UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Plataforma de generación de planes de cursado para estudiantes de Facultad de Ingeniería

Informe de Proyecto de Grado presentado por

Agustina Pérez, Mathias Ramilo y Florencia Artegoytia

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Martín Pedemonte
Manuel Freire

Montevideo, 1 de octubre de 2025

Agradecimientos

En primer lugar, queremos agradecer a nuestros tutores y docentes por su guía, paciencia y dedicación durante todo el proceso de este proyecto. Sus aportes y sugerencias fueron fundamentales para poder llevar adelante este trabajo.

A la Facultad de Ingeniería y al Instituto de Computación, por brindarnos el espacio académico y los recursos necesarios que hicieron posible la realización de este proyecto.

También agradecemos a nuestros compañeros y compañeras de carrera, quienes con sus comentarios, apoyo y discusiones contribuyeron a mejorar y enriquecer esta propuesta.

De manera muy especial, queremos expresar nuestra gratitud a nuestras familias y amistades, por acompañarnos incondicionalmente, por su apoyo constante y por brindarnos ánimo en cada etapa del camino.

Finalmente, a todas las personas que, de una forma u otra, contribuyeron a que este trabajo fuera posible: muchas gracias.

Resumen

La planificación de la carrera en la Facultad de Ingeniería presenta desafíos significativos debido a la variedad de trayectorias posibles, los requisitos de avance entre unidades curriculares y la dispersión de la información académica. Las herramientas actuales no permiten visualizar de forma integrada el progreso del estudiante ni anticipar planes de cursado viables, lo que dificulta la toma de decisiones tanto para estudiantes como para personal de orientación.

Este proyecto propone una plataforma web que centraliza y organiza la información académica relevante, permitiendo generar planes de cursado personalizados que respetan las reglas de previaturas, la oferta semestral y las preferencias del usuario. A través de una interfaz interactiva, los usuarios pueden visualizar su avance, explorar unidades curriculares disponibles y simular distintas trayectorias posibles.

El sistema se basa en una arquitectura modular compuesta por tres componentes desacoplados: un backend que implementa la lógica central y los algoritmos de generación de trayectorias; un frontend que ofrece una experiencia de usuario accesible y visualmente clara; y un microservicio auxiliar que colabora en el procesamiento de datos académicos. Los algoritmos principales están construidos sobre modelos de grafos y ordenamientos topológicos que garantizan la validez de los planes propuestos.

Los resultados obtenidos evidencian mejoras en la claridad, eficiencia y confiabilidad del proceso de planificación académica, ofreciendo una herramienta adaptable, escalable y de utilidad real para múltiples actores dentro del sistema educativo.

Palabras clave: Plataforma web, Planificación académica, Currícula personalizada, Web scraping, Algoritmo basado en grafos, Camino crítico, Scheduling

Índice general

Glosario

Este glosario reúne los términos clave utilizados a lo largo del documento, proporcionando definiciones adaptadas al contexto del proyecto y de la Facultad de Ingeniería. Su objetivo es garantizar una correcta comprensión de los conceptos académicos utilizados.

Unidad curricular: Asignatura incluida en el plan de estudios de una carrera, que otorga créditos al ser aprobada.

Unidades curriculares obligatorias: Asignaturas que todos los estudiantes deben aprobar para completar la carrera.

Unidades curriculares optativas: Asignaturas dictadas por la Facultad de Ingeniería que el estudiante puede elegir cursar y suman créditos para completar el plan de estudios.

Unidades curriculares electivas: Asignaturas dictadas por otras facultades que el estudiante puede elegir cursar y suman créditos para completar el plan de estudios.

Créditos: Unidad de medida que cuantifican la carga académica de una unidad curricular. Se utilizan para evaluar el progreso académico y el cumplimiento de requisitos de la carrera.

Grupo: Conjunto de unidades curriculares agrupadas por área de formación (por ejemplo, Matemática, Programación o Ingeniería de Software).

Previatura: Condición o requisito que debe cumplirse antes de cursar una determinada unidad curricular. Puede implicar la aprobación de otras unidades curriculares o la obtención de un mínimo de créditos.

Sistema de previaturas: Conjunto de reglas y condiciones que determinan si un estudiante está habilitado para cursar una unidad curricular.

Capítulo 1

Introducción

Este capítulo tiene como propósito contextualizar el problema abordado, motivar su estudio y definir los objetivos perseguidos por el presente proyecto de grado. En particular, se describe la situación actual en relación con la planificación académica en carreras de ingeniería, las limitaciones de las soluciones existentes, y la motivación que dio origen al desarrollo de una plataforma digital que automatiza y personaliza la construcción de trayectorias curriculares. Asimismo, se presentan los resultados alcanzados y una visión general de la solución propuesta, sentando las bases para los capítulos subsiguientes, donde se detallan los componentes técnicos, algoritmos y decisiones de diseño implementadas en el sistema.

1.1. Contexto y motivación

En el ámbito universitario, la planificación del cursado representa un desafío recurrente para los estudiantes, en particular en carreras técnicas con estructuras curriculares complejas como la de Ingeniería. La existencia de múltiples restricciones, como los requisitos de aprobación de unidades curriculares previas, la oferta semestral de unidades curriculares y la carga académica recomendada por período, dificulta la elaboración de un plan de cursado.

Actualmente, esta planificación suele realizarse de forma manual, mediante una currícula sugerida que distribuye las unidades curriculares a lo largo de los semestres. Cuando el estudiante comienza su carrera, es habitual que no siga la trayectoria sugerida de forma exacta: puede no aprobar todas las asignaturas del semestre, optar por no cursar alguna unidad curricular o modificar el orden de cursado. Esto obliga a replanificar en función de las previaturas para evitar bloqueos futuros. Ante esta realidad, surge la motivación de desarrollar una herramienta que acompañe al estudiante en este proceso, brindándole una planificación personalizada y dinámica, que tenga en cuenta lo que ya cursó, las unidades curriculares disponibles, las correlatividades y sus propios objetivos en cuanto a carga horaria disponible.

Durante el desarrollo del proyecto, y a partir de la investigación de la problemática, se notó que este desafío no es exclusivo de los estudiantes. Directores de carreras y personas que orientan académicamente también se enfrentan frecuentemente a las dificultades de planificar trayectorias curriculares viables y ajustadas a cada caso particular. El espacio de orientacion y consulta (EOC) asesora a estudiantes de todas las carreras de la Facultad, lo que implica enfrentar una gran variedad de casuísticas. Los directores de carrera, en cambio, suelen atender estudiantes avanzados o casos particulares y poseen un conocimiento más profundo del plan de estudios correspondiente. Por eso, la herramienta fue pensada no solo para ser usada directamente por estudiantes, sino también como un apoyo para quienes los asisten en su planificación. Los requisitos funcionales y no funcionales fueron definidos considerando entrevistas realizadas con este tipo de actores, además de nuestra propia experiencia como estudiantes de la carrera.

El presente proyecto consiste en el desarrollo de una plataforma web que permite a los estudiantes de la Facultad de Ingeniería planificar el cursado su carrera de manera interactiva. Para ello, se genera un grafo que representa las unidades curriculares y calcula un camino crítico que optimiza el recorrido curricular según la carga académica (cantidad de créditos) que el usuario desea asumir por semestre.

1.2. Objetivos

Diseñar e implementar una herramienta que genere una planificación académica personalizada a partir de la escolaridad del estudiante y el sistema de previaturas de la carrera.

Objetivos específicos:

- Facilitar la planificación académica considerando las dependencias entre unidades curriculares y la oferta semestral.
- Permitir que los usuarios definan restricciones personalizadas, como la carga máxima de créditos por semestre.
- Ofrecer una visualización clara del recorrido curricular faltante y la manera óptima de cursarlo.
- Brindar soporte tanto a estudiantes como a referentes académicos en la planificación de su carrera.
- Promover una toma de decisiones más informada respecto a la inscripción y organización del cursado.

Este proyecto busca brindar una herramienta de apoyo para los distintos actores mencionados, mejorando la toma de decisiones académicas y contribuyendo a optimizar la trayectoria de la carrera.

Cabe aclarar que, si bien la problemática y la motivación abarcan a todas las carreras de la Facultad de Ingeniería, en esta etapa del proyecto se trabajó exclusivamente con el plan de estudios de Ingeniería en Computación, por razones de viabilidad y tiempo. El sistema fue diseñado como un prototipo extensible al resto de las carreras, lo que implica principalmente incorporar las previaturas y cursos correspondientes, así como ajustar ciertos casos especiales. Por ejemplo, en Ingeniería en Computación no existen perfiles, mientras que otras carreras, como Ingeniería en Eléctrica, sería necesario contemplar esta lógica en el sistema. Más allá de estos ajustes, los problemas técnicos centrales ya fueron resueltos, por lo que la ampliación sería mayormente un trabajo de incorporación de datos y adaptaciones menores.

Capítulo 2

Marco conceptual

En este capítulo se describen los elementos que fundamentan el desarrollo del proyecto, tanto desde el punto de vista institucional como técnico. En la seccion 2.1, se analizan los recursos actuales que ofrece la Facultad de Ingeniería para la planificación académica, así como sus principales limitaciones. Luego, en la seccion 2.2, se presentan los conceptos teóricos necesarios para modelar el problema desde una perspectiva computacional, incluyendo estructuras de grafos y algoritmos asociados. Finalmente, en las secciones 2.3 y 2.4, se revisan trabajos previos y plataformas con objetivos similares, con el fin de identificar diferencias, aprendizajes y aportes específicos del sistema desarrollado. Este marco proporciona la base para las decisiones tomadas en las etapas de diseño e implementación.

2.1. Información institucional disponible

La planificación del cursado de una carrera universitaria, especialmente en un entorno complejo como el de la Facultad de Ingeniería, representa un desafío para muchos estudiantes. Desde los primeros semestres, en los que deben adaptarse al ritmo académico y a las exigencias de la formación profesional, hasta las etapas más avanzadas, donde entran en juego las restricciones del sistema de previaturas y la necesidad de tomar decisiones estratégicas sobre unidades curriculares obligatorias u optativas, la organización académica requiere un proceso de análisis y toma de decisiones cuidadoso.

En este contexto, esta sección describe los principales recursos institucionales que ofrece la Facultad para apoyar la planificación académica, así como las limitaciones que presentan dichos recursos. En los apartados 2.1.1 y 2.1.2 se detallan respectivamente los recursos disponibles actualmente y sus principales restricciones, las cuales motivan la necesidad de soluciones complementarias para facilitar la gestión académica.

2.1.1. Recursos disponibles actualmente

Para apoyar este proceso, la Facultad de Ingeniería pone a disposición ciertos recursos institucionales. Entre los principales se encuentran:

- **Trayectoria sugerida por semestre:** publicada en el sitio web institucional (Curricula Sugerida - Ingenieria en Computación, s.f.), que presenta una distribución semestral estática de las unidades curriculares recomendadas. Si bien esta trayectoria sirve como una guía de referencia general, su rigidez impide que se adapte a las particularidades de cada estudiante, como escolaridades irregulares (por ejemplo, cuando un estudiante pierde una unidad curricular que es un requisito previo de varias unidades curriculares en los semestres siguientes), períodos de inactividad, cambios de ritmo, o distintas disponibilidades horarias y de dedicación por semestre.
- **Sistema de previaturas:** accesible a través del portal de Bedelías (Sistema de previaturas de Ingeniería en Computación, s.f.), el cual detalla las relaciones de precedencia entre unidades curriculares. Esta información, aunque fundamental, no es trivial de interpretar, ya que muchas reglas de precedencia se expresan en estructuras lógicas anidadas (como conjunciones, disyunciones, requisitos de créditos por grupo o plan), dificultando su comprensión por parte del estudiante.
- **Información de unidades curriculares:** el portal de Bedelías brinda información individual sobre cada unidad curricular, como créditos otorgados, modalidad de cursado, y requisitos, pero su consulta no siempre resulta ágil para los estudiantes.
- **Evaluar previas:** el portal de Bedelías ofrece para los estudiantes una funcionalidad que permite seleccionar una unidad curricular y verificar automáticamente si el estudiante está habilitado para cursarla o rendirla, de acuerdo con el cumplimiento de sus previaturas y requisitos vigentes.

2.1.2. Limitaciones de los recursos actuales

Pese a su valor, estos recursos presentan limitaciones importantes. En primer lugar, la información que brindan es estática y general, sin capacidad de adaptarse automáticamente al progreso individual de cada estudiante. Esto implica que dos estudiantes con trayectorias distintas reciben la misma propuesta, independientemente de su avance real. Además, no ofrecen la posibilidad de explorar alternativas de planificación en función de preferencias personales, como la cantidad de créditos que se desea cursar por semestre o las áreas temáticas de mayor interés.

A ello se suma que, por la forma en que están definidas, las previaturas pueden resultar complejas de interpretar para quienes no están familiarizados con la estructura curricular. Las reglas de precedencia suelen incluir combinaciones lógicas y requisitos de créditos que no siempre son intuitivos, lo que dificulta

la lectura y comprensión de la información. En conjunto, estas limitaciones impactan especialmente en estudiantes con trayectorias irregulares, períodos de inactividad o situaciones particulares que requieren ajustes en la secuencia de cursado.

2.1.3. Apoyo institucional personalizado: el EOC y directores de carrera

Con el objetivo de cubrir necesidades que los recursos digitales no resuelven, la Facultad cuenta con el Espacio de Orientación y Consulta (EOC) y con directores de carrera. El EOC atiende principalmente a estudiantes de los primeros años, brindando asesoramiento individualizado y generalista, mientras que los directores de carrera suelen atender casos avanzados o particulares, con un conocimiento más específico de las particularidades de cada plan de estudios.

Sin embargo, esta tarea requiere un esfuerzo significativo. Analizar caso a caso implica recopilar y procesar manualmente la escolaridad del estudiante, comprender el estado de avance, aplicar las reglas de previaturas vigentes y proponer alternativas válidas para el cursado, lo cual puede resultar ineficiente y demandante en períodos de alta carga de trabajo.

Una limitación adicional del EOC es que su funcionamiento no resulta escalable. Actualmente opera de manera efectiva porque es utilizado por una fracción reducida de estudiantes, pero el incremento sostenido en el número de ingresos en los últimos años sugiere que, ante una mayor demanda, su capacidad de atención podría saturarse rápidamente.

2.1.4. Aportes del sistema propuesto

Este proyecto busca complementar y potenciar los recursos existentes, abordando sus limitaciones mediante una plataforma centralizada, interactiva y personalizable.

Para los estudiantes, la herramienta permite visualizar su avance académico, consultar trayectorias factibles y generar planes personalizados que consideran su progreso, disponibilidad horaria y preferencias. Por otro lado, para el EOC y los directores de carrera, facilita la carga rápida de escolaridades, mostrando de forma automática las unidades pendientes, las restricciones vigentes y caminos alternativos de cursado. Esto aporta autonomía al estudiante y eficiencia a la gestión institucional, promoviendo el trabajo conjunto entre estudiantes y orientadores.

De esta forma, la plataforma no solo empodera al estudiante con autonomía y claridad sobre su futuro académico, sino que también contribuye a una gestión institucional más eficiente y basada en datos estructurados, favoreciendo el trabajo colaborativo entre estudiantes y orientadores.

2.2. Conceptos teóricos utilizados

El desarrollo del sistema se apoya en una serie de conceptos teóricos que se describen a lo largo de este documento, los cuales permitieron abordar la problemática de planificación curricular. Estos conceptos, que fueron fundamentales para el diseño de la arquitectura como para la implementación de las funcionalidades clave del sistema, abarcan desde el modelado mediante estructuras de grafos y la aplicación de algoritmos clásicos, hasta técnicas de extracción automatizada de datos desde sitios web. A continuación, se describen las estructuras y métodos más relevantes, incluyendo sus definiciones, aplicaciones generales y su uso específico en este proyecto.

2.2.1. Grafo dirigido acíclico (DAG)

Un DAG (por su sigla en inglés: Directed Acyclic Graph) es una estructura de datos compuesta por nodos y aristas dirigidas, donde no existen ciclos [1].

Este tipo de grafos es utilizado para representar relaciones de dependencia, como flujos de tareas, versiones de archivos o, en el caso de este trabajo, prerequisitos académicos. En este proyecto, el DAG constituye la estructura base para representar las relaciones de precedencia entre unidades curriculares. Cada nodo representa una unidad curricular, y las aristas indican dependencias de cursado o examen. Esta representación permitió construir algoritmos eficientes que respeten las restricciones del plan de estudios.

2.2.2. Ordenamiento topológico

Es un algoritmo que, aplicado a un DAG, genera una secuencia lineal de los nodos tal que para cada arista dirigida de un nodo u a un nodo v, u aparece antes que v en la secuencia. Este procedimiento es comúnmente usado en planificación de proyectos, compiladores, y resolución de dependencias. En este caso, se aplicó el algoritmo de ordenamiento topológico DFS (búsqueda en profundidad) sobre el grafo de previaturas, como paso previo a la ejecución del algoritmo de camino crítico, garantizando que las dependencias se respeten.

2.2.3. Camino crítico

El análisis del camino crítico (Critical Path Method) es una técnica usada en gestión de proyectos para determinar la secuencia de tareas que define la duración mínima total. Permite identificar tareas críticas (sin holgura[2]) y optimizar recursos. En este caso, el algoritmo fue adaptado para identificar las unidades curriculares críticas dentro del plan de estudios del estudiante, es decir, aquellas que definen la duración mínima para cursado. Además, se calcularon fechas

[1] Ciclo: es un camino en un grafo que comienza y termina en el mismo vértice, con al menos una arista, y que no visita ningún otro vértice más de una vez (excepto el primero y el último).

[2] Holgura: tiempo que una actividad puede retrasarse sin afectar la fecha de finalización del proyecto ni el inicio de sus sucesoras.

de inicio más tempranas y holguras (en semestres) asociadas a cada unidad curricular, lo cual sirvió como base para decisiones de planificación curricular.

2.2.4. Algoritmo de scheduling

El scheduling o planificación consiste en asignar tareas a recursos en el tiempo, respetando restricciones de precedencia y capacidad. Se utiliza en contextos como planificación de producción, asignación de procesos en sistemas operativos y cronogramas académicos. En este proyecto se desarrolló un algoritmo de scheduling que, basándose en el grafo de dependencias, distribuye las unidades curriculares a lo largo de semestres. El algoritmo respeta restricciones de créditos máximos por semestre e intenta generar un plan de cursado equilibrado y factible para el estudiante, inspirándose en estrategias de programación greedy para asignar cada unidad curricular de manera secuencial respetando las restricciones.

2.2.5. Recursión

Es una técnica de programación donde una función se llama a sí misma para resolver subproblemas de forma jerárquica. Es frecuentemente utilizada en estructuras de árbol o problemas que requieren exploración exhaustiva. En este sistema, la recursión se utilizó principalmente en el procesamiento del sistema de previaturas. La estructura jerárquica y lógica de las reglas (AND, OR, NOT, SOME) fue representada como un árbol, donde cada nodo puede contener subreglas, y el cumplimiento de una previatura se evalúa recursivamente desde la raíz hacia los nodos hoja.

2.2.6. Web scraping

Es una técnica que consiste en extraer información de sitios web de forma automatizada, mediante scripts que simulan la navegación de un usuario. Se utiliza en minería de datos, monitoreo de precios, agregadores de contenido, entre otras. En el proyecto desarrollado, se aplicó web scraping sobre el portal de Bedelías de la Facultad de Ingeniería, con el objetivo de obtener datos estructurados sobre unidades curriculares, requisitos de previaturas, semestres de dictado y grupos disponibles. Esta información fue clave para el correcto funcionamiento del sistema y asegurar su actualización automática.

2.2.7. JSON

JSON (JavaScript Object Notation) es un formato ligero y flexible de intercambio de datos basado en texto, ampliamente utilizado por su legibilidad y compatibilidad con múltiples lenguajes de programación. JSON permite estructurar información jerárquica mediante pares clave-valor y arreglos, siendo especialmente útil para representar estructuras como árboles. En el proyecto desarrollado, JSON fue utilizado como método de persistencia y modelado del

sistema de previaturas de forma estructurada, transformando la información tabular extraída del portal web de Bedelias en una representación jerárquica que refleja la lógica interna del sistema. Esta elección permitió manipular y consultar eficientemente las condiciones de cursado de cada unidad curricular, además de facilitar la integración con el backend de la aplicación y su visualización en el frontend.

2.3. Revisión de antecedentes

Durante la etapa de análisis preliminar del proyecto, se realizó una revisión de trabajos previos con el objetivo de identificar enfoques existentes que pudieran servir como base o inspiración para el desarrollo de esta solución. En particular, existe un proyecto de grado anterior que abordó el modelado del sistema de previaturas para la carrera de Ingeniería en Computación, utilizando Prolog[3] como lenguaje para representar las restricciones entre unidades curriculares y créditos.

Este enfoque resulta interesante desde el punto de vista formal, ya que Prolog permite expresar relaciones lógicas y restricciones de manera declarativa, lo que puede ser adecuado para representar grafos de dependencias como el de previaturas. Sin embargo, al profundizar en su estructura y dinámica, se identificaron limitaciones importantes en términos de acoplamiento, complejidad del modelo y dificultad de integración con los objetivos funcionales y técnicos de este proyecto. En particular, la representación en Prolog implicaba una curva de aprendizaje elevada y una escasa flexibilidad para integrarse con tecnologías modernas de desarrollo web como Python, TypeScript[4] o JSON, fundamentales en nuestra arquitectura.

Por otro lado, la estructura de dicho proyecto estaba fuertemente ligada a un motor lógico de ejecución, lo cual dificultaba su reutilización o extensión. Además, cualquier cambio en las previaturas requería modificar directamente el código en Prolog, lo que implicaba un alto esfuerzo de mantenimiento.

Debido a estas consideraciones, se optó por descartar dicho enfoque y diseñar desde cero una solución más flexible, extensible y alineada con los requisitos funcionales y no funcionales definidos en este documento. Para esto, se desarrolló una estructura propia de representación del sistema de previaturas basada en un formato JSON jerárquico, que permite representar reglas lógicas complejas (AND, OR, NOT, SOME), así como requisitos por créditos y cursos aprobados, de forma clara y fácilmente procesable en entornos web modernos. Esta solución será descrita en detalle en las secciones correspondientes a diseño e implementación.

[3] Prolog (o PROLOG), proveniente del francés PROgrammation en LOGique, es un lenguaje de programación lógico e interpretado.

[4] Python y TypeScript: lenguajes de programación.

2.4. Acercamientos similares

Existen diversas plataformas que ofrecen funcionalidades relacionadas con la planificación académica, destacando particularmente **MiCarrera**[5]. Esta plataforma permite marcar el progreso académico, buscar información detallada sobre unidades curriculares y grupos, y mantener un seguimiento activo del avance académico del estudiante. Sin embargo, la principal diferencia con nuestra propuesta radica en el enfoque central. Mientras que MiCarrera se centra en la gestión y seguimiento del progreso académico, la plataforma elaborada en este proyecto permite planificar a mediano y largo plazo, generando planes completos adaptados al estudiante.

Las funcionalidades comunes, como el marcado de unidades curriculares aprobadas y la consulta de información sobre las mismas, son esenciales también para nuestra aplicación, ya que constituyen la base necesaria para la generación de planes efectivos. Además, nuestra solución presenta una ventaja adicional relevante: la posibilidad de cargar automáticamente una escolaridad como punto de partida. Esta funcionalidad es particularmente útil para directores de carrera y EOC, ya que agiliza el proceso evitando el marcado manual de unidades curriculares.

Asimismo, otras iniciativas similares como **Materias-computacion**[6] y **Trayectoria**[7] comparten algunas funcionalidades con nuestra propuesta, pero igualmente carecen del objetivo central y específico que aporta el sistema desarrollado: la creación personalizada y dinámica de planes académicos adaptados a las circunstancias individuales de cada estudiante. En conclusión, nuestra plataforma complementa y adopta un enfoque diferente con el fin de proporcionar herramientas avanzadas de planificación curricular enfocadas directamente en optimizar la trayectoria académica.

[5] MiCarrera: https://micarrera.uy/
[6] Materias-computacion: https://materias-computacion.netlify.app/
[7] Trayectoria: https://trayectoria.fely.dev/

Capítulo 3

Relevamiento de requisitos

En este capítulo se presenta el proceso seguido para identificar, especificar y validar los requisitos del sistema. El objetivo de esta etapa fue garantizar que la solución responda efectivamente a las necesidades reales de los distintos actores involucrados en la planificación académica.

El relevamiento se basó en entrevistas semiestructuradas y exploración de herramientas existentes, lo cual permitió comprender el contexto institucional, caracterizar los procesos actuales y detectar problemáticas comunes. A partir de esta instancia, se identificaron los perfiles de usuarios relevantes y se analizaron sus necesidades específicas.

Se detallan a continuación los actores clave, junto con la especificación de los requisitos funcionales y no funcionales priorizados según su impacto en el sistema. Finalmente, se incluye una discusión general sobre los hallazgos obtenidos, los cuales guiaron decisiones de diseño y arquitectura en las etapas posteriores del desarrollo.

3.1. Metodología

Para la recolección de requisitos, se utilizó una metodología basada en entrevistas semiestructuradas y exploratorias. Las entrevistas se realizaron de manera remota con actores clave vinculados al funcionamiento y orientación académica de la carrera, entre ellos: directores de las carreras de Computación y Eléctrica, personal del espacio de orientación y consulta, y un estudiante egresado que desarrolló un proyecto similar como parte de su trabajo de grado.

El objetivo de estas entrevistas fue comprender el contexto institucional, identificar las principales dificultades que enfrentan los estudiantes al planificar su trayecto académico, y relevar necesidades específicas que una herramienta digital podría abordar. Las entrevistas también permitieron captar experiencias previas, como intentos anteriores de automatizar parte del proceso de planificación académica.

Para cada perfil entrevistado se elaboró una guía de preguntas orientada a

su rol particular y al tipo de información que podía aportar. Se consultó sobre escenarios frecuentes, dificultades al interpretar o aplicar las reglas de previas, requerimientos para la elaboración de un perfil de egreso, y las funcionalidades que considerarían más valiosas para una herramienta de apoyo académico.

3.2. Actores identificados

Durante el proceso de relevamiento se identificaron los principales actores involucrados en el uso y desarrollo del sistema. Estos actores representan distintos perfiles de usuarios o partes interesadas que interactuarán directa o indirectamente con la plataforma. A continuación, se presentan en detalle los actores clave, su rol y las formas en que el sistema busca dar respuesta a sus necesidades.

3.2.1. Estudiantes

Los estudiantes son los usuarios principales y destinatarios finales de la plataforma. En la actualidad, enfrentan diversas dificultades al organizar su trayectoria académica: la información institucional se encuentra dispersa en múltiples fuentes (como el sitio web de la Facultad de Ingeniería, Bedelías y EVA), las currículas sugeridas no consideran casos reales de avances irregulares (están diseñadas considerando una trayectoria estudiantil que va al día con la carrera), y la planificación personalizada requiere un esfuerzo significativo.

El sistema tiene como objetivo central asistir a los estudiantes en la generación de planes de cursado adaptados a sus características individuales. Para ello, debe ofrecer las siguientes funcionalidades:

- Carga automática de escolaridades en PDF para importar el avance académico.
- Visualización del estado actual del estudiante en el plan de estudios, indicando asignaturas aprobadas y créditos faltantes.
- Generación de trayectorias académicas sugeridas ajustadas a su ritmo deseado.
- Verificación automática del cumplimiento de requisitos para egreso.

Al brindar una herramienta intuitiva, interactiva y basada en su propio avance, se espera reducir la incertidumbre, minimizar errores de planificación y fortalecer al estudiante en su proceso formativo.

3.2.2. Espacio de Orientación y Consulta

El EOC cumple un rol fundamental en el acompañamiento de los estudiantes, especialmente durante los primeros años de carrera. En la actualidad, su tarea de

orientación demanda un importante esfuerzo de interpretación de trayectorias académicas, lo cual puede ser ineficiente y susceptible a errores.

La plataforma debe ofrecer al EOC herramientas que permiten:

- Visualizar y analizar trayectorias completas de estudiantes a partir de su escolaridad.
- Detectar posibles caminos alternativos de cursado.
- Sugerir estrategias de planificación académica personalizadas y realistas.
- Reducir la carga operativa vinculada a la lectura manual de escolaridades.

De esta forma, el EOC puede concentrarse en aspectos más cualitativos del acompañamiento, mientras delega en la plataforma tareas operativas y repetitivas.

3.2.3. Directores de carrera

Los directores de carrera son responsables de la gestión académica de los distintos planes de estudio ofrecidos por la Facultad. Entre sus tareas se encuentra el análisis de solicitudes de excepciones (por ejemplo, levantado de previas, que permite a un estudiante cursar una unidad curricular aunque no cumpla con los requisitos previos), el seguimiento de cohortes y el asesoramiento a estudiantes en situaciones especiales.

El sistema facilita el trabajo de los directores mediante:

- Generación automatizada de informes académicos por estudiante, exportables en formatos legibles.
- Comprobación automática del cumplimiento de requisitos para cursar o rendir asignaturas.
- Visualización estructurada del progreso del estudiante, incluyendo unidades aprobadas y créditos pendientes.
- Posibilidad de evaluar escenarios hipotéticos al levantar una previatura, lo cual agiliza la toma de decisiones.

Esto permite optimizar tiempos, reducir errores humanos en el análisis y tomar decisiones basadas en datos objetivos.

3.3. Requisitos funcionales

Para guiar el desarrollo de la aplicación, se definieron distintos requerimientos funcionales escritos como historias de usuario y fueron categorizados según su nivel de prioridad, se clasificó los requerimientos en tres niveles de prioridad:

alta, media y baja. Esta clasificación permite enfocar los esfuerzos en las funcionalidades más relevantes para el funcionamiento inicial del sistema, sin perder de vista futuras mejoras.

A continuacion, se listan los requisitos funcionales que fueron implementados en esta versión. Para una descripción detallada de las historias de usuario, ver Anexo A.

3.3.1. Alta prioridad

- **Marcado manual de unidades curriculares aprobadas:** Los usuarios deben poder marcar manualmente las asignaturas que han aprobado, de forma que la aplicación pueda generar propuestas de planificación ajustadas a su progreso real.
- **Exportación de unidades curriculares aprobadas:** La aplicación debe permitir generar un archivo (por ejemplo, PDF o CSV) con las asignaturas aprobadas, para facilitar su presentación ante funcionarios de la facultad.
- **Visualización de información académica:** Al consultar una unidad curricular, el usuario debe poder ver información relevante como su código, créditos otorgados, requisitos previos y, si está disponible, el enlace a EVA.
- **Currícula sugerida para toda la carrera:** También debe estar disponible una propuesta de cursado de largo plazo que indique qué unidades curriculares tomar en cada semestre restante, adaptada a la situación académica del usuario.
- **Importación de unidades curriculares aprobadas:** Los usuarios deben poder cargar un archivo con sus asignaturas aprobadas para que el sistema las registre y utilice como base para la planificación.

3.3.2. Media prioridad

- **Subida de escolaridad en PDF:** Los usuarios deben poder subir su escolaridad en formato PDF para que el sistema registre automáticamente las unidades curriculares aprobadas.
- **Persistencia local de datos:** La aplicación debe guardar los datos del usuario localmente en el navegador, de forma que no se pierdan al cerrar o recargar la página.
- **Consulta de unidades curriculares del próximo semestre:** El sistema debe permitir ver las unidades curriculares que se dictarán en el siguiente semestre, con su información básica y filtros por créditos.
- **Rango de créditos por semestre:** El usuario debe poder definir un rango de créditos que desea cursar por semestre, y el sistema debe considerar esa preferencia al generar las propuestas de planificación.

- **Verificación de requisitos para egreso:** El sistema debe verificar si el conjunto de unidades curriculares aprobadas por el usuario cumple con los requisitos necesarios para recibirse, mostrando qué falta en caso contrario.

3.4. Requisitos no funcionales

Los requisitos no funcionales definen criterios de calidad que el sistema debe cumplir más allá de sus funcionalidades. Incluyen aspectos clave como la usabilidad, seguridad, compatibilidad y mantenibilidad. Aunque no describen comportamientos observables directos, estos requisitos son determinantes para la adopción, escalabilidad y sostenibilidad del sistema.

3.4.1. Usabilidad

- **Interfaz centrada en el usuario:** La plataforma debe ser intuitiva, clara y accesible para estudiantes con distintos niveles de familiaridad tecnológica. Se aplicarán principios de diseño UX que aseguren navegación fluida, retroalimentación inmediata y jerarquía visual coherente.
- **Compatibilidad con navegadores modernos:** El sistema debe funcionar correctamente en al menos las tres versiones estables más recientes de Chrome, Firefox, Safari y Edge.

3.4.2. Compatibilidad e interoperabilidad

- **Integración futura con sistemas institucionales:** La arquitectura debe permitir integraciones con plataformas como el SGAE, mediante el uso de APIs RESTful, formatos abiertos (JSON) y principios de modularidad.
- **Soporte a múltiples planes de estudio:** El sistema debe ser capaz de representar distintas versiones de planes de estudio sin necesidad de rediseño, modelando dinámicamente previaturas, perfiles y requisitos por área.

3.4.3. Mantenibilidad

- **Código limpio y modular:** El desarrollo debe seguir buenas prácticas de ingeniería (principios SOLID, separación de responsabilidades, convenciones de estilo). Se deben utilizar herramientas de linting y análisis estático.
- **Documentación técnica:** Cada componente del sistema debe contar con documentación clara: endpoints, diagramas de arquitectura, guías de instalación y mantenimiento.
- **Testing automatizado:** El sistema debe contar con pruebas automatizadas para la lógica interna (unit tests), permitiendo validación continua y facilitando futuros cambios sin introducir errores.

Capítulo 4

Diseño e implementación

Este capítulo describe en profundidad la aplicación desarrollada durante el proyecto de grado, abordando las decisiones técnicas adoptadas y el diseño arquitectónico del sistema. Se presenta una visión integral de la solución implementada, detallando la composición y responsabilidades de cada uno de los módulos que la integran, así como las tecnologías empleadas para su construcción.

A lo largo del capítulo se analizan los aspectos clave de la arquitectura del sistema, incluyendo los mecanismos de comunicación entre componentes, la organización interna del backend, el microservicio de procesamiento y la interfaz web. Asimismo, se describe la infraestructura utilizada para el despliegue en producción, junto con las decisiones tomadas en términos de seguridad, rendimiento y mantenibilidad.

4.1. Descripción general de la solución

El sistema está diseñado para gestionar y planificar la trayectoria académica de estudiantes de la Facultad de Ingeniería, proporcionando una herramienta web que automatiza la carga y procesamiento de escolaridades. Además genera planes de cursado personalizados teniendo en cuenta prerrequisitos y el avance académico.

El sistema se compone de tres módulos principales que serán luego detallados a fondo en la sección de arquitectura del sistema:

- **Frontend:** desarrollado con *React* y *TypeScript*, es la interfaz con la que interactúan estudiantes y funcionarios. Permite cargar escolaridades en PDF, visualizar el avance académico y generar planes de estudio de forma interactiva.
- **Backend principal:** desarrollado con *Node.js* y *Express*, contiene la lógica de negocio, incluyendo la generación de planes de cursado, validación

de previaturas y gestión de datos académicos. Expone una API REST consumida por el frontend y el microservicio.

- **Microservicio de scraping y procesamiento:** desarrollado con *Python* y *FastAPI*, se encarga de extraer información académica desde el portal institucional (web scraping) y procesar archivos PDF de escolaridad. Devuelve el progreso del estudiante en formato estructurado (JSON).

Flujo de funcionamiento

El funcionamiento general del sistema se puede resumir en los siguientes pasos:

1. El usuario carga su escolaridad en formato PDF a través del frontend.
2. El archivo es enviado al backend, el cual lo deriva al microservicio de procesamiento.
3. El microservicio extrae las unidades curriculares aprobadas y transforma la información en un objeto JSON estructurado.
4. El backend utiliza esta información para construir un grafo de dependencias entre unidades curriculares (previaturas) y calcular una propuesta de plan de cursado.
5. El plan generado es devuelto al frontend, donde se visualiza de forma interactiva. El usuario puede ajustar parámetros como la carga académica máxima por semestre.

Todos los módulos se comunican a través del protocolo HTTP y utilizan JSON como formato estándar para el intercambio de datos, lo que permite una arquitectura distribuida, modular y escalable.

4.2. Arquitectura del sistema

Esta sección describe la arquitectura del sistema, detallando los componentes que lo conforman, sus responsabilidades y las interacciones que permiten su funcionamiento como una solución integrada. La descripción arquitectónica tiene como propósito proporcionar una visión clara y coherente de la estructura del sistema, que sirva como base para su desarrollo, mantenimiento y evolución.

Se abordan aspectos clave como la organización de los servicios, las tecnologías empleadas, los contratos de comunicación entre componentes, y los criterios arquitectónicos adoptados en términos de escalabilidad, modularidad y robustez.

La solución contempla un backend principal, un microservicio de scraping y procesamiento, y una interfaz web que permite la interacción con el sistema. Además, se incluye una descripción general de los mecanismos de comunicación entre servicios y un resumen de los requerimientos no funcionales más relevantes (como seguridad, rendimiento y disponibilidad).

4.2.1. Visión general

La arquitectura del sistema se basa en una estructura modular distribuida, compuesta por tres componentes principales: el frontend, el backend principal y un microservicio de scraping y procesamiento. Estos módulos se comunican entre sí mediante APIs RESTful, utilizando el protocolo HTTP y el formato JSON para el intercambio de datos.

La Figura 4.1 presenta un diagrama de alto nivel que resume la arquitectura general del sistema. A continuación, se describen los roles de cada componente y los flujos de información más relevantes:

- **Frontend (React)**: permite a los usuarios interactuar con la aplicación a través de una interfaz web. Realiza solicitudes al backend utilizando llamadas HTTP con datos en formato JSON.
- **Backend principal (Node.js/Express)**: concentra la lógica de negocio del sistema, gestiona los datos académicos y coordina la comunicación con el microservicio.
- **Microservicio (Python/FastAPI)**: se encarga de obtener datos desde fuentes institucionales mediante técnicas de scraping, y de procesar archivos PDF de escolaridad, retornando los datos en formato estructurado.

Flujos principales

- **Procesamiento de escolaridad (ver Figura 4.2)**: el usuario carga un archivo PDF con su escolaridad. Este archivo es enviado al backend, que a su vez solicita al microservicio su procesamiento. El microservicio devuelve un JSON con el progreso académico del estudiante.
- **Generación de plan de cursado (ver Figura 4.3)**: una vez procesado el avance, el backend construye un grafo de dependencias curriculares y aplica algoritmos de planificación. El plan resultante es devuelto al frontend para su visualización.

Estos flujos permiten automatizar y personalizar la planificación académica, combinando la escolaridad individual del estudiante con las restricciones del plan de estudios y la oferta semestral vigente.

4.2.2. Componentes del sistema

Esta sección describe en detalle los tres componentes principales que conforman la arquitectura del sistema: el backend principal, el microservicio de scraping y procesamiento, y el frontend web. Para cada uno se presentan sus responsabilidades, organización interna, tecnologías utilizadas y estrategias de pruebas.

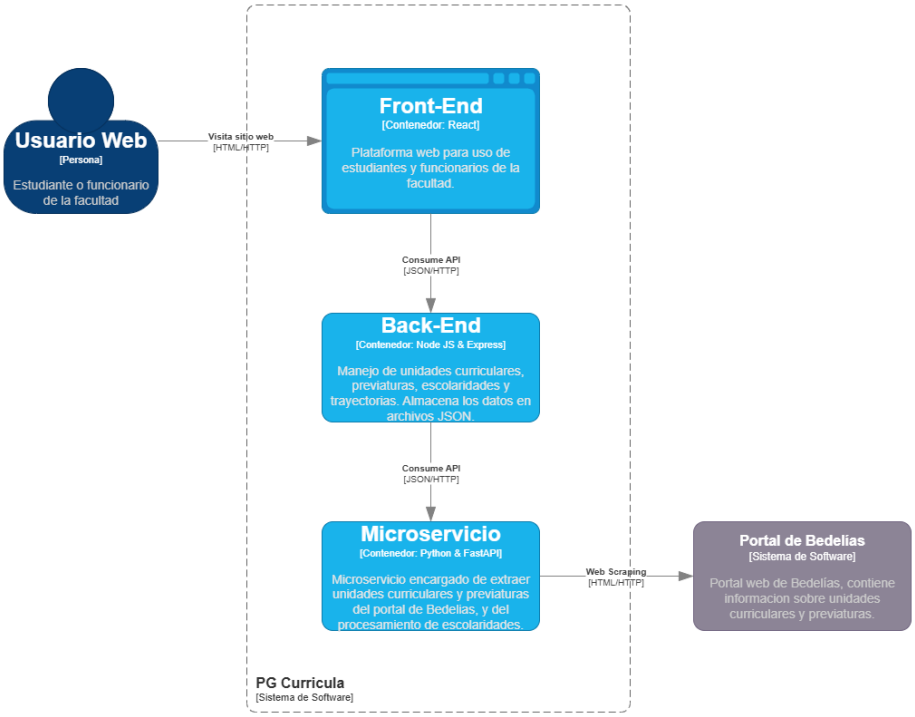

Figura 4.1: Arquitectura del sistema

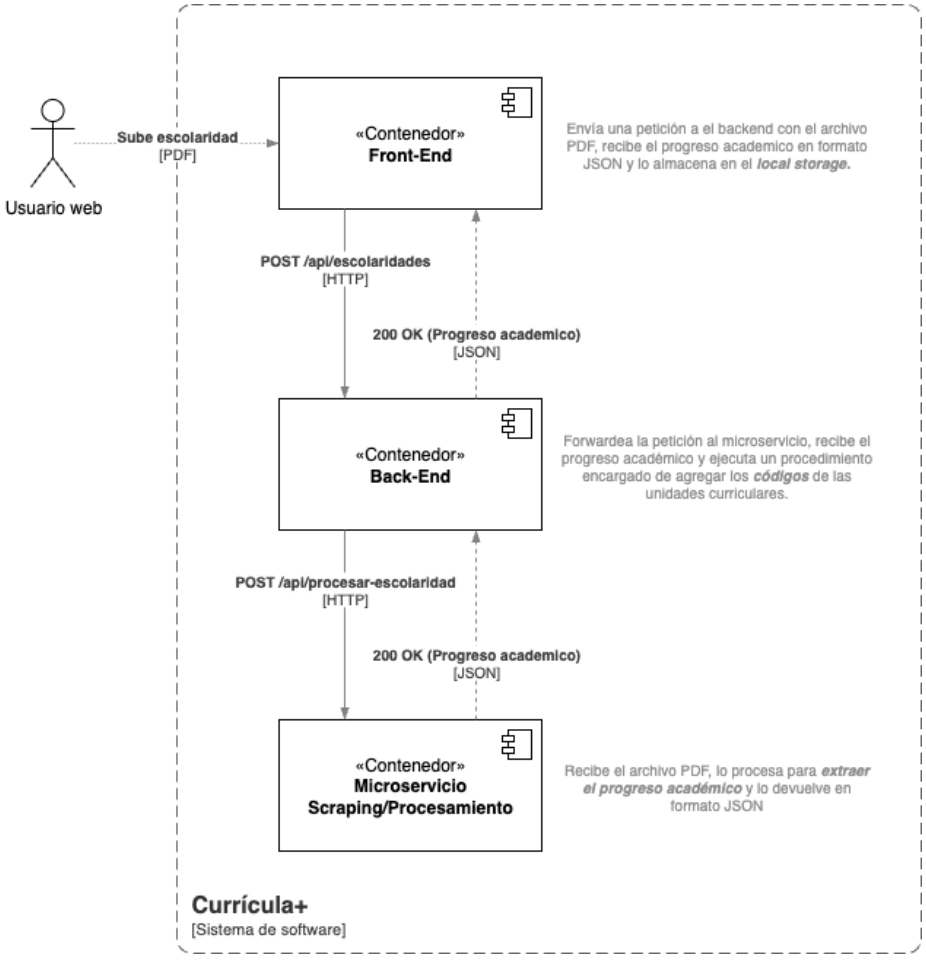

Figura 4.2: Diagrama de flujo de procesamiento de escolaridad

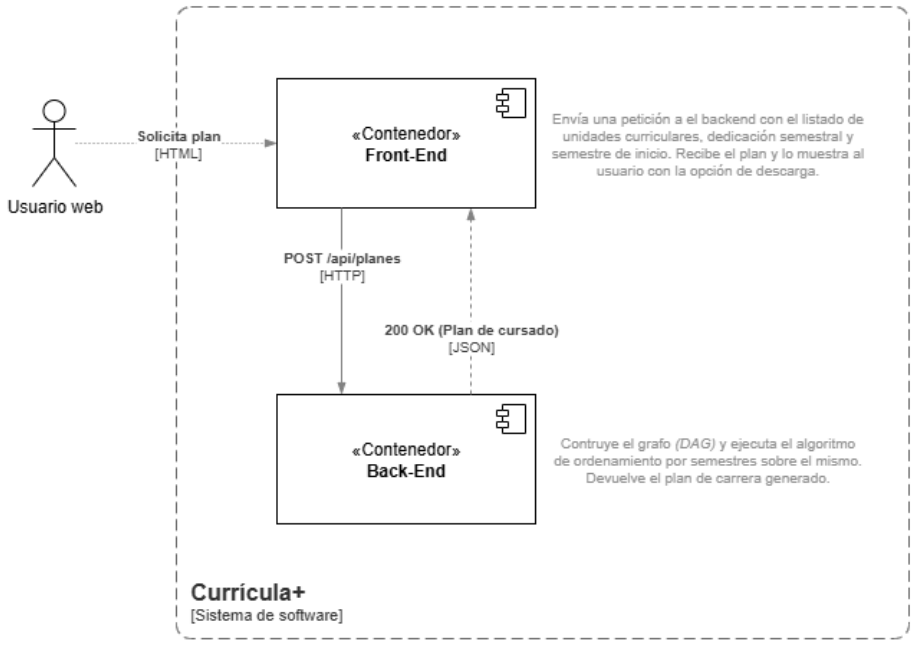

Figura 4.3: Diagrama de flujo de generación de plan de cursado

Backend principal (Node.js/Express)

El backend constituye el núcleo de la lógica de negocio del sistema. Su arquitectura está organizada en capas que separan responsabilidades, lo que favorece la mantenibilidad y escalabilidad del código.

Las principales responsabilidades funcionales del backend se organizan en los siguientes aspectos clave:

- **Gestión de unidades curriculares:** Se encarga, en colaboración con el microservicio, de obtener y mantener actualizado un listado completo de unidades curriculares. Expone endpoints para retornar dichas unidades ordenadas según la trayectoria académica sugerida, además de permitir búsquedas y filtrados específicos por código, nombre, rango de créditos, semestres de dictado, estado de habilitación de cursado y grupo académico.
- **Manejo de previaturas:** En coordinación con el microservicio, almacena una estructura completa que refleja las previaturas de todas las unidades curriculares. Provee endpoints para consultar las previaturas de una unidad curricular específica, así como para validar si un estudiante cumple con dichos prerrequisitos antes de cursar o rendir una unidad curricular.
- **Procesamiento de escolaridades:** Se encarga, con apoyo del microservicio, de procesar los datos de escolaridades recibidos, asignando a cada unidad curricular su código (ya que el microservicio entrega solo los nombres).
- **Generación de planes de cursado:** Aquí se concentra todo el flujo relacionado con la planificación académica, desde la generación inicial del listado de unidades curriculares y la construcción del grafo de previaturas, hasta la ejecución del algoritmo de scheduling que asigna cada unidad curricular a un semestre específico. Todo este proceso es expuesto mediante un endpoint dedicado.
- **Autenticación:** Se implementa un mecanismo seguro mediante API keys para proteger las solicitudes realizadas al servicio, restringiendo el acceso únicamente a usuarios autorizados y previniendo peticiones maliciosas.
- **Gestión de errores y registros:** El backend implementa manejo de errores utilizando un middleware para la captura centralizada de excepciones. Además, incorpora un sistema de logging doble (por consola para entornos de desarrollo y por archivos para entornos productivos), permitiendo un diagnóstico y seguimiento eficiente del comportamiento del sistema.

El backend del sistema sigue un diseño modular basado en una arquitectura organizada por capas. Esto facilita la mantenibilidad, mejora la escalabilidad del sistema y favorece la incorporación de nuevas funcionalidades sin afectar a las ya existentes. La organización interna está compuesta principalmente por tres capas:

- **Capa de rutas (Routes):** Actúa como punto de entrada del sistema, exponiendo los distintos endpoints mediante los cuales se reciben solicitudes desde los clientes externos. Cada ruta delega la responsabilidad de manejar las solicitudes recibidas a un controlador específico.
- **Capa de controladores (Controllers):** Cada controlador está vinculado directamente a una ruta, gestionando la recepción y validación inicial de las solicitudes entrantes, así como el envío estructurado de respuestas al cliente. Los controladores no implementan lógica de negocio compleja, su rol principal es intermediar entre la capa de rutas y la capa de servicios.
- **Capa de servicios (Services):** Contiene toda la lógica de negocio esencial del sistema, implementando los algoritmos y procedimientos internos necesarios para responder a las solicitudes provenientes de los controladores. Esta capa es la encargada de interactuar con otras capas del sistema, garantizando una separación clara y estructurada de las responsabilidades.

La persistencia de datos se realiza mediante archivos `.json`, los cuales son cargados en memoria al iniciar el servidor. Estos archivos incluyen la información estructurada sobre unidades curriculares, previaturas, oferta semestral, escolaridades utilizadas para testeo y trayectoria sugerida por facultad. La decisión de utilizar esta forma de persistencia se fundamenta por la naturaleza estática de los datos, ya que esta información no sufre modificaciones frecuentes; y por otro lado, la reducción de la complejidad del sistema, dado que al prescindir de una base de datos tradicional se simplifica considerablemente la arquitectura. Esta simplificación permite reducir la cantidad de componentes. Se pueden consultar más detalles de estos archivos en el Anexo C.

La estrategia de pruebas adoptada para el backend consiste en la implementación de pruebas unitarias, cuyo objetivo principal es verificar de forma individual y aislada cada una de las funcionalidades y métodos desarrollados. Mediante estas pruebas se busca garantizar el correcto funcionamiento del sistema, asegurar la robustez de la lógica de negocio y facilitar la detección temprana de errores, simplificando así el mantenimiento y futuras modificaciones. Actualmente no se realizan pruebas de integración, dejando estas como una posible mejora a incorporar en trabajos futuros.

Microservicio de scraping y procesamiento (Python/FastAPI)

El microservicio complementa al backend mediante la ejecución de dos tareas fundamentales: el procesamiento de archivos PDF de escolaridades y la extracción de datos desde el portal institucional de Bedelías. Este componente está implementado en Python, utilizando el framework `FastAPI` para exponer una interfaz HTTP con endpoints internos, consumidos exclusivamente por el backend.

Las responsabilidades clave del microservicio son:

- **Extracción de datos desde el portal de Bedelías:** Utilizando técnicas de scraping automatizado, el microservicio accede a distintas secciones del

sitio web de Bedelías, desde donde obtiene información sobre las unidades curriculares, grupos, oferta semestral actual y el sistema de previaturas. Para ello, se apoya en Selenium como *webdriver*[1] para automatizar la navegación y en BeautifulSoup para la extracción de datos desde el HTML. La información recolectada es transformada y almacenada en archivos JSON que posteriormente están disponibles para ser consultados por el backend principal.

- **Procesamiento de escolaridades en PDF:** A partir de los archivos PDF de escolaridades cargados por los usuarios, el microservicio extrae el contenido textual mediante la biblioteca PyPDF. Luego, interpreta la información contenida (como nombres de asignaturas y créditos) para generar un objeto estructurado en formato JSON que representa el avance académico del estudiante. Esta información es enviada al backend, que se encarga de completarla asignando los códigos correspondientes a cada unidad curricular.

En cuanto a su organización interna, el microservicio se estructura en carpetas funcionales:

- **Rutas y controladores (routers):** La carpeta routers unifica tanto las definiciones de rutas como sus manejadores (controladores). Aquí se definen los endpoints que exponen las funcionalidades del microservicio y se gestionan las solicitudes entrantes.
- **Servicios (services):** Contiene la lógica principal de scraping y procesamiento. En esta capa se implementan los procesos que automatizan la recolección de datos desde el portal de Bedelías, así como la extracción de información desde los archivos PDF de escolaridades.
- **Modelos (models), constantes (constants) y utilidades (utils):** Las estructuras de datos, constantes utilizadas en distintos puntos del microservicio y funciones auxiliares están organizadas en carpetas independientes (models, constants, utils) para promover la modularidad.
- **Datos persistidos (data):** La información extraída mediante scraping se almacena en archivos JSON dentro de la carpeta data, disponibles para ser consumidos por el backend principal.
- **Scripts de scraping (scripts):** Scripts independientes responsables de ejecutar los procesos de scraping. Se encargan de obtener y transformar los datos del portal, y de generar los archivos JSON de salida.

Por otro lado, las librerias clave utilizadas para lograr los fines deseados fueron:

[1] Un *webdriver* es una interfaz que permite automatizar navegadores web, posibilitando la interacción programática con páginas como si lo hiciera un usuario real.

- **Selenium:** Utilizado como web driver para automatizar la navegación por el portal de Bedelías, permitiendo acceder dinámicamente a secciones protegidas por interacción JavaScript o navegación estructurada.
- **BeautifulSoup:** Empleado para analizar el contenido HTML obtenido por Selenium y extraer los datos relevantes sobre unidades curriculares, grupos y previaturas.
- **PyPDF:** Utilizado para la lectura y extracción de texto desde los archivos PDF de escolaridades, a partir de los cuales se obtiene el progreso académico del estudiante.

Para más detalles sobre como funciona el microservicio se puede consultar el Anexo D.

Para las pruebas se implementaron pruebas unitarias, enfocadas en validar el comportamiento correcto de las funciones que componen la lógica de scraping y procesamiento. Estas pruebas buscan asegurar que, dadas entradas controladas, las funciones produzcan salidas esperadas, detectando errores en etapas tempranas del desarrollo. No se hicieron pruebas de integración ni de extremo a extremo, aunque su incorporación se considera una posible mejora futura.

Frontend (React/Typescript)

El **frontend** del sistema es el componente responsable de proporcionar la interfaz gráfica con la que interactúan los usuarios, tanto estudiantes como funcionarios. Desarrollado utilizando **React** y **TypeScript**, este módulo consume la API REST del backend y presenta los datos de forma clara, estructurada y navegable. Además de facilitar la interacción con las funcionalidades principales del sistema, el frontend se encarga de gestionar la experiencia de usuario, el enrutamiento de vistas y el estado de la aplicación. Las responsabilidades principales del frontend son:

- **Carga y procesamiento de escolaridades:** permite a los usuarios subir una escolaridad en formato PDF y visualizar el progreso académico extraído.
- **Visualización de unidades curriculares:** presenta el listado de unidades curriculares disponibles, con opciones de búsqueda y filtrado por nombre, código, créditos, grupo, semestre y habilitación para cursado.
- **Visualización de previaturas:** permite consultar las previaturas asociadas a cada unidad curricular y validar si se cumplen los requisitos académicos para cursarlas.
- **Generación y visualización de plan de cursado:** facilita la generación de trayectorias académicas personalizadas, indicando qué unidades curriculares cursar en cada semestre en base al avance del estudiante.

- **Gestión del flujo general de uso:** orquesta las diferentes pantallas del sistema (inicio, carga de escolaridad, exploración de unidades, generación de plan) y administra la navegación entre ellas.

En términos de organización del código, la aplicación está estructurada en componentes reutilizables:

- **api:** contiene la configuración base del cliente Axios y las funciones encargadas de realizar las llamadas asincrónicas al backend.
- **components:** agrupa todos los componentes reutilizables de la interfaz, incluyendo inputs, dropdowns, botones, entre otros.
- **contexts:** define contextos de React utilizados para manejar estado compartido en pantallas específicas.
- **layouts:** contiene los layouts base sobre los que se estructuran las distintas vistas del sistema.
- **models:** incluye constantes y tipos TypeScript utilizados en la aplicación para asegurar tipado fuerte.
- **router:** configura las rutas del sistema mediante React Router DOM.
- **screens:** contiene las pantallas principales de la aplicación, como inicio, exploración de unidades curriculares y generación de plan.
- **store:** centraliza el estado global utilizando la librería Zustand.
- **utils:** funciones auxiliares reutilizadas en distintos componentes o procesos.

El frontend implementa un enfoque mixto para la gestión del estado. Para el manejo de estado asincrónico vinculado a la API, se utiliza TanStack Query (React Query), lo que permite el *cacheo* de datos, la revalidación automática, el control de estados de carga y el manejo centralizado de errores. El estado global local se gestiona con Zustand, una solución ligera empleada principalmente para almacenar el progreso académico del estudiante. En los casos en que el estado es relevante únicamente dentro de partes específicas de la interfaz, se definen contextos React locales en el módulo `contexts`. La comunicación con APIs se realiza mediante un cliente Axios preconfigurado, que centraliza la configuración de encabezados, la base URL y el control de errores comunes. Todas las funciones de interacción con endpoints se agrupan dentro del módulo `api`. En cuanto al diseño de la interfaz, se emplea la librería `shadcn/ui` para utilizar componentes accesibles y estilizados, junto con Tailwind CSS como solución para la composición de estilos responsiva, flexible y moderna, asegurando una experiencia de usuario consistente, visualmente cuidada y ágil de desarrollar.

La interfaz del sistema fue diseñada cuidadosamente con foco en la experiencia de usuario (UX), priorizando la claridad, simplicidad y fluidez en la

navegación. Todo el diseño fue previamente trabajado en Figma, donde se definieron los flujos de interacción, jerarquía visual y disposición de componentes antes de su implementación en código. Esto permitió validar decisiones de diseño y mantener coherencia visual en todas las pantallas. Durante el desarrollo, se adoptaron buenas prácticas de diseño accesible, empleando componentes accesibles provistos por la librería shadcn/ui y aplicando estilos mediante Tailwind CSS, lo cual permitió construir una interfaz consistente y responsiva. También se prestó atención a contrastes de color, etiquetas semánticas y estados interactivos, asegurando que la plataforma sea utilizable para una amplia variedad de usuarios y dispositivos. En conjunto, el diseño de la UI se alinea con los objetivos funcionales del sistema, facilitando la comprensión y el uso de sus principales funcionalidades desde el primer ingreso.

4.2.3. Integración y comunicación

Esta sección describe de forma concisa cómo se integran los distintos componentes del sistema, qué contratos de comunicación existen entre ellos y cómo se maneja el control de errores de manera uniforme. El objetivo es garantizar un entendimiento común sobre cómo fluye la información entre servicios y cómo se preserva la consistencia en el intercambio de datos.

Contratos de API y llamadas entre servicios

La arquitectura del sistema está basada en un modelo cliente-servidor donde los componentes se comunican a través de API RESTful utilizando el formato JSON y el protocolo HTTP. Se definen dos vínculos principales de integración:

- **Frontend :: Backend principal:** El frontend realiza llamadas al backend utilizando Axios y React Query, consumiendo endpoints REST. Todas las respuestas se devuelven en formato JSON y los códigos de estado HTTP se utilizan para reflejar el resultado de la operación (éxito, error de cliente, error del servidor, etc.).
- **Backend principal :: Microservicio de scraping y procesamiento:** El backend actúa como cliente del microservicio, realizando solicitudes HTTP a sus endpoints internos. Esta comunicación también utiliza JSON como formato de intercambio, y no incluye mecanismos de autenticación, ya que el microservicio no está expuesto públicamente. Los contratos son simples y enfocados exclusivamente en transmitir datos estructurados.

Ambas interfaces siguen convenciones REST, manteniendo independencia entre componentes y facilitando su mantenimiento y evolución.

Manejo básico de errores

El sistema implementa un manejo de errores sencillo y consistente entre sus distintos componentes, con el objetivo de garantizar una experiencia de usuario

clara y facilitar el diagnóstico de fallos durante el desarrollo y despliegue. En el backend, los errores se gestionan de forma centralizada mediante un middleware que captura todas las excepciones generadas en las rutas, devolviendo una respuesta con el código HTTP correspondiente (como 400, 404 o 500) y un objeto JSON con un único campo de mensaje descriptivo. Este formato uniforme permite una interpretación sencilla desde el frontend y evita exponer detalles internos del sistema. El microservicio adopta el mismo esquema: ante errores como fallos en la lectura de archivos PDF o durante el proceso de scraping, responde con un mensaje JSON estructurado que el backend interpreta y reenvía al frontend. Por su parte, el frontend consume estos mensajes y los presenta al usuario mediante componentes visuales (generalmente *toasts*[2]), informando de forma clara qué ocurrió y cómo proceder, sin exponer información técnica ni detalles sensibles del error. Este enfoque minimalista pero coherente mantiene una interfaz de comunicación clara entre servicios y garantiza una experiencia de usuario adecuada, sin sobrecargar la capa de presentación con información innecesaria.

4.2.4. Atributos de calidad

Esta sección describe los requisitos no funcionales implementados del sistema, es decir, aquellos atributos de calidad que no están relacionados directamente con las funcionalidades del mismo, pero que son esenciales para su correcto funcionamiento, mantenimiento y evolución. A continuación, se detallan los aspectos más relevantes en cuanto a seguridad, rendimiento, disponibilidad, usabilidad y mantenimiento.

Seguridad

El sistema incorpora mecanismos básicos de seguridad orientados a proteger los servicios expuestos frente a accesos no autorizados y posibles abusos. En primer lugar, el backend principal implementa un esquema de autenticación mediante claves API, que permite restringir el uso de la API únicamente a clientes autorizados. Las claves deben ser incluidas en el encabezado de cada solicitud, lo que evita el uso indiscriminado de los servicios. Además, se incorporó un limitador de solicitudes (rate limiter) que previene comportamientos abusivos por parte de clientes que realicen un número excesivo de peticiones en un corto período de tiempo. Este mecanismo contribuye a mitigar posibles ataques de denegación de servicio (DoS por su sigla en inglés) y garantiza un uso responsable de los recursos disponibles.

Rendimiento

Aunque el sistema no está orientado a altos volúmenes de tráfico en su versión actual, se aplicaron buenas prácticas que permiten alcanzar un rendimiento ade-

[2]Un *toast* es una notificación breve y temporal que aparece en la interfaz para informar al usuario.

cuado para el contexto de uso previsto. Las respuestas del backend se optimizan para entregarse en formato JSON limpio, sin información redundante, reduciendo así el tamaño de los mensajes transmitidos. En el frontend, se aprovechan mecanismos de almacenamiento local, como localStorage, para evitar solicitudes innecesarias al servidor y acelerar la carga de información. Asimismo, se utiliza la librería React Query, que permite cachear respuestas de forma inteligente y minimizar la latencia durante la navegación del usuario por la aplicación.

Disponibilidad y monitoreo

El sistema fue diseñado para ejecutarse en entornos controlados, considerando principios básicos de disponibilidad, trazabilidad y tolerancia a fallos. Una de las decisiones clave fue la separación de responsabilidades entre el backend principal y el microservicio de scraping y procesamiento, lo que permite que el primero continúe operativo incluso ante errores temporales en la obtención o análisis de datos académicos. Adicionalmente, se implementaron mecanismos de registro de eventos en ambos servicios. Estos registros se adaptan según el entorno: salida por consola durante el desarrollo y escritura en archivos de log en producción. Esta estrategia facilita el rastreo de errores, advertencias y eventos relevantes. Ante fallos esperables, el sistema responde con mensajes estructurados y claros, minimizando el impacto sobre la experiencia del usuario y evitando interrupciones inesperadas.

Usabilidad

La interfaz de usuario fue diseñada en Figma siguiendo principios modernos de experiencia de usuario (UX), priorizando la claridad visual, la jerarquía informativa y la simplicidad en la navegación. Esto se traduce en una experiencia fluida y accesible, incluso para usuarios sin conocimientos técnicos. La aplicación es plenamente compatible con los principales navegadores del mercado, incluyendo Chrome, Firefox, Safari y Edge, lo que garantiza su correcto funcionamiento en diversos entornos. Además, el diseño es completamente responsivo, gracias al uso de Tailwind CSS, lo que permite su adecuada visualización tanto en computadoras de escritorio como en dispositivos móviles y tabletas.

Mantenimiento

Desde el punto de vista del mantenimiento, el sistema fue desarrollado con un enfoque modular y una organización del código clara y coherente. Se respetan principios de separación de responsabilidades y se emplea una estructura por capas y carpetas basadas en funcionalidades, lo que facilita la lectura, extensión y depuración del código. Además, se cuenta con documentación técnica que describe las funcionalidades principales del sistema, incluyendo los endpoints de las APIs, los tipos de datos utilizados y los contratos de respuesta esperados. Finalmente, tanto el backend como el microservicio disponen de pruebas unitarias que cubren los casos de uso más relevantes. Como trabajo futuro, se prevé

ampliar la suite de pruebas mediante la incorporación de pruebas de integración y pruebas de sistema, con el objetivo de aumentar la cobertura y robustez general de la solución.

4.2.5. Infraestructura y despliegue

El presente apartado describe la infraestructura utilizada para ejecutar la aplicación, así como el proceso de despliegue definido para facilitar su mantenimiento y actualización. Se detallan los entornos definidos, la estrategia de contenerización adoptada, la plataforma de alojamiento seleccionada, y el mecanismo de automatización implementado para garantizar una puesta en producción eficiente y segura.

Entornos definidos

Se definieron dos entornos diferenciados para el despliegue y ejecución del sistema, cada uno adaptado a sus necesidades particulares:

- **Entorno de desarrollo:** este entorno está orientado a facilitar el trabajo del equipo de desarrollo. Se ejecuta localmente en las estaciones de trabajo y permite implementar nuevas funcionalidades, realizar pruebas y depurar errores. Incluye mecanismos de *hot reload*[3] para una experiencia de desarrollo más ágil y expone los servicios (`frontend`, `backend` y `procesor-scraper`) en puertos locales accesibles.
- **Entorno de producción:** diseñado para el uso real por parte de los usuarios finales, este entorno elimina funcionalidades propias del desarrollo (como el *hot reload*) y agrega componentes necesarios para garantizar seguridad y disponibilidad. Se incluye el servicio de *Certbot*, encargado de gestionar automáticamente la provisión y renovación de certificados SSL, y *Nginx*, que se encarga de servir la aplicación web de manera segura sobre HTTPS en el puerto correspondiente.

La diferenciación entre entornos se logra manteniendo archivos específicos para cada uno: un `Dockerfile` y un archivo `docker-compose.yml` adaptados a las características y requerimientos de desarrollo y producción respectivamente. Esto permite garantizar entornos controlados, replicables y acordes a su propósito.

Contenerización con Docker

La aplicación se encuentra completamente contenerizada mediante Docker, lo que permite garantizar portabilidad, reproducibilidad y aislamiento entre entornos. Se definieron configuraciones específicas para los entornos de desarrollo

[3] El *hot reload* permite aplicar cambios en el código y reflejarlos de inmediato en la aplicación en ejecución, sin necesidad de reiniciarla.

y producción, manteniendo archivos independientes tanto para los `Dockerfile` como para los archivos `docker-compose.yml`.

En el entorno de desarrollo, los servicios se ejecutan con volúmenes montados que permiten reflejar en tiempo real los cambios realizados en el código fuente. Además, se utilizan herramientas de *hot reload* que agilizan el ciclo de desarrollo. Los servicios definidos incluyen:

- **Frontend:** aplicación web desarrollada con React y Vite, expuesta localmente en el puerto 5173.
- **Backend:** servidor principal construido con Node.js y Express, expuesto en el puerto 3000.
- **Processor-scraper:** microservicio desarrollado en Python con FastAPI, encargado del procesamiento de escolaridades, expuesto en el puerto 8000.

En el entorno de producción, los servicios son optimizados para su ejecución estable y segura. El frontend se construye en una etapa previa *(multi-stage build)* y es servido mediante Nginx. Adicionalmente, se incorpora un contenedor basado en Certbot para la provisión y renovación automática de certificados SSL. Los servicios se exponen en los puertos 80 (HTTP) y 443 (HTTPS), y se definen volúmenes persistentes para el almacenamiento de certificados.

Esta separación clara entre entornos permite garantizar una experiencia de desarrollo ágil y eficiente, mientras que en producción se priorizan la seguridad, el rendimiento y la disponibilidad del sistema.

Plataforma de infraestructura

La infraestructura de producción fue desplegada en la nube pública **Elasticloud**, un servicio provisto por ANTEL. Esta elección se vio favorecida por una alianza institucional con la Facultad de Ingeniería, la cual permitió acceder a un crédito mensual de $5000 pesos uruguayos para utilizar recursos de dicha plataforma.

Se aprovisionó una instancia virtual con **Docker Engine** preinstalado, sobre la cual se ejecuta todo el entorno de producción. La instancia fue configurada con un escalado vertical de hasta 12 *cloudlets*, garantizando así la disponibilidad de recursos suficientes ante eventuales picos de carga.

Las tareas de mantenimiento y despliegue se realizan mediante conexión remota utilizando **SSH**, lo que permite gestionar el sistema de forma segura. Desde la propia instancia se clona el repositorio del sistema, se actualiza la rama `main` y se reconstruyen los servicios de producción utilizando los archivos `Dockerfile.prod` y `docker-compose.prod.yml`.

Gracias a esta infraestructura, se logró un entorno seguro y escalable, capaz de alojar la aplicación web y exponerla públicamente con cifrado HTTPS y disponibilidad permanente.

Despliegue continuo

Con el objetivo de reducir la intervención manual, minimizar errores humanos y acelerar el ciclo de entrega, se implementó un **pipeline de despliegue continuo (CD)** utilizando **GitHub Actions**.

El pipeline de despliegue se activa automáticamente cada vez que se realiza un push a la rama `main` del repositorio. El flujo automatizado ejecuta los siguientes pasos:

1. Establece una conexión SSH segura con la instancia de producción en Elasticloud.
2. Clona o actualiza el código desde la rama principal del repositorio.
3. Reconstruye y reinicia los servicios mediante docker-compose, utilizando los archivos de configuración productiva.

Para proteger la información sensible, como la clave privada de acceso SSH y otros parámetros de conexión, se configuraron secretos cifrados en el repositorio de GitHub. Estos secretos son referenciados durante la ejecución del workflow, garantizando la seguridad del entorno de producción.

Esta estrategia permite que cualquier cambio validado en la rama main se despliegue automáticamente en producción, asegurando que la aplicación esté siempre actualizada sin comprometer su estabilidad operativa.

Conclusiones y posibles mejoras

La arquitectura de infraestructura y despliegue fue diseñada con el objetivo de ser simple, reproducible y segura, contemplando las necesidades actuales del proyecto y las capacidades técnicas disponibles. La contenerización mediante **Docker** garantiza entornos coherentes entre desarrollo y producción; y la utilización de **Elasticloud** como proveedor de infraestructura permite aprovechar una solución escalable y confiable, fruto de una alianza institucional con ANTEL.

El proceso de despliegue continuo implementado mediante **GitHub Actions** automatiza la puesta en producción, asegurando que cada actualización validada en la rama principal sea publicada de forma segura, sin intervención manual y con mínimos tiempos de inactividad.

A pesar de los logros obtenidos, existen oportunidades claras de mejora que podrían incorporarse en futuras iteraciones del sistema:

- **Escalado horizontal automático**, con múltiples réplicas de los servicios desplegados para mejorar la disponibilidad ante altas cargas de usuarios.
- **Incorporación de balanceadores de carga** para distribuir el tráfico de manera eficiente entre múltiples instancias.

- **Orquestación de contenedores con Kubernetes**, que permitiría gestionar múltiples servicios, replicación, monitoreo y escalado automático de forma más robusta.
- **Monitoreo y alertas en tiempo real**, integrando herramientas como Prometheus y Grafana para supervisar métricas del sistema y anticiparse a fallos.

Estas mejoras permitirían robustecer la solución en vistas a una eventual apertura institucional o uso masivo por parte de estudiantes y funcionarios, consolidando una herramienta confiable y escalable en el ecosistema de servicios académicos.

Capítulo 5

Extracción y representación de datos

Este capítulo describe el proceso mediante el cual se obtiene, transforma y estructura la información académica utilizada por el sistema. En particular, se aborda la extracción de datos esenciales para el funcionamiento del sistema como son unidades curriculares, grupos y el sistema de previaturas, todos elementos esenciales para el funcionamiento del sistema de generación de trayectorias personalizadas.

Se presenta la evolución del enfoque original, basado en consultas a una copia de la base de datos del Sistema de Gestión Administrativa de la Enseñanza (SGAE), hacia una solución automatizada mediante técnicas de web scraping, detallando los motivos de esta transición y los beneficios obtenidos. Asimismo, se explican los mecanismos actuales implementados para mantener actualizada esta información, así como la lógica empleada para representar jerárquicamente las previaturas mediante estructuras JSON. Estas estructuras permiten verificar el cumplimiento de requisitos académicos de forma eficiente y escalable, facilitando el análisis y la validación de planes de estudio sugeridos por la plataforma. Finalmente, se incluyen recomendaciones para el mantenimiento futuro del sistema de datos.

5.1. Obtención de datos

El sistema desarrollado requiere información precisa y actualizada sobre las unidades curriculares, sus relaciones de previatura y los grupos de formación asociados. Esta información constituye la base para la generación de planes de cursado realistas y personalizados. En esta sección se describe el proceso mediante el cual se obtiene dicha información, abarcando tanto el enfoque original como el actual, basado en técnicas de *web scraping*.

5.1.1. Enfoque original: acceso a base de datos

Durante las primeras etapas del desarrollo del sistema, la información académica era obtenida mediante consultas SQL ejecutadas sobre una copia local de la base de datos institucional del Sistema de Gestión Administrativa de la Enseñanza (SGAE). Como resultado de estas consultas se generaba un archivo CSV que contenía datos tabulares sobre unidades curriculares, previaturas y grupos.

Posteriormente, estos datos eran transformados a un formato jerárquico en archivos JSON, utilizando un conjunto de reglas que permitían estructurar lógicamente las previaturas. Esta representación jerárquica era esencial para que el backend pudiera verificar automáticamente el cumplimiento de requisitos académicos.

5.1.2. Limitaciones del enfoque original

El enfoque basado en la consulta a una copia local de la base de datos institucional presentó diversas limitaciones que comprometían su confiabilidad y mantenimiento. En primer lugar, el proceso carecía de mecanismos automáticos de actualización: cualquier cambio realizado en el portal oficial de bedelías no se reflejaba de forma inmediata en el sistema, ya que la obtención de datos dependía de solicitudes puntuales al equipo de la facultad. Esta dependencia resultaba especialmente crítica, dado que solo una persona contaba con acceso y conocimiento para generar la exportación, lo que dificultaba la coordinación y ralentizaba el flujo de trabajo.

Además, el listado extraído presentaba errores y omisiones, entre ellos previaturas mal estructuradas o unidades curriculares faltantes, lo cual afectaba la calidad de las trayectorias generadas. Por ejemplo, en el caso de la unidad curricular "Programación 3", la representación extraída inicialmente no reflejaba correctamente las condiciones requeridas para su inscripción. Estas inconsistencias, sumadas a la necesidad de intervención manual para cada actualización, motivaron la búsqueda de una solución más robusta y automatizada.

5.1.3. Extracción actual mediante *web scraping*

Con el objetivo de resolver las limitaciones del enfoque anterior, se desarrolló un microservicio especializado que automatiza por completo la extracción de información académica directamente desde el portal público de bedelías, utilizando técnicas de *web scraping*. Este servicio, implementado en Python utilizando el framework FastAPI, permite obtener de forma estructurada y programática las relaciones de previaturas entre unidades curriculares, un listado completo de unidades curriculares disponibles para la carrera y sus respectivos grupos.

Una de las principales ventajas de este nuevo enfoque es que elimina la dependencia de procesos manuales e intervención de terceros. En concreto, permite que el equipo de desarrollo o cualquier persona encargada de la tarea pueda ejecutar la extracción de datos de manera automática. Esto no solo mejora la

integridad de los datos (en el sentido de mantener una estructura interna coherente, completa y utilizable por el sistema), sino también su confiabilidad, al asegurar que la información se obtiene directamente desde la fuente oficial publicada por bedelías. Esto último permite reducir sustancialmente el riesgo de desactualizaciones o errores humanos.

Los datos extraídos por el microservicio son almacenados en archivos JSON con una estructura predefinida, listos para ser transformados y consumidos por el backend principal del sistema. Este proceso automatizado representa un componente clave para asegurar la mantenibilidad del sistema a medida que evoluciona la oferta académica.

La adopción de técnicas de *web scraping* como estrategia principal para la extracción de datos respondió a la necesidad de contar con un mecanismo más robusto, autónomo y alineado con la fuente oficial de información académica. Esta solución garantiza una mayor fidelidad a los datos publicados en el portal de bedelías, asegurando que la información procesada por el sistema sea coherente con la que consultan estudiantes y funcionarios. Además, permite una actualización automática de los datos, eliminando la necesidad de solicitar manualmente archivos institucionales o coordinar exportaciones con personal específico. Otro aspecto relevante es que ofrece una cobertura completa y actualizada del estado real de la oferta académica, lo cual es fundamental para generar trayectorias válidas y precisas. Finalmente, el uso de scripts versionados y comandos estandarizados facilita la reproducibilidad del proceso y su mantenimiento a lo largo del tiempo, permitiendo adaptar el sistema de forma ágil ante cambios en la estructura de los planes de estudio o en el propio portal de bedelías.

5.1.4. Proceso actual de extracción y actualización

El proceso actual de recolección y estructuración de datos académicos se encuentra completamente automatizado y está basado en la ejecución de comandos sobre contenedores Docker que encapsulan los distintos microservicios del sistema. Este flujo garantiza que la información extraída desde bedelías se transforme de forma controlada y consistente en los formatos requeridos por el backend de la aplicación.

El microservicio responsable de la extracción, denominado `processor-scraper`, se ejecuta como un contenedor independiente y concentra toda la lógica de scraping y procesamiento de datos. Está desarrollado en Python utilizando el framework FastAPI, y su orquestación se realiza mediante `docker-compose`. Su función principal es recolectar directamente desde el portal de bedelías las relaciones de previaturas, la asociación entre grupos de formación y unidades curriculares, y el listado completo de unidades curriculares por plan de estudios. Esta tarea se lleva a cabo mediante la ejecución de los siguientes comandos:

```
1. docker-compose exec processor-scraper python -m
scripts.scrape_previatures

2. docker-compose exec processor-scraper python -m
scripts.scrape_groups_and_subjects

3. docker-compose exec processor-scraper python -m
scripts.scrape_first_semester_subjects

4. docker-compose exec processor-scraper python -m
scripts.scrape_second_semester_subjects
```

La ejecución de estos comandos genera archivos intermedios en formato JSON, los cuales son posteriormente procesados por el backend principal del sistema. Para transformar estos archivos y adaptarlos a la estructura interna utilizada por el sistema, se utilizan los siguientes comandos:

```
1. docker-compose exec backend pnpm generate:previaturas

2. docker-compose exec backend pnpm
generate:ucs-primer-semestre

3. docker-compose exec backend pnpm
generate:ucs-segundo-semestre

4. docker-compose exec backend pnpm
generate:unidades-curriculares

5. docker-compose exec backend pnpm generate:ucs-grupos

6. docker-compose exec backend pnpm
generate:ucs-grupos-actuales
```

Cada uno de estos pasos permite transformar y consolidar la información obtenida, garantizando que esté lista para ser consumida por los distintos módulos de la plataforma de planificación académica. Esto asegura la coherencia estructural de los datos, su disponibilidad inmediata y su alineación con la fuente oficial.

Actualmente, la ejecución de este flujo de actualización se realiza de forma manual al inicio de cada semestre o ante la detección de cambios relevantes en la oferta académica. No obstante, se prevé automatizar este proceso en futuras versiones del proyecto, con el objetivo de mantener la sincronización continua entre el sistema y el portal de bedelías, minimizando la intervención humana y reduciendo los posibles errores operativos.

5.2. Sistema de previaturas

En esta sección se detalla la forma en que el sistema representa las reglas de previaturas de las unidades curriculares, utilizando una estructura jerárquica en formato JSON. Este modelo permite expresar tanto condiciones simples como estructuras lógicas complejas de forma estructurada y evaluable, facilitando el análisis automático del cumplimiento de requisitos por parte del estudiante.

El sistema de previaturas se representa como un árbol lógico, donde cada nodo corresponde a una regla (por ejemplo, `AND`, `OR`, `NOT` o `SOME`) o a un caso base (como una unidad curricular específica o un mínimo de créditos). Esta organización recursiva permite modelar relaciones de dependencia entre asignaturas y requisitos curriculares de manera flexible y extensible.

5.2.1. Estructura del JSON

El archivo JSON está organizado como un objeto global donde cada clave representa el nombre de una unidad curricular. Cada entrada contiene un objeto con la regla raíz de previatura asociada y sus condiciones anidadas. El acceso a esta información es directo, utilizando una estructura de clave-valor (*hashmap*), lo que permite consultas eficientes por parte del sistema.

A continuación, se describen las distintas reglas disponibles, junto con ejemplos representativos que ilustran su sintaxis y función:

`UC` Es una condición base que indica que el estudiante debe haber aprobado una unidad curricular específica. Se representa mediante su código, nombre y tipo de instancia (C = curso, E = examen). Por ejemplo:

```
{
  "regla": "UC",
  "codigo": "1323",
  "nombre": "PROGRAMACIÓN 3",
  "tipoInstancia": "E"
}
```

`CREDITOS_PLAN` Requiere haber alcanzado un mínimo de créditos dentro del plan de estudios. Por ejemplo:

```
{
  "regla": "CREDITOS_PLAN",
  "cantidad": 140
}
```

`CREDITOS_GRUPO` Exige un mínimo de créditos aprobados dentro de un grupo específico de unidades curriculares. Por ejemplo:

```
{
  "regla": "CREDITOS_GRUPO",
  "codigo": 4083,
  "nombre": "MATEMÁTICA",
  "cantidad": 70
}
```

`AND` Requiere que todas las condiciones dentro de la regla se cumplan para acceder a la unidad curricular. En el ejemplo se indican que el curso de PROGRAMACIÓN 3 y haber alcanzado un mínimo de 300 créditos dentro del plan de estudios son previas de PROYECTO DE GRADO.

```
"PROYECTO DE GRADO": {
  "regla": "AND",
  "previas": [
    {
      "regla": "UC",
      "codigo": "1323",
      "nombre": "PROGRAMACIÓN 3",
      "tipoInstancia": "C"
    },
    {
      "regla": "CREDITOS_PLAN",
      "cantidad": 300
    }
  ]
}
```

`OR` Permite que el estudiante cumpla con al menos una de las condiciones listadas. En el ejemplo se indican que el curso de INVESTIGACIÓN OPERATIVA I o de INVESTIGACIÓN OPERATIVA II son previas de INTRODUCCIÓN A LA ROBÓTICA.

```
"INTRODUCCIÓN A LA ROBÓTICA": {
  "regla": "OR",
  "previas": [
    {
      "regla": "UC",
      "codigo": "1650",
      "nombre": "INVESTIGACIÓN OPERATIVA I",
      "tipoInstancia": "C"
    },
    {
      "regla": "UC",
      "codigo": "1610",
      "nombre": "INVESTIGACIÓN OPERATIVA II",
      "tipoInstancia": "C"
    }
  ]
}
```

`NOT` Indica que una determinada condición no debe haberse cumplido. En el ejemplo se indican que el curso de ASPECTOS BÁSICOS DE REDES no debe estar aprobada para cursar REDES DE COMPUTADORAS

```
"REDES DE COMPUTADORAS": {
  "regla": "AND",
  "previas": [
    {
      "regla": "NOT",
      "previas": {
        "regla": "UC",
        "codigo": "1454",
        "nombre": "ASPECTOS BÁSICOS DE REDES",
        "tipoInstancia": "E"
      }
    }
  ]
}
```

`SOME` Requiere que se cumpla una cantidad mínima de condiciones dentro de un conjunto. En el ejemplo se indican que el curso de ÁLGEBRA, GEOMETRÍA o CÁLCULO I es necesaria para cursar MATEMÁTICA DISCRETA.

```
"MATEMÁTICA DISCRETA": {
  "regla": "SOME",
  "cantidad": 2,
  "previas": [
    {
      "regla": "UC",
      "codigo": "1001",
      "nombre": "ÁLGEBRA",
      "tipoInstancia": "C"
    },
    {
      "regla": "UC",
      "codigo": "1002",
      "nombre": "GEOMETRÍA",
      "tipoInstancia": "C"
    },
    {
      "regla": "UC",
      "codigo": "1003",
      "nombre": "CÁLCULO I",
      "tipoInstancia": "C"
    }
  ]
}
```


Este esquema de representación permite validar dinámicamente si un estudiante cumple con los requisitos definidos para inscribirse en cada unidad curricular, a partir del análisis del árbol de reglas correspondiente. La estructura jerárquica facilita tanto la interpretación humana como el procesamiento automático por parte del sistema.

5.3. Ventajas y conclusiones

La implementación del nuevo proceso de extracción y estructuración de datos, basado en *scraping* directo desde el portal oficial de bedelías, ha introducido mejoras significativas respecto al enfoque anterior que se apoyaba en consultas a una copia local de la base de datos institucional. Estas mejoras abarcan tanto la calidad de los datos obtenidos como la facilidad de mantenimiento del sistema y su capacidad de adaptación futura.

Uno de los principales beneficios es que los datos extraídos son confiables y están siempre actualizados, dado que provienen directamente de la fuente oficial consultada por estudiantes y funcionarios. Esto permite reflejar con precisión la realidad académica vigente, eliminando la posibilidad de operar con información desactualizada, que era una de las principales limitaciones del enfoque anterior.

Además, el nuevo proceso ha permitido eliminar errores frecuentes relacionados con previaturas incompletas, unidades curriculares faltantes o estructuras mal interpretadas. Al extraer directamente lo que se presenta públicamente en el portal de bedelías, se asegura una representación fiel y coherente de las relaciones académicas establecidas.

Otro aspecto destacable es que el proceso actual es mucho más fácil de *debugear* y mantener. Gracias al uso de scripts versionados y comandos estandarizados, es posible replicar cada paso del flujo de extracción y procesamiento, identificando rápidamente errores o inconsistencias en etapas específicas del proceso. Esto representa una mejora sustancial en términos de trazabilidad y confiabilidad operativa.

La arquitectura modular adoptada también aporta una mayor robustez y mantenibilidad al sistema. Al separar la lógica de scraping en un microservicio independiente y encapsular las transformaciones posteriores mediante comandos en el backend, se facilita la incorporación de cambios incrementales, el aislamiento de errores y la evolución del sistema en función de nuevos requerimientos o modificaciones en las fuentes de datos.

En resumen, la incorporación de este nuevo enfoque ha permitido consolidar un sistema más preciso, robusto y alineado con las necesidades reales de los usuarios. El pasaje desde una arquitectura basada en consultas a una base de datos estática hacia una solución automatizada de scraping resolvió limitaciones estructurales significativas y mejoró sustancialmente la calidad de los datos utilizados. Actualmente, el sistema cuenta con un flujo confiable y eficiente que permite obtener información académica directamente desde la fuente oficial, estructurar reglas de previaturas en un formato jerárquico y evaluable, y mantener los datos actualizados a través de comandos reproducibles. Este nuevo proceso no solo garantiza una mejor experiencia de usuario al ofrecer información veraz y actualizada, sino que también sienta las bases para un mantenimiento más ágil y una evolución sostenida del sistema a futuro.

Capítulo 6

Algoritmos implementados

En este capítulo se presentan en detalle los algoritmos desarrollados como parte del núcleo funcional del sistema. Estos algoritmos constituyen la base computacional que permite automatizar la creación de trayectorias de cursado.

En el Apartado 5.4.1 se presenta el algoritmo encargado de generar un listado de las unidades curriculares que el estudiante debe cursar. En el Apartado 5.4.2 se explica el algoritmo encargado de generar el grafo de previas. En el Apartado 5.4.3 se detalla el algoritmo responsable de la planificación académica por semestres. Los mismos serán documentados a continuación, incluyendo su propósito, entradas y salidas, fundamento teórico, proceso general, consideraciones de diseño, casos especiales y posibles mejoras futuras.

6.1. Generación de listado de unidades curriculares faltantes

El presente algoritmo tiene como objetivo generar de forma automatizada un listado de unidades curriculares que un estudiante debe cursar para cumplir con los requisitos establecidos por el plan de estudios y, en consecuencia, acceder al título correspondiente.

Fundamentación teórica

La lógica subyacente de este algoritmo se apoya en el modelado del problema como un *problema de satisfacción de restricciones* (*Constraint Satisfaction Problem*, CSP) (Bacchus, s.f.). Este tipo de problemas son ampliamente utilizados en inteligencia artificial y planificación automática, y consisten en encontrar una asignación de valores a un conjunto de variables que satisfaga un conjunto de restricciones definidas sobre dichas variables.

En el contexto de el sistema desarrollado, las variables corresponden a las unidades curriculares que el estudiante podría cursar, y sus dominios están defi-

nidos por la oferta académica restante según su progreso académico. Las restricciones están determinadas por los requisitos del plan de estudios, tales como:

- Créditos mínimos por grupo (Matemática, Programación, etc.).
- Total de créditos necesarios para la obtención del título.
- Unidades curriculares obligatorias que deben ser aprobadas.

La solución al CSP consiste en encontrar un subconjunto de unidades curriculares que, al ser cursadas, satisfacen simultáneamente todas estas condiciones. Aunque el algoritmo actual recurre a una estrategia heurística basada en selección aleatoria, este enfoque establece una base sólida para futuras extensiones. En particular, podrían incorporarse técnicas de *machine learning* para mejorar la calidad de los planes generados.

Entradas

Para ejecutar el algoritmo, se requiere la siguiente información estructurada:

- **Progreso académico del estudiante:**
  - Lista de unidades curriculares aprobadas.
  - Créditos por grupo y totales acumulados hasta el momento.
- **Requisitos del título:**
  - Total de créditos necesarios para completar la carrera.
  - Créditos mínimos requeridos por grupo (por ejemplo: Matemática, Programación, Ingeniería de Software, etc.).

Proceso

El algoritmo implementado sigue una lógica de procesamiento secuencial por etapas, detalladas a continuación:

1. **Filtrado de unidades curriculares obligatorias:**
   - Se identifican las unidades curriculares obligatorias que ya han sido aprobadas por el estudiante.
   - Se eliminan de la lista de unidades curriculares obligatorias por cursar.
   - Se agregan al listado final las unidades curriculares obligatorias filtradas.
2. **Procesamiento por grupo:**
   - Se eliminan las unidades curriculares ya aprobadas dentro de cada grupo.

- Mientras no se cumpla el mínimo de créditos requerido por grupo, se seleccionan unidades curriculares restantes.
- Actualmente, la selección se realiza de forma aleatoria, con verificación de cumplimiento parcial del criterio del grupo.

3. **Completar hasta alcanzar los créditos totales:**
    - Si, tras satisfacer los requisitos por grupo y las unidades curriculares obligatorias, aún no se alcanza el total de créditos del título, se seleccionan nuevas unidades curriculares aleatoriamente de grupos hasta completar los 450 créditos.

Ejemplo de ejecución

Dado un estudiante con 190 créditos aprobados, incluyendo 45 créditos en el área de Matemática y 38 en Programación, y considerando los siguientes requisitos del plan de estudios:

- Total de créditos requeridos: 450
- Créditos mínimos por grupo: Matemática (70), Programación (60), etc.
- UCs obligatorias: `Cálculo DIV`, `GAL 1`, `GAL 2`, entre otras.

La salida del algoritmo incluirá:

- Las unidades curriculares obligatorias que aún no han sido aprobadas.
- Un conjunto de unidades adicionales seleccionadas de los distintos grupos, cuya aprobación permite satisfacer tanto los créditos por grupo como el total del plan.

Resultado obtenido

El resultado del algoritmo consiste en una lista de unidades curriculares, cuya aprobación permitiría al estudiante alcanzar los créditos totales y por grupo requeridos, así como cumplir con todas las unidades curriculares obligatorias del plan de estudios.

Limitaciones y mejoras futuras

Actualmente, la estrategia de selección de unidades curriculares es aleatoria dentro de cada grupo, lo que implica que el algoritmo no considera preferencias del estudiante, complejidad de las unidades curriculares ni correlaciones entre áreas de conocimiento. Esto podría generar planes poco realistas, subóptimos o que no reflejen adecuadamente los intereses del estudiante.

Asimismo, aunque el algoritmo procura generar una cantidad equivalente de créditos entre semestres pares e impares, esta distribución se realiza únicamente sobre el conjunto de unidades seleccionadas aleatoriamente, sin considerar explícitamente las unidades obligatorias. Como consecuencia, pueden surgir trayectorias más largas de lo necesario.

Como línea futura de trabajo, se propone incorporar heurísticas inteligentes de selección, como:

- Priorización de unidades curriculares relacionadas con áreas donde el estudiante tiene mayor avance o afinidad.
- Consideración de popularidad o dificultad.
- Implementación de modelos de *machine learning*.

6.2. Construcción del grafo *(DAG)*

Este algoritmo tiene como finalidad construir un grafo dirigido acíclico (DAG, por su sigla en inglés) (Jean C. Digitale, 2021) que represente las relaciones de precedencia y espera entre las unidades curriculares que un estudiante debe cursar para finalizar la carrera. La estructura resultante permite modelar la secuencia factible de cursado, considerando las restricciones impuestas por el sistema de previaturas y la oferta semestral.

Fundamentación teórica

La elección de un grafo dirigido acíclico como modelo subyacente se fundamenta en su capacidad para representar relaciones de dependencia sin ambigüedades, y es ampliamente utilizada en contextos donde se requiere respetar un orden parcial entre tareas o actividades. En este caso, cada nodo del grafo representa una unidad curricular, y las aristas dirigen el flujo desde las unidades curriculares que actúan como previas hacia aquellas que dependen de ellas.

Formalmente, un DAG es un grafo dirigido tal que no existen ciclos, es decir, no existe un camino que comience y termine en el mismo nodo siguiendo la dirección de las aristas. Esta propiedad es esencial para modelar planes de cursado, dado que nunca existirán situaciones de dependencia circular entre unidades curriculares.

La representación y posterior procesamiento de este grafo permite aplicar algoritmos clásicos como el ordenamiento topológico (Sedgewick y Wayne, 2011) y camino crítico (Kelley y Walker, 1959), técnicas comúnmente empleadas en teoría de grafos y planificación de proyectos *(scheduling)*.

Entradas del algoritmo

El algoritmo requiere como entrada:

- **Progreso académico del estudiante:** lista de unidades curriculares aprobadas y créditos por grupo y totales acumulados.
- **Semestre inicial:** corresponde al semestre de partida para el que se desea realizar la planificación.
- **Listado de unidades curriculares a cursar:** conjunto de unidades curriculares que el estudiante debe aprobar para completar el plan de estudios, obtenido mediante el algoritmo previo de generación de listado de unidades curriculares o selección manual.

Estructura del grafo generado

El grafo se define de la siguiente manera:

- Cada nodo representa una unidad curricular pendiente de cursar, con excepción del nodo *inicial*, que simboliza el conjunto de unidades curriculares ya aprobadas por el estudiante al momento de comenzar la planificación, y del nodo *final*, que representa el egreso.
- Las aristas indican relaciones de previatura y dependencia temporal entre unidades curriculares. El peso asociado a cada arista refleja la cantidad mínima de semestres que deben transcurrir entre la aprobación de una unidad curricular y el cursado de su correspondiente dependiente.
- Adicionalmente, se define un conjunto separado denominado *pool*, en el cual se almacenan las unidades curriculares que no poseen otras como previas directas, pero que presentan otro tipo de restricciones, tales como requisitos de créditos aprobados totales o por grupo. Estas unidades curriculares no se incluyen en el grafo principal, pero se contemplan durante la etapa de planificación.

Proceso de construcción

El algoritmo de construcción del grafo se desarrolla en etapas iterativas, en las cuales se añaden nodos y aristas que representan las unidades curriculares faltantes y sus relaciones de precedencia. A continuación se describe el procedimiento detallado:

1. **Inicialización:** se crea un nodo inicial que representa el conjunto de unidades curriculares ya aprobadas por el estudiante. Desde este nodo se generan aristas hacia todas las unidades curriculares que se encuentran habilitadas para cursar inmediatamente, según el estado académico del estudiante.
2. **Asignación de pesos iniciales:** las aristas que parten desde el nodo inicial hacia las primeras unidades curriculares habilitadas reciben un peso según la oferta semestral:

- Peso 0: si la unidad curricular se dicta en el mismo semestre de inicio del estudiante.
- Peso 1: si la unidad curricular se dicta en el semestre siguiente.

3. **Expansión del grafo:** una vez incorporadas las primeras unidades curriculares, el algoritmo continúa iterativamente incorporando nuevas unidades habilitadas, considerando las dependencias definidas por el sistema de previaturas. Para cada par de unidades curriculares relacionadas por una dependencia, se agrega una arista con un peso que representa el desfasaje mínimo entre el cursado de una y otra. Las aristas se asignan según la siguiente lógica:

    - Peso 1: si la unidad curricular dependiente se dicta en un semestre diferente al de la unidad curricular previa (por ejemplo, una en semestre par y otra en impar), esto significa que puede ser cursada inmediatamente despues de aprobar la anterior.
    - Peso 2: si ambas unidades curriculares se dictan en el mismo semestre. En este caso, el estudiante deberá esperar un año completo (dos semestres) entre cursadas.
    - Peso 3: se asigna como marcador especial cuando la unidad curricular origen se dicta en ambos semestres. Este peso indica que la dependencia no puede resolverse estáticamente, ya que el semestre en el que se curse la unidad curricular origen será decidido posteriormente durante el proceso de planificación. El valor real del peso dependerá del resultado del algoritmo de *scheduling*.

4. **Armado del conjunto separado (*pool*):** paralelamente, se identifica un subconjunto de unidades curriculares que no tienen a otras unidades curriculares como previas directas, pero que presentan restricciones alternativas, como la aprobación de un número mínimo de créditos totales o por grupo. Estas unidades curriculares no se incorporan al grafo principal, ya que sus restricciones no pueden representarse mediante relaciones de precedencia entre nodos. En su lugar, se almacenan en un conjunto separado denominado *pool*, que será evaluado posteriormente durante el proceso de planificación.

5. **Conexión al nodo final:** una vez que todas las unidades curriculares faltantes han sido incorporadas al grafo, se agrega un nodo final que representa el egreso del estudiante. Desde cada unidad curricular terminal (aquella que no es previa de ninguna otra) se genera una arista hacia este nodo. El peso asignado a esta arista depende de la duración de la unidad curricular:

    - Peso 1: si la unidad curricular es semestral.
    - Peso 2: si la unidad curricular es anual.

Este procedimiento permite construir un grafo dirigido acíclico que respeta las dependencias académicas, incorpora restricciones temporales y semestrales. Al finalizar, deja preparada la estructura para ser procesada por algoritmos de programación temporal, como el cálculo del camino crítico o la generación de un plan de cursado factible.

Resultado obtenido

El grafo dirigido acíclico resultante constituye una representación computacional estructurada del conjunto de unidades curriculares pendientes y sus relaciones de dependencia. Esta representación habilita el planteo del problema de generación de planes de estudio desde una perspectiva algorítmica, facilitando la aplicación de estrategias de optimización para la planificación de la carrera.

El DAG obtenido sirve como base para la ejecución de algoritmos como el ordenamiento topológico, empleado en el cálculo del camino crítico, lo cual permite estimar la duración mínima restante de la carrera. Asimismo, esta estructura es fundamental para la ejecución del algoritmo de planificación por semestres, responsable de generar trayectorias curriculares personalizadas que respeten tanto las dependencias académicas como las restricciones temporales y de carga horaria.

6.3. Planificación por semestres *(scheduling)*

El presente algoritmo tiene como finalidad generar un plan de cursado personalizado y óptimo para un estudiante, distribuyendo las unidades curriculares restantes a lo largo de los semestres futuros de forma compatible con su trayectoria académica, las restricciones curriculares vigentes y su disponibilidad de carga horaria.

Este plan busca minimizar la duración total restante de la carrera, respetando las relaciones de previaturas entre unidades curriculares, las condiciones de dictado (semestres impares o pares) y los requisitos establecidos por los planes de estudio (como la aprobación de créditos mínimos por grupo y totales). Para ello, se cuenta con el modelado de la trayectoria académica como un grafo dirigido acíclico, lo que permite aplicar técnicas de análisis estructural, como el ordenamiento topológico y cálculo de camino crítico, para guiar el proceso de planificación.

En particular, el algoritmo prioriza la inclusión de unidades curriculares críticas (aquellas cuya holgura es nula) y se adapta dinámicamente a las restricciones de oferta semestral, reorganizando los nodos del grafo en función del semestre actual y ajustando las aristas cuando existen dependencias condicionadas por el dictado de las unidades curriculares.

Este procedimiento constituye el núcleo central del sistema, y es el responsable de traducir la situación académica del estudiante en un itinerario concreto y viable para la finalización de su carrera.

Fundamento teórico

El algoritmo de planificación por semestres se apoya en el modelado de la trayectoria académica como un problema de programación con restricciones, utilizando conceptos clásicos de análisis de grafos. Las unidades curriculares y sus previaturas se representan mediante un grafo dirigido acíclico, lo que permite aplicar un ordenamiento topológico para generar un recorrido lineal que respete las dependencias existentes entre las unidades curriculares. Para calcular la duración mínima de la carrera y detectar cuáles unidades son críticas, se recurre al método de camino crítico. Desde un punto de vista formal, este problema puede verse como un problema de satisfacción de restricciones, donde:

- Las variables corresponden a las unidades curriculares a asignar.
- Los dominios son los semestres posibles para cada unidad.
- Las restricciones incluyen previaturas, carga horaria y oferta semestral.

Esta base teórica permite futuras extensiones del algoritmo, por ejemplo, considerando preferencias del estudiante o restricciones adicionales de manera sencilla.

Entradas

El algoritmo de planificación por semestres requiere como insumo un conjunto de datos estructurados que representan tanto el estado académico actual del estudiante como las restricciones curriculares impuestas por el plan de estudios. A continuación, se detallan los elementos necesarios:

1. **Semestre inicial:** Indica el semestre a partir del cual se desea comenzar la planificación. Puede tomar los valores *"1"* o *"2"*, correspondientes al primer o segundo semestre del año respectivamente. Este valor determina el punto de partida para la asignación de unidades curriculares.
2. **Carga máxima de créditos por semestre**: Define el límite superior de créditos que el estudiante está dispuesto a asumir por semestre. Este parámetro es utilizado para evitar la sobrecarga académica y se aplica como umbral flexible con una tolerancia configurable (por defecto, hasta un 12 % adicional).
3. **Progreso académico del estudiante**: Objeto que representa el avance académico del estudiante, incluyendo:
   - Lista de unidades curriculares ya aprobadas.
   - Créditos acumulados totales.
   - Créditos por grupo (Matemática, Programación, etc.).

   Esta información se actualiza de forma dinámica durante la planificación a medida que se asignan nuevas unidades curriculares.

4. **Grafo de dependencias curriculares**: Estructura que representa las relaciones de previaturas y el orden de cursado de las unidades curriculares.

5. **Conjunto de unidades curriculares separado sin previaturas**: Conjunto de unidades curriculares que no presentan dependencias directas en el grafo (por ejemplo, aquellas que únicamente requieren haber alcanzado cierto número de créditos en el plan o en un grupo específico). Estas unidades se consideran en la planificación cuando existen espacios disponibles dentro de los límites de carga definidos.

Descripción general del algoritmo

El algoritmo de planificación por semestres opera de manera iterativa, construyendo un plan de cursado a partir de la situación académica actual del estudiante. En cada iteración se simula un semestre, en el cual se seleccionan las unidades curriculares más adecuadas para ser cursadas, respetando las restricciones impuestas por el plan de estudios y buscando minimizar la duración total restante de la carrera.

El proceso general se puede dividir en las siguientes etapas:

1. **Inicializacion:** Se establecen los parámetros de entrada, incluyendo el semestre inicial, la carga máxima de créditos por semestre y el estado académico del estudiante. Además, se calcula el ordenamiento topológico del grafo de unidades curriculares y se determina el camino crítico, identificando la holgura y el tiempo de inicio más temprano de cada nodo.

2. **Selección de unidades curriculares disponibles:** Para cada semestre, se identifican las unidades curriculares que pueden ser cursadas, es decir, aquellas que tienen su tiempo de inicio más temprano menor o igual al semestre actual, cumplen con todas las previaturas requeridas (tanto estructurales como de créditos) y se dictan en el semestre en curso. Las unidades disponibles se ordenan por holgura ascendente, priorizando aquellas críticas para no demorar el progreso académico.

3. **Asignación de unidades curriculares al semestre:** Se seleccionan unidades curriculares del conjunto disponible hasta alcanzar la carga máxima permitida. El algoritmo contempla la distribución de créditos de unidades curriculares anuales (como por ejemplo proyecto de grado), asignando la mitad en el semestre actual y la otra mitad en el siguiente, y la inclusión de unidades curriculares sin previaturas estructurales cuando existe espacio disponible en la carga horaria. Las unidades asignadas se agregan al semestre en construcción y se actualiza la información académica del estudiante.

4. **Actualización del estado del grafo:** Una vez asignadas las unidades curriculares de un semestre, se incrementa el semestre actual y se ajustan

los valores de tiempo de inicio más temprano y de holgura de las unidades curriculares restantes. También se aplican modificaciones sobre las aristas condicionales (por ejemplo, aquellas cuyo peso depende del semestre actual), lo que puede afectar la viabilidad del plan.

5. **Repetición hasta finalización:** El proceso se repite semestre a semestre hasta que no queden más unidades curriculares por cursar, ni pendientes sin previaturas. El resultado es un plan de cursado completo, adaptado a las características académicas y preferencias del estudiante, que respeta todas las restricciones del sistema de previaturas.

Componentes internos

El algoritmo de planificación por semestre está construido sobre una serie de componentes estructurales que permiten representar, analizar y modificar el grafo de unidades curriculares, así como evaluar el cumplimiento de restricciones académicas. A continuación, se describen los principales módulos y funciones utilizadas:

- **Ordenamiento topológico:** Se utiliza una variante clásica del algoritmo de recorrido en profundidad (DFS) (Kleinberg y Éva Tardos, 2006) para obtener un ordenamiento topológico del grafo. Esta operación es indispensable para asegurar que las unidades curriculares se planifiquen en un orden válido respecto a sus previaturas.
- **Cálculo del camino crítico:** Se implementa el algoritmo de Critical Path Method para obtener los siguientes valores para cada unidad curricular:
  - ES (Early Start): semestre más temprano en el que puede cursarse una unidad curricular.
  - LS (Late Start): semestre más tardío en el que puede cursarse sin afectar la duración mínima.
  - Holgura: diferencia entre LS y ES. Las unidades curriculares con holgura cero pertenecen al camino crítico.

  Estos valores se utilizan para priorizar las unidades curriculares más sensibles temporalmente y mejorar la eficiencia del plan.
- **Evaluación de restricciones:** El algoritmo evalúa múltiples tipos de restricciones:
  - Previaturas estructurales: se verifican utilizando el sistema de previaturas modelado en JSON, con reglas lógicas (AND, OR, NOT, SOME).
  - Créditos mínimos aprobados: tanto a nivel de plan como por grupo específico.

- Semestre de dictado: cada unidad curricular incluye información sobre en qué semestre se ofrece.

Funciones auxiliares como `cumplePreviaturas`, `seDictaEsteSemestre` y `actualizarInformacionEstudiante` permiten aplicar y actualizar estas restricciones dinámicamente.

- **Planificación semestral:** La función principal del algoritmo recorre los semestres proyectados en orden, construyendo iterativamente el plan:
  - Selecciona unidades disponibles según holgura y cantidad de créditos.
  - Controla la inclusión de unidades curriculares anuales.
  - Evalúa restricciones y actualiza el estado del grafo.
  - Agrega unidades curriculares sin previaturas cuando existen créditos disponibles.

  En caso de que se requieran reconfiguraciones dinámicas del grafo (por ejemplo, aristas condicionales que cambian según el semestre actual), se interrumpe la ejecución y se reintenta desde el comienzo mediante la función **scheduleFinal**.

Limitaciones y mejoras futuras

El algoritmo de scheduling no garantiza una distribución equitativa de créditos entre semestres; generalmente concentra la mayor parte en los iniciales, dejando los finales con menor carga. Este comportamiento puede ser aceptable según el criterio de planificación que se priorice. Asimismo, dentro de las unidades disponibles para cada semestre, que ya son ordenadas por holgura (priorizando aquellas críticas), podría aplicarse un ordenamiento adicional solo entre las unidades con la misma holgura, por ejemplo priorizando aquellas que maximicen créditos. Esto permitiría que se tenga un control más fino de la distribución de la carga académica sin que se vea afectada la factibilidad del plan.

Manejos especiales

El algoritmo de planificación por semestre contempla diversas situaciones particulares que no pueden ser resueltas únicamente mediante un análisis estático del grafo de unidades curriculares. Estos casos especiales requieren lógica adicional durante la ejecución para garantizar un resultado que cumpla con las restricciones.

En primer lugar tenemos las unidades curriculares anuales que representan un caso especial debido a su duración extendida a lo largo de dos semestres consecutivos. Para manejarlas correctamente, el algoritmo divide su carga de créditos en dos mitades iguales, luego asigna la primera mitad en el semestre actual y reserva la segunda mitad para el semestre siguiente, evita que otras

unidades curriculares anuales sean asignadas en paralelo si no hay suficiente cantidad de créditos y finalmente considera la segunda mitad como ya comprometida, impidiendo su uso por otras unidades curriculares en ese semestre.

Por otro lado, existen relaciones entre unidades curriculares que dependen del semestre en el que se cursan. En estos casos, el peso de la arista que representa la previatura puede variar.

Como se dijo anteriormente, un *valor de 3* indica que el peso debe resolverse dinámicamente durante la planificación. El algoritmo evalúa, en tiempo de ejecución, si la unidad destino se dicta en semestre impar o par. En función de esta evaluación, se asigna un peso de 1 (mismo semestre) o 2 (salto de semestre) a la arista correspondiente.

Si se detecta la presencia de una arista condicional durante la planificación, el algoritmo interrumpe el proceso y reinicia la ejecución ajustando las aristas con los pesos correctos.

Además, algunas unidades curriculares no tienen previaturas explícitas representadas en el grafo, pero sí pueden tener requisitos en forma de créditos mínimos por grupo o totales. Estas unidades son gestionadas aparte mantiénendolas en un conjunto auxiliar, separado del DAG. Se evalúa su inclusión en cada semestre solo si hay espacio dentro del límite de créditos y se verifica que el estudiante cumpla con las previaturas requeridas según el sistema de reglas del plan de estudios. Esta estrategia permite aprovechar al máximo la carga disponible en cada semestre sin comprometer la validez del plan de cursado.

Finalmente, cuando la planificación se ve interrumpida por la necesidad de resolver una arista condicional, el algoritmo devuelve un estado inválido. Para resolver esta situación, se utiliza la función `scheduleFinal` que encapsula la planificación y ejecuta múltiples intentos hasta obtener una solución válida, donde, en cada intento se ajustan las aristas condicionales en función del semestre efectivamente asignado. Este enfoque garantiza convergencia siempre que exista una solución posible.

Salida del algoritmo

El resultado del algoritmo es una planificación académica semestral detallada, compuesta por una secuencia ordenada de bloques que representan los futuros semestres del estudiante. Cada bloque contiene información sobre las unidades curriculares asignadas, la carga en créditos correspondiente y una etiqueta descriptiva del período.

La salida principal del algoritmo es un plan de cursado que especifica, para cada semestre:

- **Semestre:** número entero que indica la posición relativa del semestre dentro del plan (por ejemplo, 1 para el primer semestre planificado, 2 para el segundo, etc.).
- **Unidades curriculares:** conjunto de unidades curriculares asignadas a ese semestre. Estas cumplen con todas las restricciones curriculares esta-

blecidas, incluyendo previaturas, requisitos de dictado y condiciones de créditos acumulados.

- **Créditos:** suma total de créditos correspondientes a las unidades curriculares planificadas para ese semestre. Este valor está acotado por el umbral de carga definido en la configuración del algoritmo.
- **Etiqueta descriptiva:** texto identificador del semestre, indicando si se trata del primer o segundo semestre del año, así como el año proyectado (por ejemplo: "1er semestre 2025").

Esta planificación constituye una trayectoria académica completa, organizada por semestre, que permite al estudiante visualizar de forma clara y estructurada el camino a seguir hasta la finalización de su carrera.

Cabe destacar que el plan generado presenta una alta similitud con la trayectoria sugerida por la Facultad de Ingeniería, respetando tanto la secuencia natural de las unidades curriculares como los momentos típicos de incorporación de unidades curriculares optativas. Además, la planificación resultante es realista y factible de implementar, al considerar restricciones de carga horaria, requisitos curriculares y oferta semestral. Esto valida la efectividad del algoritmo como herramienta de apoyo concreto a la toma de decisiones académicas por parte de los estudiantes y personal de la facultad.

Ejemplo de ejecución

A modo ilustrativo, se presenta un caso concreto de ejecución del algoritmo de planificación por semestres, tomando como punto de partida a un estudiante que ha completado todas las unidades curriculares correspondientes al primer año del plan de estudios. Esto incluye las siguientes aprobaciones:

- **1er semestre:**
  1. Geometría y Álgebra Lineal 1 (GAL1)
  2. Matemática Discreta 1
  3. Programación 1
- **2do semestre:**
  1. Cálculo Diferencial e Integral en una Variable (CDIV)
  2. Geometría y Álgebra Lineal 2 (GAL2)
  3. Matemática Discreta 2
  4. Programación 2

El estudiante define como parámetros de entrada:

- **Semestre de inicio:** 1er semestre
- **Carga horaria por semestre:** 40 créditos

A partir de esta configuración, el algoritmo genera el plan de cursado representado en la tabla 6.1.

Semestre	Créditos	Unidades curriculares asignadas
1er semestre 2026	40	Programación 4, Lógica, Cálculo Diferencial e Integral en Varias Variables
2do semestre 2026	37	Programación 3, Arquitectura de Computadoras, Probabilidad y Estadística
1er semestre 2027	44	Teoría de Lenguajes, Sistemas Operativos, Introducción a la Investigación de Operaciones, Física 1
2do semestre 2027	42	Taller de Programación, Fundamentos de Bases de Datos, Redes de Computadoras
1er semestre 2028	43	Introducción a la Ingeniería de Software, Programación Funcional, Taller de Robótica Educativa, Física 2, Principios y Fundamentos del Proceso Personal de Software
2do semestre 2028	41	Proyecto de Ingeniería de Software, Métodos Numéricos, Aplicaciones del Álgebra Lineal, Políticas Científicas en Informática, Introducción al Negocio del Software
1er semestre 2029	40	Computación de Propósito General en Unidades de Procesamiento Gráfico, Procesamiento de Lenguaje Natural, Métodos Montecarlo, Administración General para Ingenieros, Ciudadanía en Entornos Digitales
2do semestre 2029	40	Proyecto de Grado (1ra mitad), Construcción Formal de Programas, Análisis y Diseño de Algoritmos Distribuidos en Redes, Práctica de Administración
1er semestre 2030	25	Proyecto de Grado (2da mitad), Administración y Seguridad de Sistemas
2do semestre 2030	36	Ecuaciones Diferenciales, Fundamentos de Aprendizaje Automático y Reconocimiento de Patrones, Control de Calidad, Simulación de Eventos Discretos

Tabla 6.1: Planificación generada por el algoritmo

Este ejemplo muestra una planificación que respeta estrictamente todas las restricciones curriculares y de créditos, asegurando un plan de estudios eficiente mediante el cual obtener el egreso. Se observa además una alta coincidencia con la trayectoria sugerida por la Facultad de Ingeniería, aunque con pequeñas variaciones en el semestre de cursado de algunas unidades curriculares no críticas

(por ejemplo, Física 1 o Métodos Numéricos).

Además, se evidencia cómo el algoritmo comienza a incluir unidades curriculares optativas a partir de los semestres intermedios (especialmente desde 2028 en adelante), lo cual también es coherente con la progresión esperada de la curricula sugerida.

Este ejemplo valida empíricamente la utilidad del algoritmo como herramienta de apoyo para la planificación académica personalizada.

Capítulo 7

Pruebas y validaciones

En esta sección se describen las diferentes etapas de prueba y validación llevadas a cabo para evaluar el correcto funcionamiento y la utilidad del sistema desarrollado. Se presentan los casos de prueba implementados, ejemplos concretos de planificación generados por la herramienta, comparaciones con métodos tradicionales de planificación, y la retroalimentación obtenida a partir de la interacción con futuros usuarios. El objetivo fue evidenciar la eficacia de la aplicación en escenarios reales, corregir errores, y detectar áreas de mejora.

7.1. Pruebas realizadas

Con el objetivo de verificar el correcto funcionamiento del sistema en diferentes situaciones reales, se realizaron pruebas manuales utilizando escolaridades representativas de distintos perfiles de estudiantes. Se buscó cubrir tanto casos típicos como escenarios borde, para asegurar que el sistema pueda adaptarse a trayectorias académicas diversas. Además, se alternaron otros factores, como la cantidad de créditos seleccionada a realizar por semestre o el semestre de inicio de la planificación.

Entre los casos evaluados se incluyeron escolaridades con unidades curriculares cursadas en el interior del país, lo que permitió identificar situaciones particulares donde las unidades curriculares no coinciden exactamente con las de Montevideo, pero son equivalentes, al menos parcialmente. Esto puso a prueba la capacidad del sistema para reconocer equivalencias y modelar correctamente trayectorias alternativas. También se consideraron estudiantes próximos a finalizar la carrera, con el fin de validar si el sistema puede generar una propuesta de cursado coherente para los últimos semestres. Del mismo modo, se analizaron casos de estudiantes con muy poco avance, que recién comienzan la carrera o que aún no han aprobado ninguna unidad curricular, con el objetivo de corroborar que el sistema sugiera trayectorias razonables desde el inicio.

Otro aspecto evaluado fue el cálculo de créditos por área, donde se revisó que el sistema contabilizara correctamente los créditos obtenidos en cada una de las áreas del plan de estudios, considerando las reglas específicas de cada carrera, como la obligatoriedad de ciertas áreas o los mínimos requeridos. Asimismo, se incluyeron pruebas con diferentes combinaciones de unidades curriculares optativas y electivas, verificando que el sistema identifica correctamente el tipo de cada unidad curricular y contabiliza sus créditos de forma adecuada.

También se contemplaron casos de cambios de plan de estudios, como el pasaje de un plan anterior al actual, comprobando la validez de las equivalencias y el impacto que estas generan sobre la trayectoria sugerida. Además, se realizaron pruebas cargando escolaridades en formato PDF, para verificar que el sistema interpreta correctamente el archivo. Como alternativa ante posibles errores de detección automática o falta de reconocimiento, se probó la selección manual de las unidades curriculares realizadas, lo que permitió evaluar la flexibilidad del sistema en la carga de información académica.

Finalmente, se llevaron a cabo pruebas para verificar la generación de planes de carrera a partir de una selección manual de unidades curriculares realizada por el propio usuario. Esto permitió confirmar que, independientemente de la combinación elegida, el sistema siempre respetaba la inclusión de las unidades obligatorias y sugería un avance razonable hacia la finalización de la carrera.

Este conjunto de pruebas contribuyó a validar el comportamiento del sistema en contextos diversos y a detectar oportunidades de mejora en el manejo de excepciones y trayectorias no tradicionales.

7.2. Ejemplos de planificación de carrera

A continuación se presentan ejemplos de planificaciones de carrera generadas por el sistema para distintos perfiles de prueba. Estos ejemplos permiten ilustrar el funcionamiento del algoritmo de generación de planes, mostrando cómo se adaptan las propuestas a las características particulares de cada caso.

Los perfiles utilizados para estas pruebas contemplan distintos niveles de avance en la carrera, desde estudiantes que recién comienzan hasta aquellos que se encuentran próximos a egresar. Esto permite verificar cómo se adapta la planificación sugerida a cada situación.

Se incluyen tablas con las planificaciones generadas, acompañadas de una breve descripción del perfil correspondiente y de las decisiones tomadas por el sistema. Entre ellas se destacan la inclusión de unidades curriculares obligatorias no realizadas, la distribución equilibrada de créditos entre semestres y la consideración de unidades seleccionadas manualmente por el usuario.

7.2.1. Perfil 1: Estudiante que inicia la carrera

Descripción del perfil: El estudiante esta iniciando la carrera, por lo que aun no ha cursado ninguna unidad curricular, desea hacer 40 créditos por semestre.

Decisiones del sistema: En la planificación generada se incluyeron todas las unidades obligatorias del plan de estudios. La carga académica fue distribuida en 12 semestres, con un promedio de 40 créditos por semestre. Posteriormente, una vez incorporadas las obligatorias, se seleccionaron de forma aleatoria las unidades curriculares faltantes con el fin de completar los requisitos necesarios.

Resultado: Los resultados se presentan en la Tabla 7.1. Como se puede observar, el plan sugerido para completar la carrera de la manera más eficiente posible, dadas las unidades curriculares seleccionadas aleatoriamente y manteniendo un máximo de 40 créditos por semestre, más el 10 % de holgura configurado (lo que eleva el máximo a 44 créditos), resulta en una duración total de 12 semestres. Cabe destacar que debido al límite de 44 créditos por semestre, no es posible adelantar unidades curriculares como Pasantía o Proyecto de grado a semestres anteriores, ya que esto excedería la carga máxima permitida. En este ejemplo particular, la selección aleatoria llevó a una trayectoria de 12 semestres; sin embargo, con un ajuste en la elección de las unidades iniciales (por ejemplo, sustituyendo Algoritmos Evolutivos por Fundamentos de Programación Entera) sería posible reducir un semestre.

En los últimos semestres se aprecia una reducción considerable en la cantidad de créditos asignados. Esta disminución se explica por el enfoque *greedy* del algoritmo, que procura asignar la mayor cantidad posible de créditos en cada semestre sin sobrepasar la carga máxima definida por el usuario más la holgura del 10 % configurada. Como resultado, las unidades curriculares tienden a concentrarse en las etapas iniciales, dejando menos cursos disponibles para los semestres finales.

Este comportamiento puede considerarse razonable en tanto refleja una práctica habitual de los estudiantes, quienes suelen concentrar mayor carga académica en las etapas iniciales de la carrera y luego cursar menos unidades en la fase final. No obstante, también constituye un aspecto a perfeccionar, ya que en algunos casos podría generar trayectorias con una distribución poco equilibrada. En ese sentido, se plantea como una línea de trabajo futuro explorar mecanismos de ajuste que eviten reducciones tan pronunciadas en los semestres finales, manteniendo un balance más uniforme en la carga académica.

7.2.2. Perfil 2: Estudiante finalizando la carrera

Descripción del perfil: El estudiante ha realizado la mayoría de las unidades curriculares necesarias, le faltan 55 créditos para recibirse, y desea hacer 30 créditos por semestre.

Semestre	Unidad Curricular	Créditos
1er semestre 2026 (41 créditos)	Programación 1	10
	Geometría y Álgebra Lineal 1	9
	Matemática Discreta 1	9
	Cálculo Dif. e Integral en Una Variable	13
2do semestre 2026 (43 créditos)	Programación 2	12
	Matemática Discreta 2	9
	Cálculo Dif. e Integral en Varias Variables	13
	Geometría y Álgebra Lineal 2	9
1er semestre 2027 (43 créditos)	Lógica	12
	Programación 4	15
	Probabilidad y Estadística	10
	Taller de Robótica Educativa	6
2do semestre 2027 (43 créditos)	Programación 3	15
	Arquitectura de Computadoras	10
	Métodos Numéricos	8
	Física 1	10
1er semestre 2028 (44 créditos)	Sistemas Operativos	12
	Teoría de Lenguajes	12
	Int. a la Investigación de Operaciones	10
	Física 3	10
2do semestre 2028 (42 créditos)	Fundamentos de Bases de Datos	15
	Taller de Programación	15
	Redes de Computadoras	12
1er semestre 2029 (41 créditos)	Int. a la Ingeniería de Software	10
	Programación Lógica	10
	Programación Funcional	10
	Taller de Sistemas Ciber-físicos	6
	Administración General para Ingenieros	5
2do semestre 2029 (43 créditos)	Proyecto de Ingeniería de Software	15
	Políticas Científicas en Inf. y Comp.	3
	Economía	7
	Redes Ópticas	10
	Ingeniería de Software Basada en Evidencias	8
1er semestre 2030 (35 créditos)	Fundamentos de Seguridad Informática	12
	Taller de Sistemas Empresariales	15
	Fundamentos de la Web Semántica	8
2do semestre 2030 (37 créditos)	Taller de Seguridad Informática	10
	Álgebra Lineal Numérica	9
	Control de Calidad	8
	Algoritmos Evolutivos	10
1er semestre 2031 (25 créditos)	Proyecto de Grado	15
	Pasantía (Computación)	10
2do semestre 2031 (15 créditos)	Proyecto de Grado	15

Tabla 7.1: Plan sugerido para el perfil 1 con una carga de 40 créditos por semestre

Semestre	Unidad Curricular	Créditos
1er semestre 2026 (28 créditos)	Proyecto de Grado	15
	Fundamentos de la Robótica Autónoma	7
	Taller de Robótica Educativa	6
2do semestre 2026 (29 créditos)	Proyecto de Grado	15
	Ingeniería de Ontologías	6
	Integración de datos	8

Tabla 7.2: Plan sugerido para el perfil 2 con una carga de 30 créditos por semestre

Decisiones del sistema: En la planificación generada se incluyeron todas las unidades curriculares obligatorias del plan de estudios, distribuyendo la carga en 2 semestres con un máximo de 33 créditos por semestre (30 créditos definidos por el estudiante más un 10 % de holgura configurado). Además, se seleccionaron unidades curriculares optativas hasta alcanzar la cantidad de créditos necesaria para la finalización de la carrera.

Resultado: Los resultados se presentan en la Tabla 7.2. Como era de esperarse, la única unidad curricular obligatoria agregada fue Proyecto de grado, dado que era la única pendiente. Además, se incluyeron tres unidades curriculares optativas adicionales que, si bien hacen que la planificación supere ligeramente el mínimo requerido, permiten alcanzar y garantizar el cumplimiento de los 450 créditos necesarios para la obtención del título.

7.3. Comparación con currícula sugerida estática

En esta sección se realiza una comparación entre el plan de cursado sugerido por el sistema desarrollado y la currícula sugerida estática actualmente disponible para la carrera de Ingeniería en Computación. Esta comparación permite visualizar las diferencias en cuanto a la flexibilidad, personalización y cobertura del avance académico entre ambas propuestas.

La trayectoria estática (tabla 7.3) define una planificación fija con un total de 329 créditos distribuidos en diez semestres. En esta propuesta, los créditos se distribuyen de manera heterogénea, con semestres que van desde los 15 hasta los 47 créditos. El plan está concebido únicamente para cumplir con los requisitos mínimos obligatorios de la carrera, sin incluir asignaturas optativas ni contemplar posibles caminos alternativos. Además, no ofrece mecanismos de adaptación al avance real del estudiante ni a sus preferencias personales. Esto significa que, si por algún motivo un estudiante se atrasa, decide cursar menos créditos en determinados semestres o prefiere realizar otras unidades curriculares distintas a las previstas, la trayectoria sugerida por la facultad deja de ser

factible y pierde parte de su utilidad práctica como guía académica.

La ventaja de la propuesta desarrollada en este proyecto es generar una planificación dinámica, adaptada al progreso individual del estudiante y a sus restricciones, sin imponer necesariamente una carga fija de 45 créditos por semestre.

Por otro lado, el sistema desarrollado genera un plan personalizado basado en la escolaridad del estudiante, el semestre de inicio y la carga académica deseada por período. Este plan busca cumplir con los 450 créditos necesarios para egresar, seleccionando automáticamente las unidades curriculares obligatorias y complementándolas con optativas según disponibilidad, requisitos y compatibilidad. A diferencia de la trayectoria estática, la propuesta se adapta a la situación académica concreta de cada estudiante, incorpora de manera automática optativas para alcanzar el total requerido, considera los prerrequisitos y permite generar trayectorias alternativas válidas. Asimismo, brinda al usuario la posibilidad de seleccionar manualmente unidades curriculares de interés, combinando flexibilidad con el cumplimiento de los criterios de egreso.

Como se puede observar en la Tabla 7.1, el sistema sugiere una trayectoria de 12 semestres (para el caso de prueba considerado), completando los 450 créditos necesarios para la titulación. En comparación, la trayectoria estática no contempla esta posibilidad, esperando que el estudiante seleccione optativas por su cuenta.

Esta flexibilidad y capacidad de adaptación del sistema permite generar trayectorias viables incluso en situaciones no contempladas por el plan estático, como por ejemplo:

- Estudiantes con unidades curriculares cursadas fuera del orden tradicional.
- Créditos equivalentes por reválidas.
- Preferencias personales en la carga horaria o tipo de unidades curriculares.

Si bien la planificación generada por nuestro sistema resultó en una duración de seis años frente a los cinco años de la trayectoria sugerida por la facultad, esta diferencia puede ajustarse calibrando el máximo de créditos permitidos por semestre. Por ejemplo, al incrementar dicho valor a 45 créditos, el sistema produciría una planificación en cinco años. Además, la trayectoria obtenida mantiene una organización muy similar a la propuesta estática en cuanto al orden de las unidades curriculares semestre a semestre, incorporando al mismo tiempo las asignaturas optativas necesarias. De esta forma, se ofrece al usuario una planificación completa y adaptada a su situación particular.

En comparación con la planificación manual basada en la trayectoria estática, el sistema simplifica considerablemente el proceso, permitiendo que el estudiante seleccione las unidades curriculares que desea cursar y complementando automáticamente la elección con optativas cuando sea necesario. Asimismo, organiza las unidades curriculares por semestre teniendo en cuenta la carga horaria

Semestre	Unidades Curriculares	Créditos
1er semestre (32 créditos)	Créditos obtenidos en prueba inicial	4
	Geometría y Álgebra Lineal 1	9
	Matemática Discreta 1	9
	Programación 1	10
2do semestre (43 créditos)	Cálculo Diferencial e Integral en Una Variable	13
	Geometría y Álgebra Lineal 2	9
	Matemática Discreta 2	9
	Programación 2	12
3er semestre (40 créditos)	Cálculo Diferencial e Integral en Varias Variables	13
	Lógica	12
	Programación 4	15
4to semestre (43 créditos)	Probabilidad y Estadística	10
	Programación 3	15
	Arquitectura de Computadoras	10
	Métodos Numéricos	8
5to semestre (39 créditos)	Introducción a la Investigación de Operaciones	10
	Sistemas Operativos	12
	Teoría de Lenguajes	12
	Administración General para Ingenieros (*)	5
6to semestre (47 créditos)	Fundamentos de Bases de Datos	15
	Taller de Programación	15
	Redes de Computadoras	12
	Práctica de Administración para Ingenieros (*)	5
7mo semestre (30 créditos)	Taller Introductorio de Ingeniería de Software	10
	Programación Funcional o Programación Lógica	10
	Física 1 (*)	10
8vo semestre (25 créditos)	Proyecto de Ingeniería de Software	15
	Economía (*)	7
	Políticas Científicas en Inf. y Comp. (*)	3
9no semestre (15 créditos)	Proyecto de Grado	15
10mo semestre (15 créditos)	Proyecto de Grado	15
Total		**329**

Tabla 7.3: Trayectoria sugerida estática para la carrera de Ingeniería en Computación

disponible y el estado académico actual, logrando así una propuesta personalizada y más ajustada a las condiciones reales de cursado.

En resumen, el sistema propuesto permite automatizar y personalizar la planificación académica de forma más completa y realista que la trayectoria estática, facilitando la toma de decisiones y reduciendo errores o cuellos de botella derivados de la planificación manual.

7.4. Validación con futuros usuarios

Con el objetivo de validar la herramienta desarrollada y detectar posibles errores y oportunidades de mejora, se elaboró un formulario de Google destinado a estudiantes de la Facultad de Ingeniería. El formulario incluía tres preguntas abiertas, que debían responderse luego de interactuar con el sistema:

- *¿Encontraste algún error?*
- *¿Hay algo que te gustaría que agreguemos?*
- *Dejanos tu comentario*

A partir de las 20 respuestas recibidas, se pudieron identificar diversas observaciones, sugerencias y comentarios generales. En lo que respecta a los errores reportados, se mencionaron fallas en la carga de escolaridad, inconsistencias en el manejo de algunas unidades curriculares (como Arquitectura de Computadoras y Tutorías Entre Pares) y problemas en la sección de búsqueda de cursos opcionales. También se destacaron casos específicos, como cursos equivalentes que no estaban correctamente modelados o reválidas que otorgaban créditos de manera distinta. El sistema fue puesto a prueba en contextos que inicialmente no se habían considerado, lo que permitió exponer realidades menos comunes. Estos hallazgos evidenciaron además ciertas limitaciones derivadas de la falta de actualización de los sistemas de donde se toma la información, como Bedelías, que aún no refleja correctamente algunos cursos recientemente modificados o eliminados. A partir de estas observaciones se corrigieron errores puntuales y se contemplaron nuevas condiciones, ampliando la robustez del sistema.

En cuanto a las sugerencias de mejora, varios estudiantes propusieron la incorporación de funcionalidades adicionales, tales como integrar unidades curriculares dictadas por otras facultades, agregar los módulos de Taller y Extensión, o permitir fijar cursos manualmente en semestres específicos. Algunas de estas recomendaciones fueron implementadas directamente, mientras que otras se consideraron como posibles líneas de mejora futura, dado que amplían el alcance y la adaptabilidad del sistema frente a trayectorias académicas diversas.

Por último, los comentarios generales fueron mayoritariamente positivos. Se destacó la utilidad de la herramienta, especialmente por su flexibilidad al

permitir planificaciones más realistas frente a situaciones atípicas. Un ejemplo de esto son los casos en los que, por resoluciones especiales, un estudiante pudo cursar y aprobar una unidad curricular sin haber aprobado todas sus previas de manera formal.

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se presentan las conclusiones del desarrollo del proyecto, una evaluación crítica de los resultados, las principales dificultades encontradas y una descripción de posibles extensiones y mejoras al sistema.

8.1. Conclusiones generales

El presente proyecto tuvo como objetivo central el desarrollo de una plataforma web para facilitar a los estudiantes de la Facultad de Ingeniería la generación de planes de cursado personalizados. Esta herramienta permite centralizar información crítica relacionada con unidades curriculares, sistemas de previaturas y grupos, la cual previamente se encontraba dispersa en diferentes sistemas y formatos.

Entre los principales resultados obtenidos, se destaca la implementación exitosa de un sistema interactivo que proporciona al estudiante información actualizada y clara sobre su avance académico, permitiendo la planificación tanto a corto como a largo plazo. Asimismo, el sistema logra generar automáticamente trayectorias de carrera, optimizando la toma de decisiones y reduciendo la necesidad de asesoramiento manual repetitivo.

La incorporación de funcionalidades clave, como la importación automatizada del progreso académico mediante escolaridades en formato PDF, representa una funcionalidad útil para el usuario. Por otra parte, la representación del sistema de previaturas mediante estructuras de datos en formato JSON facilitó notablemente la manipulación y gestión de dicha información.

En términos de impacto, la plataforma promete ser valiosa para estudiantes que necesitan ayuda para planificar su carrera y elegir unidades curriculares para cursar, teniendo en cuenta restricciones específicas como la carga horaria semestral. La automatización ofrecida por el sistema permite concentrar esfuerzos en asesorías especializadas y gestionar con mayor rapidez las solicitudes

académicas.

En resumen, el desarrollo de esta plataforma ha cumplido satisfactoriamente con los objetivos planteados, proporcionando una solución integral, eficiente y funcional, que responde adecuadamente a las necesidades identificadas en la fase inicial del proyecto y establece una sólida base para futuras mejoras y ampliaciones del sistema.

8.2. Desafíos afrontados

El desarrollo de este proyecto representó un desafío constante, con problemas nuevos que debieron ser afrontados durante el proceso. A lo largo del proyecto surgieron múltiples desafíos, de los cuales se identificaron tres particularmente demandantes: la obtención de datos sobre unidades curriculares como sus previaturas, grupos y semestres en los que se dictan las mismas. En segundo lugar, la implementación del algoritmo de generación de planes de carrera (scheduling), que resultó especialmente complejo debido a la gran cantidad de personalizaciones y adaptaciones necesarias. Por último, la extracción del progreso académico a partir de escolaridades en formato PDF.

8.2.1. Extracción y representación de datos

La obtención de los datos relativos a unidades curriculares como previaturas, grupos y semestres significó un desafío desde las etapas iniciales del proyecto. En un principio, el enfoque inicial consistía en obtener dicha información desde una copia de la base de datos de SECIU. Sin embargo, la complejidad y las inconsistencias en dicha base de datos tornaron inviable su utilización, dado que cualquier error en las previaturas o ausencia de unidades curriculares impactaría significativamente en la usabilidad y realismo del sistema. Desde las etapas tempranas se comprendió la importancia de contar con datos altamente fiables, por lo que, a mediados del proyecto, se adoptó una nueva estrategia basada en técnicas de web scraping sobre el portal web de Bedelías. Este nuevo enfoque resultó vital para el éxito del proyecto, ya que permitió acceder a datos precisos y actualizados, y automatizar su extracción, factor clave para la sostenibilidad futura del sistema.

8.2.2. Algoritmo de generación de planes

El segundo componente crítico para el éxito del proyecto fue la generación de planes factibles y útiles para los estudiantes. El desarrollo del algoritmo de generación de planes supuso constantes ajustes y adaptaciones. Partiendo de técnicas como construcción de grafos, camino crítico y algoritmos de scheduling, se realizaron numerosas iteraciones y personalizaciones específicas para cumplir con los requerimientos particulares del proyecto. Estas adaptaciones permitieron implementar un algoritmo eficaz, capaz de ofrecer trayectorias académicas

valiosas para los estudiantes. Las personalizaciones específicas y detalles técnicos del algoritmo se describen en profundidad en la sección correspondiente de diseño e implementación.

8.2.3. Extracción del progreso académico

Finalmente, otro desafío significativo fue la extracción del avance académico a partir de escolaridades en formato PDF. Este desafío presentó múltiples dificultades, comenzando con la complejidad del formato del documento de la escolaridad, que contiene mucha información extra, dificultando su precisa y limpia extracción. El proceso implicó numerosas iteraciones y una constante prueba y error.

Adicionalmente, a comienzos de 2025 la UDELAR culminó el proceso de modificación de la escala de notas, pasando de un formato numérico a uno conceptual. Este cambio implicó una transformación sustancial en el formato de la escolaridad y requirió un retrabajo considerable para adaptar el algoritmo, volviendo a enfrentar desafíos ya resueltos. Sin embargo, la experiencia adquirida en la primera implementación permitió abordar estos problemas con mayor eficacia y reducir de forma significativa el esfuerzo necesario en comparación con la etapa inicial del proyecto.

8.3. Trabajo futuro

Si bien el proyecto resultó exitoso, generando una plataforma robusta y útil que implementó la mayoría de las funcionalidades propuestas, se identificaron múltiples áreas de mejora tanto en términos de funcionalidades como en la arquitectura del sistema. En primer lugar, dado el alcance del proyecto, se consideró únicamente la carrera de Ingeniería en Computación; sería conveniente que futuras iteraciones extiendan el sistema para abarcar múltiples carreras. Además, el algoritmo de generación de planes presenta una limitación considerable en la selección automática de unidades curriculares, ya que actualmente la selección es aleatoria y no considera aspectos tales como las preferencias individuales del estudiante o la popularidad de las unidades curriculares. Por último, en cuanto a la arquitectura del sistema, se identificó como mejora significativa la implementación de una base de datos relacional en lugar del actual almacenamiento basado en archivos JSON.

8.3.1. Extensión para múltiples carreras

Desde las etapas iniciales del proyecto se consideró que, para garantizar su éxito, era necesario enfocar los esfuerzos en una única carrera. Se seleccionó la carrera de Ingeniería en Computación debido a la experiencia y afinidad existente con la misma, buscando beneficiar a estudiantes que atraviesan desafíos académicos similares. Sin embargo, en todo momento se procuró desarrollar el sistema con una estructura extensible hacia otras carreras. Aunque la tarea

no es trivial, una mejora relevante a futuro sería ampliar la plataforma para dar soporte a múltiples carreras, incrementando significativamente su utilidad y beneficiando a un mayor número de estudiantes y personal académico.

8.3.2. Mejora del algoritmo de selección de unidades curriculares

Como se mencionó anteriormente, debido a restricciones temporales, el algoritmo utilizado para seleccionar las unidades curriculares en la generación de planes se basa actualmente en una selección aleatoria. Una mejora sustancial consistiría en considerar factores clave como las preferencias individuales del estudiante y las unidades curriculares de mayor popularidad o valoración por parte de los estudiantes. Esta mejora incrementaría notablemente el realismo y la utilidad de los planes generados. Asimismo, sería interesante explorar la implementación de modelos basados en machine learning, capaces de predecir con mayor precisión las unidades curriculares más adecuadas según el contexto académico del estudiante. Otra línea futura consiste en contemplar un mayor equilibrio en la distribución de créditos entre semestres pares e impares, incluyendo también las unidades curriculares obligatorias.

8.3.3. Base de datos relacional

En lo que refiere a la arquitectura del sistema, la primera mejora que se recomienda realizar es la incorporación de una base de datos relacional, como PostgreSQL, para almacenar los datos del sistema. Esta implementación representaría una mejora significativa en términos de robustez, escalabilidad y rendimiento de la capa de persistencia. El almacenamiento actual basado en archivos JSON no solo limita la eficiencia y escalabilidad del sistema, sino que también se desvía de las buenas prácticas de desarrollo de software, afectando negativamente la percepción profesional del sistema.

Referencias

Bacchus, F. (s.f.). *Constraint satisfaction problems: Basic introduction.* https://www.cs.toronto.edu/~fbacchus/Presentations/CSP-BasicIntro.pdf. (Universidad de Toronto, Departamento de Ciencias de la Computación)

Curricula Sugerida - Ingenieria en Computación. (s.f.). *Curricula sugerida de ingeniería en computación.* https://eva.fing.edu.uy/pluginfile.php/496157/mod_resource/content/5/TrayectoriaSugerida_2025_Montevideo.pdf.

Jean C. Digitale, M. M. G., Jeffrey N. Martin. (2021). *Tutorial on directed acyclic graphs.* Elsevier. Descargado de https://www.sciencedirect.com/science/article/pii/S0895435621002407

Kelley, J. E. J., y Walker, M. R. (1959). Critical-path planning and scheduling. En *Proceedings of the eastern joint computer conference* (pp. 160–173).

Kleinberg, J., y Éva Tardos. (2006). *Algorithm design.* Boston, MA: Addison Wesley.

Sedgewick, R., y Wayne, K. (2011). *Algoritmos* (4.ª ed.). Madrid: Addison-Wesley.

Sistema de previaturas de Ingeniería en Computación. (s.f.). *Sistema de previaturas de ingeniería en computación.* https://bedelias.udelar.edu.uy/views/public/desktop/consultarSistemaPreviatura/consultarSistemaPreviatura02.xhtml?cid=1.

Anexo A

Historias de Usuario

Este anexo contiene el detalle completo de las historias de usuario relevadas durante la etapa de análisis, organizadas según su prioridad. Cada historia describe una necesidad funcional desde la perspectiva del usuario, acompañada de una breve descripción y criterios de aceptación que guiaron el desarrollo de la funcionalidad.

Clasificación por prioridad

Para organizar las funcionalidades del sistema, se establecieron tres niveles de prioridad:

- **Alta prioridad:** Requerimientos indispensables para que la aplicación cumpla sus objetivos principales. Son necesarios para generar valor desde el primer uso por parte de estudiantes y funcionarios.
- **Media prioridad:** Funcionalidades que complementan y mejoran la experiencia del usuario, pero que no son críticas para el funcionamiento inicial del sistema.
- **Baja prioridad:** Características pensadas para una integración futura (por ejemplo, con sistemas institucionales como SECIU), que no fueron implementadas en esta versión, pero se consideran valiosas a largo plazo.

A.1. Historias de prioridad alta

A.1.1. Marcado manual de unidades curriculares aprobadas

Como usuario, **quiero** poder marcar manualmente unidades curriculares de la carrera seleccionada como aprobadas, **para** realizar planificaciones de currícula a futuro.

Descripción: El usuario necesita marcar manualmente las unidades curriculares aprobadas para que el plan generado refleje su progreso académico real.

Criterios de aceptación:

- El usuario puede seleccionar UCs desde una lista.
- El sistema actualiza el estado como aprobado.
- Se muestra un resumen de UCs marcadas.
- Los cambios se guardan en el perfil del usuario.
- Se confirma la acción con una notificación.

A.1.2. Exportar unidades curriculares aprobadas

Como usuario, **quiero** exportar las unidades curriculares aprobadas, **para** presentarlas ante un funcionario de Facultad.

Descripción: El sistema debe generar un archivo con las UCs aprobadas en formatos como PDF o CSV.

Criterios de aceptación:

- El usuario puede acceder a la opción de exportación.
- Puede elegir el formato y rango de fechas/semestres.
- El sistema genera un archivo legible con la información.
- Se muestra una notificación al finalizar.

A.1.3. Obtener información sobre unidades curriculares

Como usuario, **quiero** obtener información importante sobre las unidades curriculares, **para** tomar decisiones informadas sobre mi plan de estudio.

Descripción: El usuario puede consultar información detallada como créditos, previas requeridas y enlace a EVA, si está disponible.

Criterios de aceptación:

- El sistema muestra nombre, código, créditos y previas.
- La información debe estar actualizada y ser precisa.
- Se muestra un enlace a EVA si está disponible.

A.1.4. Generar currícula sugerida para el siguiente semestre

Como usuario, **quiero** obtener una currícula sugerida para el semestre entrante, **para** planificar mis estudios próximos.

Descripción: El sistema genera una propuesta de cursado según unidades curriculares aprobadas, carga horaria disponible, intereses y electivas deseadas.

Criterios de aceptación:

- El usuario puede ingresar su carga horaria disponible.
- Puede seleccionar áreas de interés y electivas.
- El sistema genera y muestra una propuesta de cursado.
- Se puede descargar la currícula sugerida.

A.1.5. Generar currícula sugerida para toda la carrera

Como usuario, **quiero** obtener una currícula sugerida completa, **para** planificar mis estudios a largo plazo.

Descripción: El sistema genera una propuesta de cursado semestre por semestre basada en la situación académica del usuario.

Criterios de aceptación:

- El usuario puede definir carga horaria e intereses.
- El sistema calcula una distribución equilibrada por semestre.
- Se muestra la propuesta completa y se puede exportar.

A.1.6. Importar unidades curriculares aprobadas

Como usuario, **quiero** importar un archivo con mis UCs aprobadas, **para** cargar mis datos académicos de forma eficiente.

Descripción: El sistema debe permitir cargar un archivo (por ejemplo CSV) con las UCs aprobadas y validarlas antes de confirmarlas.

Criterios de aceptación:

- El usuario puede seleccionar un archivo con formato válido.
- El sistema valida, procesa y muestra una vista previa.
- Se confirma la importación antes de aplicarla.
- Se notifican errores en caso de inconsistencias.

A.2. Historias de prioridad media

A.2.1. Subida de escolaridad en PDF

Como usuario, **quiero** subir la escolaridad en formato PDF, **para** registrar automáticamente mis unidades curriculares aprobadas.

Descripción: El sistema debe permitir cargar un archivo PDF con la escolaridad del estudiante y procesarlo para registrar las unidades curriculares aprobadas, diferenciando cursos y exámenes.

Criterios de aceptación:

- El usuario puede subir un archivo PDF desde una sección dedicada.
- El sistema procesa el PDF y extrae la información relevante.
- Se muestra una vista previa para confirmar el registro.
- Se notifican errores si el archivo es inválido o tiene problemas.

A.2.2. Persistencia local de datos

Como usuario, **quiero** que la aplicación guarde mis datos localmente, **para** evitar perderlos al cerrar o recargar la página.

Descripción: El sistema debe guardar localmente las unidades curriculares aprobadas para asegurar la continuidad en la experiencia de usuario.

Criterios de aceptación:

- Las unidades curriculares aprobadas se guardan en el navegador.
- Los datos persisten al cerrar o recargar la página.
- Se cargan automáticamente al volver a ingresar.

A.2.3. Consulta de unidades curriculares del siguiente semestre

Como usuario, **quiero** consultar las unidades curriculares que se dictarán el próximo semestre, **para** planificar mi inscripción de manera informada.

Descripción: El sistema debe permitir acceder a la lista de unidades curriculares programadas para el próximo semestre, con filtros por créditos y descripciones.

Criterios de aceptación:

- El usuario accede a una sección con las unidades curriculares disponibles.
- Puede filtrar por créditos y consultar información detallada.
- La información debe ser precisa y actualizada.

A.2.4. Rango de créditos por semestre

Como usuario, **quiero** definir un rango de créditos por semestre, **para** planificar mi carga académica acorde a mi disponibilidad.

Descripción: El sistema debe permitir al usuario establecer un mínimo y máximo de créditos para cursar en cada semestre.

Criterios de aceptación:

- El usuario puede ingresar valores de créditos mínimos y máximos.
- El sistema utiliza este rango al generar las propuestas.
- Se aplica una holgura razonable por defecto.

A.2.5. Verificación de requisitos para egreso

Como usuario, **quiero** verificar si cumplo con los requisitos para recibirme, **para** saber qué unidades curriculares o créditos me faltan.

Descripción: El sistema debe comprobar si el usuario cumple con los requisitos de egreso de la carrera: créditos totales, obligatorios, optativos y por área.

Criterios de aceptación:

- Se verifica el cumplimiento de créditos totales y por área.
- Se verifica la aprobación de unidades curriculares obligatorias.
- Se muestra un resumen claro indicando lo que falta o lo que se completó.

A.3. Historias de prioridad baja

A.3.1. Inicio de sesión para usuarios

Como usuario, **quiero** iniciar sesión en la aplicación, **para** acceder a mi información académica.

Descripción: El sistema debe permitir la autenticación del usuario mediante correo electrónico y contraseña u otro método como Google.

Criterios de aceptación:

- El usuario puede iniciar sesión con credenciales válidas.
- El sistema valida las credenciales y redirige al perfil.
- Se muestran mensajes de error en caso de fallos.

A.3.2. Persistencia de datos para usuarios con sesión iniciada

Como usuario autenticado, **quiero** que se guarden mis unidades curriculares aprobadas y currículas creadas, **para** mantener mi progreso entre sesiones.

Descripción: El sistema debe almacenar de forma persistente los datos académicos del usuario autenticado en una base de datos asociada a su perfil.

Criterios de aceptación:

- Las UCs aprobadas y currículas creadas se guardan en el servidor.
- El usuario accede a sus datos al iniciar sesión nuevamente.
- Los datos son accesibles solo para el usuario correspondiente.

A.3.3. Consulta de unidades curriculares aprobadas por cédula

Como usuario o funcionario, **quiero** ingresar una cédula, **para** obtener las unidades curriculares aprobadas del estudiante correspondiente.

Descripción: El sistema debe permitir la consulta de las UCs aprobadas a partir de una cédula de identidad.

Criterios de aceptación:

- El funcionario puede ingresar una cédula en un campo de búsqueda.
- El sistema muestra el listado de UCs aprobadas.
- Se informa si la cédula no corresponde a ningún estudiante.
- Se incluyen nombre de la UC, semestre y calificación.

A.3.4. Visualización de unidades curriculares por fecha de aprobación

Como funcionario, **quiero** ver las UCs aprobadas ordenadas por fecha de aprobación, **para** revisar el progreso académico del estudiante.

Descripción: El sistema debe mostrar las UCs aprobadas ordenadas cronológicamente con opción de exportación.

Criterios de aceptación:

- Las UCs aprobadas se ordenan por fecha de aprobación.
- Se incluyen nombre, fecha y calificación.
- Se puede exportar el listado en formato PDF o Excel.
- Se notifica si la cédula no corresponde a ningún estudiante.

A.3.5. Selección de carrera y perfil

Como usuario, **quiero** seleccionar mi carrera (obligatorio) y perfil (opcional), **para** recibir recomendaciones personalizadas.

Descripción: El sistema debe permitir seleccionar una carrera obligatoriamente y un perfil adicional si aplica, para adaptar la planificación a las características del estudiante.

Criterios de aceptación:

- El usuario puede seleccionar su carrera desde una lista.
- El perfil es opcional y se guarda junto con la carrera.
- El sistema confirma que los datos fueron guardados correctamente.

Anexo B

Instalación y Mantenimiento

Este anexo presenta la información necesaria para la instalación, ejecución, mantenimiento y actualización del sistema desarrollado. Se describen los requisitos previos, la estructura general del proyecto y las instrucciones detalladas para levantar el entorno tanto en desarrollo como en producción utilizando Docker y Docker Compose. Asimismo, se documentan los procedimientos para mantener sincronizados los datos de la aplicación con los sistemas oficiales de la Facultad de Ingeniería, a través de la ejecución de scripts específicos de scraping y generación de datos. Finalmente, se brindan consideraciones generales que garantizan la correcta operación del sistema.

B.1. Requisitos previos

Para ejecutar el entorno de desarrollo o producción, es necesario contar con:

- **Docker y Docker Compose** instalados y corriendo localmente.
- Puertos **5173**, **3000** y **8000** disponibles.
- Conexión a internet para descargar imágenes y dependencias.

B.2. Estructura del proyecto

El repositorio se encuentra organizado en tres servicios principales:

- **frontend:** Interfaz de usuario desarrollada en React + Vite.
- **backend:** API principal desarrollada con Node.js y Express.
- **processor-scraper:** Microservicio desarrollado con Python y FastAPI para scraping y procesamiento de escolaridades.

B.3. Entorno de desarrollo

Levantar el entorno

Desde el directorio raíz del proyecto, ejecutar:

```
docker-compose up
```

Este comando construirá las imágenes necesarias y levantará los tres servicios principales con soporte para hot reload. Los servicios quedarán expuestos en los siguientes puertos:

- frontend: `http://localhost:5173`
- backend: `http://localhost:3000`
- processor-scraper: `http://localhost:8000`

B.4. Entorno de producción

Configuración

El entorno de producción está configurado mediante:

- Dockerfiles separados para desarrollo (Dockerfile) y producción (Dockerfile.prod) por cada servicio.
- Un archivo docker-compose.prod.yml que define la infraestructura productiva.

Servicios adicionales

En producción se añade el servicio certbot, encargado de generar y renovar certificados SSL automáticamente.

Comando de despliegue

Para levantar la aplicación en producción:

```
docker compose -f docker-compose.prod.yml up
```

Puede requerirse una configuración previa de variables de entorno, dominios y certificado SSL antes del primer uso.

B.5. Mantenimiento y actualización de datos

Para mantener la aplicación sincronizada con los datos oficiales del sistema Bedelías, se debe ejecutar un conjunto de scripts que recolectan y procesan la información actualizada de unidades curriculares, grupos y previaturas.

Scraping desde processor-scraper

```
docker-compose exec processor-scraper python -m scripts.scrape_previatures
docker-compose exec processor-scraper python -m scripts.scrape_groups_and_subjects
```

Generación y actualización de datos en backend

```
docker-compose exec backend pnpm generate:unidades-curriculares
docker-compose exec backend pnpm generate:previaturas
docker-compose exec backend pnpm generate:ucs-grupos
docker-compose exec backend pnpm generate:ucs-grupos-actuales
```

Se recomienda seguir este orden para garantizar la consistencia del sistema.

B.6. Consideraciones finales

- Todos los servicios están orquestados vía Docker, por lo tanto, no es necesario instalar dependencias locales ni gestionar entornos virtuales manualmente.
- El sistema fue diseñado para facilitar el desarrollo colaborativo y el despliegue automatizado con mínima configuración.
- Se recomienda documentar cualquier personalización local en un archivo `.env.local` excluido del control de versiones.

Anexo C

Archivos de persistencia

C.1. Archivos de persistencia y estructuras JSON

El backend del sistema utiliza archivos en formato `.json` como mecanismo de persistencia temporal y fuente de datos académicos. Estos archivos se cargan en memoria al iniciar el servidor y se utilizan para representar tanto información estática como estados intermedios.

A continuación se detallan los principales archivos utilizados, su propósito y una muestra de su estructura.

`unidades-curriculares.json`

Propósito: contiene el listado completo de unidades curriculares del plan de estudios, incluyendo su código, nombre, créditos, semestre, etc.

Estructura (fragmento):

```
[
  {
    "nombre": "CALCULO DIF. E INTEGRAL EN VARIAS VARIABLES",
     "codigo": "1062",
     "creditos": 13,
     "nombreGrupoPadre": "MATERIAS BASICAS",
     "codigoGrupoPadre": "2436",
     "nombreGrupoHijo": "MATEMATICA",
     "codigoGrupoHijo": "4083",
     "semestres": ["1","2"]
  },
  {
    "codigo": "1372",
    "nombre": "Programación 1",
    "creditos": 10,
    "nombreGrupoPadre": "BASICO-TEC,TECNICAS E INTEG.",
```

```
    "codigoGrupoPadre": "3215",
    "nombreGrupoHijo": "PROGRAMACION",
    "codigoGrupoHijo": "4580",
    "semestres": ["1", "2"]
  }
]
```

previaturas.json

Propósito: define las reglas de precedencia entre unidades curriculares (previaturas), utilizando estructuras lógicas que pueden incluir operadores `AND`, `OR`, `NOT` y `SOME`.

Estructura (fragmento):

```
"1321": {
    "regla": "AND",
    "previas": [
      {
        "regla": "NOT",
        "previas": {
          "regla": "SOME",
          "cantidad": 1,
          "previas": [
            {
              "regla": "UC",
              "codigo": "CP37",
              "nombre": "PROGRAMACION I (P. 74)",
              "tipoInstancia": "E"
            },
            {
              "regla": "UC",
              "codigo": "CP15",
              "nombre": "PROGRAMACION II",
              "tipoInstancia": "E"
            },
            ...
          ]
        }
      },
      {
        "regla": "SOME",
        "cantidad": 1,
        "previas": [
          {
            "regla": "UC",
```

```
            "codigo": "1373",
            "nombre": "PROGRAMACION 1",
            "tipoInstancia": "C"
          },
          {
            "regla": "UC",
            "codigo": "CENURLN",
            "nombre": "SRN17 - PROGRAMACIÓN I- CIO CT-LHRH",
            "tipoInstancia": "C"
          },
          ...
        ]
      }
   ]
}
```


ucs-grupos.json y ucs-grupos-actuales.json

Propósito: clasificación de las unidades curriculares por grupos.
Estructura (fragmento):


```
{
  "PROGRAMACION": [
        "1353",
        "1349",
        "1347",
        "1316",
        "1350",
        "1340"
    ],
    "ARQUIT, S.OP. Y REDES DE COMP.": [
        "5907",
        "1466",
        "1891",
        "1556",
        "1952",
        "1440",
        "1447",
        "1434",
        "5916"
    ],
    ...
}
```

ucs-primer-semestre.json y ucs-segundo-semestre.json

Propósito: unidades curriculares que actualmente se dictan en cada semestre.

Estructura (fragmento):

```
{
   {
        "codigo": "1944",
        "nombre": "ADMINISTRACIÓN GENERAL PARA INGENIEROS"
    },
    {
        "codigo": "1918",
        "nombre": "ADMINISTRACIÓN Y SEGURIDAD DE SISTEMAS"
    },
    {
        "codigo": "1438",
        "nombre": "APLICACIONES DE TEORÍA DE LA INFORMACIÓN AL PROCESAMIENT"
    },
    {
        "codigo": "1949",
        "nombre": "BASES DE DATOS NO RELACIONALES"
    },
    ...
}
```

trayectoria-sugerida.json

Propósito: trayectoria académica recomendada por la facultad.

Estructura (fragmento):

```
{
   {
    "semestre": 1,
    "unidadesCurriculares": [
      { "codigo": "MI2", "nombre": "MATEMÁTICA INICIAL" },
      { "codigo": "1030", "nombre": "GEOMETRÍA Y ÁLGEBRA LINEAL 1" },
      { "codigo": "1023", "nombre": "MATEMÁTICA DISCRETA 1" },
      { "codigo": "1373", "nombre": "PROGRAMACIÓN 1" }
    ]
  },
  {
    "semestre": 2,
    "unidadesCurriculares": [
      {
        "codigo": "1061",
```

```
        "nombre": "CALCULO DIFERENCIAL E INTEGRAL EN UNA VARIABLE"
      },
      { "codigo": "1031", "nombre": "GEOMETRÍA Y ÁLGEBRA LINEAL 2" },
      { "codigo": "1026", "nombre": "MATEMÁTICA DISCRETA 2" },
      { "codigo": "1321", "nombre": "PROGRAMACIÓN 2" }
    ]
  },
  ...
}
```

requisitos-titulo.json

Propósito: información sobre requisitos específicos para obtener el título.
Estructura (fragmento):

```
{
  "Creditos Totales": 450,
  "MATEMATICA": 70,
  "CIENCIAS EXPERIMENTALES": 10,
  "PROGRAMACION": 60,
  "ARQUIT, S.OP. Y REDES DE COMP.": 30,
  "INT.ARTIFICIAL Y ROBOTICA": 0,
  "B.DATOS Y SIST. DE INFORMACION": 10,
  "CALCULO NUMERICO Y SIMBOLICO": 8,
  "INVESTIGACION OPERATIVA": 10,
  "INGENIERIA DE SOFTWARE": 10,
  "A.INTEG,TALLERES,PASANT.Y PROY": 45,
  "GESTION EN ORGANIZACIONES": 10,
  "CIENCIAS HUMANAS Y SOCIALES": 10,
  "MATERIAS OPCIONALES": 0
}
```

Nota: Todos los archivos mencionados se encuentran en la carpeta `data/` del backend. Actualmente, estos datos son estáticos y se cargan al iniciar el servidor o se pueden volver a generar con comandos especificos. Se contempla como mejora futura migración a una base de datos relacional o documental.

Anexo D

Funcionamiento de microservicio

Este microservicio implementa dos flujos principales de procesamiento: la **extracción de información académica desde el portal de Bedelías** mediante web scraping y el **procesamiento de escolaridades** desde archivos PDF. A continuación se describe el primero de ellos, que implica navegar dinámicamente el sitio web institucional y construir una representación estructurada de las unidades curriculares y sus previaturas.

D.1. Extracción de información académica

El proceso de extracción de unidades curriculares y previaturas se inicia accediendo al portal de Bedelías, navegando hasta el plan de estudios más reciente de Ingeniería en Computación. Desde esta vista se extrae información sobre la composición de los grupos académicos y las unidades curriculares asociadas, utilizando Selenium para automatizar la navegación y BeautifulSoup para analizar el contenido HTML. Una vez obtenidas las unidades curriculares, el flujo continúa accediendo al sistema de previaturas del portal, donde es necesario recorrer todas sus páginas para extraer la información correspondiente a cada unidad curricular. Para ello:

1. Se identifica cada unidad curricular mediante un atributo data-ri y se accede a su sección de previaturas.

2. Como el árbol de previaturas es dinámico y su contenido solo se carga al ser expandido, se automatiza la expansión de todos los nodos mediante iteración con Selenium.

3. Una vez expandido el árbol, se realiza un scraping recursivo del HTML, reconstruyendo la estructura completa de previaturas mediante objetos anidados que representan reglas como AND, OR, NOT, SOME, entre otras.

4. Finalmente, la información obtenida se transforma en objetos estructurados y se almacena en archivos JSON que pueden ser consumidos por el backend.

Este flujo permite construir una representación precisa y programáticamente utilizable del modelo de dependencias académicas de la carrera, que de otro modo solo está disponible en forma visual y no estructurada en el portal institucional.

D.2. Procesamiento de escolaridades

El procesamiento de escolaridades se realiza a partir de archivos PDF cargados por los usuarios, que contienen el historial académico de un estudiante. Estos archivos son analizados mediante la librería PyPDF para extraer el texto, y luego se interpreta su contenido a través de una lógica que reconstruye el progreso académico en formato estructurado. El microservicio contempla dos variantes de escolaridades: con resultados intermedios, donde se presenta tanto la nota del curso como el resultado final del examen; y con resultados finales únicamente, donde solo se registra el resultado definitivo.

Para cada caso existe una función específica, ya que la estructura del contenido difiere levemente. En ambos flujos, el procesamiento sigue el mismo enfoque general:

1. **Inicialización del objeto de progreso académico**, representado por una estructura que agrupa la información por área de formación y grupos.
2. **Recorrido del texto del PDF**, identificando las líneas que contienen nombres de unidades curriculares, calificaciones y créditos, a través de patrones definidos y funciones auxiliares.
3. **Filtrado de líneas irrelevantes**, como encabezados, separadores u otras entradas que no aportan datos útiles.
4. **Extracción de atributos clave:**
   - Nombre de la unidad curricular
   - Fecha de aprobación.
   - Calificación obtenida (concepto)
   - Créditos correspondientes
   - Tipo de aprobación (curso o examen)
   - Área y grupo académico al que pertenece
5. **Construcción del progreso académico**, añadiendo al objeto final cada unidad curricular aprobada y acumulando los créditos totales obtenidos.

Debido a que los PDFs no incluyen los códigos de las unidades curriculares, este campo es inicializado en blanco (`codigo = `) y posteriormente completado por el backend, que se encarga de cruzar esta información con los datos a los que tiene acceso. Este procesamiento permite transformar documentos no estructurados en un formato legible por el sistema, lo que resulta fundamental para personalizar trayectorias de cursado y validar requisitos académicos de forma automática.