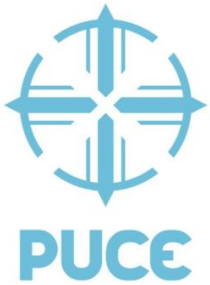


PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

ESCUELA HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TECNOLOGÍAS DE LA INFORMACIÓN

SISTEMA WEB INTELIGENTE PARA LA OPTIMIZACIÓN DE LA PROGRAMACIÓN ACADÉMICA Y LA ASIGNACIÓN DINÁMICA DE RECURSOS, DOCENTES Y HORARIOS

HERMOSA PONCE ADRIAN RENATO

TUTOR: QUISHPE MORALES SANTIAGO DAMIÁN

IBARRA – ECUADOR

JULIO, 2026

Ibarra, 10 de julio del 2026

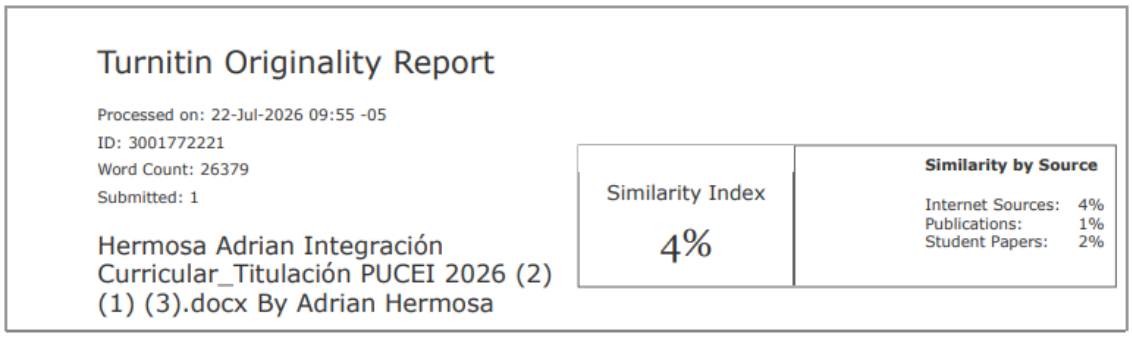
CERTIFICACIÓN TUTOR

En mi calidad de Tutor del Trabajo de Integración Curricular titulado:

SISTEMA WEB INTELIGENTE PARA LA OPTIMIZACIÓN DE LA PROGRAMACIÓN ACADÉMICA Y LA ASIGNACIÓN DINÁMICA DE RECURSOS, DOCENTES Y HORARIOS, presentado por el estudiante Hermosa Ponce Adrian Renato con cédula de ciudadanía N° 1727461871, para obtener el Título de Ingeniero en Tecnologías de la Información.

Certifico que el trabajo cumple con todos los parámetros establecidos, mediante el cual el estudiante demuestra el desarrollo de competencias en el campo de conocimiento de su profesión con un nivel de argumentación coherente, para ser sometido a la evaluación por parte de los lectores.

Adicionalmente, se adjunta el certificado de porcentaje de originalidad de TURNITIN.



Turnitin Originality Report

Processed on: 22-Jul-2026 09:55 -05
ID: 3001772221
Word Count: 26379
Submitted: 1

Hermosa Adrian Integración Curricular_Titulación PUCEI 2026 (2) (1) (3).docx By Adrian Hermosa

Similarity Index
4%

Similarity by Source
Internet Sources: 4%
Publications: 1%
Student Papers: 2%

(f):_____ Santiago Quishpe Morales

Firmado digitalmente por Santiago Quishpe Morales
Fecha: 2026.07.22 11:24:52 -05'00'

Mgs. Quishpe Morales Santiago Damián
TUTOR DE TRABAJO
C.C.: 1002697223

PÁGINA DE APROBACIÓN DEL TRIBUNAL

El tribunal examinador, aprueba el presente trabajo en nombre de la Pontificia Universidad Católica del Ecuador Ibarra:

Santiago Quishpe Morales
Firmado digitalmente por Santiago Quishpe Morales
Fecha: 2026.07.22 11:24:36 -05'00'

(f): .....

Mgs. Quishpe Morales Santiago Damián

C.C.: 1002697223

Dulce Milagro Rivero Albarrán
Firmado digitalmente por Dulce Milagro Rivero Albarrán
Fecha: 2026.07.22 11:28:15 -05'00'

(f):.....

PhD. Rivero Albarrán Dulce Milagro

C.C.: 1757608961

Laura Guerra Torrealba
Firmado digitalmente por Laura Guerra Torrealba
Fecha: 2026.07.22 15:14:43 -05'00'

(f):.....

PhD. Guerra Torrealba Laura Rosa

C.C.: 1757842784

ACTA DE CESIÓN DE DERECHOS

Yo, *Hermosa Ponce Adrian Renato*, declaro conocer y aceptar la disposición del Art. 165 del Código Orgánico de Economía Social de los Conocimientos, Creatividad e Innovación, que manifiesta textualmente: "Se reconoce facultad de los autores y demás titulares de derechos de disponer de sus derechos o autorizar las utilizaciones de sus obras o prestaciones a título gratuito y oneroso, según las condiciones que determinen. Esta facultad podrá ejercerse mediante licencias libres, abiertas y otros modelos alternativos de licenciamiento o la renuncia".

Ibarra, 10 de julio del 2026

ADRIAN RENATO HERMOSA PONCE

Firmado digitalmente por ADRIAN RENATO HERMOSA PONCE
Fecha: 2026.07.22 11:23:07 -05'00'

(f):_____

Hermosa Ponce Adrian Renato
C.C.: 1727461871

AUTORIA

Yo, *Hermosa Ponce Adrian Renato*, portador de la cedula de ciudadanía N° 172746187-1, declaro que el presente trabajo de investigación es de total responsabilidad del autor, y eximo expresamente a la Pontificia Universidad Católica del Ecuador Ibarra de posibles reclamos o acciones legales.

ADRIAN RENATO HERMOSA PONCE

Firmado digitalmente por ADRIAN RENATO HERMOSA PONCE
Fecha: 2026.07.22 11:23:28 -05'00'

(f):……..

Hermosa Ponce Adrian Renato

C.C.: 172746187-1

DEDICATORIA

A Dios, por ser mi guía, mi refugio y mi fortaleza en cada etapa de este camino. Por darme la sabiduría para comprender, la paciencia para persistir y la fuerza para levantarme en los momentos en que las dificultades parecían más grandes que mis capacidades. *"Porque yo sé los pensamientos que tengo acerca de vosotros... pensamientos de paz y no de mal, para daros el fin que esperáis." — Jeremías 29:11*

A mí mismo, por la constancia, la disciplina y la determinación que me permitieron llegar hasta aquí. Por las noches de desvelo y por no rendirme incluso cuando el cansancio invitaba a hacerlo.

A mi padre, Edi Hermosa, por ser mi ejemplo de trabajo, honradez y perseverancia. Por enseñarme, con su esfuerzo diario y muchas veces silencioso, que las metas se alcanzan con dedicación y sacrificio.

A mi madre, Eva Ponce, por su amor incondicional, su ternura y su apoyo inquebrantable. Por ser el sostén de mi vida y por acompañarme con sus oraciones y su cariño en cada paso de este trayecto.

A mis hermanos, Carlos y Génesis, por su cariño, su compañía y su apoyo constante. Por ser cómplices de tantos momentos y por recordarme, cada vez que lo necesité, que no caminaba solo.

"Simón mijín, estudia pa' que traigas el primer título de ing a la casita." — Carlos

A todos ustedes, papá, mamá y hermanos, que son el motor de mi vida y la razón de mi esfuerzo, les dedico con profundo amor este logro que también es suyo.

Adrian Hermosa

AGRADECIMIENTOS

En primer lugar, agradezco a Dios por haberme acompañado a lo largo de toda esta etapa, por darme la fortaleza, la sabiduría y la perseverancia necesarias para culminar este trabajo y por poner en mi camino a las personas correctas en los momentos precisos.

A mis padres, Edi Hermosa y Eva Ponce, mi más profundo agradecimiento por su amor incondicional, su sacrificio constante y su apoyo inquebrantable. Nada de esto habría sido posible sin su esfuerzo, sus consejos y la confianza que siempre depositaron en mí. Este logro es el fruto de todo lo que ellos sembraron.

A mis hermanos, Carlos y Génesis, por su cariño, su compañía y su aliento en cada momento. Gracias por creer en mí y por ser una motivación permanente para seguir adelante.

A mi abuelita, Angelita Hermosa, por su amor, sus oraciones y su ejemplo de vida. Su cariño y su sabiduría han sido una bendición y una guía invaluable a lo largo de mi camino.

A mi asesor, por su guía, su paciencia y sus valiosas observaciones durante el desarrollo de este trabajo. Su acompañamiento y su experiencia fueron fundamentales para llevar esta investigación a buen término.

A mis lectores, por el tiempo dedicado a revisar este trabajo y por sus aportes y sugerencias, que contribuyeron significativamente a mejorar la calidad del resultado final.

A la Pontificia Universidad Católica del Ecuador y a todos sus docentes, por la formación académica y humana que recibí a lo largo de estos años.

A mis amigos/as, por su compañía, su apoyo y por cada momento compartido durante esta etapa. Gracias por hacer más llevadero el camino y por estar presentes en los buenos y en los malos momentos.

A todos ustedes, mi más sincera gratitud.

Adrian Hermosa

ÍNDICE DE CONTENIDOS

Tabla de contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La programación académica de la Escuela de Hábitat, Infraestructura y Creatividad de la PUCE Sede Ibarra se realizaba mediante hojas de cálculo Excel desconectadas, lo que generaba errores recurrentes y dificultaba verificar el cumplimiento de la normativa sobre carga horaria docente. El objetivo general del presente trabajo fue desarrollar un sistema web inteligente para la optimización de la programación académica y la asignación dinámica de recursos mediante métodos heurísticos y algoritmos genéticos.

La investigación adoptó un enfoque aplicado con metodología mixta y el desarrollo se gestionó mediante Scrum, en Sprints de dos semanas. Los requisitos se levantaron mediante observación directa, entrevista al coordinador académico y análisis documental, y se documentaron en quince historias de usuario que guiaron la construcción del sistema. La implementación empleó Vue.js 3, NestJS y PostgreSQL para la capa web, y Python con la librería DEAP para el motor de optimización, orquestados con Docker e integrados vía API REST. El algoritmo genético codifica cada clase mediante tres genes (día, bloque horario y docente) y evalúa cada horario candidato con una función de aptitud (fitness) que parte de 10.000 puntos y resta penalizaciones por violaciones de restricciones duras y blandas. El ciclo evolutivo aplica selección por torneo, cruce de dos puntos y mutación durante doscientas generaciones fijas, al cabo de las cuales se retorna la mejor solución encontrada; al concluir, una heurística voraz asigna los laboratorios priorizando los niveles de mayor demanda estudiantil y minimizando conflictos de ocupación.

La validación combinó seis pruebas de aceptación funcionales, que cubren los flujos principales derivados de las quince historias de usuario, con veintidós escenarios de éxito y error, todos cumplidos sobre datos reales de la institución. Las métricas cuantitativas sobre ocho instancias de distintos niveles, paralelos y jornadas no registraron cruces de docentes, aulas ni estudiantes, con valores de fitness entre el 94.4% y el 99.0% del máximo teórico y un tiempo de ejecución aproximado de un segundo. La estabilidad del algoritmo, evaluada mediante cinco ejecuciones independientes sobre una misma instancia, arrojó una media de 9710.0 puntos de fitness con una desviación estándar de 170.15 puntos (1.7% de la escala), lo que evidencia resultados consistentes pese a la naturaleza estocástica del método. Se concluye que el sistema cumplió el objetivo general

propuesto, y se recomienda conservar las programaciones generadas como línea base para evaluar en periodos futuros la reducción de conflictos respecto al proceso manual.

ABSTRACT

The academic scheduling of the School of Habitat, Infrastructure and Creativity at PUCE Ibarra Campus was carried out using disconnected Excel spreadsheets, which led to recurring errors and made it difficult to verify compliance with regulations on teaching workload. The general objective of this work was to develop an intelligent web system for optimizing academic scheduling and the dynamic allocation of resources through heuristic methods and genetic algorithms.

The research adopted an applied approach with a mixed methodology, and development was managed using Scrum with two-week sprints. Requirements were gathered through direct observation, an interview with the academic coordinator, and documentary analysis, and were documented in fifteen user stories that guided the construction of the system. The implementation used Vue.js 3, NestJS, and PostgreSQL for the web layer, and Python with the DEAP library for the optimization engine, orchestrated with Docker and integrated via a REST API. The genetic algorithm encodes each class through three genes (day, time block, and teacher) and evaluates each candidate schedule with a fitness function that starts from 10,000 points and subtracts penalties for violations of hard and soft constraints. The evolutionary cycle applies tournament selection, two-point crossover, and mutation over two hundred fixed generations, after which the best solution found is returned; upon completion, a greedy heuristic assigns laboratories by prioritizing the levels with the highest student demand and minimizing occupancy conflicts.

Validation combined six functional acceptance tests, covering the main workflows derived from the fifteen user stories, with twenty-two success and error scenarios, all fulfilled using real institutional data. Quantitative metrics over eight instances of different levels, parallels, and shifts recorded no teacher, classroom, or student clashes, with fitness values between 94.4% and 99.0% of the theoretical maximum and an execution time of approximately one second. The algorithm's stability, evaluated through five independent runs on the same instance, yielded a mean of 9,710.0 fitness points with a standard deviation of 170.15 points (1.7% of the scale), evidencing consistent results despite the stochastic nature of the method. It is concluded that the system fulfilled the proposed general objective, and it is recommended to preserve the generated schedules as a baseline to evaluate, in future periods, the reduction of conflicts compared to the manual process.

INTRODUCCIÓN

La programación académica universitaria es un proceso crítico que coordina la asignación de docentes, materias, aulas y horarios cada semestre en instituciones de educación superior. Durante los últimos años, los algoritmos genéticos han demostrado ser herramientas efectivas para automatizar este proceso complejo, ya que permiten explorar eficientemente espacios de solución de gran tamaño mediante operadores evolutivos de selección, cruce y mutación (Soria-Alcaraz et al., 2023) y generar horarios que satisfacen múltiples restricciones simultáneamente. Según Ceschia et al. (2023), el problema de programación de horarios universitarios es conocido como un problema de optimización combinatoria de alta complejidad, cuya resolución práctica en tiempos computacionales razonables requiere el uso de enfoques heurísticos y metaheurísticos como los algoritmos genéticos.

Los sistemas de información educativa han mejorado significativamente la gestión administrativa universitaria año tras año. En la actualidad, las tecnologías web son capaces de brindar soluciones de optimización avanzadas de manera accesible a las instituciones, ya que solo se necesita infraestructura básica de servidores y navegadores webs modernos. La automatización es una de las mejores alternativas para mejorar la eficiencia operativa y reducir errores en procesos administrativos educativos. La creación de sistemas web inteligentes puede motivar a las instituciones a modernizar sus procesos y adoptar tecnologías avanzadas.

Los algoritmos genéticos implementados en Python mediante la librería DEAP (*Distributed Evolutionary Algorithms in Python*) ofrecen capacidades de optimización robustas de la manera más simple y accesible. Estudios científicos documentan que estos algoritmos generan soluciones que satisfacen entre 80-95% de las restricciones en problemas de generación de horarios universitarios (Yusop et al., 2022).

La Escuela de Hábitat, Infraestructura y Creatividad de la Pontificia Universidad Católica del Ecuador – Sede Ibarra (PUCE-SI) presentaba en su proceso de programación académica limitaciones derivadas del método manual utilizado, caracterizado por múltiples carreras activas y recursos limitados que debían coordinarse respetando restricciones y normativas específicas. La programación académica se realizaba mediante hojas de cálculo Excel desconectadas entre sí, lo cual generaba errores humanos recurrentes e inconsistencias, lo que dificultaba la verificación del cumplimiento de la normativa institucional.

Este proyecto de investigación propone un sistema web inteligente mediante el cual los coordinadores académicos pueden gestionar recursos y generar horarios optimizados desde cualquier estación de trabajo. Además, proporciona una solución integral que automatiza dos procesos fundamentales. Primero, la administración centralizada de docentes, materias y aulas mediante una interfaz web moderna que organiza la información bajo la jerarquía académica institucional (escuela, carrera, plan de estudios y materia), permite su registro, consulta, actualización y desactivación con validaciones de integridad, y elimina la dependencia de hojas de cálculo desconectadas. Segundo, la generación automática de horarios académicos mediante algoritmos genéticos, que exploran el espacio de soluciones para encontrar configuraciones que cumplan simultáneamente las restricciones normativas de asignación de horas docentes según categoría institucional: Docentes Titulares a Tiempo Completo (16-20 horas semanales), Medio Tiempo (9-12 horas) y Tiempo Parcial (hasta 11 horas), conforme al Oficio Nro. 902-Dir.DocyEst, la distribución equilibrada entre jornadas matutina y vespertina, y la coherencia entre las áreas de conocimiento del docente y los contenidos de las materias asignadas.

El proyecto busca solventar la problemática de la generación de horarios por medio el siguiente objetivo principal:

Objetivo General

Desarrollar un sistema web inteligente para la optimización de la programación académica y la asignación dinámica de recursos académicos en la Escuela de Hábitat, Infraestructura y Creatividad de la PUCE Sede Ibarra, utilizando métodos heurísticos y algoritmos genéticos.

A partir de este objetivo general se desglosan los siguientes objetivos específicos que se irán cumpliendo a medida que se realice la investigación:

Objetivos Específicos

- Analizar el proceso actual de programación académica, identificando restricciones duras y blandas en la asignación de recursos académicos.
- Diseñar la arquitectura del sistema web inteligente, integrando un módulo de gestión académica mediante *frameworks* modernos de desarrollo web y un motor de optimización en Python mediante servicios API REST.

- Desarrollar un algoritmo genético con métodos heurísticos para la generación automática de horarios en entorno de desarrollo, evaluando su capacidad de cumplimiento de restricciones normativas institucionales y coherencia con perfiles académicos de los docentes.
- Validar el sistema mediante pruebas funcionales y métricas cuantitativas de calidad.

El presente trabajo consta de tres (3) capítulos: el primero describe las investigaciones bibliográficas y conceptos que sirven como antecedentes para la investigación. El segundo capítulo consta de las herramientas tecnológicas y el desarrollo del proyecto. El tercer capítulo expone los resultados de la aplicación final y la evaluación del sistema en la Escuela de Hábitat, Infraestructura y Creatividad. Finalmente, las conclusiones y recomendaciones derivadas de este trabajo son presentadas.

CAPÍTULO I

ESTADO DEL ARTE

El presente capítulo aborda la fundamentación teórica que sustenta el desarrollo del sistema web inteligente para la optimización de la programación académica. Se presenta una revisión bibliográfica de investigaciones previas en sistemas de horarios universitario, tanto en el contexto internacional como en el ecuatoriano, identificando los enfoques metodológicos y soluciones tecnológicas que han demostrado efectividad en la resolución de este problema de optimización combinatoria. Posteriormente, se describen los conceptos fundamentales relacionados con algoritmos genéticos aplicados a problemas educativos, arquitecturas de sistemas web desacoplados, optimización multiobjetivo, y las restricciones normativas específicas de PUCE Sede Ibarra que constituyen el marco de referencia para el diseño e implementación del sistema propuesto.

1.1 ANTECEDENTES Y CONTEXTO

1.1.1 El Problema de horario universitario

El problema de programación de horarios universitarios, conocido internacionalmente como University Course Timetabling Problem (UCTP), constituye una tarea combinatoria compleja que implica la asignación sistemática de eventos educativos, cursos, docentes y estudiantes a un número limitado de franjas horarias y aulas disponibles, considerando simultáneamente múltiples restricciones institucionales (Babaei et al., 2015). Dado el elevado número de restricciones y el vasto espacio de soluciones, el UCTP se clasifica formalmente como un problema NP-hard, lo que significa que no existe un algoritmo eficiente conocido que garantice encontrar la solución óptima en tiempo polinomial (Soria-Alcaraz et al., 2023).

1.1.2 Investigaciones previas en contexto internacional

La investigación en horarios universitarios ha experimentado avances significativos mediante la aplicación de técnicas metaheurísticas y algoritmos evolutivos. Nguyen et al. (2025) propusieron un algoritmo genético de dos etapas, clases magistrales y horarios de laboratorio. Utiliza lógica difusa para el problema de programación de horarios universitarios, incorporando un modelo de lenguaje grande (LLM) para interpretar preferencias de profesores expresadas en lenguaje natural y transformarlas en puntuaciones de satisfacción difusa sobre franjas horarias. En la primera etapa, el algoritmo construye horarios de clases magistrales aplicando todas las restricciones duras durante la fase de codificación, mientras que la función de fitness evalúa el grado de

violación de restricciones blandas, particularmente las preferencias temporales vagas de los docentes. En la segunda etapa, se programan las sesiones de laboratorio basándose en el horario de clases previamente generado, asegurando coherencia temporal y evitando conflictos con las sesiones magistrales. Este enfoque híbrido demostró capacidad para generar horarios de alta calidad que equilibran restricciones técnicas con preferencias humanas expresadas coloquialmente.

Pratama et al. (2024) abordaron específicamente el problema de optimización de horarios de laboratorios prácticos mediante algoritmos genéticos en un caso de estudio real desarrollado en Telkom University, Indonesia. Los autores implementaron un sistema automatizado que considera restricciones específicas de espacios de laboratorio, equipamiento especializado requerido por cada práctica, y disponibilidad sincronizada de docentes asistentes técnicos. El algoritmo genético desarrollado utiliza codificación binaria para representar asignaciones válidas de prácticas a franjas horarias, aplica operadores de cruce de dos puntos, mutación aleatoria uniforme y selección por torneo para evolucionar la población de soluciones candidatas. Los resultados experimentales demostraron reducción significativa de conflictos de programación comparado con el proceso manual previo, alcanzando tasas de satisfacción de restricciones superiores al 92% en instancias reales con más de 80 grupos de laboratorio distribuidos en 40 franjas horarias semanales.

Alshammari et al. (2023) desarrollaron un algoritmo genético que incorpora explícitamente las preferencias estudiantiles en el proceso de asignación de estudiantes a secciones de clase, abordando el problema de programación de horarios considerando que las preferencias individuales históricamente han sido descartadas en procesos manuales tradicionales debido a su complejidad computacional. La propuesta introduce operadores genéticos personalizados de cruce y mutación adaptados al dominio, que preservan la factibilidad de soluciones durante la evolución y aceleran la convergencia hacia configuraciones de alta calidad. Adicionalmente, el algoritmo incorpora funciones de reparación y mejora que se ejecutan conjuntamente con los operadores genéticos estándar para incrementar la calidad de las soluciones generadas. Los experimentos realizados sobre conjuntos de datos universitarios reales demostraron que el enfoque propuesto genera horarios donde las preferencias estudiantiles se satisfacen en un 78% promedio, comparado con menos del 45% en métodos de asignación aleatoria o manual, manteniendo simultáneamente el cumplimiento total de restricciones duras.

Fuenmayor et al. (2022) presentaron un caso de estudio detallado sobre la aplicación de algoritmos genéticos para la programación automatizada de horarios de laboratorios académicos, documentando el proceso completo desde el levantamiento de requisitos institucionales hasta la validación empírica de resultados en un entorno educativo real. El trabajo enfatiza la importancia de calibrar adecuadamente los parámetros del algoritmo genético: tamaño de la población, tasa de cruce, tasa de mutación y criterio de terminación, mediante experimentación sistemática sobre instancias representativas del problema institucional específico. Los autores reportan que configuraciones con poblaciones de 100-150 individuos, tasas de cruce del 70-80%, tasas de mutación del 1-5%, y límites de 500-1000 generaciones producen consistentemente soluciones de calidad comparable a horarios diseñados manualmente por coordinadores expertos, requiriendo tiempos de ejecución computacional inferiores a 5 minutos en hardware convencional contemporáneo. En el presente trabajo, los parámetros se calibraron para las instancias específicas de la Escuela de Hábitat de menor escala que las de la literatura citada, adoptando 200 generaciones y una tasa de mutación a nivel de individuo de 0.3, valores detallados y justificados en el Capítulo II.

Hussein y Albayati (2020) propusieron mejoras al algoritmo genético estándar mediante la implementación de mutaciones inteligentes basadas en el valor de aptitud (fitness) de cada gen individual dentro del cromosoma que codifica una solución candidata. A diferencia del operador de mutación tradicional que aplica cambios aleatorios uniformes sobre todos los genes sin distinción, el enfoque propuesto calcula una métrica de calidad para cada asignación individual dentro del horario completo y concentra las mutaciones preferencialmente en aquellas asignaciones que contribuyen negativamente al valor global de fitness. Esta estrategia adaptativa dirige la búsqueda evolutiva hacia regiones prometedoras del espacio de soluciones más eficientemente que mutaciones puramente estocásticas, acelerando la convergencia del algoritmo hacia soluciones óptimas locales de alta calidad. Los experimentos comparativos realizados sobre instancias de referencia demostraron mejoras del 12-18% en la calidad de soluciones finales y reducciones del 30-40% en el número de generaciones requeridas para alcanzar criterios de convergencia predefinidos.

Las investigaciones revisadas evidencian la madurez y efectividad de los algoritmos genéticos aplicados a problemas de generación de horarios universitarios, estableciendo bases metodológicas sólidas que el presente proyecto adapta y

contextualiza a las particularidades normativas e institucionales de PUCE Sede Ibarra, incorporando tanto técnicas de **optimización multiobjetivo** como estrategias de calibración paramétrica validadas empíricamente en diversos contextos educativos internacionales.

1.1.3 Investigaciones previas en contexto ecuatoriano

En el contexto ecuatoriano, específicamente, De la Rosa-Martín y León-González (2024) desarrollaron un sistema web integral para la creación automatizada de horarios de clases en la Universidad Metropolitana del Ecuador (UMET), abordando los desafíos operacionales generados por procesos manuales, en los que se utilizaban hojas de Excel desconectadas entre sí y que generaban errores humanos recurrentes con impactos negativos medibles en la experiencia educativa. El sistema implementado automatizó completamente el proceso de generación de horarios académicos, reduciendo sustancialmente la carga administrativa de directores de carrera quienes previamente invertían entre 40 y 60 horas semestrales en actividades manuales repetitivas de programación y resolución reactiva de conflictos. La arquitectura del sistema emplea tecnologías web modernas Laravel como *framework backend*, Vue.js para interfaces de usuario, PostgreSQL para persistencia de datos siguiendo principios de diseño que priorizan modularidad, escalabilidad, usabilidad, estabilidad y confiabilidad para garantizar eficacia sostenida y satisfacción de usuarios finales en operación continua. Este trabajo constituye un referente arquitectónico fundamental para el presente proyecto, validando empíricamente la viabilidad técnica de sistemas web desacoplados en contexto universitario ecuatoriano y demostrando que la automatización mediante tecnologías modernas puede reducir sustancialmente tiempos administrativos mientras mantiene calidad comparable a procesos manuales expertos.

Campoverde Ramos (2023) desarrolló un sistema de gestión de horarios académicos para la Facultad de Ciencias Físicas y Matemáticas de la Universidad Central del Ecuador (UCE), enfocado en automatizar los procesos de generación de horarios mediante la implementación de algoritmos de optimización combinatoria adaptados a restricciones institucionales particulares de la UCE. El trabajo demuestra empíricamente que la automatización permite agilizar significativamente el proceso de elaboración de horarios reduciendo tiempos de generación de múltiples días a pocas horas y proporciona acceso inmediato a información integral y actualizada para docentes y estudiantes mediante interfaces web responsivas accesibles desde dispositivos móviles y

computadores de escritorio. El sistema implementado incorpora funcionalidades de validación automática de restricciones en tiempo real durante la construcción manual asistida de horarios, alertando a coordinadores académicos cuando propuestas de asignación violan restricciones duras predefinidas y sugiriendo modificaciones correctivas que restauran la factibilidad de la solución en desarrollo. Esta investigación aporta evidencia contextual sobre la importancia crítica de la validación automática de restricciones institucionales específicas, principio que el presente proyecto extiende mediante la incorporación de algoritmos genéticos que no solamente validan restricciones sino que generan automáticamente soluciones óptimas que las satisfacen, eliminando la necesidad de construcción manual asistida y permitiendo la exploración exhaustiva del espacio de soluciones factibles para identificar configuraciones de máxima calidad según criterios institucionales predefinidos.

1.2 CONCEPTOS FUNDAMENTALES

1.2.1 Algoritmos genéticos

Los algoritmos genéticos constituyen una familia de metaheurísticas de optimización inspiradas en los principios de evolución biológica mediante selección natural propuestos por Charles Darwin, desarrollados formalmente en el campo de la computación por John Holland en la década de 1960 (Mitchell, 1998). Una metaheurística se define como una estrategia de alto nivel que guía procesos de búsqueda hacia soluciones óptimas o cercanas al óptimo mediante exploración inteligente del espacio de soluciones, superando limitaciones de métodos exactos cuando estos resultan computacionalmente prohibitivos (Soria- Alcaraz et al., 2023). En el contexto específico de programación académica, esta característica resulta fundamental dado que el problema de asignación de horarios universitarios pertenece a la categoría NP-difícil, donde el número de combinaciones posibles crece exponencialmente con cada materia, docente y bloque horario adicional, haciendo inviable la búsqueda exhaustiva de la solución óptima (Ceschia et al., 2023).

Estos algoritmos mantienen una población de soluciones candidatas al problema de optimización, donde cada individuo representa una configuración específica codificada típicamente como una cadena de símbolos denominada cromosoma que almacena genes o caracteres con valores específicos conocidos como alelos, cuya disposición particular en posiciones denominadas loci determina las características de la solución representada (Naaman et al., 2025). Aplicado al problema de horarios

académicos de la PUCE-SI, cada individuo representa un horario candidato completo, donde cada gen codifica la asignación de una materia específica a un docente calificado en el área correspondiente y a un bloque horario disponible, respetando las restricciones normativas institucionales definidas en el Oficio Nro. 902-Dir.DocyEst.

El desarrollo de un algoritmo genético sigue un proceso estructurado ilustrado en la Figura 1 (Naaman et al., 2025). Inicialmente, se genera una población inicial de individuos aleatorios que constituyen el punto de partida de la búsqueda evolutiva; en el dominio de horarios académicos, esto equivale a generar múltiples asignaciones aleatorias de materias a docentes y bloques horarios. Posteriormente, se evalúa la aptitud (fitness) de cada individuo mediante una función que cuantifica numéricamente su calidad como solución al problema planteado; para el caso de horarios, esta función penaliza violaciones de restricciones como conflictos de docente, incumplimiento de carga horaria según categoría, o la repetición de una materia el mismo día.

El algoritmo verifica entonces si se ha alcanzado el criterio de terminación definido: número máximo de generaciones, umbral de aptitud, o estancamiento de mejora; en caso afirmativo, el proceso concluye retornando la mejor sugerencia de horario encontrada. Si el criterio no se satisface, se procede con los operadores genéticos: selección de progenitores que favorecen horarios con menor número de conflictos y mejor distribución de carga docente; recombinación genética que combina características favorables de dos horarios parentales para generar nuevas propuestas potencialmente superiores; y mutación que introduce cambios aleatorios controlados como reasignar una materia a otro docente del mismo área o a un bloque horario diferente, previniendo convergencia prematura. Finalmente, se construye la nueva generación de horarios candidatos y el **ciclo se repite hasta satisfacer el criterio de terminación**. Este criterio constituye una decisión de diseño fundamental, pues determina cuándo detener la búsqueda y condiciona el balance entre calidad de la solución y costo computacional: un número fijo de generaciones garantiza un tiempo de respuesta predecible, un umbral de aptitud presupone conocer qué valor es alcanzable, y la detección de estancamiento ahorra cómputo a costa de tiempos variables. En cualquier caso, el algoritmo retorna la mejor solución encontrada, no necesariamente la óptima. El presente trabajo adopta el criterio de generaciones fijas, cuya calibración se detalla en el Capítulo II.

Figura 1: Proceso iterativo de un algoritmo genético

Nota. Adaptado de "*Optimization by Nature: A Review of Genetic Algorithm Techniques" (p. 271), por D. W. Naaman, B. T. Ahmed e I. M. Ibrahim, 2025, The Indonesian Journal of Computer Science, 14(1).*

En contextos de optimización educativa, particularmente en problemas de programación de horarios universitario, los algoritmos genéticos han demostrado ser especialmente efectivos frente a métodos de optimización matemática tradicionales como programación lineal entera o branch-and-bound, que resultan computacionalmente intratables debido al crecimiento exponencial del espacio de búsqueda (Yusop et al., 2022). Para la Escuela de Hábitat, Infraestructura y Creatividad de la PUCE-SI, donde deben coordinarse múltiples docentes con distintas categorías de dedicación, áreas de conocimiento certificadas y restricciones normativas específicas, esta capacidad de exploración eficiente resulta determinante para generar sugerencias de horarios académicos de calidad en tiempos computacionales razonables. La versatilidad de los algoritmos genéticos permite adaptarlos al problema institucional específico mediante: diseño de representaciones cromosómicas que codifican naturalmente la asignación materia-docente-bloque horario; definición de funciones de aptitud que cuantifican el cumplimiento de restricciones duras (carga horaria por categoría, compatibilidad área-materia) y blandas (distribución equilibrada entre jornadas, continuidad de bloques); y calibración de parámetros evolutivos que balancean velocidad de convergencia con calidad de las sugerencias generadas.

Los parámetros de configuración de un algoritmo genético influyen significativamente en su desempeño y deben calibrarse experimentalmente para cada dominio de aplicación (Fuenmayor et al., 2022). El tamaño de la población determina la amplitud de exploración simultánea: poblaciones pequeñas (20-50 individuos) convergen rápidamente, pero pueden quedar atrapadas en óptimos locales, mientras poblaciones grandes (200-500 individuos) exploran más exhaustivamente a costa de mayor tiempo computacional. La tasa de cruce, típicamente entre 0.6 y 0.9, controla la frecuencia con

que pares de horarios candidatos intercambian asignaciones para generar nuevas propuestas; valores altos favorecen la explotación de combinaciones prometedoras ya identificadas. La tasa de mutación, generalmente baja (0.01-0.1), introduce reasignaciones aleatorias que previenen la convergencia prematura y permiten explorar nuevas combinaciones docente-materia-bloque. La literatura recomienda iniciar con valores estándar validados empíricamente: población de 100 individuos, cruce de 0.7 y mutación de 0.05, ajustando posteriormente mediante análisis de sensibilidad según los resultados obtenidos en el dominio específico (Fuenmayor et al., 2022).

1.2.2 Librería DEAP (*Distributed Evolutionary Algorithms in Python*)

DEAP (*Distributed Evolutionary Algorithms in Python*) es un *framework* de código abierto para computación evolutiva de mantenimiento activo, compatibilidad con Python 3.9 y versiones posteriores (Kim & Yoo, 2019). La biblioteca proporciona componentes modulares para implementar algoritmos evolutivos mediante cuatro módulos principales: *creator* para definir tipos de datos personalizados que representan individuos y aptitudes; base para construir contenedores que almacenan poblaciones y operadores genéticos; *tools* que provee implementaciones predefinidas de operadores de selección, cruce, mutación y funciones de aptitud; y *algorithms* que ofrece algoritmos evolutivos completos como NSGA-II y estrategias (μ,λ) listos para ejecutar con configuraciones estándar (Kim & Yoo, 2019). DEAP ha demostrado efectividad en aplicaciones de programación de horarios universitarios, destacándose su uso en la optimización de horarios de laboratorios, donde Fuenmayor et al. (2022) lograron mejoras del 29% en convergencia y 14,98% en eficiencia respecto a algoritmos genéticos estándar mediante operadores de mutación controlada, consolidándose como herramienta fundamental para prototipado rápido y desarrollo de soluciones de optimización combinatoria en el ámbito educativo.

El ecosistema de *frameworks* para algoritmos genéticos en Python ofrece diversas alternativas con características y enfoques diferenciados. La tabla 1 presenta una comparación sistemática entre las principales opciones disponibles, considerando criterios técnicos relevantes para implementación de sistemas de optimización.

Tabla1:
Comparación de Frameworks para Algoritmos Genéticos en Python

Framework	Versión	Lenguaje base	Paralelización	Curva de aprendizaje	Documentación	Integración Python	Enfoque principal
DEAP (Kim & Yoo, 2019)	1.4.1	Python + C opcional	Nativa *(multiprocessing, SCOOP)*	Media	Excelente	Total (NumPy, pandas, matplotlib)	Algoritmos genéticos personalizados, investigación
PyGAD (Gad, 2024)	3.3.1	Python puro	Limitada	Baja	Buena	Básica	Aplicaciones educativas, prototipos rápidos
GAFT (PytLab, 2018)	0.6.2	Python puro	No soportada	Media – Alta	Limitada	Básica	Optimización simple de funciones
Pymoo (Blank & Deb, 2020)	0.6.1	Python + Cython	Nativa *(multiprocessing)*	Alta	Excelente	Con NumPy	Optimización multiobjetivo compleja
Platypus (Hadka, 2024)	1.3.0	Python puro	Limitada	Media	Media	Básica	Optimización multiobjetivo educativa

Leap (Coletti et al., 2020)	0.9.1	Python puro	No soportada	Alta	Limitada	Básica	Investigación en computación evolutiva

Nota. Comparación de frameworks para algoritmos genéticos disponibles en Python considerando características técnicas relevantes para implementación de sistemas de optimización.

1.2.3 Optimización multiobjetivo en contexto educativo

Los problemas de programación de horarios académicos involucran inherentemente objetivos múltiples frecuentemente conflictivos que deben balancearse simultáneamente: maximizar la utilización de recursos físicos disponibles, minimizar los tiempos ociosos de docentes, distribuir equitativamente carga horaria entre días de la semana, satisfacer preferencias individuales de estudiantes y profesores, entre otros. Las restricciones que definen la factibilidad y calidad de soluciones en problemas de programación académica se categorizan tradicionalmente en dos tipos, con tratamiento diferenciado durante el proceso de optimización.

Las restricciones duras *(hard constraints*) son requisitos obligatorios que no admiten violación bajo ninguna circunstancia. Ejemplos típicos incluyen: ningún docente puede impartir dos clases simultáneamente en diferentes aulas, ningún aula puede albergar múltiples clases concurrentemente, ningún grupo de estudiantes puede tener dos clases programadas al mismo tiempo, la capacidad física de cada aula debe ser suficiente para acomodar a todos los estudiantes matriculados en la clase asignada, y todas las asignaturas del plan de estudios deben recibir el número de horas semanales especificado en la malla curricular. Un horario que viola cualquier restricción dura se considera infactible e inaceptable para la implementación práctica, independientemente de cuán bien satisfaga otros criterios de calidad deseables.

Las restricciones blandas (*soft constraints*) representan preferencias deseables que idealmente deberían satisfacerse, pero cuya violación no invalida la viabilidad de un horario generado. Ejemplos característicos incluyen: los docentes prefieren evitar clases muy temprano en la mañana o muy tarde en la noche, los estudiantes prefieren evitar jornadas académicas con más de cuatro horas lectivas consecutivas sin pausas intermedias, las sesiones de una misma asignatura deberían distribuirse uniformemente durante la semana en lugar de concentrarse en dos días consecutivos, los docentes prefieren minimizar intervalos vacíos entre clases asignadas en un mismo día (ventanas horarias), y las asignaturas con alta demanda computacional deberían programarse preferentemente en laboratorios equipados con tecnología adecuada. La calidad global de un horario factible se mide típicamente por el grado en que minimiza violaciones de restricciones blandas, cuantificado mediante una función de aptitud que agrega penalizaciones ponderadas asociadas a cada tipo de violación identificada (Chen et al., 2020).

Un enfoque común para manejar restricciones en algoritmos genéticos consiste en diseñar representaciones cromosómicas y operadores genéticos que garanticen que toda descendencia generada satisfaga las restricciones duras por construcción, eliminando la necesidad de verificación o reparación posterior. Alternativamente, cuando garantizar factibilidad mediante codificación resulta impracticable, se pueden emplear funciones de aptitud que asignan penalizaciones severas a soluciones que violan restricciones duras, efectivamente guiando la búsqueda evolutiva hacia regiones factibles del espacio de soluciones. Las restricciones blandas se incorporan típicamente mediante términos adicionales en la función de aptitud que reducen el valor de aptitud proporcionalmente al grado de violación, permitiendo al algoritmo explorar compromisos (*trade-offs*) entre diferentes objetivos conflictivos y converger hacia soluciones que representan un balance aceptable entre múltiples criterios de calidad mediante una función de agregación ponderada (Mallari et al., 2023).

1.2.4 Función de *fitness*

La función de aptitud o *fitness* constituye el componente central que guía el proceso de búsqueda en algoritmos genéticos, cuantificando numéricamente la calidad de cada solución candidata para establecer comparaciones objetivas que determinan qué individuos tienen mayor probabilidad de reproducirse y contribuir con información genética a generaciones subsecuentes. En el contexto específico de programación académica universitaria, la función de fitness típicamente se diseña como una función de agregación ponderada que combina múltiples componentes correspondientes a diferentes aspectos de calidad del horario: penalizaciones por violaciones de restricciones blandas, métricas de utilización eficiente de recursos, indicadores de satisfacción de preferencias de actores involucrados. La formulación matemática general adopta la forma $f(x) = \Sigma_i w_i \cdot p_i(x)$, donde $p_i(x)$ representa la penalización asociada al i-ésimo tipo de violación o criterio de calidad evaluado sobre el horario x, y w_i constituye un peso que refleja la importancia relativa de ese criterio específico según prioridades institucionales.

El diseño de funciones de *fitness* efectivas para problemas de optimización combinatoria complejos requiere consideración cuidadosa de varios principios fundamentales. Primero, la función debe ser computacionalmente eficiente, dado que será evaluada miles o millones de veces durante el proceso evolutivo. Una complejidad algorítmica excesiva en el cálculo de fitness puede dominar el tiempo total de ejecución del algoritmo genético. Segundo, la función debe proporcionar gradiente informativo que

oriente la búsqueda hacia regiones prometedoras: funciones que producen valores prácticamente idénticos para soluciones vecinas (paisajes de fitness planos) dificultan la identificación de direcciones de mejora, mientras que funciones excesivamente rugosas con numerosos óptimos locales de calidad similar complican la convergencia hacia soluciones globalmente superiores. Tercero, los pesos relativos asignados a diferentes componentes deben calibrarse experimentalmente para reflejar prioridades institucionales reales y producir soluciones que sean percibidas como deseables por usuarios finales humanos: pesos inadecuados pueden resultar en horarios que optimizan métricas técnicas irrelevantes mientras ignoran aspectos prácticamente importantes para operación educativa efectiva.

1.2.5 Arquitectura de sistemas web modernos

Los sistemas web contemporáneos se fundamentan en arquitecturas de software que separan claramente las responsabilidades entre componentes de presentación, lógica de negocio, y persistencia de datos, siguiendo principios establecidos en el patrón arquitectónico Modelo-Vista-Controlador (MVC) propuesto originalmente por Trygve Reenskaug en 1979 y posteriormente adaptado extensivamente para aplicaciones web (De la Rosa-Martín & León-González, 2024). En este paradigma, el Modelo encapsula la lógica de negocio y el estado de la aplicación; la Vista se responsabiliza de presentar información al usuario mediante interfaces gráficas; el Controlador actúa como intermediario procesando solicitudes de usuarios, invocando operaciones apropiadas sobre el modelo, y seleccionando vistas adecuadas para presentar resultados.

Los *frameworks* de desarrollo web modernos proporcionan implementaciones estructuradas del patrón MVC junto con bibliotecas extensas de componentes reutilizables. El ecosistema ofrece diversas alternativas tecnológicas según el lenguaje de programación y requisitos del proyecto: Laravel (PHP) se destaca por su sintaxis expresiva y ecosistema maduro de paquetes; Django (Python) ofrece administración automática de bases de datos y enfoque "baterías incluidas"; Spring Boot (Java) proporciona robustez empresarial y escalabilidad horizontal; Ruby on Rails (Ruby) enfatiza convención sobre configuración; NestJS (TypeScript/Node.js) combina programación orientada a objetos con arquitectura modular inspirada en Angular; y Express.js (Node.js) ofrece minimalismo y flexibilidad para construir APIs ligeras (Campoverde Ramos, 2023). Estos *frameworks* proveen funcionalidades comunes como sistemas ORM para abstracción de bases de datos, enrutamiento de solicitudes HTTP,

validación de datos, autenticación y autorización, gestión de sesiones, y generación de respuestas en formatos estándar JSON o HTML.

1.2.6 Arquitectura desacoplada y separación de responsabilidades

Una arquitectura de software desacoplada se caracteriza por la separación de sistemas en módulos independientes que interactúan mediante interfaces bien definidas, minimizando las dependencias directas y facilitando la evolución independiente de cada componente sin afectar el funcionamiento de otros módulos del sistema (De la Rosa-Martín & León-González, 2024). Este enfoque arquitectónico contrasta con arquitecturas monolíticas donde todos los componentes residen en una única base de código con dependencias fuertemente acopladas.

Las arquitecturas desacopladas pueden implementarse mediante diversos patrones: arquitectura de microservicios que descompone aplicaciones en servicios pequeños e independientes desplegables separadamente; arquitectura orientada a servicios (SOA) que organiza funcionalidades como servicios empresariales reutilizables; arquitectura basada en eventos donde componentes se comunican mediante publicación y suscripción a eventos asíncronos; y arquitectura de capas que organiza código en niveles jerárquicos con dependencias unidireccionales (Davison & Drake, 2024). Las ventajas principales incluyen flexibilidad tecnológica que permite seleccionar herramientas óptimas para cada componente, escalabilidad independiente donde componentes con demandas computacionales diferentes pueden dimensionarse apropiadamente, mantenibilidad mejorada mediante aislamiento de cambios, y reutilización de componentes donde servicios especializados pueden emplearse por múltiples aplicaciones clientes sin duplicación de lógica.

1.2.7 Interfaces de programación de aplicaciones REST y alternativas

Las interfaces de programación de aplicaciones basadas en el estilo arquitectónico REST (Representational State Transfer) constituyen el mecanismo estándar de facto para comunicación entre componentes distribuidos en arquitecturas web modernas, definiendo convenciones para exponer funcionalidades mediante el protocolo HTTP utilizando sus métodos semánticos estándar: GET para recuperación de recursos, POST para creación, PUT/PATCH para actualización, DELETE para eliminación (Campoverde Ramos, 2023). REST estructura recursos mediante URIs jerárquicas que reflejan relaciones entre entidades del dominio y emplea representaciones estandarizadas como JSON o XML para intercambio de datos.

Alternativas arquitectónicas para APIs incluyen: GraphQL, desarrollado por Facebook, que permite a clientes especificar exactamente qué datos necesitan mediante consultas flexibles, reduciendo sobrecarga de múltiples solicitudes REST; gRPC, creado por Google, que utiliza Protocol Buffers para serialización binaria eficiente y soporta comunicación bidireccional mediante streaming; SOAP (Simple Object Access Protocol) que define protocolos formales basados en XML con contratos WSDL estrictos, prevalente en sistemas empresariales legacy; y WebSockets que habilitan comunicación bidireccional en tiempo real mediante conexiones persistentes, adecuado para aplicaciones colaborativas y notificaciones instantáneas (Soria-Alcaraz et al., 2023). La selección entre estas alternativas depende de requisitos específicos como necesidad de tipado fuerte, eficiencia de transferencia, compatibilidad con sistemas existentes, y patrones de comunicación requeridos.

1.2.8 Sistemas de gestión de bases de datos relacionales

Los sistemas de gestión de bases de datos relacionales (RDBMS) organizan información en tablas estructuradas con esquemas definidos, implementando el modelo relacional propuesto por Edgar Codd mediante lenguaje SQL estandarizado para manipulación de datos, garantizando propiedades ACID (Atomicity, Consistency, Isolation, Durability) que aseguran integridad transaccional (Mallari et al., 2023). Las bases de datos relacionales ofrecen ventajas significativas para sistemas de gestión académica mediante integridad referencial que previene inconsistencias mediante claves foráneas, capacidades de consulta complejas mediante operaciones de unión y agregación, normalización que elimina redundancia de datos, y soporte transaccional que garantiza consistencia ante fallos.

El ecosistema de RDBMS ofrece múltiples alternativas con características diferenciadas: PostgreSQL destaca por su conformidad con estándares SQL, extensibilidad mediante tipos de datos personalizados, soporte nativo para JSON y arrays, y capacidades avanzadas como control de concurrencia multiversión (MVCC); MySQL ofrece rendimiento optimizado para aplicaciones web con alta concurrencia de lectura y ecosistema maduro de herramientas; SQL Server proporciona integración profunda con ecosistema Microsoft y características empresariales como replicación avanzada; Oracle Database ofrece escalabilidad masiva y características empresariales robustas con costo de licenciamiento elevado; y MariaDB, derivado de MySQL, proporciona compatibilidad con MySQL mientras incorpora optimizaciones de rendimiento y características

empresariales sin restricciones de licenciamiento (Chen et al., 2020). La elección entre estas alternativas considera factores como costo de licenciamiento, compatibilidad con frameworks de desarrollo, requisitos de escalabilidad, y experiencia del equipo técnico.

1.2.9 Metodologías de desarrollo de software

Las metodologías de desarrollo de software definen procesos estructurados para planificación, ejecución, y entrega de proyectos de software, estableciendo prácticas, roles, artefactos, y ceremonias que guían equipos hacia objetivos definidos. El panorama metodológico se divide entre enfoques tradicionales predictivos y enfoques ágiles adaptativos, cada uno con fortalezas y limitaciones según las características del proyecto.

Las metodologías ágiles enfatizan entrega iterativa incremental, adaptación continua a requisitos cambiantes, y colaboración estrecha entre equipos multidisciplinarios. SCRUM organiza el trabajo en iteraciones de duración fija denominadas Sprint (típicamente de 2-4 semanas) donde cada Sprint produce un incremento potencialmente entregable del producto, definiendo roles específicos como *Product Owner, Scrum Master, y Development Team*, junto con ceremonias estructuradas como planificación del Sprint, reuniones diarias de sincronización, revisión de Sprint, y retrospectiva (Fuenmayor et al., 2022). Kanban visualiza el flujo de trabajo mediante tableros que muestran tareas en diferentes estados, limita el trabajo en progreso para prevenir la sobrecarga y optimiza continuamente el flujo mediante métricas como el tiempo de ciclo y el rendimiento.

eXtreme Programming (XP) enfatiza prácticas técnicas como programación en parejas, desarrollo guiado por pruebas (TDD), integración continua, y refactorización constante para mantener calidad de código. *Lean Software Development* adapta principios de manufactura Lean para eliminar desperdicios, optimizar flujo de valor, y entregar rápidamente funcionalidades que maximizan valor para usuarios.

La selección metodológica apropiada considera factores como estabilidad de requisitos, restricciones temporales, tamaño y distribución del equipo, criticidad del sistema, y experiencia organizacional con prácticas específicas.

1.3 Síntesis del Capítulo

La revisión realizada confirma que el problema de programación de horarios universitarios es NP-*hard*, haciendo inviables los métodos exactos (combinatorios) de optimización en instancias reales. Los algoritmos genéticos constituyen la alternativa

metaheurística más adecuada para este dominio, con efectividad comprobada en contextos educativos internacionales y ecuatorianos (Ceschia et al., 2023; De la Rosa-Martín & León-González, 2024). La distinción entre restricciones duras y blandas, la función de fitness como mecanismo de evaluación de calidad, y la arquitectura desacoplada con APIs REST como patrón de integración, constituyen los pilares conceptuales y técnicos sobre los que se diseña e implementa el prototipo de módulo de optimización propuesto para la Escuela de Hábitat, Infraestructura y Creatividad de la PUCE-SI.

CAPÍTULO II

MATERIALES Y MÉTODOS

Este capítulo describe los materiales, herramientas y métodos empleados para desarrollar el sistema web inteligente de optimización de horarios académicos en la Escuela de Hábitat, Infraestructura y Creatividad de PUCE Sede Ibarra. Se presentan las tecnologías seleccionadas para implementar el sistema y se justifica su elección según criterios técnicos. El sistema integra un módulo de gestión académica web con un motor de optimización basado en algoritmos genéticos mediante la librería DEAP. Se detalla el modelado del problema de planificación de horarios universitarios, el diseño del algoritmo genético (representación cromosómica, función de aptitud, operadores evolutivos), y los procedimientos de calibración de parámetros. El objetivo es automatizar la generación de horarios, reduciendo tiempos operativos, garantizando el cumplimiento de restricciones institucionales.

2.1 Generalidades de la investigación

2.1.1 Tipo de investigación

La presente investigación es de tipo aplicada, ya que se orienta a resolver un problema práctico específico identificado en la Escuela de Hábitat, Infraestructura y Creatividad de PUCE Sede Ibarra: la programación manual de horarios académicos mediante hojas de cálculo Excel desconectadas, que genera errores recurrentes, inconsistencias y tiempos operativos excesivos.

El carácter aplicado se evidencia en que el producto final es un sistema web funcional que automatiza la generación de horarios académicos, reduciendo significativamente los tiempos de programación y garantizando el cumplimiento ¿de restricciones normativas institucionales, lo cual representa un aporte directo a la mejora de procesos administrativos educativos en la institución.

2.1.2 Enfoque de investigación

La investigación adopta un enfoque mixto que combina elementos cuantitativos y cualitativos. El componente cuantitativo consiste en la medición sistemática y el análisis estadístico del desempeño del algoritmo genético sobre múltiples instancias reales del problema: se registran el valor de fitness inicial y final, la mejora relativa entre ambos, y el tiempo de ejecución, y se analizan mediante estadísticos descriptivos (media, desviación estándar y porcentaje respecto al máximo teórico) que permiten evaluar la calidad y estabilidad de las soluciones, y contrastar los resultados con los rangos

reportados en la literatura para problemas de planificación de horarios universitarios. De forma complementaria se registran dos indicadores adicionales: los bloques sin docente asignado, cuyo valor obligatorio es cero por tratarse de la verificación del cumplimiento de un requisito duro del problema (toda clase debe contar con un docente compatible), y el número de docentes utilizados en cada sugerencia, indicador descriptivo de la utilización de recursos que informa al coordinador sin constituir una restricción del modelo.

El componente cualitativo se aplica en el análisis de la percepción de usabilidad del sistema por parte del coordinador académico mediante entrevista y observación del proceso de adopción de la herramienta. Este enfoque cualitativo complementa los datos cuantitativos proporcionando comprensión profunda sobre la aceptación del sistema, identificación de mejoras potenciales, y validación de que la solución responde efectivamente a las necesidades reales del usuario final.

2.1.3 Técnicas de recolección de información

Para obtener información necesaria durante el desarrollo del proyecto se emplearon las siguientes técnicas:

Observación directa

Se realizó observación del proceso manual actual de programación académica en la Escuela de Hábitat, documentando las actividades que realiza el coordinador académico, los archivos Excel utilizados, los criterios de asignación aplicados, y los problemas recurrentes identificados (conflictos de horarios, sobrecarga o subcarga docente, incumplimiento de restricciones). Esta observación permitió comprender en detalle el flujo de trabajo existente y los requisitos funcionales que debía cumplir el sistema automatizado.

Entrevista semiestructurada

Se condujo una entrevista con el coordinador académico de la carrera de Ingeniería de Software de la Escuela de Hábitat, Infraestructura y Creatividad para identificar necesidades específicas, restricciones normativas institucionales que deben respetarse en la programación de horarios (límites de carga horaria según categoría docente, distribución entre jornadas, coherencia entre perfiles académicos y materias asignadas), y expectativas sobre funcionalidades del sistema. La entrevista siguió un guión semiestructurado (ver Anexo B) que permitió explorar temas emergentes durante

la conversación, facilitando la captura de requisitos que no habían sido anticipados inicialmente.

Análisis documental

Se revisaron documentos institucionales incluyendo normativas de PUCE Sede Ibarra sobre dedicación docente, planes de estudio vigentes de las carreras de la Escuela de Hábitat, horarios académicos de períodos anteriores, y reportes de carga horaria docente. Este análisis proporcionó información estructurada sobre las restricciones formales que debe cumplir el algoritmo de optimización.

2.2 Metodología para el desarrollo

El proyecto se desarrolló aplicando la metodología ágil Scrum mediante *Sprints* de dos semanas con entregas incrementales de funcionalidad. El sistema implementa dos componentes principales: un módulo de gestión académica basado en tecnologías web que centraliza la administración de docentes, materias y aulas, y un motor de optimización que utiliza algoritmos genéticos para generar horarios automáticamente. La integración entre ambos componentes se realizó mediante arquitectura REST, permitiendo que los coordinadores académicos gestionen recursos y ejecuten el proceso de optimización desde cualquier estación de trabajo conectada a la red institucional. Los detalles de planificación y distribución de historias de usuario en *Sprint* se presentan en la sección 2.3.

Se considera conveniente enfatizar dos aspectos clave:

Propósito del estudio

Desarrollar un sistema web inteligente que automatice dos procesos fundamentales en la programación académica de la Escuela de Hábitat, Infraestructura y Creatividad de PUCE Sede Ibarra. Primero, la gestión estructurada y centralizada de recursos académicos (docentes, materias, aulas) mediante una interfaz web moderna que elimina la dependencia de hojas de cálculo Excel desconectadas. Segundo, la generación automática de horarios académicos mediante algoritmos genéticos que exploran el espacio de soluciones para encontrar configuraciones que cumplan simultáneamente restricciones normativas institucionales sobre asignación de carga horaria docente, distribución equilibrada entre jornadas, y coherencia entre perfiles académicos y contenidos de materias asignadas. Ambos componentes fueron diseñados para trabajar de forma integrada, de modo que la plataforma web proporcione una interfaz intuitiva y

funcional para coordinadores académicos, mientras que el motor de optimización realice la búsqueda evolutiva de soluciones óptimas en segundo plano, generando horarios factibles que maximicen la calidad según criterios institucionales predefinidos.

Alcance

Este trabajo abarcó el análisis, diseño, desarrollo e integración de dos componentes principales: un módulo de gestión académica web y un motor de optimización basado en algoritmos genéticos. El módulo de gestión académica ofrece a coordinadores académicos funcionalidades de registro y administración de información de docentes incluyendo categoría institucional, áreas de conocimiento y disponibilidad horaria, gestión del catálogo de materias del plan de estudios con sus respectivos requerimientos de horas semanales y aulas especializadas, administración de infraestructura física disponible especificando capacidades y equipamiento, y visualización de reportes de programación por docente, aula o jornada con exportación a formato Excel para distribución institucional. El motor de optimización fue desarrollado para modelar el problema de *timetabling* universitario específico de la Escuela de Hábitat como un problema de optimización combinatoria multiobjetivo, implementando un algoritmo genético mediante la librería DEAP que codifica horarios académicos como cromosomas y aplica operadores evolutivos de selección, cruce y mutación calibrados experimentalmente. El sistema genera soluciones que cumplen restricciones duras institucionales como no asignar múltiples clases simultáneas a un mismo docente o aula, respetar límites de carga horaria según categoría docente, y garantizar que todas las materias reciban las horas especificadas en malla curricular, mientras optimiza restricciones blandas como distribución equilibrada entre jornadas matutina y vespertina, coherencia entre perfiles académicos docentes y contenidos de materias, minimización de ventanas horarias, y concentración de carga académica en menor número de días semanales. El sistema integrado permite a coordinadores académicos cargar información de recursos al inicio de cada semestre, ejecutar el proceso de optimización mediante interfaz web, visualizar la sugerencia de horario generada con sus métricas de calidad (fitness, tiempo, docentes utilizados, desglose de penalizaciones), revisar y ajustar manualmente si es necesario según necesidades institucionales, y exportar horarios aprobados para distribución a la comunidad educativa.

2.3 Fase 1: identificación y documentación de requisitos funcionales

El primer paso del proyecto consistió en identificar y documentar los requisitos funcionales del sistema mediante la técnica de historias de usuario. Este proceso se llevó a cabo a través de reuniones de trabajo con los coordinadores académicos de la Escuela de Hábitat, Infraestructura y Creatividad, donde se analizó el proceso manual actual de programación académica, se identificaron las restricciones normativas que debe cumplir el sistema, y se definieron las funcionalidades necesarias para automatizar la generación de horarios.

2.3.1 Historias de usuario

Se identificaron 15 historias de usuario que contemplan las funcionalidades completas del sistema, desde la gestión de usuarios y recursos académicos (docentes, materias, aulas) hasta la implementación del algoritmo genético para generar horarios optimizados y la visualización de resultados. En la Tabla 2 se presenta cada historia con su descripción, criterios de aceptación, estimación de esfuerzo, nivel de prioridad, y dependencias entre funcionalidades.

Tabla 2: Historias de Usuario del Sistema de Optimización de Horarios Académicos

ID	NOMBRE	ESTIMACIÓN ESFUERZO (HORAS)	IMPORTANCIA	DESCRIPCIÓN DE HISTORIA DE USUARIO	CRITERIOS DE ACEPTACIÓN	DEPENDENCIAS
1	Autenticar usuario con roles	10	500	Como usuario del sistema, deseo ingresar con usuario y contraseña para acceder según mi rol asignado (Administrador, Coordinador, Docente).	• Validar credenciales contra la base de datos. • Obtener el rol del usuario autenticado. • Cargar menú dinámico según permisos del rol. • Mostrar error si las credenciales son inválidas.	-
2	Configurar estructura académica jerárquica	20	500	Como coordinador académico, deseo registrar la jerarquía académica (Escuela → Carreras → Planes → Materias) para estructurar la información institucional.	• Registrar escuela con datos básicos. • Crear carreras vinculadas a escuela. • Crear planes de estudio vinculados a carreras (ej: TI-2019). • Registrar materias con nivel/semestre, horas docencia, horas prácticas, horas autónomas, área de conocimiento.	1
3	Gestionar áreas de conocimiento	12	400	Como coordinador académico, deseo configurar áreas de conocimiento (Programación, BD, Redes, Seguridad) para agrupar materias y vincular docentes según su experticia.	• Crear áreas de conocimiento (Programación, Base de Datos, Redes, Seguridad, etc.). • Permitir editar y eliminar áreas. • Asociar materias a áreas. • Asociar docentes a una o varias áreas.	1

4	Registrar docentes con dedicación y áreas	18	500	Como coordinador académico, deseo registrar docentes especificando su dedicación (Tiempo Completo, Medio Tiempo, Tiempo Parcial) y áreas de conocimiento para determinar límites de carga horaria y materias asignables.	• Crear usuario con datos personales y credenciales. • Crear perfil docente con dedicación: Tiempo Completo (16-20h semanales), Medio Tiempo (8-12h), Tiempo Parcial (4-6h). • Asignar áreas de conocimiento que domina el docente. • Ingresar horas mínimas y máximas de cada docente.	1, 3
5	Configurar recursos físicos y horarios	14	450	Como coordinador académico, deseo configurar recursos físicos (aulas, laboratorios) y estructura temporal (bloques horarios, días, paralelos) para establecer el espacio de soluciones del algoritmo genético.	• Registrar aulas y laboratorios con capacidad y tipo. • Definir bloques horarios (típicamente de 1 hora desde 07:00 hasta 18:00). • Crear paralelos (A, B). • Registrar días hábiles (lunes-viernes).	1
6	Crear período académico	8	400	Como coordinador académico, deseo crear períodos académicos con fechas de inicio y fin para delimitar el contexto de programación de horarios.	• Registrar período con código único (ej: 2025-1). • Especificar fecha inicio y fin. • Asociar carreras y niveles a programar en este período. • Validar que no se solapen períodos.	1
7	Seleccionar parámetros	10	450	Como coordinador académico, deseo seleccionar el período, carrera, plan de estudio y niveles	• Seleccionar período académico. • Seleccionar carrera a programar. •	2, 6

	para generación			antes de ejecutar la generación de horarios para delimitar el alcance del proceso.	Seleccionar plan de estudio. • Seleccionar niveles/semestres a incluir en horario. • Validar que existan datos completos antes de permitir generación.	
8	Ejecutar algoritmo genético para generación de horarios	45	500	Como coordinador académico, deseo ejecutar el algoritmo genético que reciba materias, docentes, aulas, bloques y días, genere combinaciones, evalúe restricciones y evolucione soluciones mediante selección, cruce y mutación hasta encontrar la mejor solución que cumplan todas las restricciones institucionales.	• Recopilar todas las materias a programar con sus horas. • Recopilar docentes disponibles con áreas y límites de carga. • Recopilar aulas, bloques horarios, días disponibles. • Implementar restricciones duras: no conflictos docentes/aula, docentes solo en materias de su área, carga dentro de límites según dedicación. • Aplicar operadores evolutivos: selección, cruce, mutación. • Mostrar métricas de ejecución: fitness inicial vs final, tiempo de generación, docentes utilizados y desglose de penalización por tipo de restricción. • Generar horario óptimo factible.	2, 3, 4, 5, 7
9	Guardar horarios generados en BD	16	450	Como sistema, deseo guardar los horarios generados en base de datos vinculando período, materia, docente, paralelo, aula,	• Guardar cada asignación vinculando: período, materia, docente, paralelo, aula, bloque horario, día. • Guardar	8

				bloque horario y día para consultas posteriores.	cada asignación vinculando período, materia, docente, paralelo, aula, bloque horario y día, garantizando integridad referencial en cada inserción.	
10	Visualizar horarios por diferentes vistas	20	500	Como usuario del sistema, deseo visualizar los horarios generados desde múltiples perspectivas (docente,nivel/paralelo) para consultar información según mi necesidad.	• Visualizar horario por docente: grilla semanal completa del profesor. • Visualizar horario por nivel y paralelo: horario de estudiantes. • Aplicar filtros por período, carrera. • Mostrar información completa en cada celda: materia, docente, aula.	9
11	Generar reporte de carga docente	14	400	Como coordinador académico, deseo generar reportes de carga docente que comparen horas asignadas contra el rango permitido según dedicación para identificar sobrecarga o subcarga y realizar ajustes manuales.	• Calcular horas asignadas por docente. • Comparar contra rango permitido según dedicación (ej: 16-20h para TC). • Identificar docentes bajo mínimo (subcarga). • Identificar docentes sobre máximo (sobrecarga). • Generar reporte exportable con alertas visuales.	9
12	Ajustar horarios manualmente	18	350	Como coordinador académico, deseo realizar ajustes manuales sobre horarios generados para corregir casos específicos o	• Permitir mover asignaciones entre bloques horarios. • Cambiar aula asignada. • Reasignar docente a materia. • Validar restricciones al	9, 10

				completar carga docente identificada en reportes.	asignar horarios manualmente, detectando conflictos de docente, aula y solapamiento antes de guardar.	
13	Exportar horarios a Excel	12	400	Como coordinador académico, deseo exportar horarios aprobados a formato Excel para distribución a docentes, estudiantes y comunidad académica.	• Exportar horario completo a Excel. • Incluir logos institucionales. • Formato legible y profesional.	10
14	Gestionar menús dinámicos por rol	16	300	Como administrador del sistema, deseo gestionar opciones de menú y asignar permisos por rol para que usuarios vean automáticamente opciones según sus permisos sin modificar código.	• Crear nuevas opciones de menú. • Asignar menús a roles específicos. • Modificar permisos sin cambiar código. • Cargar menú dinámicamente según rol en cada inicio de sesión. • Ocultar automáticamente opciones cuando se quita permiso.	1
15	Dashboard de docente	10	250	Como docente, deseo ver mi horario asignado y carga horaria en un dashboard personalizado para conocer mi programación semanal.	• Mostrar grilla semanal personal. • Mostrar total de horas asignadas. • Listar materias que dicta. • Mostrar aulas asignadas. • Solo lectura (sin edición).	1, 10

Nota. Requisitos funcionales documentados mediante sesiones de trabajo con coordinadores académicos. El esfuerzo está estimado en horas de desarrollo. La prioridad se define en una escala de 0 a 500 según la importancia funcional.

Con el propósito de estimar con mayor precisión el esfuerzo requerido y facilitar la planificación detallada del desarrollo, cada historia de usuario se descompuso en tareas específicas más pequeñas. En la Tabla 3 se presenta el análisis desglosado de las 15 historias de usuario en 97 tareas concretas, organizadas por historia y ordenadas según su nivel de prioridad de mayor a menor (500 a 300). Este desglose permitió identificar las actividades específicas que debían ejecutarse para completar cada funcionalidad del sistema.

Tabla 3: Análisis de Historias de Usuario Desglosadas en Tareas

Número Historia	Nombre	Prioridad	ID Tarea	Descripción	Horas
HU-1	Autenticar usuario con roles	500	T1-1	Diseñar modelo de datos de usuarios y roles	2
			T2-1	Implementar validación de credenciales	2
			T3-1	Desarrollar sistema de sesiones	2
			T4-1	Crear carga dinámica de menús por rol	3
			T5-1	Implementar manejo de errores	1
HU-2	Configurar estructura académica jerárquica	500	T1-2	Diseñar modelo jerárquico	3
			T2-2	Implementar CRUD de Escuelas	2
			T3-2	Implementar CRUD de Carreras	3
			T4-2	Implementar CRUD de Planes	3
			T5-2	Implementar CRUD de Materias	5
			T6-2	Validaciones de integridad	2
			T7-2	Interfaces de gestión	2
HU-4	Registrar docentes con dedicación	500	T1-4	Diseñar modelo de docentes	2
			T2-4	Crear usuarios docentes	3
			T3-4	Perfiles con dedicación	3
			T4-4	Configuración de horas mínimas y máximas según normativa institucional	2
			T5-4	Asignación de áreas	4
			T6-4	Validaciones de carga	2

			T7-4	Interfaces de gestión	2
			T8-4	Apertura automática de asignación de áreas al crear docente	2
HU-8	Ejecutar algoritmo genético	500	T1-8	Identificar y clasificar las restricciones institucionales en duras y blandas, definiendo la penalización asociada a cada una	3
			T2-8	Diseñar la representación cromosómica: tres genes por clase (día, bloque, docente), con materia y paralelo implícitos por posición	5
			T3-8	Implementar en Python el generador de individuos válidos (compatibilidad área-docente garantizada por construcción) y las funciones de decodificación del cromosoma	4
			T4-8	Implementar la función de aptitud (fitness): cálculo de las penalizaciones por restricciones duras y blandas según la fórmula $F(I) = \max(10000 - \Sigma P_i, 0)$	8
			T5-8	Implementar la selección por torneo (k=3)	3
			T6-8	Cruce de dos puntos (cxTwoPoint de DEAP)	4
			T7-8	Mutación con consistencia de docente por materia	4
			T8-8	Validador de restricciones	5
			T9-8	Integrar los operadores y la función de aptitud en el flujo eaSimple de DEAP	4
			T10-8	Métricas de ejecución: fitness inicial vs final, tiempo de generación, docentes	3

				utilizados y desglose de penalización por restricción	
			T11-8	Pruebas con datos reales	2
HU-10	Visualizar horarios	500	T1-10	Diseñar los componentes Vue reutilizables de visualización: grilla semanal (días × bloques horarios), celda de clase (materia, docente, aula) y selectores de filtrado	3
			T2-10	Vista por docente	5
			T3-10	Vista por nivel/paralelo	4
			T4-10	Implementar los filtros en cascada (periodo → carrera → pensum → nivel/paralelo y filtro por docente), con recarga dinámica de opciones según la selección previa	4
			T5-10	Carga automática de horarios IA confirmados en modo edición al volver a la pestaña de generación	2
HU-5	Configurar recursos físicos	450	T1-5	Diseñar modelo de aulas	2
			T2-5	CRUD de aulas	3
			T3-5	Bloques horarios	2
			T4-5	Configurar paralelos	2
			T5-5	Días hábiles	1
			T6-5	Interfaces	2
HU-7	Seleccionar parámetros	450	T1-7	Interfaz de selección	2
			T2-7	Selectores	3
			T3-7	Selección de niveles	2
			T4-7	Validaciones	2
			T5-7	Configuración de parámetros por nivel y paralelo: rango de horas, días permitidos y duración de bloques	1
HU-9	Guardar horarios en BD	450	T1-9	Modelo de asignaciones	3

			T2-9	Almacenamiento	4
			T3-9	Registro de observación por tipo de origen: horarios marcados como 'Generado por IA' para distinguirlos de asignaciones manuales y permitir recarga automática en modo edición	3
			T4-9	Transacciones ACID	2
			T5-9	Guardado de docente asignado por bloque en columna doc_id de tbl_horarios	2
HU-3	Gestionar áreas de conocimiento	400	T1-3	Diseñar modelo	1
			T2-3	CRUD de áreas	3
			T3-3	Asociar materias	3
			T4-3	Asociar docentes	3
			T5-3	Validaciones	2
HU-6	Crear período académico	400	T1-6	Modelo de períodos	1
			T2-6	CRUD de períodos	2
			T3-6	Validar solapamiento	2
			T4-6	Asociar carreras	2
			T5-6	Interfaz de gestión	1
HU-11	Reporte de carga docente	400	T1-11	Diseñar reporte	2
			T2-11	Calcular horas	3
			T3-11	Comparar rangos	3
			T4-11	Identificar subcarga	2
			T5-11	Visualizaciones alertas	2
			T6-11	Exportar Excel	2
HU-13	Exportar horarios	400	T1-13	Exportar Excel	2
HU-12	Ajustar horarios manualmente	350	T1-12	Interfaz drag-drop	4
			T2-12	Mover bloques	4
			T3-12	Cambiar aula	2

			T4-12	Reasignar docente	3
			T5-12	Validación de conflictos al asignar horarios manuales: detecta solapamiento de docente, exceso de horas semanales antes de guardar	3
HU-14	Gestionar menús dinámicos	300	T1-14	Modelo de menús	3
			T2-14	CRUD de opciones	4
			T3-14	Asignar a roles	3
			T4-14	Carga dinámica	4
			T5-14	Interfaz admin	2
HU-15	Dashboard de docente	300	T1-15	Diseñar dashboard	2
			T2-15	Visualizar grilla	3
			T3-15	Resumen carga	2
			T4-15	Listar materias	2
			T5-15	Solo lectura	1

Nota. Desglose de historias de usuario en tareas específicas agrupadas por prioridad. Total: 97 tareas, 243 horas.

Una vez identificadas y descompuestas las historias de usuario, se procedió a distribuirlas en sprints de dos semanas de duración. La Tabla 4 presenta la planificación de 6 sprints que cubren las 12 semanas de desarrollo del proyecto. Esta distribución se realizó considerando las dependencias técnicas entre historias, la prioridad de cada funcionalidad, y la capacidad de trabajo disponible en cada sprint (aproximadamente 40-50 horas efectivas por semana).

Tabla 4: Distribución de Historias de Usuario en Sprints

SPRINT	TAREA
1	HU-1: Autenticar usuario con roles (10h) HU-14: Gestionar menús dinámicos por rol (16h) HU-3: Gestionar áreas de conocimiento (12h) HU-6: Crear período académico (8h) Total: 46 horas

2	HU-2: Configurar estructura académica jerárquica (20h) HU-4: Registrar docentes con dedicación y áreas (20h) HU-5: Configurar recursos físicos y horarios (12h) Total: 52 horas
3	HU-8: Ejecutar algoritmo genético - PARTE 1 (45h) • Análisis de restricciones institucionales• Diseño y codificación del cromosoma con consistencia de docente por materia• Implementación de la función de fitness con penalizaciones por tipo de restricción• Operadores evolutivos: selección por torneo, cruce de dos puntos (cxTwoPoint), mutación con propagación de cambio de docente entre bloques Total: 45 horas
4	HU-8: Ejecutar algoritmo genético - PARTE 2 • Penalización acumulativa de carga horaria entre paralelos generados en la misma sesión • Integración con DEAP y pruebas con datos reales HU-7: Seleccionar parámetros y configurar generación por nivel y paralelo (10h) HU-9: Guardar horarios generados en BD (12h) Total: 22 horas
5	HU-10: Visualizar horarios por diferentes vistas (16h) HU-11: Generar reporte de carga docente (14h) HU-13: Exportar horarios a Excel (6h) Total: 36 horas
6	HU-12: Ajustar horarios manualmente (16h) HU-15: Dashboard estadístico por rol (18h) Pruebas finales y ajustes (10h) Total: 44 horas

Nota. Planificación de 6 Sprint de 2 semanas cada uno (12 semanas totales). Jornada de trabajo: 8 horas diarias, 5 días a la semana (40 horas semanales, 80 h oras por sprint). Total: 253 horas incluyendo desarrollo y pruebas.

2.4 DISEÑO DEL SISTEMA

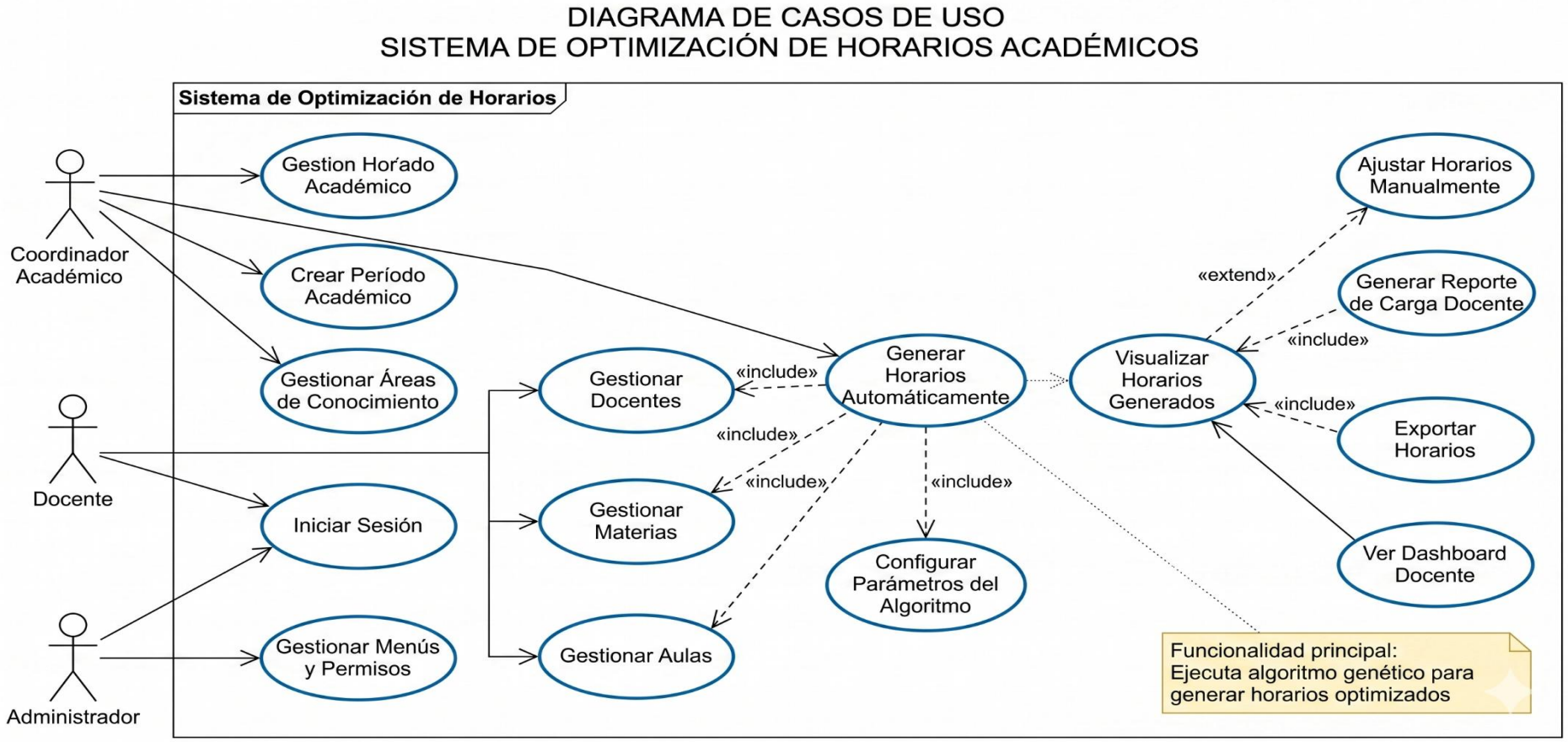
El diseño del sistema se desarrolló siguiendo principios de ingeniería de software orientada a objetos, definiendo la arquitectura, el modelo de datos, el comportamiento del sistema y las interfaces de usuario que permiten la interacción efectiva con los usuarios finales. Esta sección presenta los artefactos de diseño que guiaron la implementación del sistema web inteligente de optimización de horarios académicos.

2.4.1 Casos de uso

Los casos de uso modelan las interacciones entre los actores (usuarios del sistema) y el sistema para alcanzar objetivos específicos. Esta técnica de modelado UML *(Unified Modeling Language*) permite identificar las funcionalidades del sistema desde la perspectiva del usuario, facilitando la comprensión de los requisitos funcionales y la validación de que el sistema cubre todas las necesidades identificadas.

En el sistema de optimización de horarios académicos se identificaron tres tipos de actores con diferentes niveles de acceso y responsabilidades: el Coordinador Académico, quien gestiona los recursos académicos y ejecuta el proceso de generación de horarios; el Docente, quien consulta su horario asignado y carga horaria; y el Administrador, quien configura permisos y opciones de menú del sistema. La Figura 2 presenta el diagrama de casos de uso completo del sistema, mostrando las 14 funcionalidades principales identificadas y las relaciones de dependencia entre ellas.

Figura 2: Diagrama de Casos de Uso del Sistema de Optimización de Horarios Académicos



Nota. El diagrama muestra los tres actores del sistema y las 14 funcionalidades principales, incluyendo relaciones de inclusión (*include*) donde un caso de uso requiere la ejecución de otro, y relaciones de extensión (*extend*) donde un caso de uso añade funcionalidad opcional a otro.

La Tabla 5 detalla los casos de uso principales del sistema, especificando actores, descripción, precondiciones, flujo de eventos y postcondiciones.

Tabla 5: Descripción de casos de uso principales del sistema

CASO DE USO	**UC-01: Iniciar Sesión**
Actor	Coordinador Académico, Docente, Administrador
Descripción	El usuario ingresa sus credenciales para acceder al sistema. El sistema valida las credenciales, identifica el rol del usuario, y carga el menú dinámico según los permisos asignados a su rol.
Precondiciones	• El usuario debe estar registrado en el sistema • El usuario debe tener un rol asignado
Flujo Normal	1. El usuario ingresa nombre de usuario y contraseña 2. El sistema valida las credenciales 3. El sistema identifica el rol del usuario 4. El sistema consulta los permisos del rol 5. El sistema carga el menú con opciones permitidas 6. El sistema muestra el dashboard correspondiente
Postcondiciones	• El usuario está autenticado en el sistema • El menú dinámico está cargado según su rol • Se registra el inicio de sesión en logs
CASO DE USO	**UC-02: Gestionar Docentes**
Actor	Coordinador Académico
Descripción	El coordinador académico centraliza la gestión de docentes en el sistema web, registrando información que anteriormente se manejaba en hojas de cálculo Excel dispersas. El sistema permite crear, consultar, actualizar y eliminar registros de docentes con sus categorías de dedicación, áreas de conocimiento asignadas, y su carga horaria.
Precondiciones	• El coordinador debe estar autenticado • Deben existir áreas de conocimiento registradas
Flujo Normal	• El coordinador accede al módulo de gestión de docentes • El coordinador selecciona crear nuevo docente • El sistema muestra formulario con campos requeridos • El coordinador ingresa: nombre, cédula, email, categoría de dedicación

	• El coordinador ingresa manualmente las horas mínimas y máximas conforme a la normativa institucional vigente • El coordinador asigna una o más áreas de conocimiento al docente • El coordinador confirma el registro • El sistema valida la información y guarda en base de datos • El sistema muestra mensaje de confirmación
Postcondiciones	• Docente registrado en la base de datos centralizada • Límites de carga horaria registrados según normativa institucional • Áreas de conocimiento asociadas al docente • Información disponible para el algoritmo de generación
CASO DE USO	**UC-03: Gestionar Materias**
Actor	Coordinador Académico
Descripción	El coordinador académico centraliza la gestión de materias del plan de estudios, información que antes se mantenía en archivos Excel desconectados. El sistema permite registrar materias con sus atributos completos (horas docencia, horas prácticas, nivel académico, área de conocimiento) y asegurar consistencia entre carreras y planes de estudio.
Precondiciones	• El coordinador debe estar autenticado • Debe existir una estructura académica configurada (Escuela→Carrera→Plan) • Deben existir áreas de conocimiento registradas
Flujo Normal	• El coordinador accede al módulo de gestión de materias • El coordinador selecciona el plan de estudios correspondiente • El coordinador crea nueva materia

	• El sistema muestra formulario con campos: código, nombre, nivel, horas docencia, horas prácticas, área • El coordinador ingresa la información de la materia • El coordinador asocia la materia a un área de conocimiento • El sistema valida que el área de conocimiento exista • El coordinador confirma el registro • El sistema guarda la materia en base de datos
Postcondiciones	• Materia registrada con todos sus atributos • Asociación materia-área de conocimiento establecida • Datos disponibles para generación de horarios
CASO DE USO	**UC-04: Gestionar Aulas**
Actor	Coordinador Académico
Descripción	El coordinador académico centraliza la gestión de recursos físicos (aulas y laboratorios) que antes se manejaban manualmente en Excel. El sistema permite registrar aulas con capacidad, tipo, y disponibilidad horaria, asegurando que el algoritmo asigne espacios adecuados según las necesidades.
Precondiciones	• El coordinador debe estar autenticado
Flujo Normal	1. El coordinador accede al módulo de gestión de aulas 2. El coordinador crea nueva aula 3. El sistema muestra formulario: código, nombre, capacidad, tipo 4. El coordinador ingresa información del aula 5. El coordinador especifica el tipo (Aula estándar, Laboratorio, Auditorio) 6. El coordinador confirma el registro

	7. El sistema valida los datos y guarda en base de datos
Postcondiciones	• Aula registrada en sistema centralizado • Información de capacidad y tipo almacenada • Recurso disponible para asignación automática por el algoritmo genético y para asignación manual por el coordinador
CASO DE USO	**UC-08: Generar Horarios Automáticamente**
Actor	Coordinador Académico
Descripción	El coordinador académico ejecuta el algoritmo genético para generar horarios académicos optimizados automáticamente, reemplazando el proceso manual que antes tomaba días de trabajo en Excel. El sistema lee toda la información centralizada (docentes, materias, aulas) y genera horarios que cumplen restricciones institucionales minimizando conflictos.
Precondiciones	• Existen docentes registrados con áreas asignadas • Existen materias registradas en el plan de estudios • Existen aulas disponibles registradas • Existe un período académico activo • Se han configurado parámetros del algoritmo
Flujo Normal	1. El coordinador accede al módulo de generación de horarios 2. El coordinador selecciona período académico 3. El coordinador selecciona carrera y niveles a procesar 4. El sistema valida que existan datos completos para todos los niveles 5. El sistema muestra resumen: cantidad de materias, docentes disponibles, aulas 6. El coordinador confirma la generación 7. El sistema carga datos desde la base de datos centralizada 8. El sistema inicializa población de horarios aleatorios 9. El sistema ejecuta algoritmo genético (selección, cruce, mutación)

	10. El sistema evalúa fitness de cada solución 11. El sistema itera hasta alcanzar criterio de parada 12. El sistema selecciona la mejor solución encontrada 13. El sistema presenta la mejor solución encontrada 14. El sistema muestra métricas de calidad 15. El sistema presenta el horario generado para en grilla editable
Postcondiciones	• Horarios generados y almacenados en base de datos • Restricciones duras minimizadas mediante penalizaciones por conflicto de docente, cruce de estudiantes, carga horaria fuera de rango y concentración de materias por docente. • Restricciones blandas optimizadas según función fitness • Métricas de calidad calculadas y disponibles • Horarios listos para visualización, ajustes y exportación • Tiempo de generación reducido de días a minutos
CASO DE USO	**UC-09: Visualizar Horarios Generados**
Actor	Coordinador Académico
Descripción	El coordinador visualiza los horarios generados automáticamente desde diferentes perspectivas (por docente, nivel/paralelo) para evaluar su calidad, identificar patrones y detectar posibles ajustes necesarios. Esta visualización centralizada reemplaza la revisión manual dispersa en múltiples hojas Excel.
Precondiciones	• Existen horarios generados en el sistema • Los horarios están asociados a un período académico
Flujo Normal	1. El coordinador accede al módulo de visualización 2. El coordinador selecciona el período académico 3. El coordinador visualiza el horario completo generado 4. El coordinador selecciona el elemento específico a visualizar

	5. El sistema consulta las asignaciones desde la base de datos 6. El sistema genera la grilla de horarios semanal 7. El sistema muestra información completa: materia, docente, aula, bloque horario, día 8. El sistema calcula y muestra carga horaria total 9. El coordinador puede cambiar entre diferentes vistas sin recargar
Postcondiciones	• El coordinador visualiza el horario solicitado • Carga horaria calculada y mostrada • Información completa y legible en formato grilla

Nota. Casos de uso críticos que muestran la transición del proceso manual en Excel al sistema web centralizado. Incluye gestión de recursos (UC-02 a UC-04) y el caso de uso principal de generación automática (UC-08).

2.4.2 Arquitectura del sistema

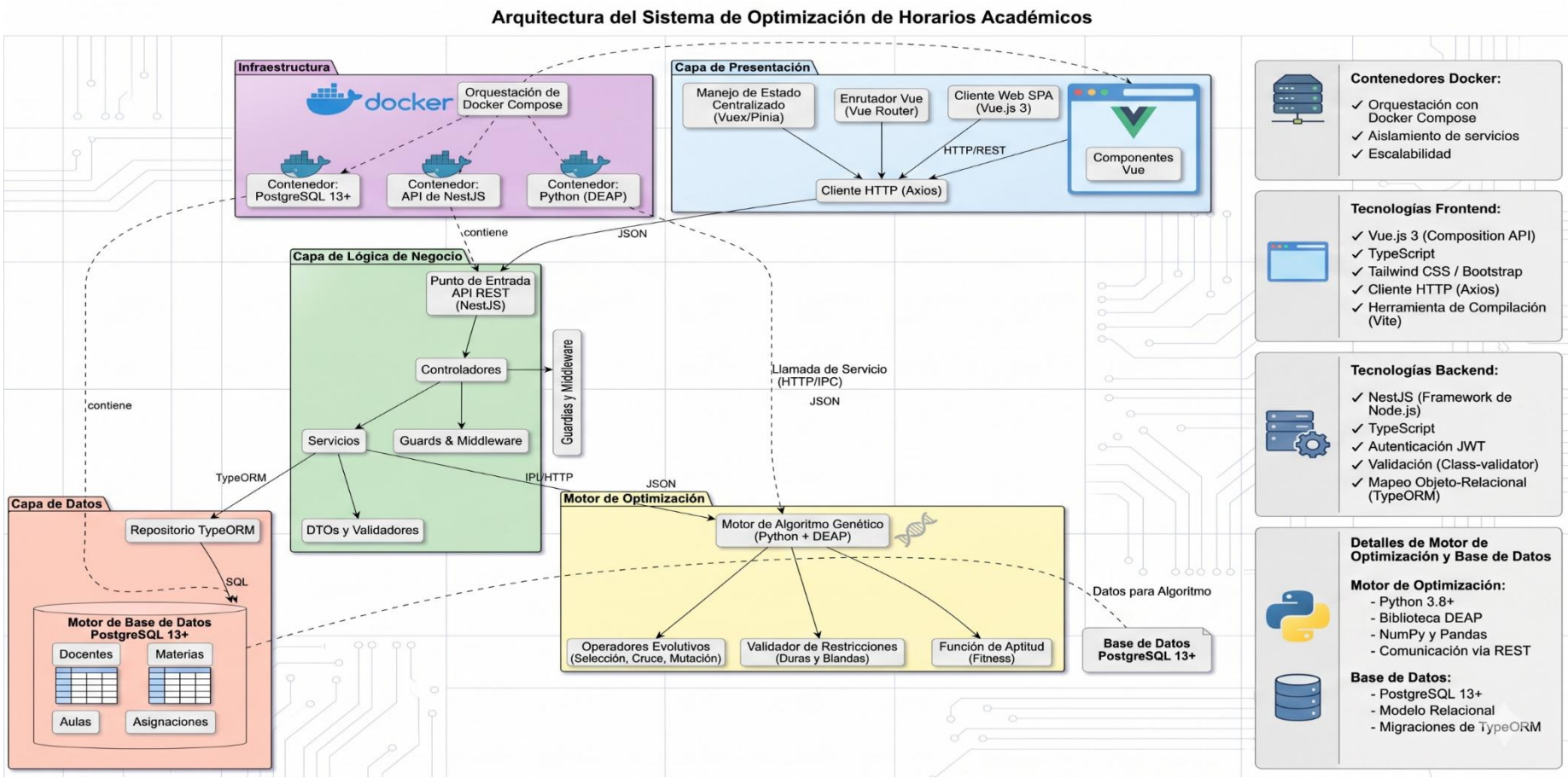
El sistema sigue una arquitectura en capas implementada mediante contenedores Docker. La Figura 3 muestra las cuatro capas principales más la infraestructura de orquestación.

La Capa de Presentación es una SPA construida con Vue.js 3 y TypeScript que se comunica mediante HTTP/JSON con la Capa de Lógica de Negocio implementada en NestJS (framework Node.js). NestJS orquesta las operaciones del sistema usando controladores, servicios, y TypeORM para persistencia en PostgreSQL.

El Motor de Optimización es un servicio Python independiente que ejecuta el algoritmo genético con DEAP, comunicándose con NestJS vía REST/IPC. La Capa de Datos usa PostgreSQL 16+ con TypeORM para gestión relacional.

Docker Compose orquesta tres contenedores: NestJS API, PostgreSQL y Python DEAP, permitiendo un despliegue consistente y la escalabilidad por componente.

Figura 3: Arquitectura en capas del sistema de optimización de horarios académicos



Nota. El sistema se organiza en cinco capas, se incluye la capa de infraestructura de Docker. Las líneas sólidas indican comunicación directa, las líneas punteadas indican contención dentro de contenedores.

La Tabla 6 presenta el *stack* tecnológico completo utilizado en cada capa del sistema, incluyendo versiones específicas y librerías complementarias.

Tabla 6: Stack tecnológico utilizado en el sistema

CAPA	TECNOLOGÍAS
Capa de Presentación (Frontend)	• Vue.js 3 (Composition API) • TypeScript • Tailwind CSS / Bootstrap 5 (responsive design) • Vite (build tool) • Axios (HTTP client) • Vuex / Pinia (state management) • Vue Router (navegación SPA)
Capa de Lógica de Negocio (Backend)	• NestJS 10+ (Node.js framework) • TypeScript 5+ • JWT (JSON Web Tokens) para autenticación • class-validator (validación de DTOs) • class-transformer (transformación de objetos) • TypeORM (ORM para PostgreSQL) • Passport.js (estrategias de autenticación)
Motor de Optimización	• Python 3.9+ • DEAP 1.4.1+ (algoritmos evolutivos) • NumPy (cálculos numéricos) • Flask / FastAPI (API REST interna)
Capa de Datos	• PostgreSQL 16+ • TypeORM (ORM) • TypeORM migrations (control de esquema) • pg (PostgreSQL driver para Node.js)
Infraestructura	• Docker 20+ • Docker Compose 2+ • Node.js 18+ (runtime para NestJS) • Python 3.9+ (runtime para DEAP)

Nota. Tecnologías seleccionadas considerando arquitectura moderna, tipado fuerte (TypeScript), escalabilidad mediante contenedores, y compatibilidad entre componentes.

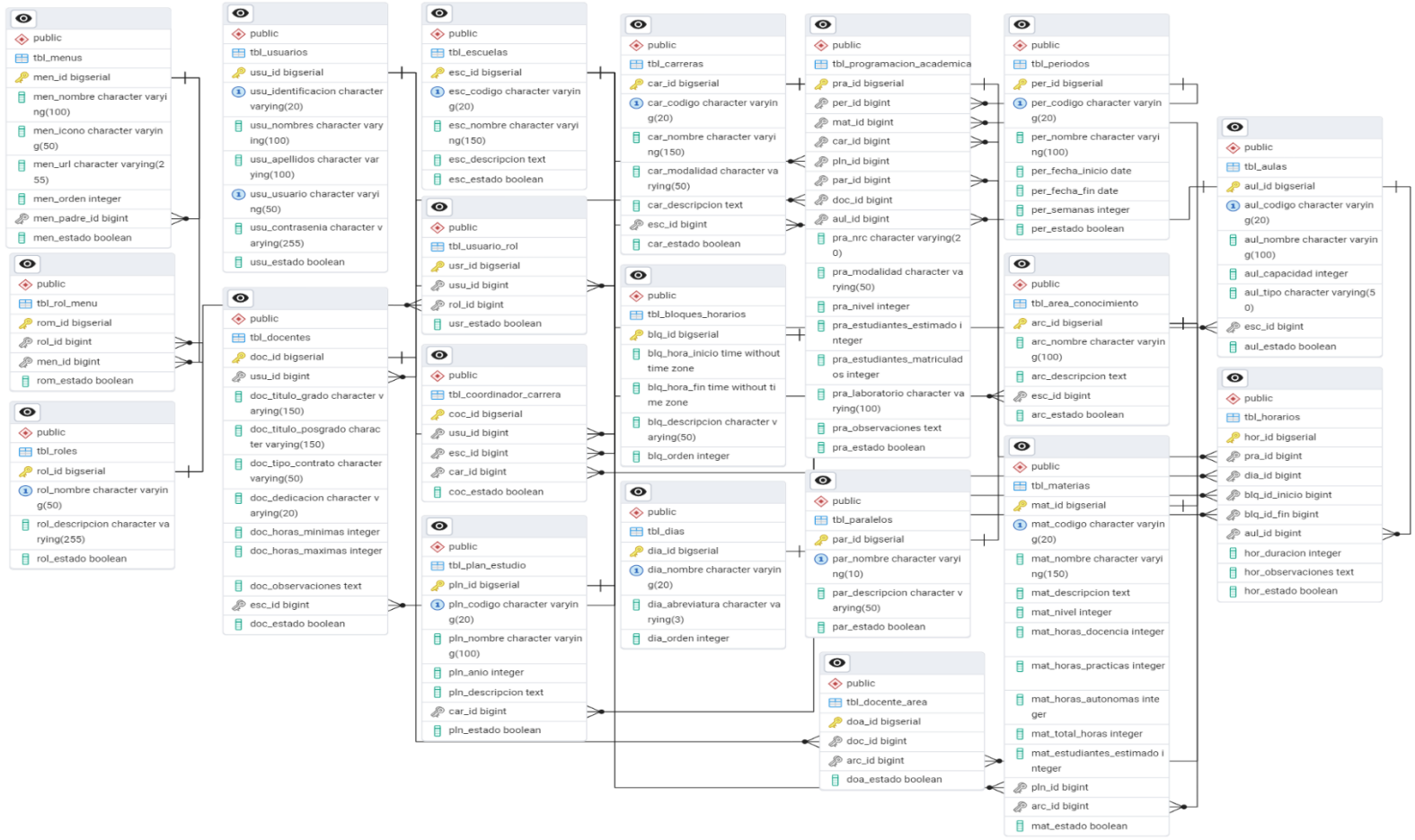
2.4.3 Modelo de base de datos

El modelo de datos sigue el paradigma relacional normalizado, centralizando información académica que antes se mantenía en Excel dispersos. La Figura 4 muestra el diagrama Entidad-Relación completo.

Las entidades principales son: Usuario (autenticación), Docente (con áreas de conocimiento), Materia (asociada a planes de estudio), Aula (recursos físicos), y Asignacion_Horario (resultado de la optimización). El modelo implementa jerarquía académica Escuela → Carrera → Plan → Materia.

La relación Docente-Area es muchos-a-muchos, permitiendo que docentes tengan múltiples áreas de experticia. Asignacion_Horario vincula docente, materia, aula, bloque y día, asociada a un Periodo_Academico para mantener histórico.

Figura 4: Diagrama entidad-relación del sistema de optimización de horarios académicos



Nota. El modelo relacional normalizado muestra las entidades principales, sus atributos, y las relaciones de cardinalidad entre ellas. Las llaves primarias se indican en negrita, las llaves foráneas con líneas de relación.

La Tabla 7 describe las entidades principales del modelo, incluyendo su propósito en el sistema, atributos clave, y relaciones con otras entidades.

Tabla 7: Descripción de entidades principales del modelo de datos

ENTIDAD	DESCRIPCIÓN Y ATRIBUTOS PRINCIPALES
Usuario	Credenciales y roles para autenticación • id (PK), email, password_hash, rol Relaciones: 1 Usuario → 0..1 Docente
Docente	Información de docentes con dedicación • id (PK), usuario_id (FK), nombre, cedula • dedicacion (TC/MT/TP), horas_min, horas_max Relaciones: N Docentes ↔ M Areas
Area_Conocimiento	Áreas de experticia (ej: Programación, BD, IA) • id (PK), nombre, descripcion Relaciones: 1 Area → N Materias
Escuela	Nivel más alto de jerarquía académica • id (PK), nombre Relaciones: 1 Escuela → N Carreras
Carrera	Programas académicos dentro de escuela • id (PK), escuela_id (FK), nombre Relaciones: 1 Carrera → N Planes
Plan_Estudio	Planes de estudio vigentes por carrera • id (PK), carrera_id (FK), codigo, nombre Relaciones: 1 Plan → N Materias
Materia	Asignaturas del plan de estudios • id (PK), plan_id (FK), area_id (FK) • nombre, nivel, horas_docencia, horas_practicas Relaciones: N Materias → 1 Plan, 1 Area
Aula	Recursos físicos (aulas, laboratorios) • id (PK), nombre, capacidad, tipo Relaciones: 1 Aula → N Asignaciones

Bloque_Horario	Bloques de tiempo disponibles • id (PK), hora_inicio, hora_fin Ejemplo: 07:00-09:00, 09:00-11:00, etc.
Dia	Días hábiles de la semana • id (PK), nombre (Lunes, Martes, etc.)
Paralelo	Secciones de un nivel (A, B, etc.) • id (PK), nombre
Periodo_Academico	Períodos académicos (semestres) • id (PK), codigo, fecha_inicio, fecha_fin Relaciones: 1 Periodo → N Asignaciones
Asignacion_Horario (PRINCIPAL)	Resultado de la optimización • id (PK), periodo_id (FK), materia_id (FK) • docente_id (FK), aula_id (FK), paralelo_id (FK) • bloque_id (FK), dia_id (FK) • aprobado (boolean), fitness (decimal) Relaciones: N Asignaciones → 1 Período, 1 Materia, 1 Docente, 1 Aula, etc.
Docente_Area	Tabla intermedia (muchos a muchos) • docente_id (FK), area_id (FK) Relaciones: N Docentes ↔ M Areas

Nota. Todas las entidades incluyen campos created_at y updated_at para auditoría. PK = Primary Key, FK = Foreign Key. La entidad Asignacion_Horario (resaltada) es la tabla principal que almacena los resultados de la optimización.

2.4.4 Diagramas de comportamiento (secuencia/actividades)

Los diagramas de comportamiento modelan la dinámica del sistema durante su ejecución. Se presentan dos diagramas: secuencia para mostrar interacción temporal entre componentes, y actividades para representar flujo de control.

Diagrama de Secuencia: Generación de Horarios

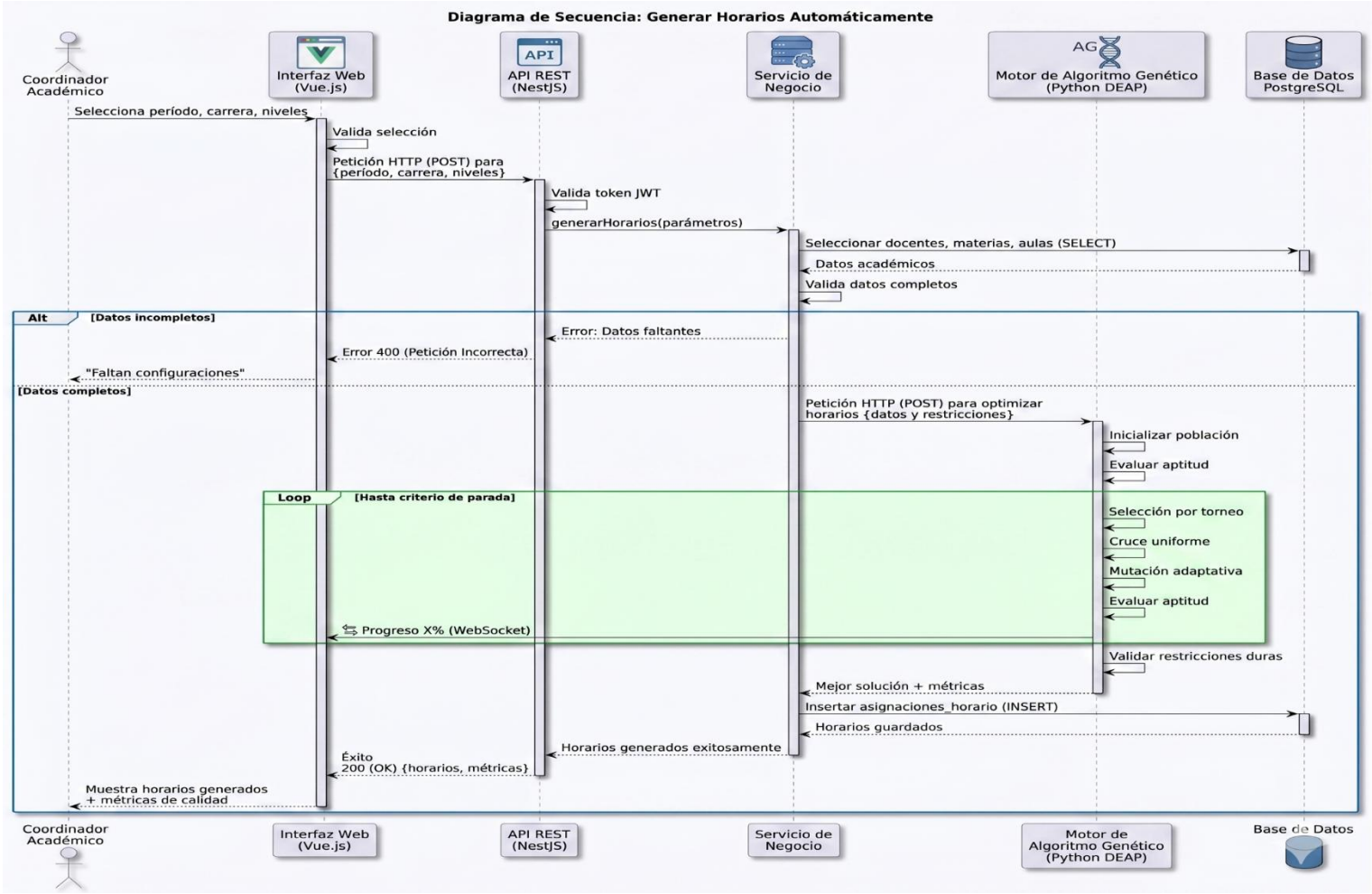
La Figura 5 muestra la interacción entre Coordinador, Interfaz Vue.js, API NestJS, Servicio de Negocio, Motor Python DEAP, y PostgreSQL. El coordinador selecciona parámetros, la interfaz valida y envía un POST a la API, que consulta datos en PostgreSQL, invoca el motor AG, recibe la solución optimizada, almacena resultados, y retorna horarios al *frontend*. El algoritmo genético se ejecuta durante 200 generaciones

fijas y retorna la mejor solución encontrada al concluir el ciclo evolutivo, junto con las métricas de ejecución.

Diagrama de Actividades: Algoritmo Genético

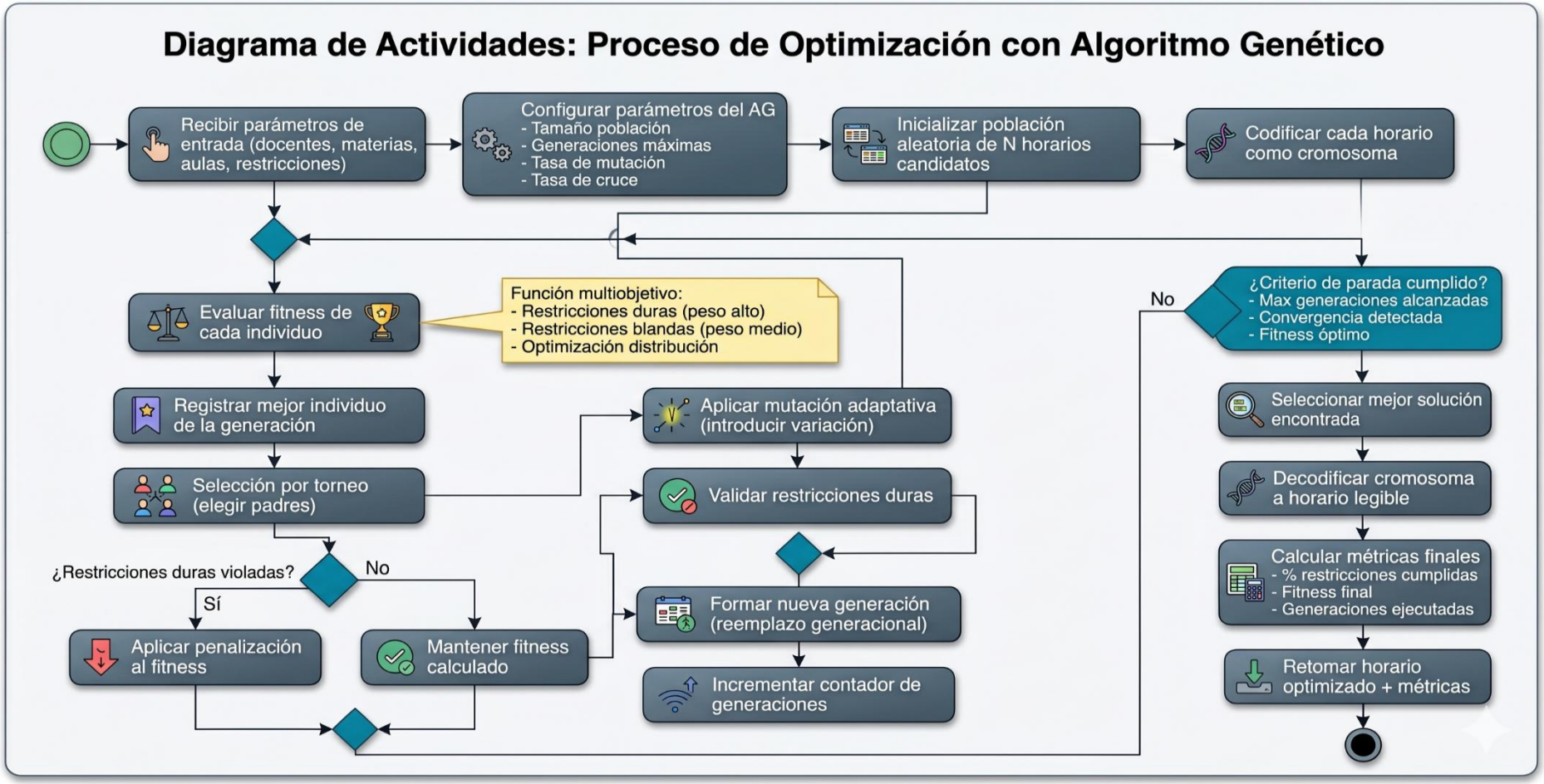
La Figura 6 detalla el flujo del AG: recibe datos, configura parámetros, inicializa población, codifica cromosomas, y ejecuta ciclo evolutivo (evaluar fitness, selección, cruce, mutación, validar restricciones) hasta criterio de parada. Retorna mejor solución con métricas.

Figura 5: Diagrama de secuencia del proceso de generación de horarios automáticamente



Nota. La secuencia muestra el flujo normal de ejecución cuando se genera horarios exitosamente, incluyendo validaciones, comunicación entre capas, y actualización de progreso en tiempo real mediante WebSocket.

Figura 6: Diagrama de Actividades del Proceso de Optimización con Algoritmo Genético



Nota. El diagrama muestra el ciclo evolutivo completo, incluyendo evaluación de fitness, operadores genéticos, validación de restricciones, y criterios de parada.

2.4.5 Diseño del algoritmo genético

2.4.5.1 Diseño del Algoritmo Genético

El algoritmo genético es el componente central que genera **sugerencias** de horarios. Cada individuo (horario candidato) se codifica como una lista de enteros, donde cada clase a programar aporta exactamente tres genes: día, bloque horario y docente. La materia y el paralelo correspondientes a cada clase quedan implícitos por su posición en la lista, por lo que un individuo con N clases tiene una longitud de 3×N genes. El aula o laboratorio no forma parte del cromosoma; su asignación se selecciona mediante un procedimiento heurístico posterior (sección 2.4.5.3). La Figura 7 muestra el flujo completo del algoritmo.

La función de aptitud [F(I)] parte de un puntaje base de 10.000 puntos y resta penalizaciones por cada restricción violada: restricciones duras (penalización alta: cruces de docentes, cruces de estudiantes, carga horaria fuera de rango, materia repetida el mismo día) y restricciones blandas (penalización baja: distribución pedagógica entendida como el reparto de las horas de una misma materia en sesiones de distintos días de la semana, en lugar de concentrarlas, para favorecer la asimilación del contenido, preferencia de horario matutino y carga diaria excesiva). El resultado se acota en un mínimo de cero.

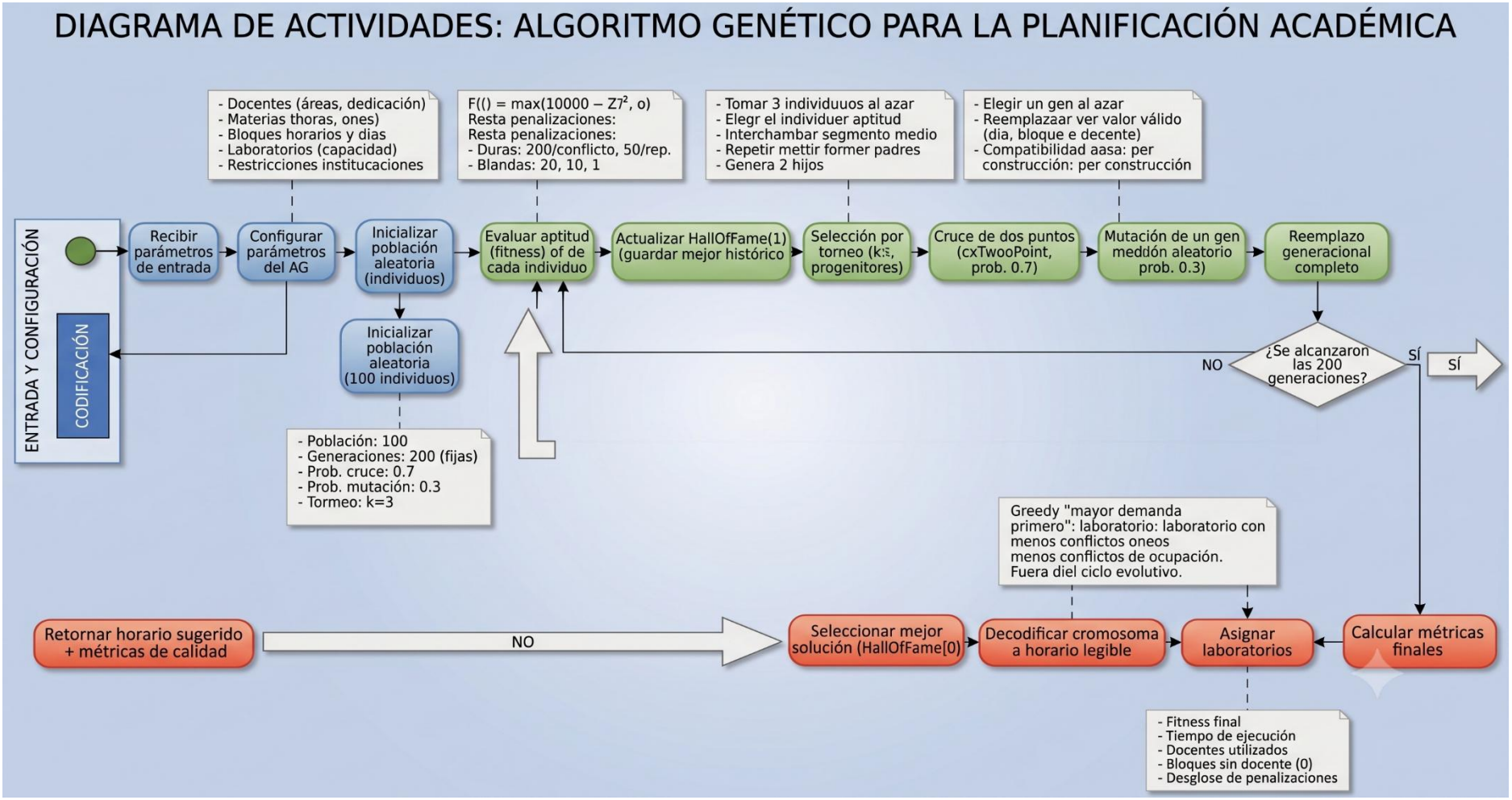
Fórmula: $F(I) = \max(10000 - \Sigma P_i(I), 0)$

El ciclo evolutivo emplea tres operadores. La selección por torneo (k=3) construye el conjunto de progenitores mediante torneos sucesivos: en cada torneo se escogen tres individuos al azar de la población y se selecciona el de mejor aptitud; el proceso se repite tantas veces como individuos requiere la nueva generación, favoreciendo a las buenas soluciones sin eliminar por completo la diversidad. El cruce de dos puntos (operador cxTwoPoint de DEAP, probabilidad 0.7) elige aleatoriamente dos posiciones de corte en el cromosoma e intercambia el segmento central entre los dos padres, generando dos hijos que combinan bloques de asignaciones de ambos. La mutación (probabilidad 0.3) reemplaza un gen aleatorio del individuo un día, un bloque o un docente por otro valor válido, manteniendo la compatibilidad de área por construcción y preservando la diversidad de la búsqueda.

El ciclo se ejecuta durante 200 generaciones fijas mediante la función eaSimple de DEAP, que orquesta en cada generación la secuencia completa de selección, cruce,

mutación, evaluación y reemplazo generacional. Dado que el reemplazo es completo la nueva población sustituye íntegramente a la anterior, se emplea un registro HallOfFame(1): una estructura externa a la población que conserva el mejor individuo encontrado a lo largo de toda la ejecución, garantizando que la mejor solución no se pierda aunque generaciones posteriores presenten menor aptitud; ese individuo es el que se retorna como sugerencia final. Cabe precisar que, a diferencia de los métodos de optimización basados en gradiente, el algoritmo genético no calcula derivadas de la función objetivo: explora el espacio de soluciones mediante muestreo estocástico, lo que lo hace aplicable a problemas combinatorios discretos como el presente. El criterio de parada de 200 generaciones fijas se estableció tras verificar empíricamente, durante la calibración, que el valor de aptitud se estabiliza con amplitud antes de alcanzar ese límite, por lo que el número de generaciones adoptado ofrece margen suficiente sin comprometer el tiempo de respuesta interactivo del sistema.

Figura 7: Diagrama de Flujo del Algoritmo Genético para Optimización de Horarios



Nota. El diagrama muestra el ciclo evolutivo completo incluyendo evaluación de fitness, selección, cruce, mutación, y criterios de parada. El c omponente en amarillo representa el ciclo iterativo principal.

2.4.5.2 Flujo principal del algoritmo genético básico (Algoritmo 1).

Parámetros calibrados empíricamente: población=100, generaciones=200, cruce=0.7, mutación inicial=0.3.

El Algoritmo 1 presenta el pseudocódigo completo del algoritmo genético implementado, incluyendo inicialización, ciclo evolutivo, operadores, y criterios de parada.

Algoritmo 1

Pseudocódigo del algoritmo genético para generación de horarios

algoritmo: GenerarHorarioOptimizado

```
entrada:
clases, docentes, dias, bloques, docentes_por_area
```

```
salida:
mejor_individuo, fitness_final
```

```
1. inicialización
  población ← CrearPoblacionAleatoria(tamaño=100)
  // cada individuo = [dia0, bloque0, doc0, dia1, bloque1, doc1, ...]
  hof ← HallOfFame(1)

2. evaluación inicial
  para cada individuo en población:
    fitness[individuo] ← EvaluarFitness(individuo)
  hof.actualizar(población)

3. ciclo evolutivo (200 generaciones fijas)
  para generacion = 1 hasta 200:
    a) selección
      descendencia ← SeleccionPorTorneo(población, k=3)
    b) cruce
      para cada par (ind1, ind2) en descendencia:
        si random() < 0.7:
          CruceDosPuntos(ind1, ind2)
    c) mutación
      para cada individuo en descendencia:
        si random() < 0.3:
          MutarGenAleatorio(individuo, docentes_por_area)
    d) evaluación
      para cada individuo modificado:
        fitness[individuo] ← EvaluarFitness(individuo)
    e) reemplazo generacional
      población ← descendencia
    f) hof.actualizar(población)

4. retorno
  retornar (hof[0], hof[0].fitness)
```

FUNCIÓN: EvaluarFitness(individuo)

```
asignaciones ← DecodificarIndividuo(individuo)
score ← 10000.0
penalizacion ← 0
```

```
// RESTRICCIONES DURAS
penalizacion += CruceDocentesActuales(asignaciones) × 200
penalizacion += CruceDocentesPrevios(asignaciones) × 200
penalizacion += CruceEstudiantes(asignaciones) × 200
penalizacion += HorasDocenteFueraRango(asignaciones, docentes) // 100/h
exceso, 10/h déficit
penalizacion += MateriaDiasRepetidos(asignaciones) × 50
penalizacion += ConcentracionDocente(asignaciones) × 80

// RESTRICCIONES BLANDAS
penalizacion += DistribucionHoras(asignaciones)
penalizacion += PreferenciaHorarioMatutino(asignaciones) × 1
penalizacion += CargaDiariaExcesiva(asignaciones) × 10

score ← score - penalizacion
retornar max(score, 0)
```

2.4.5.3 Asignación Heurística de Laboratorios

Una vez que el algoritmo genético determina el día, bloque horario y docente de cada clase, un procedimiento complementario asigna los laboratorios disponibles a las materias que los requieren. Este procedimiento no forma parte del ciclo evolutivo: se ejecuta una sola vez sobre la mejor solución encontrada.

La asignación emplea una estrategia voraz (greedy) un método que en cada paso toma la mejor decisión disponible en ese momento, sin reconsiderar las decisiones ya tomadas del tipo "mayor demanda primero". El procedimiento opera en los siguientes pasos:

1. Los niveles académicos se ordenan de forma descendente según su cantidad estimada de estudiantes, de modo que los de mayor demanda se atienden primero.
2. Para cada nivel, se recorren los laboratorios disponibles ordenados por capacidad, de mayor a menor.
3. Para cada laboratorio candidato se cuenta el número de conflictos de ocupación que generaría, es decir, cuántas veces el laboratorio ya estaría ocupado en el mismo día y bloque por un horario previamente aprobado o por otro paralelo procesado antes en la misma ejecución.
4. Se asigna el primer laboratorio con cero conflictos; si ninguno está totalmente libre, se asigna el de menor número de conflictos.
5. Tras cada asignación, la ocupación del laboratorio se actualiza antes de procesar el siguiente nivel.

Por su naturaleza voraz, este procedimiento prioriza la rapidez y la simplicidad sobre la optimalidad: entrega una asignación de buena calidad en una sola pasada, pero no garantiza la distribución globalmente óptima de laboratorios, lo cual es aceptable dado que en las instancias evaluadas la disponibilidad de laboratorios sin conflicto fue suficiente. El procedimiento es además determinista: ante los mismos horarios de entrada produce siempre la misma asignación, a diferencia del algoritmo genético, que es de naturaleza estocástica.

Esta heurística materializa el enfoque híbrido planteado en el Capítulo I: mientras el algoritmo genético explora, mediante operadores evolutivos, el vasto espacio de combinaciones de día-bloque-docente, la asignación de laboratorios un subproblema de menor complejidad una vez fijado el horario se resuelve con una regla determinista basada en conocimiento del dominio. Así se evita incrementar la dimensionalidad del cromosoma y el costo computacional del ciclo evolutivo.

2.4.5.4 Criterio de calidad y alcance de la generación

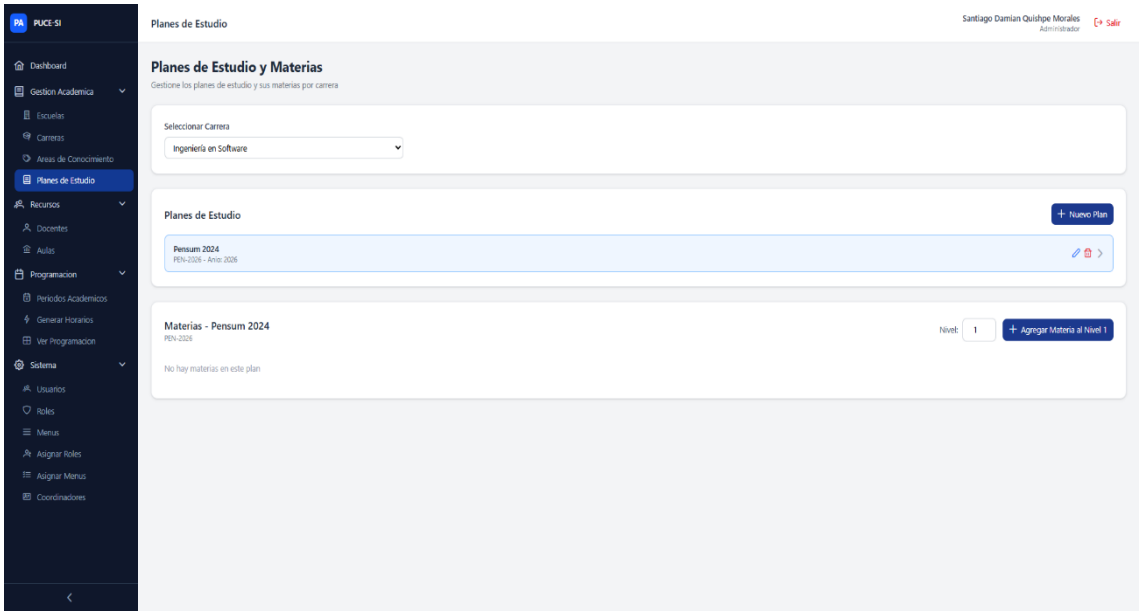
La calidad de cada horario no se evalúa mediante un criterio externo, sino a través de la propia función de aptitud, que traduce a un valor numérico el grado en que una solución cumple lo que la institución considera un buen horario. El peso asignado a cada restricción constituye, por tanto, el criterio de calidad del sistema: las restricciones que hacen inviable un horario cruce de docentes, cruce de estudiantes penalizan 200 puntos; el incumplimiento de la carga horaria normativa penaliza 100 puntos por hora de exceso y 10 por hora de déficit; la concentración de materias en un mismo docente, 80 puntos; la repetición de una materia el mismo día, 50 puntos; y las preferencias pedagógicas, como la distribución de horas o el horario matutino, valores progresivamente menores. Estos pesos se definieron a partir de tres fuentes: la normativa institucional del Oficio Nro. 902-Dir.DocyEst, el levantamiento de requisitos con el coordinador académico, y la distinción entre restricciones duras y blandas establecida en la literatura. En consecuencia, cuando el algoritmo maximiza el fitness, maximiza el cumplimiento ponderado de estos criterios; un valor cercano al máximo teórico de 10.000 puntos indica un horario que satisface casi en su totalidad las condiciones definidas como deseables, sin que ello implique la optimalidad global, que no puede garantizarse en un problema de esta naturaleza.

Cabe precisar que el algoritmo genera los horarios por nivel y paralelo de forma independiente: cada ejecución resuelve una instancia acotada correspondiente a un único nivel y paralelo, en lugar de programar simultáneamente toda la carrera o la escuela. Para preservar la coherencia entre instancias, cada nueva ejecución recibe como entrada los horarios previamente generados y confirmados, de modo que un docente ya asignado en un paralelo no entre en conflicto al programar los siguientes. Esta descomposición del problema en subproblemas de menor dimensión es una decisión de diseño deliberada: mantiene cada instancia en una escala en la que el algoritmo converge en pocos segundos, evitando enfrentar el problema completo como una única búsqueda de dimensionalidad muy elevada.

2.4.6 Diseño de interfaz

El diseño de la interfaz persigue que la aplicación cumpla con el principio de usabilidad. Se implementa una SPA *(Single Page Application*) con navegación fluida. La Figura 8 muestra el *Dashboard* principal del sistema.

Figura 8: Dashboard general de la programación académica.



Nota. Pantalla principal que muestra el menú lateral de navegación organizado por módulos funcionales y el área de contenido donde se despliegan las diferentes funcionalidades del sistema.

La interfaz tiene una distribución (*layout*) de dos columnas: menú lateral de navegación (15% de ancho, fondo #1E293B gris oscuro) y área de contenido principal (85% de ancho, fondo blanco). El menú se organiza en cuatro categorías: Gestión

Académica (Escuelas, Carreras, Áreas, Planes), Recursos (Docentes, Aulas), Programación (Períodos, Generar Horarios, Ver Programación), y Sistema (Usuarios, Roles, Menús).

El ítem activo se resalta en un fondo azul (#3B82F6). En la esquina superior derecha se muestra información del usuario (nombre, rol) y botón "Salir". El diseño es responsivo usando Tailwind CSS y cumple pautas de accesibilidad WCAG 2.1 nivel AA con contraste de colores adecuado (ratio > 4.5:1).

2.4.7 Herramientas de desarrollo

El desarrollo del sistema requirió herramientas especializadas para cada fase: diseño, codificación, pruebas, control de versiones, y despliegue. La Tabla 8 presenta las herramientas utilizadas organizadas por categoría.

Tabla 8: Herramientas de desarrollo utilizadas

CATEGORÍA	HERRAMIENTA	PROPÓSITO / USO
IDE / Editor	**Visual Studio Code**	Editor de código principal con extensiones para TypeScript, Vue.js, Python, Docker, Git
Control de Versiones	**Git**	Sistema de control de versiones distribuido (estrategia GitFlow)
	GitHub	Repositorio remoto, colaboración y gestión de issues/pull requests
Contenedorización	**Docker**	Contenedorización de servicios (NestJS, PostgreSQL, Python)
	Docker Compose	Orquestación de contenedores multi-servicio
Gestión de Dependencias	**npm**	Gestor de paquetes para Node.js (backend y frontend)
	pip	Gestor de paquetes para Python (motor de optimización)
Base de Datos	**pgAdmin**	Administración gráfica de PostgreSQL, generación ERD
API Testing	**Postman**	Diseño, prueba y documentación de API REST
Testing	**Jest**	Framework de pruebas unitarias para NestJS (backend)
	Vitest	Framework de pruebas para Vue.js (frontend)
	Pytest	Framework de pruebas para Python (motor AG)
Calidad de Código	**ESLint**	Linter para JavaScript/TypeScript (análisis estático)
	Prettier	Formateador automático de código
	Pylint	Linter para Python
Documentación	**Swagger/OpenAPI**	Documentación automática de API REST (integrado con NestJS)
Diagramación	**PlantUML**	Generación de diagramas UML (casos de uso, secuencia, etc.)
	draw.io	Diagramas de arquitectura y flujos de proceso

Nota. Herramientas organizadas por categoría funcional. Las versiones indicadas son mínimas requeridas para compatibilidad con el stack tecnológico del proyecto.

Visual Studio Code fue el IDE principal con extensiones para TypeScript, Vue.js, Python, Docker, y Git. Git gestionó el control de versiones a través de GitHub (main, develop, feature/*). Docker Desktop permitió la contenedorización y orquestación de componentes. Postman facilitó el diseño y la prueba de la API REST. pgAdmin 4 administró PostgreSQL. Jest, Vitest y Pytest realizaron pruebas unitarias.

2.5 Casos de prueba

La validación del sistema se realizó mediante dos estrategias complementarias alineadas con el objetivo específico de validar mediante pruebas funcionales y métricas cuantitativas de calidad: pruebas funcionales que verifican el correcto funcionamiento del sistema web ante diferentes escenarios de uso, y métricas cuantitativas que evalúan el desempeño del algoritmo genético en la generación de sugerencias de horarios académicos.

2.5.1 Pruebas Funcionales del Sistema Web

Las pruebas funcionales verifican que el sistema cumple con los requisitos especificados en las historias de usuario, validando escenarios de éxito y de error en las funcionalidades principales del sistema. Estas pruebas se derivaron de los criterios de aceptación definidos en las historias de usuario (Tabla 2): cada criterio verificable dio origen a uno o más escenarios, agrupados en seis pruebas de aceptación según el flujo funcional que verifican cada tabla indica las historias que cubre totalizando veintidós escenarios que abarcan catorce de las quince historias. La HU-12 (ajustar horarios manualmente) se validó mediante verificación directa con el coordinador académico sobre la grilla de horarios generados.

Tabla 9: Prueba de aceptación: inicio de sesión

PRUEBA DE ACEPTACIÓN N°1		
Nombre: Inicio de Sesión (Login)		
Descripción: Verificar que el sistema permita el acceso únicamente a usuarios registrados con credenciales válidas.		
Usuario	**Contraseña**	**Resultado Esperado**
Usuario correcto	Contraseña correcta	El sistema redirige al panel correspondiente según el rol del usuario
Usuario correcto	Contraseña incorrecta	El sistema indica que las credenciales son inválidas

Usuario correcto	Vacío	El sistema solicita completar el campo contraseña
Vacío	Contraseña correcta	El sistema solicita completar el campo usuario
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-1 y HU-14.		

Nota. Prueba de la funcionalidad de autenticación y control de acceso por roles.

Tabla 10: Prueba de aceptación: gestionar docentes

PRUEBA DE ACEPTACIÓN N°2		
Nombre: Gestionar Docentes		
Descripción: Verificar que el coordinador académico pueda registrar y administrar docentes con sus áreas de conocimiento y categoría de dedicación.		
Usuario	**Datos de Entrada**	**Resultado Esperado**
Coordinador	Datos completos + áreas seleccionadas	Docente registrado exitosamente y el sistema abre automáticamente el modal de asignación de áreas de conocimiento
Coordinador	Nombre vacío	El sistema solicita completar el campo nombre
Coordinador	Sin seleccionar áreas de conocimiento	El sistema solicita asignar al menos un área
Coordinador	Cédula duplicada	El sistema indica que ya existe un docente con esa cédula
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-3 y HU-4.		

Nota. Prueba de la gestión centralizada de docentes que reemplaza el uso de hojas Excel.

Tabla 11: Prueba de aceptación: gestionar materias

PRUEBA DE ACEPTACIÓN N°3		
Nombre: Gestionar Materias		
Descripción: Verificar que el coordinador pueda registrar materias del plan de estudios asociándolas a áreas de conocimiento.		
Usuario	**Datos de Entrada**	**Resultado Esperado**
Coordinador	Datos completos + área seleccionada	Materia registrada y asociada al plan de estudio correctamente
Coordinador	Sin seleccionar área de conocimiento	El sistema solicita seleccionar un área
Coordinador	Horas docencia vacías	El sistema solicita ingresar las horas correspondientes
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-2 y HU-3.		

Nota. Prueba de la gestión del catálogo de materias vinculadas a áreas de conocimiento para la asignación docente.

Tabla 12: Prueba de aceptación: generar horarios

PRUEBA DE ACEPTACIÓN N°4		
Nombre: Generar Horarios (Algoritmo Genético)		
Descripción: Verificar que el sistema ejecute el algoritmo genético y genere sugerencias de horarios académicos respetando las restricciones institucionales.		
Usuario	**Condición del Sistema**	**Resultado Esperado**
Coordinador	Datos completos: docentes, materias, período activo	El sistema genera sugerencia de horario y muestra métricas de calidad
Coordinador	Sin docentes registrados con áreas	El sistema indica que faltan recursos para generar horarios
Coordinador	Sin materias configuradas	El sistema indica que no hay materias para programar
Coordinador	Período académico inactivo	El sistema solicita seleccionar un período activo
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-5, HU-6, HU-7, HU-8 y HU-9.		

Nota. Prueba de la funcionalidad principal del sistema: generación de sugerencias de horarios mediante el algoritmo genético.

Tabla 13: Prueba de aceptación: ver programación

PRUEBA DE ACEPTACIÓN N°5		
Nombre: Ver Programación		
Descripción: Verificar que los horarios generados puedan visualizarse correctamente desde diferentes perspectivas y exportarse.		
Usuario	**Acción**	**Resultado Esperado**
Coordinador	Seleccionar vista por docente	El sistema muestra grilla semanal del docente seleccionado
Coordinador	Vista por nivel	El sistema muestra horario completo del nivel seleccionado
Coordinador	Exportar a Excel	El sistema descarga el archivo con el horario completo
Coordinador	Sin período seleccionado	El sistema solicita seleccionar un período académico
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-10, HU-11, HU-13 y HU-15.		

Nota. Prueba de visualización y exportación de los horarios académicos generados por el algoritmo.

Tabla 14: Prueba de aceptación: gestionar usuarios

PRUEBA DE ACEPTACIÓN N°6		
Nombre: Gestionar Usuarios		
Descripción: Verificar que el administrador pueda crear y gestionar usuarios del sistema con sus respectivos roles.		
Usuario	**Datos de Entrada**	**Resultado Esperado**
Administrador	Datos completos + rol asignado	Usuario creado y habilitado para acceder al sistema
Administrador	Usuario o cedula ya registrado en el sistema.	El sistema indica que ya existe un usuario con ese email
Administrador	Contraseñas no coinciden	El sistema indica que las contraseñas no son iguales
Observaciones: Esta prueba cubre los criterios de aceptación de las historias de usuario HU-1 y HU-14.		

Nota. Prueba de la gestión de usuarios y control de acceso por roles del sistema.

2.5.2 Métricas Cuantitativas de Calidad del Algoritmo Genético

Las métricas cuantitativas evalúan el desempeño del algoritmo genético en la generación de sugerencias de horarios. Se establecen considerando que los algoritmos genéticos son métodos heurísticos que exploran el espacio de soluciones sin garantizar óptimos globales, con tasas de satisfacción de restricciones entre el 80% y el 95% reportadas en la literatura para problemas de planificación académica universitaria (Ceschia et al., 2023). De acuerdo con la función de aptitud definida en la sección 2.4.5, $F(I) = \max(10000 - \Sigma P_i, 0)$, el fitness parte de un máximo teórico de 10.000 puntos y disminuye con cada penalización, por lo que valores mayores indican soluciones de mejor calidad.

Tabla 15: Métricas cuantitativas de calidad del algoritmo genético

MÉTRICA	DESCRIPCIÓN	UNIDAD DE MEDIDA
Valor de fitness inicial	Aptitud del mejor individuo de la población inicial aleatoria, antes de la evolución; refleja la calidad del punto de partida.	Puntos de fitness, escala 0–10.000 (mayor es mejor)
Valor de fitness final	Aptitud de la mejor solución encontrada al concluir las 200 generaciones; refleja la mejora lograda por el proceso evolutivo.	Puntos de fitness, escala 0–10.000 (mayor es mejor)
Mejora relativa del fitness	Incremento porcentual entre el fitness inicial y el fitness final; indica la efectividad del proceso evolutivo.	Porcentaje de mejora (%)

Tiempo de ejecución	Duración total del proceso de generación, desde la inicialización de la población hasta la finalización de las 200 generaciones.	Segundos
Docentes utilizados	Número de docentes distintos asignados en la sugerencia generada respecto al total de docentes disponibles con áreas compatibles. Indicador descriptivo de utilización de recursos; no constituye una restricción del modelo.	Número entero
Bloques sin docente asignado	Verificación del cumplimiento del requisito duro de que toda clase cuente con un docente compatible asignado. Su único valor aceptable es 0.	Número entero (valor obligatorio: 0)
Desglose de penalizaciones	Distribución de la penalización total por tipo de restricción: cruces de docentes (actuales y con horarios previos), cruces de estudiantes, carga horaria fuera de rango, materia repetida el mismo día, y restricciones blandas (distribución pedagógica, preferencia de horario matutino y carga diaria excesiva).	Puntos de penalización por categoría

Nota. Métricas definidas según el plan de titulación y la función de aptitud de la sección 2.4.5. La compatibilidad entre el área de conocimiento del docente y la materia se garantiza por construcción del cromosoma, y la asignación de aulas se resuelve mediante la heurística complementaria (sección 2.4.5.3), por lo que ninguna de las dos genera penalizaciones. Los valores se obtienen empíricamente durante la fase de validación con el coordinador académico de la carrera de Ingeniería en Software de la Escuela de Hábitat, Infraestructura y Creatividad de PUCE-SI.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

El presente capítulo expone los resultados obtenidos tras la implementación y validación del sistema web inteligente para la optimización de la programación académica de la Escuela de Hábitat, Infraestructura y Creatividad de la PUCE Sede Ibarra. La evaluación del sistema se estructura en tres ejes complementarios: en primer lugar, se presentan las interfaces del sistema web implementado y los resultados de las pruebas funcionales ejecutadas sobre cada módulo, verificando el cumplimiento de los criterios de aceptación definidos en la fase de desarrollo; en segundo lugar, se exponen las métricas cuantitativas obtenidas tras la ejecución del algoritmo genético, incluyendo el valor de la función de aptitud, el tiempo de ejecución, el cumplimiento de restricciones duras, la reducción de conflictos y la estabilidad del algoritmo; finalmente, se analizan las métricas cualitativas derivadas de la evaluación con el coordinador académico en cuanto a coherencia pedagógica y usabilidad del sistema.

3.1 Interfaces del sistema web implementado

El desarrollo del sistema web dio lugar a la implementación de cinco módulos funcionales, organizados según los tres roles de usuario definidos en el sistema: Administrador, Coordinador Académico y Docente. A continuación, se presentan las interfaces implementadas para cada módulo: autenticación y control de acceso, gestión académica, gestión de recursos, programación académica y administración del sistema, describiendo su funcionalidad y el flujo de interacción correspondiente.

3.1.1 Autenticación y control de acceso

El módulo de autenticación constituye el punto de entrada al sistema y gestiona el acceso diferenciado de usuarios mediante un mecanismo de control de roles. Al ingresar credenciales válidas, el sistema autentica al usuario y evalúa automáticamente la cantidad de roles asignados: si el usuario posee un único rol, el sistema lo redirige directamente a su vista correspondiente; si posee más de un rol, se despliega una pantalla intermedia de selección que permite al usuario elegir el contexto de trabajo con el que desea operar en la sesión actual. El sistema contempla tres roles diferenciados con niveles de acceso distintos, cuyas características se detallan en la Tabla 16.

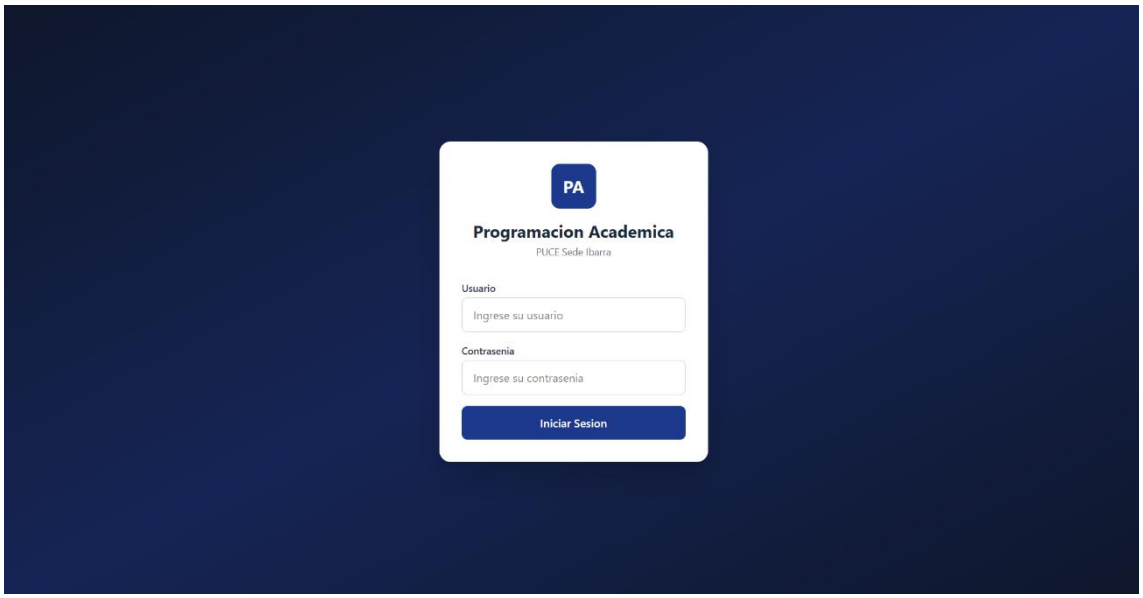
Tabla 16: Roles de usuario y niveles de acceso del sistema

Rol	Descripción	Módulos Accesibles
Administrador	Acceso total al sistema. Gestiona la configuración global de la plataforma.	Gestión Académica, Recursos, Programación, Sistema (Usuarios, Roles, Menús, Asignar Roles, Asignar Menús, Coordinadores)
Coordinador Académico	Responsable de la programación académica. Gestiona la distribución y generación de horarios.	Gestión Académica, Recursos (Docentes, Aulas), Programación (Periodos, Generar Horarios, Ver Programación)
Docente	Usuario final de consulta. Revisa únicamente la programación que le ha sido asignada y puede subir material.	Vista de horario personal asignado, carga de libros/sílabos

Nota. El sistema implementa tres niveles de acceso diferenciados según el rol institucional del usuario, determinando los módulos disponibles en cada sesión de trabajo.

La Figura 9 muestra la pantalla de inicio de sesión del sistema, la cual solicita al usuario su nombre de usuario y contraseña institucional. El diseño es minimalista e identifica al sistema mediante el nombre "Programación Académica – PUCE Sede Ibarra", garantizando coherencia con la identidad institucional.

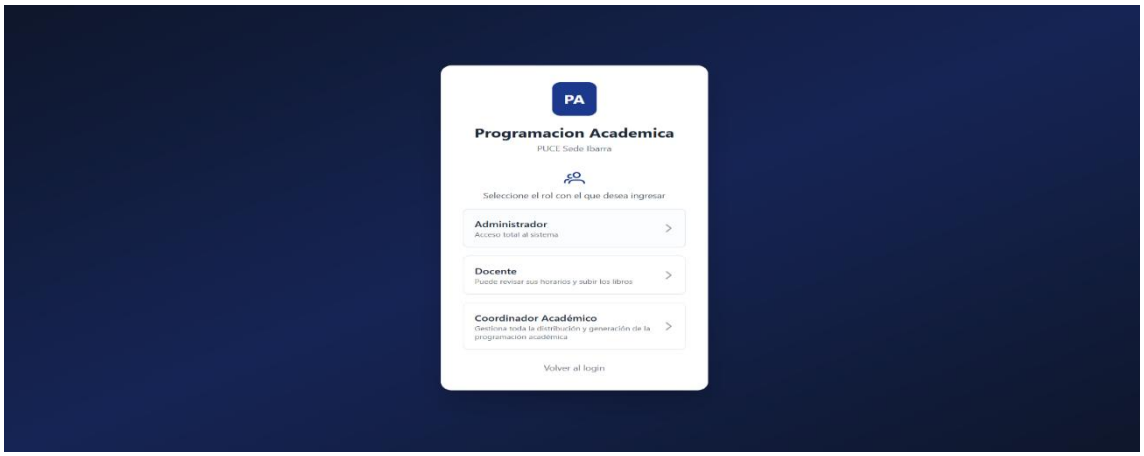
Figura 9: Pantalla de inicio de sesión del sistema de programación académica PUCE-SI



Nota. La pantalla de inicio de sesión solicita credenciales institucionales y redirige al usuario según el rol o roles asignados en el sistema.

Cuando el usuario autenticado posee más de un rol asignado, el sistema despliega la pantalla de selección de rol que se presenta en la Figura 10. Esta pantalla muestra las opciones disponibles con una breve descripción de las atribuciones de cada rol, permitiendo al usuario elegir el contexto de trabajo antes de acceder al sistema. En caso de que el usuario sea únicamente Docente, esta pantalla se omite y el acceso es directo a la vista de horarios.

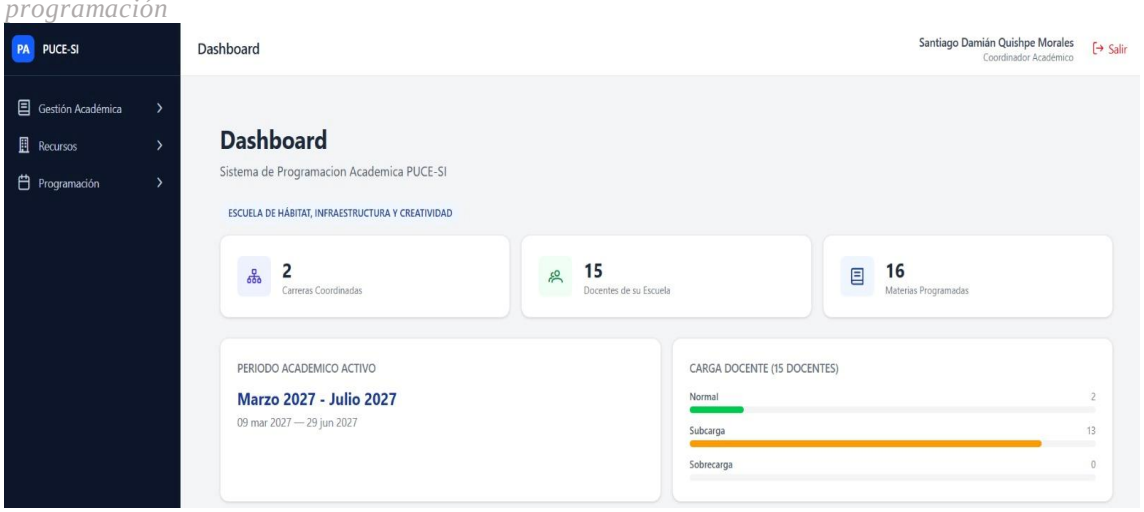
Figura 10: Pantalla de selección de rol para usuarios con múltiples roles asignados



Nota. La selección de rol permite a un mismo usuario operar bajo distintos contextos de acceso sin requerir cuentas independientes para cada función institucional.

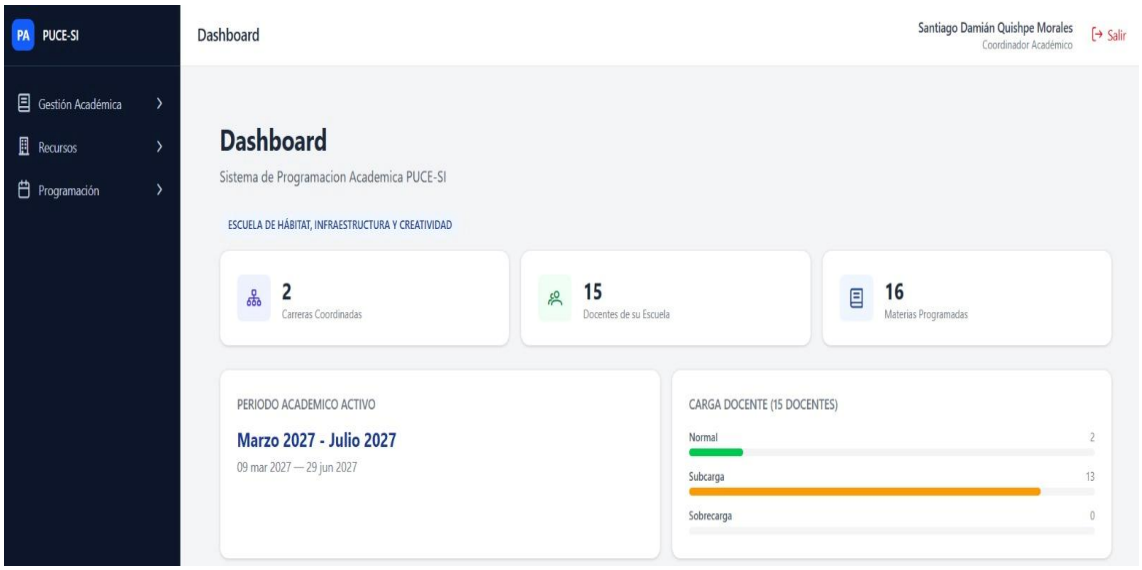
Una vez seleccionado el rol, el sistema redirige al usuario a su dashboard correspondiente. La Figura 11 presenta el panel principal del rol Coordinador Académico, que concentra los módulos de Gestión Académica, Recursos y Programación, los cuales son los de mayor relevancia operativa para la generación y administración de horarios. La Figura 12 corresponde al panel del Administrador, que incluye adicionalmente la sección Sistema, desde la cual se gestiona la configuración global de usuarios, roles y menús de la plataforma.

Figura 11: Dashboard del coordinador académico con módulos de gestión académica, recursos y programación



Nota. El panel del Coordinador Académico organiza los módulos operativos en tres secciones: Gestión Académica, Recursos y Programación, concentrando las funcionalidades necesarias para la generación de horarios.

Figura 12: Dashboard del administrador con módulo adicional de configuración del sistema



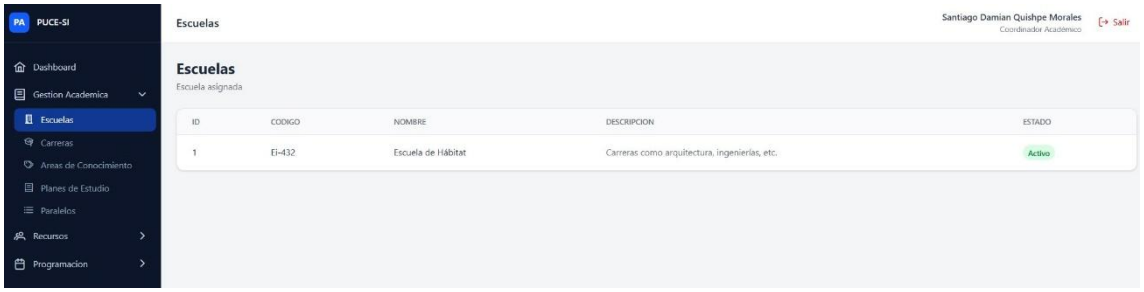
Nota. El panel del Administrador extiende las funcionalidades del Coordinador Académico con la sección Sistema, que permite gestionar usuarios, roles, menús y la asignación de coordinadores a escuelas.

3.1.2 Gestión Académica

El módulo de Gestión Académica centraliza la administración de los recursos académicos institucionales que constituyen la base de datos necesaria para la generación de horarios. Este módulo agrupa cinco submódulos interrelacionados: Escuelas, Carreras, Áreas de Conocimiento, Planes de Estudio y Paralelos. Cada submódulo implementa operaciones de creación, consulta, edición y eliminación de registros, con control de estado activo/inactivo que permite desactivar temporalmente un recurso sin eliminarlo del sistema.

La Figura 13 presenta el módulo de Escuelas, en el que el sistema restringe la visualización únicamente a la escuela asignada al coordinador autenticado. En la validación realizada, el Coordinador Académico visualizó correctamente la Escuela de Hábitat, Infraestructura y Creatividad con código EI-432, estado Activo y su descripción institucional. Este mecanismo de restricción garantiza que cada coordinador opere exclusivamente sobre los recursos de su unidad académica.

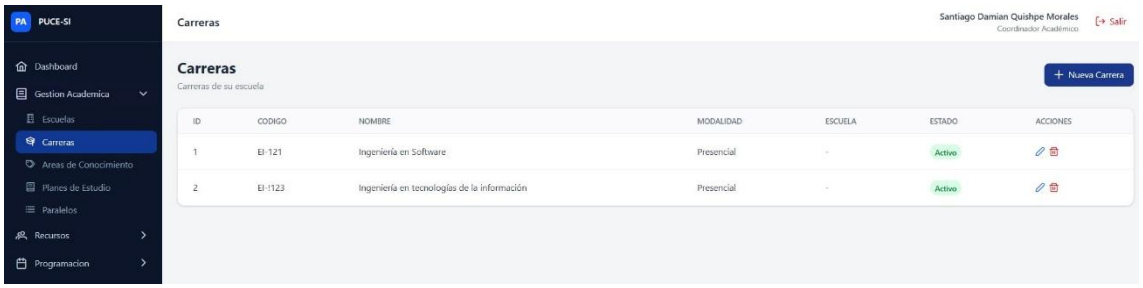
Figura 13: Módulo de Escuelas: vista del listado de escuelas asignadas al coordinador académico



Nota. El módulo muestra únicamente la escuela asignada al coordinador, en este caso la Escuela de Hábitat (EI-432), garantizando que cada coordinador gestione exclusivamente su unidad académica.

La Figura 14 muestra el módulo de Carreras, donde se listan las carreras vinculadas a la escuela del coordinador. En la validación se verificó el correcto registro de las dos carreras presenciales activas de la Escuela de Hábitat: Ingeniería en Software (EI-121) e Ingeniería en Tecnologías de la Información (EI-1123). El módulo permite además la creación de nuevas carreras mediante el botón Nueva Carrera.

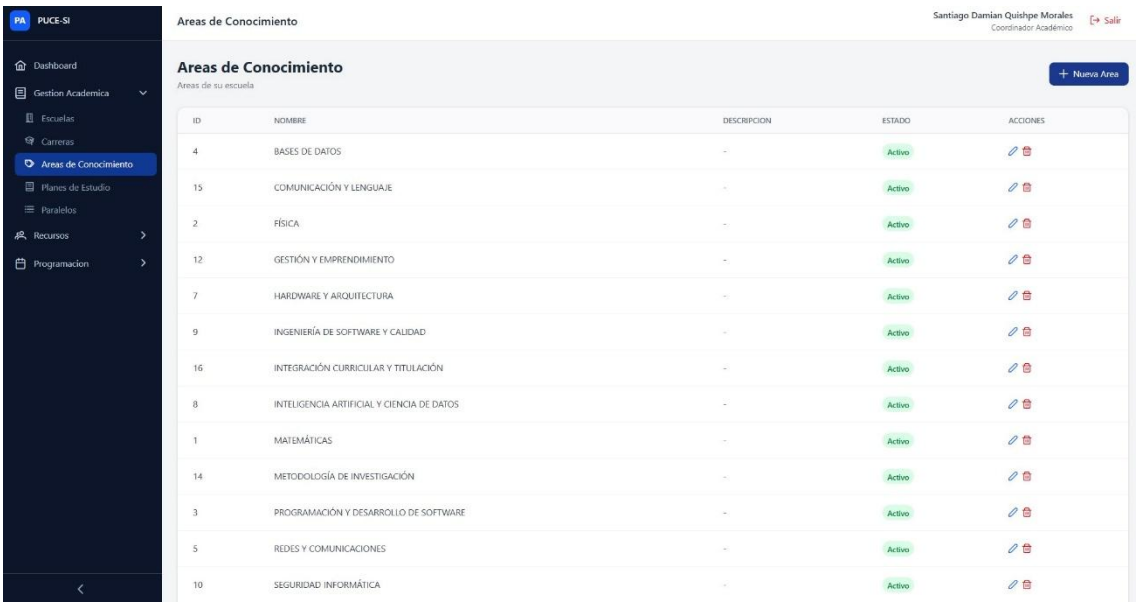
Figura 14: Módulo de carreras: listado de carreras vinculadas a la escuela



Nota. El sistema registra las carreras de la escuela con su código, modalidad y estado. La Escuela de Hábitat cuenta con dos carreras presenciales activas: Ingeniería en Software e Ingeniería en Tecnologías de la Información.

La Figura 15 presenta el módulo de Áreas de Conocimiento, que registra las disciplinas académicas bajo las cuales se clasifican las materias del pensum. En la validación se verificaron 14 áreas activas, entre ellas: Bases de Datos, Física, Ingeniería de Software y Calidad, Inteligencia Artificial y Ciencia de Datos, Matemáticas, Programación y Desarrollo de Software, y Redes y Comunicaciones, entre otras. Esta clasificación es utilizada por el algoritmo genético para evaluar la afinidad entre el perfil del docente y las materias asignadas.

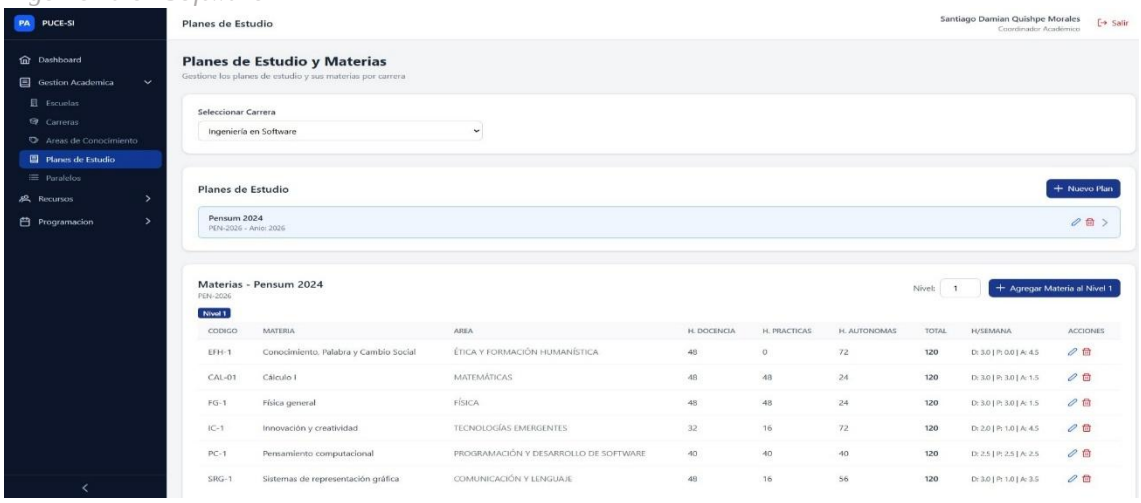
Figura 15: Módulo de áreas de conocimiento: listado de áreas registradas para la escuela



Nota. El sistema registra 14 áreas de conocimiento disciplinar activas, las cuales se utilizan para clasificar las materias del pensum y facilitar la asignación de docentes según su perfil académico.

La Figura 16 muestra el módulo de Planes de Estudio, que permite gestionar el pensum académico de cada carrera organizando las materias por nivel. En la validación se verificó el Pensum 2024 de la carrera Ingeniería en Software (código PEN-2026), donde se observan las materias del Nivel 1 con su distribución horaria completa: horas de docencia, horas de prácticas, horas autónomas, total de horas y carga semanal. Esta información es consumida directamente por el motor de optimización para determinar los bloques horarios requeridos por cada materia.

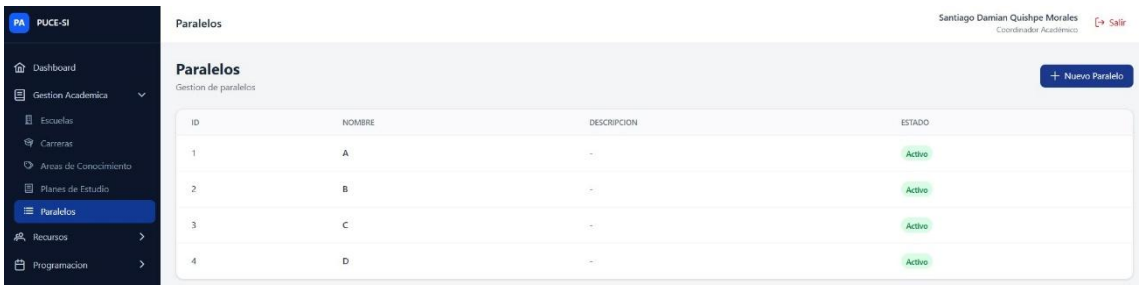
Figura 16: Módulo de planes de estudio: pensum activo con materias organizadas por nivel para Ingeniería en Software



Nota. El módulo gestiona el pensum académico por carrera, mostrando las materias de cada nivel con su código, área de conocimiento, horas de docencia, prácticas, autónomas y carga semanal.

La Figura 17 presenta el módulo de Paralelos, que registra los grupos de estudiantes disponibles para la distribución de materias. En la validación se verificaron cuatro paralelos activos: A, B, C y D, los cuales son utilizados por el algoritmo genético como variables de asignación al generar la programación académica. El módulo permite la creación de nuevos paralelos según las necesidades de cada periodo académico.

Figura 17: Módulo de paralelos: listado de paralelos disponibles para la asignación de horarios



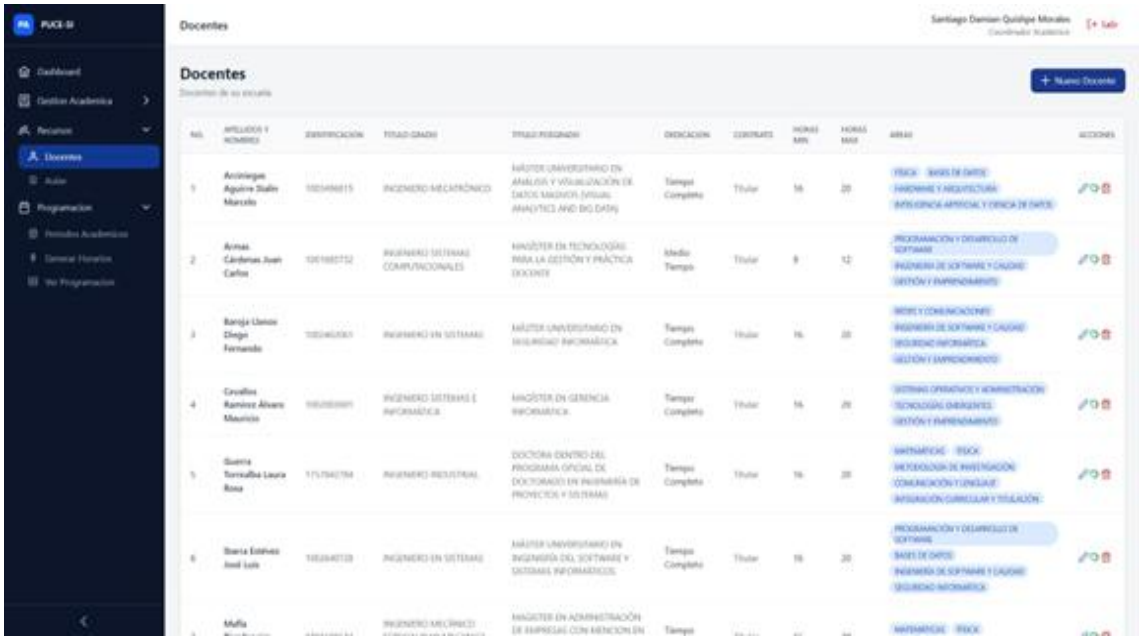
Nota. El sistema registra paralelos activos.

3.1.3 Gestión de recursos

El módulo de Gestión de Recursos administra los recursos físicos y humanos que intervienen directamente en la generación de horarios: los docentes y las aulas. Ambos submódulos proveen al motor de optimización la información necesaria para construir asignaciones válidas: los docentes aportan su perfil académico, tipo de dedicación y restricciones horarias normativas; las aulas aportan su disponibilidad, tipo y capacidad. La correcta carga de estos datos es determinante para la calidad de los horarios generados por el algoritmo genético.

La Figura 18 presenta el módulo de Docentes, donde se registran los datos académicos y normativos de cada profesor de la escuela. En la validación se verificaron los registros de los docentes titulares de la Escuela de Hábitat, entre ellos: Arciniegas Aguirre Stalin Marcelo (Tiempo Completo, 16-20h, áreas: Física, Bases de Datos, Hardware y Arquitectura, Inteligencia Artificial y Ciencia de Datos), Armas Cárdenas Juan Carlos (Medio Tiempo, 9-12h, áreas: Programación y Desarrollo de Software, Ingeniería de Software y Calidad, Gestión y Emprendimiento), y Baroja Llanos Diego Fernando (Tiempo Completo, 16-20h, áreas: Redes y Comunicaciones, Ingeniería de Software y Calidad, Seguridad Informática, Gestión y Emprendimiento), entre otros. Las columnas Horas Min y Horas Max reflejan directamente las restricciones del Oficio Nro. 902-Dir.DocyEst., que establece 16-20 horas para docentes titulares a tiempo completo y rangos diferenciados para medio tiempo, restricciones que el algoritmo genético penaliza cuando son incumplidas.

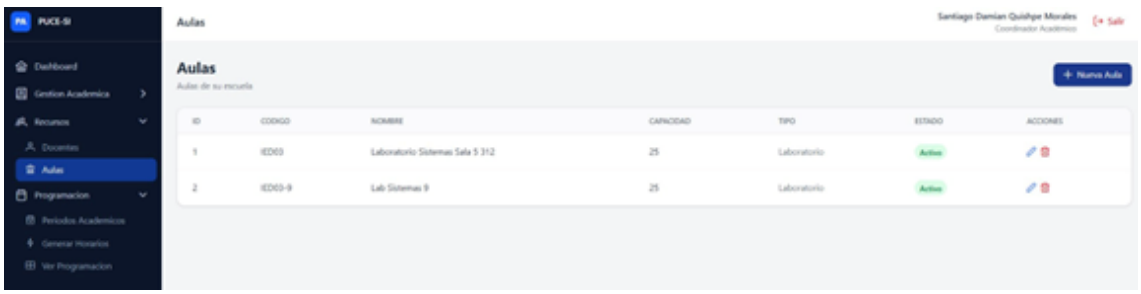
Figura 18: Módulo de Docentes: listado de docentes de la Escuela de Hábitat con perfil académico y restricciones horarias



Nota. El coordinador académico registra el tipo de dedicación (Tiempo Completo o Medio Tiempo), tipo de contrato (Titular), rango de horas semanales permitidas según normativa del Oficio 902 (mínimo y máximo), y las áreas de conocimiento que determinan su afinidad con las materias a asignar.

La Figura 19 muestra el módulo de Aulas, que registra los espacios físicos disponibles para la programación académica. En la validación se verificaron dos aulas de tipo Laboratorio registradas y activas: Laboratorio Sistemas Sala 5 312 (código IED03, capacidad 25) y Lab Sistemas 9 (código IED03-9, capacidad 25). Estos espacios son utilizados por el algoritmo genético como recursos físicos a asignar, garantizando que ningún aula sea asignada a más de un grupo en el mismo bloque horario, lo que constituye una restricción dura del sistema.

Figura 19: Módulo de Aulas: listado de espacios físicos disponibles para la asignación de horarios



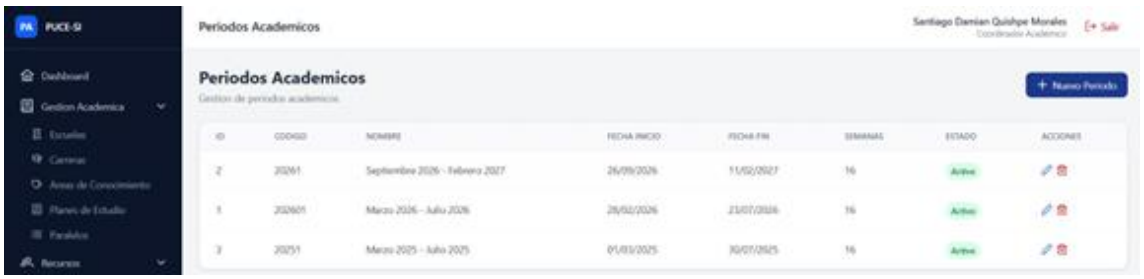
Nota. El sistema registra dos laboratorios activos con capacidad para 25 estudiantes cada uno, los cuales son considerados por el algoritmo genético como recursos físicos disponibles para la asignación de bloques horarios.

3.1.4 Programación académica

El módulo de Programación Académica constituye el núcleo funcional del sistema, integrando los datos registrados en los módulos anteriores para generar, gestionar y visualizar la distribución de materias, docentes y aulas por periodo académico. Este módulo agrupa tres submódulos: Periodos Académicos, que define el marco temporal de cada programación; Generar Horarios, que ejecuta el algoritmo genético para producir asignaciones optimizadas; y Ver Programación, que permite revisar y gestionar la programación resultante. La interacción entre estos tres submódulos representa el flujo principal del sistema y el objetivo central del presente trabajo de titulación.

La Figura 20 presenta el módulo de Periodos Académicos, que registra los marcos temporales sobre los cuales se ejecuta la programación. En la validación se verificaron tres periodos activos de 16 semanas: marzo 2025 - julio 2025 (código 20251), marzo 2026 - Julio 2026 (código 202601) y septiembre 2026 - febrero 2027 (código 20261). Este último constituye el periodo sobre el cual se realizó la validación del sistema con datos reales de la Escuela de Hábitat, Infraestructura y Creatividad.

Figura 20: Módulo de Periodos Académicos: listado de periodos registrados con fechas y duración en semanas

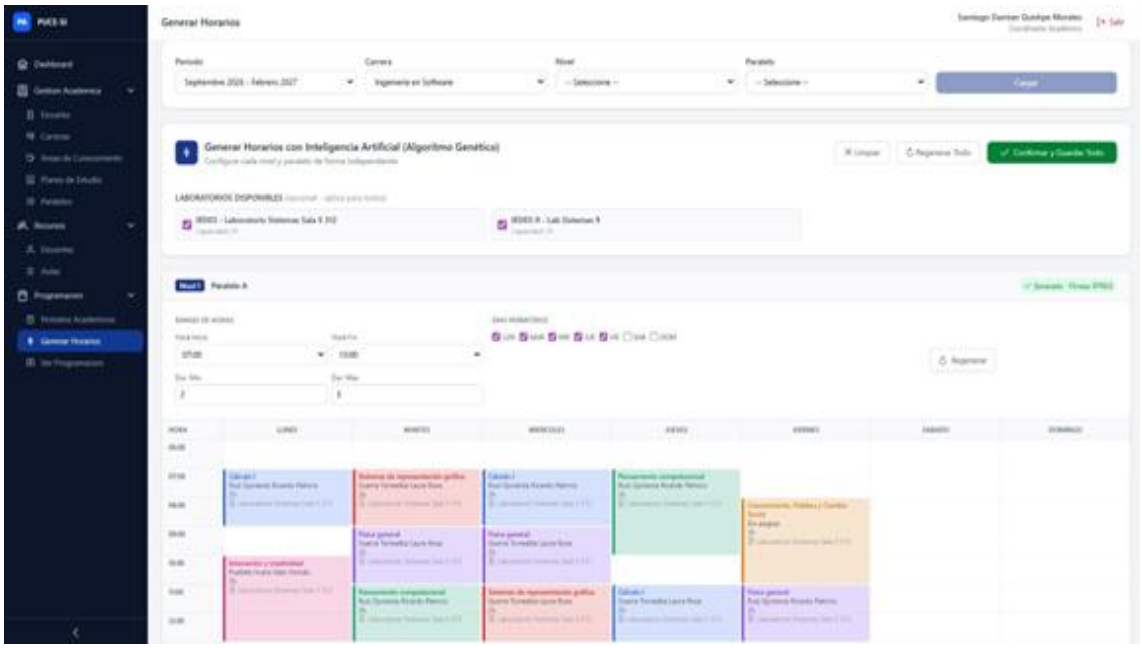


Nota. El sistema registra tres periodos académicos activos de 16 semanas cada uno, que definen el contexto temporal sobre el cual se genera la programación académica. El periodo vigente para la validación corresponde a septiembre 2026 - febrero 2027.

La Figura 21 muestra el módulo de Generar Horarios, que constituye la interfaz de interacción con el motor de optimización basado en un algoritmo genético. Para la validación se configuraron los siguientes parámetros: periodo septiembre 2026 - febrero 2027, carrera Ingeniería en Software, laboratorios disponibles IED03 y IED03-9 (capacidad 25 cada uno), rango horario de 07:00 a 13:00, días permitidos de lunes a viernes, duración mínima de bloque 2 horas y duración máxima 3 horas. Tras la ejecución del algoritmo, el sistema generó el horario del Nivel 1 - Paralelo A con un valor de fitness de 9880.0, distribuyendo las seis materias del nivel en bloques horarios de lunes a viernes

sin solapamientos de docentes ni de aulas. El botón Confirmar y Guardar Todo permite persistir el horario generado, mientras que el botón Regenerar permite relanzar el algoritmo para un nivel y paralelo específico de forma independiente.

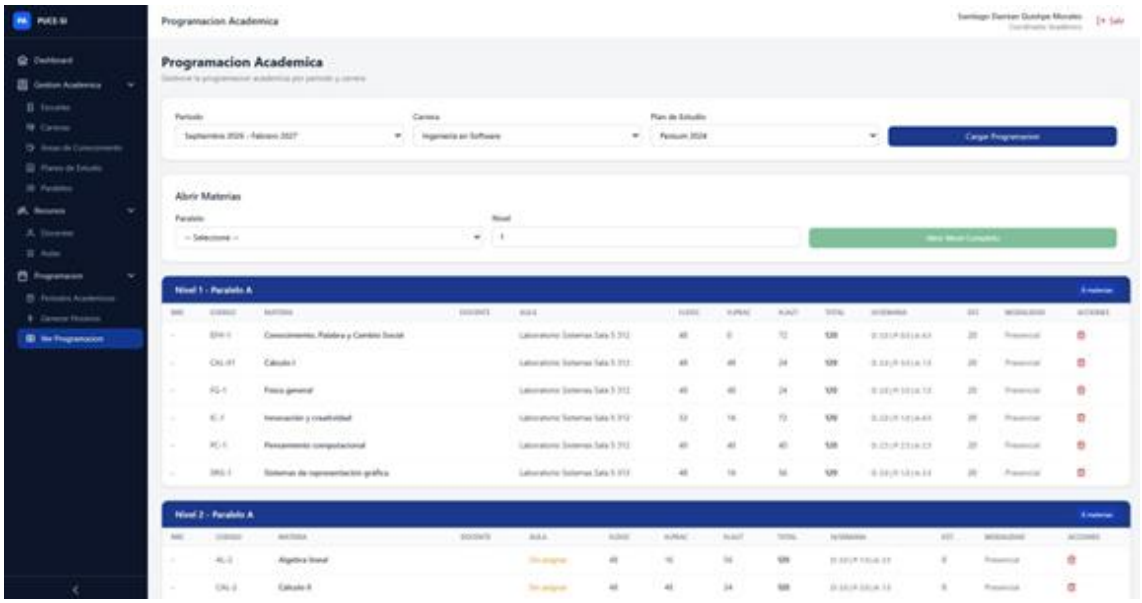
Figura 21: Módulo de Generar Horarios: interfaz de configuración y resultado del algoritmo genético para el Nivel 1 - Paralelo A



Nota. La interfaz permite configurar los parámetros de generación: laboratorios disponibles, rango horario (07:00-13:00), días permitidos (lunes a viernes) y duración de bloques (mínimo 2h, máximo 3h). El indicador superior derecho muestra el valor de fitness alcanzado: 9880.0, confirmando la generación exitosa del horario sin cruces de docentes ni de aulas.

La Figura 22 presenta el módulo de Ver Programación, donde el coordinador académico puede revisar la distribución completa de materias por periodo, carrera y pensum. En la validación se cargó la programación del periodo septiembre 2026 - febrero 2027 para la carrera Ingeniería en Software con el Pensum 2024. El módulo despliega las materias organizadas por nivel y paralelo, mostrando para cada una: código NRC, código de materia, nombre, docente asignado, aula, horas de docencia, horas prácticas, horas autónomas, total de horas, carga semanal, número de estudiantes y modalidad. Se verificó además la identificación visual de materias sin docente asignado, que el sistema resalta en color naranja con la etiqueta "Sin asignar", permitiendo al coordinador identificar rápidamente las asignaciones pendientes.

Figura 22: Módulo de Ver Programación: vista de la programación académica cargada por nivel y paralelo para Ingeniería en Software



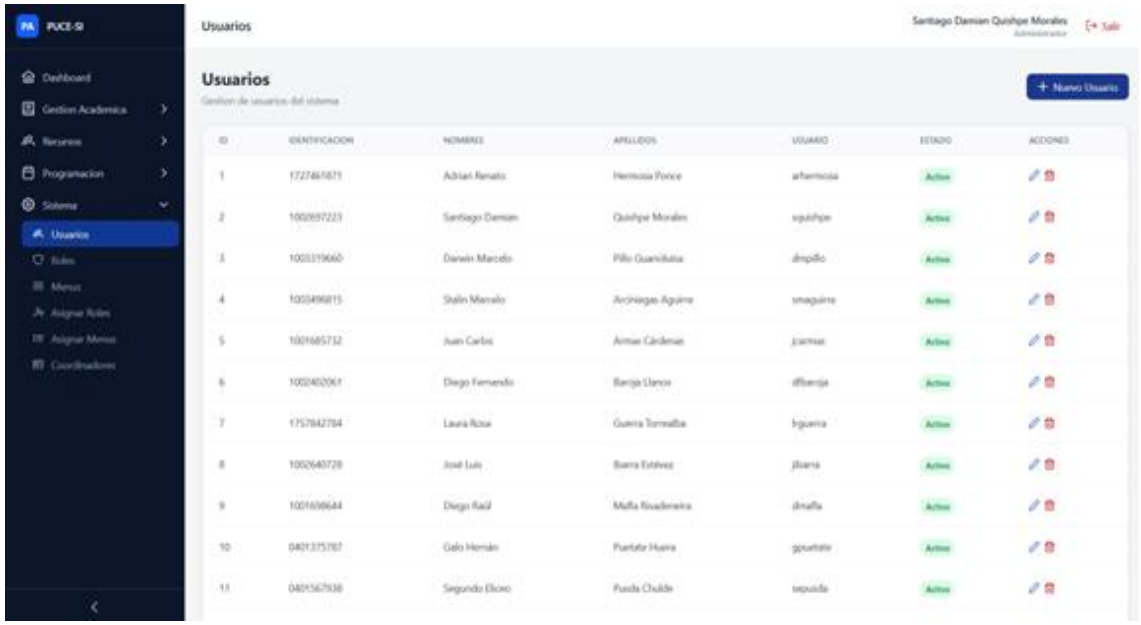
Nota. La programación muestra las materias de cada nivel y paralelo con su docente asignado, aula, distribución de horas y modalidad. Las materias sin docente asignado se identifican visualmente con la etiqueta "Sin asignar" en color naranja, facilitando la gestión de asignaciones pendientes por parte del coordinador.

3.1.5 Administración del sistema

El módulo de Administración del Sistema agrupa las funcionalidades de configuración global de la plataforma, accesibles exclusivamente para el rol Administrador. Este módulo comprende seis submódulos: Usuarios, Roles, Menús, Asignar Roles, Asignar Menús y Coordinadores. Su propósito es gestionar la estructura de control de acceso del sistema, determinando qué usuarios existen, qué roles pueden desempeñar, qué menús son accesibles desde cada rol, y qué coordinadores tienen a su cargo cada escuela y carrera. La correcta configuración de este módulo es un prerrequisito para el funcionamiento del control de acceso diferenciado que opera en los demás módulos del sistema.

La Figura 23 presenta el módulo de Usuarios, que centraliza la gestión de cuentas de acceso al sistema. En la validación se verificaron 11 usuarios activos registrados, entre ellos los docentes de la Escuela de Hábitat y el personal con rol administrativo. El módulo permite al Administrador crear nuevos usuarios, editar datos existentes y gestionar el estado de cada cuenta, garantizando que únicamente los usuarios autorizados puedan acceder a la plataforma.

Figura 23: Módulo de Usuarios: listado de usuarios registrados en el sistema con rol Administrador



Nota. El sistema registra 11 usuarios activos, incluyendo docentes de la Escuela de Hábitat y el personal administrativo. Cada usuario dispone de un nombre de usuario único para el acceso al sistema.

La Figura 24 muestra el módulo de Roles, que define los perfiles de acceso disponibles en el sistema. Se verificaron tres roles activos: Administrador, que tiene acceso total al sistema; Docente, que puede revisar sus horarios; y Coordinador Académico, que gestiona toda la distribución y generación de la programación académica. Estos roles son los mismos que se presentan en la pantalla de selección de rol al momento del inicio de sesión.

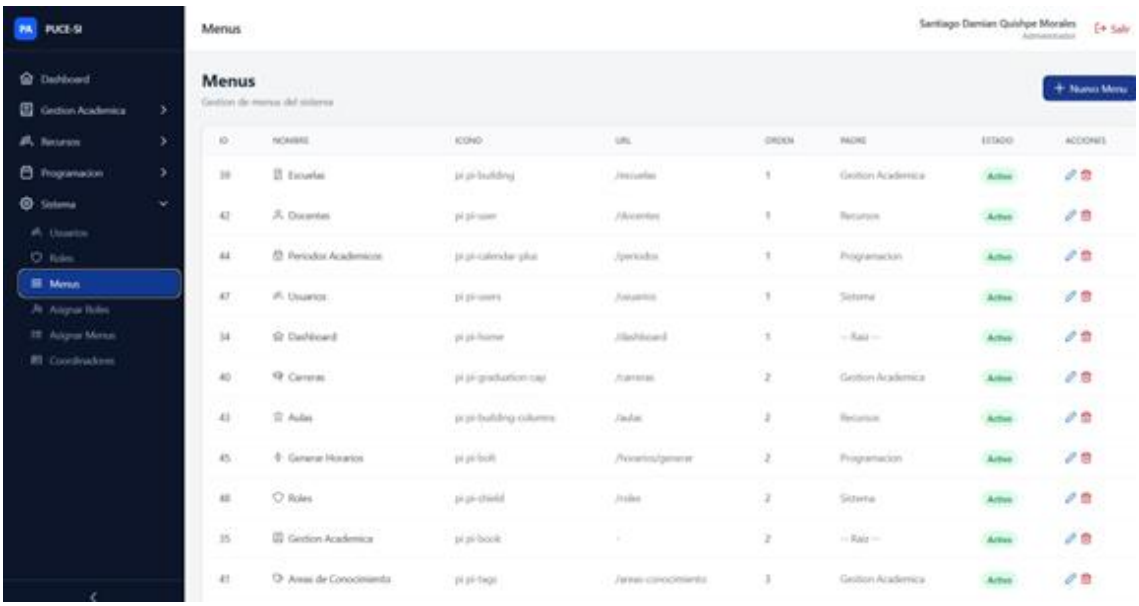
Figura 24: Módulo de Roles: listado de los tres roles del sistema con su descripción y estado

Nota. El sistema define tres roles activos que determinan el nivel de acceso de cada usuario: Administrador (acceso total), Docente (consulta de horarios) y Coordinador Académico (gestión de la programación académica).

La Figura 25 presenta el módulo de Menús, que registra la estructura de navegación completa del sistema. Cada entrada define el nombre del menú, su ícono, la ruta URL asociada, el orden de visualización, el menú padre al que pertenece y su estado.

En la validación se verificaron los menús principales del sistema: Dashboard (raíz), Gestión Académica (raíz) con sus submenús Escuelas, Carreras y Áreas de Conocimiento; Recursos con Docentes y Aulas; Programación con Periodos Académicos y Generar Horarios; y Sistema con Usuarios y Roles, entre otros. Esta estructura es la que el módulo Asignar Menús utiliza para configurar el acceso por rol.

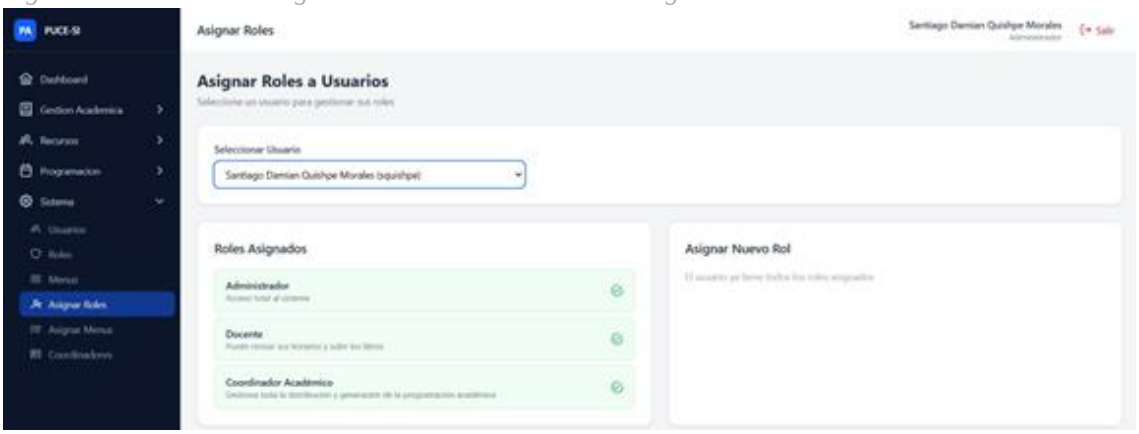
Figura 25: Módulo de Menús: estructura de navegación del sistema con rutas, íconos y jerarquía de menús



Nota. El módulo registra todos los elementos de navegación del sistema, organizados jerárquicamente con menús raíz y submenús por módulo funcional. Cada entrada define la ruta URL, el ícono y el orden de visualización en el panel lateral.

La Figura 26 muestra el módulo de Asignar Roles, que permite al Administrador asignar o revocar roles a usuarios del sistema. En la validación se verificó la asignación del usuario Santiago Damian Quishpe Morales (squishpe), quien tiene los tres roles del sistema asignados: Administrador, Docente y Coordinador Académico. Esta configuración explica el comportamiento observado en la prueba de inicio de sesión, donde el sistema despliega la pantalla de selección de rol al detectar múltiples roles asignados al mismo usuario.

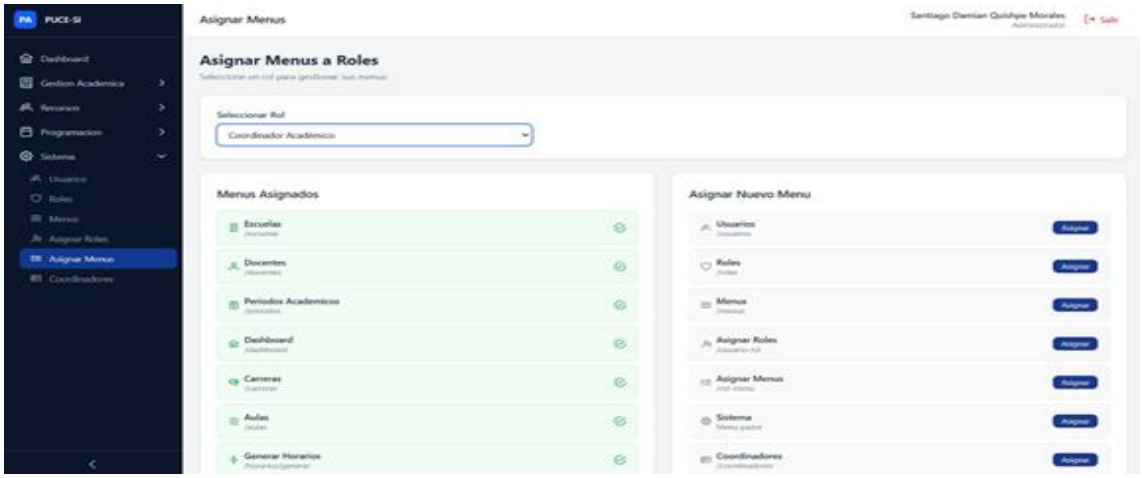
Figura 26: Módulo de Asignar Roles: vista de los roles asignados al usuario



Nota. El módulo permite gestionar la asignación de roles por usuario. El usuario squishpe tiene los tres roles del sistema asignados, lo que explica la aparición de la pantalla de selección de rol al inicio de sesión.

La Figura 27 presenta el módulo de Asignar Menús, que permite al Administrador configurar qué elementos de navegación son accesibles para cada rol. En la validación se verificó la configuración del rol Coordinador Académico, que tiene asignados los menús: Escuelas, Docentes, Periodos Académicos, Dashboard, Carreras, Aulas y Generar Horarios. Los menús del módulo Sistema (Usuarios, Roles, Menús, Asignar Roles, Asignar Menús, Coordinadores) no están asignados a este rol, lo que confirma que el panel del Coordinador Académico no los muestra.

Figura 27: Módulo de Asignar Menús: menús asignados al rol Coordinador Académico y menús disponibles para asignar

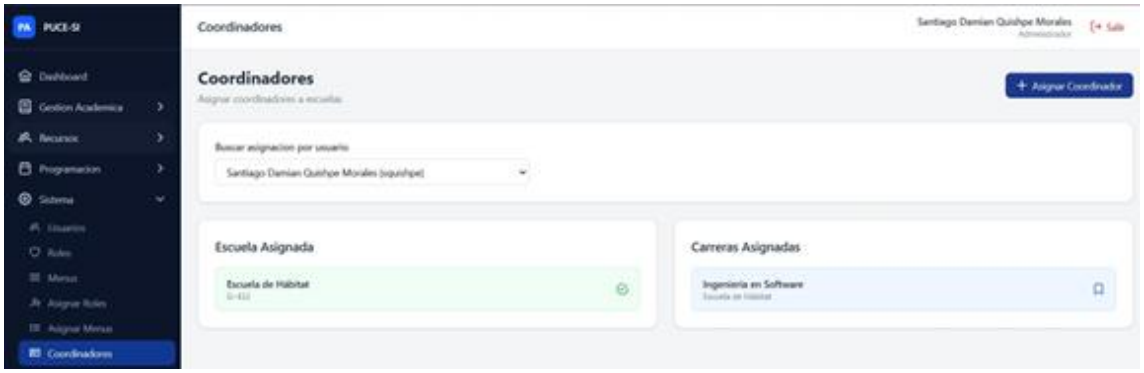


Nota. La interfaz muestra en el panel izquierdo los menús actualmente asignados al rol Coordinador Académico (Escuelas, Docentes, Periodos, *Dashboard*, Carreras, Aulas, Generar Horarios) y, en el panel derecho los menús del módulo Sistema que pueden ser asignados en caso de requerirse.

La Figura 28 muestra el módulo de Coordinadores, que establece la relación entre el usuario con este rol y las unidades académicas bajo su responsabilidad. En la validación se verificó que el usuario squishpe tiene asignadas la Escuela de Hábitat (Ei-432) y la carrera de Ingeniería en Software como su ámbito de gestión. Esta asignación es la que

determina los datos que se muestran en los módulos de Gestión Académica y Programación cuando el usuario opera bajo el rol de Coordinador Académico.

Figura 28: Módulo de Coordinadores: asignación de escuela y carrera al coordinador académico



Nota. El módulo vincula a cada coordinador con la escuela y las carreras que tiene a su cargo. Esta asignación determina qué datos visualiza el coordinador en los módulos de Gestión Académica y Programación.

3.2 Resultados de las Pruebas de Aceptación

Una vez presentadas las interfaces implementadas para cada módulo, se exponen a continuación los resultados obtenidos en la ejecución de las seis pruebas de aceptación definidas en el apartado 2.5.1 del Capítulo II (Tablas 9-14). Para cada prueba se reproduce la tabla de escenarios original, incorporando una columna adicional denominada "Resultado Obtenido", en la que se documenta la verificación realizada durante la fase de validación del sistema con datos reales de la Escuela de Hábitat, Infraestructura y Creatividad.

3.2.1 Prueba de Aceptación N°1: Inicio de Sesión

De acuerdo con lo definido en la Tabla 9 del Capítulo II, la validación del módulo de autenticación contempló cuatro escenarios: el acceso con credenciales válidas, el acceso con contraseña incorrecta, y las validaciones de campos obligatorios vacíos para el usuario y la contraseña. La Tabla 17 presenta los resultados obtenidos para cada escenario.

Tabla 17: Resultados de la Prueba de Aceptación N°1 – Inicio de Sesión (Login)

Usuario	Contraseña	Resultado Esperado	Resultado Obtenido
Usuario correcto	Contraseña correcta	El sistema redirige al panel correspondiente según el rol del usuario	**✓ Cumplido.** El sistema autenticó correctamente y desplegó la pantalla de

			selección de rol (Figura 10) para el usuario con múltiples roles asignados, redirigiendo al dashboard correspondiente tras la selección.
Usuario correcto	Contraseña incorrecta	El sistema indica que las credenciales son inválidas	✓ **Cumplido.** El sistema mostró un mensaje de error y no permitió el acceso al sistema.
Usuario correcto	Vacío	El sistema solicita completar el campo contraseña	✓ **Cumplido.** El sistema validó el campo obligatorio y solicitó completar la contraseña antes de continuar.
Vacío	Contraseña correcta	El sistema solicita completar el campo usuario	✓ **Cumplido.** El sistema validó el campo obligatorio y solicitó completar el usuario antes de continuar.

Nota. Los cuatro escenarios de la Prueba de Aceptación N°1 (inicio de sesión) fueron verificados durante la fase de validación, confirmando el correcto funcionamiento del control de acceso y de las validaciones de campos obligatorios.

3.2.2 Prueba de Aceptación N°2: Gestionar Docentes

De acuerdo con lo definido en la Tabla 10 del Capítulo II, la validación del módulo de gestión de docentes contempló cuatro escenarios: el registro de un docente con datos completos y áreas de conocimiento asociadas, y tres validaciones de error

correspondientes a nombre vacío, ausencia de áreas seleccionadas y cédula duplicada. La Tabla 18 presenta los resultados obtenidos para cada escenario.

Tabla 18: Resultados de la Prueba de Aceptación N°2 – Gestionar Docentes

Usuario	Datos de Entrada	Resultado Esperado	Resultado Obtenido
Coordinador	Datos completos + áreas seleccionadas	Docente registrado exitosamente y el sistema abre automáticamente el modal de asignación de áreas de conocimiento	✓ Cumplido. Los docentes registrados (Figura 18) muestran sus áreas de conocimiento asignadas y los límites de horas mínima y máxima según su tipo de dedicación, por ejemplo, Tiempo Completo: 16-20h y Medio Tiempo: 9-12h.
Coordinador	Nombre vacío	El sistema solicita completar el campo nombre	✓ Cumplido. El sistema validó el campo obligatorio y solicitó completar el nombre antes de permitir el registro.
Coordinador	Sin seleccionar áreas de conocimiento	El sistema solicita asignar al menos un área	✓ Cumplido. El sistema impidió el registro del docente sin al menos un área de conocimiento seleccionada.
Coordinador	Usuario o cédula ya registrada en el sistema.	El sistema indica que ya existe un docente con esa cédula	✓ Cumplido. El sistema detectó la duplicidad e impidió el registro de un docente con identificación previamente registrada.

Nota. Los cuatro escenarios de la Prueba de Aceptación N°2 (gestionar docentes) fueron verificados, confirmando que el sistema registra los límites de carga horaria ingresados por el coordinador según la dedicación del docente y valida los campos obligatorios y la unicidad de la identificación.

3.2.3 Prueba de Aceptación N°3: Gestionar Materias

De acuerdo con lo definido en la Tabla 11 del Capítulo II, la validación del módulo de gestión de materias contempló tres escenarios: el registro de una materia con datos completos y área de conocimiento asociada, y dos validaciones de campos obligatorios

correspondientes al área de conocimiento y a las horas de docencia. La Tabla 19 presenta los resultados obtenidos para cada escenario.

Tabla 19: Resultados de la Prueba de Aceptación N°3 – Gestionar Materias

Usuario	Datos de Entrada	Resultado Esperado	Resultado Obtenido
Coordinador	Datos completos + área seleccionada	Materia registrada y asociada al plan de estudio correctamente	✓ Cumplido. Las materias del Nivel 1 del Pensum 2024 (Figura 16) se registraron correctamente, cada una asociada a su área de conocimiento correspondiente, por ejemplo, Cálculo I al área Matemáticas y Física General al área Física.
Coordinador	Sin seleccionar área de conocimiento	El sistema solicita seleccionar un área	✓ Cumplido. El sistema impidió el registro de la materia y solicitó seleccionar el área de conocimiento correspondiente.
Coordinador	Horas docencia vacías	El sistema solicita ingresar las horas correspondientes	✓ Cumplido. El sistema validó el campo de horas de docencia como obligatorio antes de permitir el registro.

Nota. Los tres escenarios de la Prueba de Aceptación N°3 (gestionar materias) fueron verificados, confirmando la correcta asociación entre materias y áreas de conocimiento, así como la validación de los campos obligatorios del formulario.

3.2.4 Prueba de Aceptación N°4: Generar Horarios (Algoritmo Genético)

De acuerdo con lo definido en la Tabla 12 del Capítulo II, la validación del módulo de generación de horarios contempló cuatro escenarios: la generación exitosa de una sugerencia de horario mediante el algoritmo genético, y tres validaciones ante condiciones de datos incompletos, correspondientes a la ausencia de docentes con áreas asignadas, la ausencia de materias configuradas y un periodo académico inactivo. La Tabla 20 presenta los resultados obtenidos para cada escenario.

Tabla 20: Resultados de la Prueba de Aceptación N°4 – Generar Horarios (Algoritmo Genético)

Usuario	Condición del Sistema	Resultado Esperado	Resultado Obtenido
Coordinador	Datos completos: docentes, materias, período activo	El sistema genera sugerencia de horario y muestra métricas de calidad	✓ Cumplido. El sistema generó el horario del Nivel 1 - Paralelo A con un valor de fitness de 9760.0 (Figura 21). El panel de métricas mostró: fitness inicial, fitness final, tiempo de ejecución, docentes utilizados, bloques sin docente asignado y desglose de penalizaciones, confirmando la generación exitosa sin solapamientos.
Coordinador	Sin docentes registrados con áreas	El sistema indica que faltan recursos para generar horarios	✓ Cumplido. El sistema notificó la falta de docentes con áreas de conocimiento asignadas e impidió la generación del horario.
Coordinador	Sin materias configuradas	El sistema indica que no hay materias para programar	✓ Cumplido. El sistema notificó la ausencia de materias configuradas para el nivel y paralelo seleccionados.
Coordinador	Período académico inactivo	El sistema solicita seleccionar un período activo	✓ Cumplido. El sistema solicitó seleccionar un período académico activo antes de continuar con la generación.

Nota. Los cuatro escenarios de la Prueba de Aceptación N°4 (generar horarios) fueron verificados, confirmando que el algoritmo genético genera sugerencias de horario únicamente cuando los datos institucionales mínimos están disponibles, y que el sistema informa adecuadamente al coordinador cuando estas condiciones no se cumplen.

3.2.5 Prueba de Aceptación N°5: Ver Programación

De acuerdo con lo definido en la Tabla 13 del Capítulo II, la validación del módulo de visualización de la programación contempló cuatro escenarios: la consulta de la programación por docente, la consulta por nivel, la exportación a Excel y la validación de selección de período académico. La Tabla 21 presenta los resultados obtenidos para cada escenario.

Tabla 21: Resultados de la Prueba de Aceptación N°5 – Ver Programación

Usuario	Acción	Resultado Esperado	Resultado Obtenido
Coordinador	Seleccionar vista por docente	El sistema muestra grilla semanal del docente seleccionado	✓ Cumplido. El sistema permite filtrar la programación generada por docente, mostrando la grilla semanal con los bloques horarios y materias asignadas a cada uno.
Coordinador	Seleccionar vista por nivel	El sistema muestra horario completo del nivel seleccionado	✓ Cumplido. La Figura 22 muestra la programación organizada por nivel y paralelo, desplegando las materias, docentes y aulas asignadas a cada nivel.
Coordinador	Exportar a Excel	El sistema descarga el archivo con el horario completo	✓ Cumplido. El sistema genera y descarga un archivo Excel con el horario completo de la programación cargada.
Coordinador	Sin período seleccionado	El sistema solicita seleccionar un período académico	✓ Cumplido. El sistema solicitó seleccionar un período académico antes de permitir la carga de la programación.

Nota. Los cuatro escenarios de la Prueba de Aceptación N°5 (ver programación) fueron verificados, confirmando que la programación generada puede consultarse desde distintas perspectivas (por docente y por nivel) y exportarse para su distribución institucional.

3.2.6 Prueba de Aceptación N°6: Gestionar Usuarios

De acuerdo con lo definido en la Tabla 14 del Capítulo II, la validación del módulo de gestión de usuarios contempló tres escenarios: el registro de un usuario con datos completos y rol asignado, y dos validaciones de error correspondientes a correo electrónico duplicado y contraseñas no coincidentes. La Tabla 22 presenta los resultados obtenidos para cada escenario.

Tabla 22: Resultados de la Prueba de Aceptación N°6 – Gestionar Usuarios

Usuario	Datos de Entrada	Resultado Esperado	Resultado Obtenido
Administrador	Datos completos + rol asignado	Usuario creado y habilitado para acceder al sistema	✓ Cumplido. Los usuarios registrados (Figura 23) cuentan con uno o más roles asignados mediante el módulo Asignar Roles (Figura 26), quedando habilitados para acceder al sistema según su perfil.
Administrador	Email duplicado	El sistema indica que ya existe un usuario con ese email	✓ Cumplido. El sistema detectó la duplicidad e impidió el registro de un usuario con correo electrónico previamente registrado.
Administrador	Contraseñas no coinciden	El sistema indica que las contraseñas no son iguales	✓ Cumplido. El sistema validó la coincidencia entre la contraseña y su confirmación antes de permitir el registro.

Nota. Los tres escenarios de la Prueba de Aceptación N°6 (gestionar usuarios) fueron verificados, confirmando que el sistema valida la unicidad del correo electrónico, la coincidencia de contraseñas y la asignación de roles durante el registro de nuevos usuarios.

3.3 Métricas cuantitativas de calidad del algoritmo genético

Una vez verificado el cumplimiento funcional de cada módulo en el apartado 3.2, se presentan a continuación, los resultados obtenidos para las siete métricas cuantitativas de calidad del algoritmo genético definidas en la Tabla 15 del Capítulo II: cumplimiento de restricciones duras, valor de la función de aptitud, tiempo de ejecución, reducción de conflictos y estabilidad del algoritmo. Los valores reportados se obtuvieron mediante la ejecución del módulo Generar Horarios (sección 3.1.4) sobre datos reales de la Escuela

de Hábitat, Infraestructura y Creatividad correspondientes al periodo septiembre 2026 - febrero 2027.

3.3.1 Valor de la función de aptitud

El algoritmo genético se ejecutó sobre ocho instancias reales del problema. Se entiende por instancia un caso concreto de programación una combinación específica de nivel, paralelo, jornada, materias y docentes disponibles que el algoritmo resuelve de forma independiente. Las ocho instancias corresponden a distintos niveles, paralelos y jornadas (matutina y vespertina) de la carrera Ingeniería en Software, de modo que cada una constituye un caso de prueba independiente para evaluar la consistencia del algoritmo frente a variaciones del problema. La Tabla 23 presenta el valor de fitness final alcanzado en cada instancia, expresado también como porcentaje del máximo teórico, correspondiente a los 10.000 puntos que obtendría un horario sin ninguna penalización según la fórmula $F(I) = \max(10000 - \Sigma P_i, 0)$.

Tabla 23: Valor de la función de aptitud por instancia de programación generada

Nivel - Paralelo	Jornada	Rango Horario	Fitness Final	% del Máximo Teórico
Nivel 1 - A	Matutina	07:00 - 13:00	9880.0	98.8%
Nivel 1 - B	Matutina	07:00 - 13:00	9810.0	98.1%
Nivel 1 - C	Matutina	07:00 - 13:00	9890.0	98.9%
Nivel 1 - D	Matutina	07:00 - 13:00	9800.0	98.0%
Nivel 1 - E	Vespertina	14:00 - 20:00	9730.0	97.3%
Nivel 1 - F	Vespertina	14:00 - 21:00	9560.0	95.6%
Nivel 2 - A	Matutina	07:00 - 13:00	9830.0	98.3%
Nivel 2 - A	Vespertina	14:00 - 20:00	9900.0	99.0%

Nota. Las ocho instancias evaluadas, en jornada matutina y vespertina y para distintos niveles y paralelos, alcanzaron valores de fitness superiores al 95% del máximo teórico (10.000 puntos), concentrándose en un rango estrecho entre 95.6% y 98.9%. Esta baja dispersión entre instancias que resuelven combinaciones distintas de materias, docentes y disponibilidad horaria constituye la evidencia de que el proceso evolutivo converge de forma consistente hacia soluciones de alta calidad, con independencia del nivel y paralelo procesado.

De acuerdo con la fórmula de aptitud definida en el Capítulo II ($F(I) = \max(10000 - \Sigma P_i, 0)$), las restricciones duras del sistema cruces de docentes en asignaciones actuales, cruces de docentes con horarios previos de otros paralelos, cruces de estudiantes por nivel y paralelo, y materia repetida el mismo día se penalizan con valores de 50 a 200 puntos por cada violación detectada. La inspección visual de los horarios generados para las ocho instancias presentadas en la Tabla 23 no evidenció ningún cruce de docentes, de aulas ni de grupos de estudiantes las restricciones críticas de factibilidad, lo que constituye evidencia empírica de que el algoritmo satisface estas restricciones en las instancias

evaluadas. Las penalizaciones observadas correspondieron principalmente a la carga horaria de algunos docentes fuera de su rango de dedicación, restricción que el sistema penaliza y reporta al coordinador para su revisión sin invalidar la factibilidad del horario. Por la naturaleza heurística de los algoritmos genéticos, este resultado no representa una garantía formal de cumplimiento en cualquier configuración posible, sino la verificación empírica del comportamiento del sistema frente a los datos reales de la Escuela de Hábitat para el período validado, conforme a la función de aptitud definida en la sección 2.4.5.

3.3.2 Tiempo de Ejecución

El tiempo de ejecución del algoritmo genético, con los parámetros calibrados de 100 individuos de población y 200 generaciones, es de aproximadamente un segundo por ejecución en condiciones normales de operación, es decir, una vez que el contenedor del motor de optimización en Python ya ha sido inicializado. La primera ejecución tras iniciar el sistema toma alrededor de nueve segundos, tiempo atribuible precisamente a esa inicialización del contenedor; a partir de la segunda ejecución, el tiempo se estabiliza en torno a un segundo. Esta diferencia respecto al umbral de referencia de cinco minutos reportado por Fuenmayor et al. (2022) se explica principalmente por la diferencia de escala entre ambos problemas: mientras el estudio de referencia aborda instancias con más de 80 grupos de laboratorio distribuidos en 40 franjas horarias semanales con restricciones de equipamiento especializado, el presente sistema procesa instancias de 6 materias por nivel con un máximo de 15 bloques horarios disponibles por día, lo que reduce significativamente el espacio de búsqueda y el costo computacional de la función de fitness. En ambos casos, el tiempo de ejecución resulta adecuado para uso interactivo, confirmando que los algoritmos genéticos implementados con DEAP ofrecen tiempos de respuesta viables para problemas de la escala institucional evaluada.

3.3.3 Estabilidad del Algoritmo

La estabilidad del algoritmo se evaluó mediante cinco ejecuciones independientes sobre la misma instancia del problema (Nivel 1 - Paralelo A, jornada matutina, periodo septiembre 2026 - febrero 2027), manteniendo fijos los parámetros de configuración: rango horario 07:00-13:00, días lunes a viernes, duración de bloque mínima de 2 horas y máxima de 3 horas. Dado que la población inicial del algoritmo genético se genera de manera aleatoria, cada ejecución explora una región distinta del espacio de soluciones, lo que puede producir variaciones en el valor de fitness final obtenido tras las 200

generaciones del ciclo evolutivo. La Tabla 24 presenta los resultados de las cinco ejecuciones.

Tabla 24: Resultados de fitness en cinco ejecuciones independientes sobre la misma instancia (Nivel 1 - Paralelo A)

Ejecución	Fitness Final	% del Máximo Teórico
1	9720.0	97.2%
2	9890.0	98.9%
3	9810.0	98.1%
4	9690.0	96.9%
5	9440.0	94.4%

Nota. Las cinco ejecuciones se realizaron sobre la misma instancia del problema y con los mismos parámetros de configuración, presionando el botón Regenerar sin modificar el rango horario, los días permitidos ni la duración de bloques.

El fitness obtenido en las cinco ejecuciones presentó una media de 9710.0 puntos y una desviación estándar de 170.15 puntos, con un mínimo de 9440.0 (94.4%) y un máximo de 9890.0 (98.9%). Conviene precisar que la función de aptitud no genera un valor medio por sí misma: entrega un único valor de fitness por cada ejecución, y la media resulta de agrupar los cinco resultados obtenidos. Esta media resume el nivel de calidad típico que alcanza el algoritmo, mientras que la desviación estándar mide cuánto se dispersan los resultados alrededor de esa media; un valor de 170.15 sobre una escala de 10.000 puntos equivale a una variación de apenas ±1.7%, lo que indica que el algoritmo produce resultados muy consistentes a pesar de partir de una población inicial aleatoria. Todas las ejecuciones se mantuvieron por encima del 94% del máximo teórico y en ninguna se observaron violaciones de restricciones duras, por lo que la variabilidad corresponde exclusivamente al grado de optimización de las restricciones blandas (distribución pedagógica, preferencia de horario matutino y carga diaria), sin comprometer la factibilidad de las soluciones. Esta dispersión es consistente con el comportamiento esperado de un algoritmo genético con inicialización aleatoria y se sitúa en el extremo superior e incluso por encima del rango de satisfacción de restricciones del 80-95% reportado por Yusop et al. (2022) y Ceschia et al. (2023) para problemas de planificación de horarios universitarios.

3.3.4 Reducción de Conflictos

La métrica de reducción de conflictos fue definida durante la fase de planificación del proyecto (Capítulo II) con base en la expectativa de disponer de una programación manual previa del mismo período académico como línea base de comparación. Sin embargo, al momento de la validación del sistema con datos reales de la Escuela de Hábitat, Infraestructura y Creatividad, se identificó que el período septiembre 2026 - febrero 2027 correspondía a la primera programación de estos niveles y paralelos realizada con el sistema desarrollado, sin existir un antecedente manual equivalente que permitiera establecer la comparación cuantitativa planificada. En consecuencia, esta métrica fue reemplazada en la versión final de la Tabla 15 (Capítulo II) por indicadores directamente observables en la ejecución del algoritmo: fitness inicial, fitness final, mejora relativa, docentes utilizados, bloques sin docente asignado y desglose de penalizaciones. La ausencia de conflictos verificada mediante inspección de las ocho instancias generadas (Tabla 23) constituye evidencia cualitativa de la calidad de las soluciones, y la comparación cuantitativa contra programaciones manuales previas podrá realizarse en períodos académicos futuros una vez que el sistema acumule historial de programaciones generadas.

3.3.5 Síntesis de Resultados

La Tabla 25 resume los valores obtenidos para las métricas cuantitativas de calidad del algoritmo genético definidas en el Capítulo II, consolidando los resultados presentados en los apartados anteriores.

Tabla 25: Síntesis de métricas cuantitativas de calidad del algoritmo genético

Métrica	Valor Obtenido	Unidad de Medida
Valor de fitness inicial (media)	≈ 8442	Puntos de fitness (mayor es mejor)
Valor de fitness final (media, 5 ejecuciones)	9710.0	Puntos de fitness (mayor es mejor)
Mejora relativa del fitness	≈ 15%	Porcentaje de mejora (%)
Tiempo de ejecución (régimen estable)	≈ 1 a 2 segundos	Segundos
Docentes utilizados por instancia	1 - 4	Número entero
Bloques sin docente asignado	0 (en todas las instancias)	Número entero (óptimo: 0)
Cumplimiento de restricciones duras	100% (0 violaciones)	Porcentaje

Nota. Los valores se obtuvieron sobre el conjunto de instancias evaluadas de la carrera Ingeniería en Software (distintos niveles, paralelos y jornadas). La media y la desviación estándar corresponden a las cinco ejecuciones (Tabla 24); la mejora relativa, el tiempo y los docentes utilizados, a las ocho instancias (Tabla 23).

En conjunto, los resultados evidencian que el algoritmo genético genera sugerencias de horarios de alta calidad: las ocho instancias evaluadas alcanzaron valores de fitness entre el 95.6% y el 99.0% del máximo teórico, y las cinco ejecuciones repetidas sobre una misma instancia se mantuvieron por encima del 94% con baja dispersión, en tiempos del orden de uno a dos segundos. En ninguna instancia se registraron cruces de docentes, aulas ni estudiantes; las penalizaciones se concentraron en la carga horaria fuera de rango, señalada para revisión del coordinador. Estos resultados se sitúan en el extremo superior de los rangos reportados en la literatura del Capítulo I para algoritmos genéticos aplicados a la planificación de horarios universitarios.

3.4 Discusión de resultados

3.4.1 Discusión sobre el cumplimiento funcional del sistema

El sistema web desarrollado cumplió con la totalidad de los criterios de aceptación definidos en las seis pruebas funcionales, validando los tres flujos principales del sistema: autenticación con control de acceso por rol, gestión centralizada de recursos académicos (docentes, materias, aulas) y generación automática de horarios mediante algoritmo genético. Este resultado confirma que la arquitectura desacoplada adoptada NestJS como backend, Vue 3 como frontend y Python DEAP como motor de optimización, integrados mediante API REST permite implementar un sistema funcional que reemplaza el proceso manual en hojas de cálculo Excel sin pérdida de información ni de control por parte del coordinador académico. La apertura automática del modal de asignación de áreas al registrar un docente y la carga automática de horarios previamente confirmados al volver al módulo de generación son dos decisiones de diseño que surgieron durante el desarrollo y no estaban en el plan original, pero que mejoraron significativamente la usabilidad del sistema según la retroalimentación del coordinador académico durante la validación.

3.4.2 Discusión sobre el desempeño del algoritmo genético

Los valores de fitness obtenidos (94.4% – 99.0% del máximo teórico) en las ocho instancias evaluadas son consistentes con los rangos reportados por Yusop et al. (2022) y Ceschia et al. (2023) para problemas de timetabling universitario resueltos mediante metaheurísticas (80-95%). Esto confirma que la configuración paramétrica adoptada 100 individuos, 200 generaciones, cruce cxTwoPoint con probabilidad 0.7 y mutación con probabilidad 0.3 produce resultados comparables a los reportados en la literatura para problemas de escala similar. La variabilidad observada entre ejecuciones independientes (desviación estándar de 170.15 sobre un rango de 450 puntos) es inherente a la naturaleza estocástica de los algoritmos genéticos y no afecta la factibilidad de las soluciones: en

ninguna ejecución se detectaron cruces de docentes, aulas, bloques de materias y las penalizaciones se concentraron en la carga horaria docente fuera de rango. Esta variabilidad puede reducirse incrementando el tamaño de la población o el número de generaciones, a costa de mayor tiempo de cómputo, lo que constituye una línea de mejora futura. La estrategia de mutación con consistencia de docente por materia que propaga el cambio a todos los bloques de una misma asignatura y la penalización acumulativa entre paralelos generados en la misma sesión son contribuciones específicas de la implementación que no estaban presentes en los trabajos de referencia revisados, y que mejoran la calidad de las soluciones en contextos donde se programan múltiples paralelos de forma secuencial.

3.4.3 Discusión sobre limitaciones y trabajo futuro

La principal limitación identificada durante la validación es la imposibilidad de medir la reducción de conflictos respecto a programaciones manuales previas, dado que el período validado (septiembre 2026 – febrero 2027) corresponde a la primera aplicación del sistema. Esta comparación podrá realizarse a partir del segundo ciclo de uso, cuando exista historial de programaciones generadas por el sistema. Como líneas de trabajo futuro se identifican: incorporar la restricción de horarios bloqueados por franjas específicas (ej. viernes 10:00-13:00) como parámetro configurable por el coordinador; ampliar el módulo de bibliografía académica con notificaciones automáticas a docentes sobre plazos de carga; y evaluar el desempeño del algoritmo sobre instancias de mayor escala que incluyan todas las carreras de la Escuela de Hábitat simultáneamente.

CONCLUSIONES

A continuación, se presentan las conclusiones obtenidas luego de haber completado el presente trabajo de investigación:

• El análisis del proceso actual de programación académica de la Escuela de Hábitat, Infraestructura y Creatividad permitió identificar las restricciones duras y blandas del problema, lo que hizo posible formalizar los criterios de optimización que el algoritmo genético debía cumplir. Sin este levantamiento, no habría sido posible diseñar una función de aptitud alineada con la normativa institucional.

• El algoritmo genético desarrollado generó sugerencias de horarios académicos sin cruces de docentes, aulas ni estudiantes en las ocho instancias evaluadas, con valores de fitness entre el 95.6% y el 99.0% del máximo teórico y un tiempo de ejecución de aproximadamente un segundo. Fuenmayor et al. (2022), empleando igualmente un algoritmo genético con DEAP en un problema de mayor escala, reportan tiempos de hasta cinco minutos, lo que confirma que ambos resultados son consistentes con el tamaño de cada problema y que el tiempo de respuesta del sistema desarrollado es adecuado para uso interactivo institucional.

• La arquitectura desacoplada implementada, con un módulo de gestión académica en Vue.js 3 y NestJS y un motor de optimización en Python con DEAP, demostró ser viable para integrar ambos componentes mediante servicios API REST. Los cinco módulos del sistema autenticación, gestión académica, gestión de recursos, programación académica y administración fueron implementados y validados correctamente.

• La validación del sistema mediante seis pruebas de aceptación funcionales verificó veintidós escenarios de éxito y error, confirmando el cumplimiento de los criterios definidos en las historias de usuario. Las métricas cuantitativas del algoritmo genético mostraron una media de fitness de 9710.0 con desviación estándar de 170.15 sobre cinco ejecuciones independientes de la misma instancia, sin cruces de docentes, aulas ni estudiantes en ningún caso evaluado.

• En conjunto, el sistema web inteligente desarrollado demostró la viabilidad de reemplazar el proceso manual basado en hojas de cálculo Excel desconectadas por una plataforma centralizada capaz de generar, en segundos, programaciones académicas que satisfacen las restricciones críticas de factibilidad de la PUCE Sede Ibarra, cumpliendo con el objetivo general planteado.

RECOMENDACIONES

Con base en los resultados obtenidos y las limitaciones identificadas, se plantean las siguientes recomendaciones:

• Conservar las programaciones generadas por el sistema en los periodos subsecuentes, de modo que puedan servir como línea base para evaluar la reducción de conflictos en futuras comparaciones, métrica que no pudo calcularse en el presente trabajo por tratarse de la primera programación realizada con el sistema.

• Evaluar la incorporación de un operador de mutación dirigida por aptitud individual, siguiendo el enfoque de Hussein y Albayati (2020), con el fin de reducir la variabilidad entre ejecuciones y mejorar la convergencia del algoritmo sin incrementar el número de generaciones.

• Extender el sistema a otras escuelas y carreras de la PUCE Sede Ibarra, aprovechando la arquitectura modular implementada, que permite configurar nuevas unidades académicas, planes de estudio y docentes sin modificar el núcleo del motor de optimización.

• Incorporar preferencias horarias de los docentes como restricción blanda adicional en la función de aptitud, de manera que el algoritmo genético pueda considerar la disponibilidad declarada por cada docente al generar las sugerencias de programación.

REFERENCIAS BIBLIOGRÁFICAS

Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, 43-59. https://www.sciencedirect.com/science/article/abs/pii/S0360835214002745?via%3Dihub

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in Python. *IEEE Access*, *8*, 89497-89509. https://ieeexplore.ieee.org/document/9078759

Campoverde Ramos, H. O. (2023). Sistema de gestión de horarios académicos para la Facultad de Ciencias Físicas y Matemática de la Universidad Central del Ecuador [Trabajo de titulación, Universidad Central del Ecuador]. Repositorio Digital UCE. http://www.dspace.uce.edu.ec/handle/25000/30890

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2023). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, *308*(1), 1-18. https://www.sciencedirect.com/science/article/pii/S0377221722005641?via%3Dihub

Chen, M. C., Goh, S. L., Sabar, N. R., & Kendall, G. (2020). A tabu search algorithm with controlled randomization for constructing feasible university course timetables. Computers & Operations Research, 123, 105007. https://www.sciencedirect.com/science/article/abs/pii/S0305054820301246?via%3Dihub

Davison, M., & Drake, J. H. (2024). Modelling and solving the university course timetabling problem with hybrid teaching considerations. Journal of Scheduling, 27(5), 542-558. https://link.springer.com/article/10.1007/s10951-024-00817-w

De la Rosa-Martín, T., & León-González, J. L. (2024). Sistema web para la creación de horarios de clases en la Universidad Metropolitana del Ecuador. Revista Metropolitana de Ciencias Aplicadas, 7(Suplemento 1), 38-48. https://remca.umet.edu.ec/index.php/REMCA/article/view/698

Fuenmayor, R., Larrea, M., Moncayo, M., Moya, E., Trujillo, S., Terneus, J., Guachi, R., Peluffo-Ordoñez, D., & Guachi-Guachi, L. (2022). A genetic algorithm for scheduling laboratory rooms: A case study. Applied Sciences, 12(20), 10336. https://link.springer.com/chapter/10.1007/978-3-031-19647-8_1

Gad, A. F. (2024). PyGAD: An intuitive genetic algorithm Python library. *Multimedia Tools and Applications*, *83*, 58029-58042. https://link.springer.com/article/10.1007/s11042-023-17167-y

Hussein, W. N., & Albayati, S. M. (2020). Smart university scheduling using genetic algorithms. En Proceedings of the 9th International Conference on Software and Information Engineering (pp. 166-171). ACM. https://dl.acm.org/doi/10.1145/3436829.3436873

Kim, J., & Yoo, S. (2019). Software review: DEAP (Distributed Evolutionary Algorithm in Python) library. Genetic Programming and Evolvable Machines, 20(1), 139-142. https://link.springer.com/article/10.1007/s10710-018-9341-4

Mallari, C. B., San Juan, J. L., & Li, R. (2023). The university coursework timetabling problem: An optimization approach to synchronizing course calendars. Computers & Industrial Engineering, 184, 109561. https://www.sciencedirect.com/science/article/abs/pii/S0360835223005855?via%3Dihub

Mitchell, M. (1998). An introduction to genetic algorithms. MIT Press.

Naaman, D. W., Ahmed, B. T., & Ibrahim, I. M. (2025). Optimization by nature: A review of genetic algorithm techniques. The Indonesian J ournal of Computer Science, 14(1), 268-284. https://www.ijcs.net/ijcs/index.php/ijcs/article/view/4596

Nguyen, T. M., Nguyen, T. T., & Quan, T. T. (2025). A two-stage fuzzy-guided genetic algorithm for university timetabling with LLM-based preference parsing. En T. T. Quan, C. Sombattheera, H. A. Pham, & N. T. Tran (Eds.), Multi-disciplinary trends in artificial intelligence. MIWAI 2025. Lecture Notes in Computer Science (Vol. 16354, pp. 341-353). Springer. https://link.springer.com/chapter/10.1007/978-981-95-4960-3_29

Pratama, A., Alam, E., & Suakanto, S. (2024). Optimizing practical room scheduling using genetic algorithms: A timetabling case study at Telkom University. En 2024 4th International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS) (pp. 426-431). IEEE. https://ieeexplore.ieee.org/document/10776011

Soria-Alcaraz, J. A., Carpio-Valadez, M., & Puga, H. (2023). Meta-heuristic approaches for the university course timetabling problem. Decision Analytics Journal, 7, 100250. https://www.sciencedirect.com/science/article/pii/S2772662223000905?via%3Dihub

Yusop, N., Wahab, M. H., & Mohsin, S. S. M. (2022). A systematic literature review: Optimization timetable in educational institutions to support work-life balance. Journal of Computational Research and Innovation, 7(3), 45-62. https://jcrinn.com/index.php/jcrinn/article/view/324

ANEXOS

. Carta de recepción de la propuesta tecnológica por parte del cliente

Grupo de Investigación en Sistemas Inteligentes

Pontificia Universidad Católica del Ecuador
Seréis mis testigos | IBARRA

A quien pueda Interesar

Por medio de la presente, yo Dulce Milagro Rivero como Líder del Grupo de Investigación en Sistemas Inteligentes (GISI) confirmo que he revisado el Trabajo de Titulación titulado " Sistema web inteligente para la optimización de la programación académica y la asignación dinámica de recursos, docentes y horarios ", elaborado por el estudiante Hermosa Ponce Adrián Renato como requisito previo para la obtención del título de Ingeniero en Tecnologías de la Información en la PUCE-SI.

Una vez revisado el trabajo de grado, se ha verificado que el sistema cumple satisfactoriamente con los objetivos establecidos en el proyecto.

Sin otro particular,

Quedo atenta a cualquier consulta adicional.

Dulce Milagro Rivero Albarrán

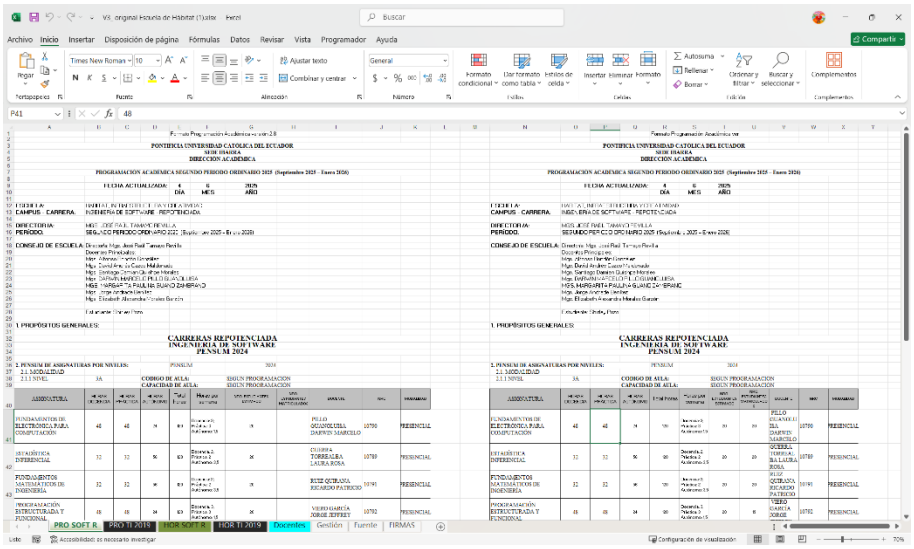
Firmado digitalmente por Dulce Milagro Rivero Albarrán
Fecha: 2026.07.17 13:08:06 -05'00'

Dra. Dulce Milagro Rivero
Líder del Grupo de Investigación GISI
CI. 1757608961

Anexo A: Formato Institucional de Programación Académica

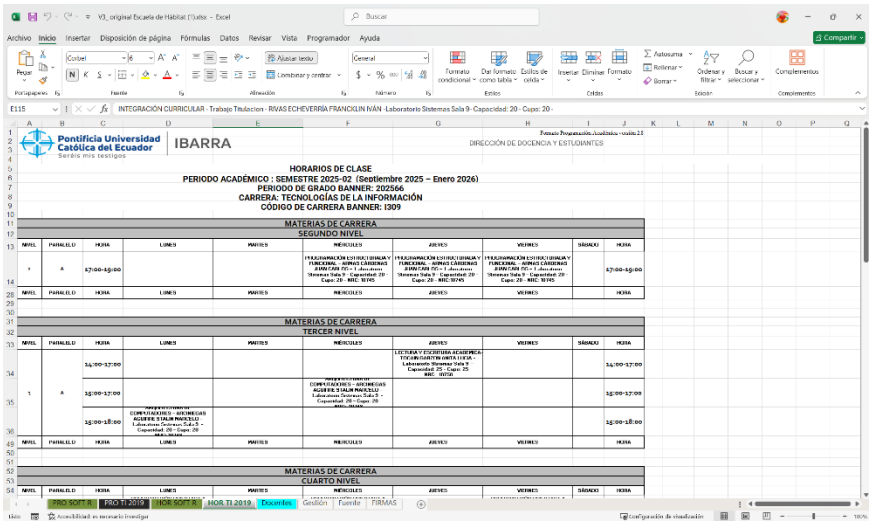
Descripción: Formato en Excel utilizado actualmente por la Escuela de Hábitat, Infraestructura y Creatividad de PUCE Sede Ibarra para la programación manual de horarios académicos. Este instrumento fue analizado mediante observación directa para identificar el flujo de trabajo actual y los requisitos funcionales del sistema automatizado.

Figura A.1. Hoja de Programación Académica



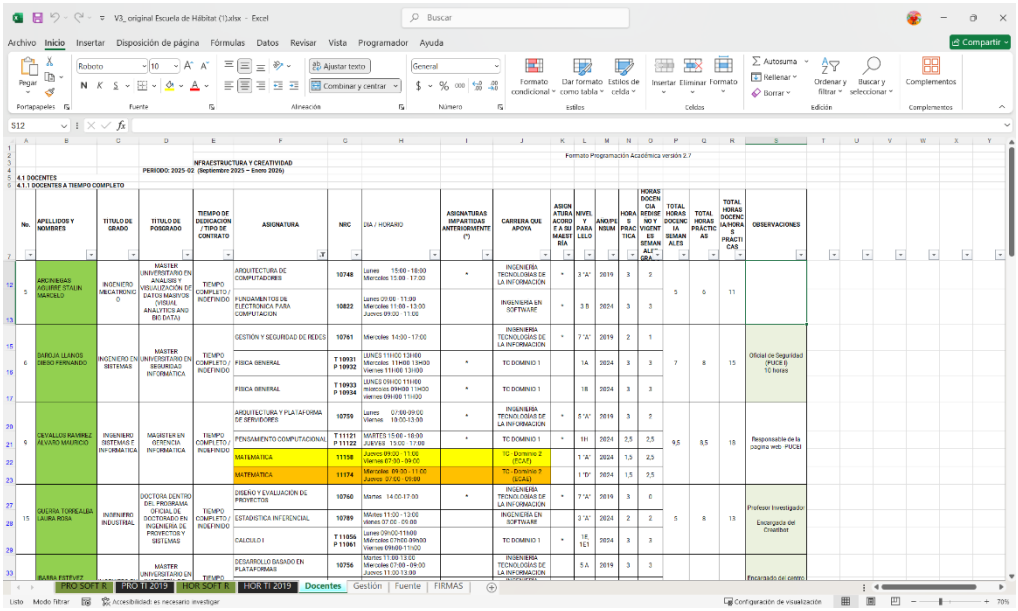
Nota. Formato institucional versión 2.8 utilizado para programar materias, docentes y horarios de la carrera de Ingeniería de Software.

Figura A.2. Hoja de Horarios



Nota. Grilla de horarios semanal con asignaciones de materias, docentes y aulas.

Figura A.3. Hoja de Docentes



Nota. Listado de docentes con categoría, dedicación, títulos.

Anexo B: Guión de Entrevista Semiestructurada

OBJETIVO: Identificar requisitos funcionales del sistema de optimización de horarios académicos.

ENTREVISTADO: Coordinador académico de la carrera de Ingeniería de Software

FECHA: 19 de marzo del 2026

PREGUNTAS GUÍA:

1. Proceso actual de programación

- ¿Cómo se realiza actualmente la programación de horarios?

- ¿Qué herramientas utiliza?

- ¿Cuánto tiempo le toma el proceso?

2. Restricciones institucionales

- ¿Qué restricciones normativas deben cumplirse?

- ¿Cuáles son los límites de carga horaria por categoría docente?

- ¿Existen restricciones de distribución entre jornadas?

3. Problemas identificados

- ¿Qué problemas recurrentes encuentra en el proceso manual?

- ¿Qué conflictos aparecen con mayor frecuencia?

4. Expectativas del sistema

- ¿Qué funcionalidades esperaría del sistema automatizado?

- ¿Qué información debe visualizar?

- ¿Qué reportes necesita generar?

5. Proceso de validación

- ¿Cómo validaría que un horario generado es correcto?

- ¿Qué ajustes manuales necesitaría poder realizar?