflejen un nivel sólido y satisfactorio de aceptación por parte de los participantes. En este contexto, los resultados obtenidos en la evaluación de las dimensiones mencionadas presentados en la Tabla 5 se ubicaron predominantemente en los rangos alto y muy alto, lo que respalda la eficacia del instrumento utilizado. Estos hallazgos confirman que las preguntas formuladas fueron percibidas como claras, coherentes y altamente relevantes en función de los objetivos del estudio, contribuyendo así a una recolección de datos precisa y alineada.

b. Resultados de las entrevistas aplicadas a los encargados de laboratorio de la facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Los resultados obtenidos a partir de las entrevistas aplicadas al personal encargado de laboratorio, cuyos análisis completos se presentan en las Tabla 18, Tabla 19 y Tabla 20, permitieron identificar información valiosa sobre las dinámicas actuales de gestión de horarios, el uso de espacios académicos y las principales problemáticas enfrentadas en estos procesos. Las respuestas reflejan tanto las experiencias como las necesidades del personal involucrado, aportando elementos clave que orientan el diseño y desarrollo del sistema propuesto.

A continuación, se presenta en la Figura 5 un resume de forma visual y cuantitativa las palabras más recurrentes en las respuestas proporcionadas durante las entrevistas. Este análisis léxico permite identificar con mayor claridad los temas centrales abordados por los participantes, así como los patrones y conceptos que se repitieron con mayor frecuencia. La visualización de estos elementos resulta clave para comprender las preocupaciones, necesidades y prioridades expresadas por el personal encargado de laboratorio, y proporciona una base sólida para orientar las decisiones en el diseño e implementación del sistema de optimización de horarios.

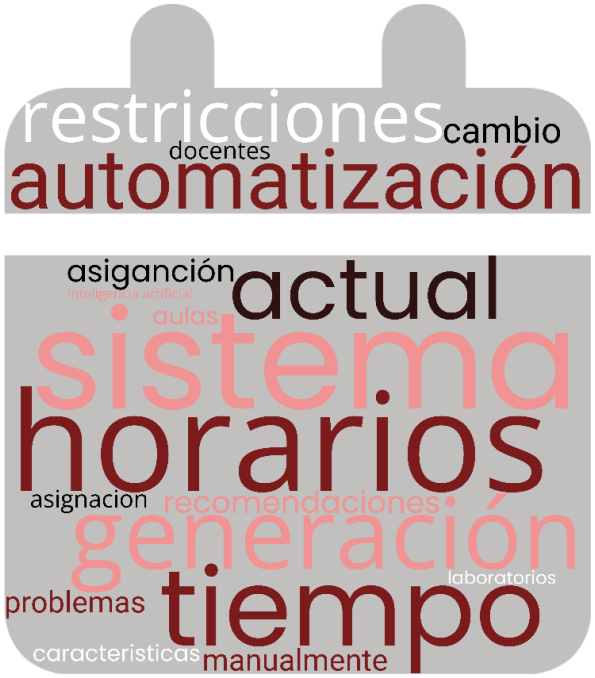


Figura 5. Nube de palabras de las entrevistas aplicadas al personal de laboratorio

2.2.4 Procesamiento y análisis de datos

Para analizar las entrevistas realizadas que se encuentra en el **Anexo A**, al personal de la Facultad de Sistemas, Electrónica e Industrial (FISEI), se utilizó un enfoque cualitativo orientado a identificar información clave sobre el uso del sistema actual de asignación de horarios y sus necesidades asociadas. Este análisis se basó en la identificación de patrones comunes en las respuestas, enfocándose en las fortalezas, limitaciones y oportunidades de mejora percibidas por los usuarios del sistema .

Entre los puntos destacados, se identificó que el sistema actual permite ciertas funcionalidades básicas, como la asignación de horarios por bloques. Sin embargo, los entrevistados señalaron lo siguiente:

- **Falta de automatización en el proceso de generación de horarios:**

El sistema actual no cuenta con herramientas que optimicen el tiempo de trabajo, lo que obliga a realizar tareas de forma manual, incrementando el riesgo de errores.

- **Ausencia de integración de restricciones clave:**

No se consideran de manera efectiva restricciones importantes como la disponibilidad de aulas, laboratorios o las preferencias horarias de los docentes, lo que afecta negativamente la calidad y viabilidad de la planificación académica.

Otro aspecto relevante fue la solicitud de funcionalidades adicionales, como la generación de reportes específicos, la capacidad de realizar ajustes manuales tras la optimización y la posibilidad de elegir entre varias opciones de horarios propuestos. Estas observaciones permiten identificar áreas prioritarias que deben ser atendidas en el diseño del nuevo sistema.

Además, los entrevistados recalcaron la importancia de contar con un sistema que permita la edición y generación de horarios de forma más eficiente y que reduzca la dependencia de herramientas externas o procesos manuales tediosos. Esto resalta la necesidad de una solución tecnológica más moderna y ajustada a las necesidades actuales de la facultad.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

3.1 Análisis del proceso actual

El análisis del proceso actual de gestión de horarios en la facultad se fundamenta en la información recopilada mediante entrevistas, observaciones y revisión documental. Se identifican las prácticas vigentes, así como las dificultades y limitaciones enfrentadas por el personal encargado, lo que permite detectar oportunidades de mejora. Esta comprensión resulta fundamental para orientar el diseño y desarrollo del sistema propuesto, asegurando que responda a las necesidades reales del contexto institucional.

3.1.1 Identificador de factores necesarios

Actualmente, el proceso de creación de horario en la facultad es un proceso manual el cual toma mucho tiempo, debido a que la persona encargada del proceso debe tomar en cuenta varios factores, que se describen a continuación:

a. Características de aulas y laboratorios

Cada aula y laboratorio cuenta con ciertas especificaciones las cuales son tomadas en cuenta para poder asignar los horarios de clase. Estas son:

- Capacidad máxima de estudiantes.
- Cantidad de computadoras.
- Disponibilidad de proyector.
- Disponibilidad de aire acondicionado.
- Tipo de aula.

b. Actividades extracurriculares

La mayoría de docentes llevan actividades extracurriculares las cuales van desde reuniones, coordinación, investigación, entre otras. Las cuales son obligatorias y deben ser realizadas en horarios y aulas las cuales no interrumpan las actividades académicas de los docentes y estudiantes.

c. Requisitos de materias

Al inicio de cada semestre la persona encargada recibe un listado con todas las materias existentes de cada carrera y semestre, en la cual se especifican las necesidades de cada una de ellas, como:

- Necesidad de computadoras o maquinas específicas.
- Estudiantes estimados para la materia.
- Horas por semana.

d. Necesidades de docentes

También en el proceso de creación de los horarios se toma en cuenta las necesidades de los docentes para poder facilitar la adaptabilidad de los horarios a los mismos, estos son algunos criterios que se evalúan:

- Hora de Ingreso y Salida de clases.
- Días no disponibles, en estos días los docentes se encargan de realizar actividades extracurriculares.
- Materias máximas impartidas por día.

3.1.2 Proceso actual

Actualmente, el proceso de creación de horario en la facultad es un proceso manual el cual toma mucho tiempo, porque la persona encargada del proceso debe tomar en cuenta varios factores, que se describen a continuación:

1. **Recepción de distributivos y restricciones**

    - El encargado de laboratorio inicia el proceso con la recepción de los **distributivos de los docentes**, que contienen la información sobre las materias asignadas.

    - Posteriormente, se reciben también las **restricciones horarias** de los docentes (por ejemplo, días no disponibles o limitaciones personales).

2. **Registro de datos en la aplicación Horarios aSc**

    - Se procede a ingresar todos los **datos académicos** (carreras, semestres, materias, docentes, paralelos) y la **infraestructura académica** (aulas, laboratorios, pisos, edificios) en la **aplicación horarios aSc**, una herramienta de terceros utilizada para la creación inicial de los horarios.

3. **Creación y envío de horarios**

    - Utilizando la aplicación aSc, se realiza la **creación manual del horario** tomando en cuenta las restricciones y condiciones.

    - Una vez generados los horarios preliminares, estos se **envían a los coordinadores de carrera** para su revisión.

4. **Revisión y aprobación por parte del coordinador**

    - El **coordinador de carrera revisa los horarios**. En caso de identificar errores o inconsistencias, **solicita modificaciones al encargado**, quien los corrige en la herramienta horarios aSc.

    - Si no hay cambios, el coordinador **aprueba los horarios** y se procede con la siguiente fase.

5. **Registro en la aplicación FISEI**

    - Una vez aprobado los horarios, se ingresan los mismos **datos académicos e infraestructura** en la **aplicación horarios FISEI**, para mantener la consistencia entre ambas plataformas.

- Se **registran los horarios creados** uno por uno dentro de la aplicación FISEI.
- Luego, se realiza una **revisión final de los horarios** dentro del sistema FISEI para verificar la coherencia.

6. **Validación y generación de hojas de control**
   - Si existen correcciones posteriores, se realizan las respectivas **modificaciones** y se valida nuevamente el horario.
   - En caso de requerirlo, se seleccionan los datos necesarios para **generar hojas de control**, utilizadas por los laboratorios para la gestión diaria.
7. **Publicación y difusión**
   - Finalmente, los horarios aprobados se **publican en la página oficial de la FISEI**.
   - La **difusión se realiza a través de medios oficiales**, como redes sociales o correo institucional.

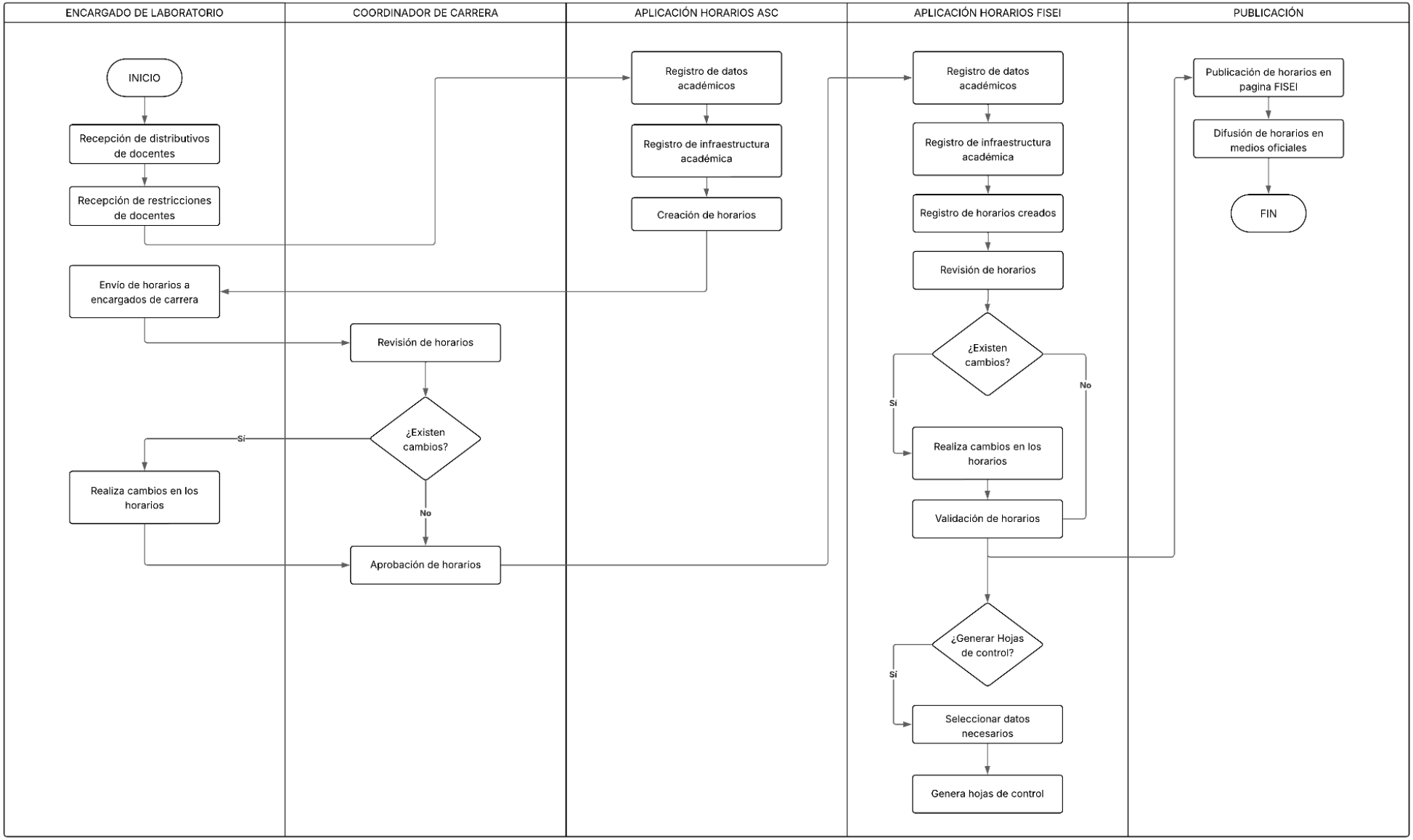


Figura 6. Diagrama de flujo del proceso actual

3.1.3 Ineficiencias en el proceso actual

Actualmente, el proceso manual para la asignación de horarios y aulas en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial presenta diversas limitaciones, entre las cuales se destacan:

- **Tiempo:** La elaboración de los horarios demanda una gran cantidad de tiempo, debido a que se realiza de manera manual. Esta tarea consume varias jornadas laborales y requiere de una constante revisión para evitar errores o conflictos en la asignación.

- **Uso limitado del software actual:** Solo una persona puede utilizar el software actual al mismo tiempo, lo que genera un cuello de botella en el proceso. Esto restringe la posibilidad de colaboración entre varios miembros del equipo, ralentizando aún más la planificación.

- **Falta de criterios objetivos en la asignación:** En algunos casos, el proceso puede verse afectado por la omisión de ciertas características relevantes de las aulas (como capacidad, recursos tecnológicos disponibles o ubicación). Esto puede deberse a que las personas encargadas de asignar los horarios, al no contar con una herramienta de verificación automática, pueden no considerar todos los factores necesarios, lo que genera una asignación ineficiente o poco óptima de los espacios.

3.1.4 Oportunidades de optimización

A partir del diagnóstico realizado al proceso actual de planificación de horarios académicos, se identificaron diversas oportunidades para optimizar la gestión institucional. Una de las principales mejoras potenciales radica en la **automatización del proceso de asignación de horarios**, lo cual permitiría reducir significativamente el tiempo requerido para la planificación y minimizar los errores humanos. Asimismo,

se presenta la posibilidad de **incorporar algoritmos de optimización** que consideren de forma dinámica las restricciones existentes, como la disponibilidad de aulas, los requerimientos técnicos de las materias, y las preferencias de los docentes, facilitando así una asignación más precisa y eficiente. Otra oportunidad clave es la **centralización de la información en un sistema único**, que permita una gestión integral y en tiempo real de los datos académicos, evitando la duplicidad de tareas y mejorando la trazabilidad. Finalmente, la **incorporación de herramientas de visualización y validación previa de horarios** brindaría mayor control a los usuarios y permitiría detectar inconsistencias antes de la publicación oficial, mejorando la calidad y confiabilidad del proceso.

3.2 Análisis de algoritmos de optimización

La generación automática de horarios académicos es un problema complejo de optimización que requiere considerar múltiples restricciones duras y suaves. Por ello, la selección adecuada del algoritmo de optimización es clave para obtener soluciones viables, eficientes y adaptadas a las necesidades institucionales. El objetivo de esta sección es analizar los algoritmos más representativos utilizados en investigaciones recientes, tomando como base el perfil de tesis y la literatura revisada, alineándose con las necesidades específicas y necesarias para la generación de horarios.

3.2.1 Análisis y selección de algoritmo

Dado que el sistema web estará orientado a la gestión de restricciones y a la generación de horarios académicos basados en dichas condiciones, resulta fundamental identificar y seleccionar el o los algoritmos más apropiados para llevar a cabo tanto el proceso de optimización como su implementación en el sistema. A continuación, se describen los algoritmos más relevantes, considerando sus principales características, fortalezas y debilidades dentro del contexto de la generación de horarios, con el fin de evaluar su adecuación a los requerimientos del proyecto y contribuir a su correcta ejecución.

a. Algoritmo Genético (GA)

El algoritmo genético es uno de los más utilizados en los estudios analizados [1], [8], [10], [16], [17], [19], [22], [24], [25], [26], [27], y se basa en los principios de la evolución natural, como la selección, el cruce y la mutación, para encontrar soluciones óptimas. Este enfoque es particularmente eficaz en problemas complejos con múltiples restricciones, como la generación de horarios académicos. Su fortaleza radica en la capacidad de explorar espacios de búsqueda amplios y encontrar soluciones viables cercanas al óptimo global a través de un proceso iterativo, en el que se optimiza la satisfacción de las restricciones definidas. No obstante, su tiempo de convergencia puede ser elevado, lo cual representa una de sus principales desventajas.

b. Particle Swarm Optimization (PSO)

El algoritmo de optimización por enjambre de partículas (PSO), utilizado en [16], [19], [24], está inspirado en el comportamiento colectivo de grupos de animales, como aves o peces. Este método trabaja con una población de soluciones denominadas "partículas", las cuales se desplazan dentro del espacio de búsqueda influenciadas por su propia experiencia y la de sus compañeras. PSO se caracteriza por su simplicidad de implementación y por su capacidad de adaptarse a cambios dinámicos, lo que lo hace especialmente útil en contextos variables como la generación de horarios académicos. Cada partícula representa una posible solución y coopera con las demás para aproximarse a un óptimo global. Sin embargo, una de sus limitaciones es la posibilidad de quedar atrapado en óptimos locales si los parámetros no son ajustados adecuadamente. Este enfoque permite explorar combinaciones equilibradas que satisfacen múltiples restricciones, contribuyendo a la construcción de horarios eficientes.

c. Simulated Annealing (SA)

El algoritmo de recocido simulado (Simulated Annealing), utilizado en [16], [20], [21], emula el proceso físico de enfriamiento de los metales, donde se permite cierto desorden antes de alcanzar un estado estable. Esta técnica es particularmente útil para evitar caer en óptimos locales, ya que permite aceptar soluciones temporalmente peores durante ciertas etapas del proceso de búsqueda, lo que favorece una mejor

exploración del espacio de soluciones. En el contexto de la generación de horarios, este enfoque permite realizar pequeños ajustes iterativos que contribuyen a mejorar gradualmente la solución, considerando tanto restricciones duras como suaves. No obstante, su rendimiento depende en gran medida de una adecuada configuración de parámetros como la temperatura inicial y la tasa de enfriamiento, los cuales determinan la eficacia del algoritmo en encontrar una solución óptima.

d. Algoritmo de programación dinámica

Aunque menos frecuente, es útil en problemas donde se puede descomponer la solución en etapas. Su principal desventaja es el consumo elevado de memoria y tiempo en problemas grandes.

e. Programación Lineal

La programación lineal es un método matemático orientado a encontrar la mejor solución posible, ya sea de maximización o minimización, de una función lineal conocida como función objetivo. Esta optimización se realiza dentro de un conjunto definido de restricciones, también lineales, que establecen los límites o condiciones que deben respetarse. Dichas restricciones se formulan a través de ecuaciones o inecuaciones lineales que delimitan el espacio de soluciones factibles.

f. Algoritmos híbridos

Algunos estudios combinan algoritmos como el genético con búsqueda local o PSO para mejorar los tiempos de ejecución y la calidad de las soluciones. Esta combinación busca aprovechar las fortalezas de cada enfoque y compensar sus debilidades.

3.2.2 Comparativa de algoritmos

La siguiente tabla comparativa presenta un análisis detallado de diversos algoritmos, evaluados bajo criterios clave para su aplicación en la resolución de problemas complejos. Estos criterios incluyen el **Tipo de Problema** que cada algoritmo aborda eficazmente, su capacidad de **Manejo de Restricciones** específicas, la **Eficiencia**

Computacional en términos de velocidad y uso de recursos, la **Facilidad de Implementación** práctica, y su **Adaptabilidad** a diferentes tipos de problemas y condiciones operativas. Esta evaluación permitirá una comprensión clara de las fortalezas y debilidades de cada algoritmo, facilitando la selección más adecuada para desafíos particulares.

Tabla 6. Comparativa de algoritmos

Algoritmo	Tipo de Problema	Manejo de Restricciones	Eficiencia Computacional	Facilidad de Implementación	Adaptabilidad
Algoritmo Genético (GA)	Combinatorio, Discreto, Multimodal	Alta	Media	Media	Alta
PSO	Continuo, Multidimensional	Media	Alta	Alta	Media
SA	Combinatorio, Multimodal	Alta	Media	Alta	Alta
Programación Dinámica	Secuencial, Estructurado	Alta	Baja	Baja	Baja
Programación Lineal	Lineal, Estructurado	Alta	Alta	Media	Baja
Algoritmos Híbridos	Combinatorio, Multimodal, Complejo	Muy Alta	Alta	Baja	Muy Alta

En base a la investigación realizada y a los resultados obtenidos en la Tabla 6 se llegó a la conclusión que la **mejor opción** es la **combinación de algoritmos genéticos (AG) y programación lineal (PL),** debido a que representa una estrategia robusta y eficaz para abordad el problema de generación de horarios académicos, los cuales se caracteriza por múltiples restricciones y la necesidad de adaptabilidad de los horarios a esas restricciones y a las infraestructuras disponibles de la facultad, esta conclusión se llevó a cabo tomando en cuenta los siguientes puntos:

a. Sinergia entre Algoritmos Genéticos (AG) y Programación Lineal (PL)

Los AG son eficaces para explorar amplios espacios de soluciones y encontrar soluciones cercanas al óptimo global, especialmente en problemas con múltiples restricciones y variables complejas. Por otro lado, la PL es adecuada para optimizar funciones lineales sujetas a restricciones lineales, proporcionando soluciones exactas

y eficientes cuando el problema puede modelarse de esta manera. La combinación de AG y PL permite aprovechar las fortalezas de ambos enfoques: los AG pueden generar soluciones iniciales que luego se refinan mediante PL para mejorar la calidad de la solución. Este enfoque híbrido es especialmente útil en problemas de generación de horarios académicos, donde se requiere tanto exploración global como optimización local.

b. Adaptabilidad y Flexibilidad

Los AG son especialmente eficaces en entornos donde las variables y restricciones cambian con frecuencia. Su capacidad para explorar múltiples soluciones y evolucionar hacia las más óptimas permite adaptarse rápidamente a nuevas condiciones sin necesidad de reestructurar completamente el modelo. Esta flexibilidad es crucial en contextos educativos, donde los cambios en la disponibilidad de recursos humanos y físicos son comunes.

La PL es una técnica matemática que busca optimizar una función objetivo lineal sujeta a un conjunto de restricciones también lineales. Su estructura permite incorporar nuevas restricciones de manera sistemática, facilitando la adaptación del modelo a diferentes escenarios. Esto es particularmente útil cuando se requiere ajustar el modelo a cambios en las políticas institucionales o en la disponibilidad de recursos.

c. Manejo eficaz de restricciones complejas

Los AG son especialmente adecuados para explorar grandes espacios de soluciones en problemas con múltiples restricciones, como la disponibilidad de docentes, asignación de aulas y distribución de materias. Su capacidad para encontrar soluciones cercanas al óptimo global los hace ideales para este tipo de problemas. Por otro lado, la PL es eficaz para optimizar funciones lineales sujetas a restricciones lineales, proporcionando soluciones exactas y eficientes cuando el problema puede modelarse de esta manera.

d. Eficiencia comprobada

Los AG son técnicas de búsqueda heurística inspiradas en los procesos de evolución natural. Su capacidad para explorar grandes espacios de soluciones y adaptarse a

múltiples restricciones los hace especialmente adecuados para problemas complejos como la generación de horarios. Un estudio realizado en la Universidad Nacional Agraria de la Selva (UNAS) evidenció que la implementación de un algoritmo genético para la elaboración de horarios de clases resultó en una mejora significativa en la eficiencia del proceso. Específicamente, se observó una reducción del 96.97% en el tiempo de elaboración y del 96.46% en el esfuerzo requerido, en comparación con los métodos tradicionales. Además, se logró un cumplimiento del 100% de las restricciones duras, lo que indica una alta calidad en las soluciones generadas [42].

La PL es una técnica matemática que busca optimizar una función objetivo lineal sujeta a un conjunto de restricciones también lineales. Su aplicabilidad en la generación de horarios ha sido ampliamente estudiada. Por ejemplo, en la Escuela Politécnica Nacional (EPN) de Ecuador, se desarrolló un modelo de programación lineal entera para la asignación de horarios de clase. Este modelo permitió abordar diferentes particularidades de la unidad académica, como la existencia de sesiones de clase heterogéneas y la inclusión de disponibilidades y preferencias de horario de los profesores. Los resultados computacionales demostraron que el modelo era capaz de generar horarios eficientes y adaptados a las necesidades específicas de la institución [43].

Estos estudios han demostrado que los AG pueden generar horarios factibles en tiempos significativamente menores comparados con métodos manuales o tradicionales. La PL, cuando se aplica a problemas bien definidos, puede encontrar soluciones óptimas en tiempos computacionales reducidos, especialmente con el uso de software especializado.

e. Conclusión

La integración de algoritmos genéticos y programación lineal proporciona una solución robusta y adaptable para la generación de horarios académicos complejos. Esta combinación permite manejar eficazmente múltiples restricciones, adaptarse a cambios en las condiciones del problema y encontrar soluciones eficientes en tiempos computacionales razonables. Por lo tanto, esta estrategia se alinea perfectamente con los objetivos de tu tesis y ofrece una base sólida para el desarrollo de un sistema de generación de horarios académico eficiente y flexible.

3.3 Desarrollo de la Propuesta

En esta sección se detallan los resultados obtenidos durante el desarrollo del proyecto, enfocado en la generación automatizada de horarios académicos mediante la aplicación de algoritmos genéticos y programación lineal. Se expone la metodología ágil seleccionada para la gestión del proyecto, justificando su elección en función de las necesidades específicas del mismo. Asimismo, se presentan los requerimientos funcionales y no funcionales identificados, esenciales para satisfacer las necesidades de los usuarios y garantizar el cumplimiento de las restricciones del sistema. Se realiza una comparación de las herramientas utilizadas en el desarrollo, evaluando su adecuación y eficiencia. Además, se describe la arquitectura del sistema y el flujo propuesto para la generación de horarios, proporcionando una visión integral del proceso de desarrollo y las decisiones técnicas adoptadas.

3.3.1 Metodología Ágil

Para el desarrollo del proyecto y en base a la revisión bibliográfica realizada [44], la cual enfatiza la relevancia de las metodologías ágiles en la gestión del desarrollo de proyectos de software, este trabajo considera diversas opciones prominentes como SCRUM, Extreme Programming (XP), Rapid Application Development (RAD) y Kanban. Cada una de estas metodologías presenta atributos ventajosos que pueden contribuir significativamente al desarrollo del proyecto. Entre los beneficios inherentes a los enfoques ágiles se destaca la optimización del ciclo de desarrollo y una mayor sinergia entre las necesidades del proyecto y las del cliente.

a. Scrum

Scrum es un marco de trabajo ágil diseñado para ayudar a los equipos a desarrollar, entregar y mantener productos complejos. Se basa en iteraciones llamadas "sprints", que son períodos de tiempo fijos durante los cuales se produce un incremento del

producto. Scrum enfatiza la colaboración, la responsabilidad y el progreso iterativo hacia un objetivo bien definido.

Características adicionales:

- **Transparencia:** Todos los aspectos significativos del proceso deben ser visibles para aquellos responsables del resultado.
- **Inspección:** Los usuarios de Scrum deben inspeccionar con frecuencia los artefactos y el progreso hacia un objetivo para detectar variaciones no deseadas.
- **Adaptación:** Si se detecta que algún aspecto del proceso se desvía fuera de los límites aceptables, el proceso o el material en proceso deben ajustarse.

b. Extreme Programming (XP)

Extreme Programming (XP) es una metodología ágil de desarrollo de software que tiene como objetivo mejorar la calidad del software y la capacidad de respuesta al cambio. XP promueve lanzamientos frecuentes en ciclos de desarrollo cortos, lo que mejora la productividad y presenta puntos de verificación para adoptar nuevos requisitos del cliente.

Características adicionales:

- **Programación en parejas:** Dos programadores trabajan juntos en una sola estación de trabajo, colaborando en el mismo código.
- **Propiedad colectiva del código:** Cualquier miembro del equipo puede cambiar cualquier parte del código en cualquier momento.
- **Integración continua:** El código se integra y prueba varias veces al día, lo que permite detectar errores rápidamente.

c. Rapid Application Development (RAD)

Introducción:

Rapid Application Development (RAD) es una metodología de desarrollo de software que prioriza la rapidez y la flexibilidad mediante el uso de prototipos y la retroalimentación continua del usuario. RAD busca acelerar el proceso de desarrollo sin sacrificar la calidad.

Características adicionales:

- **Prototipado rápido:** Se crean prototipos funcionales rápidamente para validar requisitos y funcionalidades.
- **Iteración y mejora continua:** Los prototipos se refinan a través de múltiples iteraciones basadas en la retroalimentación del usuario.
- **Equipos pequeños y altamente capacitados:** RAD se basa en equipos pequeños de desarrolladores experimentados que pueden tomar decisiones rápidamente.

d. Kanban

Kanban es un método de gestión del trabajo que visualiza el flujo de trabajo y busca mejorar continuamente los procesos. Se originó en el sistema de producción de Toyota y se ha adaptado al desarrollo de software y otros campos.

Características adicionales:

- **Visualización del flujo de trabajo:** Se utilizan tableros Kanban para representar visualmente las tareas y su estado actual.
- **Límites de trabajo en curso (WIP):** Se establecen límites en la cantidad de trabajo que puede estar en progreso para evitar sobrecarga.
- **Gestión basada en datos:** Se utilizan métricas como el tiempo de ciclo y el tiempo de entrega para identificar áreas de mejora.

e. Comparación de metodologías ágiles

Tabla 7. Comparación de Metodologías Ágiles

Característica	Scrum	Extreme Programming (XP)	Rapid Application Development (RAD)	Kanban
Enfoque	Desarrollo iterativo e incremental mediante sprints de duración fija (1 a 4 semanas).	Desarrollo iterativo con énfasis en prácticas técnicas como programación en parejas y pruebas continuas.	Desarrollo rápido mediante prototipos y retroalimentación continua del usuario.	Flujo continuo de trabajo visualizado en un tablero, sin iteraciones fijas.
Duración de Iteraciones	Sprints de 1 a 4 semanas.	Iteraciones cortas de 1 a 2 semanas.	Iteraciones rápidas basadas en prototipos funcionales.	No utiliza iteraciones; el trabajo fluye de manera continua.
Roles Definidos	Product Owner, Scrum Master y Equipo de Desarrollo.	Cliente, Programadores y Coach.	Usuarios finales, Gerentes de Proyecto y Desarrolladores.	No establece roles específicos; se enfoca en la colaboración del equipo.
Planificación	Planificación detallada al inicio de cada sprint.	Planificación continua con enfoque en la retroalimentación del cliente.	Planificación basada en la construcción rápida de prototipos y la retroalimentación del usuario.	Planificación adaptativa basada en la demanda y la capacidad del equipo.
Flexibilidad al Cambio	Cambios permitidos solo entre sprints.	Alta flexibilidad; los cambios pueden incorporarse en cualquier momento.	Alta flexibilidad; se adapta rápidamente a los cambios en los requisitos del usuario.	Muy alta; permite cambios en cualquier momento según la capacidad del equipo.
Énfasis Principal	Entregas incrementales y mejora continua del proceso.	Calidad del código y satisfacción del cliente mediante prácticas técnicas rigurosas.	Rapidez en el desarrollo y satisfacción del usuario mediante prototipos funcionales.	Visualización del flujo de trabajo y mejora continua del proceso.
Adecuado para	Proyectos con requisitos cambiantes que se benefician de entregas regulares y estructura clara.	Proyectos con alta complejidad técnica y necesidad de alta calidad en el código.	Proyectos que requieren desarrollo rápido y donde la retroalimentación del usuario es esencial.	Proyectos con flujo de trabajo continuo y necesidad de adaptarse rápidamente a cambios en la demanda.

f. Selección de metodología

En base a la investigación realizada y el análisis de la Tabla 7, se llegó a la conclusión de que la metodología ágil más adecuada para el desarrollo del proyecto es RAD(Rapid

Application Development). La capacidad de RAD para generar prototipos funcionales con celeridad facilita la obtención temprana de retroalimentación por parte de los usuarios finales, un aspecto crucial en proyectos con requisitos dinámicos, como la generación de horarios académicos. Esta metodología promueve una colaboración continua entre desarrolladores y usuarios, asegurando que el producto final se alinee estrechamente con sus expectativas y necesidades, un factor de éxito primordial en contextos educativos. La alta adaptabilidad de RAD a la evolución de requisitos permite incorporar o modificar funcionalidades sin comprometer significativamente el cronograma, una flexibilidad valiosa en la planificación académica. Finalmente, su enfoque en la reutilización de componentes y la automatización de procesos optimiza la utilización de recursos, contribuyendo a la eficiencia en costos y tiempos de desarrollo.

3.3.2 Selección de Herramientas

a. FrontEnd

Para el desarrollo del frontend del sistema de generación de horarios académicos, es fundamental seleccionar un framework que ofrezca eficiencia, escalabilidad y una experiencia de usuario óptima. La elección adecuada permitirá construir interfaces dinámicas, modulares y de fácil mantenimiento, alineadas con las necesidades específicas del proyecto. A continuación, se presenta una comparación detallada de cuatro frameworks populares: React, Next.js, Astro y Angular.

Tabla 8. Análisis entre React, Next.js, Astro, Angular

Criterio	React	Next.js	Astro	Angular
Tipo	Biblioteca de UI basada en componentes	Framework de React para aplicaciones full-stack	Framework moderno para sitios estáticos y contenido dinámico	Framework completo para aplicaciones SPA
Lenguaje base	JavaScript / JSX	JavaScript / TypeScript	JavaScript / TypeScript	TypeScript
Renderizado	Client-side rendering (CSR)	Soporta SSR, SSG, ISR y CSR	Generación estática por defecto; carga JavaScript solo cuando es	Principalmente CSR; soporte para SSR con Angular Universal

			necesario (arquitectura de islas)	
Routing	Requiere bibliotecas externas (e.g., React Router)	Routing basado en el sistema de archivos	Routing basado en archivos; permite integración con múltiples frameworks	Routing basado en módulos y componentes
Curva de aprendizaje	Moderada	Moderada	Baja a moderada; enfoque en simplicidad y rendimiento	Alta; requiere comprensión profunda de TypeScript y del ecosistema Angular
Optimización SEO	Limitada sin SSR	Alta; ideal para aplicaciones indexables	Muy alta; enfoque en rendimiento y carga rápida	Alta con SSR; requiere configuración adicional
Casos de uso ideales	Interfaces interactivas, dashboards	Aplicaciones web escalables, e-commerce	Sitios de contenido, blogs, documentación técnica	Aplicaciones empresariales complejas, sistemas de gestión
Ecosistema	Amplio; gran cantidad de bibliotecas y herramientas	Amplio; respaldado por Vercel	En crecimiento; enfoque en rendimiento y simplicidad	Muy amplio; respaldado por Google

Según el análisis de la Tabla 8, se ha seleccionado Next.js como framework para el desarrollo del frontend debido a su capacidad de renderizado híbrido (estático y dinámico), lo que mejora significativamente el rendimiento de la aplicación y la experiencia del usuario. Además, al estar basado en React, facilita un desarrollo ágil y modular, con una comunidad activa y herramientas modernas. Su integración nativa con funcionalidades como enrutamiento automático, manejo de API y compatibilidad con TypeScript hacen de Next.js una opción robusta y escalable para aplicaciones web académicas como la del presente proyecto.

b. BackEnd

Para el desarrollo del backend del sistema de generación de horarios académicos, es esencial seleccionar un framework que ofrezca robustez, escalabilidad y una arquitectura mantenible. La elección adecuada permitirá gestionar eficientemente la lógica de negocio, las interacciones con la base de datos y la seguridad del sistema. A continuación, se presenta una comparación detallada de cuatro frameworks populares: Django, Nest.js, Express.js y Spring Boot.

Tabla 9. Análisis entre Django, Nest.js, Express.js, Spring Boot

Criterio	Django	Nest.js	Express.js	Spring Boot
Lenguaje base	Python	TypeScript / JavaScript	JavaScript	Java
Arquitectura	Modelo-Vista-Controlador (MVC)	Modular, inspirado en Angular	Minimalista, basado en middleware	Basado en anotaciones, compatible con microservicios
Rendimiento	Alto, aunque puede ser menos eficiente en aplicaciones en tiempo real debido a su naturaleza sincrónica	Excelente, construido sobre Node.js con soporte para programación asíncrona	Muy alto para aplicaciones I/O intensivas, gracias a su naturaleza asíncrona	Muy alto, especialmente en aplicaciones empresariales de gran escala
Escalabilidad	Alta, adecuada para aplicaciones con gran volumen de datos	Alta, gracias a su arquitectura modular y soporte para microservicios	Moderada, depende de la estructura implementada por el desarrollador	Muy alta, ideal para sistemas empresariales complejos
Curva de aprendizaje	Moderada, facilitada por su documentación extensa y comunidad activa	Moderada, especialmente para desarrolladores familiarizados con Angular	Baja, ideal para principiantes	Alta, requiere conocimientos sólidos de Java y sus ecosistemas
Seguridad	Excelente, incluye protección contra ataques comunes como CSRF, XSS y inyecciones SQL	Buena, con soporte para autenticación y autorización mediante módulos	Básica, depende de middleware y configuraciones adicionales	Muy alta, con soporte integrado para autenticación y autorización a nivel empresarial
Casos de uso ideales	Aplicaciones web completas, sistemas de gestión de contenido, plataformas educativas	Aplicaciones empresariales, APIs RESTful, microservicios	APIs RESTful, aplicaciones en tiempo real, microservicios	Sistemas bancarios, plataformas empresariales, aplicaciones de gran escala
Comunidad y soporte	Muy amplia, con abundante documentación y recursos	En crecimiento, con una comunidad activa y recursos en expansión	Muy amplia, con una gran cantidad de paquetes y middleware disponibles	Muy amplia, respaldada por una comunidad empresarial sólida y extensa documentación

Según el análisis de la Tabla 9, para el backend se optó por **Nest.js** por su arquitectura modular, su fuerte tipado mediante TypeScript y su enfoque orientado a controladores y servicios, que favorece la escalabilidad y el mantenimiento del sistema. Nest.js también facilita la implementación de arquitecturas limpias y la integración con bases

de datos, JWT para autenticación, y librerías para manejo de lógica compleja, lo cual es esencial para gestionar múltiples restricciones y optimizaciones como las requeridas en la generación de horarios académicos.

c. Base de Datos

Para el desarrollo del sistema de generación de horarios académicos, es crucial seleccionar un sistema de gestión de bases de datos (SGBD) que garantice integridad, rendimiento y escalabilidad. La elección adecuada permitirá manejar eficientemente las relaciones entre entidades como carreras, semestres, materias y docentes, asegurando una gestión óptima de los datos. A continuación, se presenta una comparación detallada entre tres SGBD ampliamente utilizados: MySQL, PostgreSQL y MongoDB.

Tabla 10. Análisis entre MySQL, PostgreSQL, MongoDB

Criterio	MySQL	PostgreSQL	MongoDB
Tipo de base de datos	Relacional (RDBMS)	Relacional (ORDBMS)	NoSQL (orientada a documentos)
Modelo de datos	Tablas con filas y columnas	Tablas con filas y columnas	Documentos en formato BSON (similar a JSON)
Esquema	Esquema fijo, aunque permite ciertas modificaciones	Esquema definido, pero admite tipos de datos complejos y JSON	Esquema flexible o sin esquema
Lenguaje de consulta	SQL estándar	SQL avanzado con extensiones (soporte para JSON, XML, procedimientos almacenados)	Lenguaje de consulta propio (MQL)
Cumplimiento ACID	Cumple con ACID, aunque con ciertas limitaciones en transacciones complejas	Cumplimiento completo de ACID, ideal para aplicaciones críticas	Cumple ACID a nivel de documento; soporte para transacciones multi-documento en versiones recientes
Rendimiento	Alto rendimiento en operaciones de lectura; adecuado para aplicaciones web y sistemas de gestión de contenido	Optimizado para consultas complejas y operaciones de escritura intensivas	Excelente rendimiento en manejo de grandes volúmenes de datos no estructurados
Escalabilidad	Escalabilidad vertical; soporte limitado para escalabilidad horizontal	Escalabilidad vertical y horizontal mediante particionamiento y herramientas externas	Escalabilidad horizontal nativa mediante fragmentación (sharding)
Seguridad	Soporte para autenticación y control	Mecanismos avanzados de	Autenticación mediante certificados

	de accesos; cifrado de datos en tránsito y en reposo	seguridad; control de acceso detallado y cifrado	y protocolos como Kerberos y LDAP; cifrado de datos
Casos de uso ideales	Aplicaciones web, sistemas de gestión de contenido, plataformas de comercio electrónico	Aplicaciones empresariales, sistemas financieros, análisis de datos complejos	Aplicaciones que manejan grandes volúmenes de datos no estructurados, análisis en tiempo real, sistemas de recomendación

Según el análisis de la Tabla 10, **MySQL** ha sido elegida como sistema gestor de base de datos por su estabilidad, eficiencia en consultas estructuradas y amplia compatibilidad con herramientas de backend modernas como TypeORM en Nest.js. Es ideal para almacenar datos relacionales como carreras, materias, aulas y restricciones. Además, su soporte transaccional asegura la integridad de los datos durante procesos complejos de planificación y actualización de horarios.

d. Cifrado

Para proteger la confidencialidad de los datos en el sistema de generación de horarios académicos, es necesario emplear técnicas de cifrado robustas. La elección del algoritmo adecuado depende de los requerimientos del sistema, como la velocidad de cifrado, la naturaleza de los datos y la necesidad de intercambio seguro de información. En esta sección se comparan dos de los algoritmos más utilizados: **AES (Advanced Encryption Standard)**, de cifrado simétrico, y **RSA (Rivest–Shamir–Adleman)**, de cifrado asimétrico.

Tabla 11. Análisis entre AES, RSA

Criterio	**AES**	**RSA**
Tipo de cifrado	Simétrico (misma clave para cifrar y descifrar)	Asimétrico (clave pública para cifrar, privada para descifrar)
Velocidad de procesamiento	Alta, ideal para cifrar grandes volúmenes de datos	Lento, no adecuado para grandes volúmenes de datos
Seguridad	Muy seguro con claves de 128, 192 o 256 bits	Seguro, pero requiere claves más largas (1024+ bits)
Uso común	Cifrado de archivos, bases de datos, conexiones TLS	Intercambio de claves, firmas digitales, autenticación
Consumo de recursos	Bajo	Alto, especialmente en dispositivos con recursos limitados
Aplicabilidad	Ideal para proteger datos en reposo y en tránsito	Ideal para intercambio seguro de claves y autenticación

Según el análisis de la Tabla 11, para garantizar la seguridad de los datos, especialmente en procesos de autenticación y manejo de información sensible, se utilizará el algoritmo **AES (Advanced Encryption Standard)**. AES es un estándar ampliamente aceptado por su velocidad, nivel de seguridad y eficiencia en entornos web. Al ser un sistema de cifrado simétrico, permite encriptar y desencriptar datos de manera rápida sin comprometer la seguridad del sistema.

e. Control de Versiones

El control de versiones es una práctica fundamental en el desarrollo de software moderno, ya que permite gestionar los cambios en el código fuente a lo largo del tiempo, facilitando la colaboración entre desarrolladores, el seguimiento de errores y la recuperación de versiones anteriores. Para este proyecto, se evalúan las herramientas más reconocidas en esta área: **Git**, **Subversion (SVN)** y **Mercurial**, cada una con características particulares en cuanto a distribución, facilidad de uso y soporte.

Tabla 12. Análisis entre Git, Subversion y Mercurial

Criterio	**Git**	**Subversion (SVN)**	**Mercurial**
Tipo de control	Distribuido	Centralizado	Distribuido
Velocidad	Alta en operaciones locales	Depende del servidor, generalmente más lenta	Alta, similar a Git
Facilidad de uso	Requiere curva de aprendizaje	Más fácil de aprender para usuarios nuevos	Interfaz simple, comandos consistentes
Popularidad	Muy alta, estándar de facto en la industria	Uso en proyectos más antiguos o conservadores	Menor popularidad
Soporte de ramas	Robusto, ramas livianas y fáciles de manejar	Soporte limitado, ramas son más pesadas	Buen manejo de ramas
Comunidad y soporte	Amplia comunidad, gran documentación y herramientas	Comunidad más limitada, menos innovaciones recientes	Comunidad activa pero más pequeña que Git
Herramientas asociadas	GitHub, GitLab, Bitbucket	Apache Subversion, TortoiseSVN	Bitbucket, plugins para IDEs
Repositorio offline	Sí	No, requiere conexión al servidor	Sí

Según el análisis de la Tabla 12. El sistema de control de versiones seleccionado es **Git**, por ser el más utilizado y soportado actualmente en entornos profesionales. Su enfoque distribuido permite tener copias completas del repositorio local, lo que facilita el trabajo colaborativo, la trazabilidad de cambios y la integración con plataformas

como GitHub o GitLab. Esto permite mantener una gestión eficiente del código fuente durante el ciclo de desarrollo del sistema.

La elección de las herramientas tecnológicas para el desarrollo de esta investigación se fundamenta en sus características técnicas y capacidades de optimización para la gestión académica. En el backend, Nest.js ha sido seleccionado debido a su arquitectura modular basada en TypeScript, que facilita el desarrollo de aplicaciones escalables y bien estructuradas, además de ofrecer soporte robusto para la construcción de API RESTful necesarias para la integración de los algoritmos de optimización. Para el frontend, Next.js se ha elegido por su capacidad para renderizar tanto del lado del cliente como del servidor, mejorando la velocidad y experiencia de usuario en la interacción con la aplicación. Además, su compatibilidad con React asegura un desarrollo de interfaz eficiente y flexible, ideal para la visualización y manejo de datos académicos. Por último, MySQL se ha seleccionado como sistema de gestión de bases de datos por su estabilidad, eficiencia en la manipulación de grandes volúmenes de datos y su capacidad para realizar consultas complejas, lo que es esencial en la organización de información relacionada con aulas, horarios y restricciones de recursos. Esta combinación de herramientas permitirá construir una aplicación optimizada, segura y escalable, adecuada para las necesidades de la institución educativa.

3.3.3 Fase 1: Planificación de requerimientos

Como primera etapa del desarrollo del sistema propuesto, se llevó a cabo el levantamiento de requerimientos funcionales, no funcionales y restricciones del sistema, siguiendo los lineamientos establecidos por la norma IEEE 830. Esta fase corresponde a la **etapa de "Definición de Requisitos"** dentro de la metodología RAD, en la cual se identifican los objetivos del sistema y se definen sus características generales mediante una comunicación directa con los usuarios clave. En este caso, se aplicó un cuestionario estructurado, presentado en el **Anexo E**, que permitió recoger de forma precisa las necesidades y expectativas del personal responsable de la gestión académica.

Posteriormente, los requerimientos fueron documentados en el **Anexo F**, utilizando las normas sugeridas por IEEE 830. Cada requerimiento fue clasificado con su respectivo identificador, nombre, características, descripción, requerimientos no funcionales relacionados, prioridad, criterios de aceptación y observaciones relevantes. Esta documentación estructurada constituye la base fundamental para las siguientes fases de desarrollo del sistema, asegurando una adecuada trazabilidad y alineación con las metas institucionales.

Con base en el levantamiento de requerimientos realizado durante la fase de "Definición de Requisitos" de la metodología RAD y documentado conforme a la norma IEEE 830 en el **Anexo F**, se procedió a organizar los requerimientos del sistema en función de iteraciones de desarrollo. Esta estrategia permite priorizar y distribuir progresivamente las funcionalidades en ciclos cortos e incrementales, favoreciendo una construcción ágil, validación temprana y retroalimentación continua por parte de los usuarios.

Tabla 13. Planificación de requerimientos del sistema por iteraciones

Iteración	**N.**	**ID**	**Requerimiento**	**Tiempo estimado**	
				Horas	**Días**
1	1	RF01	Gestión de Carreras (CRUD)	8	1
	2	RF02	Gestión de Semestres (CRUD)	8	1
	3	RF03	Gestión de Materias (CRUD)	12	1.5
	4	RF04	Gestión de Períodos Académicos (CRUD)	12	1.5
	5	RF05	Gestión de Edificios (CRUD)	8	1
	6	RF06	Gestión de Pisos (CRUD)	8	1
	7	RF07	Gestión de Laboratoristas (CRUD)	8	1
	8	RF08	Gestión de Docentes (CRUD)	10	1.25
2	9	RF09	Gestión de Software (CRUD)	8	1

	10	RF10	Gestión de Aulas (CRUD)	8	1
	11	RF11	Gestión de Máquinas (CRUD)	10	1.25
	12	RF12	Gestión de Horarios (CRUD)	16	2
	13	RF13	Gestión de Usuarios (CRUD)	12	1.5
3	14	RF14	Generación de Hojas de Control	8	1
	15	RF01-H	Restricciones para generación de horarios	40	5
	16	RF02-H	Generación de horarios mediante IA	40	5
	17	**RF15**	**Módulo de Reservas**	**¿?**	**¿?**

3.3.4 Fase 2: Diseño de usuario

a. Análisis del proceso propuesto de creación de horarios

El proceso propuesto para la generación automática de horarios académicos se desarrolla de forma estructurada mediante una secuencia de actividades automatizadas y validadas, con el objetivo de garantizar eficiencia, adaptabilidad y precisión. A continuación, se describe cada etapa del flujo:

1. **Recepción de distributivos y restricciones de docentes**
    - El encargado de laboratorio recibe los distributivos y restricciones de disponibilidad de los docentes, lo que permite conocer sus cargas y limitaciones para la planificación académica.
2. **Registro de estructura e infraestructura académica**
    - En el sistema de horarios se realiza el registro de la estructura académica (carreras, semestres, materias, paralelos, etc.) y de la infraestructura (aulas, laboratorios, edificios, software, etc.).

3. **Creación del periodo académico**

    - Se define un nuevo periodo académico, estableciendo fechas de inicio y fin, así como el estado del periodo (activo/inactivo).

4. **Decisión: ¿Generar horarios académicos con IA?**

    - El sistema permite elegir entre generar los horarios automáticamente mediante el algoritmo de optimización o crearlos de forma manual.

5. **Generación automática de horarios (en caso afirmativo)**

    - Se registra la asignación académica (materia-docente), las restricciones de los docentes, y se selecciona el periodo académico.

    - El sistema ejecuta el algoritmo de optimización y genera los horarios tomando en cuenta todas las restricciones y recursos disponibles.

6. **Generación manual de horarios (en caso negativo)**

    - El usuario crea los horarios de manera manual dentro del sistema, según la disponibilidad y asignaciones previamente registradas.

7. **Creación de horarios**

    - Ya sea de forma automática o manual, se realiza la generación de los horarios académicos dentro del sistema.

8. **Envío de horarios a los coordinadores de carrera**

    - Una vez generados, los horarios son enviados a los coordinadores de carrera para su revisión.

9. **Revisión de horarios por parte del coordinador**

    - El coordinador de carrera evalúa los horarios propuestos. Si detecta inconsistencias o mejoras, se solicitan los cambios necesarios.

10. **Realización de cambios en los horarios**

- El encargado de laboratorio, en conjunto con el sistema, realiza los ajustes solicitados por los coordinadores.

11. Aprobación de horarios

- Una vez que no existen más observaciones, los horarios son aprobados oficialmente.

12. Publicación y difusión de horarios

- Los horarios aprobados se publican en la página web de la FISEI y se difunden a través de los medios oficiales de comunicación institucional.

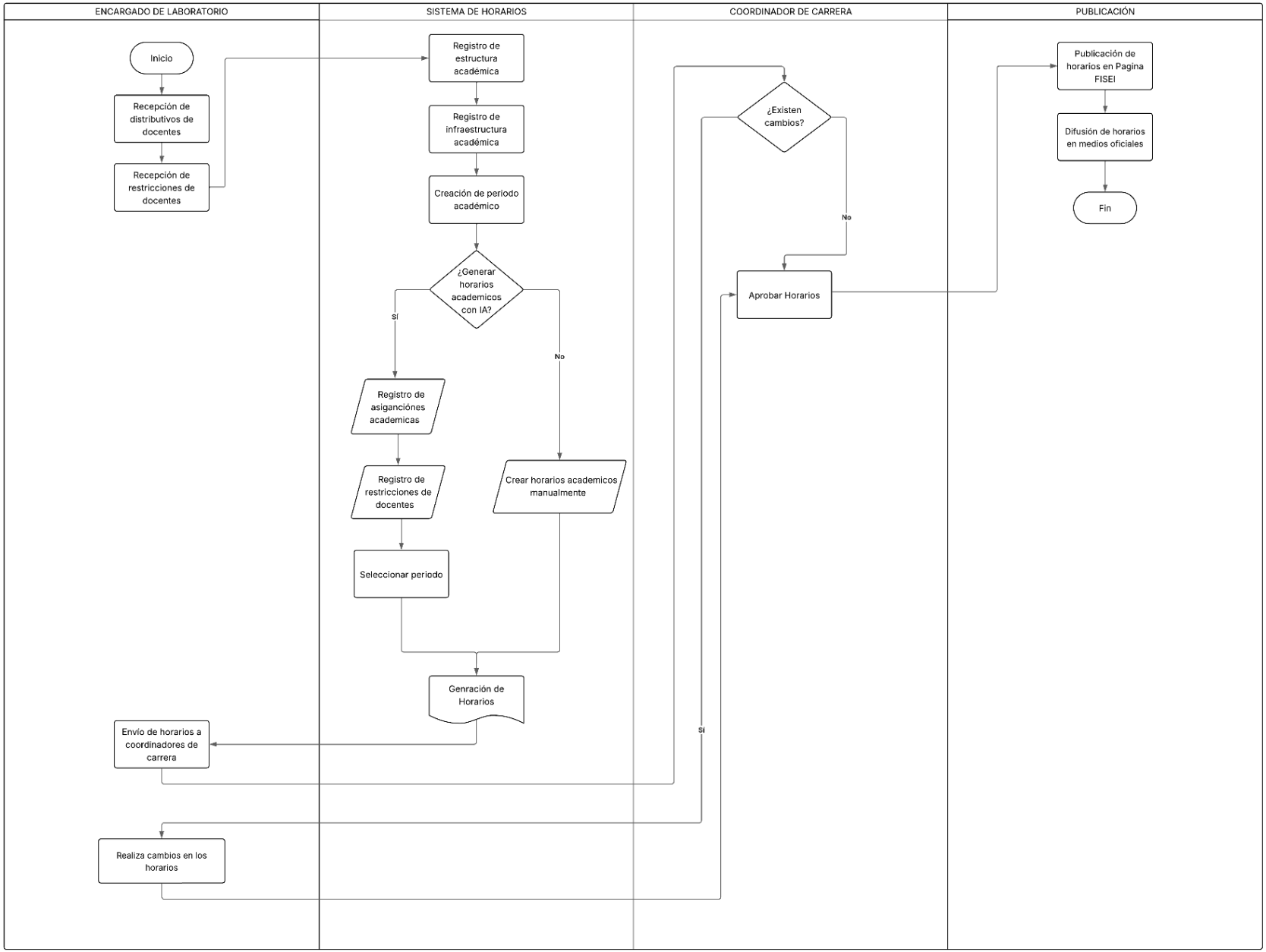


Figura 7. Diagrama de flujo del proceso de creación de horarios propuesto

b. Análisis del proceso propuesto de generación de hojas de control

Este es el proceso propuesto para la generación de hojas de control

1. **Inicio del proceso**
    - El encargado de laboratorio inicia el procedimiento decidiendo si desea generar hojas de control **por bloque** o de forma **individual**.
2. **Generación por bloque**
    - Si se elige la opción por bloque:
        - Se accede al formulario correspondiente.
        - Se selecciona el bloque (edificio) para el cual se generarán las hojas.
        - Se escoge la fecha específica del día.
        - Se asignan los encargados de laboratorio de la jornada.
        - Se indica la jornada (mañana o tarde).
        - Finalmente, se genera la hoja de control para todo el bloque seleccionado.
3. **Generación individual**
    - Si se selecciona la opción individual, el sistema solicita definir el **tipo de hoja:**
    - **Hoja Libre:**
        - Seleccionar la carrera, semestre, materia y paralelo.
        - Elegir el docente, las horas de jornada y el laboratorio.
        - Asignar al encargado de laboratorio.
    - **Hoja de Control:**

- Seleccionar directamente al docente.
- Asignar el encargado de laboratorio.
- Seleccionar la fecha y la materia correspondiente.

4. **Generación y entrega**

- Una vez completada la información requerida en cualquiera de las opciones, el sistema procede a **generar la hoja o las hojas de control.**
- Las hojas generadas son **entregadas al docente encargado**, quien las recibe como comprobante de uso y control del laboratorio.
- El proceso finaliza tras la entrega.

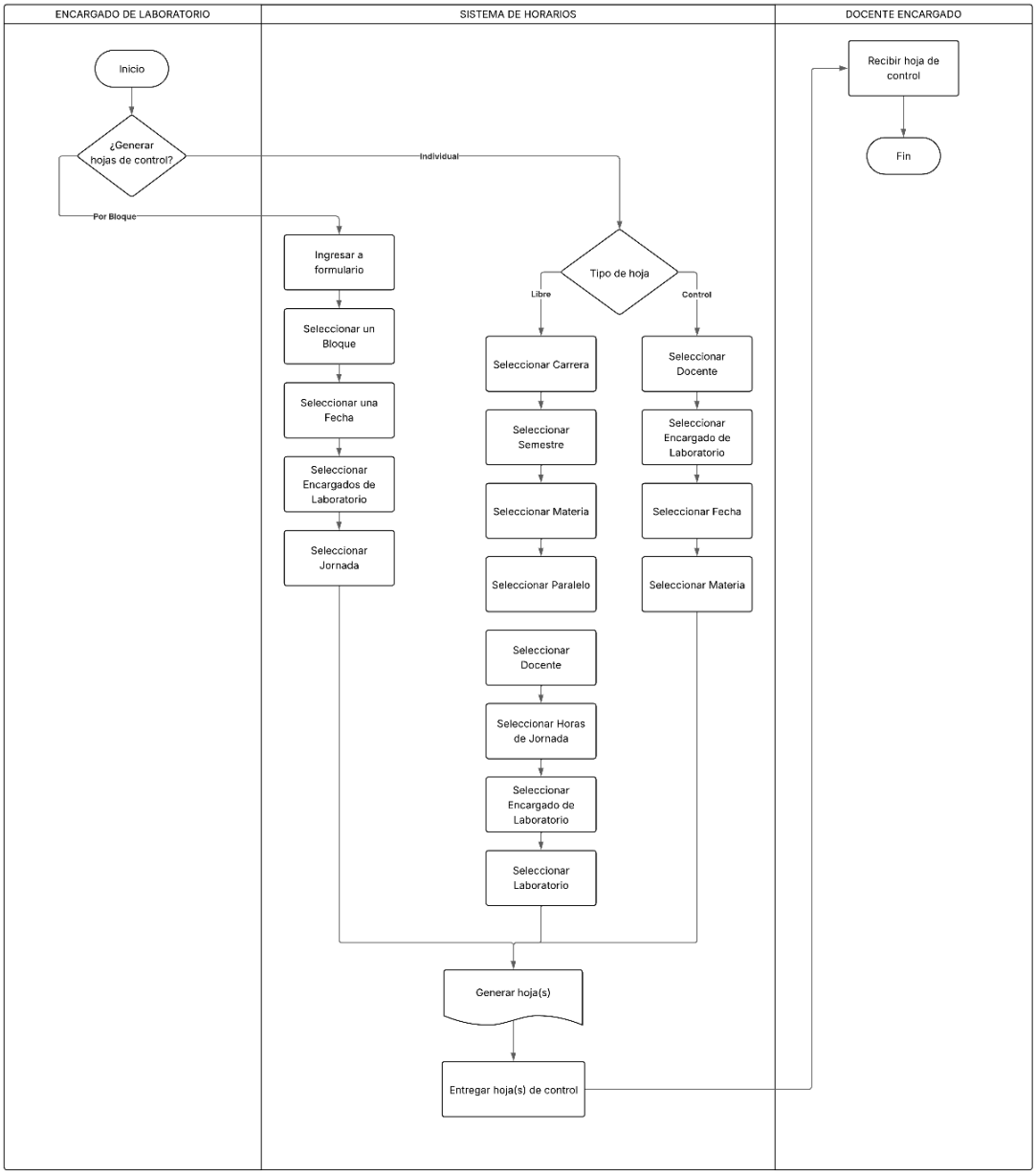


Figura 8. Diagrama de flujo del proceso propuesto de generación de hojas de control

c. Análisis del proceso propuesto de reservas de aulas y laboratorios

1. **Solicitud inicial por parte del usuario**

- El proceso inicia cuando un usuario necesita realizar una reserva.
- El usuario se **acerca al departamento de administración de redes** para solicitar la reserva de un aula o laboratorio.
- Realiza una **consulta o petición de información** sobre la disponibilidad y procedimiento.

2. **Verificación de disponibilidad**

- El **encargado de laboratorio verifica la información** general sobre disponibilidad en el sistema.
- En el sistema de horarios:
    - Se **selecciona el bloque (edificio).**
    - Se **define el tipo de aula** (aula normal o laboratorio).
    - Se **selecciona el aula o laboratorio específico** para revisar su disponibilidad.

3. **Validación de horarios**

- El encargado de laboratorio revisa el horario existente del aula/laboratorio seleccionado.
- Si se encuentra disponibilidad, se procede a solicitar los datos necesarios al usuario (como nombre, tipo de actividad, horario deseado, entre otros).

4. **Registro de la reserva**

- El usuario **proporciona los datos requeridos** al encargado.
- El encargado **ingresa los datos** en el sistema.
- En el sistema se registra:
    - La **información del usuario** solicitante.
    - La **información de la reserva**, como fecha, hora y aula o laboratorio reservado.

5. **Finalización del proceso**

- El proceso finaliza una vez completado el **registro de la reserva**, quedando almacenada y visible en el sistema para futuras consultas o supervisión.

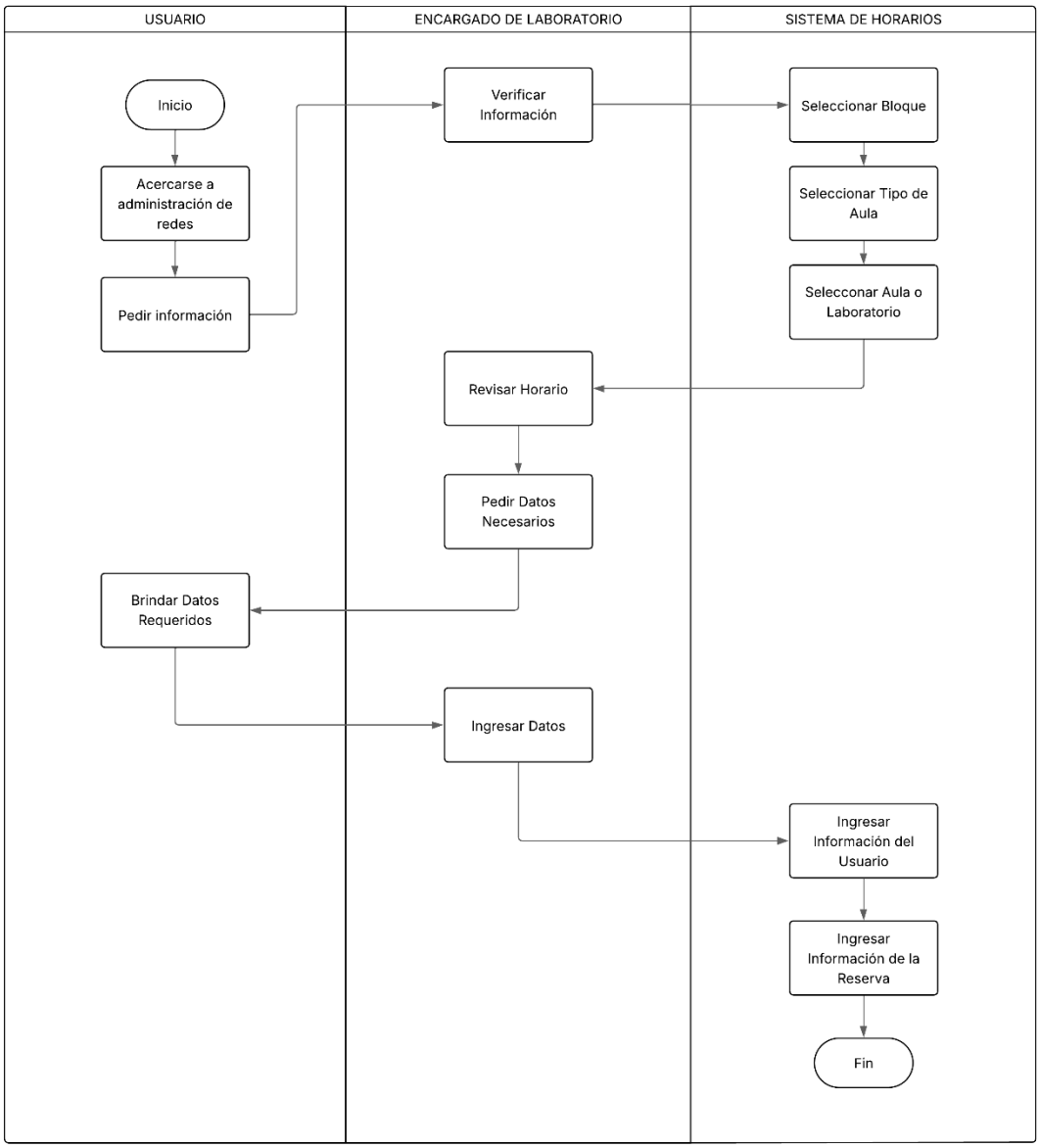


Figura 9. Diagrama de flujo del proceso propuesto de reservar aulas o laboratorios

d. Arquitectura

La Figura 10 desarrollada para la gestión y optimización de horarios académicos, estructurada en tres capas fundamentales: presentación, negocio y datos. En la **capa de presentación** se encuentra el frontend desarrollado con Next.js, el cual permite la interacción directa con el usuario mediante una interfaz web moderna y dinámica, enviando peticiones y recibiendo respuestas desde los servicios del sistema. La **capa de negocio** está compuesta por dos backends independientes: uno principal desarrollado en Nest.js, encargado de la administración central de los datos

académicos, como horarios, docentes y espacios, permitiendo organizarlos por semestres y asegurar un seguimiento eficiente de los recursos educativos; y un segundo backend desarrollado en Python, encargado de ejecutar los algoritmos de optimización que procesan la información académica, ajustando las asignaciones de horarios de acuerdo con los requerimientos específicos de la estructura curricular, las disponibilidades del cuerpo docente y las condiciones propias de la infraestructura académica. Ambos backends se comunican entre sí para garantizar una operación coordinada y coherente. Finalmente, en la **capa de datos** se encuentra una base de datos MySQL, que centraliza el almacenamiento de la información, permitiendo consultas y actualizaciones por parte de ambos servidores. Esta arquitectura modular y bien definida permite una separación clara de responsabilidades, facilitando la escalabilidad, el mantenimiento y una comunicación eficiente entre sus componentes, con el fin de ofrecer una solución integral para la asignación automatizada de recursos en entornos académicos.

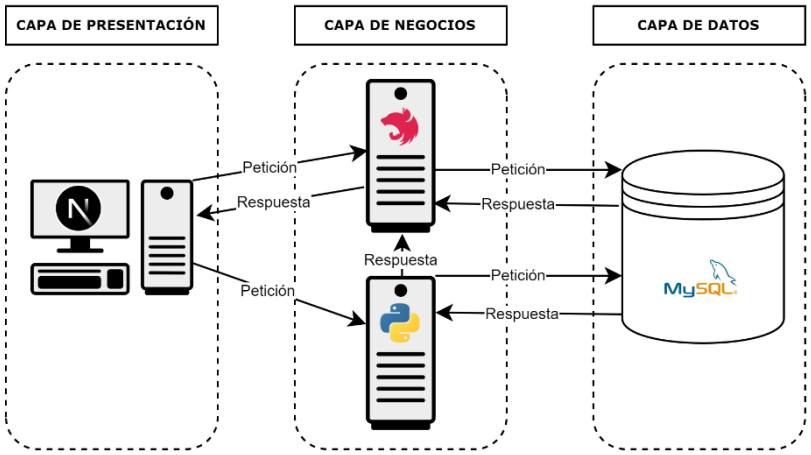


Figura 10: Arquitectura de la aplicación

e. Prototipado

Como parte de la metodología RAD, la fase de **Diseño del Usuario** incluye la creación de prototipos que permiten validar tempranamente la estructura, posible interfaz y funcionalidades del sistema. Esta etapa es fundamental, ya que facilita una comunicación constante entre los desarrolladores y los usuarios finales, permitiendo recibir retroalimentación continua y realizar ajustes antes del desarrollo definitivo. En

el contexto de este proyecto, se desarrollaron prototipos de las principales interfaces del sistema de gestión académica, con el objetivo de representar visualmente cómo se organizarán los módulos, qué datos se presentarán y cómo será la experiencia de uso. Esto asegura que el diseño final del sistema responda de forma efectiva a las necesidades detectadas durante el levantamiento de requerimientos. Para realizar este prototipado, se tomó en cuenta la paleta de colores representativa de la facultad.

En la Figura 11 se presenta el prototipo de inicio de sesión, la cual está diseñada con el logotipo institucional en la parte superior. El prototipo incluye dos campos principales: uno para ingresar el correo electrónico y otro para la contraseña. Debajo de estos campos se ubica un botón de color rojo que permite al usuario acceder al sistema una vez que los datos hayan sido ingresados correctamente.

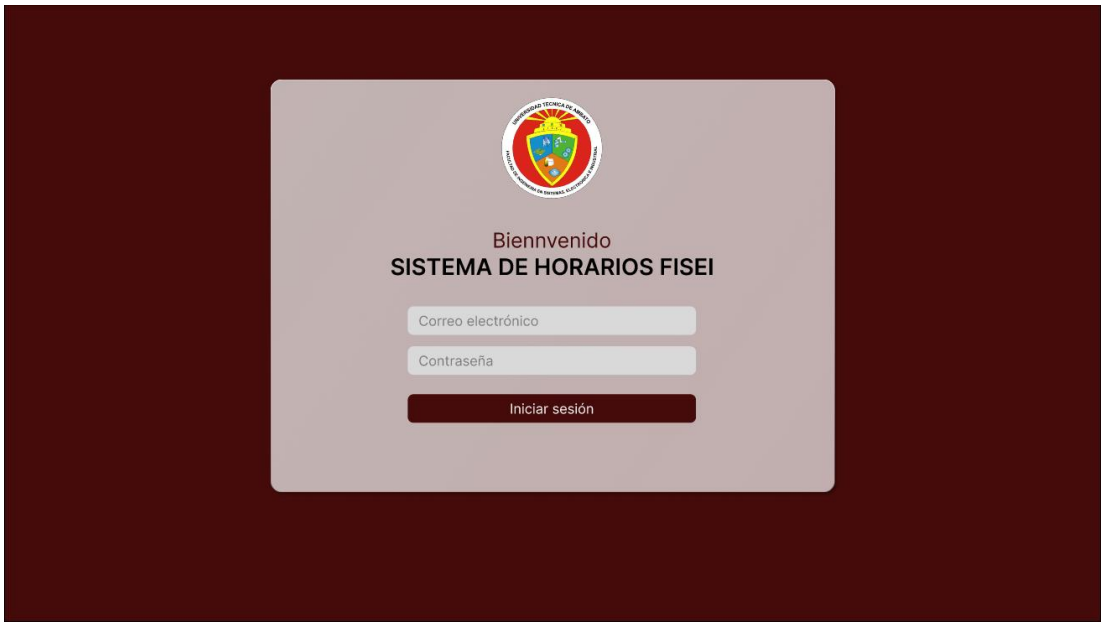


Figura 11. Prototipo de inicio de sesión

En la Figura 12 se muestra el prototipo del panel de navegación de la aplicación, el cual presenta una disposición estructurada por módulos. Cada módulo agrupa categorías y páginas relacionadas que facilitan el acceso organizado a las distintas funcionalidades del sistema. Cada categoría cuenta con un menú desplegable que, al interactuar con él, revela las distintas páginas o funcionalidades asociadas. Esto permite una navegación más limpia y jerárquica, evitando la sobrecarga visual y facilitando el acceso a las opciones específicas dentro de cada grupo funcional. Esta estructura modular permite al usuario navegar de forma intuitiva por las diferentes

secciones, como la gestión académica, consultas de horarios, administración de recursos o generación de reportes. Gracias a esta organización, se mejora la usabilidad del sistema y se optimiza el flujo de trabajo, permitiendo que los usuarios accedan rápidamente a las herramientas necesarias.

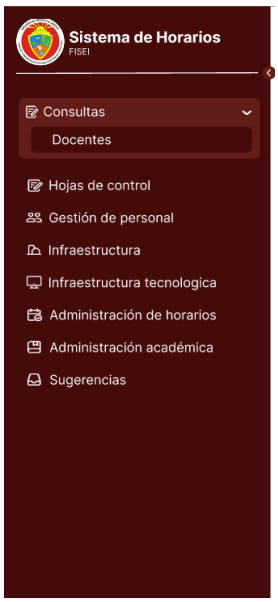


Figura 12. Prototipo de panel de navegación

En la Figura 13 se presenta el prototipo de la pantalla principal del sistema, la cual está diseñada para ofrecer una vista general y accesible de los datos registrados. En la parte inferior izquierda se incluyen accesos rápidos, entre los cuales destacan la generación de hojas de control (individuales o por bloques), la consulta rápida de horarios de docentes y de aulas/laboratorios. En el área inferior central se encuentra un calendario interactivo que permite visualizar las fechas relevantes, mientras que en la parte inferior derecha se visualiza un panel de actividades programadas, el cual incluye en su parte superior una barra de búsqueda que permite filtrar eficientemente los registros por nombre de docente, materia u otros campos relevantes, ya sea para el día actual o para la fecha seleccionada en el calendario. Este diseño busca mejorar la eficiencia operativa y facilitar la navegación del usuario dentro del sistema.



Figura 13. Prototipo de panel de control

En la Figura 14 se presenta el prototipo de la pantalla destinada a la consulta de horarios, tanto de docentes como de aulas y laboratorios. En el panel izquierdo se visualiza el listado completo de docentes o espacios registrados, acompañado de una barra de búsqueda que facilita el filtrado por nombre. En el panel derecho se muestra el horario correspondiente al elemento seleccionado, representado mediante tarjetas informativas que detallan el nombre de la actividad, el lugar asignado, la hora de inicio y fin, el docente responsable y una breve descripción de la actividad. Cada tarjeta utiliza un código de colores para identificar el estado temporal de la actividad: gris para actividades pasadas, rojo para las que están en curso y verde para las próximas.

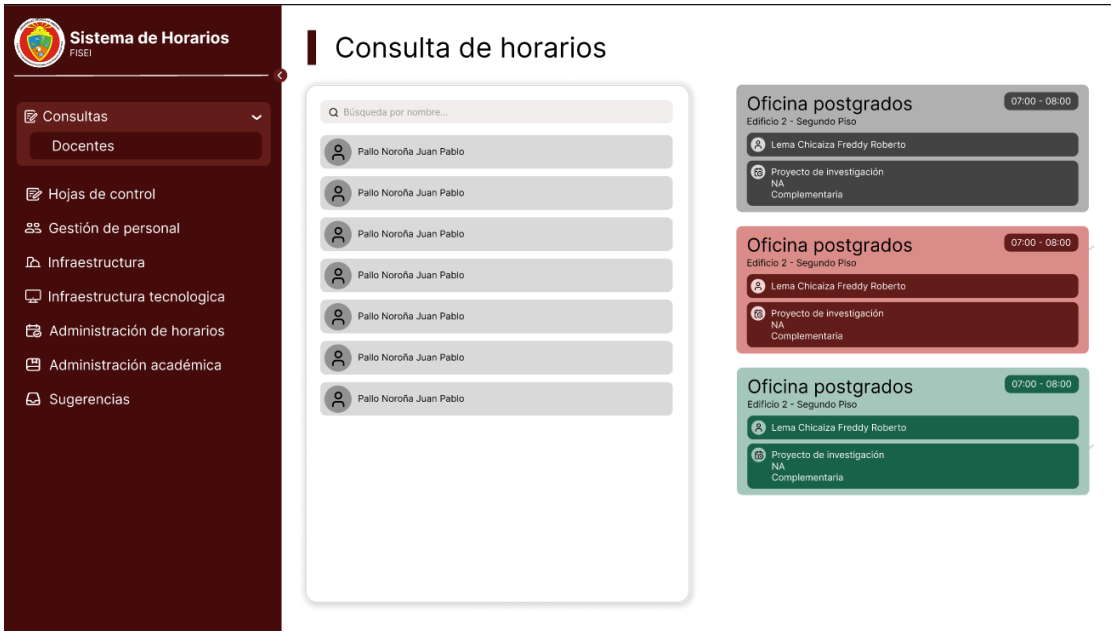


Figura 14. Prototipo de consulta de horarios por docentes

En la Figura 15 se presenta el prototipo del formulario para la generación de una hoja de control. Este formulario permite al usuario ingresar toda la información necesaria, incluyendo la carrera, semestre, paralelo, docente, materia, hora de inicio y fin, laboratorio asignado y el laboratorista responsable. Una vez completados todos los campos, el usuario puede proceder con la generación de la hoja de control haciendo clic en el botón rojo ubicado en la parte inferior del formulario.

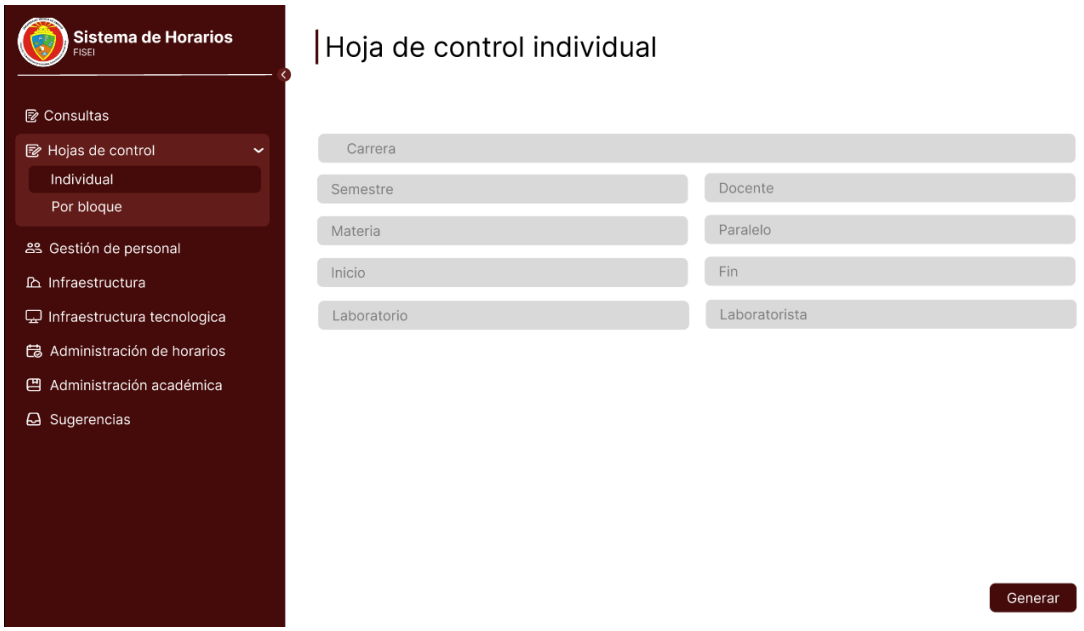


Figura 15. Prototipo de formulario para la generación de hojas de control individual

En la Figura 16 se presenta el formulario para la generación de hojas de control por bloque (edificio). Esta interfaz permite al usuario seleccionar el bloque correspondiente, especificar la fecha para la cual se desea generar las hojas de control, así como asignar al encargado de laboratorio tanto para la jornada matutina como vespertina. Además, se puede definir el rango horario mediante la selección de la hora de inicio y la hora de finalización del periodo a cubrir, permitiendo así generar automáticamente las hojas de control dentro del intervalo establecido.

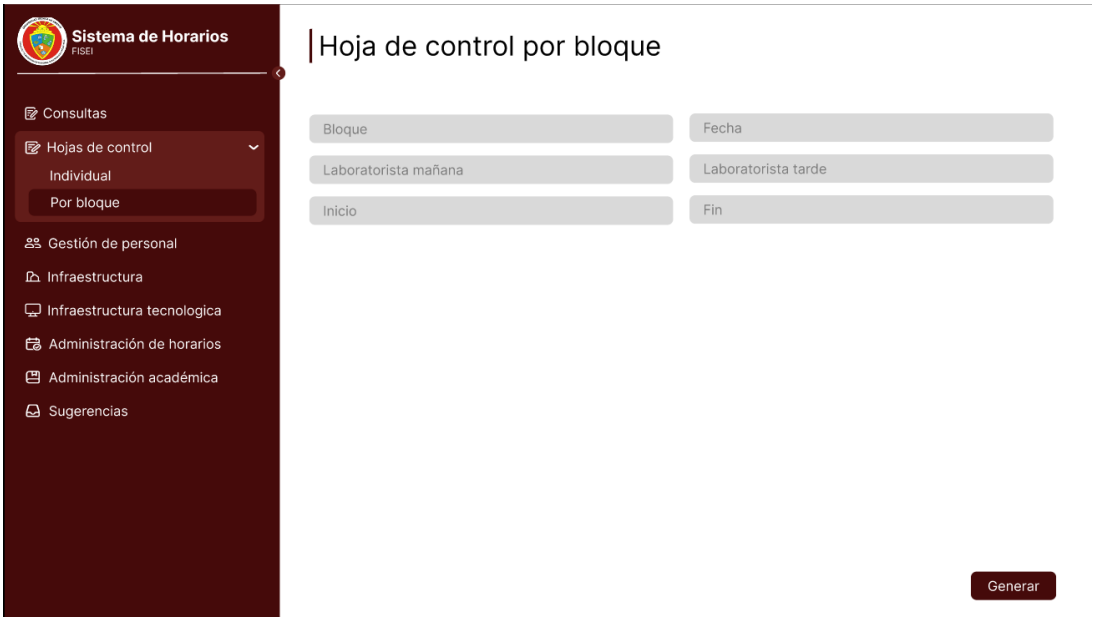


Figura 16. Prototipo de formulario para la generación de hojas de control por bloque

Para la gestión de datos registrados, como se observa en la Figura 17 , el prototipo muestra una estructura la cual se conforma de una tabla donde el usuario puede visualizar de manera ordenada toda la información relevante de cada registro. Cada fila incluye opciones para modificar o eliminar el registro correspondiente, mediante botones con iconos representativos: un lápiz para editar y un basurero para eliminar. En la parte superior de la tabla se dispone de una barra de búsqueda que permite filtrar los registros de forma eficiente según cualquier campo. Además, se incorpora un botón de color rojo que permite agregar un nuevo registro; al hacer clic en él, el sistema redirecciona al formulario correspondiente, como se muestra en la Figura 18.

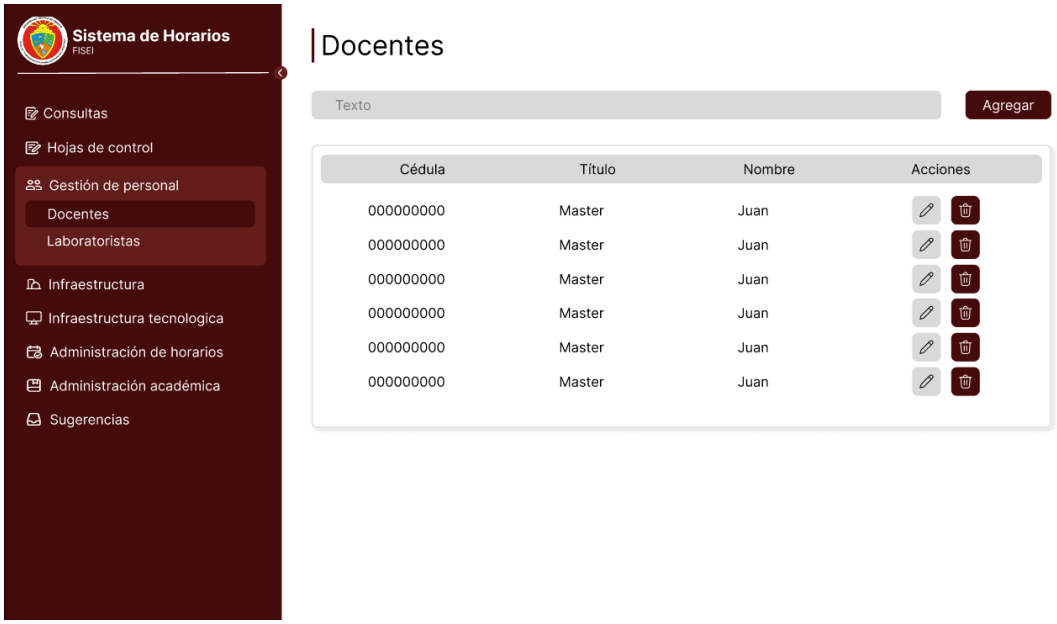


Figura 17. Prototipo de interfaz para la gestión de datos almacenados

En la Figura 18 se muestra el formulario de registro de datos. Este prototipo muestra cómo se permitirá al usuario ingresar la información necesaria para la creación de un nuevo registro, los campos del formulario cambiaran según el registro que se desee agregar o modificar. Los campos están organizados de forma clara para facilitar su uso. En la parte inferior derecha del formulario se encuentran los botones de acción: el botón “Cancelar”, de color gris, permite descartar la operación, mientras que el botón “Agregar”, de color rojo, de confirma el registro del nuevo docente en el sistema. El nombre del botón puede ser “Actualizar” si se está modificando un registro existente.

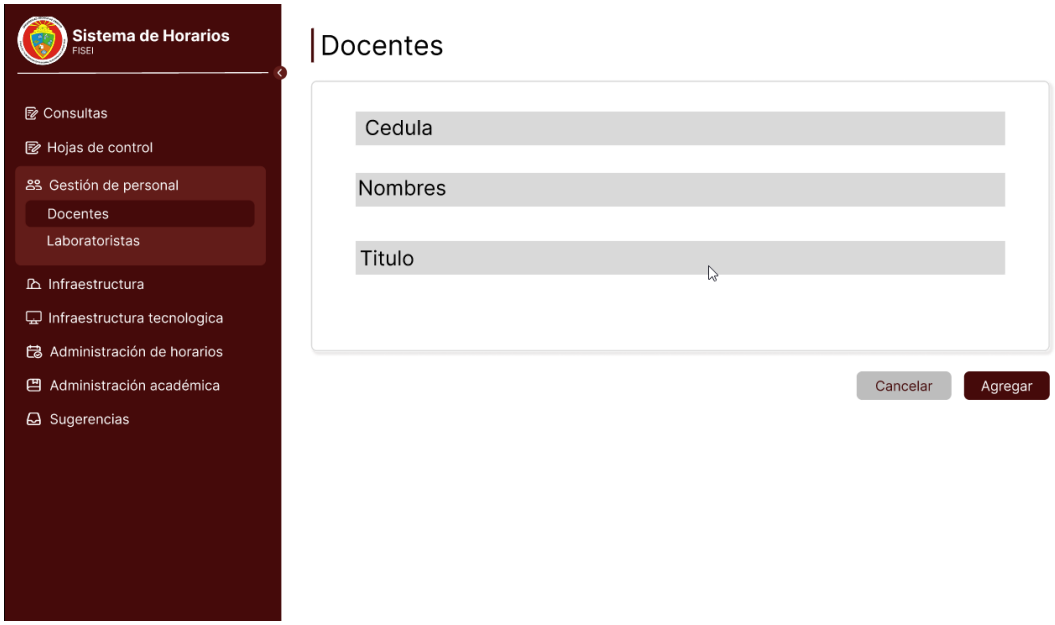


Figura 18. Prototipo de formulario de registro de datos

En la Figura 19 se muestra el prototipo de la pantalla de reservas; En prototipo se muestra la tabla de horarios correspondiente al aula o laboratorio seleccionado, que abarca desde el día lunes hasta el día viernes, con las horas listadas; cada celda estará identificada con un color dependiendo de su estado, si el aula está libre, si está ocupada por alguna actividad académica o si está reservada.

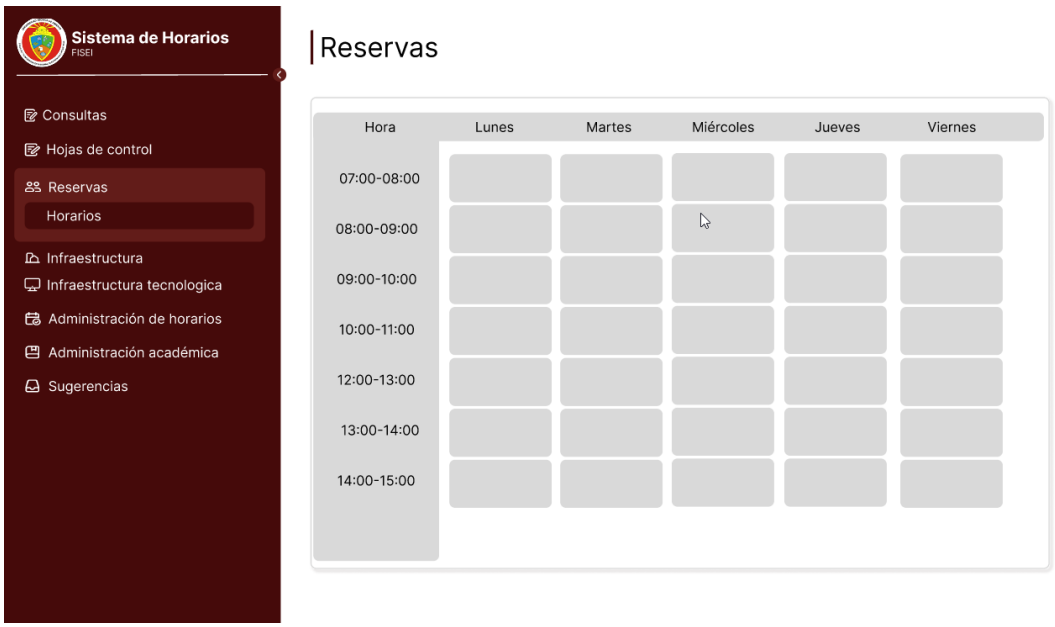


Figura 19. Prototipo de reserva de aulas y laboratorios

La Figura 20 representa el prototipo destinado a la visualización de los horarios generados mediante los algoritmos implementados, considerando todas las restricciones y requerimientos definidos en el sistema. En la parte superior se encuentra un selector que permite elegir el período académico para el cual se desea generar el horario; posteriormente, se debe hacer clic en el botón rojo “GENERAR” para iniciar el proceso. Una vez completada la generación, los horarios se mostrarán en una tabla estructurada por aulas y laboratorios, con los días de la semana y las horas correspondientes. Cada celda indicará la actividad programada, la cual estará identificada con un color único y una breve descripción de la misma.

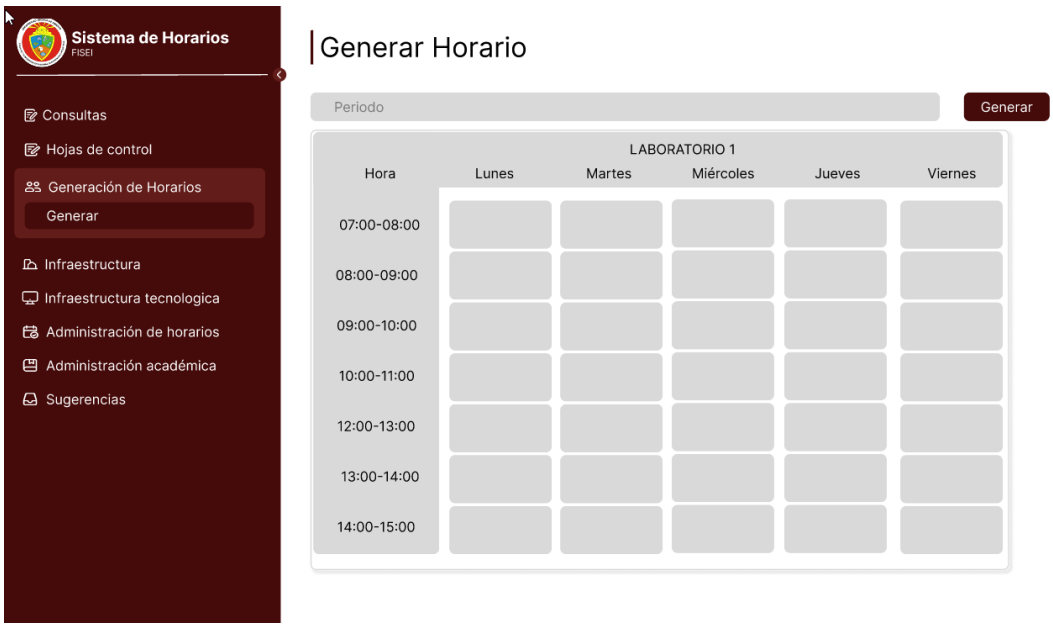


Figura 20. Prototipo de generación de horarios

3.3.5 Fase 3: Construcción

a. Diseño Conceptual de la Base de Datos

Como parte del proceso de construcción del sistema, se desarrolló el diseño conceptual de la base de datos con el objetivo de representar de manera clara y estructurada las entidades y relaciones fundamentales del dominio académico. Este diseño se enfocó en capturar los principales componentes del sistema, incluyendo la gestión de horarios,

reservas de aulas, restricciones de docentes y generación automatizada de horarios, reflejando así la lógica de negocio desde una perspectiva semántica.

El modelo conceptual se organizó en distintas secciones funcionales: **Reservas**, **Horarios**, **Restricciones** y **Generación**, lo que permitió agrupar las entidades según el contexto operativo del sistema. Se definieron entidades clave como *Docentes*, *Materias*, *Aulas*, *Paralelos*, *Horarios*, *Restricciones por Aula*, *Reservas*, entre otras, así como sus atributos principales y relaciones entre ellas. Las asociaciones fueron modeladas tomando en cuenta la cardinalidad y opcionalidad de cada relación, asegurando una correcta interpretación del comportamiento del sistema.

Este diseño conceptual fue elaborado como base inicial para el posterior diseño lógico e implementación en un gestor de base de datos relacional (MySQL). La estructura resultante facilita la integración con los módulos desarrollados en los backend de NestJS y Python, y proporciona una visión clara y comprensible tanto para el equipo de desarrollo como para los usuarios técnicos involucrados en el proyecto.

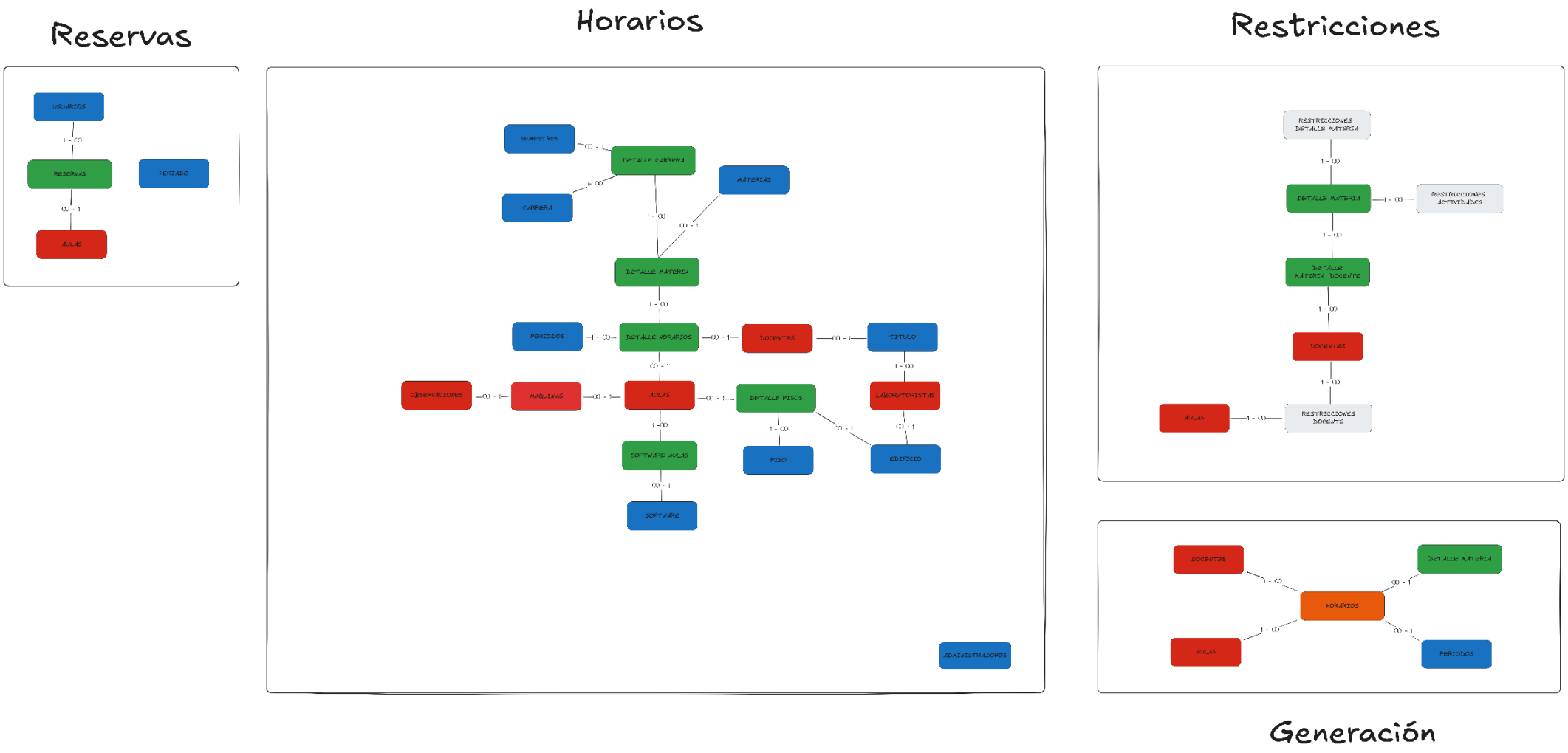


Figura 21. Diseño Conceptual de la Base de Datos

b. Backend

Para el desarrollo del backend se utilizó el framework **NestJS**, una plataforma robusta basada en Node.js y TypeScript que permite construir aplicaciones escalables mediante una arquitectura modular. Este enfoque modular permitió organizar el sistema en diferentes **módulos funcionales**, como usuarios, carreras, docentes, aulas, materias, horarios, entre otros. Cada módulo contiene su propio controlador, servicio, entidades y rutas, lo que facilita la **separación de responsabilidades**, la reutilización de código y el mantenimiento del sistema.

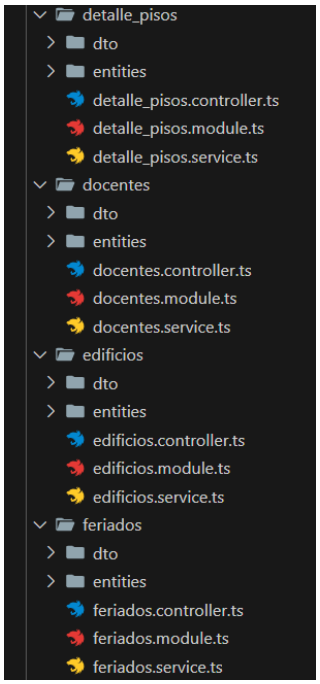


Figura 22. Arquitectura proyecto Backend

En cuanto a la persistencia de datos, se empleó el ORM **TypeORM**, que facilita la interacción con la base de datos relacional **MySQL**. Con TypeORM, se definieron las entidades correspondientes a cada tabla de la base de datos, asegurando que la estructura del modelo de datos se mantenga sincronizada con el diseño lógico y físico. Además, se aprovecharon sus funcionalidades para la creación automática de migraciones, validación de esquemas y relaciones entre entidades (uno a uno, uno a muchos, muchos a muchos), lo cual es fundamental en un sistema con una estructura académica compleja.

Se implementaron **servicios reutilizables** para operaciones comunes (CRUD) en cada módulo, garantizando eficiencia en las consultas y la lógica del negocio. El backend también incluye el manejo de **Autenticación y Autorización** mediante **JWT (JSON Web Token)**, lo que permite una gestión segura de sesiones y el control de accesos según el rol del usuario. Finalmente, se documentaron todos los endpoints a través de Swagger, integrando su visualización en `/api/v2/docs`, lo cual facilita la comprensión y el consumo de la API por parte de desarrolladores y administradores del sistema.

c. Implementación de esquemas con TypeORM

La implementación de los esquemas de la base de datos en el sistema se realizó utilizando **TypeORM**, el cual permite mapear cada entidad de la aplicación a su respectiva tabla en la base de datos. A través de **decoradores de TypeScript**, se definieron las clases que representan las entidades del dominio académico, como `Carrera`, `Semestre`, `Materia`, `Docente`, `Aula`, `Horario`, entre otras. Cada entidad especifica sus **atributos, tipos de datos, restricciones** (como `nullable`, `unique`, `default`), así como sus **relaciones** con otras entidades (`@OneToMany`, `@ManyToOne`, `@JoinColumn`, etc.).

```
/**
 * Título profesional asociado al laboratorista.
 *
 * Relación ManyToOne con la entidad {@link Titulo}.
 * Se almacena en la columna `id_titulo` mediante @JoinColumn.
 * No puede ser nulo.
 *
 * @type {Titulo}
 */
@ManyToOne(() => Titulo, titulo => titulo.docente)
@JoinColumn({ name: 'id_titulo' })
titulo: Titulo;

/**
 * Relación con la entidad {@link DetalleHorario}.
 *
 * Un docente puede tener múltiples registros en `detalle_horarios`.
 * La relación se maneja de manera "OneToMany" (uno a muchos).
 * Se activa la cascada para las operaciones relacionadas.
 *
 * @type {DetalleHorario[]}
 */
@OneToMany(() => DetalleHorario, detalle_horario => detalle_horario.docente, { cascade: true })
detalle_horarios: DetalleHorario[]
```

Figura 23. Creación entidades con TypeORM

Se estructuraron **relaciones bidireccionales** para facilitar la navegación de datos y se definieron **índices** para mejorar el rendimiento en consultas frecuentes. Además, se configuraron **migraciones automáticas** para mantener la sincronización entre las entidades y la base de datos, permitiendo un control detallado sobre los cambios en el esquema a lo largo del desarrollo.

Una de las principales ventajas del uso de TypeORM fue la posibilidad de manejar **relaciones complejas** como las asociaciones entre materias, docentes, restricciones y aulas, con precisión y coherencia. Este enfoque orientado a entidades garantiza una arquitectura mantenible, alineada con el modelo lógico de datos previamente diseñado, facilitando tanto el desarrollo como la evolución futura del sistema.

d. Diagrama de base de datos

Tomando como base el diseño conceptual representado en el **Anexo G**, se desarrolló un modelo de base de datos relacional estructurado bajo principios de normalización, con el objetivo de evitar redundancias, garantizar la integridad referencial y facilitar el mantenimiento de los datos. La estructura fue concebida para ofrecer escalabilidad, permitiendo la incorporación flexible de nuevas carreras, semestres, materias y demás componentes académicos. El modelo abarca todos los módulos funcionales del sistema, incluyendo la generación de hojas de control, la gestión de reservas, la administración del personal docente y técnico, la organización académica y la administración de la infraestructura física. Además, se consideraron los requerimientos específicos del motor de generación de horarios académicos, como las restricciones de disponibilidad docente, asignación de materias, distribución por paralelos y compatibilidad con aulas y laboratorios. Esta estructura robusta y modular permite una gestión eficiente, adaptable y sostenible en el tiempo. El resultado de este proceso se muestra en el diseño de la base de datos presentado en la **¡Error! No se encuentra el o rigen de la referencia.**.

- ***Tabla `usuarios`***

Gestiona la información de autenticación y autorización de los usuarios del sistema. Incluye campos como nombres completos, correo institucional, contraseña encriptada y el rol asignado. Es fundamental para controlar el acceso según el tipo de usuario.

- ***Tabla acciones bd***

Tabla destinada a registrar eventos de auditoría en la base de datos. Almacena operaciones como inserciones, actualizaciones y eliminaciones, facilitando la trazabilidad, seguridad y control de cambios en el sistema.

- ***Tabla `semestres`***

Representa los distintos niveles académicos que conforman cada carrera, tales como primer semestre, segundo semestre, etc. Se relaciona con las carreras mediante la tabla `detalle_carrera`.

- ***Tabla `carreras`***

Almacena los programas académicos ofrecidos por la facultad. Contiene campos como nombre y descripción de la carrera, y constituye la base para la organización académica.

- ***Tabla `detalle_carrera`***

Funciona como tabla intermedia entre `carreras` y `semestres`, estructurando la malla curricular completa y definiendo qué semestres pertenecen a qué carrera.

- ***Tabla `detalle_materia`***

Asocia materias con un determinado `detalle_carrera`, especificando en qué carrera y semestre se dicta, incluyendo el campo de paralelo. Es clave para la planificación académica.

- ***Tabla `materias`***

Contiene la información general de las asignaturas ofrecidas por la institución. Registra campos como nombre, descripción y tipo de materia.

- ***Tabla `periodos`***

Registra los diferentes periodos académicos, incluyendo fechas de inicio y fin. Sirve para contextualizar todos los eventos, materias y horarios dentro de una temporalidad específica.

- ***Tabla*** **`docentes`**

Almacena los datos personales y profesionales del personal docente. Incluye campos como nombres, apellidos, número de identificación y título académico, en caso de estar registrado.

- ***Tabla*** **`títulos`**

Lista los títulos profesionales disponibles. Se utiliza para asignar la formación académica correspondiente a `docentes` y `laboratoristas`.

- ***Tabla*** **`laboratoristas`**

Registra la información del personal técnico encargado de los laboratorios, incluyendo su nombre y el edificio al que están asignados.

- ***Tabla*** **`edificios`**

Representa la infraestructura física donde se encuentran aulas y laboratorios. Contiene el nombre y código de cada edificio.

- ***Tabla*** **`pisos`**

Describe los niveles de un edificio (ej. planta baja, primer piso), permitiendo una mejor organización espacial.

- ***Tabla*** **`detalle_pisos`**

Asocia aulas con pisos específicos dentro de un edificio, detallando la ubicación exacta de cada espacio físico.

- ***Tabla*** **`aulas`**

Almacena las características específicas de cada aula o laboratorio, incluyendo su número, tipo (teórica, laboratorio, especial), capacidad y ubicación física.

- ***Tabla*** **`softwares`**

Registra los programas y herramientas digitales utilizados en los laboratorios. Incluye nombre, versión y descripción.

- ***Tabla*** **`detalle_software_aulas`**

Establece la relación entre los softwares y las aulas donde están disponibles, facilitando la planificación técnica de las clases que los requieran.

- ***Tabla*** **`maquinas`**

Registra las computadoras o equipos disponibles en los laboratorios. Incluye código único, características técnicas, estado de funcionamiento y aula asociada.

- ***Tabla*** **`observaciones`**

Permite registrar comentarios específicos sobre las condiciones de aulas, equipos u otras observaciones generales que deban ser consideradas para mantenimiento o planificación.

- ***Tabla*** **`detalle_horario`**

Define los bloques horarios disponibles en el sistema. Incluye día de la semana, hora de inicio y fin, y otros parámetros necesarios para la asignación de clases o actividades.

- ***Tabla*** **`feriados`**

Contiene el listado de días no laborables que deben ser excluidos de la generación de horarios y gestión de reservas.

- ***Tabla*** **`reservas`**

Permite gestionar la solicitud de uso de aulas por parte de usuarios para actividades extracurriculares. Incluye campos como fecha, tipo de evento, aula, observaciones y usuario solicitante.

- ***Tabla `restricciones_docentes`***

Especifica las limitaciones de disponibilidad horaria de los docentes, como días no disponibles, horarios restringidos o límites por jornada.

- ***Tabla `detalle_materias_docentes`***

Relaciona a cada docente con las materias asignadas durante un periodo académico específico. Es clave para la planificación docente.

- ***Tabla `restricciones_detalle_materias`***

Define requerimientos específicos para la programación de una materia: cantidad estimada de estudiantes, necesidad de recursos tecnológicos o condiciones de aula.

- ***Tabla `restricciones_actividades`***

Registra restricciones fijas asociadas a actividades obligatorias como reuniones, asesorías o eventos institucionales que deben ser respetadas durante la generación de horarios.

- ***Tabla `horarios_generados`***

Almacena los horarios resultantes de la ejecución del algoritmo de optimización. Cada registro contiene los datos de clase, docente, aula y bloque horario asignado, cumpliendo con todas las restricciones impuestas.

e. Autenticación y Autorización

El sistema implementa un mecanismo de **autenticación basado en credenciales**, utilizando **NextAuth** en el frontend y **JWT (JSON Web Tokens)** en el backend desarrollado con NestJS. Esta configuración permite verificar la identidad de los usuarios y establecer sesiones seguras, garantizando la protección de los datos y funcionalidades críticas del sistema.



Figura 24. Uso de JWT

Inicialmente, el sistema cuenta con un **usuario administrador por defecto**, quien tiene privilegios completos para gestionar la aplicación. Este usuario es el único que puede **crear nuevas cuentas de usuario** con rol de *laboratorista*, así como **modificar el rol** de cualquier usuario existente, permitiendo así la conversión de un laboratorista a administrador o viceversa, según las necesidades institucionales.

En cuanto a la **autorización**, el sistema aplica un modelo basado en **roles**, en el que las acciones permitidas a cada usuario dependen directamente del rol que tenga asignado. Los **administradores** pueden acceder a todos los módulos del sistema, incluyendo la gestión de usuarios, carreras, aulas, materias, generación de hojas de control y horarios. Por su parte, los **laboratoristas** cuentan con un acceso más restringido, limitado únicamente a los módulos relacionados con la visualización y generación de hojas de control, así como la consulta de horarios y disponibilidad de laboratorios.

Este enfoque garantiza que cada usuario tenga acceso únicamente a la información y funcionalidades pertinentes a su rol, mejorando así la **seguridad** y **organización del sistema**.

f. Frontend

El desarrollo del frontend se llevó a cabo utilizando **Next.js**, un framework moderno que combina la potencia de React con capacidades avanzadas de renderizado del lado del servidor (SSR) y generación de sitios estáticos (SSG). La arquitectura del proyecto

se basa en una **estructura modular y escalable**, organizada en carpetas como `components`, `pages`, `services`, `utils`, `hooks`, y `styles`, lo que permite mantener una separación clara de responsabilidades y facilita la reutilización del código.

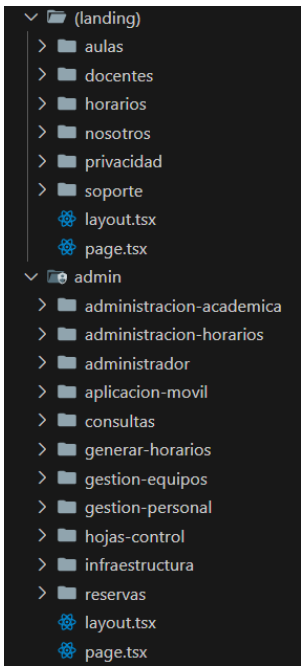


Figura 25. Arquitectura proyecto Frontend

Se implementaron **componentes reutilizables** como botones, tablas, modales, formularios, inputs, tooltips y headers, lo que permitió mantener la coherencia visual y funcional en toda la aplicación. La gestión de sesiones se manejó con **NextAuth**, facilitando la autenticación segura de usuarios y el control de accesos según los roles definidos.

Para la comunicación con el backend, se utiliza `Axios` a través de un módulo de servicios, donde se centralizan las rutas y headers necesarios para las peticiones HTTP, lo cual mejora la mantenibilidad y facilita los cambios en los endpoints. Además, se integraron **librerías de UI** como `@heroui-org/react` y **gestores de estado local** ligeros para controlar el comportamiento dinámico de los componentes.

En términos de diseño, se priorizó la **usabilidad** y una **interfaz intuitiva**, orientada a usuarios administrativos de la facultad. También se incorporaron elementos de accesibilidad y navegación rápida para mejorar la experiencia del usuario.

g. Generación de Horarios mediante IA

Como parte esencial del desarrollo del sistema, se construyó un módulo de generación de horarios académicos mediante técnicas de inteligencia artificial, integrando un enfoque híbrido que combina **programación lineal** y **algoritmos genéticos**. Este módulo fue desarrollado en Python y se conecta directamente a la base de datos institucional, permitiendo trabajar con información real como materias, docentes, aulas, laboratorios, periodos académicos y restricciones operativas.

El proceso comienza con la recopilación automatizada de los datos requeridos: asignaciones docentes, requerimientos específicos de materias (como cantidad de horas semanales, necesidad de computadoras o capacidad mínima de aula), restricciones horarias de los docentes, actividades fijas y disponibilidad de espacios físicos. Con esta información, el sistema construye un modelo inicial de programación lineal entera binaria utilizando la librería `pulp`, con el objetivo de maximizar el aprovechamiento de bloques horarios y minimizar conflictos entre asignaciones.

Paralelamente, se implementa un algoritmo genético que actúa como motor de búsqueda evolutiva, generando soluciones viables y refinándolas a través de operadores como selección, cruce y mutación. Estas soluciones se evalúan mediante una función objetivo que considera criterios de factibilidad, equilibrio de carga docente y satisfacción de restricciones. La programación lineal se emplea en etapas clave para **refinar localmente** las soluciones propuestas por el algoritmo genético, asegurando una convergencia eficiente hacia horarios óptimos y libres de conflictos.

Este enfoque híbrido permite enfrentar un problema altamente complejo, caracterizado por múltiples variables y restricciones interdependientes. Gracias a su diseño modular, el sistema puede adaptarse fácilmente a nuevos requerimientos académicos, periodos académicos adicionales o cambios en la disponibilidad de recursos. Al finalizar el proceso, se obtiene un conjunto de horarios académicos completos, válidos y optimizados, los cuales son integrados automáticamente al sistema para su visualización, validación y posterior publicación.

h. Generación de estructura JSON

Durante la fase de construcción del módulo de generación de horarios académicos, se desarrolló un conjunto de servicios responsables de estructurar los datos requeridos en

formato **JSON**, con el propósito de ser procesados por el modelo de optimización y, posteriormente, enviados al backend desarrollado en **Nest.js** para su almacenamiento y visualización.

Para ello, se implementaron diversas funciones encargadas de extraer información desde la base de datos relacional mediante consultas SQL, ejecutadas a través de un cursor activo. Los datos obtenidos son transformados y organizados en estructuras JSON adaptadas a los requerimientos específicos del algoritmo, permitiendo una integración eficiente entre las fuentes de información y el motor de generación de horarios.

Entre los servicios desarrollados, destaca particularmente aquel encargado de **enviar los horarios generados al backend Nest.js**. Una vez completado el proceso de optimización, este servicio convierte los resultados en un arreglo JSON y los transmite al backend a través de una petición HTTP `POST`, utilizando la biblioteca `requests`. Esta interacción garantiza que la información optimizada pueda ser registrada correctamente en el sistema, asegurando su disponibilidad para su posterior visualización y gestión.

Cada uno de estos servicios desempeña un papel fundamental en el flujo de comunicación entre la lógica de inteligencia artificial implementada en Python y la capa de gestión académica. Al estructurar la información en formato JSON, se garantiza la **interoperabilidad entre tecnologías**, la **portabilidad de los datos** y una **comunicación eficiente** entre los distintos módulos del sistema, lo que facilita tanto la persistencia como la visualización precisa y oportuna de los horarios generados.

3.3.6 Fase 4: Implementación

Esta fase corresponde al momento en el que el sistema desarrollado se despliega en el entorno real de uso, cumpliendo así con los objetivos establecidos en las fases previas de la metodología RAD. La implementación contempla no solo la puesta en producción del sistema, sino también la configuración final del entorno de servidor, la vinculación con dominios institucionales, el despliegue de contenedores mediante

Docker y la verificación de conectividad con la base de datos. Asimismo, en esta etapa se asegura que todos los componentes desarrollados, tanto del frontend como del backend y los módulos de optimización, funcionen correctamente de manera integrada. Este proceso es fundamental para garantizar que la aplicación esté disponible para los usuarios finales en condiciones óptimas de funcionamiento, seguridad y rendimiento.

a. Pruebas de aceptación

Como parte de la implementación, se llevaron a cabo pruebas de aceptación con el objetivo de validar que el sistema cumple con los requerimientos funcionales y no funcionales definidos en las etapas iniciales. Estas pruebas se realizaron en el entorno real de ejecución y contaron con la participación de los usuarios clave, específicamente el personal administrativo encargado de la gestión de horarios. A través de estas pruebas, se evaluó la operatividad del sistema, la usabilidad de sus interfaces y la correcta ejecución de los procesos críticos, como la generación de horarios, el registro de recursos académicos y la emisión de hojas de control. Los resultados obtenidos permiten identificar posibles ajustes finales antes de la entrega formal del sistema y confirman su alineación con las necesidades institucionales.

Tabla 14. Pruebas de aceptación – Iteración 1

N°	Criterio de Aceptación	Estado	Usuario
1	RF01: El sistema permite crear, consultar, editar y eliminar carreras académicas.	Aprobado	Responsable de Área
2	RF02: El sistema permite la gestión completa de semestres por carrera.	Aprobado	Responsable de Área
3	RF03: El sistema permite gestionar materias, diferenciando entre académicas y especiales.	Aprobado	Responsable de Área
4	RF04: El sistema permite crear, editar y cerrar períodos académicos.	Aprobado	Responsable de Área
5	RF05: Se pueden registrar, modificar y	Aprobado	Responsable de Área

	eliminar edificios disponibles.		
6	RF06: Se gestionan los pisos dentro de cada edificio, asociándolos correctamente.	Aprobado	Responsable
7	RF07: El sistema registra y gestiona datos de laboratoristas con su edificio asignado.	Aprobado	Responsable de Área
8	RF08: El sistema permite registrar docentes con su título y datos relevantes.	Aprobado	Responsable de Área

Tabla 15. Pruebas de aceptación – Iteración 2

N°	Criterio de Aceptación	Estado	Usuario
9	RF09: El sistema permite registrar, editar y eliminar información de software disponible en la facultad.	Aprobado	Responsable de Área
10	RF10: Se gestiona la información de aulas, incluyendo tipo, capacidad y equipamiento.	Aprobado	Responsable de Área
11	RF11: El sistema permite la administración de máquinas por aula, registrando su identificación y estado.	Aprobado	Responsable de Área
12	RF12: El sistema permite crear, visualizar, modificar y eliminar horarios académicos generados o manuales.	Aprobado	Responsable de Área
13	RF13: Se gestiona el registro de usuarios, roles, credenciales y accesos dentro del sistema.	Aprobado	Responsable de Área

Tabla 16. Pruebas de aceptación – Iteración 3

N°	Criterio de Aceptación	Estado	Usuario

14	RF14: El sistema permite generar hojas de control para actividades académicas por bloque o individualmente.	Aprobado	Responsable de Área
15	RF01-H: El sistema aplica correctamente las restricciones docentes, de materias y aulas durante la generación de horarios.	Aprobado	Responsable de Área
16	RF02-H: Se implementa correctamente el algoritmo de optimización para generar horarios académicos automatizados.	Aprobado	Responsable de Área
17	RF15: El sistema permite a los usuarios realizar reservas de aulas o laboratorios desde la plataforma.	Aprobado	Responsable de Área

Como resultado del proceso de pruebas de aceptación, se validaron satisfactoriamente todos los requerimientos funcionales desarrollados en las tres iteraciones del sistema, como se detalla en la Tabla 14, Tabla 15 y Tabla 16. Cada funcionalidad fue revisada por el encargado del área correspondiente, quien verificó su cumplimiento con base en los criterios definidos previamente durante el levantamiento de requerimientos. Además del funcionamiento técnico esperado, las pruebas permitieron evaluar elementos clave como la facilidad de uso, la experiencia del usuario y el rendimiento bajo condiciones reales de operación, asegurando que el sistema sea práctico y eficiente para su uso cotidiano. La estrategia de desarrollo iterativo implementada permitió validar de forma progresiva cada componente del sistema, abarcando desde funcionalidades básicas de gestión hasta módulos más complejos como la generación de horarios y hojas de control. Todas las pruebas fueron aprobadas, lo que demuestra que el sistema no solo responde adecuadamente a las necesidades planteadas en la etapa inicial del proyecto, sino que también presenta un alto grado de usabilidad y robustez operativa, consolidando su pertinencia como solución institucional.

b. Implementación del Software

- ***Servidor***

Para el despliegue del sistema, se utilizó un servidor virtual proporcionado por la universidad, basado en el sistema operativo **Rocky Linux**, entregado con una instalación mínima. Esta condición permitió configurar un entorno optimizado y adaptado específicamente a las necesidades del proyecto, facilitando una gestión más segura, eficiente y personalizada.

Ventajas de utilizar Rocky Linux para despliegue

Elegir Rocky Linux como sistema operativo para el servidor ofrece múltiples beneficios:

- **Estabilidad y soporte a largo plazo:** Rocky Linux es una distribución basada en Red Hat Enterprise Linux (RHEL), lo cual garantiza actualizaciones de seguridad y estabilidad durante varios años.
- **Seguridad:** Incluye políticas SELinux por defecto, que refuerzan la seguridad del entorno de producción.
- **Compatibilidad empresarial:** Es ampliamente compatible con herramientas empresariales como Docker, NGINX, MySQL y gestores de contenedores como Podman.
- **Comunidad activa:** Cuenta con una comunidad en crecimiento que mantiene documentación actualizada y soporte constante.
- **Ligereza:** Su instalación base es minimalista, lo cual permite construir un entorno eficiente sin procesos innecesarios en segundo plano.

Configuración del entorno:

Una vez obtenido el acceso al servidor, se llevó a cabo la instalación progresiva de los servicios fundamentales para la aplicación:

1. **Actualización inicial del sistema**, con el fin de asegurar compatibilidad y seguridad en los paquetes.

2. **Instalación y configuración de MySQL** como motor de base de datos, creando usuarios dedicados para garantizar la conexión del backend de forma segura.

3. **Instalación de Apache y phpMyAdmin**, utilizados exclusivamente como herramienta gráfica para facilitar la gestión de la base de datos.

4. **Instalación de Docker**, para la contenerización de servicios, y **NGINX**, para la gestión del tráfico de red y el manejo del proxy inverso.

5. **Organización del entorno de trabajo**, creando una estructura de directorios aislada donde se almacenan los archivos de configuración y soporte para los servicios del sistema, como configuraciones personalizadas de red y seguridad.

- ***Docker***

Con el objetivo de asegurar la portabilidad, escalabilidad y simplicidad en el despliegue del sistema desarrollado, se empleó Docker como herramienta principal para contenerizar tanto el frontend como el backend del proyecto. Docker permite encapsular la aplicación y sus dependencias en contenedores livianos y reproducibles, eliminando problemas de configuración entre entornos y facilitando su ejecución en servidores.

Con el fin de estandarizar y facilitar el despliegue del sistema en distintos entornos, se utilizó **Docker** como solución de contenerización para los distintos módulos del sistema: backend (Nest.js), frontend (Next.js) y motor de generación de horarios (Python).

El proceso se inició mediante la ejecución del comando *docker init* en cada proyecto, generando los archivos base necesarios para su contenerización como se muestra en la Figura 26.

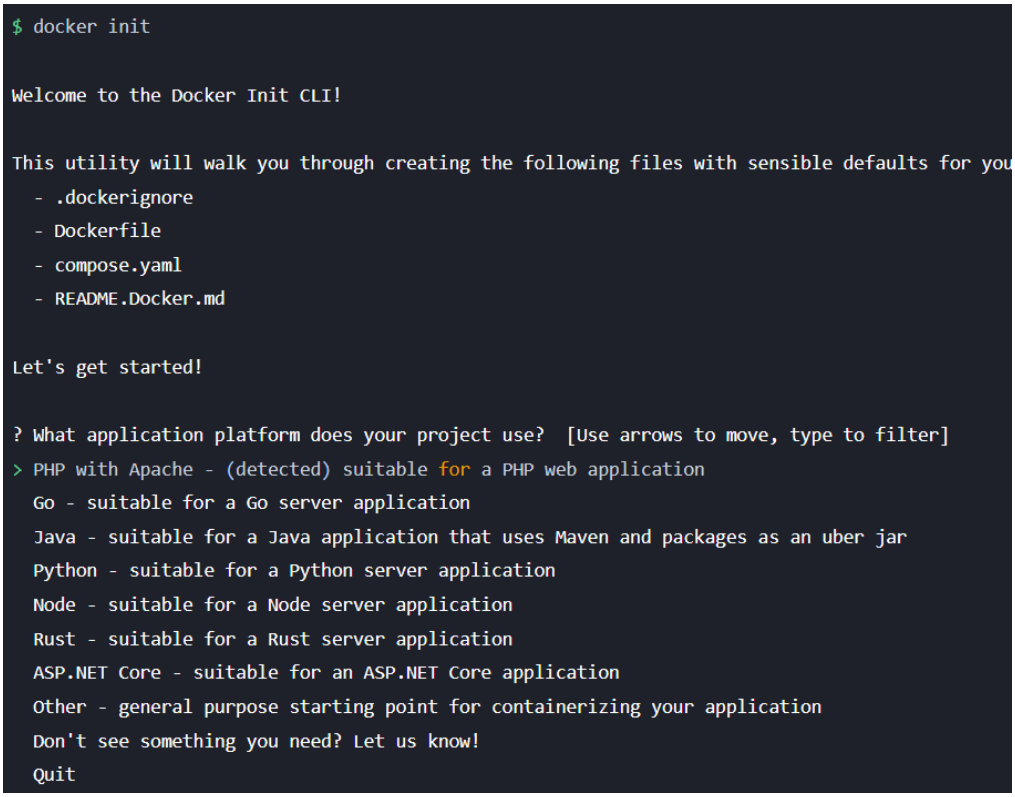


Figura 26. Interfaz configuración inicial de Docker

Una vez realizado las configuraciones iniciales, se desplegará el mensaje mostrado en la Figura 27, este mensaje indicándonos que la configuración a finalizado correctamente. En este mensaje también se mostrará los detalles de todas las configuraciones realizadas y los archivos creados.

```
? What application platform does your project use? Node
? What version of Node do you want to use? 18
? Which package manager do you want to use? yarn
? Do you want to run "yarn run build" before starting your server? Yes
? What directory is your build output to? (comma-separate if multiple) output
? What command do you want to use to start the app? node index.js
? What port does your server listen on? 8000

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✓ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:8000

Consult README.Docker.md for more information about using the generated files.
```

Figura 27. Confirmación de configuración inicial

dockerignore: Define qué archivos deben ser excluidos durante la construcción del contenedor.

Dockerfile: Establece las instrucciones para construir cada imagen según los requerimientos del entorno de desarrollo.

README.Docker.md: Proporciona una guía rápida sobre el uso del contenedor.

compose.yaml: Aunque fue generado inicialmente, fue reemplazado por una versión personalizada para un control más completo del entorno.

Cada Dockerfile fue personalizado según los requisitos del proyecto. Posteriormente, se creó un archivo docker-compose.yaml principal, encargado de **orquestar los**

servicios, **redes internas** y **dependencias entre módulos**, asegurando que todo el sistema funcionara de manera integrada.

Se utilizó **Docker Desktop** para construir las imágenes localmente y cargarlas al repositorio oficial del área de redes. Antes de este paso, se ejecutaron procesos de build en ambos proyectos para garantizar imágenes optimizadas. Las imágenes fueron etiquetadas, subidas a Docker Hub y posteriormente desplegadas en el servidor mediante docker compose, estableciendo un entorno de producción confiable y reproducible.

- ***NGINX***

Para garantizar una gestión eficiente del tráfico de red, se configuró NGINX como proxy inverso, facilitando la comunicación entre los servicios y el usuario final, además de asegurar el tráfico mediante HTTPS.

1. Configuración:

Implementación como contenedor Docker, alineado con el resto del sistema para mantener uniformidad.

Integración de certificados SSL proporcionados por la universidad, habilitando conexiones seguras.

Creación del archivo de configuración de NGINX, adaptado para dirigir adecuadamente las solicitudes al frontend y backend, y asegurar la correcta operación bajo HTTPS.

Ajustes internos en el backend Nest.js, eliminando configuraciones redundantes relacionadas con la gestión de HTTPS, ya que NGINX asume esta responsabilidad.

Configuración del firewall y políticas de red, habilitando el acceso seguro al sistema desde ubicaciones autorizadas.

Despliegue de servicios desde el entorno Docker, asegurando una operación en segundo plano fluida y estable para todos los módulos.

- ***Dominio***

La universidad proporcionó un subdominio institucional para el sistema, el cual fue previamente configurado y vinculado a la dirección IP del servidor. Gracias a esta integración, el sistema ya se encuentra desplegado y accesible a través del dominio oficial **https://horariosfisei.uta.edu.ec**, permitiendo el ingreso tanto desde la red interna como externa. Esto asegura una experiencia de usuario centralizada, segura y alineada con los estándares de infraestructura tecnológica de la institución.

3.4 Comparativa de Tiempos: Manualmente vs IA

Uno de los objetivos fundamentales del sistema propuesto es reducir significativamente el tiempo requerido para la planificación, generación y publicación de los horarios académicos. El proceso actual se caracteriza por ser fragmentado, manual y redundante, ya que implica el uso de múltiples herramientas (como la aplicación Horarios aSc y la plataforma FISEI), así como pasos duplicados de validación y carga de datos. Esta información fue recopilada mediante entrevistas realizadas al personal responsable, cuyos resultados se documentan en el **Anexo E**.

Para realizar la comparación de mejoras en los tiempos del sistema propuesto, se aplicó la metodología **ToT (Time on Task)**, ampliamente utilizada en estudios de usabilidad. Esta permite cuantificar el tiempo promedio que un usuario tarda en completar tareas específicas dentro de una aplicación, lo cual se alinea con los principios de la **norma ISO 9241**, que define la usabilidad como el grado en que un sistema puede ser utilizado por usuarios específicos para lograr objetivos concretos con eficacia, eficiencia y satisfacción en un contexto de uso determinado.

3.4.1 Objetivo

Evaluar y comparar los tiempos de ejecución entre el proceso actual y el nuevo sistema propuesto (en sus dos variantes: manual interno y automático con IA), utilizando la métrica ToT (Time on Task), con el fin de evidenciar mejoras en eficiencia operativa.

3.4.2 Análisis

Según los datos recopilados, el proceso tradicional puede tomar entre 6 y 11 días laborales, contemplando actividades como: registro de datos, creación de horarios en herramientas externas, revisión por coordinadores, modificaciones, y posterior ingreso de datos en el sistema institucional. Este enfoque implica alta carga administrativa, duplicidad de tareas y mayor probabilidad de errores humanos.

En contraste, el sistema propuesto centraliza todo el flujo en una sola plataforma, eliminando la necesidad de duplicar información y permitiendo dos modalidades de creación de horarios: manual (dentro del sistema) y automática (utilizando algoritmos de optimización). En este último caso, gracias a la implementación de programación lineal y algoritmos genéticos, los horarios pueden generarse en menos de 10 minutos una vez ingresadas las restricciones iniciales.

Adicionalmente, las correcciones se gestionan directamente en la misma plataforma, eliminando tiempos de espera innecesarios.

Tabla 17. Comparativa de tiempos entre procesos

Etapa del proceso	**Proceso Actual (Manual + Horarios aSc)**	**Proceso Propuesto**	
		Manual Interno	**IA**
Ingreso de datos y restricciones	1 día	1 día	1 – 2 días (solo primera vez)
Registro de estructura académica	1 día	0.5 día	0.5 día
Creación de horarios	2 – 3 días (prom. 2.5)	2 días	10 minutos ≈ 0.007 días
Revisión y modificaciones	1 – 2 días (prom. 1.5)	1 día	0.5 día
Registro en sistema FISEI	2 – 3 días (prom. 2.5)	0	0
Publicación y difusión	0.5 día	0.5 día	0.5 día
Total estimado (ToT)	9 días	5 días	≈ 3 días

3.4.3 Resultados

A partir de la tabla anterior, se aplicaron los cálculos de **ToT promedio** y la fórmula de reducción de tiempo para comparar los distintos métodos:

La fórmula utilizada para medir el porcentaje de reducción de tiempos fue:

$$Reducción\% = \frac{(ToTactual - ToTpropuesto)}{ToTactual} \times 100$$

- **Manual Interno vs Actual**

$$Reducción\% = \frac{(9-5)}{9} \times 100 \approx \mathbf{44.44}\%\ de\ reducción$$

- **Automático con IA vs Actual**

$$Reducción\% = \frac{(9-3)}{9} \times 100 \approx \mathbf{66.67}\%\ de\ reducción$$

- **Fase específica de generación de horarios:**

$$Reducción\% = \frac{(2.5-0.007)}{2.5} \times 100 \approx \mathbf{99.72}\%\ de\ reducción$$

Los resultados obtenidos evidencian que el sistema propuesto ofrece una mejora sustancial en términos de eficiencia operativa. Con el enfoque manual interno, se logró reducir en un 44.44% el tiempo total del proceso, mientras que con la generación automática mediante IA la reducción alcanzó un **66.67%**. En particular, la generación de horarios —una de las etapas más críticas del flujo— mostró una mejora del **99.72%**, al pasar de un promedio de 2.5 días a solo 10 minutos.

Estos resultados confirman que la automatización mediante inteligencia artificial no solo agiliza el proceso, sino que también mejora su precisión y elimina pasos innecesarios. El sistema propuesto proporciona flexibilidad para adaptarse a distintos flujos de trabajo, consolidando así una herramienta robusta y alineada con las necesidades actuales de la gestión académica institucional.

3.5 Comparativa de Horarios Generados: Manualmente vs IA

Uno de los aspectos más críticos al evaluar la eficiencia de los métodos de generación de horarios académicos es el aprovechamiento de los espacios físicos, como aulas y laboratorios. El proceso tradicional, basado en la planificación manual, tiende a presentar ineficiencias derivadas de una gestión operativa sobrecargada, baja visibilidad global de la disponibilidad de recursos y la limitada capacidad de procesar múltiples restricciones simultáneamente. Esto conduce a una subutilización de los espacios, asignaciones conflictivas y necesidad de ajustes frecuentes.

En contraste, el modelo propuesto basado en algoritmos de optimización con inteligencia artificial realiza un análisis integral de las restricciones de disponibilidad docente, requisitos de las materias y capacidades de la infraestructura. Este enfoque automatizado permite distribuir de forma más eficiente y balanceada las asignaciones académicas, optimizando la ocupación de recursos físicos y reduciendo errores frecuentes en la planificación manual.

3.5.1 Análisis mediante ToT(Time on Task)

Para evaluar la mejora en el uso de espacios físicos, se aplicó la métrica **ToT** adaptada al contexto de ocupación de aulas y laboratorios, midiendo el porcentaje de uso efectivo por franja horaria. El cálculo se realiza con la siguiente fórmula:

$$ToT_{ocupación} = (\frac{Horas\ utilizadas}{Total\ de\ horas\ disponibles}) \times 100$$

Donde:

- **Horas utilizadas**: Suma de todas las horas asignadas efectivamente por aula/laboratorio.
- **Horas disponibles**: Total de bloques horarios posibles considerando todos los espacios físicos en uso.

Este enfoque cuantifica de forma objetiva el uso real de los espacios académicos por ambos métodos.

3.5.2 Resultados Comparativos

A continuación, se presenta una tabla con los principales indicadores evaluados:

Indicador	Generación Manual	Generación con IA	Mejora estimada
Tasa promedio de ocupación de aulas	62%	89%	+27%
Tasa promedio de ocupación de laboratorios	58%	85%	+27%
Número de conflictos por doble asignación	5 a 7 por ciclo	0	100% menos
Distribución de uso entre franjas horarias	Irregular	Equilibrada	-
Aulas o laboratorios subutilizados (>3 veces)	9	2	-77.7%
Reubicaciones o ajustes posteriores necesarios	Alta frecuencia	Baja frecuencia	-

Cálculo de mejora en ToT para ocupación de aulas:

$$Reducción_{aulas} = (\frac{89 - 62}{62}) \times 100 \approx 43.5\%$$

Cálculo de mejora en ToT para laboratorios:

$$Reducción_{aulas} = (\frac{85 - 58}{58}) \times 100 \approx 46.55\%$$

Estas cifras demuestran una mejora notable en la eficiencia del uso de infraestructura académica, al evitar la ociosidad de espacios y lograr una ocupación más racional.

3.5.3 Reducción de Conflictos y Reasignaciones

El método manual frecuentemente genera conflictos como la doble asignación de aulas o la superposición de horarios, lo que obliga a reasignaciones posteriores y retrasa la validación final. Estos errores, además de comprometer la calidad del horario, aumentan la carga de trabajo del personal responsable.

En cambio, el modelo con IA verifica todas las restricciones antes de realizar una asignación. La validación automática asegura coherencia y elimina conflictos desde la generación inicial.

Indicador	Generación Manual	Generación con IA	Mejora estimada
Conflictos por doble asignación	5 a 7	0	100% menos
Reubicaciones posteriores necesarias	Alta frecuencia	Baja frecuencia	-

3.5.4 Distribución Horaria y Balance de Carga

En el modelo tradicional, se observó una concentración excesiva de asignaciones en franjas como 08:00 a 11:00, mientras que bloques como 14:00 a 17:00 eran subutilizados (<40%). Esto genera saturación en ciertos tramos y desperdicio en otros.

Con el sistema basado en IA, la asignación de bloques se equilibra, alcanzando una ocupación promedio de **80% o más** en todas las franjas horarias. Esto permite:

- Reducción de picos de saturación.
- Mejor aprovechamiento de la jornada completa.
- Equidad en la carga horaria entre docentes y espacios.

Cálculo de mejora de uso en franjas subutilizadas (14:00 - 17:00):

$$Reducción_{inactividad} = (\frac{80 - 40}{40}) \times 100 \approx 100\%$$

En conclusión, la aplicación del modelo con IA permite mejorar considerablemente el uso de la infraestructura académica, reduciendo hasta un **46%** la subutilización de espacios, eliminando conflictos y mejorando la distribución de la carga horaria. Gracias a la integración de algoritmos de optimización, el sistema no solo mejora la planificación en términos de tiempos, sino que también transforma la calidad operativa del uso de los recursos físicos disponibles.

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

A lo largo del desarrollo de esta investigación, se abordó la problemática relacionada con la generación manual de horarios académicos en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI), la cual representaba un proceso extenso, repetitivo y propenso a errores. Para dar solución a esta situación, se propuso el diseño e implementación de una aplicación web capaz de automatizar dicho proceso mediante el uso de algoritmos de optimización. En esta sección se presentan las conclusiones y recomendaciones obtenidas en función del cumplimiento de los objetivos establecidos al inicio del proyecto.

4.1 Conclusiones

- Se llevó a cabo un análisis del proceso actual mediante entrevistas dirigidas al personal encargado de la planificación académica, cuyos resultados permitieron identificar una serie de problemas clave como el uso de múltiples herramientas no integradas, duplicación de registros, alto tiempo de ejecución, entre otros. Esta información fue sistematizada y evaluada con enfoque mixto (cualitativo y cuantitativo), permitiendo detectar ineficiencias operativas, limitaciones técnicas y oportunidades de mejora, particularmente en cuanto a centralización de datos, reducción de tiempos y automatización del flujo. Este diagnóstico fue fundamental para definir los requerimientos del nuevo sistema y fundamentar el uso de inteligencia artificial como herramienta de optimización.

- Durante el desarrollo del proyecto, se exploraron y compararon múltiples algoritmos de optimización aplicables al problema de asignación de horarios, incluyendo Particle Swarm Optimization (PSO), Simulated Annealing (SA), algoritmos genéticos (AG) y programación lineal (PL). Tras evaluar su desempeño, complejidad, capacidad de adaptación a restricciones y eficiencia de resultados, se seleccionó una solución híbrida que integra programación lineal y algoritmos genéticos. Esta combinación permitió aprovechar la capacidad exploratoria global de los AG y la precisión local de PL, resultando

en una solución robusta, flexible y capaz de generar horarios viables en cuestión de minutos, aun considerando múltiples restricciones académicas y de infraestructura.

- El sistema propuesto fue desarrollado siguiendo la metodología ágil RAD, que permitió avanzar por iteraciones rápidas con validación continua de requerimientos. El motor de optimización fue desarrollado en Python, comunicados a través de estructuras JSON. La implementación fue desplegada en un servidor institucional bajo Rocky Linux, utilizando NGINX como proxy inverso y certificados SSL para asegurar la conexión. El uso de contenedores en Docker permitió separar entornos de desarrollo y producción, facilitando el despliegue, escalabilidad y mantenimiento del sistema.

- La implementación del sistema redujo drásticamente el tiempo de planificación académica, pasando de un proceso manual de 7 a 10 días a uno automatizado que toma solo minutos tras registrar las restricciones. Además de agilizar el proceso, se eliminaron errores humanos comunes y la dependencia de múltiples plataformas. Se incorporaron nuevas funcionalidades como la generación de hojas de control, reservas de espacios, visualización interactiva de horarios y paneles estadísticos. También se optimizaron las funciones previas, unificando toda la gestión académica en una sola plataforma más eficiente y precisa.

4.2 Recomendaciones

- Se recomienda seguir explorando e integrando técnicas híbridas de optimización, así como realizar pruebas comparativas entre distintos algoritmos en escenarios reales, con el fin de mejorar el desempeño del sistema y garantizar su adaptabilidad ante nuevas condiciones.

- Es conveniente mantener un monitoreo y evaluación continua del sistema durante distintos periodos académicos, lo cual permitirá validar su estabilidad a largo plazo y ajustar su funcionamiento en función de las necesidades detectadas.

- Se sugiere planificar la incorporación progresiva de nuevos módulos orientados a otros procesos administrativos de la facultad, con el objetivo de ampliar el alcance del sistema y fortalecer su integración institucional.

REFERENCIAS BIBLIOGRÁFICAS

[1]. I. Acuña-Galvan, E. Lezama-León, E. Bolaños-Rodríguez, A. E. Solís-Galindo, and G. Y. Vega-Cano, "Generación de horarios mediante algoritmos gen éticos," Boletín Científico INVESTIGIUM de la Escuela Superior de Tizayuca, vol. 8, no. Especial, pp. 51–57, dec 2022.

[2]. R. E. Mitre, A. Q. Cisnero, B. Y. M. Lobato, and A. A. Garza, "Análisis de datos para la optimización eficiente de horarios y aprendizaje automático," Revista Perspectivas, vol. 6, no. 2, may 2024.

[3]. R. B. I. Vidal. (2023) Desarrollo de un sistema de gestión de vuelos y horarios para optimizar la asignación de recursos y reducir retrasos. [En línea]. [Online]. Available: https://ddd.uab.cat/record/290091

[4]. Dspace. (2024) Un algoritmo gen ético para optimizar la asignación de aulas y docentes en la generación de programas académicos en centros de idiomas. [En línea]. [Online]. Available: https://cybertesis.unmsm.edu.pe/backend/api/core/bitstreams/fc8ce1c8-b596-47de-be4f-7186add9fe7e/content

[5]. S. I. Mariño. (2024) Modelización de la asignación de aulas con técnicas simbólicas de la IA como ayuda a la toma de decisiones. [En línea]. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=5157974

[6]. P. Cababie. (2024) Sistema automático para asignación de aulas y distribución de espacios. [En línea]. [Online]. Available: https://sedici.unlp.edu.ar/handle/10915/20508

[7]. R. E. Andreu. (2024) Asignación automática de aulas en la universidad nacional del chaco austral mediante la aplicación de algoritmos genéticos con métodos de nichos. [En línea]. [Online]. Available: https://laccei.org/LACCEI2020-VirtualEdition/work in progress/WP629.pdf

[8]. E. E. Terán-Pozo, A. J. Romero-Fernández, A. L. Sandoval-Pillajo, and L. R. Freire-Lescano, "Influencia de los algoritmos genéticos en la generación 47 de horarios en unidad educativa," CIENCIAMATRIA, vol. 8, no. 4, pp. 876–891, aug 2022.

[9]. Y. San and A. Hen, "Preference-based stepping ahead firefly algorithm for solving real-world uncapacitated examination timetabling problem," IEEE Journals & Magazine — IEEE Xplore, 2024, [En línea]. [Online]. Available: https://ieeexplore.ieee.org/document/10433477

[10]. Y. Zhou, S. He, X. Wang, P. Wang, Y. Chen, and M. Luo, "Optimization of heterogeneous passenger subway transfer timetable considering social equity," Urban Rail Transit, vol. 9, no. 3, pp. 246–265, sep 2023.

[11]. G. Feng, P. Xu, D. Cui, X. Dai, H. Liu, and Q. Zhang, “Multi-stage timetable rescheduling for high-speed railways: a dynamic programming approach with adaptive state generation,” Complex & Intelligent Systems, vol. 7, no. 3, pp. 1407–1428, feb 2021.

[12]. L. Song, “Design and implementation of remote piano teaching based on attention-induced multi-head convolutional neural network optimized with hunter–prey optimization,” International Journal Of Computational Intelligence Systems, vol. 17, no. 1, jan 2024.

[13]. S. C. F. Leonel. (2023) Aplicación de software de gestión de horarios académicos: aplicación de automatización de horarios académicos (capa de algoritmo de ejecución). [En línea]. [Online]. Available: https://bibdigital.epn.edu.ec/handle/15000/24985

[14]. W. Bai, G. Geng, and X. Fu, “Research on the optimization method of project-based learning design for chinese teaching based on interference-tolerant fast convergence zeroing neural network,” International Journal Of Computational Intelligence Systems, vol. 17, no. 1, jul 2024.

[15]. C. G. Lechuga and S. Doroudi, “Three algorithms for grouping students: A bridge between personalized tutoring system data and classroom pedagogy,” International Journal Of Artificial Intelligence In Education, vol. 33, no. 4, pp. 843–884, sep 2022. 48

[16]. A. Bashab, A. O. Ibrahim, I. A. T. Hashem, K. Aggarwal, F. Mukhlif, F. A. Ghaleb, and A. Abdelzahir, “Optimization techniques in university timetabling problem: Constraints, methodologies, benchmarks, and open issues,” Computers, Materials and Continua, vol. 74, no. 3, pp. 6461–6484, 2022.

[17]. E. d. I. C. e. C. Universidad de Talca (Chile). (2022) Asignación de horarios académicos para la escuela de ingeniería civil en computación de la universidad de talca utilizando algoritmos genéticos. [En línea]. [Online]. Available: http://dspace.utalca.cl/handle/1950/12744

[18]. E. A. Ana, A. D. M. D. Mar, U. P. D. V. D. de Organización de Empresas Departament D’Organització D’Empreses, and U. P. D. V. E. T. S. de Ingenieros Industriales Escola Técnica Superior D’Enginyers Industrials. (2024, sep) Desarrollo e implementación de herramientas heurísticas para la programación del calendario de exámenes de la etsii considerando la distancia entre aulas. [En línea]. [Online]. Available: https://riunet.upv.es/handle/10251/207271

[19]. H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, “A review of optimization algorithms for university timetable scheduling,” Engineering Technology & Applied Science Research, vol. 10, no. 6, pp. 6410–6417, dec 2020.

[20]. S. Abdipoor, R. Yaakob, S. L. Goh, and S. Abdullah, “Meta-heuristic approaches for the university course timetabling problem,” Intelligent Systems with Applications, vol. 19, 2023.

[21]. S. Ceschia, L. D. Gaspero, and A. Schaerf, “Educational timetabling: Problems, benchmarks, and state-of-the-art results,” European Journal of Operational Research, vol. 308, no. 1, pp. 1–18, 2023.

[22]. Y. Chen, M. Bayanati, M. Ebrahimi, and S. Khalijian, “A novel optimization approach for educational class scheduling with considering the students and 49 teachers’ preferences,” Discrete Dynamics In Nature And Society, vol. 2022, no. 1, jan 2022.

[23]. C. S. I. Alberto. (2021, nov) Aplicación web para la automatización y gestión de horarios de clases mediante algoritmos basados en el modelo de programación lineal en la escuela Lic. Angélica Villón Lindao. [En línea]. [Online]. Available: https://repositorio.upse.edu.ec/handle/46000/6485

[24]. O. C. A. David. (2021) Comparación de métodos de optimización para la generación de horarios para personal de enfermería. [En línea]. [Online]. Available: https://repositorio.escuelaing.edu.co/handle/001/1536

[25]. P. O. J. Ricardo. (2021, mar) Implementación de un sistema de asignación automática y gestión de aulas en el campus sur de la universidad politécnica salesiana. [En línea]. [Online]. Available: https://dspace.ups.edu.ec/handle/123456789/19860

[26]. (2021) A novel approach to the optimization of a public bus schedule using k-means and a genetic algorithm. [En línea]. [Online]. Available: https://ieeexplore.ieee.org/document/9431184

[27]. (2024) Timetabling problems and the effort toward generic algorithms: A comprehensive survey. [En línea]. [Online]. Available: https://ieeexplore. ieee.org/document/10684187

[28]. (2019) Comprehensive optimization of a metro timetable considering passenger waiting time and energy efficiency. [En línea]. [Online]. Available: https://ieeexplore.ieee.org/document/8888256

[29]. «Aplicaciones Web 2.ª edición», *Google Books*. https://books.google.es/books?hl=es&lr=&id=43G6AwAAQBAJ&oi=fnd&pg=PA1&dq=que+es+Aplicaci%C3%B3n+web&ots=Di7doZv2CH&sig=CZedKaSz4LymRE77Ru0W5X2XU3Q#v=onepage&q=que%20es%20Aplicaci%C3%B3n%20web&f=false

[30]. «Backend: ¿Qué es y para qué sirve?», *Gluo*. https://www.gluo.mx/blog/backend-que-es-y-para-que-sirve

[31]. «Front End frente a back-end: diferencia entre el desarrollo de aplicaciones - AWS», *Amazon Web Services, Inc.* https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/

[32]. «What Is a Database?», 24 de noviembre de 2020. https://www.oracle.com/mx/database/what-is-database/
[33]. Arsys, «¿Qué es un framework en programación y para qué sirve?», *Arsys*, 2 de marzo de 2025. https://www.arsys.es/blog/que-es-un-framework-en-programacion-y-para-que-sirve
[34]. Ibm, «API REST», *¿Qué es una API REST?*, 15 de abril de 2025. https://www.ibm.com/mx-es/think/topics/rest-apis#:~:text=Una%20API%20REST%20(API%20RESTful,componentes%20en%20arquitecturas%20de%20microservicios.
[35]. «¿Qué es un ORM?», *EDteam - En Español Nadie Te Explica Mejor*. https://ed.team/blog/que-es-un-orm
[36]. Atlassian, «Qué es el control de versiones | Atlassian Git Tutorial», *Atlassian*. https://www.atlassian.com/es/git/tutorials/what-is-version-control
[37]. A. Navarro, «Contenedores vs Entornos Virtuales: ¿Cuál usar y cuándo?», *Junco TIC*, 24 de abril de 2025. https://juncotic.com/contenedores-vs-entornos-virtuales-cual-usar-y-cuando/
[38]. T. Portal, «Servidores», *TIC Portal*, 4 de diciembre de 2024. https://www.ticportal.es/glosario-tic/servidores
[39]. G. C. Cuadrado, «Qué es un lenguaje de programación: Guía para principiantes», *OpenWebinars.net*, 6 de septiembre de 2024. https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/
[40]. E. Admin, «Metodología Agile en ONGs - Fundación Exit», *FundacioExit*, 16 de abril de 2024. https://fundacionexit.org/metodologia-agile-en-ongs/?gad_source=1&gad_campaignid=21225288716&gclid=Cj0KCQjwxJvBBhDuARIsAGUgNfgW8Qr_PS9ODhX8VI4bPw7Y1Jsxs0RBWzI6_M5RtkKtyRMWakXD1JQaAv1DEALw_wcB
[41]. «Vista de Cómo validar una entrevista de preguntas abiertas: una propuesta para investigación filosófica empírica». https://sabereseducativos.uchile.cl/index.php/RSED/article/view/71389/737099
[42]. E. G. Sama Retis y J. L. Vergara Palomino, *Algoritmo genético para la elaboración de horarios de clases en la Universidad Nacional Agraria de la Selva*, Universidad Nacional Agraria de la Selva, Tingo María, Perú, 2025. [En línea]. Disponible en: https://repositorio.unas.edu.pe/items/0f8f0868-92e8-4e49-ab51-7abcb4abaa0f
[43]. Q. A. M. Alejandro, «Modelo de programación lineal entera para la asignación de horarios de clase compactos en la facultad de ciencias de la escuela politécnica nacional.», 1 de mayo de 2022. https://bibdigital.epn.edu.ec/handle/15000/22863
[44]. «¿Qué es la metodología ágil? ¿Para qué sirve?», *Zendesk*, 14 de febrero de 2023. https://www.zendesk.com.mx/blog/metodologia-agil-que-es/
[45]. M. N. López, «¿Qué son las herramientas tecnológicas?», *Campus Training*, 8 de abril de 2025. https://www.campustraining.es/noticias/que-son-herramientas-tecnologicas/
[46]. B. Schneier, *Schneier's Cryptography Classics Library : Applied Cryptography, Secrets and Lies, and Practical Cryptography*. 2007. [En línea]. Disponible en:

https://openlibrary.org/books/OL10295122M/Schneier's_Cryptography_Classics_Library

ANEXOS

Anexo A. Cuestionario de la entrevista sobre la comparación de proceso manual de creación de horarios y creación automatizada

1. ¿Qué aspectos del sistema actual para la asignación de horarios considera útiles?
2. ¿Cuáles son las principales limitaciones o problemas que enfrenta al utilizar el sistema actual?
3. ¿Cuánto tiempo, en promedio, le toma realizar o ajustar un horario con el sistema actual?
4. ¿Qué funcionalidades le gustaría agregar o mejorar en el sistema actual?
5. ¿Qué tipo de reportes o estadísticas le gustaría obtener de un sistema de asignación de horarios?
6. ¿Qué herramientas o características cree que deberían ser prioritarias en el nuevo sistema?
7. En su experiencia, ¿cuáles son las principales diferencias entre crear horarios manualmente y utilizando un sistema automatizado?
8. ¿Cómo considera que un sistema automatizado podría reducir significativamente el tiempo invertido en la creación de horarios y por qué?
9. ¿Qué ventajas cree que tendría un sistema optimizado frente al método manual?
10. ¿Qué tan importante considera que el sistema sea preciso en cumplir con restricciones como no solapar horarios o maximizar el uso de aulas?
11. Si el sistema automatizado propusiera soluciones alternativas, ¿le parecería útil poder elegir entre varias opciones?

12. ¿Qué nivel de interacción le gustaría tener con el sistema al momento de generar horarios? Por ejemplo, ajustes manuales tras la optimización.

13. ¿Qué tan satisfecho está actualmente con el proceso de asignación de horarios?

14. Si pudiera implementar una sola mejora en el proceso actual, ¿cuál sería?

15. ¿Hay algo más que le gustaría mencionar sobre su experiencia con los sistemas de asignación de horarios?

16. ¿Cómo cree que un nuevo sistema de optimización impactaría su trabajo y el de sus colegas?

Anexo B. Entrevista sobre el proceso de gestión de horarios

Información General

- Entrevistado: Ing. Sebastián González
- Fecha de la Entrevista: 10 de marzo, 2025
- Duración: 09:49 minutos
- Lugar: Administración Redes Bloque 1.

Objetivo

El objetivo de esta entrevista es identificar los problemas y desafíos que enfrentan los usuarios al utilizar el sistema actual de asignación de horarios, comprendiendo sus necesidades y expectativas para un nuevo sistema de optimización. Esto incluye descubrir las funcionalidades prioritarias que consideran esenciales, analizar su percepción sobre la utilidad de la automatización para mejorar su trabajo y evaluar las diferencias en eficiencia, tiempo y precisión entre los métodos manuales y los automatizados.

Metodología

La metodología utilizada en esta entrevista se basó en preguntas específicas dirigidas a obtener información detallada sobre los procesos dentro de un estudio de pertinencia de los programas académicos. Además, la entrevista sigue un enfoque semiestructurado con el fin de recolectar toda la información posible y necesaria.

Transcripción de la entrevista

Tabla 18: Entrevista sobre la gestión de horarios

Nº	Pregunta	Respuesta	Observación
1	¿Qué aspectos del sistema actual consideras que son útiles para la asignación de horarios?	La parte de cómo se permite en este caso la generación de las datas por bloque, y también de manera individual, es lo que rescataría del sistema actual.	
2	¿Cuáles crees que son las principales limitaciones que enfrenta el sistema actual?	Que no permite evidenciar si el docente tiene alguna ocupación en cierta hora, y que al estudiante le toca estar escalando sistemas de terceros para saber cierta información.	
3	¿Cuánto tiempo, en promedio, le toma realizar o ajustar un horario con el sistema actual?	Yo diría que unos 10 a 15 minutos. & Refiriéndose a modificar algún horario ya creado	Refiriéndose a modificar algún horario ya creado
4	¿Qué funcionalidades le gustaría agregar o mejorar en el sistema actual?	Me gustaría que exista una impresión por medio de horas, o por medio de una manera más flexible, ya sea que existe una orden de que solo va a haber media jornada o algo, y con ello se evita el desperdicio de hojas y también de generación de actos que no se van a llenar. Otra sería que haya la funcionalidad de ingreso por usuarios, no solo por uno como es actualmente.	
5	¿Qué tipo de reportes o estadísticas le gustaría obtener de un sistema de asignación de horarios?	Reportes como visualización de número de aulas y laboratorios	
6	¿Qué herramientas o características cree que deberían ser prioritarias en el nuevo sistema?	Por ejemplo, que nos pasa todo el tiempo esa consulta para saber dónde están los centros, o sea, como dices, si las aulas están completamente llenas. Ese tipo de consultas sería viable, o sea,	

		simplemente tener un apartado y con eso tener toda la información de generación de actas, de edición de materias y docentes, y también de consultas para saber si esta aula docente está ocupada o está disponible.	
7	En toda su experiencia con el sistema, ¿cuáles son las principales diferencias entre, o cuáles cree que serían las principales diferencias entre crear los horarios manualmente y utilizando el sistema automatizado?	El sistema automatizado permite la generación por días, y con ello solo se selecciona el día, la fecha, y ya se genera por lo que eso trae el sistema de la base de datos. Y la generación manual en cambio permite la generación por hora, materia, aula y docente, que eso es como algo que se genera a última hora en caso de que haya algún cambio o una modificación que el docente necesite un cambio.	
8	¿Cómo considera que un sistema automatizado podría reducir significativamente el tiempo invertido en la creación de horarios y por qué?	Podría tomarse desde una manera más flexible, ya sea al momento de la generación para la impresión, como dije, así es que permite a su vez generar desde cierta franja horaria o desde otras variables que se podrían implementar hacia lo que es la generación de horarios de manera automática. En la generación manual sería algo más amigable que se genere en este caso por medio de solo selección y no sea llenado para evitar en este caso a veces errores de tipeo y una sobre escritura.	
9	¿Qué ventajas cree que tendría un sistema optimizado frente al método manual?	Una ventaja significativa vendría a ser la variable tiempo, ya que se ahorra	

		demasiado tiempo en la generación en bloque, a diferencia de la generación manual, si nos vamos de la generación de todo un día de trabajo.	
10	¿Qué tan importante considera que el sistema sea preciso en cumplir con restricciones como no solapar horarios o maximizar el uso de aulas?	Podría ser, en este caso, teniendo en cuenta que se evita la sobre escritura al momento que por obvio se necesite modificar algo dentro del sistema, se evita la sobre escritura y con ello haya la confusión momentánea de la generación del bloque de las actas.	
11	Si el sistema automatizado propusiera soluciones alternativas, ¿le parecería útil poder elegir entre varias opciones?	Sí, sería bastante significativo esa idea de equipamiento.	
12	¿Qué nivel de interacción le gustaría tener con el sistema al momento de generar horarios? Por ejemplo, ajustes manuales tras la optimización.	Yo pienso que sería lo más cómodo dado hacia el usuario, ya que lo que aquí se trata de llegar a una finalidad es que se evite la menor inversión de tiempo al momento de estar generando ideas a las actas o implementando un cambio en el sistema. Así que en mi opinión sería lo más automatizado posible, sea por medio de selección, sea por medio de carga automática, de preseleccionados.	
13	¿Qué tan satisfecho está actualmente con el proceso de asignación de horarios?	¿Todo el proceso de creación, asignación a los docentes, las aulas? Yo diría un 3 sobre 10.	
14	Si pudiera implementar una sola mejora en el proceso actual, ¿cuál sería?	Automatización al momento de la generación manual. Eso es demasiado tedioso y hay una pérdida significativa de tiempo.	

15	¿Hay algo más que le gustaría mencionar sobre su experiencia con los sistemas de asignación de horarios?	Que actualmente pienso que se podría, por medio de un proyecto de investigación o tesis, llegar a un nivel de calidad similar a los que he manejado en empresas, porque aquí pienso que hay la creatividad, la investigación y los componentes para llegar a ese nivel de calidad y desempeño de los sistemas.	
16	¿Cómo cree que un nuevo sistema de optimización impactaría su trabajo y el de sus colegas?	Pienso que sería bastante significativo porque para mí y mis compañeros ocupamos el sistema para generación y con ellos llevaría menos tiempo la edición y también la generación sería algo bastante significativo.	
Conclusiones: La entrevista realizada muestra que, aunque el sistema actual cumple algunas funciones, no satisface completamente las necesidades del personal y docentes. La automatización se percibe como clave para optimizar tiempo, reducir errores y mejorar la precisión, mientras que la posibilidad de personalizar horarios y generar reportes detallados es prioritaria.			

Anexo C. Entrevista sobre el proceso de gestión de horarios

Información General

- Entrevistado: Ing. Santiago Jara
- Fecha de la Entrevista: 10 de marzo, 2025
- Duración: 08:54 minutos
- Lugar: Administración Redes Bloque 2.

Objetivo

El objetivo de esta entrevista es identificar los problemas que enfrentan los usuarios al utilizar el sistema actual de asignación de horarios. Analizando su percepción sobre la utilidad de la automatización para mejorar su trabajo y evaluar las diferencias en eficiencia, tiempo y precisión entre los métodos manuales y los automatizados en la creación de horarios. Además, se busca recopilar retroalimentación para medir el nivel de aceptación hacia la propuesta de un sistema automatizado.

Metodología

La metodología utilizada en esta entrevista se basó en preguntas específicas dirigidas a obtener información detallada sobre los procesos dentro de un estudio de pertinencia de los programas académicos. Además, la entrevista sigue un enfoque semiestructurado con el fin de recolectar toda la información posible y necesaria.

Transcripción de la entrevista

Tabla 19: Entrevista sobre el proceso de gestión de horarios

Nº	Pregunta	Respuesta	Observación
1	¿Qué aspectos del sistema actual consideras que son útiles para la asignación de horarios?	Consultas de profesores, de laboratorios, aulas. Eso, y obviamente toda la parte de gestión, que es el ingreso de profesores, ingreso de materias, horarios. Eso, es todo lo que tiene que es importante.	
2	¿Cuáles crees que son las principales limitaciones que enfrenta el sistema actual?	El sistema fue desarrollado en una aplicación con Flutter, pero en la parte web también lo hicieron con Flutter. Y ahí es donde daba bastantes inconvenientes. Por ejemplo, si no me equivoco, era uno de los problemas de asignar un horario. Queda entre docente y materia. Entonces ahí les botaba bastantes errores. No permitía a veces modificar, a veces eliminaba, pero no se reflejaba eso en la base de datos. Entonces sí tiene algunos problemas. Y también no me da recomendaciones de horarios.	
3	¿Cuánto tiempo, en promedio, le toma realizar o ajustar un horario con el sistema actual?	En cuanto al tiempo. Modificar un horario. Yo creo que, de unos 3 minutos a 5 minutos, digamos.	
4	¿Qué funcionalidades le gustaría agregar o mejorar en el sistema actual?	Inteligencia artificial. Para que me pueda recomendar los horarios. Sería bueno que me recomienda los horarios en base a algunas restricciones o recomendaciones que nos dan los mismos docentes. O restricciones en sí de cuando son dos	

		paralelos. Sí, varias restricciones. Sería bueno que me recomienda horarios en base a restricciones que se debe al sistema.	
5	¿Qué tipo de reportes o estadísticas le gustaría obtener de un sistema de asignación de horarios?		No contesto la pregunta
6	¿Qué herramientas o características cree que deberían ser prioritarias en el nuevo sistema?	Herramientas o características. Creo que sería lo que ya viene mencionando. Adicionales. Inteligencia artificial. Con recomendaciones de horarios. Características. Más que características, las mismas características que ya existían, pero sin problemas. Que no haya esos problemas. Sería muy importante probar bien esto antes de hacer la producción. En cuanto a las restricciones para crear los horarios, tanto para asignar a los docentes y aulas.	
7	En toda su experiencia con el sistema, ¿cuáles son las principales diferencias entre, o cuáles cree que serían las principales diferencias entre crear los horarios manualmente y utilizando el sistema automatizado?	Bastante, pues. Es un apoyo grande porque la elaboración de los horarios en vacaciones. Lleva aproximadamente unas 3 semanas. Que la creación y todo. Unas 3 semanas aproximadamente me llevo haciendo los horarios. Con cambios y todo. Entonces sí es complejo. Sería bueno que un sistema que me recomiende ya horarios o que me dé haciendo eso. Sería éxito. Solo voy a ir revisando partes importantes que tenga que cambiar.	Refiriéndose a la creación de horarios, asignación de horarios.
8	¿Cómo considera que un sistema automatizado podría reducir	Disminuiría demasiado	

	significativamente el tiempo invertido en la creación de horarios y por qué?		
9	¿Qué ventajas cree que tendría un sistema optimizado frente al método manual?	Creo que sería lo que ya viene mencionando. Adicionales. Inteligencia artificial. Con recomendaciones de horarios. Características. Más que características, las mismas características que ya existían, pero sin problemas. Que no haya esos problemas. Sería muy importante probar bien esto antes de hacer la producción. En cuanto a las restricciones para crear los horarios, tanto para asignar a los docentes y aulas.	
10	¿Qué tan importante considera que el sistema sea preciso en cumplir con restricciones como no solapar horarios o maximizar el uso de aulas?	Tiene que ser súper importante. 100\%. Tiene que ser súper importante la precisión de las limitaciones o restricciones que se debe a ese sistema.	
11	Si el sistema automatizado propusiera soluciones alternativas, ¿le parecería útil poder elegir entre varias opciones?	Recomendaciones de varias opciones. Eso es importante para que los mismos coordinadores realicen esa revisión. Porque ellos son los que aprueban y los que suben los horarios del sistema de la UTA. Entonces, ellos revisarían las recomendaciones, las opciones que mide el sistema, y verían la más adecuada. Y esa sería la versión final para verificar ese sistema. Bueno, para terminar la entrevista.	
12	¿Qué nivel de interacción le gustaría tener con el sistema al momento de generar horarios? Por ejemplo, ajustes manuales tras la optimización.	Muy poco	

13	¿Qué tan satisfecho está actualmente con el proceso de asignación de horarios?	Intermedio. Mejorable	
14	Si pudiera implementar una sola mejora en el proceso actual, ¿cuál sería?	Recomendación de horarios	
15	¿Hay algo más que le gustaría mencionar sobre su experiencia con los sistemas de asignación de horarios?	Creo que no. Lo que sí faltó mencionar durante toda la entrevista fue que este sistema debe generar las hojas para los laboratorios. Que eso también forma parte de los horarios. En base al horario debe generar la hoja de uso de los laboratorios. Eso también es súper importante dentro del sistema. En cuanto a experiencia, no. La parte de las limitaciones es lo más complicado en el desarrollo de horarios. Porque si no, sería más fácil desarrollar. Los horarios se asignarían fácilmente. Horario, materia y demás. Y no se tendría que estar modificando constantemente el horario. Ese es el problema. Las limitaciones que ponen los docentes en el momento de revisar su horario.	Se desvía temporalmente de la pregunta.
16	¿Cómo cree que un nuevo sistema de optimización impactaría su trabajo y el de sus colegas?	Habría una buena aceptación por parte de todos. Porque va a minimizar bastante los tiempos. El simple hecho de que nos den varias recomendaciones. Nos dan varias recomendaciones de horarios. Entonces es más fácil. Lo único que tiene que hacer es revisar, no hacer. Revisan, aprueban y listo. Entonces van a	

		tener una buena aceptación.	
Conclusiones: El sistema actual presenta limitaciones críticas, como errores en asignaciones y falta de recomendaciones automáticas. Se destaca la necesidad de implementar inteligencia artificial para optimizar horarios, generar reportes detallados y garantizar precisión en restricciones. Esto reduciría tiempos, mejoraría la experiencia del usuario y aumentaría la eficiencia del proceso administrativo y docente.			

Anexo D. Entrevista sobre el proceso de gestión de horarios

Información General

- Entrevistado: Ing. Natasha Villacis
- Fecha de la Entrevista: 10 de marzo, 2025
- Duración: 07:34 minutos
- Lugar: Administración Redes Bloque 1.

Objetivo

El objetivo de esta entrevista es identificar los problemas y desafíos que enfrentan los usuarios al utilizar el sistema actual de asignación de horarios, comprendiendo sus necesidades y expectativas para un nuevo sistema de optimización. Esto incluye descubrir las funcionalidades prioritarias que consideran esenciales, analizar su percepción sobre la utilidad de la automatización para mejorar su trabajo y evaluar las diferencias en eficiencia, tiempo y precisión entre los métodos manuales y los automatizados.

Metodología

La metodología utilizada en esta entrevista se basó en preguntas específicas dirigidas a obtener información detallada sobre los procesos dentro de un estudio de pertinencia de los programas académicos. Además, la entrevista sigue un enfoque semiestructurado con el fin de recolectar toda la información posible y necesaria.

Transcripción de la entrevista

Tabla 20: Entrevista sobre el proceso de gestión de horarios

Nº	Pregunta	Respuesta	Observación
1	¿Qué aspectos del sistema actual consideras que son útiles para la asignación de horarios?	Actualmente, con el sistema de horarios que manejamos, lo rescatable de ese es que la interfaz es un poco intuitiva y también está ordenada.	
2	¿Cuáles crees que son las principales limitaciones que enfrenta el sistema actual?	El sistema actual no permite filtrar diferentes horarios, como por ejemplo por horas, por bloque o individuales. Esa es una limitación muy grande, ya que, si se quiere filtrar por alguna de las anteriores respuestas, no se puede hacer.	
3	¿Cuánto tiempo, en promedio, le toma realizar o ajustar un horario con el sistema actual?	Pongo una de media hora.	
4	¿Qué funcionalidades le gustaría agregar o mejorar en el sistema actual?	Podemos mejorar la interfaz, que sea un poco más amigable para el usuario. Que haya los filtros que ya mencioné anteriormente. Y la generación de actas por bloque e individual, que eso es demasiado necesario. Y los filtros que ya lo dije.	
5	¿Qué tipo de reportes o estadísticas le gustaría obtener de un sistema de asignación de horarios?	Bueno, reporte sería eficiente, ya que de esa manera podríamos ver si los horarios chocan entre los profesores. Y eso sí sería muy eficiente.	
6	¿Qué herramientas o características cree que deberían ser prioritarias en el nuevo sistema?	En el nuevo sistema. Los filtros, por horas.	
7	En toda su experiencia con el sistema, ¿cuáles son las principales diferencias entre, o	La gran diferencia entre crear un horario manualmente y un automatizado es el	

	cuáles cree que serían las principales diferencias entre crear los horarios manualmente y utilizando el sistema automatizado?	tiempo. Porque, en este caso, nosotros generamos los horarios actualmente manuales, entonces nos lleva demasiado tiempo. Y, en este caso, automatizando sería por bloque, como se lo dijo anteriormente, lo que nos conlleva recortar tiempo una barbaridad.	
8	¿Cómo considera que un sistema automatizado podría reducir significativamente el tiempo invertido en la creación de horarios y por qué?	Bueno, en este caso, algún sistema automatizado eficiente realmente recorta mucho tiempo en cuanto a trabajo. Entonces, sí, yo digo que cualquier sistema automatizado eficiente es útil en cualquier área de trabajo. Y, por ejemplo, en esta que es admin redes, nos facilita demasiado.	
9	¿Qué ventajas cree que tendría un sistema optimizado frente al método manual?	Se reduciría mucho tiempo invertido	
10	¿Qué tan importante considera que el sistema sea preciso en cumplir con restricciones como no solapar horarios o maximizar el uso de aulas?	Muy importante	
11	Si el sistema automatizado propusiera soluciones alternativas, ¿le parecería útil poder elegir entre varias opciones?	Sí, porque a veces se necesita sólo el horario de la mañana, por ejemplo. Entonces, sólo imprimir de 7 a 12 de la mañana fuera muy eficiente. Como, por ejemplo, igual de la tarde, sería de 2 a 6 de la tarde.	Se desvía temporalmente de la pregunta
12	¿Qué nivel de interacción le gustaría tener con el sistema al momento de generar horarios? Por ejemplo, ajustes manuales tras la optimización.	Que no exista mucha interacción	
13	¿Qué tan satisfecho está actualmente con el	De 1 a 10, son 3.	

	proceso de asignación de horarios?		
14	Si pudiera implementar una sola mejora en el proceso actual, ¿cuál sería?	Generación por bloque individual.	
15	¿Hay algo más que le gustaría mencionar sobre su experiencia con los sistemas de asignación de horarios?	Como ya lo dije anteriormente, creo que es bueno implementar todos los filtros que acabo de mencionar.	Se desvía temporalmente de la pregunta.
16	¿Cómo cree que un nuevo sistema de optimización impactaría su trabajo y el de sus colegas?	Yo pienso que impactaría demasiado, ya que eso disminuiría lo que viene a ser la carga laboral. El tiempo que nos conlleva en hacer un horario manual, 2. Y el tratar de controlar mucho más lo que vienen a ser las horas entre los docentes y que no choquen.	
Conclusiones: Con base en la entrevista realizada, se concluye que el sistema actual de asignación de horarios presenta limitaciones significativas que afectan la eficiencia del proceso, especialmente en cuanto a la falta de filtros y opciones personalizables. Los entrevistados valoran la posibilidad de mejorar la interfaz, reducir el tiempo requerido para generar horarios y contar con reportes detallados que permitan evitar conflictos entre los horarios de los docentes. Se considera que un sistema automatizado y optimizado podría reducir significativamente la carga laboral, mejorar la precisión, cumplir con restricciones clave y ofrecer soluciones flexibles, impactando positivamente tanto en el tiempo invertido como en la satisfacción general del personal administrativo y docente.			

Anexo E. Entrevista de levantamiento de requerimientos

Requerimientos Funcionales

RF01: Gestión de Carreras(CRUD)

Descripción: El sistema debe permitir al administrador agregar, consultar, actualizar y eliminar las diferentes carreras existentes en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Preguntas Planteadas:

1. ¿Qué campos son necesarios para la creación de una carrera?
2. ¿La creación de una carrera depende de algún factor externo?

RF02: Gestión de Semestres(CRUD)

Descripción: El sistema debe permitir al administrador agregar, consultar, actualizar y eliminar los diferentes semestres que pueden existir dentro de una carrera de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Preguntas Planteadas:

3. ¿Qué campos son necesarios para la creación de un semestre? (Nombre, descripción, etc.)

RF03: Gestión de Materias(CRUD)

Descripción: El sistema debe permitir al administrador agregar, consultar, actualizar y eliminar todas las materias existentes dentro de la facultad, incluyendo actividades especiales.

Preguntas Planteadas:

4. ¿Qué campos son necesarios para la creación de una nueva materia?
5. ¿Cómo se podría distinguir una materia académica de una especial?

RF04: Gestión de Periodos Académicos(CRUD)

Descripción: El sistema debe permitir la administración de los diferentes periodos académicos en los que se organiza la actividad académica de la facultad. Esto implica que el administrador podrá crear, visualizar, modificar y eliminar los periodos, definiendo sus fechas de inicio y fin, y cualquier otra información relevante para su identificación.

Preguntas Planteadas:

6. ¿Qué información específica se necesita definir para un periodo académico? (Por ejemplo, nombre o código del periodo, fecha de inicio, fecha de fin)

7. ¿Qué información adicional podría ser relevante asociar a un periodo académico (Por ejemplo, estado activo/inactivo)?

8. ¿Cómo se gestionará la transición entre periodos académicos en el sistema? ¿Se requerirá algún proceso específico para "cerrar" un periodo y "abrir" uno nuevo?

RF05: Gestión de Edificios(CRUD)

Descripción: El sistema debe permitir la administración de los edificios que componen la infraestructura de la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar la información de cada edificio. Esta gestión será fundamental para la posterior asignación de aulas y laboratorios.

Preguntas Planteadas:

9. ¿Qué atributos son necesarios para definir un edificio?

10. ¿Cómo se identificarán de forma única los edificios dentro del sistema? ¿Se utilizará un código interno o se basará en su nombre?

RF06: Gestión de Pisos(CRUD)

Descripción: El sistema debe permitir la administración de los pisos que componen cada edificio de la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar la información de cada piso.

Preguntas Planteadas:

11. ¿Qué atributos son necesarios para definir un piso?

12. ¿Cómo se identificarán de forma única los pisos dentro de un edificio? ¿Se utilizará un número secuencial o una designación alfanumérica?

RF07: Gestión de Laboratoristas(CRUD)

Descripción: El sistema debe permitir la administración de la información de los laboratoristas de la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar los datos personales y profesionales de cada laboratorista, incluyendo la asignación de su título profesional (RF11) y su vinculación con algún edificio específico.

Preguntas Planteadas:

13. ¿Qué campos son necesarios para registrar la información básica de un laboratorista (por ejemplo, nombres, número de identificación)?

RF08: Gestión de Docentes(CRUD)

Descripción: El sistema debe permitir la administración de la información de los docentes de la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar los datos personales y profesionales de cada docente, incluyendo la asignación opcional de su título profesional (según RF11).

Preguntas Planteadas:

14. ¿Qué campos son necesarios para registrar la información básica de un docente (por ejemplo, nombres, apellidos, número de identificación, información de contacto)?

15. ¿Se necesita registrar información adicional específica para los docentes, como su área de especialización o información de contacto adicional?

16. ¿Necesitamos capturar información adicional sobre la vinculación de un docente con una materia o un curso en particular?

RF09: Gestión de Software(CRUD)

Descripción: El sistema debe permitir la administración del software disponible en la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar la información de cada software, incluyendo su nombre, versión, licencia (si aplica), y cualquier otra información relevante para su gestión.

Preguntas Planteadas:

17. ¿Qué campos son necesarios para registrar la información de un software (por ejemplo, nombre del software, versión)?

RF10: Gestión de Aula(CRUD)

Descripción: El sistema debe permitir la administración de las aulas disponibles en la facultad. Esto implica que el administrador podrá registrar, visualizar, modificar y eliminar la información de cada aula, incluyendo su identificación, capacidad, ubicación (piso y edificio), y el tipo de aula que es (por ejemplo, aula, laboratorio, especial).

Preguntas Planteadas:

18. ¿Qué campos son necesarios para registrar la información básica de un aula (por ejemplo, número o código del aula, capacidad máxima de estudiantes)?

19. ¿Se requiere información sobre el equipamiento fijo de cada aula (por ejemplo, proyector, pizarra interactiva)? Si es así, ¿cómo se gestionaría esta información (en la misma tabla de aulas o en una tabla separada)?

RF11: Gestión de Maquinas(CRUD)

Descripción: El sistema debe permitir la administración de las máquinas o equipos específicos ubicados en cada aula o laboratorio de la facultad. Esto implica que el administrador podrá especificar, visualizar, modificar y eliminar la asignación de máquinas a cada espacio. Esta funcionalidad permitirá conocer el inventario de equipos disponibles en cada aula para la planificación de actividades académicas.

Preguntas Planteadas:

20. ¿Qué información es necesaria registrar sobre cada máquina (por ejemplo, tipo de máquina, modelo, número de serie, fecha de adquisición, estado de funcionamiento)?

21. ¿Es posible que una misma máquina (del catálogo general) se encuentre en varias aulas?

RF12: Gestión de Horarios(CRUD)

Descripción: El sistema debe permitir la creación, visualización, modificación y eliminación de los horarios de clases y otras actividades académicas. Esta funcionalidad central integrará información de carreras, semestres, materias, docentes, aulas, periodos académicos y posiblemente otros recursos, permitiendo la planificación y consulta de la distribución temporal de las actividades de la facultad. Dada su complejidad, se anticipan múltiples consultas para generar las vistas de horarios necesarias.

Preguntas Planteadas:

22. ¿Qué campos específicos necesitamos para definir una entrada de horario (por ejemplo, identificador único de la clase, día de la semana, hora de inicio, hora de fin)?

23. ¿Cómo se gestionará la asignación de un docente a una materia en un semestre específico y en un aula particular dentro de un periodo académico?

24. ¿Se permitirá la asignación de múltiples docentes a una misma clase o la división de una clase en diferentes horarios o aulas?

25. ¿Cómo se manejarán las restricciones de capacidad de las aulas al momento de asignar horarios?

26. ¿Qué criterios de validación se aplicarán al crear un horario para evitar conflictos (por ejemplo, un docente asignado a dos clases al mismo tiempo, un aula reservada para dos actividades simultáneamente)?

27. ¿Se requerirán funcionalidades para la generación automática de horarios o sugerencias basadas en la disponibilidad de recursos?

28. ¿Se necesitará llevar un historial de los horarios o versiones anteriores?

RF13: Gestión de Usuarios (CRUD)

Descripción: El sistema debe permitir la administración de las cuentas de usuario que accederán a la aplicación. Esto implica que el administrador principal podrá crear, visualizar, modificar y eliminar las cuentas de usuario, asignándoles un rol específico que determinará sus permisos y funcionalidades dentro del sistema.

Preguntas Planteadas:

29. ¿Qué campos son necesarios para crear una cuenta de usuario (por ejemplo, nombre de usuario, contraseña, correo electrónico, nombre completo)?¿Cómo se gestionará la asignación de un docente a una materia en un semestre específico y en un aula particular dentro de un periodo académico?

30. ¿Cómo se gestionará la asignación de roles a los usuarios? ¿Se utilizará un campo desplegable con los roles predefinidos ("admin", "user")? ¿Se prevén otros roles en el futuro?

31. ¿Se implementarán mecanismos de seguridad para la gestión de contraseñas (por ejemplo, requisitos de complejidad, almacenamiento cifrado)?

32. ¿Se requerirá un proceso de autenticación (login) para acceder al sistema?

33. ¿Se llevará un registro de la actividad de los usuarios (logs de acceso, modificaciones realizadas)?

34. ¿Se necesitará la capacidad de asignar permisos más granularmente a los usuarios más allá de los roles predefinidos?

35. ¿Existirá algún proceso para la desactivación o bloqueo de cuentas de usuario?

36. ¿Cómo se gestionará la recuperación de contraseñas olvidadas?

RF14: Generación de Hojas de Control

Descripción: El sistema debe permitir la generación de hojas de control para ciertas aulas y laboratorios los cuales cuentan con ordenadores o máquinas electrónicas. Esto implica que el administrador o usuario normal podrá generar hojas utilizadas para el registro de asistencia de estudiantes y registro de observaciones de una maquina dentro de una aula o laboratorios.

Preguntas Planteadas:

37. ¿Qué tipos de aulas son necesarias para la generación de estas hojas de control?

38. ¿Cómo sería el proceso de generación de hojas de control?

Requerimientos No Funcionales

RNF01: Seguridad de la información

Descripción: Todos los datos personales y sensibles almacenados en la base de datos deben estar encriptados para garantizar la seguridad y privacidad de la información.

Preguntas Planteadas:

1. ¿Qué algoritmos de encriptación específicos se deben utilizar para proteger los datos sensibles en la base de datos? ¿Existen regulaciones o estándares específicos que debamos cumplir en cuanto a los métodos de encriptación?

2. ¿En qué momento del ciclo de vida de los datos se realizará la encriptación (por ejemplo, al ser almacenados en la base de datos, durante la transmisión)?

3. ¿Se requerirá algún mecanismo de gestión de claves de encriptación? ¿Cómo se almacenarán y rotarán estas claves de forma segura?

RNF02: Diseño del Sistema

Descripción: La interfaz del sistema deberá respetar y contener los colores y logos institucionales de la universidad y la facultad.

Preguntas Planteadas:

4. ¿Cuáles son los colores institucionales específicos de la universidad y la facultad que deben incorporarse en la interfaz del sistema (proporcionar códigos de color exactos como hexadecimal, RGB)?

5. ¿Dónde y cómo se deben utilizar estos colores dentro de la interfaz (por ejemplo, barra de navegación, encabezados, botones, elementos de énfasis)?

6. ¿Cuáles son los logos institucionales que deben incluirse en el sistema (proporcionar archivos de alta resolución en los formatos requeridos)?

7. ¿En qué ubicaciones específicas de la interfaz se deben mostrar los logos (por ejemplo, encabezado, pie de página, página de inicio)? ¿Existen pautas sobre su tamaño y disposición?

8. ¿Hay alguna otra directriz de diseño institucional que deba seguirse en la interfaz (por ejemplo, tipografía, espaciado, iconografía)? ¿Existen manuales de identidad visual o guías de estilo de la interfaz de usuario (UI) que podamos consultar?

RNF03: Generación de Hojas de Control

Descripción: El sistema debe permitir la generación de hojas de control para ciertas aulas y laboratorios los cuales cuentan con ordenadores o máquinas electrónicas. Esto implica que el administrador o usuario normal podrá generar hojas utilizadas para el registro de asistencia de estudiantes y registro de observaciones de una maquina dentro de una aula o laboratorios.

Preguntas Planteadas:

9. ¿El diseño de las hojas de control deben ser alguno en específico?

10. ¿Es necesario tener más de un formato de hojas de control? Para algún aula o laboratorio en específico.

Requerimientos Funcionales Generación Horarios

RF01: Restricciones

Descripción: Para que se realice la generación de horarios, el sistema debe respetar ciertas restricciones para la generación correcta de los horarios.

Preguntas Planteadas:

1. ¿Quiénes deben respetar restricciones?(Docentes, Materias, etc.)
2. ¿Cuáles son todas las restricciones que el sistema deben tomar en cuenta para generar los horarios?

RF02: Generación de Horarios

Descripción: Para que se realice la generación de horarios, el sistema debe respetar ciertas restricciones para la generación correcta de los horarios.

Preguntas Planteadas:

3. ¿Podría utilizar cualquier algoritmo de optimización?¿Qué algoritmos de optimización podría utilizar para generar los horarios tomando en cuenta las restricciones?
4. ¿De qué manera se debería poder visualizar los horarios generados?
5. ¿Se debería poder cambiar o actualizar los horarios generados?

Anexo F. Levantamiento de requerimientos – IEEE 830

Empleando el estándar IEEE Practica Recomendada para Especificaciones de Requisitos de Software ANSI/IEEE 830, 1998. Se realizo las siguientes especificaciones de requerimientos para el sistema de Gestión de horario y aulas de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial. Cuyo propósito es definir las especificaciones funcionales y no funcionales para el desarrollo del sistema web el cual será utilizado por el personal del área de Administración de Redes de la facultad. A continuación, se detallarán minuciosamente los requisitos que el sistema debe cumplir.

Requisitos Funcionales

Identificador	**RF01**
Nombre	Gestión de Carreras (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de las carreras existentes en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar las carreras académicas, incluyendo la creación de nuevas carreras, la visualización de las existentes, la modificación de sus datos y la eliminación cuando sea necesario.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear una nueva carrera ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de carreras existentes. - El administrador puede actualizar la información de una carrera existente. - El administrador puede eliminar una carrera, previa confirmación.
Observaciones	La creación de una carrera no depende de factores externos, pero debe asegurarse que los campos requeridos estén correctamente definidos.

Identificador	**RF02**
Nombre	Gestión de Semestres (CRUD)

Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de los semestres asociados a las carreras de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar los semestres académicos, incluyendo la creación de nuevos semestres, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear un nuevo semestre ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de semestres existentes. - El administrador puede actualizar la información de un semestre existente. - El administrador puede eliminar un semestre, previa confirmación.
Observaciones	Para la creación de un semestre, se consideran campos como: nombre del semestre, descripción y código identificador.

Identificador	**RF03**
Nombre	Gestión de Materias (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de todas las materias existentes dentro de la facultad, incluyendo actividades especiales.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar las materias académicas y actividades especiales, incluyendo la creación de nuevas materias, la visualización de las existentes, la modificación de sus datos y la eliminación cuando sea necesario.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear una nueva materia ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de materias existentes. - El administrador puede actualizar la información de una materia existente. - El administrador puede eliminar una materia, previa confirmación. - El sistema permite distinguir entre materias

	académicas y actividades especiales mediante un campo específico.
Observaciones	Para la creación de una materia, se consideran campos como: nombre de la materia, código, descripción, número de créditos, tipo de materia (académica o especial), y requisitos previos. La distinción entre una materia académica y una especial se realiza mediante el campo "tipo de materia", donde se especifica si es académica o especial.

Identificador	**RF04**
Nombre	Gestión de Períodos Académicos (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de los diferentes períodos académicos en los que se organiza la actividad académica de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar los períodos académicos, incluyendo la creación de nuevos períodos, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario. Cada período debe incluir información como nombre o código del período, fecha de inicio, fecha de fin y estado (activo/inactivo). Además, el sistema debe permitir la transición entre períodos académicos, asegurando que solo un período esté activo a la vez y que se realicen los procesos necesarios para cerrar un período y abrir uno nuevo.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear un nuevo período académico ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de períodos académicos existentes. - El administrador puede actualizar la información de un período académico existente. - El administrador puede eliminar un período académico, previa confirmación. - El sistema permite marcar un período como activo o inactivo. - El sistema asegura que solo un período académico esté activo a la vez. - El sistema proporciona mecanismos para cerrar un período académico y abrir uno nuevo, asegurando la integridad de los datos y la continuidad de las operaciones académicas.
Observaciones	Para la creación de un período académico, se consideran campos como: nombre o código del período, fecha de inicio, fecha de fin y estado (activo/inactivo). La gestión de la transición

	entre períodos académicos implica procesos específicos para cerrar un período y abrir uno nuevo, asegurando que solo un período esté activo a la vez y que se mantenga la integridad de los datos y la continuidad de las operaciones académicas.

Identificador	**RF05**
Nombre	Gestión de Edificios (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de los edificios que componen la infraestructura de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de los edificios de la facultad, incluyendo la creación de nuevos registros, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario. Esta gestión será fundamental para la posterior asignación de aulas y laboratorios.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear un nuevo edificio ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de edificios existentes. - El administrador puede actualizar la información de un edificio existente. - El administrador puede eliminar un edificio, previa confirmación. - Cada edificio es identificado de forma única mediante un código interno asignado por el sistema.
Observaciones	Para la creación de un edificio, se consideran campos como: nombre del edificio, código interno, ubicación, número de pisos, capacidad total y descripción general. La identificación única de los edificios se realizará mediante un código interno asignado por el sistema, garantizando la unicidad y facilitando su gestión.

Identificador	**RF06**
Nombre	Gestión de Pisos (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de los pisos que componen cada edificio de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar los pisos dentro de cada edificio, incluyendo la creación de

	nuevos registros, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario. Esta gestión es esencial para la correcta asignación y organización de aulas y laboratorios en la infraestructura de la facultad.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear un nuevo piso ingresando los campos requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de pisos existentes en un edificio. - El administrador puede actualizar la información de un piso existente. - El administrador puede eliminar un piso, previa confirmación. - Cada piso es identificado de forma única dentro de su edificio mediante un número secuencial o una designación alfanumérica.
Observaciones	Para la creación de un piso, se consideran campos como: número o designación del piso (por ejemplo, "Piso 1" o "PB"), descripción, y cualquier otra información relevante. La identificación única de los pisos dentro de un edificio se puede realizar mediante un número secuencial ascendente (por ejemplo, 1, 2, 3) o una designación alfanumérica (por ejemplo, "PB" para planta baja, "S1" para sótano 1), según las necesidades de la institución.

Identificador	**RF07**
Nombre	Gestión de Laboratoristas (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de la información de los laboratoristas de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de los laboratoristas, incluyendo la creación de nuevos registros, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario. Además, debe permitir la asignación de títulos profesionales y la vinculación de cada laboratorista con un edificio específico.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	- El administrador puede crear un nuevo registro de laboratorista ingresando los campos

	requeridos. - El sistema valida que todos los campos obligatorios estén completos antes de guardar. - El administrador puede consultar la lista de laboratoristas existentes. - El administrador puede actualizar la información de un laboratorista existente. - El administrador puede eliminar un registro de laboratorista, previa confirmación. - El sistema permite asignar un título profesional a cada laboratorista. - El sistema permite vincular a cada laboratorista con un edificio específico.
Observaciones	Para la creación de un registro de laboratorista, se consideran campos como: nombres, apellidos, número de identificación, correo electrónico institucional, número de teléfono, título profesional (referenciado en RF11) y edificio asignado. La identificación única de cada laboratorista se realizará mediante su número de identificación.

Identificador	**RF08**
Nombre	Gestión de Docentes (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de la información de los docentes de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de los docentes, incluyendo la creación de nuevos registros, la visualización de los existentes, la modificación de sus datos y la eliminación cuando sea necesario. Además, debe permitir la asignación opcional de títulos profesionales y la vinculación de cada docente con materias o cursos específicos.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	
Observaciones	Para la creación de un registro de docente, se consideran campos como: nombres, apellidos, número de identificación, correo electrónico institucional, número de teléfono, título profesional (referenciado en RF11), área de especialización y materias o cursos asignados. La identificación única de cada docente se realizará mediante su número de identificación.

Identificador	**RF09**
Nombre	Gestión de Software (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación

	de la información del software disponible en la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de los programas informáticos utilizados en la facultad. Esto incluye registrar nuevos software, visualizar los existentes, modificar sus datos y eliminarlos cuando sea necesario. La gestión debe contemplar detalles como el nombre del software, versión, tipo de licencia y otra información relevante para su administración.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	
Observaciones	Para la creación de un registro de software, se consideran campos como: nombre del software, versión, tipo de licencia (por ejemplo, libre, comercial, académica), proveedor o desarrollador, fecha de adquisición, fecha de expiración de la licencia (si aplica), y descripción general. La identificación única de cada software se realizará mediante una combinación de su nombre y versión.

Identificador	**RF10**
Nombre	Gestión de Aulas (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de la información de las aulas disponibles en la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de las aulas, incluyendo su identificación, capacidad, ubicación (piso y edificio), tipo de aula (por ejemplo, aula, laboratorio, especial) y equipamiento disponible. Esta gestión es esencial para la correcta asignación y organización de los espacios académicos en la facultad.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	
Observaciones	Para la creación de un registro de aula, se consideran campos como: número o código del aula, capacidad máxima de estudiantes, ubicación (piso y edificio), tipo de aula (por ejemplo, aula, laboratorio, especial) y equipamiento fijo disponible (por ejemplo, proyector, pizarra interactiva). La identificación

	única de cada aula se realizará mediante su número o código. La información sobre el equipamiento fijo puede gestionarse en una tabla separada relacionada con la tabla de aulas para facilitar su administración y mantenimiento.

Identificador	**RF11**
Nombre	Gestión de Máquinas (CRUD)
Características	Permite al administrador realizar operaciones de creación, consulta, actualización y eliminación de la información de las máquinas o equipos específicos ubicados en cada aula o laboratorio de la facultad.
Descripción	El sistema debe proporcionar una interfaz que permita al administrador gestionar la información de las máquinas o equipos, incluyendo su identificación, características técnicas, estado de funcionamiento y ubicación. Esta funcionalidad es esencial para mantener un inventario actualizado y facilitar la planificación de actividades académicas que requieran el uso de estos equipos.
Requerimientos no funcionales	RNF01: La interfaz debe ser intuitiva y de fácil navegación. RNF02: Las operaciones de gestión deben realizarse en un tiempo máximo de 2 segundos.
Prioridad del requerimiento	Alta
Criterios de aceptación	
Observaciones	Para la creación de un registro de máquina, se consideran campos como: tipo de máquina, marca, modelo, número de serie, fecha de adquisición, estado de funcionamiento, ubicación (aula o laboratorio), y observaciones adicionales. La identificación única de cada máquina se realizará mediante su número de serie. En caso de que una misma máquina se encuentre en varias aulas, el sistema debe permitir registrar múltiples ubicaciones asociadas a una sola máquina.

Identificador	**RF12**
Nombre	Gestión de Horarios (CRUD)
Características	Permite la creación, visualización, modificación y eliminación de los horarios de clases y otras actividades académicas, integrando información de carreras, semestres, materias, docentes, aulas, periodos académicos y otros recursos relevantes.
Descripción	El sistema debe permitir la planificación y consulta de la distribución temporal de las actividades académicas de la facultad. Debido a su complejidad, se anticipan múltiples consultas para generar las vistas necesarias de los horarios, asegurando una gestión integral y eficiente.

Requerimientos no funcionales	RNF01: El sistema debe responder con rapidez a las consultas, con un tiempo máximo de respuesta de 3 segundos.RNF02: La interfaz debe ser intuitiva para facilitar la gestión por parte del usuario.RNF03: El sistema debe garantizar la integridad y consistencia de los datos ante operaciones simultáneas.
Prioridad del requerimiento	Alta
Criterios de aceptación	- Permitir la creación de entradas de horario con campos como identificador único, día de la semana, hora de inicio y fin.- Gestionar correctamente la asignación de docentes a materias, aulas y semestres dentro de periodos académicos.- Validar que no existan conflictos de horarios ni sobrecupo de aulas.- Permitir asignación de múltiples docentes a una misma clase si es necesario.- Soportar historial de versiones de horarios.
Observaciones	Para la definición de una entrada de horario se consideran campos como: identificador único de la clase, día de la semana, hora de inicio, hora de fin, docente, materia, aula y periodo académico. El sistema debe manejar restricciones como la capacidad de aulas y conflictos de asignación de docentes para evitar solapamientos. También se evaluará la posibilidad de generar horarios automáticos o sugerencias basadas en la disponibilidad de recursos.

Identificador	**RF13**
Nombre	Gestión de Usuarios (CRUD)
Características	Permite al administrador principal crear, visualizar, modificar y eliminar cuentas de usuario, asignando roles específicos que determinan los permisos y funcionalidades dentro del sistema.
Descripción	El sistema debe facilitar la administración completa de las cuentas de usuario que accederán a la aplicación, incluyendo la gestión de roles y permisos asociados. Esto asegurará que cada usuario tenga acceso acorde a sus responsabilidades, manteniendo la seguridad y control sobre las operaciones dentro del sistema.
Requerimientos no funcionales	RNF01: Las contraseñas deben almacenarse cifradas para garantizar la seguridad.RNF02: La interfaz debe ser segura y cumplir con estándares de autenticación.RNF03: El sistema debe registrar la actividad de los usuarios para auditorías.RNF04: Los tiempos de respuesta para gestión de usuarios deben ser menores a 2 segundos.RNF05: Se debe permitir la recuperación segura de contraseñas olvidadas.
Prioridad del requerimiento	Alta
Criterios de aceptación	- Creación de cuentas con campos: nombre de usuario, contraseña, correo electrónico y nombre completo.- Gestión de roles a través de campos

	desplegables con roles predefinidos y posibilidad de ampliar roles futuros.- Implementación de mecanismos de seguridad para contraseñas, incluyendo requisitos de complejidad y almacenamiento cifrado.- Proceso de autenticación para acceso al sistema.- Registro de logs de actividad y modificaciones.- Mecanismos para desactivar o bloquear cuentas.- Proceso seguro para recuperación de contraseñas.
Observaciones	La gestión de usuarios debe considerar roles predefinidos, con posibilidad de ampliar permisos de manera granular si se requiere. Además, se debe garantizar la seguridad en la administración de credenciales y el control de acceso mediante procesos de autenticación y registro de actividad. Se debe contemplar la funcionalidad para bloqueo y recuperación de cuentas, asegurando la integridad y confidencialidad de la información del sistema y sus usuarios.

Identificador	**RF14**
Nombre	Generación de Hojas de Control
Características	Permite a los administradores y usuarios generar hojas de control para aulas y laboratorios equipados con ordenadores o máquinas electrónicas, facilitando el registro de asistencia de estudiantes y observaciones sobre el estado o uso de los equipos.
Descripción	El sistema debe ofrecer la funcionalidad para crear hojas de control específicas que registren la asistencia de los estudiantes y las observaciones relacionadas con las máquinas dentro de aulas o laboratorios. Esta funcionalidad ayuda a mantener un control actualizado y organizado sobre el uso y estado de los recursos electrónicos disponibles en la facultad.
Requerimientos no funcionales	RNF01: La generación de hojas debe ser rápida y eficiente, con tiempos de respuesta inferiores a 3 segundos.RNF02: Las hojas generadas deben poder exportarse en formatos comunes (PDF, Excel).RNF03: La interfaz debe ser intuitiva para facilitar la generación por parte de usuarios con distintos niveles de experiencia.
Prioridad del requerimiento	Media
Criterios de aceptación	- Identificación clara de los tipos de aulas y laboratorios disponibles para generación de hojas.- Proceso definido para la creación y personalización de hojas de control.- Posibilidad de registrar asistencia y observaciones de máquinas en las hojas.- Exportación y almacenamiento adecuado de las hojas generadas.
Observaciones	Se debe especificar claramente qué tipos de aulas y laboratorios requieren estas hojas de control. El

	sistema debe permitir flexibilidad en el proceso de generación para adaptarse a diferentes escenarios y facilitar el seguimiento y mantenimiento de los recursos tecnológicos.

Requerimientos No Funcionales

Identificador	**RNF01**
Nombre	Seguridad de la información
Características	Garantiza la protección y privacidad de los datos personales y sensibles almacenados en la base de datos mediante técnicas de encriptación robustas y cumplimiento de estándares y regulaciones aplicables.
Descripción	Todos los datos personales y sensibles almacenados en la base de datos deben estar cifrados para asegurar la confidencialidad y evitar accesos no autorizados. Se debe implementar un esquema de encriptación durante el almacenamiento y/o transmisión, además de mecanismos adecuados para la gestión segura de las claves criptográficas.
Requerimientos no funcionales	RNF01: Los datos deben estar encriptados usando algoritmos reconocidos y seguros.RNF02: La encriptación debe aplicarse tanto en reposo (almacenamiento) como en tránsito (transmisión).RNF03: Se debe implementar un sistema seguro para la gestión, almacenamiento y rotación periódica de las claves de encriptación.
Prioridad del requerimiento	Alta
Criterios de aceptación	- Selección de algoritmos criptográficos aprobados y adecuados.- Implementación efectiva de encriptación en la base de datos y en la comunicación.- Existencia de un sistema seguro para manejo de claves, incluyendo generación, almacenamiento, rotación y revocación.- Cumplimiento de regulaciones y estándares de seguridad aplicables.
Observaciones	Es fundamental considerar normativas legales y estándares internacionales en materia de protección de datos (por ejemplo, GDPR, ISO 27001) para definir los métodos y procesos de encriptación. Además, la gestión segura de claves es crítica para evitar vulnerabilidades en el sistema.

Identificador	**RNF02**
Nombre	Diseño del Sistema
Características	La interfaz del sistema debe respetar la identidad visual institucional, incluyendo el uso de colores, logos y otros elementos gráficos oficiales de la universidad y la facultad para garantizar coherencia y reconocimiento.

Descripción	La interfaz del sistema deberá incorporar los colores institucionales específicos y los logos oficiales de la universidad y la facultad, asegurando que se mantengan las pautas de identidad visual establecidas para reflejar correctamente la imagen institucional en todos los elementos de la interfaz.
Requerimientos no funcionales	RNF01: Uso de los colores institucionales con códigos exactos (hexadecimal, RGB) en los componentes UI.RNF02: Inclusión de los logos oficiales en las ubicaciones definidas.RNF03: Cumplimiento de pautas de diseño como tipografía, espaciado e iconografía según manuales institucionales.
Prioridad del requerimiento	Media
Criterios de aceptación	- Incorporación precisa de colores institucionales en la interfaz.- Inclusión correcta y visible de logos institucionales en ubicaciones definidas (encabezado, pie de página, etc.).- Uso conforme a guías de estilo oficiales.- Consulta y aplicación de manuales de identidad visual y UI proporcionados.
Observaciones	Es necesario obtener los códigos de color exactos, los archivos oficiales de los logos en alta resolución y los manuales de identidad visual y UI de la universidad y facultad para garantizar el cumplimiento riguroso de este requerimiento.

Identificador	**RNF03**
Nombre	Generación de Hojas de Control
Características	El sistema debe permitir la generación de hojas de control para aulas y laboratorios con ordenadores o máquinas electrónicas, facilitando el registro de asistencia y observaciones sobre los equipos utilizados.
Descripción	El sistema debe ofrecer funcionalidades para que los usuarios autorizados (administradores o usuarios normales) puedan generar hojas de control que registren la asistencia de estudiantes y las observaciones técnicas sobre las máquinas dentro de aulas o laboratorios específicos.
Requerimientos no funcionales	RNF01: Los formatos de las hojas de control deben ser claros, uniformes y legibles.RNF02: Soporte para uno o múltiples formatos según el tipo de aula o laboratorio.RNF03: Diseño adaptable a las necesidades específicas de cada espacio.
Prioridad del requerimiento	Media
Criterios de aceptación	- Definición y uso de un diseño específico para las hojas de control.- Capacidad para generar diferentes formatos de hojas según el aula o laboratorio.- Aprobación del diseño por parte de los usuarios finales o responsables.
Observaciones	Se recomienda recopilar los formatos actuales utilizados para las hojas de control y las necesidades específicas de cada aula o

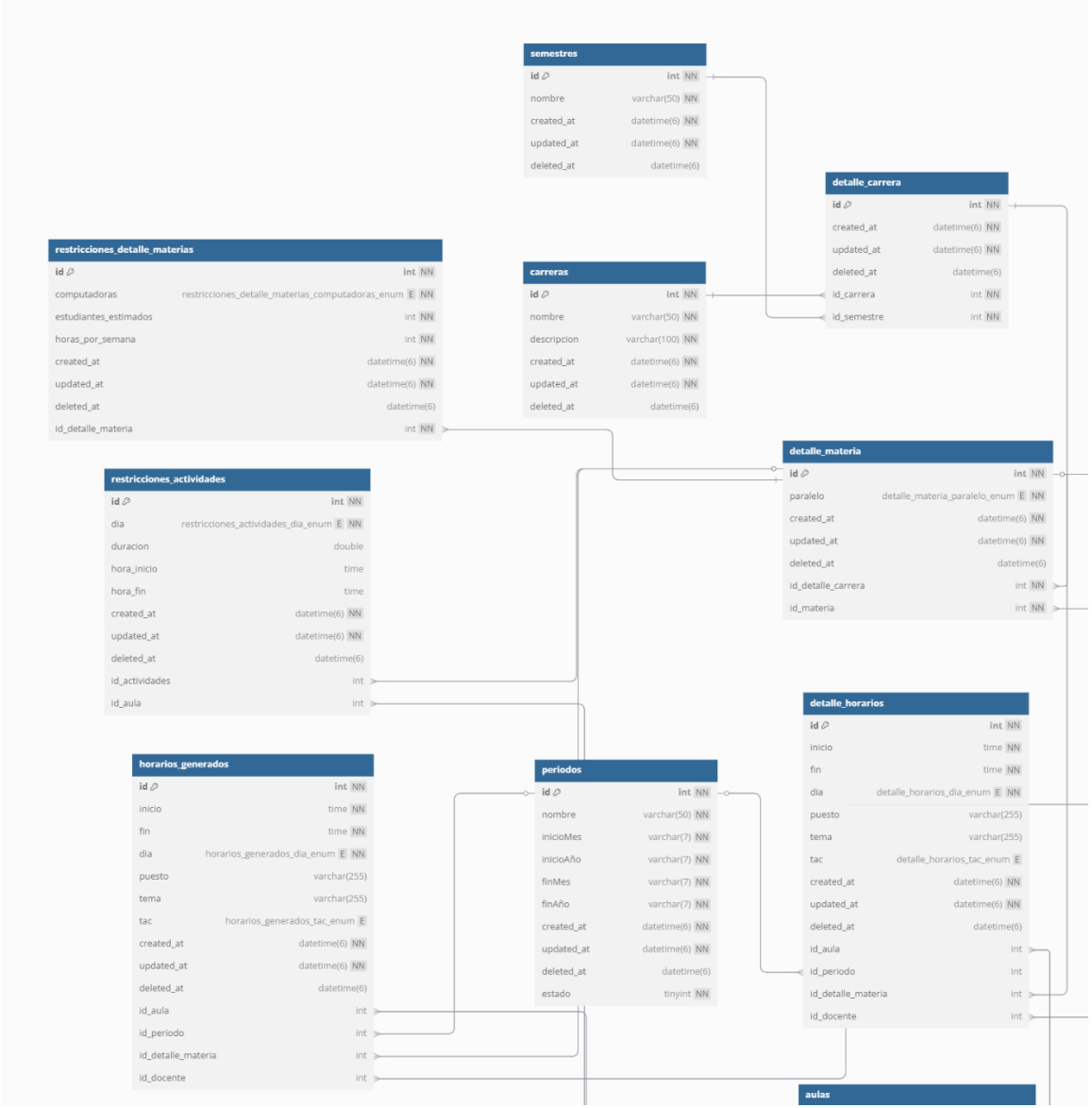
	laboratorio para definir diseños adecuados y garantizar su funcionalidad y usabilidad.

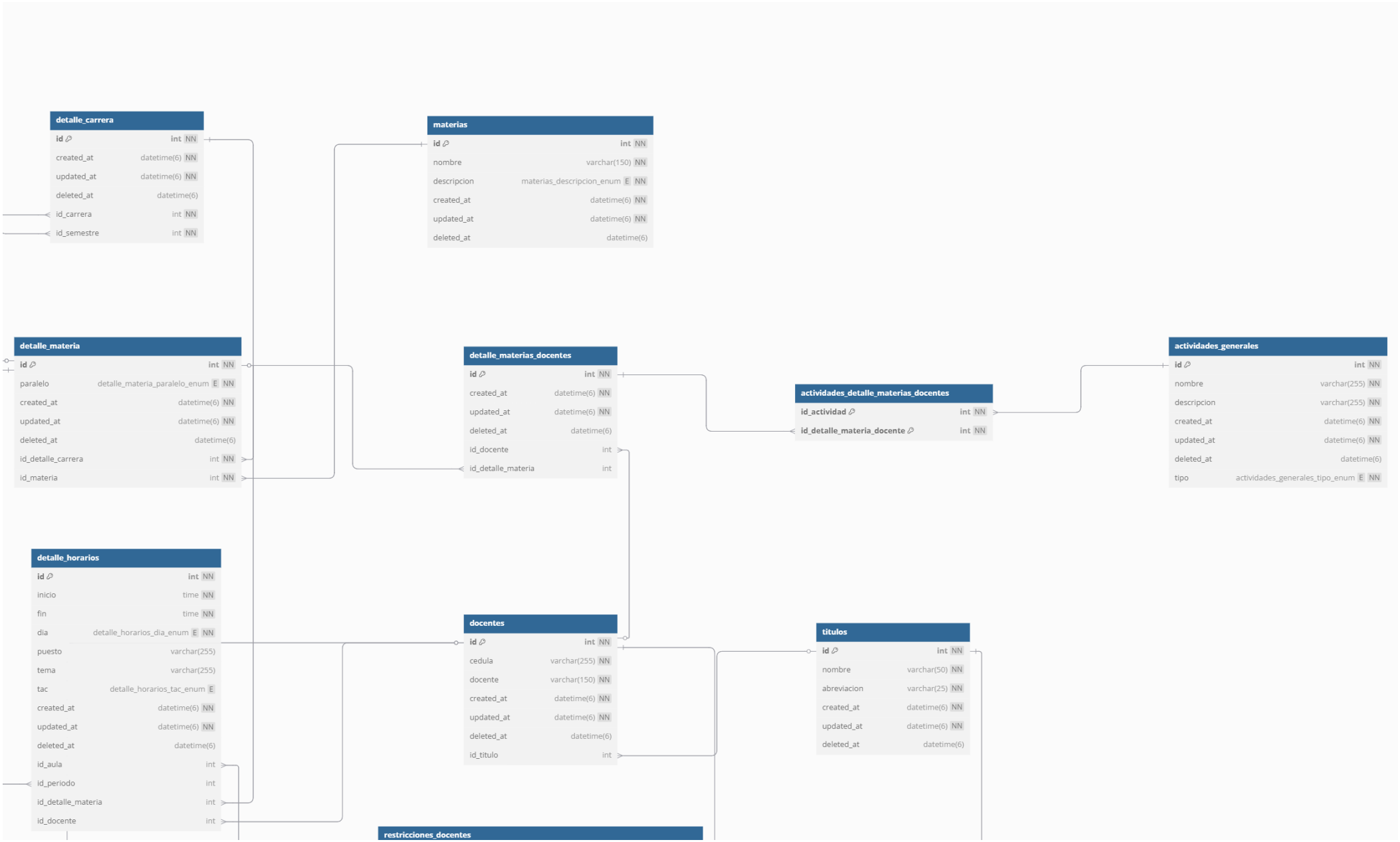
Requerimientos de Generación Horarios

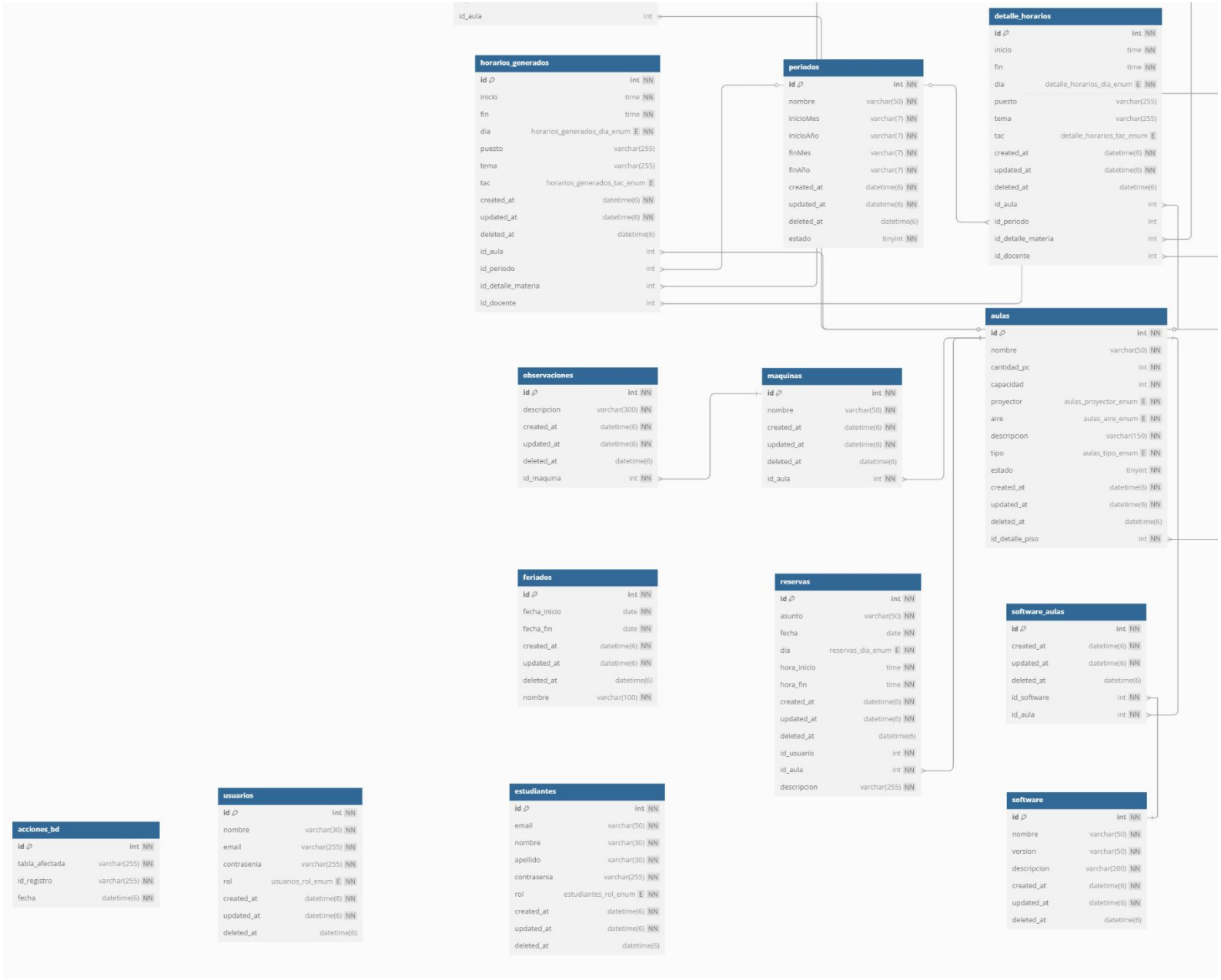
Identificador	**RF01**
Nombre	Restricciones
Características	El sistema debe garantizar que la generación de horarios respete todas las restricciones definidas para asegurar la correcta planificación y asignación de clases, docentes, aulas y otros recursos académicos.
Descripción	Para la correcta generación de horarios, el sistema debe implementar reglas y condiciones que limiten o controlen la asignación de actividades académicas, evitando conflictos como solapamiento de horarios, sobrecarga de docentes o uso simultáneo de aulas.
Requerimientos no funcionales	-
Prioridad del requerimiento	Alta
Criterios de aceptación	- Identificación clara de los sujetos que deben respetar las restricciones (docentes, materias, aulas, etc.).- Definición y aplicación de todas las restricciones necesarias para evitar conflictos en los horarios generados.
Observaciones	Es importante recopilar y validar con los usuarios todas las restricciones específicas que deben ser consideradas, para garantizar que la generación automática o manual de horarios sea confiable y funcional.

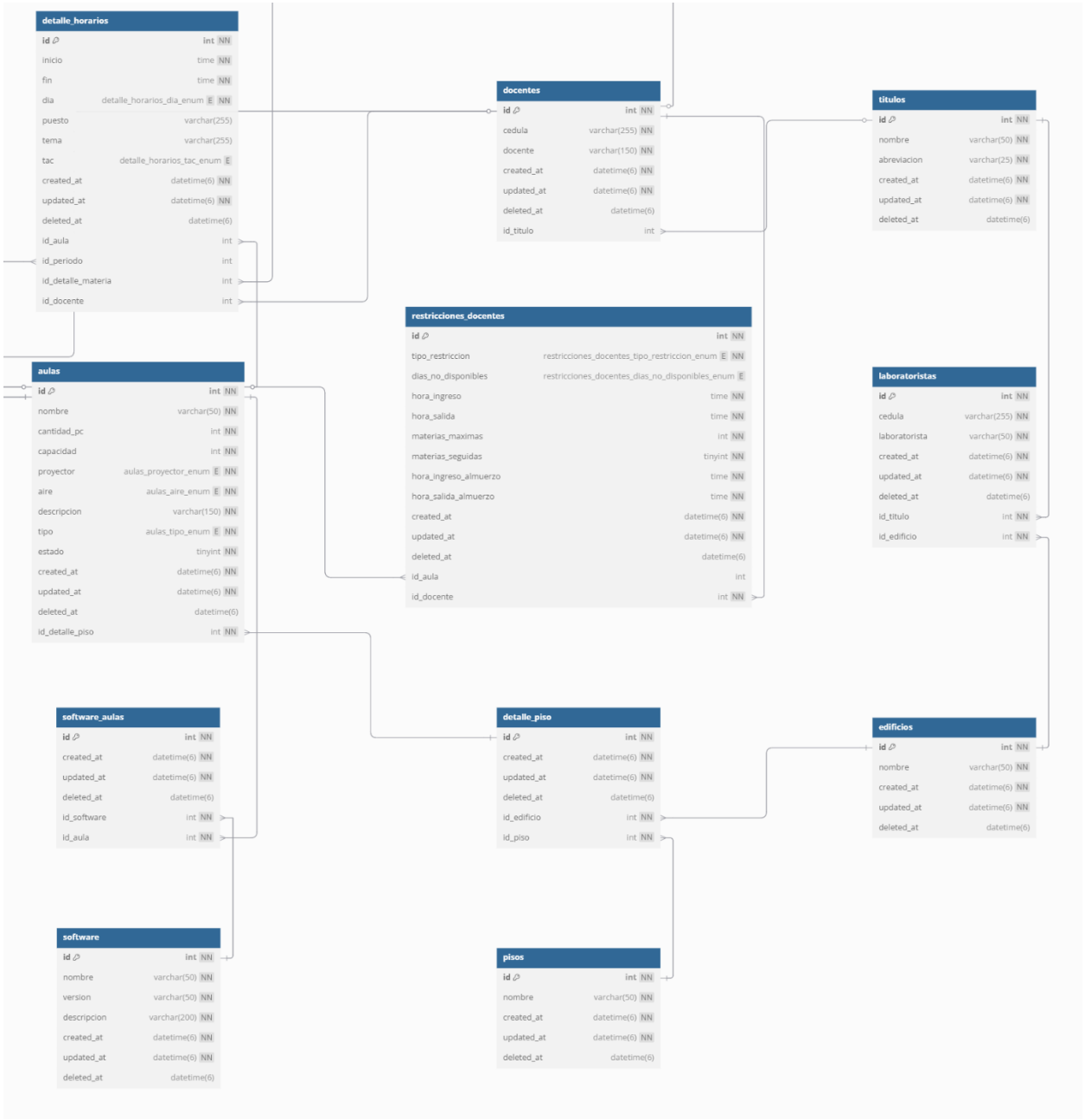
Identificador	**RF02**
Nombre	Generación de Horarios
Características	El sistema debe generar horarios respetando todas las restricciones definidas, utilizando algoritmos de optimización adecuados para producir una planificación eficiente y sin conflictos.
Descripción	Para realizar la generación de horarios, el sistema implementará métodos y algoritmos que consideren las restricciones necesarias, garantizando que los horarios generados sean viables y cumplan con los criterios establecidos.
Requerimientos no funcionales	-
Prioridad del requerimiento	Alta
Criterios de aceptación	- El sistema debe poder utilizar uno o más algoritmos de optimización para la generación automática de horarios.- Los horarios generados deben ser visualizables de forma clara y accesible.- Se debe permitir la modificación o actualización manual de los horarios generados.
Observaciones	Se recomienda investigar y seleccionar algoritmos de optimización que mejor se adapten a las características y restricciones del entorno académico. La visualización debe facilitar la

	consulta y posibles ajustes posteriores por parte del usuario.


semestres
id int NN
nombre varchar(50) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
detalle_carrera
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_carrera int NN
id_semestre int NN
restricciones_detalle_materias
id int NN
computadoras restricciones_detalle_materias_computadoras_enum E NN
estudiantes_estimados int NN
horas_por_semana int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_detalle_materia int NN
carreras
id int NN
nombre varchar(50) NN
descripcion varchar(100) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
detalle_materia
id int NN
paralelo detalle_materia_paralelo_enum E NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_detalle_carrera int NN
id_materia int NN
restricciones_actividades
id int NN
dia restricciones_actividades_dia_enum E NN
duracion double
hora_inicio time
hora_fin time
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_actividades int
id_aula int
detalle_horarios
id int NN
inicio time NN
fin time NN
dia detalle_horarios_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac detalle_horarios_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
horarios_generados
id int NN
inicio time NN
fin time NN
dia horarios_generados_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac horarios_generados_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
periodos
id int NN
nombre varchar(50) NN
inicioMes varchar(7) NN
inicioAño varchar(7) NN
finMes varchar(7) NN
finAño varchar(7) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
estado tinyint NN
aulas


detalle_carrera
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_carrera int NN
id_semestre int NN
materias
id int NN
nombre varchar(150) NN
descripcion materias_descripcion_enum E NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
detalle_materia
id int NN
paralelo detalle_materia_paralelo_enum E NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_detalle_carrera int NN
id_materia int NN
detalle_materias_docentes
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_docente int
id_detalle_materia int
actividades_detalle_materias_docentes
id_actividad int NN
id_detalle_materia_docente int NN
actividades_generales
id int NN
nombre varchar(255) NN
descripcion varchar(255) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
tipo actividades_generales_tipo_enum E NN
detalle_horarios
id int NN
inicio time NN
fin time NN
dia detalle_horarios_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac detalle_horarios_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
docentes
id int NN
cedula varchar(255) NN
docente varchar(150) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_titulo int
titulos
id int NN
nombre varchar(50) NN
abreviacion varchar(25) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
restricciones_docentes

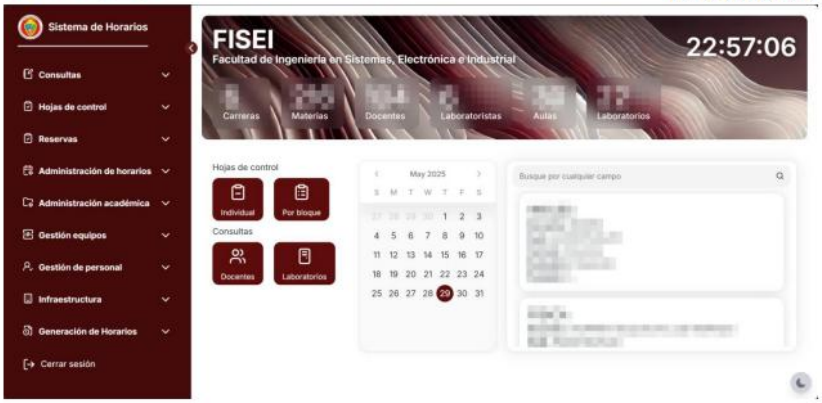

id_aula int
horarios_generados
id int NN
inicio time NN
fin time NN
dia horarios_generados_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac horarios_generados_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
periodos
id int NN
nombre varchar(50) NN
inicioMes varchar(7) NN
inicioAño varchar(7) NN
finMes varchar(7) NN
finAño varchar(7) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
estado tinyint NN
detalle_horarios
id int NN
inicio time NN
fin time NN
dia detalle_horarios_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac detalle_horarios_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
aulas
id int NN
nombre varchar(50) NN
cantidad_pc int NN
capacidad int NN
proyector aulas_proyector_enum E NN
aire aulas_aire_enum E NN
descripcion varchar(150) NN
tipo aulas_tipo_enum E NN
estado tinyint NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_detalle_piso int NN
observaciones
id int NN
descripcion varchar(300) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_maquina int NN
maquinas
id int NN
nombre varchar(50) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int NN
feriados
id int NN
fecha_inicio date NN
fecha_fin date NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
nombre varchar(100) NN
reservas
id int NN
asunto varchar(50) NN
fecha date NN
dia reservas_dia_enum E NN
hora_inicio time NN
hora_fin time NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_usuario int NN
id_aula int NN
descripcion varchar(255) NN
software_aulas
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_software int NN
id_aula int NN
software
id int NN
nombre varchar(50) NN
version varchar(50) NN
descripcion varchar(200) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
acciones_bd
id int NN
tabla_afectada varchar(255) NN
id_registro varchar(255) NN
fecha datetime(6) NN
usuarios
id int NN
nombre varchar(30) NN
email varchar(255) NN
contrasenia varchar(255) NN
rol usuarios_rol_enum E NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
estudiantes
id int NN
email varchar(50) NN
nombre varchar(30) NN
apellido varchar(30) NN
contrasenia varchar(255) NN
rol estudiantes_rol_enum E NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)


detalle_horarios
id int NN
inicio time NN
fin time NN
dia detalle_horarios_dia_enum E NN
puesto varchar(255)
tema varchar(255)
tac detalle_horarios_tac_enum E
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_periodo int
id_detalle_materia int
id_docente int
docentes
id int NN
cedula varchar(255) NN
docente varchar(150) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_titulo int
titulos
id int NN
nombre varchar(50) NN
abreviacion varchar(25) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
restricciones_docentes
id int NN
tipo_restriccion restricciones_docentes_tipo_restriccion_enum E NN
dias_no_disponibles restricciones_docentes_dias_no_disponibles_enum E
hora_ingreso time NN
hora_salida time NN
materias_maximas int NN
materias_seguidas tinyint NN
hora_ingreso_almuerzo time NN
hora_salida_almuerzo time NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_aula int
id_docente int NN
aulas
id int NN
nombre varchar(50) NN
cantidad_pc int NN
capacidad int NN
proyector aulas_proyector_enum E NN
aire aulas_aire_enum E NN
descripcion varchar(150) NN
tipo aulas_tipo_enum E NN
estado tinyint NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_detalle_piso int NN
laboratoristas
id int NN
cedula varchar(255) NN
laboratorista varchar(50) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_titulo int NN
id_edificio int NN
software_aulas
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_software int NN
id_aula int NN
detalle_piso
id int NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
id_edificio int NN
id_piso int NN
edificios
id int NN
nombre varchar(50) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
software
id int NN
nombre varchar(50) NN
version varchar(50) NN
descripcion varchar(200) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)
pisos
id int NN
nombre varchar(50) NN
created_at datetime(6) NN
updated_at datetime(6) NN
deleted_at datetime(6)

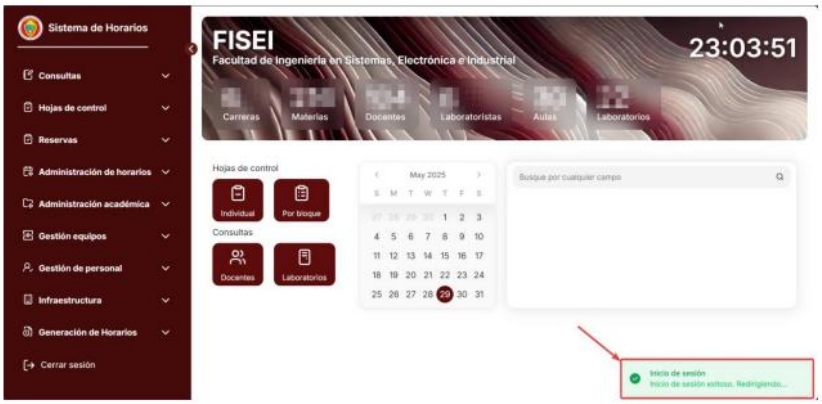
Anexo H. Manual de Usuario

Autor	Christian Chico
Fecha	15 de junio de 2025
Titulo	Manual de Usuario de Panel Administrativo
URL	Manual de Usuario Administrador.pdf



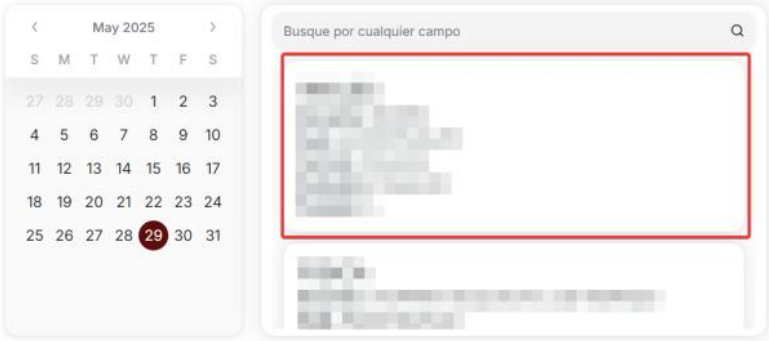


Adicional a ellos, se le mostrara una notificación en la parte inferior derecha de la pantalla



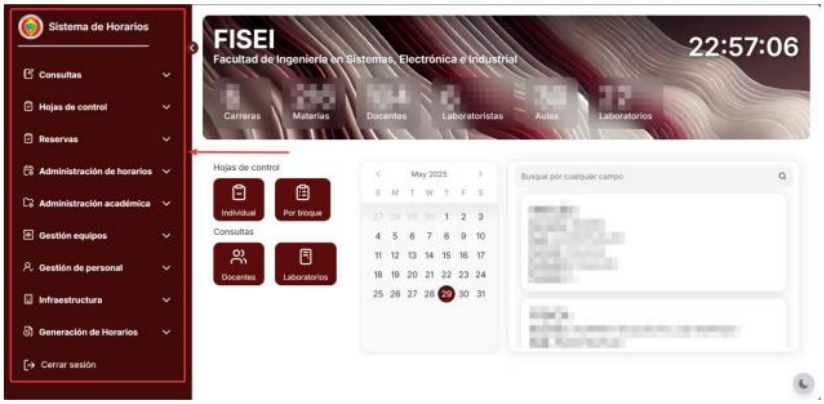
3. Primeros Pasos

Para empezar, te guiaremos por las **funciones clave del panel administrativo principal** de la aplicación. A continuación, te mostraremos la ubicación de cada elemento importante y explicaremos su propósito.



3.4.Panel de Navegación

El panel de navegación se encuentra en toda la parte izquierda de la pantalla, este es el panel de navegación por el cual podrá acceder a las distintas funcionalidades de la aplicación:



Nota: Utilizando la rueda del ratón podrá explorar las distintas categorías y funciones existentes