

INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA  
ESCUELA DE INGENIERÍA Y GESTIÓN

## AUTO SCHEDULER

### Planificador Automático de Cuatrimestres

Autores: Terenziani, Santiago (Leg. N° 57240)  
Rojas, Cristobal (Leg. N° 58564)  
Tawara, Enrique (Leg. N° 58717)

Tutor: Huerta, Jorge Ernesto

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE  
INGENIERO EN INFORMÁTICA

Lugar: Buenos Aires, Argentina  
Fecha: 25 de Marzo de 2024

# Índice

<b>Índice</b>	<b>1</b>
<b>Introducción</b>	<b>3</b>
<b>Marco Teórico</b>	<b>4</b>
<b>Estado del arte</b>	<b>5</b>
<b>Extensión y Alcance del Proyecto</b>	<b>8</b>
Limitaciones	9
Infraestructura	9
Funcionalidad	9
<b>Producto Desarrollado</b>	<b>11</b>
Especificación de Requerimientos del Proyecto	11
Especificación de Requisitos Funcionales	11
Especificación de Requisitos No-Funcionales	12
Atributos de Calidad	12
Diagrama de Clases	13
Infraestructura	14
Análisis	14
Elección de bases de datos	15
Infraestructura Propuesta para Producción	16
Infraestructura del Prototipo	19
Back-End	21
Aspectos técnicos	21
Estructura del Proyecto	22
Algoritmo	23
Algoritmos desarrollados	24
Algoritmo Base	24
Algoritmo con Poda	28
Algoritmo TimeGreedy	29
Algoritmo Genético	32
Análisis de los Algoritmos	33
Algoritmo Base con poda	34
Algoritmo TimeGreedy	40
Algoritmo Genético	43
Elección de Algoritmo	48
Front-End	52
Aspectos técnicos	52
Estructura del Proyecto	54
Diseño Inicial de Interfaces	55
Elección de Colores	55
Elección de Logotipo	56
Diseño original	56
Landing Page	59

Vistas de Autenticación	59
Página de Inicio para Universidades	61
Páginas de Sedes	63
Páginas de Períodos	65
Páginas de Materias y Comisiones	66
Páginas de Carreras	69
Página de Inicio para Estudiantes	71
Página de Inicio para Administradores	73
Decisiones para Mejora de Experiencia del Usuario	74
Pruebas de Usuario Iniciales	77
<b>Resultados</b>	<b>79</b>
Metodología	79
Paso 1	80
Paso 2	80
Paso 3	81
Paso 4	81
Paso 5	81
Paso 6	82
Paso 7	82
Observaciones	83
Feedback	85
Ajustes	86
Barra de Búsqueda	86
Botón de Creación	87
Página del Curso	87
Títulos de Formularios	88
Créditos	88
Horarios Predeterminados	89
Auto-Selección de Carrera	89
Asociación de Carreras al Crear Curso	90
Mayor Información en Lista de Cursos	92
Cambio de Flujo para Estudiantes	92
<b>Trabajo Futuro</b>	<b>93</b>
Soporte para Carga de Datos Alternativa	93
Filtros Adicionales	93
Calificación de Dificultad	94
Refinamiento de la función de puntaje	94
Portabilidad	94
<b>Conclusiones</b>	<b>95</b>
<b>Manual de Uso</b>	<b>95</b>
<b>Referencias</b>	<b>96</b>
<b>Anexo</b>	<b>97</b>

## Introducción

Una carrera universitaria conlleva, además de un importante esfuerzo intelectual, innumerables sacrificios. Más allá del precio de la matrícula (si la institución fuese privada) y los años dedicados para graduarse, se incluyen gastos en menor escala: Cada viaje a la facultad implica una inversión en tiempo y dinero, generalmente proporcional a la distancia, y cada hora de tiempo muerto es una oportunidad desaprovechada.

En aquellos casos de alumnos afortunadamente capaces de afrontar estos costos, una mala distribución horaria, donde los días de clase son largos, presentan prolongados intervalos entre cursadas, o los viajes ocurren durante horas pico, puede repercutir negativamente en su productividad y dificultar su progreso académico al punto de abandonar la carrera.

Cada semestre, los universitarios se encuentran ante el dilema de decidir en qué materias inscribirse frente al abanico de posibilidades horarias que ofrece su institución. Considerando las problemáticas mencionadas, siempre intentarán buscar la combinación más eficiente. Es decir, la que mejor se ajuste a sus actividades extracurriculares.

Con el fin de hallar este cronograma ideal, el estudiante debe primero llevar a cabo una exigente tarea de análisis que comienza mediante el estudio de qué materias y horarios están disponibles. Luego filtrar aquellas que no le sean convenientes y finalmente evaluar combinaciones entre las cursadas preseleccionadas, hasta hallar una agenda que satisfaga sus necesidades. Por más óptima que ésta pueda resultar, no quita que su búsqueda consume tiempo y durante la misma se pueden pasar por alto organigramas aún más convenientes.

La aplicación web **AutoScheduler** surge como una propuesta para mitigar este problema, automatizando el proceso de selección de materias, la generación de combinaciones válidas, y la evaluación de cada opción resultante. De esta manera, el estudiante ahorra tiempo sobre el proceso de análisis previo a la inscripción y se asegura una eficiente distribución de horarios durante el semestre.

En el presente informe se describe el proceso de desarrollo de esta herramienta de principio a fin, detallando el contexto bajo el cual fue creado, cómo se diseñó el sistema, y qué decisiones fueron tomadas para asegurar que el producto resultante sea no sólo práctico sino fácil de utilizar.

## Marco Teórico

En primer lugar, se deben aclarar las definiciones y esquemas considerados para describir el modelo en el cual se basa *AutoScheduler*, ya que pueden existir diferencias entre distintas organizaciones educativas. Para el presente trabajo se tomó como referencia el modelo utilizado por el Instituto Tecnológico de Buenos Aires, el cual no difiere considerablemente del de la mayoría de las universidades argentinas.

Una universidad, o cualquier instituto educativo, comúnmente cuenta con ubicaciones físicas donde se llevan a cabo sus actividades académicas; sea en un edificio único, en un campus que reúne diferentes edificios, o múltiples edificios distribuidos a lo largo de una ciudad o región. También existen casos donde las actividades se desarrollan exclusivamente en línea, a través de una plataforma de videoconferencia. Sea cual fuere el caso, a cada uno de estos edificios (contando la plataforma virtual como uno de ellos) se lo considera una **sede** perteneciente a la institución.

Dentro de estas sedes, los estudiantes asistirán a clases y eventos con el fin de recibir su título una vez que completen su **carrera**, la cual consiste en una serie de asignaturas, cursos o **materias** (estos términos se utilizarán indistintamente durante el desarrollo de este informe) que deben aprobarse para poder graduarse. Las mismas pueden ser **obligatorias** o **electivas**, donde estas últimas no son necesarias para completar el plan de carrera, pero podría requerirse que el estudiante apruebe una cantidad determinada de ellas para graduarse.

En general, no todas las materias pueden cursarse en paralelo, ya que muchas son **correlativas** entre sí. Un alumno no estaría habilitado para cursar una asignatura sin antes haber aprobado otras que la institución considere un requisito para la comprensión de ésta. A modo de ejemplo, si un estudiante no aprobó Matemática I, no tendría sentido que su universidad le permita inscribirse en Matemática II cuando los contenidos de ésta requieren de los conocimientos aprendidos en su antecesora.

Por esta razón, se deben distribuir las materias cursadas a lo largo de varios  **períodos**, comúnmente de duración semestral, donde se dictará una variedad de cursos. Algunos podrían dictarse únicamente en períodos específicos. Por ejemplo, en casos donde la baja cantidad de alumnos inscriptos no justifique ofrecerlos en todos los ciclos lectivos.

Incluso si esto ocurre, esto no significa que sus horarios sean siempre iguales. Los mismos pueden verse afectados por un cambio de disponibilidad de su cuerpo docente o de la cantidad de inscriptos de un período al otro. A su vez, en un mismo período podría haber suficientes alumnos anotados para justificar la creación de múltiples **comisiones**. Esto es, un grupo de estudiantes y profesores que se reúnen en horarios determinados para que el docente dicte su cátedra. Si una materia fuese demandada por un muy alto número de estudiantes, como

puede suceder en el caso de aquellas dictadas para varias carreras, tiene sentido que existan distintas comisiones. Esto permite distribuir la cantidad de alumnos entre distintos profesores, franjas horarias y aulas, y además proveer una mayor variedad de horarios posibles para los estudiantes y con ello evitar la superposición con otras asignaturas de interés para los mismos.

Por último, el concepto de **créditos** no es aplicable a todas las universidades, pero sí uno importante en las que lo utilizan. Los mismos actúan como un valor asignado a una materia que el alumno recibe una vez aprobada. La acumulación de estos créditos puede utilizarse para demostrar experiencia por parte del alumno sin importar la fuente exacta de dónde provienen. Por ejemplo, pueden utilizarse para restringir un curso hasta que el estudiante esté suficientemente cercano a graduarse y así logre relacionar su contenido con conocimientos generales aprendidos a lo largo de la carrera, sin provenir éstos de una materia específica.

## Estado del arte

Dentro del Instituto Tecnológico de Buenos Aires (ITBA), el proceso para inscribirse en los cursos de un cuatrimestre se realiza a través del Sistema de Gestión Académica (SGA), donde los estudiantes pueden además consultar el contenido de cada plan de estudios y las comisiones existentes para cualquier materia en cada período.

Es de esta forma que los alumnos pueden investigar sus opciones para la próxima inscripción antes de que la misma se habilite; descubriendo qué asignaturas están habilitados a cursar (revisando manualmente la lista de correlativas de cada una) y los horarios de cada comisión. Para la mayoría de los estudiantes, el proceso de selección suele consistir en esta etapa de investigación, seguida de la etapa de planificación, donde se evalúan diferentes combinaciones (comúnmente en una tabla horaria), para finalmente elegir aquella que más conveniente le resulte.

La idea de AutoScheduler nació desconociendo la existencia de herramientas preexistentes para automatizar el proceso mencionado en el párrafo anterior. Sin embargo, la investigación reveló que estudiantes del mismo ITBA ya habían ideado un sistema con el mismo fin.

El **Combinador de Horarios**<sup>[1]</sup> es una aplicación desarrollada por el centro de estudiantes de dicha universidad, con el objetivo de facilitar a los miembros una herramienta que les permita encontrar planes que mejor se ajusten a sus necesidades. El mismo cuenta con información actualizada de los horarios de comisiones disponibles en el ITBA para el cuatrimestre más cercano.

Esta herramienta requiere que el usuario liste las materias en las que se encuentra interesado y las comisiones que considera aceptables antes de ejecutar el algoritmo. Por lo

tanto, debe estar al tanto de sus opciones antes de completar el formulario. Si bien este sistema puede resultar útil para casos donde un usuario tiene decidido en qué inscribirse, pero no en cuáles comisiones, resulta impráctico para casos donde éste cuenta con diversas opciones, y necesita hallar la mejor combinación de horarios que incluya cualquier materia para la cual esté habilitado.

Otra herramienta desarrollada por alumnos del ITBA es **Combinatrix**<sup>[2]</sup>. La misma también permite combinar horarios de distintas comisiones, pero además solicita que el usuario aporte información de las asignaturas sobre las cuales ejecutar el algoritmo de búsqueda. El programa funciona parseando un archivo de texto que el usuario debe crear copiando y pegando los horarios según se muestran en SGA. Al hacerlo, éste obtendrá acceso a una interfaz gráfica poco intuitiva a través de una terminal. Esto implica que el estudiante debería cargar la información de SGA manualmente para cada materia que esté interesado en cursar.

Una oferta similar puede encontrarse en **Coursicle**<sup>[3]</sup> y **Semester.ly**<sup>[4]</sup>, las cuales cuentan con datos de múltiples universidades. Una vez seleccionada una de éstas, el usuario puede buscar los cursos disponibles por nombre, y añadir sus horarios a un cronograma provisorio, permitiendo construir así su agenda a través de una interfaz práctica e intuitiva que visualice sus selecciones hasta el momento. Sin embargo, el propósito de esta aplicación se limita a ofrecer al usuario una interfaz más conveniente que la del SGA, haciendo foco en la automatización del proceso. En el caso de *Coursicle*, también se hace foco en el envío de notificaciones cuando la inscripción a sus cursos se habilite y exista una vacante en los mismos, mientras que *Semester.ly* ofrece ver distintas posibilidades de cronogramas al variar las comisiones de las asignaturas seleccionadas.

Utah State University<sup>[5]</sup> y University of North Carolina<sup>[6]</sup> también cuentan con armadores de cronogramas para sus estudiantes donde estos deben seleccionar sus cursos de interés primero.

Si bien existen otras herramientas como **Bullet Education Scheduling and Timetabling**<sup>[7]</sup> o **Course Scheduler**<sup>[8]</sup>, estas tienen como principal usuario a la universidad, ya que buscan administrar los recursos de la misma de manera eficiente. El primero lo logra distribuyendo horarios de cursada según la disponibilidad docente y capacidad de aulas, mientras que el segundo utiliza los requerimientos de cada materia en un plan de estudios para armar planes de carrera que aseguren la menor duración de la misma para el estudiante. En ambos casos, el foco está en la institución y sus recursos, considerando al alumnado como un conjunto, en lugar de individuos con necesidades propias.

En contraste a los servicios mencionados, el objetivo de *AutoScheduler* es brindar asistencia a un alumno que desconoce sus opciones de materias a cursar en un próximo período y no tiene una fuerte preferencia por una sobre otra. Con solo definir cuáles tiene aprobadas hasta la fecha, puede deducirse cuáles tiene permitido cursar. Y en base a sus necesidades, cuáles comisiones pertenecientes a esas asignaturas ofrecerían la mejor combinación que se ajuste a su propósito.

Si bien herramientas como *Combinador de Horarios* o *Combinatrix* pueden resultar útiles cuando el alumno tiene decididos sus cursos pero no la comisión a elegir, esto tiende a ser menos frecuente en universidades pequeñas, con baja cantidad de inscriptos, o simplemente con materias exclusivas a una carrera.

A modo de ejemplo, tomando las comisiones disponibles para inicios de 2023, durante el segundo semestre de sus respectivos planes de carrera, un ingeniero industrial del ITBA (plan I22) estaría cursando cinco materias, donde cada una de ellas ofrece como mínimo dos horarios diferentes según la comisión elegida, mientras que un ingeniero informático (plan S10) estaría cursando cuatro, de las cuales sólo dos ofrecerían más de una opción en cuanto a horarios.

En este último caso, *Combinador de Horarios* sólo agilizaría el proceso de encontrar las pocas posibles combinaciones horarias de estas asignaturas, mientras que *AutoScheduler* buscaría otras posibilidades donde sea más conveniente para el alumno sacrificar una de ellas a cambio de otra correspondiente a otro momento del plan. Por ejemplo, en lugar de Física I, correspondiente al segundo cuatrimestre del plan, cursar Química o Economía para Ingenieros, sugeridas para cuatrimestres siguientes pero habilitadas para ser cursadas en cualquier instancia de la carrera, dándole una mayor variedad de opciones a considerar.

Es importante mencionar además que a medida que una carrera avanza, generalmente las materias exclusivas a la misma se hacen más frecuentes, y por diversos factores, disminuye la cantidad de alumnos. Por lo tanto, no debería sobrestimarse el beneficio de intercambiar comisiones, pues en muchas ocasiones los cursos tienen una única comisión.

De esta forma, *AutoScheduler* aporta funcionalidad nueva que beneficia a aquellos usuarios que desconocen todas las materias para las cuales están habilitados a cursar. Este caso se vuelve más recurrente a partir de los años intermedios de la carrera, a medida que se aprueban correlativas más comunes.

## Extensión y Alcance del Proyecto

*AutoScheduler* permite a sus usuarios elegir uno de dos roles al registrarse en la aplicación: **Estudiante** o **Universidad**. Cada uno tendrá acceso a una interfaz separada de la otra, donde podrá realizar acciones exclusivas para su rol.

La herramienta permite a un usuario registrado como *universidad* cargar manualmente la información relevante a su institución. Esto es, datos acerca de sus sedes, carreras, materias, comisiones y períodos. En otras palabras, se da la posibilidad de realizar operaciones CRUD sobre una base de datos donde se almacena la información necesaria para proporcionar cronogramas a los estudiantes.

Por fuera de la aplicación, la universidad deberá ponerse en contacto con un administrador de *AutoScheduler* para verificarla como cuenta oficial de la institución que dice representar y así aparecer en una lista pública de universidades. Una vez añadida a esta lista, los *estudiantes* podrán registrarse como pertenecientes a dicha institución al crear su cuenta.

Al ingresar, se le ofrece a este segundo grupo de usuarios un sistema que permite llevar registro de sus materias aprobadas (necesario para el cálculo de materias habilitadas según correlatividad) y un buscador a través del cual se obtendrán los cronogramas según las preferencias que indique en un formulario.

De la misma manera que sitios orientados al turismo permiten buscar transporte entre dos ciudades con posibilidad de indicar requerimientos de equipaje o preferencias de escala, en este formulario el estudiante puede establecer sus preferencias en cuanto a los cronogramas que desea obtener. Las preferencias presentes en el buscador son:

- Cantidad ideal de horas de clase por semana.
- Rangos horarios en los cuales el estudiante elige no cursar.
- Opción de reducir la cantidad de días de cursada.
- Opción de dar prioridad a aquellas materias que directa o indirectamente habiliten la mayor cantidad de correlativas una vez aprobadas.

Existe además un tercer rol de usuario **Administrador**, el cual por motivos de seguridad sólo puede ser creado directamente en la base de datos. Este tipo de usuario se encargará de marcar como “verificadas” a universidades que hayan demostrado su identidad a un representante de *AutoScheduler*. De esta manera se evita el abuso por parte de agentes fraudulentos que dicen ser representantes de una institución a la cual no están afiliados.

## Limitaciones

Debido a la índole exclusivamente académica del proyecto, se realizaron sacrificios en cuanto a las funcionalidades que presenta la aplicación y los costos que incurriría mantener el sistema en funcionamiento.

### Infraestructura

Considerando que el producto desarrollado no es más que un prototipo, cuyo tráfico representaría tan sólo una fracción del volumen que recibiría un servicio real, la infraestructura ocupada por el mismo se intentó simplificar lo más posible. De otra manera, mantener una red de alta complejidad para un proyecto meramente demostrativo hubiese sido prohibitivamente costoso e innecesario.

Más adelante en este informe se comparan las diferencias entre una arquitectura propuesta para un producto final y la utilizada para el prototipo.

### Funcionalidad

AutoScheduler no ofrece a sus usuarios de tipo *universidad* una opción de carga de datos automática, por lo que éstas deben cargarlos manualmente para poder ser utilizados en la plataforma. Con el fin de garantizar un sistema de carga de datos genérico, que pueda ser utilizado por cualquier universidad, se optó por priorizar una interfaz simple de entender, que permita cargar información sin dependencias externas como podría ser el sitio web de la institución o una base de datos perteneciente a la misma. De esta forma, el usuario podrá cargar data sin tener que programar un adaptador o seguir un formato específico para un archivo de texto. Se admite, sin embargo, que esto llevará a un proceso de carga de datos más lento de lo deseado.

Esto también influye en la actualización de datos disponibles para los estudiantes, ya que en caso de que la universidad no actualice su información, la disponible quedará obsoleta y no será de utilidad para estudiantes que busquen horarios para un período próximo.

Adicionalmente, los planes de carrera no consideran detalles como el orden sugerido de los cursos (el cual daría prioridad a unas materias sobre otras en caso de estar habilitado a cursarlas simultáneamente) o especializaciones dentro del mismo plan de estudios. En este último caso, las asignaturas electivas serían agrupadas dentro de una misma categoría, y queda a discreción del alumno elegir cuáles de ellas cursar para obtener esa especialización, o alternativamente, en manos de la universidad crear diferentes planes de estudio que solo ofrezcan como electivas aquellas materias que aseguren la obtención de esa especialidad.

En cuanto a los usuarios de tipo *estudiante*, los parámetros de búsqueda de combinaciones se limitaron a un número suficiente para demostrar el funcionamiento del algoritmo a grandes rasgos sin complejizar demasiado el desarrollo. Estos usuarios podrían tener preferencias que no son tomadas en cuenta por *AutoScheduler*. Por ejemplo, garantizar la inclusión de determinados cursos en los cronogramas retornados. Funcionalidades de esa índole podrían añadirse al producto en el futuro.

Otra limitación es la falta de soporte para cursos que abarcan más de un período, ya que los mismos se definen únicamente por su fecha de inicio. En caso de que una universidad cuyos ciclos lectivos se rigen por semestres quisiera definir una materia anual, debería crear dos de ellas que representen una primera y segunda parte de la misma, y los estudiantes deberían asegurarse de que la segunda mitad forma parte de su cronograma definitivo si se encuentran a mitad de cursada de dicha asignatura. Es por eso que la funcionalidad de garantizar la presencia de cursos específicos en los organigramas obtenidos es una adición a futuro que el equipo considera fundamental.

En cuanto a la implementación del back-end, la principal limitación se dio en la complejidad temporal del algoritmo. En casos donde el estudiante tuviese a su disposición demasiadas materias y comisiones para combinar, un abordaje de fuerza bruta podría llegar a demorar varios minutos en procesar las millones de combinaciones posibles. Debido a que semejante demora resulta inaceptable en una aplicación web, se negociaron compromisos para asegurar el funcionamiento fluido de la misma. Utilizando algoritmos de características greedy, se sacrificaron algunos resultados con puntaje alto a cambio de una velocidad de procesamiento mucho mayor.

# Producto Desarrollado

## Especificación de Requerimientos del Proyecto

Para orientar el desarrollo y fijar objetivos para considerar el proyecto completado, se definieron al inicio los siguientes requisitos que se esperaba que el producto final cumpliese:

### Especificación de Requisitos Funcionales

Habiendo identificado dos tipos potenciales de usuarios, siendo éstos denominados *Universidad* y *Estudiante*, se definió que la herramienta desarrollada debía cumplir los siguientes requisitos funcionales:

- Para ambos tipos de usuarios la aplicación debe:
  - Soportar autenticación mediante usuario y contraseña.
- Para una Universidad, la aplicación debe:
  - Cumplir con un proceso de verificación para garantizar la identidad de ella al crear la cuenta.
  - Permitir que las universidades definan sedes y el tiempo requerido para trasladarse entre cada una.
  - Permitir la configuración de diferentes planes de carrera para una misma cuenta.
    - Para crear una carrera se debe proveer código, nombre, materias obligatorias y electivas.
  - Permitir que las universidades definan materias dictadas.
    - Para crear una materia se debe proveer código, nombre y materias correlativas bajo un plan de carrera.
  - Permitir la configuración de períodos de cursada y la fecha de inicio de los mismos.
  - Permitir la creación de comisiones para una materia. Estas almacenan información de sus horarios y sedes para cada clase.
- Para un Estudiante, la aplicación debe:
  - Permitir que seleccione la universidad y el plan de carrera al que pertenece.
  - Permitir que seleccione las materias que ya tiene aprobadas.
  - Proveer al estudiante uno o varios posibles cronogramas para un período, en base a las materias aprobadas hasta la fecha y descarte de materias sin habilitación para cursar.
    - Se permite también la configuración de condiciones para filtrar durante la búsqueda los posibles cronogramas indeseables para el estudiante.

## Especificación de Requisitos No-Funcionales

Más allá de las funcionalidades ofrecidas, se consideró que la aplicación debía además cumplir con lo siguiente:

- Ofrecer una interfaz atractiva, sencilla e intuitiva para facilitar su uso por usuarios de distintos grados de experiencia tecnológica.
- Ser utilizable dentro de navegadores web en dispositivos con dimensiones y resolución diferentes.
- Proporcionar mensajes claros e intuitivos al usuario en caso de que ocurra un error.

## Atributos de Calidad

Al diseñar la solución al problema planteado, se tuvieron en cuenta cuatro atributos clave que el sistema debe cumplir:

- Al ser una aplicación apuntada a un público general, la interfaz debería ser sencilla y fácil de comprender para que un usuario con poca pericia utilizando páginas web o con desconocimiento de conceptos manejados dentro del sistema pueda rápidamente adaptarse y utilizar la aplicación sin mayores inconvenientes. Por lo tanto, se debe poner énfasis en la **usabilidad**.
- Para ajustarse a futuras necesidades del público, la funcionalidad de la aplicación debería poder ser expandida con facilidad en el futuro. Por ejemplo, para añadir nuevas opciones de filtro en la búsqueda de cronogramas. En otras palabras, el código debe presentar **mantenibilidad** para ser modificado con el menor esfuerzo posible.
- Considerando que el año escolar tiende a comenzar en una época puntual del año, y que ese es el momento donde los usuarios estarían más interesados en utilizar AutoScheduler, es importante contar con **escalabilidad** para poder atender los pedidos al sistema durante estos picos. Se puede estimar que los períodos clave serían los periodos de Enero-Abril y Julio-Septiembre, ya que estos suelen ser los meses donde inician las actividades escolares en ambos hemisferios del planeta.
- Por último, la **disponibilidad** es otro atributo decisivo, ya que la aplicación debería estar en funcionamiento y disponible para los usuarios sea cual sea el momento en el que la quieran usar.

## Diagrama de Clases

Para el diseño de la estructura de las clases se respetaron los principios de buen diseño de objetos. Siguiendo el principio de responsabilidad única, se logró que cada clase represente una única entidad que permita el cambio interno de su estructura sin afectar sus relaciones o interacciones con las demás clases.

Se buscó adicionalmente seguir el principio DRY, del inglés “Don't Repeat Yourself” a través de segmentación de información en su clase o relación correspondiente.

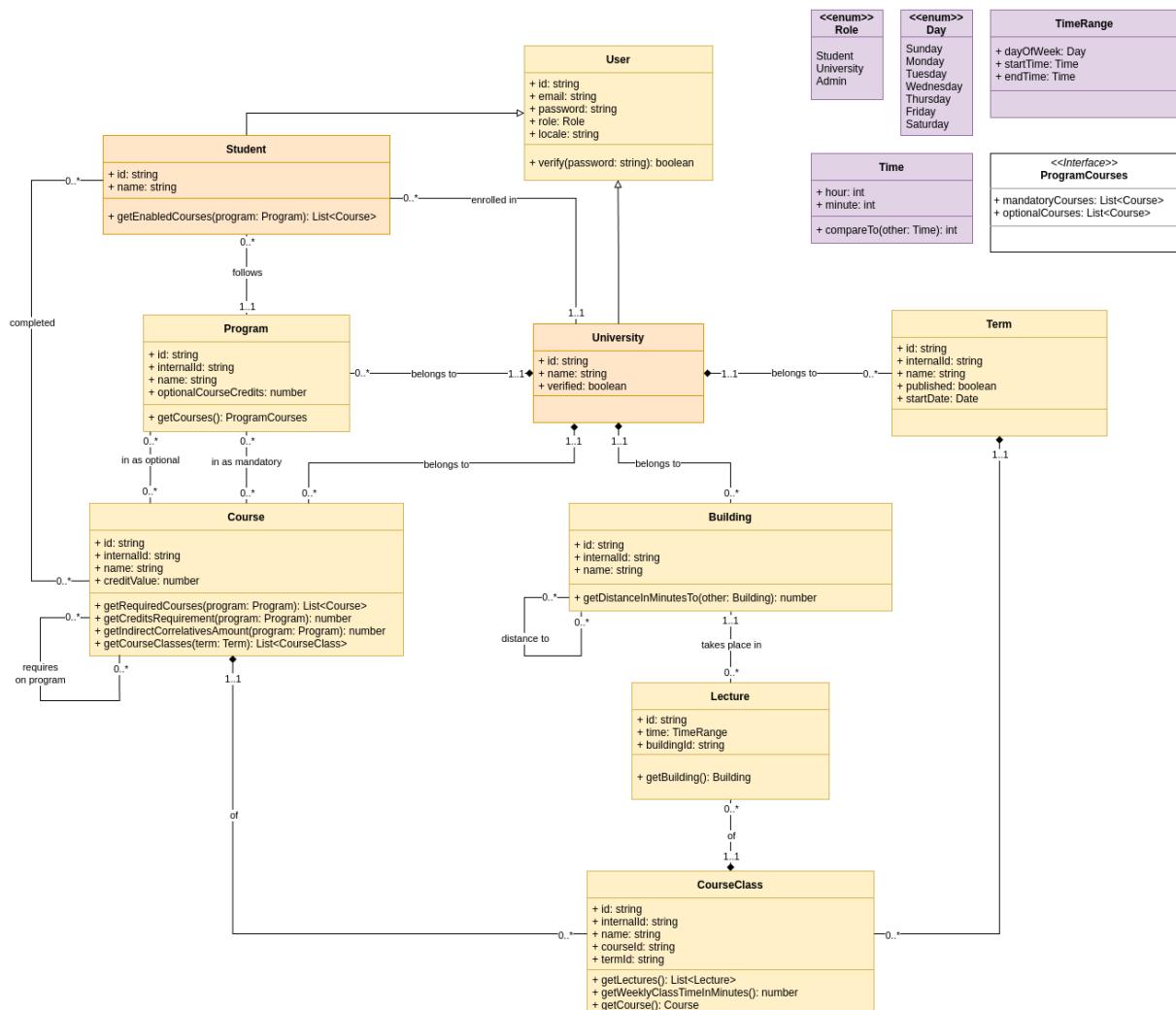


Figura 1: Diagrama de clases. Imagen completa: <https://ibb.co/3zQj9Bx>

El diagrama de clases en la figura 1 representa las relaciones entre las distintas entidades involucradas en el funcionamiento del programa y su cardinalidad, al igual que enums y tipos de datos auxiliares utilizados para representar intervalos de tiempo.

## Infraestructura

### Análisis

Para determinar qué tipo de infraestructura es adecuada para este producto se tuvieron en consideración los siguientes factores:

1. **Cantidad de usuarios iniciales esperados:** La cantidad de usuarios iniciales va a determinar la dimensión de la infraestructura para soportar esa cantidad de conexiones.
2. **Frecuencia y anticipación de aumento de nuevos usuarios:** Qué tan seguido aparecen nuevos usuarios, en qué cantidades y qué tan fácil es de anticipar su aparición. Esto determinará cómo deben crecer los recursos de la infraestructura, siendo estos el tamaño de la base de datos o la cantidad de servidores.
3. **Frecuencia y anticipación de picos de tráfico:** Conocer en qué momentos aparecen los picos de tráfico, qué tan fácil es predecirlos y cuál sería el máximo pico posible ayudará a determinar qué tan reactivo a picos inesperados debe ser la infraestructura.
4. **Frecuencia de desarrollo:** Dependiendo de qué tan seguido se introduzcan nuevas funcionalidades al sitio, puede llegar a ser necesario invertir en un pipeline de CI/CD.
5. **Tipos de datos:** Considerando las entidades que conforman la lógica del sistema y sus relaciones, un tipo de base de datos puede ofrecer beneficios sobre las demás.
6. **Presupuesto:** El presupuesto es un factor limitante. A menor presupuesto, mayor cantidad de funcionalidades de infraestructura deberán sacrificarse.

Dado que se trata de un producto presentado al Instituto Tecnológico de Buenos Aires como mínimo producto viable para demostrar su funcionamiento, el número de usuarios iniciales esperados corresponde a una universidad y un porcentaje de su respectiva cantidad de estudiantes (la cual aproximamos ronda los 3000 activos<sup>[9]</sup>). Luego, la frecuencia de aumento de nuevos usuarios estará ligada a la aceptación del producto en otras instituciones, lo cual se estima será un proceso relativamente lento y fácil de anticipar, dado que los dueños del producto deben conversar con estas universidades antes de verificarlas en su plataforma. Cada una trae consigo un propio cuerpo de estudiantes que asumimos conocido al momento de introducirla como cliente al sistema y por lo tanto será fácil estimar la necesidad de cómputo.

En cuanto al tercer factor, dado que en Argentina las universidades suelen operar bajo un esquema de dos cuatrimestres anuales, con matriculación a principio y mitad de año, se espera que los picos de tráfico coincidan con estas fechas y por ende es muy fácil anticipar el tráfico. Se supone además que entre periodos de matriculación el tráfico va a ser mínimo y por ende se pueden ahorrar recursos de infraestructura al observar menor demanda.

Dada la naturaleza de la utilización del producto, donde existen meses de muy baja actividad, se puede también determinar que en estos periodos se facilita desarrollar nuevas funcionalidades, contando con un amplio margen de tiempo para realizar una actualización manual de la infraestructura, por lo cual no sería necesario introducir un pipeline de CI/CD a menos que el equipo de desarrollo deseé lo contrario y sus fondos lo permitan.

Finalmente, el presupuesto de este proyecto es relativamente bajo, dado que el producto no genera ningún tipo de ganancia económica directa, sino que el producto aporta valor como un beneficio para sus estudiantes, por lo que los recursos económicos de este producto van a estar ligados a las contribuciones que puedan aportar dichas universidades que formen parte del proyecto.

### Elección de bases de datos

En cuanto a los tipos de datos, puede apreciarse en el diagrama de clases de la figura 1 que las entidades presentan fuertes relaciones entre sí. Principalmente, los vínculos entre materias correlativas, y la pertenencia de comisiones a estas, de la misma forma que dichas asignaturas pueden pertenecer a varias carreras a la vez.

Tratándose de un esquema donde las relaciones juegan un rol clave, una base de datos de grafos presenta una ventaja para analizar y obtener estas relaciones sin necesidad de combinar tablas como ocurre en una base de datos relacional. Esto será particularmente importante a la hora de analizar la importancia de una materia basándose en la cantidad que dependen de su aprobación para poder ser cursadas.

Sin embargo, los datos propios de la cuenta de un usuario son independientes del resto de las entidades. Considerando esto, si bien la base de grafos contendrá nodos para relacionar estudiantes y universidades con otras entidades, una base de documentos permite almacenar credenciales, preferencias y metadata de los usuarios con el beneficio de no poseer un esquema fijo. Esta flexibilidad permitirá, por ejemplo, añadir la funcionalidad de personalizar la aplicación con un modo oscuro o modo diurno, guardando dicha propiedad en el documento de autenticación una vez seleccionada sin necesidad de reconfigurar el esquema de la base de datos.

## Infraestructura Propuesta para Producción

Se propone a continuación una posible arquitectura de infraestructura basada en los servicios de *AWS*<sup>[10]</sup>. La utilización de estos servicios de Cloud Computing destaca por una variedad de razones, entre las cuales se destacan la simplicidad de centralizar la totalidad del producto en un solo sistema, la cercanía del servidor en São Paulo a la ubicación geográfica de los principales clientes de la aplicación, el bajo costo de sus servicios y la familiaridad del equipo de desarrollo con la tecnología de AWS.

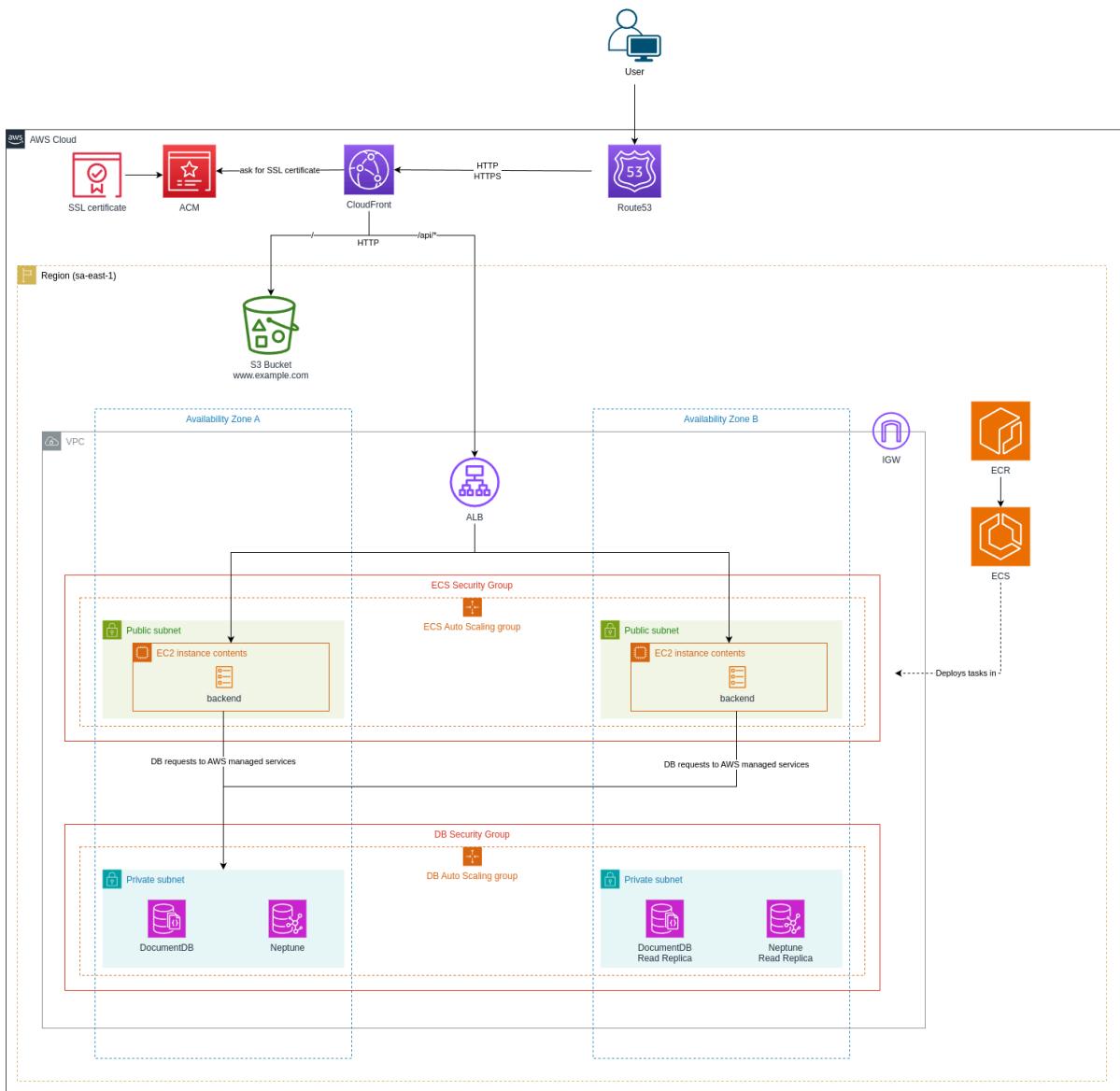
La propuesta para el servicio de back-end se basa en el uso de contenedores de Docker, manejados por el servicio *Elastic Container Service*<sup>[11]</sup> debido a la facilidad para levantar nuevos contenedores en los meses de elevado tráfico y luego eliminarlos en los períodos de inactividad.

Para almacenar el front-end, el cual es un sitio web estático, se utiliza *S3*<sup>[12]</sup> (Simple Storage Service) dado que es un servicio de bajo costo que permite el desentendimiento de cuestiones de escala o costos, ya que éste escala libremente y se paga únicamente por el tráfico recibido.

Para el almacenamiento, se optó por servicios manejados de AWS, en particular *DocumentDB*<sup>[13]</sup> para el guardado de credenciales de usuarios y *Neptune*<sup>[14]</sup> como servicio de base de datos de grafos. La ventaja de estos servicios, manejados por AWS, es que pueden configurarse para levantar réplicas de lectura en momentos de mucho tráfico, y dichas réplicas se pueden eliminar seguramente durante los períodos de inactividad, de forma idéntica a los contenedores.

De esta manera, se logra reducir el costo incurrido en los períodos de inactividad de forma considerable, responder a los aumentos de tráfico y no requiere mucho conocimiento técnico por lo cual, con un poco de capacitación, un administrador no técnico puede encargarse de realizar estos cambios cuando corresponda.

En la figura 2 se muestra el diagrama de la infraestructura y una explicación más detallada sobre los componentes.



**Figura 2:** Arquitectura para producción. Imagen completa: <https://ibb.co/HgKPPdn>

El servicio Route 53<sup>[15]</sup> se utiliza para manejar los registros DNS realizando el ruteo del nombre de dominio al servicio de CloudFront. CloudFront actúa como red de distribución de contenidos para reducir no solo la latencia, usando el punto de presencia más cercano al usuario para responder al pedido, sino también los costos, al disminuir la cantidad de llamadas a los servicios internos mediante el uso de memoria caché.

A su lado se aprecia el servicio de manejo de certificados de AWS, el cual se encarga de proveer un certificado SSL para resolver pedidos seguros HTTPS. Como se puede observar, la comunicación vía HTTPS es únicamente hasta CloudFront. De ahí en adelante se utilizará HTTP, dado que se trata de servicios internos donde el equipo tiene control sobre el origen de los pedidos.

Cuando CloudFront no tenga la respuesta a un pedido en su memoria, diferenciará según

el camino del pedido si debe redirigir ese pedido al balanceador de carga de aplicación, el cual conecta con el servicio de back-end, o al servicio S3 donde está guardado el front-end. La regla es: Si la ruta incluye el prefijo *api/*, entonces redirige el pedido al servicio de back-end. En caso contrario, lo redirige al servicio de front-end.

El servicio S3 provee buckets donde se almacenarán los archivos HTML, CSS, JavaScript y auxiliares necesarios para el funcionamiento del front-end. En este caso, se cuenta con un único bucket con el nombre del dominio del sitio web y está configurado para actuar como host del sitio web estático.

El balanceador de carga de aplicación se encarga de mantener un estado de las réplicas de los servicios del back-end en las distintas zonas de disponibilidad, y determina a cuál de esas copias debe redirigir el pedido según dicho estado. Por ejemplo, si la copia del servicio back-end en la zona A está ocupada, entonces redirige el pedido a la copia del servicio en la zona B.

El servicio de back-end está compuesto por un conjunto de contenedores de Docker manejado por el ECS (Elastic Container Service), el cual se encarga de levantar dichos contenedores según la configuración del mismo. Estos contenedores están dentro de una instancia EC2<sup>[16]</sup>, computadora que provee las capacidades de cómputo a los servicios. El ECS obtiene la imagen necesaria para levantar el contenedor del ECR (Elastic Container Registry), un repositorio donde el equipo de desarrollo del producto podrá publicar las imágenes cuando realice cambios a la plataforma. De ser necesario dividir la lógica del back-end en microservicios, para por ejemplo tener un microservicio dedicado al algoritmo de búsqueda, eso se logra editando la configuración del ECS.

Las instancias EC2 están configuradas bajo el mismo grupo de seguridad, el cual determina que sólo pueden recibir pedidos a ciertos puertos de red (en particular el 3000 para el servicio de back-end) y solamente pueden recibir esos pedidos si provienen del balanceador de carga de aplicación. El grupo de seguridad determina además que tienen permitido comunicarse con la red privada donde están las bases de datos y que pueden salir a internet vía el *IGW*<sup>[17]</sup> (Internet Gateway), un servicio que le da salida a internet a los servicios de AWS dentro del mismo *VPC* (Virtual Private Cloud).

Como se mencionó anteriormente, los servicios de base de datos recomendados para el proyecto son *DocumentDB*, una base de documentos manejada por AWS la cual se utiliza para guardar las credenciales y metadata de los usuarios, y *Neptune*, una base de datos de grafos, también manejada por AWS, para guardar el resto de la información. Estas bases son fácilmente configurables para agregar nuevas réplicas de lectura, y proveen endpoints únicos para que los servicios de back-end no tengan que preocuparse por cuál instancia está del otro lado de la comunicación, lo que facilita el escalado del sitio. Estos servicios están bajo un mismo grupo de seguridad, el cual prohíbe cualquier tipo de salida a internet y sólo permite escuchar y

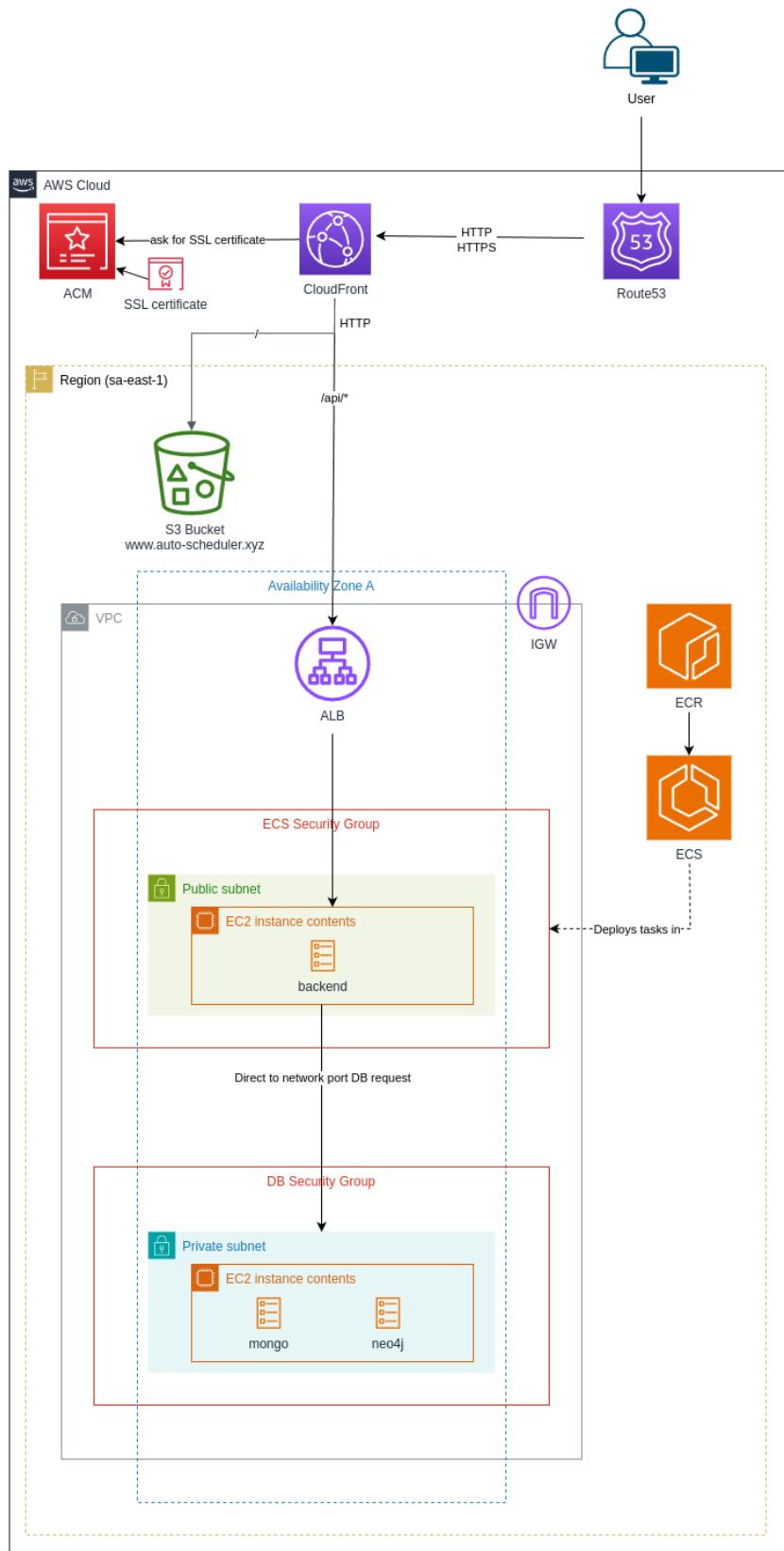
responder a pedidos provenientes de la subred pública. En particular, están permitidos únicamente los puertos relacionados con estas bases de datos, con el fin de mitigar la posibilidad de robo de información.

Opcionalmente se pueden configurar *Auto-Scaling Groups* (grupos de escalado automático) para automatizar el proceso de creación de nuevas instancias EC2 o réplicas de lectura en las bases de datos para períodos de tráfico alto y eliminar dichas instancias EC2 y réplicas cuando el tráfico decrece. Nuevamente, esto es opcional dado que las fluctuaciones de tráfico son fácilmente predecibles y poco frecuentes, pudiendo resolverse de forma manual.

Si se decidiera implementar un pipeline de CI/CD (continuous integration, continuous delivery) se debería automatizar el proceso de buildeo de la imagen Docker, posiblemente en un pipeline de BitBucket, donde reside actualmente el código del proyecto. Luego esa imagen de Docker tiene que ser automáticamente subida al ECR. Finalmente, se debe crear un *cron job*, el cual es un servicio que periódicamente ejecuta lógica configurable para revisar si existe una nueva versión de la imagen en ECR y, en tal caso, forzar un nuevo deployment en ECS. Luego, para el front-end simplemente alcanzaría con tener un pipeline que ejecute el proceso de build del proyecto de React y lo suba al servicio S3.

## Infraestructura del Prototipo

Tratándose de un prototipo para demostrar el concepto de este proyecto, se espera tener una baja cantidad de accesos al sitio y un bajo volumen de datos a ser almacenado en las bases. Por este motivo, se optó por configurar una arquitectura simplificada, con el objetivo de no incurrir en gastos innecesarios cuando no se llega a observar un tráfico que justifique dicha inversión.



**Figura 3:** Arquitectura del prototipo. Imagen completa: <https://ibb.co/kQp8Mpf>

La arquitectura simplificada consta de una única zona de disponibilidad, con una única instancia del servicio de back-end. Route 53 se encarga del manejo de los registros DNS para el dominio de prueba [auto-scheduler.xyz](#) y redirige los pedidos al CloudFront de manera idéntica a la versión propuesta para producción. El servicio S3 está configurado de la misma manera que se planteó en la sección anterior, dado que uno de los beneficios de dicho servicio es que su costo es proporcional a su uso.

Se utilizan dos instancias EC2, una actuando como host para el servicio de ECS y otra con dos containers de docker manualmente configurados; uno sirviendo una base de datos Mongo<sup>[18]</sup> y otro una base Neo4J<sup>[19]</sup>.

Esta versión simplificada de la arquitectura permite ser transformada en la versión propuesta para producción con pocos cambios, siendo estos la expansión de los servicios de ECS en otro availability zone y el reemplazo de la instancia EC2 que contiene las bases de datos por los servicios manejados de AWS, DocumentDB y Neptune, agregando opcionalmente los Auto-Scaling Groups.

## Back-End

Para garantizar la *mantenibilidad*, se buscó utilizar tecnologías bien documentadas y patrones de diseño que faciliten el mantenimiento y la extensión del código escrito.

### Aspectos técnicos

El back-end se construyó utilizando Node<sup>[20]</sup>, el entorno de ejecución de JavaScript<sup>[21]</sup>, dado que éste permite la creación de aplicaciones de red escalables que permiten procesar pedidos de clientes de manera eficiente y no-bloqueante.

Se utilizó además el framework Express<sup>[22]</sup>. El mismo ofrece herramientas que facilitan la configuración de una API como son la definición de rutas y el uso de middlewares.

Para tener un mayor control sobre el código, se optó por utilizar TypeScript<sup>[23]</sup>, un lenguaje basado en JavaScript con el beneficio adicional de tipado estático. Este permite compilar a JavaScript tras un proceso de compilación donde se analiza si el código presenta errores sintácticos. De esta forma, no sólo se hace el código más legible gracias a la especificación de tipos, sino que además garantiza consistencia y evita errores imprevistos donde el programador obvia un cambio de tipos y en consecuencia, corre el riesgo de que sus variables exhiban un comportamiento inesperado durante la ejecución.

## Estructura del Proyecto

El proyecto de back-end está organizado según el siguiente árbol de directorios:

- **resources**
  - **emailTemplates**
- **src**
  - **routes**
  - **controllers**
  - **services**
  - **persistence**
  - **dtos**
  - **models**
  - **factories**
  - **helpers**
  - **middlewares**
  - **constants**
  - **exceptions**
  - **interfaces**

En **resources** se encuentran aquellos recursos que no forman parte del código fuente. En este caso, sólo abarca los templates utilizados para el contenido de emails enviados a los usuarios al registrarse, ser verificados, o solicitar un cambio de contraseña.

En **src** se encuentra todo el código que hace funcionar la aplicación.

**routes** define las rutas que ofrecerá la API REST<sup>[24]</sup> expuesta por el back-end. Es aquí donde se definirá, dado un método HTTP y un path, qué *middlewares* utilizar para verificar si quien hace el pedido tiene permitido llamar a un método definido en *controllers* para atenderlo.

Los **controllers** se encargan de recibir una solicitud y realizar validaciones sobre sus parámetros. Cualquier error de formato será detectado en esta instancia, antes de realizar un llamado a un *service*.

En **service** se encuentran los intermediarios entre los *controllers* y *persistence*. Si bien la mayoría de los casos solo requieren pasar el pedido entre estas dos capas para realizar un llamado a la base de datos, algunos services como *scheduleService* realizan operaciones complejas con la data recibida desde la capa de persistencia para devolver al controller una respuesta procesada en forma de un *model*.

En **persistence** se encuentran las interfaces de las bases de datos y las implementaciones de las mismas actualmente en uso (Mongo y Neo4J). Esta capa se encargará de comunicarse con la base y pasar al service los datos leídos o modificados.

Retornando hacia el *controller*, si se debe retornar contenido, se utilizará uno de los Data Transfer Object definidos en **dtos**. Estos definen el formato que debe respetar cierto objeto a la hora de ser retornado como respuesta al usuario de la API. Cada DTO cuenta con un método que, recibiendo un *model*, retorna los campos relevantes para una respuesta y URLs por los cuales se pueden leer relaciones del objeto leído. Por ejemplo, el DTO de una materia contendría no sólo su nombre, identificador interno y valor en créditos, sino también los endpoints a través de los cuales consultar su lista de comisiones, la lista de correlativas previas, y la cantidad de créditos requeridos para cursar dicha materia en cada programa.

Los **models** mencionados no son más que instancias de las clases definidas en el diagrama de clases.

Los **factories** se encargan de obtener o inicializar las instancias singleton de las implementaciones en *persistence*. En caso de que dicha instancia no estuviese inicializada, se intentaría definir en la base de datos los constraints que involucran al tipo de dato que maneja el factory/persistence en cuestión, en caso de que estos tampoco estuviesen definidos.

Los **helpers**, como su nombre indica, ofrecen métodos que facilitan tareas como, por ejemplo, procesar una respuesta de la base de datos, realizar operaciones con los tipos personalizados Time y TimeRange, validar que un string cumple el formato esperado para un tipo de dato, etc.

Por último, los **middlewares** son esencialmente filtros llamados al recibir un request. Por ejemplo, pueden verificar que el usuario que ejecuta el pedido esté autenticado y tenga un rol determinado, o que los parámetros utilizados en una consulta que utilice paginación posean el formato correcto.

## Algoritmo

Para poder calcular los mejores cronogramas disponibles para el estudiante, debe ejecutarse un algoritmo que analice un rango variado de posibles combinaciones de comisiones y retorne, entre aquellas que el usuario está habilitado para cursar plenamente, las que mejor se ajusten a las preferencias definidas por el mismo.

Debido a la gran cantidad de datos que maneja y la complejidad del cálculo, se decidió que la ejecución del algoritmo ocurriría en el back-end, y no en el front-end. Transferir entre capas toda la información necesaria para ejecutarlo resultaría ineficiente y no todos los dispositivos contarían con el poder de cómputo necesario para ejecutar el algoritmo sin afectar la performance de otros procesos que el usuario esté utilizando en su dispositivo.

Se diseñaron consultas específicas para poder calcular métricas como sería la importancia de una materia mientras se leen los datos de la base. Una vez obtenida esta información, se puede iniciar el algoritmo en el back-end y al finalizar la ejecución, ya sea al terminar de procesar todos los casos o por superar el límite de tiempo asignado, responder al request enviado desde el front-end con los mejores cronogramas ya formados.

Tras realizar las llamadas a la base de datos, el servicio relacionado a los cronogramas delega la responsabilidad de ejecutar alguna de las implementaciones de los posibles algoritmos a un servicio dedicado a la misma, con el fin de facilitar el agregado de nuevas alternativas o la modificación de las existentes sin interferir con los demás. De esta forma se asegura la mantenibilidad del código.

### Algoritmos desarrollados

Con el fin de encontrar una metodología óptima para la obtención de cronogramas, se desarrollaron varios algoritmos con tácticas diferentes para alcanzar una optimización entre calidad de resultados y costo computacional.

Para todos los algoritmos que se detallan a continuación, se recibirán como parámetros un plan de carrera, un período de cursada, un número entero *horasDeseadas*, valores booleanos *priorizarCorrelativas* y *reducirDías* y un listado de franjas horarias prohibidas, además de la lista de posibles asignaturas a cursar. Este último argumento se obtiene consultando la base de datos a partir de las materias ya aprobadas por el usuario estudiante en cuestión.

#### *Algoritmo Base*

En primera instancia, se definió un algoritmo que funciona de la siguiente manera:

1. Obtener listado de materias pertenecientes al plan de carrera recibido como parámetro.
2. Filtrar la lista de materias
  - a. Remover aquellas materias que el alumno aún no está habilitado para cursar.
  - b. Remover aquellas materias que el alumno ya tiene aprobadas.
3. Asignar a cada materia en la lista un valor ***importancia***, correspondiente a la cantidad de materias que directa o indirectamente requieran de la aprobación de ésta para ser cursadas. En otras palabras, para cada materia se calcula su influencia en la habilitación de futuras materias.
4. Para cada materia en la lista, obtener la lista de comisiones existentes para el período recibido como parámetro.
5. Eliminar, de la colección de comisiones, aquellas con horarios que se superpongan con las franjas horarias indeseables, recibidas como parámetro.

6. Calcular todas las posibles combinaciones de comisiones para obtener todos los posibles cronogramas.
7. Descartar aquellos cronogramas inválidos:
  - a. Un cronograma no puede incluir más de una comisión por materia.
  - b. Un cronograma no puede incluir comisiones cuyos horarios se superpongan.
  - c. Un cronograma no puede incluir comisiones tal que dos clases consecutivas tengan menor tiempo entre sí que el requerido para trasladarse entre sus respectivas sedes.
  - d. Un cronograma no puede aportarle al estudiante más créditos provenientes de materias electivas que los necesarios para finalizar su carrera, a menos que todas las materias electivas presentes en el cronograma sean necesarias para alcanzar ese valor. Es decir, no deberían incluirse electivas de más.
8. Para cada cronograma válido, calcular:
  - a. *diasTotales* := Número de días en los cuales hay al menos una clase.
  - b. *horasTotales* := Horas totales de clase por semana.
  - c. *importanciaTotal* := Suma de valores *importancia* de cada materia que conforme el cronograma.
  - d. *tasaElectivas* := Cantidad de horas semanales correspondientes a materias electivas para el plan, dividido por la cantidad de horas totales en el cronograma.
9. Con todos estos datos calculados, calcularemos el puntaje total para cada cronograma:
 

```

p1 := 1 - tasaElectivas
p2 := | horasDeseadas - horasTotales |
p3 := 7 - diasTotales
p4 := importanciaTotal
A := 1 si reducirDías es verdadero, 0 si es falso
B := 1 si priorizarCorrelativas es verdadero, 0 si es falso
      
```

$$puntaje := 10*p1 - 1.25*p2 + A*3.5*p3 + B*p4$$

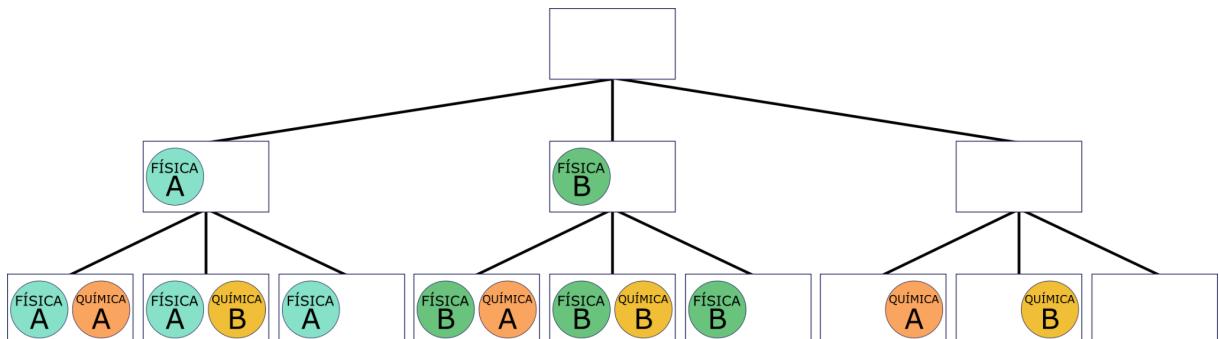
10. Ordenar la lista de cronogramas en orden descendente según su puntaje correspondiente y retornar los primeros 10 elementos. En el caso donde los 10 tengan el mismo puntaje, extender este número hasta incluir todos los resultados que empaten por el primer puesto, con un máximo de 25 elementos.

Con el fin de priorizar la eficiencia, la implementación en código alteró ligeramente el orden de los pasos establecidos en el algoritmo teórico. Asimismo, se tomaron recaudos para no considerar cronogramas con extremadamente baja posibilidad de mejoría en caso de añadirle más materias:

1. Obtener la data necesaria de la base de datos en la menor cantidad de llamados posibles y almacenarla en memoria. Esta información se recibiría filtrada a través de la consulta realizada, englobando los primeros 4 pasos del algoritmo original.
2. Generar un mapa donde a cada identificador de una materia se mapeen los IDs de comisiones que no entren en conflicto con los horarios especificados por el estudiante.
3. A partir de dicho mapa, generar un arreglo de arreglos de comisiones, de tal forma que el arreglo padre quede ordenado según la *importancia* de las materias que cada índice representa, y dentro de un índice se encuentre un arreglo con las comisiones pertenecientes a esa materia. De esta forma, si la ejecución corta antes de encontrar todas las combinaciones, se garantiza que las materias más importantes habrán sido consideradas al encontrarse al inicio del arreglo padre. En caso de empate entre dos materias, el orden entre ellas será aleatorio para asegurar una mayor variedad de resultados en caso de habilitar poda o una cota máxima de tiempo de ejecución.
4. Crear una lista vacía de **cronogramas válidos definitivos**.
5. Generar las combinaciones posibles hasta recorrer todo el arreglo o superar el plazo de tiempo asignado como timeout. Para hacer esto, se recorre el arreglo de comisiones en orden descendente de importancia y se ejecutan los siguientes subpasos:
  - a. Crear una lista vacía de **cronogramas válidos provisorios**.
  - b. Si la materia actual es una electiva de la carrera para la cual se buscan planes, tomar registro de su valor en créditos.
  - c. Considerar el caso donde un cronograma incluye únicamente al curso actual. Para ello, crear arreglos de comisiones que incluyan cada comisión del curso como único elemento. Estos arreglos se sumarán a nuestros **cronogramas válidos provisorios**.
  - d. Para cada uno de los **cronogramas válidos definitivos**:
    - i. Si la materia actual es electiva, y el cronograma definido ya supera la cantidad de créditos provenientes de electivas, se lo ignora, pues expandirlo generaría un cronograma inválido (Paso 7d original)
    - ii. Por cada comisión de la materia actual:
      1. Proponer un arreglo que combine el cronograma válido definitivo actual y esa comisión.

2. Calcular el tiempo de clase que implica ese cronograma. Si supera en un 25% el número de horas que solicitó el estudiante, descartarlo, ya que si supera con creces la expectativa, cualquier futura expansión del cronograma también lo hará, dando un resultado aún más lejano al objetivo.
  3. Si el cronograma es válido (es decir, cumple las demás restricciones del paso 7 original), agregarlo a los ***cronogramas válidos provisорios***.
  - e. Una vez que generamos todos los ***cronogramas válidos provisорios*** posibles que incluyan la materia actual, concatenar estos cronogramas provisорios a los ***cronogramas válidos definitivos*** y continuar el ciclo con el próximo curso.
6. Ejecutar los pasos 8-10 originales con normalidad.

Puede interpretarse esta implementación como la construcción de un árbol donde la raíz es un cronograma vacío y en cada nivel se abren ramas, donde al nodo padre se le añade una de las posibles comisiones del próximo curso o ninguna de ellas.



**Figura 4:** Diagrama para ejemplificar la construcción del árbol con el algoritmo Base.

### *Algoritmo con Poda*

Una vez implementado el algoritmo anterior, se observó que la complejidad del mismo resultaba en un tiempo de respuesta más lento que el deseado. Dado que para el prototipo en desarrollo la ejecución de este algoritmo estaría ocupando los recursos del servidor y por lo tanto imposibilitaría la atención a otros requests entrantes, se tomó la decisión de configurar parámetros de poda, de forma tal que se descarten cronogramas poco prometedores en una etapa temprana y se evite expandirlos al añadir materias adicionales al mismo.

Extendiendo el *Algoritmo Base*, se añadieron controles para determinar si una nueva combinación válida debe ser descartada. También se movió el cálculo del puntaje al momento previo a agregarla al conjunto de cronogramas válidos definitivos. De estos cambios surge un nuevo algoritmo apodado internamente como *CourseGreedy*, o simplemente *algoritmo con poda*.

Una vez que se hayan procesado suficientes cursos y/o combinaciones (cantidades personalizables a través de variables de entorno), si el cronograma siendo analizado supera una carga horaria mínima (el mínimo entre un número configurable por parámetro y la mitad de la cantidad de horas solicitada por el usuario), se comparará su puntaje con el promedio observado hasta el curso actual.

De esta forma, si el cronograma analizado es suficientemente maduro (ofrece suficientes materias y carga horaria para considerarlo una base sólida y desarrollada sobre la cual construir) y es además una mejora en comparación a las bases sobre las cuales se construyó, se lo conservará como un cronograma válido. Si su puntaje es visto como una desmejora, se lo descartará.

Si bien esta poda sacrifica cronogramas que podrían llevar a soluciones óptimas bajando sobre esa rama, agiliza considerablemente la ejecución del algoritmo al obviar ramas que generalmente no llevan a una solución ideal.

### *Algoritmo TimeGreedy*

Este algoritmo parte de la hipótesis de que el mejor cronograma para una determinada cantidad de *horasDeseadas* se puede analizar en subproblemas, calculando primero el mejor cronograma con una carga horaria pequeña, e ir aumentando la carga horaria del mismo hasta llegar a una carga igual al *horasDeseadas* provisto.

La implementación del algoritmo consiste de los siguientes pasos:

1. Armar una tabla donde cada columna representa una comisión disponible, sin agruparlas por cursos. Las filas representan una cierta cantidad de minutos y las celdas representan el mejor cronograma válido con igual o menor carga horaria que el valor de la fila y a su vez contenga la comisión de la columna, En caso de que no exista un cronograma que cumpla ambas condiciones, la celda quedará vacía.
2. Calcular valores auxiliares necesarios para completar la tabla:
  - a.  $minutosDeseados := horasDeseadas * 60$
  - b. *paso*: Es el valor incrementado entre filas, debe ser lo suficientemente grande para que dos filas consecutivas no resulten en resultados iguales, pero no al punto que reduzca la variedad de cronogramas para expandir, ya que podría llevar a ignorar casos donde una expansión hubiera dado mejores resultados. En esta implementación, se calcula tomando el mínimo entre la cantidad de horas semanales de la materia con menor carga horaria y la mínima diferencia no-nula de carga horaria semanal entre dos materias diferentes.
  - c. *minutosMínimos*: Es el valor correspondiente a la primera fila, debe ser menor que la cantidad de minutos de la materia de menor carga horaria y cumplir la condición de que sumado una cantidad determinada de *paso* resulte equivalente a *minutosDeseados*. De esta manera se obtendrán los cronogramas con *minutosDeseados* en la última fila del algoritmo.
  - d. *minutosActuales* := *minutosMínimos*. Representa la cantidad de minutos correspondiente a la fila actual.
3. Completar la primera fila con cronogramas vacíos, dado que no hay cronogramas con carga horaria menor o igual valor que *minutosMínimos*.
4. Incrementar *minutosActuales* por un valor equivalente a *paso*.
5. Completar las celdas de la fila actual con el cronograma de mayor puntaje entre los siguientes candidatos, donde *comisionActual* representa la comisión correspondiente la columna de la celda:
  - a. El cronograma correspondiente a la fila anterior, en la misma columna. Es decir, mantener el resultado actual para esta comisión.

- b. Si ( $minutosActuales$  - carga horaria de  $comisionActual$ ) es mayor o igual a  $minutosMínimos$ , contemplar los cronogramas válidos resultantes de añadir  $comisionActual$  a los distintos cronogramas de la fila correspondiente al tiempo ( $minutosActuales$  - minutos de la materia de  $comisionActual$ ). En otras palabras, añadir la comisión actual a un cronograma donde anteriormente no encajaba debido al límite de tiempo.

La función de puntaje a utilizar es la misma definida en el algoritmo base.

	Comisión A	Comisión B	Comisión C	Comisión D	Comisión E
$minutosMinimos$	-	-	-	-	-
$minutosMinimos + paso$	-	B	C	-	E
$minutosMinimos + 2*paso$	A	BE	CE	D	CE
...	...	...	...	...	...
$minutosMinimos + n*paso$	AE	BD	CD	CD	AE
$minutosDeseados$	ACD	BCD	BCD	CDE	CDE

**Figura 5:** Diagrama para ejemplificar la construcción de la tabla en el algoritmo TimeGreedy.

6. Si  $minutosActuales$  es igual a  $minutosDeseados$  o se cumple alguna condición de corte, retornar los mejores 10 cronogramas únicos de la fila de  $minutosActuales$ . Caso contrario, volver al paso 4.

Esta implementación del algoritmo deja en claro una debilidad y un error que presenta la hipótesis del mismo. Dicho error es asumir que el mejor cronograma se puede obtener de forma greedy, ya que el subconjunto ideal para una cierta materia podría no ser el subconjunto con mejor puntaje en su respectiva fila. Para el ejemplo de la figura 5, podría suceder que el mejor puntaje lo tiene la combinación ABC. Sin embargo, ninguna de las tres posibles columnas donde se podría llegar a generar ABC las puede generar dado que no existen celdas con AB, BC o AC para C, A ó B cuando éstas intentan buscar un subconjunto para la fila de  $minutosDeseados$ . A nivel práctico, la posibilidad de que ocurra un caso como éste depende de la función de puntaje. Para este caso, dicha probabilidad recae fuertemente en el peso que se le asigna a  $diasTotales$ .

Por otro lado, la principal debilidad del algoritmo es la reducida variedad de combinaciones que retorna, siendo consecuencia de que en cada fila de la tabla se queda con uno solo de los varios cronogramas con un puntaje alto similar o idéntico al mejor.

Ambas fallas se pueden mitigar con una ligera modificación al algoritmo, donde no sólo se almacena un cronograma por celda, si no los mejores  $N$  cronogramas. De esta manera, se reducen las probabilidades de que el subconjunto ideal para cierta materia sea descartado y por lo tanto, aumenta la variedad de posibles combinaciones sin perder considerablemente la mejora de rendimiento que provee el algoritmo.

Adicionalmente, este algoritmo corre con la debilidad de que, en caso de alcanzar el límite de tiempo asignado, el mismo sólo retornará cronogramas con *minutosActuales* menores a *minutosDeseados*, siendo posible el caso donde retorne opciones con una carga horaria notoriamente menor que la deseada.

Dado que la generación de la tabla para almacenar los datos puede resultar en un alto uso de memoria, es posible refinar la implementación de este algoritmo con los siguientes cambios:

- Utilizar sets sin repetición para los cronogramas de cada fila para evitar repetidos.
- Borrar las filas que ya no sean necesarias; aquellas con carga horaria inferior a la resta entre *minutosActuales* y la mayor carga horaria entre las materias consideradas.

## *Algoritmo Genético*

Teniendo en cuenta que la naturaleza del problema no corresponde totalmente al de un problema que puede ser resuelto de forma greedy, se decidió implementar un algoritmo genético que inicialice un conjunto de combinaciones al azar, y utilice las mismas como base para realizar modificaciones sobre las mismas.

El primer paso del algoritmo consiste en aleatoriamente armar un conjunto de combinaciones de comisiones de manera aleatoria y calcular el puntaje de cada una. En casos donde el cronograma resultante de esa selección aleatoria no sea válido, se le asignará un puntaje de valor mínimo.

Las mejores combinaciones de este conjunto se guardarán en una colección aparte **bestSchedules**, donde almacenaremos los mejores resultados observados durante la ejecución.

A continuación, se entrará en un ciclo hasta que se itere hasta el número de generación definida como última por parámetro. En cada ciclo, se seleccionarán aleatoriamente dos padres mediante el mecanismo Tournament<sup>[25]</sup> para mezclarse (intercambiando algunas comisiones) y dar lugar a dos combinaciones “hijas”, las cuales podrían sufrir una mutación donde la comisión de un curso (o la falta de la misma) puede reemplazarse por otra (o desaparecer), independiente de si la nueva comisión estaba incluida en alguno de los cronogramas padres o no.

Esta cruce se repetirá hasta obtener suficientes hijos para conformar una generación del mismo tamaño que la anterior. A partir de ésta se tomará registro de los mejores resultados que la conforman, y se repetirá el ciclo cruzando padres provenientes de esta generación.

Debido a que no todas las comisiones son compatibles unas con otras, los resultados preliminares de este algoritmo dejan mucho que desear. Al generar cronogramas aleatoriamente, existe una alta probabilidad de que una combinación no sea válida, haciendo que pocos de los integrantes de una generación tengan un buen puntaje. Es por esto que se dificulta encontrar algún máximo global con este algoritmo, ya que todo depende de la selección inicial aleatoria y de que los cruces y mutaciones lleven a uno de los resultados que se intenta buscar.

## Análisis de los Algoritmos

Debido a la ineficiencia del algoritmo base, resultó necesario escoger para el prototipo una implementación que permita obtener resultados suficientemente buenos en una fracción del tiempo que demanda el base.

Para definir los mejores parámetros para cada algoritmo, se realizaron experimentos con dos conjuntos de datos. El primero fue un estudiante de ingeniería informática nuevo “A” al ITBA, sin ninguna asignatura aprobada. El segundo fue uno avanzado “B” con las siguientes materias aprobadas:

- 93.58 - Algebra
- 93.26 - Análisis Matemático I
- 93.28 - Análisis Matemático II
- 97.35 - Arquitectura de Computadoras
- 72.39 - Autómatas, Teoría de Lenguajes y Compiladores
- 72.37 - Base de Datos I
- 82.31 - Estructura de Datos y Algoritmos
- 72.35 - Ingeniería de Software I
- 72.36 - Interacción Hombre-Computadora (HCI)
- 72.03 - Introducción a la Informática
- 93.35 - Lógica Computacional
- 93.59 - Matemática Discreta
- 93.07 - Métodos Numéricos
- 93.24 - Probabilidad y Estadística
- 72.32 - Procesamiento de Documentos XML
- 72.31 - Programación Imperativa
- 72.33 - Programación Orientada a Objetos
- 72.07 - Protocolos de Comunicación
- 72.38 - Proyecto de Aplicaciones Web
- 72.11 - Sistemas Operativos

Para ambos estudiantes, se calcularon cronogramas con *reducirDías* y *priorizarCorrelativas* en verdadero.

Como criterio para medir la efectividad de un algoritmo, se definieron las siguientes métricas:

- Tiempo de ejecución: Tiempo desde que el servicio recibe toda la información necesaria de la base de datos hasta que el algoritmo responde con una lista de cronogramas.

- Cercanía al puntaje máximo: Cociente entre el puntaje obtenido por el algoritmo testeado y el valor del máximo puntaje posible utilizando la configuración utilizada. Posteriormente denominado simplemente “cercanía”.
- Cobertura del puntaje máximo: Cantidad de resultados obtenidos por el algoritmo cuyo puntaje equivale al máximo posible con la configuración utilizada, dividido por la cantidad de cronogramas con máximo puntaje posible con la configuración utilizada. Posteriormente denominado simplemente “cobertura”. En pocas palabras, el porcentaje de resultados “perfectos” que se encontraron.

Para obtener el puntaje máximo de una configuración dada y la cantidad de cronogramas con dicho valor, se ejecutó el algoritmo base sin restricción de cuántas opciones podía retornar el mismo.

Los cursos y comisiones utilizados para el experimento incluyeron todas las materias obligatorias para el plan de Ingeniería Informática para el primer cuatrimestre de 2024 según el SGA, además de todas sus comisiones. Adicionalmente, se incluyeron algunas electivas seleccionadas y sus respectivas comisiones. Esto dio un techo de 80,639 combinaciones posibles para el primer estudiante (quien tenía 10 materias habilitadas, con 28 comisiones entre ellas) y 1,665,269,759 para el segundo (19 materias, con 55 comisiones).

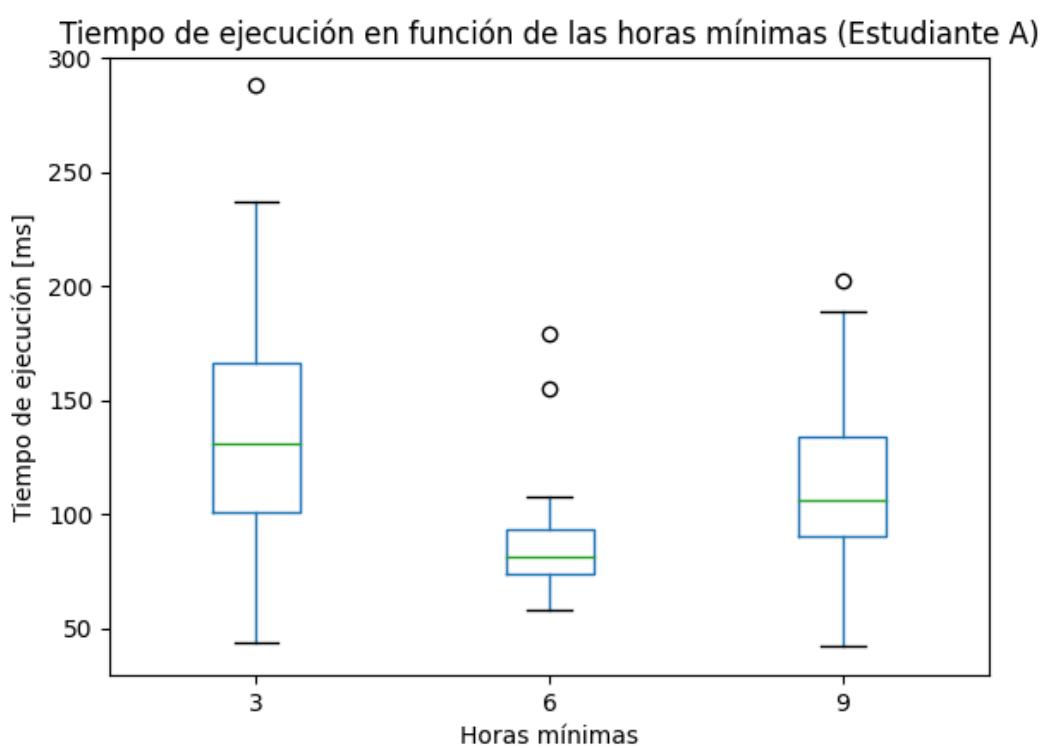
Experimentando con diferentes parámetros, se llegó a las siguientes conclusiones:

#### *Algoritmo Base con poda*

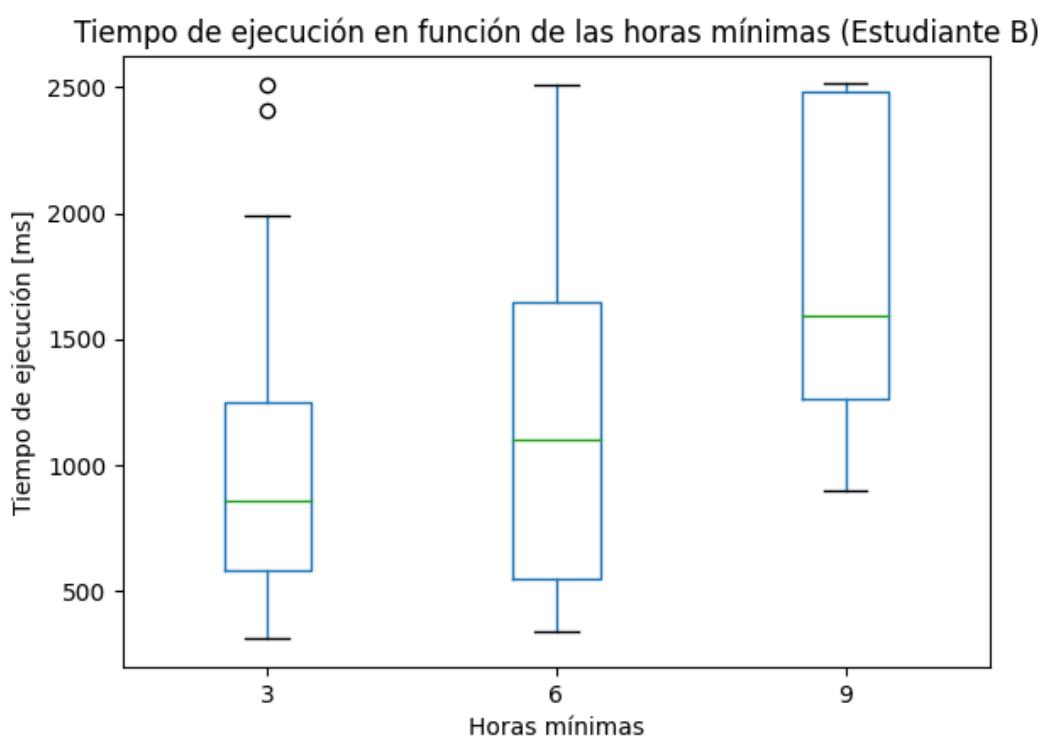
Con el fin de definir la cantidad de horas mínimas y la cantidad de cursos antes de que el algoritmo decida podar se llevaron a cabo pruebas con ambos estudiantes A y B, con *horasDeseadas* en 18, 21, 24 y 27, con cinco repeticiones para cada configuración.

Para el estudiante A, la cobertura fue máxima en todas las configuraciones. Es decir, sin importar los parámetros, se lograban encontrar todos los resultados óptimos, mostrando diferencias únicamente en el tiempo de ejecución.

Para definir las horas mínimas ideales, se varió este parámetro con los valores 3, 6 y 9, con la cantidad de cursos mínimos a procesar en cero para que este parámetro no interfiera con la condición de poda. Es decir, un cronograma será considerado para la poda sólo si su carga horaria supera el parámetro especificado.



*Figura 6: Tiempo de ejecución en función de cantidad de horas mínimas para podar para el estudiante A*



*Figura 7: Tiempo de ejecución en función de cantidad de horas mínimas para podar para el estudiante B*

Cobertura del puntaje máximo en función de las horas mínimas (Estudiante B)

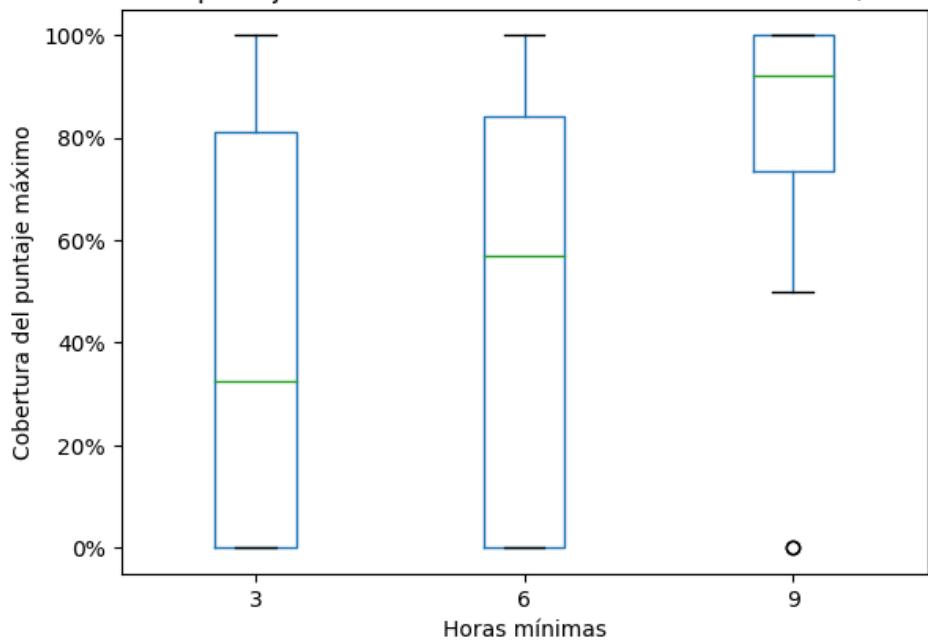


Figura 8: Gráfico de la cobertura en función de cantidad de horas mínimas para podar para el estudiante B

Cercanía al puntaje máximo en función de las horas mínimas (Estudiante B)

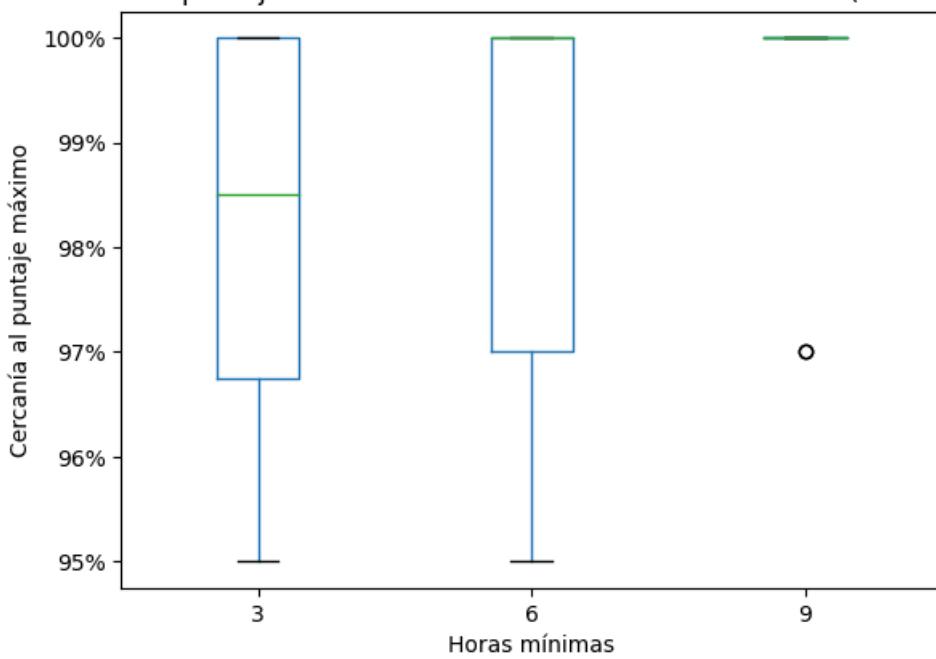


Figura 9: Gráfico de la cercanía en función de cantidad de horas mínimas para podar para el estudiante B

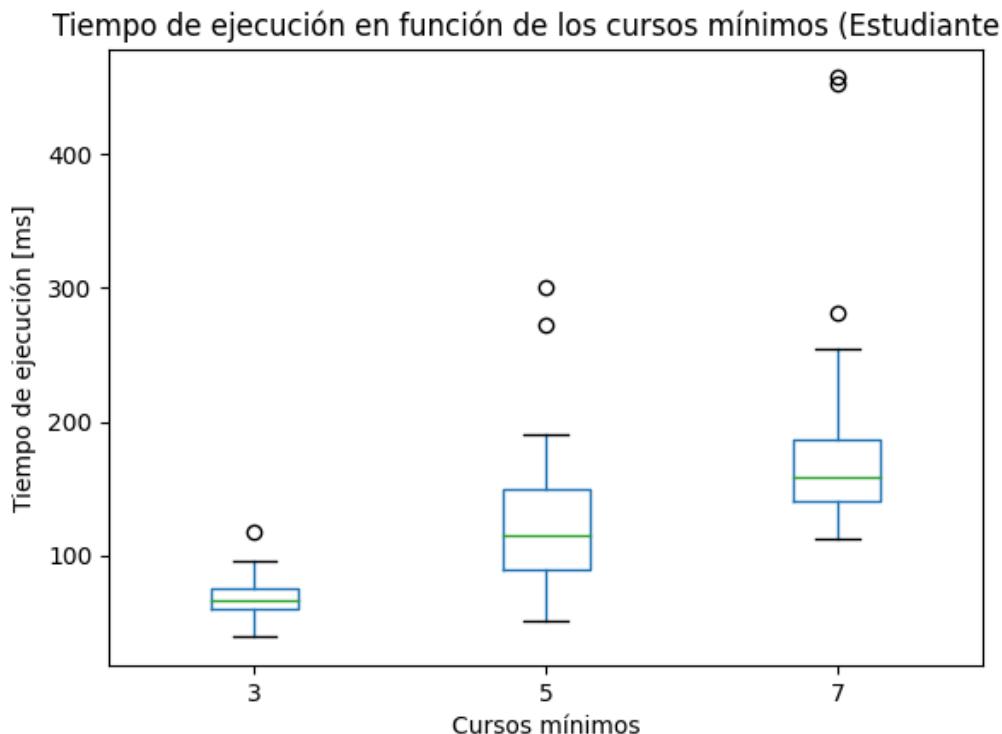
A partir de la figura 8 es posible observar cómo la cobertura según las horas mínimas aumenta notoriamente entre las 6 y 9 horas. Sin embargo, aún en los casos de 3 y 6 horas en el gráfico de la figura 9, la cercanía es muy alta, especialmente en el caso de 6 horas, donde el promedio es apenas inferior al 100%.

A pesar de ello, es posible visibilizar según el tiempo de ejecución en la figura 7 que esperar a alcanzar las 9 horas causa un aumento de medio segundo en comparación a la configuración de 6 horas, mientras que el empeoramiento entre 3 y 6 horas es menor, rondando 200 ms en promedio.

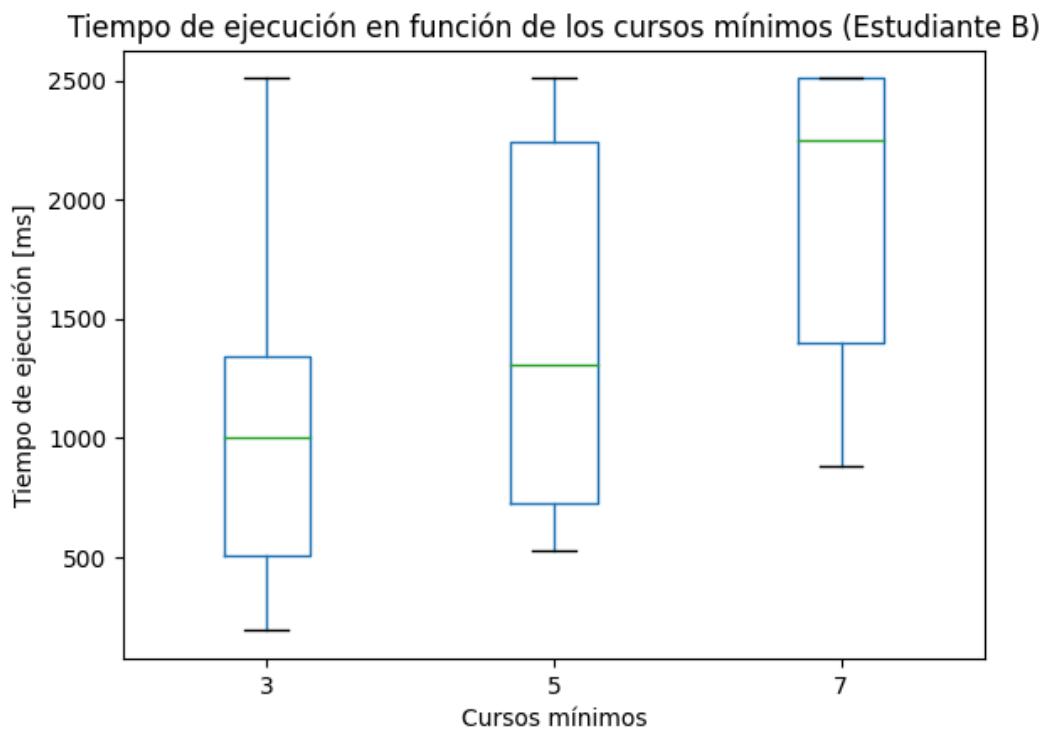
Por otro lado, para la figura 6 ocurre el fenómeno inesperado donde 6 horas presenta un mejor tiempo de ejecución que sus competidores, dado el bajo margen de diferencia en milisegundos observado.

Con estas observaciones, es claro que un valor de 9 horas ofrece resultados muy superiores que el resto a un costo de tiempo mayor. Al mismo tiempo, 6 horas ofrece un balance entre calidad y velocidad. Se optó por este último valor, dado que el tiempo ahorrado y la cercanía al puntaje máximo son suficientes para justificar el empeoramiento de cobertura.

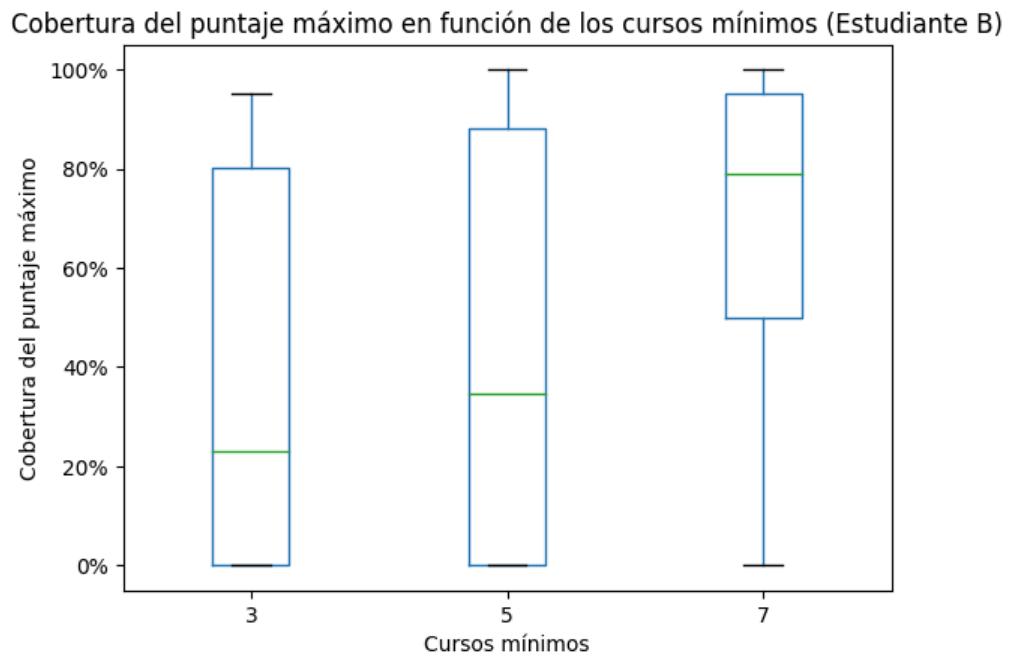
Para definir la cantidad de cursos antes de podar, se testeó con 3, 5 y 7 materias antes de podar, ahora con cero en el valor de horas mínimas. En resumen, la poda iniciará una vez que se hayan analizado todos los cronogramas que incluyan (o excluyan) las primeras asignaturas en la lista de prioridades pasada al algoritmo.



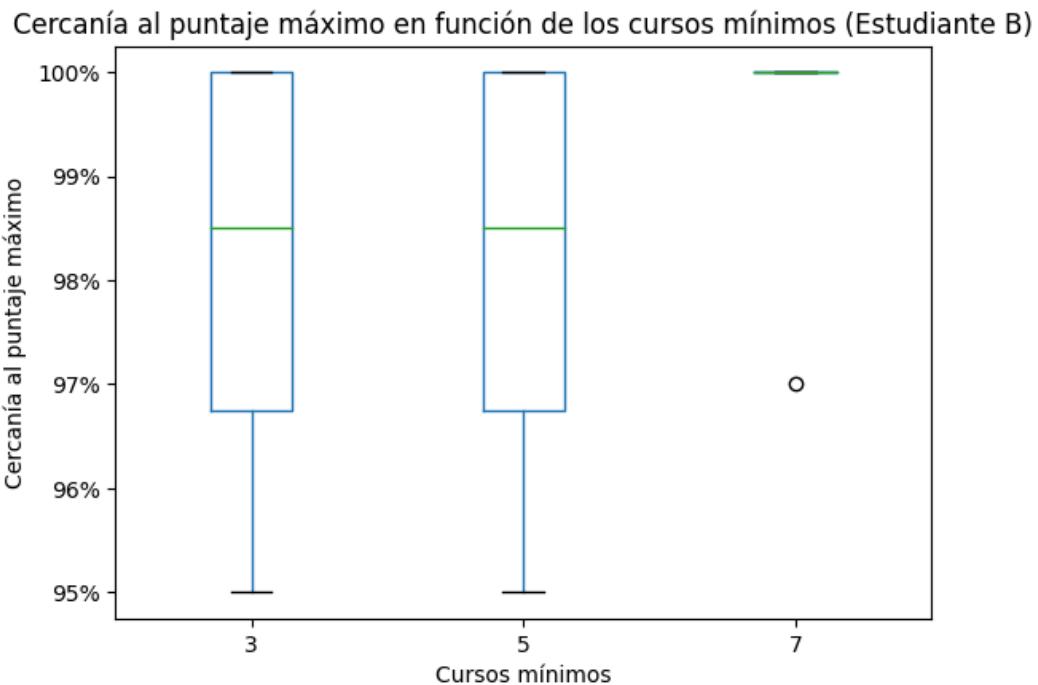
**Figura 10:** Tiempo de ejecución en función de cantidad de cursos procesados antes de podar para el estudiante A



**Figura 11:** Tiempo de ejecución función de cantidad de cursos procesados antes de podar para el estudiante B



**Figura 12:** Cobertura en función de cantidad de cursos procesados antes de podar para el estudiante B



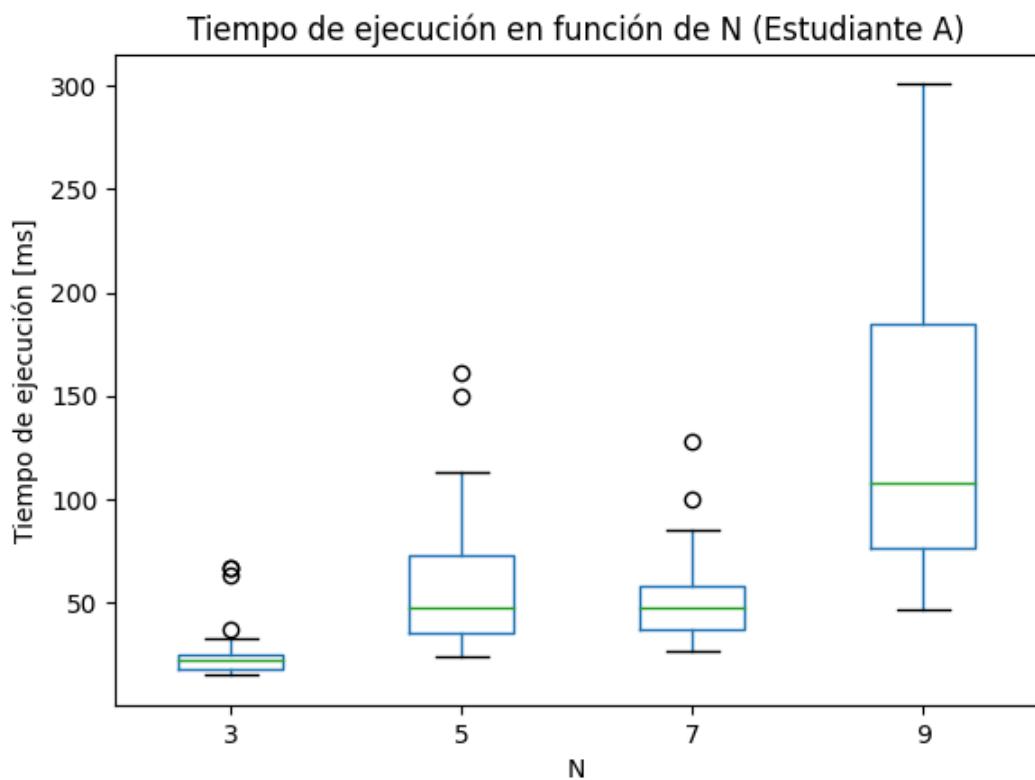
**Figura 13:** Cercanía al puntaje máx. en función de cantidad de cursos procesados antes de podar para el estudiante B

Para el caso de cursos mínimos, la cobertura surge un fenómeno similar al observado durante el análisis de las horas mínimas, como se puede observar en las figuras 12 y 13. Por otro lado, en este caso la diferencia en tiempo de ejecución entre cambios de parámetro es considerable para ambos estudiantes.

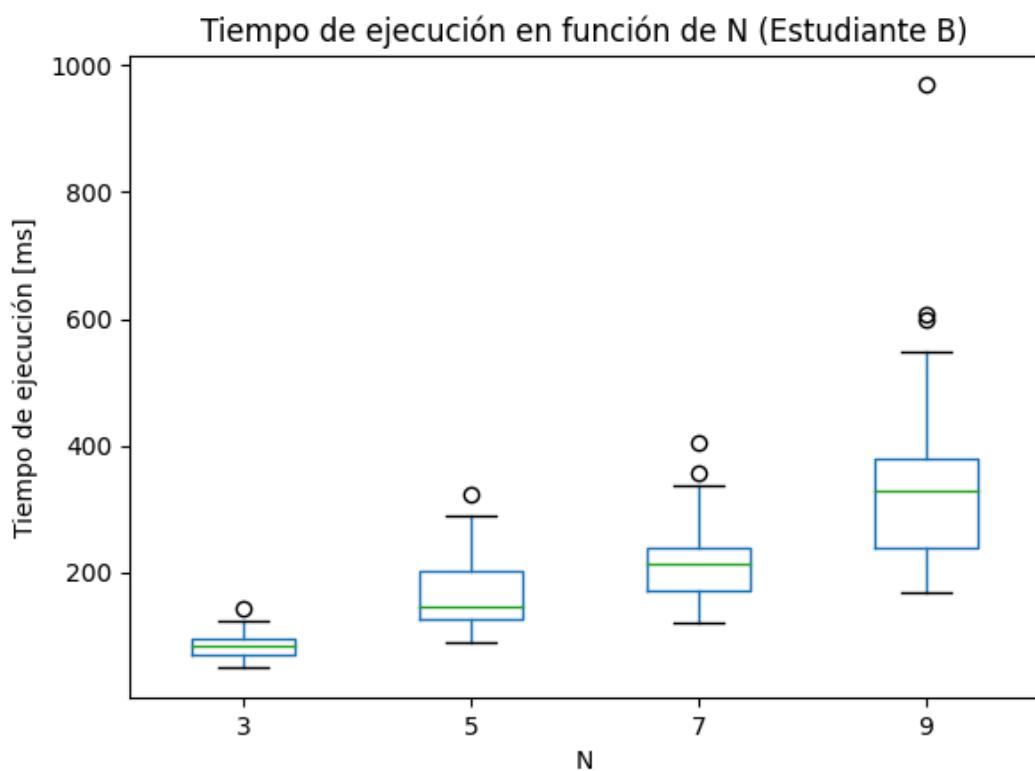
Por ello, a pesar de los prometedores resultados observados al tomar 7 asignaturas como valor, se decidió podar después de sólo 3, confiando que la condición de poda adicional de 6 horas por cronograma ayude a mejorar el puntaje de los cronogramas resultantes.

### *Algoritmo TimeGreedy*

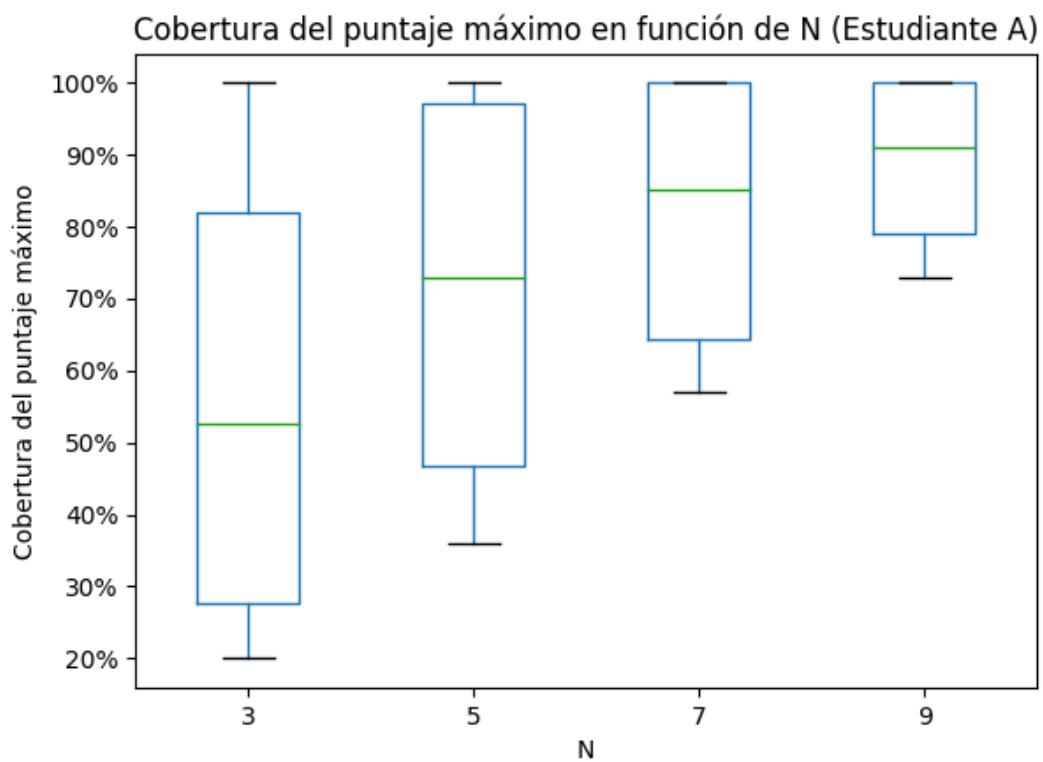
Se realizó un análisis sobre diferentes configuraciones de la cantidad de cronogramas almacenados por celda durante el algoritmo, denominado a posterior como N, para obtener su valor ideal. Se tomaron como opciones los valores 3, 5, 7 y 9 para ambos estudiantes, con 10 iteraciones para cada configuración. Se omite el gráfico de cercanía dado que para todas las configuraciones testeadas se alcanzó una cercanía del 100%.



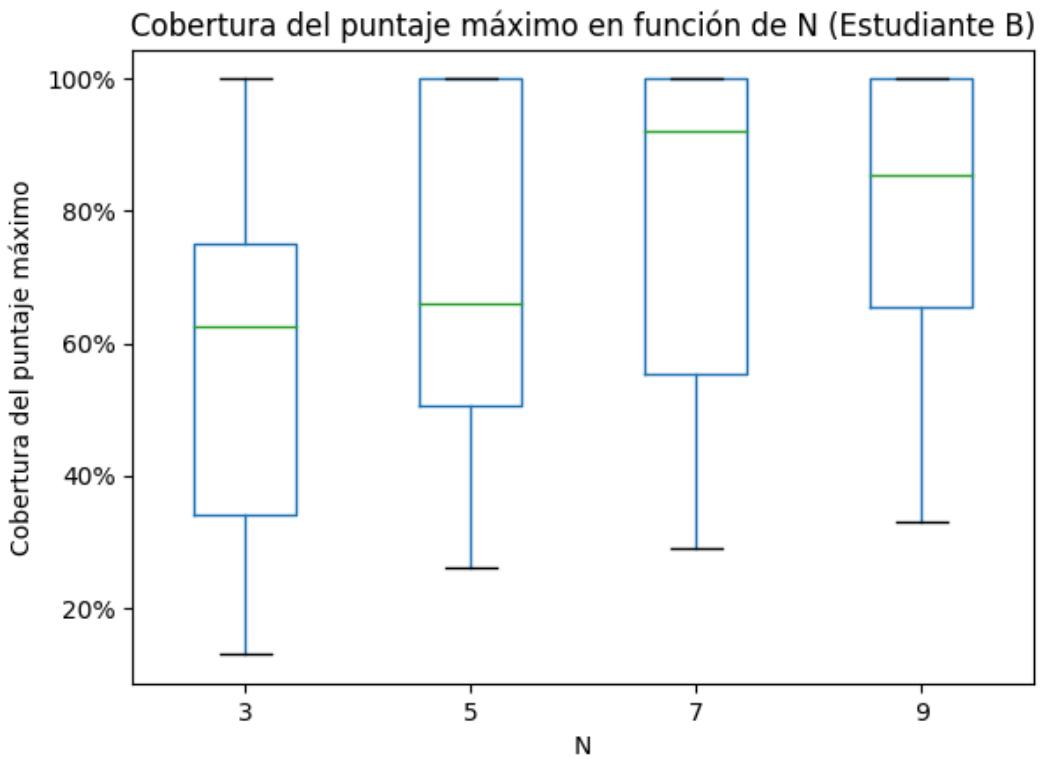
**Figura 14:** Tiempo de ejecución en función de N para el estudiante A



*Figura 15: Tiempo de ejecución en función de N para el estudiante B*



*Figura 16: Cobertura en función de N para el estudiante A*



*Figura 17: Cobertura en función de N para el estudiante B*

Con respecto a la cobertura, es posible apreciar, para ambos estudiantes, cómo en las figuras 16 y 17 aumenta toda la distribución de la cobertura junto al N, donde el tercer cuartil alcanza el 100% en ambos casos para el N más alto.

Por esta mejora constante, la decisión de qué valor de N tomar recae en cuánto sacrificio en el tiempo de ejecución uno está dispuesto a tolerar. Para el estudiante A en la figura 14 es visible que el salto ocurre con un N igual a 9, duplicando el segundo peor resultado promedio. Un fenómeno similar ocurre para el estudiante B en la figura 15, pero con efectos menos drásticos.

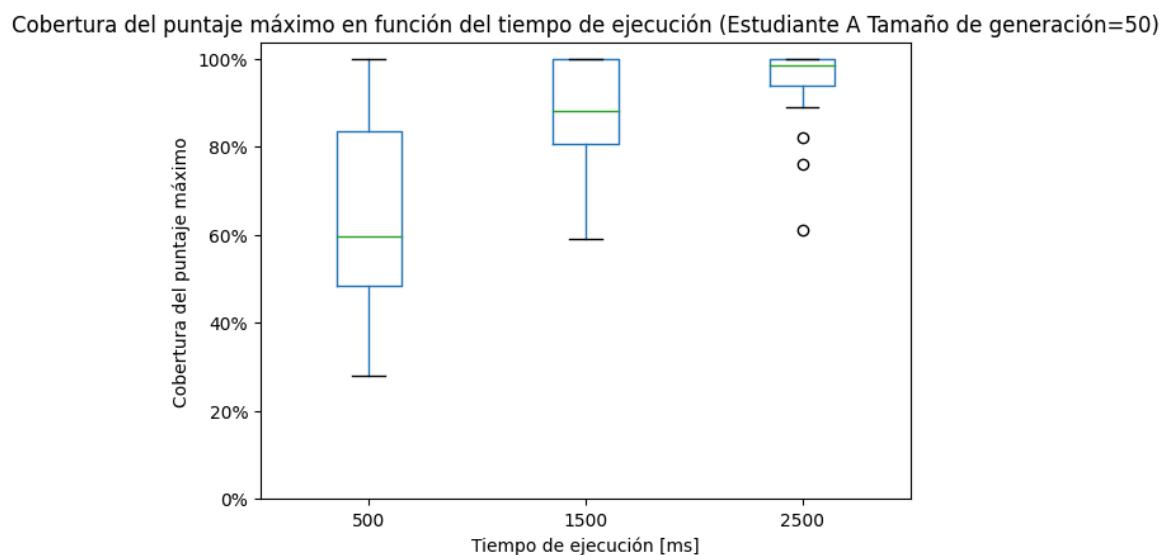
Como punto intermedio, se optó por un N igual a 7 como configuración ideal para este algoritmo.

### *Algoritmo Genético*

El análisis de los parámetros a definir para el algoritmo genético son el tamaño de generación y la cantidad de generaciones. Se utilizó un número de generaciones alto y se limitó la cota temporal a 500 ms, 1500 ms y 2500 ms, actuando como limitante sobre esta cantidad. Vale la pena mencionar que este último valor de 2500 ms es además la única cota definida para los algoritmos anteriores y por lo tanto, su máximo tiempo de ejecución.

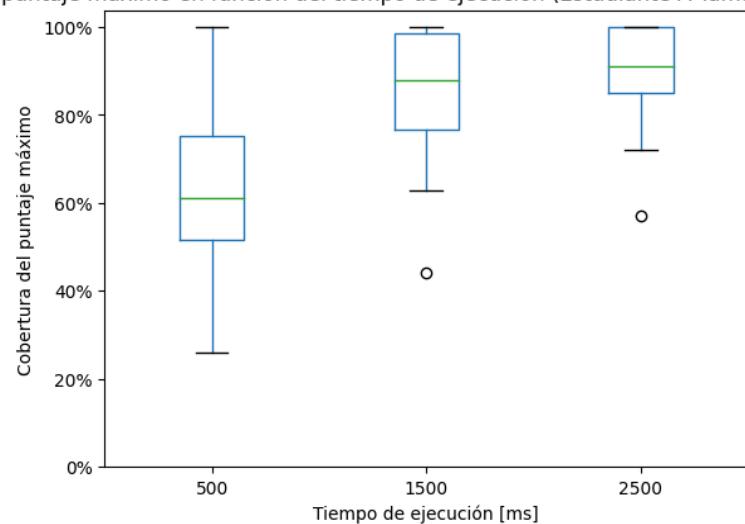
La estrategia tomada consistió en evaluar la cercanía o cobertura para diferentes tamaños de generación (estos siendo 50, 100, 200 y 400) bajo diferentes cotas temporales para obtener la mínima cantidad de generaciones antes de que deje de mejorar la cercanía de forma notable. Se utiliza la cobertura para el estudiante A y la cercanía para el estudiante B, dado que para este segundo la cobertura es de 0% para todos los casos.

Una vez obtenido el tamaño de generación y tiempo límite adecuado, se mide aproximadamente la cantidad de generaciones que se alcanza en dicho intervalo temporal para fijarlo como parámetro.



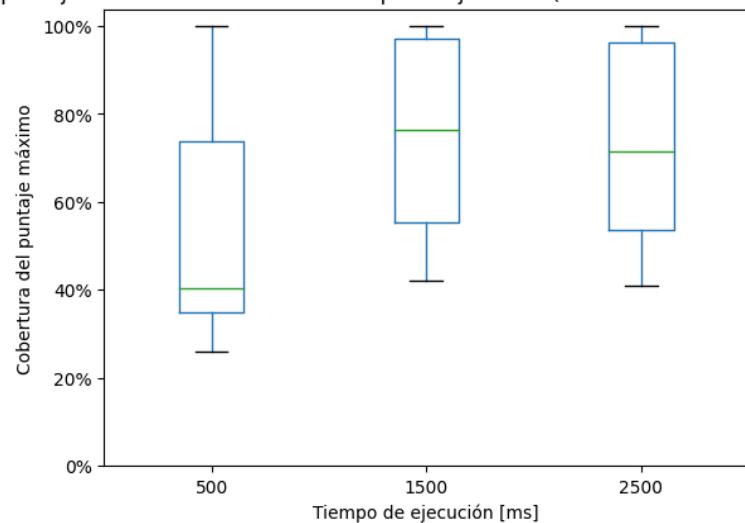
**Figura 18:** Cobertura en función de tiempo de ejecución con tamaño de generación 50 para el estudiante A

Cobertura del puntaje máximo en función del tiempo de ejecución (Estudiante A Tamaño de generación=100)



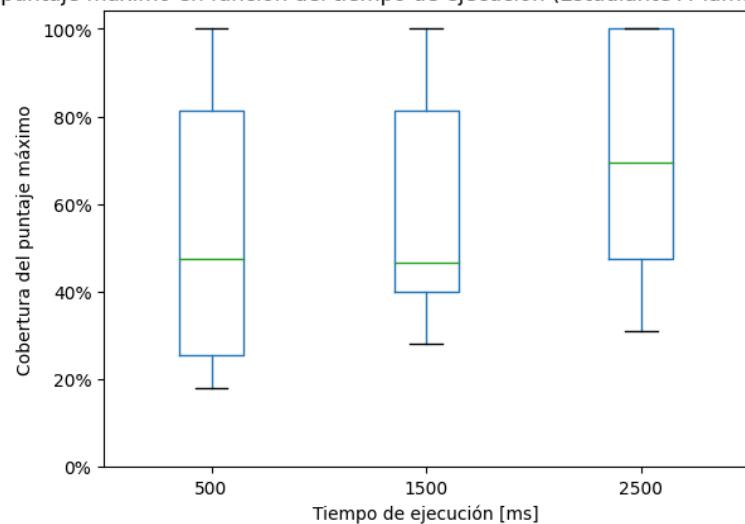
**Figura 19:** Cobertura en función de tiempo de ejecución con tamaño de generación 100 para el estudiante A

Cobertura del puntaje máximo en función del tiempo de ejecución (Estudiante A Tamaño de generación=200)



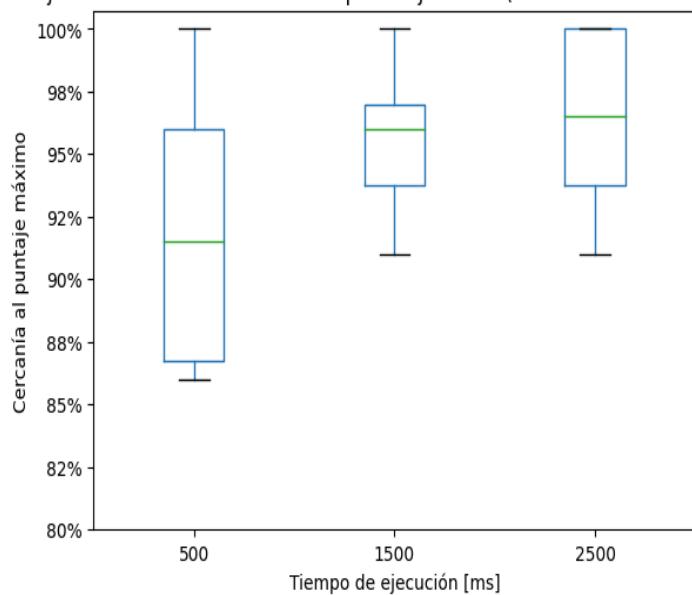
**Figura 20:** Cobertura en función de tiempo de ejecución con tamaño de generación 200 para el estudiante A

Cobertura del puntaje máximo en función del tiempo de ejecución (Estudiante A Tamaño de generación=400)



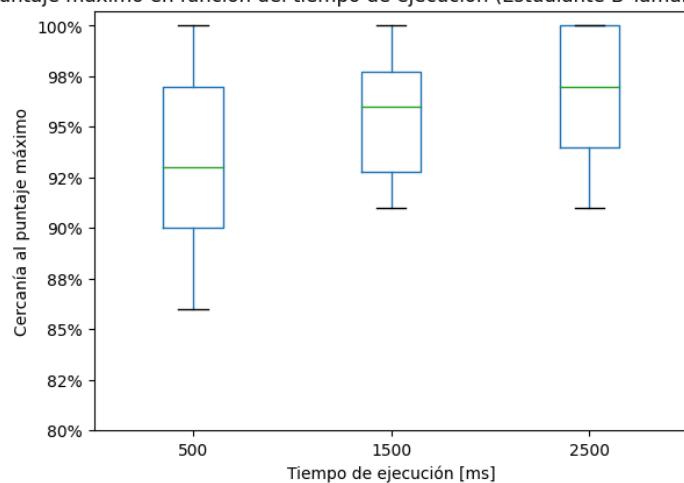
*Figura 21: Cobertura en función de tiempo de ejecución con tamaño de generación 400 para el estudiante A*

Cercanía al puntaje máximo en función del tiempo de ejecución (Estudiante B Tamaño de generación=50)



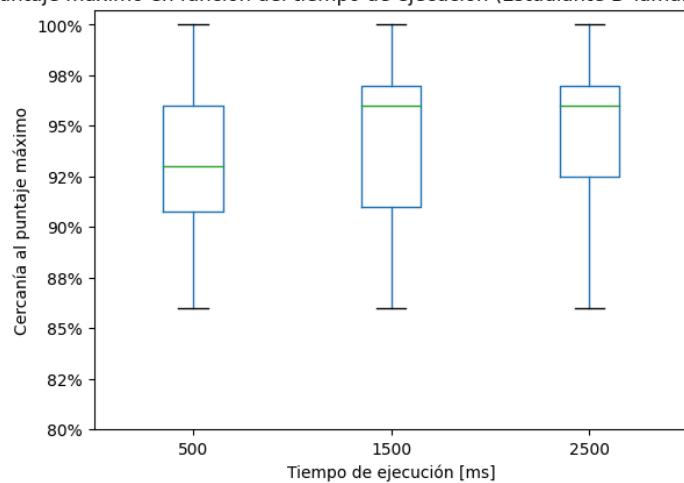
*Figura 22: Cercanía en función de tiempo de ejecución con tamaño de generación 50 para el estudiante B*

Cercanía al puntaje máximo en función del tiempo de ejecución (Estudiante B Tamaño de generación=100)



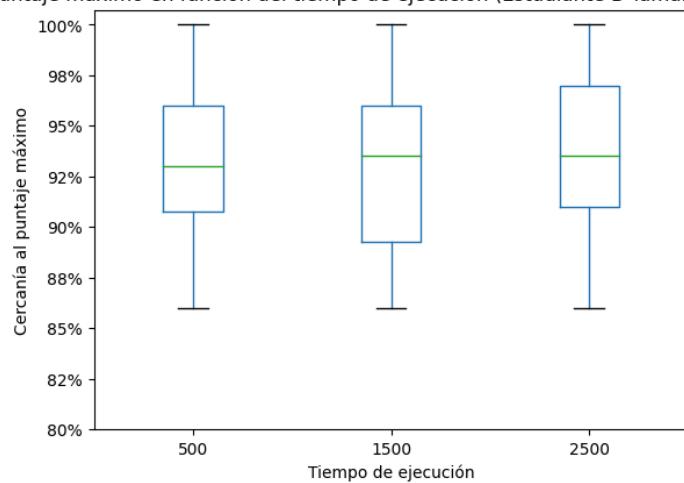
**Figura 23:** Cercanía en función de tiempo de ejecución con tamaño de generación 100 para el estudiante B

Cercanía al puntaje máximo en función del tiempo de ejecución (Estudiante B Tamaño de generación=200)



**Figura 24:** Cercanía en función de tiempo de ejecución con tamaño de generación 200 para el estudiante B

Cercanía al puntaje máximo en función del tiempo de ejecución (Estudiante B Tamaño de generación=400)



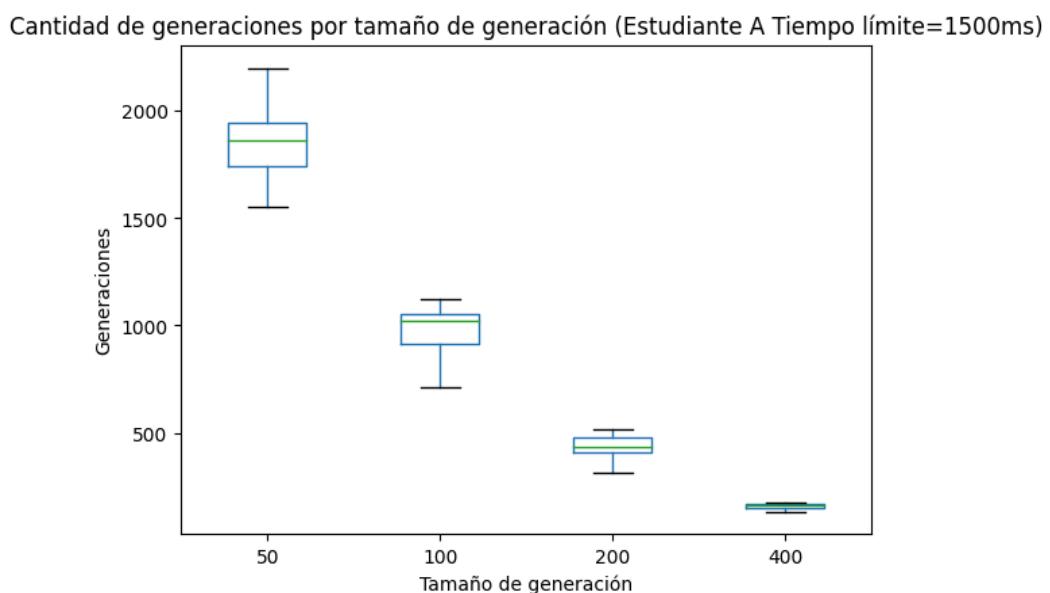
**Figura 25:** Cercanía en función de tiempo de ejecución con tamaño de generación 400 para el estudiante B

A partir de los datos visibles en las figuras 18 a 21, es posible observar como un mayor tiempo de ejecución no necesariamente implica una mejora en la cobertura, especialmente para el tamaño de generación 200 y 400, donde la mejora parece ínfima o inclusive inexistente. Por otro lado, para 50 y 100 en el tamaño de generación, la mejora con respecto al promedio de la cercanía se detiene alrededor de 1500 ms, con diferencias en el primer y tercer cuartil.

Ocurre un comportamiento similar para los datos de las figuras 22 a 25, esta vez para la cercanía, reafirmando el rendimiento con las generaciones pequeñas, destacando el valor de 50, el cual presenta mejor cobertura en comparación a su competencia en las cotas temporales de 1500 y 2500 ms.

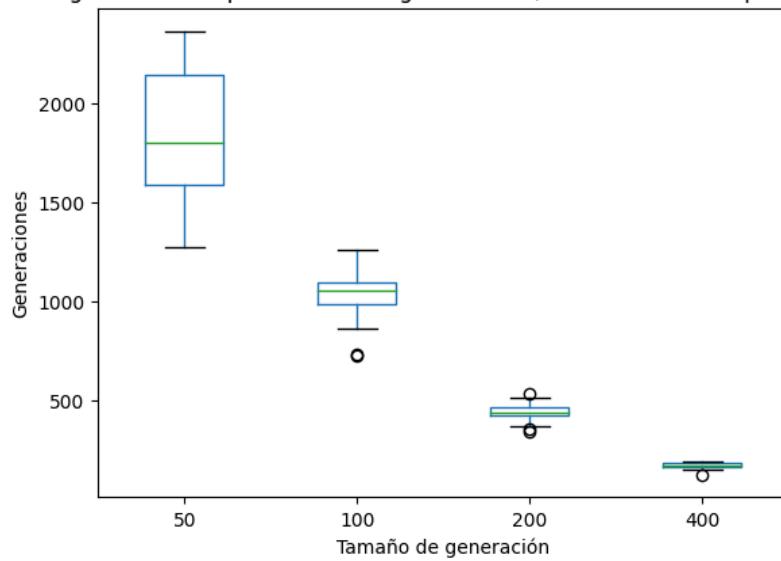
Se optó por 50 como valor ideal dado que presenta mayores probabilidades de proveer cronogramas con mejor puntaje.

Para el siguiente análisis, se definió una cota de 1500 ms ya que es un valor acorde a los casos observados en el algoritmo de poda, el cual se mostraba más lento que TimeGreedy.



**Figura 26:** Cantidad de generaciones exploradas en 1500ms, en función del tamaño de generación, para el estudiante A

Cantidad de generaciones por tamaño de generación (Estudiante B Tiempo límite=1500ms)



**Figura 27:** Cantidad de generaciones exploradas en 1500ms, en función del tamaño de generación, para el estudiante B

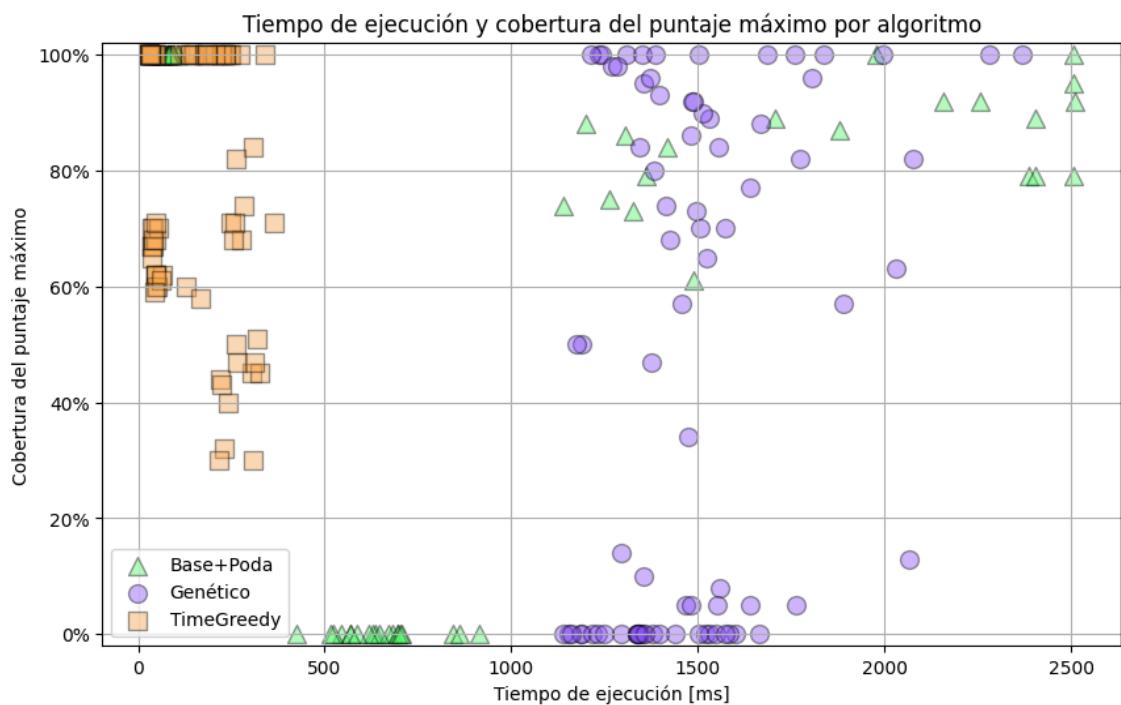
Tomando en cuenta la cantidad de generaciones generadas con 50 integrantes cada una, con tiempo límite de 1500 ms, las figuras 26 y 27 muestran cómo para ambos estudiantes se alcanzan alrededor de 1841 generaciones, tomando el valor ligeramente mayor 1900 para el parámetro de cantidad de generaciones.

Adicionalmente se puede observar la relación inversamente proporcional entre el tamaño de generación y la cantidad de generaciones, justificando como un mayor tamaño de generación no implica necesariamente una mejor cercanía cuando se fija una cota temporal, dado que al aumentar el tamaño de generación, más tiempo del disponible se empleará en procesar cada una de ellas.

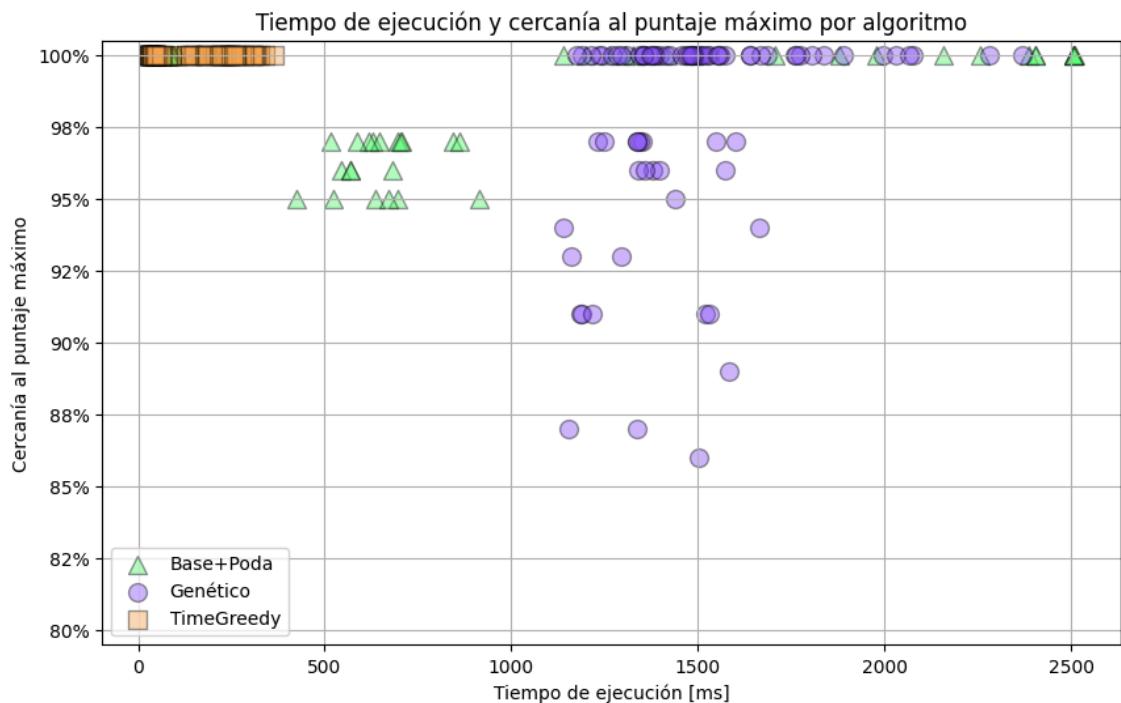
### Elección de Algoritmo

Tras hallar los parámetros óptimos para cada uno de los tres algoritmos, se llevó a cabo una comparación entre ellos repitiendo los experimentos con los dos estudiantes A y B, buscando nuevamente cronogramas con 18, 21, 24 y 27 horas deseadas para las mismas configuraciones detalladas en la sección anterior.

Luego de diez ejecuciones de cada valor de *horasDeseadas* y cada uno de los tres algoritmos con sus parámetros óptimos, se llega a los siguientes resultados:



**Figura 28:** Cobertura y tiempo de ejecución de cada algoritmo con cota máxima en 2500ms



**Figura 29:** Cercanía y tiempo de ejecución de cada algoritmo con cota máxima en 2500ms

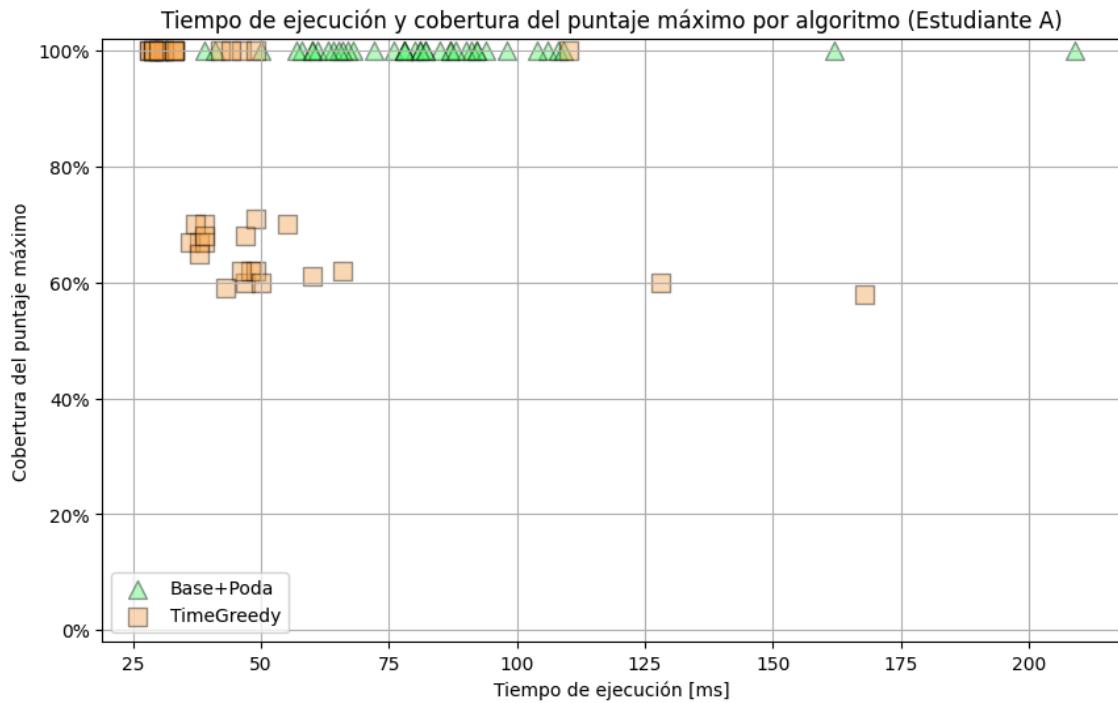
A partir de las figuras 28 y 29 se puede observar que TimeGreedy destaca por su bajo tiempo de ejecución en comparación a los otros algoritmos, manteniéndose similar para todos los casos testeados, con un tiempo menor a los 400 ms aún en los casos más lentos. Se aprecia que la cota inferior de cobertura es de aproximadamente 30%, cuando los otros algoritmos presentan varios casos con nula cobertura. Esto indica que, para las configuraciones testeadas, TimeGreedy es capaz de alcanzar el puntaje máximo con mayor probabilidad.

El algoritmo con poda genera resultados similares a TimeGreedy en algunas configuraciones, dejando lugar para un análisis de dichos casos, mientras que en otros presenta un tiempo de ejecución mucho peor, con algunos casos cercanos a la cota de tiempo permitido. Considerando el sector con nula cobertura entre los 500 y 1000 ms, se puede deducir que la poda está eliminando prematuramente las ramas que desencadenan en el mejor puntaje para lograr dicha duración. En el gráfico de cercanía queda evidente que el sacrificio realizado lleva a resultados alejados entre 2 y 5 puntos porcentuales del máximo puntaje posible.

El algoritmo genético presenta en general un peor desempeño en cuanto a tiempo de ejecución frente a los demás, sin una cobertura o cercanía alta que justifique dicha demora, dado que el algoritmo con poda alcanza resultados con mejor cercanía y un porcentaje de cobertura más consistente para un rango temporal dado.

A rasgos generales, las pruebas realizadas llevan a la conclusión de que TimeGreedy es un mejor algoritmo, mientras que el genético falla en las expectativas como competencia a los otros dos.

Para explorar el potencial del algoritmo base con poda, se hizo foco en los casos donde éste alcanzaba la cobertura máxima, obteniendo así el siguiente gráfico:



**Figura 30:** Cobertura y tiempo de ejecución del algoritmo base con poda y TimeGreedy con cota máxima en 2500ms para estudiante A

Observando los resultados para las configuraciones que incluyen al estudiante A, se puede apreciar como la diferencia de tiempo de ejecución entre ellas es de alrededor de 50 ms, mientras que el algoritmo con poda provee una cobertura del 100% en configuraciones donde TimeGreedy falla en lograrlo.

Considerando que este sacrificio en tiempo de ejecución es ínfimo respecto a la demora de la API en sí a cambio de la mayor cobertura, se llega a la conclusión de que en casos donde existe una baja cantidad de cronogramas posibles, es conveniente utilizar el algoritmo con poda por sobre TimeGreedy.

Para obtener lo mejor de los dos mundos, se añadió un parámetro configurable desde variables de entorno que defina cuál algoritmo utilizar según el techo de combinaciones posibles para las comisiones recibidas, sean sus cronogramas resultantes válidos o no. De esta forma, un caso donde el máximo de posibles combinaciones es relativamente bajo utilizará el algoritmo con poda. Asimismo, si el techo es superior al valor definido, se utilizará TimeGreedy para garantizar resultados en un plazo más corto.

Como punto medio entre las permutaciones disponibles para ambos estudiantes, se eligió un valor de 100 millones de posibilidades. Nuevamente, tanto el parámetro como el algoritmo a utilizar (incluyendo este modo automático) son configurables desde el archivo de variables de entorno del proyecto.

## Front-End

Para garantizar *usabilidad* en la capa de presentación, se diseñó una página web tal que el usuario disponga de una interfaz sencilla y atractiva mediante la cual pueda interactuar con la API y obtener cualquier información solicitada de la misma de manera ordenada y fácil de interpretar.

### Aspectos técnicos

El front-end de la aplicación fue desarrollado en JavaScript, con ayuda de ReactJS<sup>[26]</sup>, la cual es una librería gratuita y open-source mantenida por Meta que facilita la creación de Single-Page Applications. Éstas son páginas que dinámicamente reescriben el contenido del árbol HTML en vez de recargar toda la página, dando a los usuarios una experiencia más fluida.

Entre las tres principales herramientas de desarrollo front-end en JavaScript, siendo las alternativas el framework Angular<sup>[27]</sup> y la librería VueJS<sup>[28]</sup>, no se encontró una clara ventaja de una herramienta por sobre las demás considerando el alcance específico de la aplicación. Las tres opciones son regularmente utilizadas en ámbitos profesionales y cuentan con una extensa red de recursos por fuera de la documentación oficial. Ya que el equipo contaba con experiencia previa trabajando con React, finalmente se optó por utilizar esta herramienta con el fin de agilizar el desarrollo y ahorrarse el tiempo que implicaría aprender a utilizar un nuevo set de herramientas desde el comienzo.

Si bien la aplicación se ideó para ser utilizada desde una computadora, se tomaron recaudos para hacer que ésta sea responsive y utilizable en la mayor cantidad de tamaños y resoluciones posibles. De esta forma, por más que el usuario acceda desde el navegador en su dispositivo móvil, podrá navegar por la página y cumplir su objetivo sin problemas. Todo elemento de la aplicación se adapta a las dimensiones de la pantalla utilizada y mantiene un orden suficientemente estético. Sin embargo, para mantener campos visibles y legibles, algunas vistas no logran mantener el nivel estético deseado en resoluciones demasiado bajas. Por ejemplo, alinear cuatro pestañas horizontalmente resulta imposible si la resolución horizontal es demasiado acotada.

Para lograr esta interfaz responsive, se utilizó también el toolkit Bootstrap<sup>[29]</sup>. Esto permite importar y personalizar componentes visuales prefabricados, como las pestañas anteriormente mencionadas, grillas de filas y columnas, etc. Específicamente, se empleó el framework ReactBootstrap<sup>[30]</sup>, el cual adapta los componentes ofrecidos por Bootstrap para ser importados como componentes propios de React.

Para hacer la aplicación accesible a una audiencia más amplia, se utilizó el paquete React-i18next<sup>[31]</sup> para configurar la página web en varios idiomas. Actualmente el contenido está

disponible en inglés y español, y el idioma mostrado será el que esté configurado en el navegador. Si el idioma del navegador no es uno de estos dos, se utilizará inglés por defecto.

Para el almacenamiento del JWT<sup>[32]</sup> a utilizar al autenticarse ante la API, se utiliza el localStorage del navegador. Junto al token se almacena la información relevante del usuario activo, con el fin de asegurar la disponibilidad de sus datos sin necesidad de revisar la caché o realizar otra llamada a la API, además del horario de expiración del token. Éste último se revisará cada vez que se intente acceder a otra página de la aplicación o se intente realizar un nuevo llamado a la API. En caso de que el token haya expirado, éste se eliminará del almacenamiento y se redirigirá al usuario a la página de inicio de sesión para re-autenticarse.

El ruteo, es decir, el procesamiento de un componente dado un cambio en la URL, es manejado por el archivo App.js, donde el componente Router de React permite mapear un *path* a un contenido HTML, que en este caso serían nuestros componentes de React.

## Estructura del Proyecto

El proyecto de front-end está organizado según el siguiente árbol de directorios:

- **public**
  - **locales**
    - **en**
    - **es**
- **src**
  - **\_tests\_**
  - **components**
    - **Accounts**
    - **Common**
    - **Lists**
    - **Pages**
  - **resources**
  - **services**

En **public** se encuentran los recursos que utilizará el navegador al cargar la página. Se trata de archivos de internacionalización (con traducciones en inglés y español), íconos, los archivos de configuración estándar para aplicaciones web *manifest.json* y *robots.txt* y el *index.html* que importa los stylesheets y muestra el contenido.

Dentro de **src** podemos encontrar el código fuente. Sueltos en ese directorio se encuentran los archivos *index.js*, el cual carga el componente principal de la aplicación en la raíz del documento HTML a presentar al usuario, y *App.jsx*, el cual configura el uso del Navbar en todas las vistas y el enrutamiento del contenido según el URL.

En **resources** pueden encontrarse stylesheets, archivos que contienen constantes y el logotipo utilizado en el navbar de la aplicación. **services** contiene archivos JavaScript que se encargan de configurar la conexión con la API y el servicio de i18n para brindar traducciones a los navegadores. Junto a estas dos carpetas, **\_tests\_** contiene los tests de React, mientras que **components** agrupa los componentes según su utilidad.

**Accounts** contiene componentes relacionados con la autenticación. Es decir, los formularios de creación de usuario, restablecimiento de contraseña e inicio de sesión.

**Common** contiene elementos a ser utilizados dentro de otros componentes de mayor nivel. En su mayoría son componentes genéricos con la intención de ser reutilizados en toda clase de componentes; campos de formularios que siguen un estilo visual común, carteles de error, componentes de paginación, etc.

**Lists** agrupa componentes cuya función es listar objetos de algún tipo, como se vio en las distintas listas a las que puede acceder un usuario de tipo universidad en su página de inicio. Si bien la mayoría de éstas se utiliza en un único lugar, algunas como las listas de cursos son utilizadas por estudiantes y profesores dentro de diferentes componentes.

Por último, **Pages** contiene los componentes centrales de cada vista. Por ejemplo, páginas de formularios de creación y edición, la pantalla de inicio (la cual se construye cargando desde *Common* el componente que corresponde al tipo de usuario activo). Estas serían los componentes personalizados de mayor nivel, que importan los definidos en los tres directorios mencionados anteriormente.

## Diseño Inicial de Interfaces

A la hora de proponer una interfaz gráfica para interactuar con el sistema, un principio que guió el desarrollo fue el mantener la interfaz sencilla y ordenada. De esta forma se evitaría confundir a los usuarios al encontrarse con demasiadas opciones o una sobrecarga de información, lo que podría causar confusión entre opciones muy similares o hacerlos perder de vista la opción que buscaban, al encontrarse entre tantas alternativas.

A continuación, se describen las decisiones que se tomaron para construir la interfaz, previo a la presentación de un prototipo a un grupo de usuarios de prueba. Estos aportaron feedback valioso que influenció cambios en la interfaz que no se incluyen en las figuras presentes en esta sección, sino que se detallarán en la sección *Ajustes*.

## Elección de Colores

Tratándose de un proyecto desarrollado por estudiantes del ITBA, orientado en un principio, a otros alumnos de la misma institución (con la idea de ofrecer la herramienta a la institución en sí para integrarla con sus sistemas existentes), se consideró apropiado que la paleta de colores incluya distintos tonos de azul y blanco, siendo estos los colores de la universidad y frecuentemente utilizados en sus propias páginas web.

En contraste a los diferentes tonos de estos dos colores, se utilizó un verde azulado como color secundario para destacar componentes de la interfaz con un tono que complemente al azul oscuro tan comúnmente utilizado dentro de la aplicación.

Como color de fondo, ya que el blanco utilizado por defecto daba una sensación de vacío e incompletitud y además destacaba por sobre los componentes de color azul oscuro, se eligió un tono celeste de saturación baja para combinar con el azul principal sin desviar la atención de los componentes que se mostrarían por encima de él. De esta manera, el texto blanco utilizado sobre un componente de fondo azul sería claramente legible, y el color que rodea a dicho componente, al ser un tono más apagado, no competiría con el texto por la atención del usuario.

## Elección de Logotipo

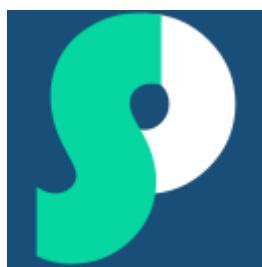
En todo momento durante el uso de la aplicación se verá el logotipo de *AutoScheduler* en la esquina superior izquierda, dentro del NavBar. El mismo utiliza los colores blanco y verde azulado mencionados anteriormente para destacar sobre el fondo azul oscuro del componente. De esta manera, se incorporan los dos colores principales mencionados anteriormente, en conjunto con un blanco neutro.



*Figura 31:* Logotipo de AutoScheduler

La tipografía seleccionada fue Bauhaus 93<sup>[33]</sup> debido a su prolífico atractivo visual. La simplicidad y forma de curvatura en las figuras da una sensación de orden, que es lo que el usuario buscaría al solicitar un cronograma que se ajuste a sus necesidades. En un principio se intentó darle una apariencia más original y dinámica, haciendo que la o minúscula y la S mayúscula estuvieran superpuestas para representar dos piezas encajando para formar una nueva figura.

Esta idea no prosperó pues el logotipo no resultaba suficientemente legible. Sin embargo, la figura resultante de esta cruza era lo suficientemente única como para utilizar una variante de la misma como ícono de la aplicación. Aún si el usuario no reconoce que las letras y sus colores son idénticas a las utilizadas en el logotipo completo, por lo menos podrá identificar la combinación de colores y una silueta particular.



*Figura 32:* Ícono de AutoScheduler

## Diseño original

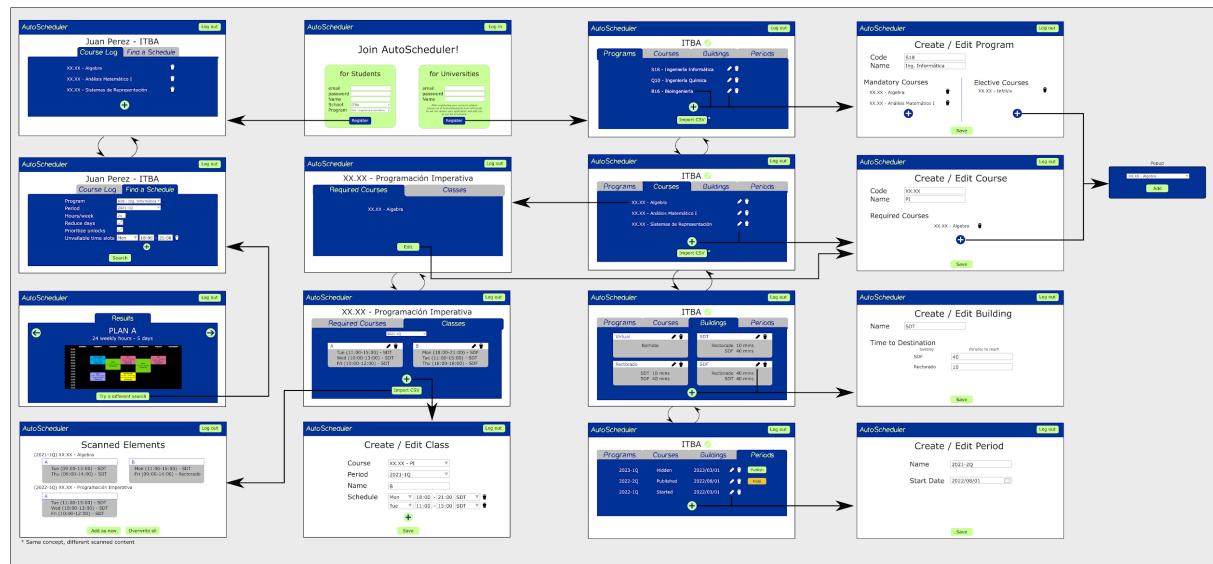
Buscando que el usuario no se demore en encontrar lo que busca al entrar a la página, un concepto que se decidió desde la primera propuesta de diseño fue tener disponible la mayor información posible en la página principal, distribuido entre distintas *pestañas* de un mismo contenedor; no más de cuatro de ellas a la vez. Así se logra categorizar las opciones presentadas

a la vez que se da a entender que las mismas tienen algo en común al formar parte del mismo componente a simple vista.

Para visualizar este concepto, se bocetó una serie de imágenes representativas de la vista correspondiente a la página principal del usuario Estudiante, donde sus únicas opciones serían una pestaña con el nombre “Materias Aprobadas” y otra titulada “Buscar Cronogramas”. La intención era simplificar las opciones visibles a sólo dos: Mantener y ordenar la información que se utilizaría para toda búsqueda futura de ese estudiante, y realizar la propia búsqueda configurando parámetros más variables (Por ejemplo, puede que de un período al siguiente el usuario decida cursar más horas o asuma nuevas responsabilidades que limiten su disponibilidad horaria)

Al mismo tiempo, se había diseñado una vista para visualizar los resultados de una búsqueda como una galería de tablas horarias que el usuario podría explorar de izquierda a derecha, en orden de qué tan bien se ajustan a sus necesidades. Debajo de cada tabla existiría un botón que redireccionaría a la pantalla anterior, formando un ciclo que abarca las funcionalidades clave para este tipo de usuario.

Partiendo de esa idea, se diseñaron vistas exclusivas para el usuario *universidad*, y se diagramó un flujo completo de la aplicación, obviando vistas correspondientes a componentes más sencillos o implícitos, como ventanas de confirmación al eliminar un elemento, o botones para trasladarse entre páginas de una lista larga.



**Figura 33:** Boceto de diagrama de flujo de la aplicación

En la figura 33 se muestra el diseño original de la aplicación y el flujo entre las distintas vistas. Varios detalles se han modificado en la versión final, y elementos como la funcionalidad

para cargar datos a través de un archivo CSV fueron omitidos en el producto desarrollado. Sin embargo, la mayoría de las vistas han respetado en mayor medida el diseño original.

El concepto de este diseño se conservó en la mayoría de los componentes desarrollados, añadiendo detalles que fueron omitidos por simplificación en el boceto de la figura 33. Los cambios serán mencionados en las siguientes secciones.

## Landing Page

Para dar la bienvenida a los usuarios cuando ingresen al sitio, se les mostrará una breve descripción de las funcionalidades que ofrece *AutoScheduler* a los estudiantes, con botones para iniciar sesión o registrarse.

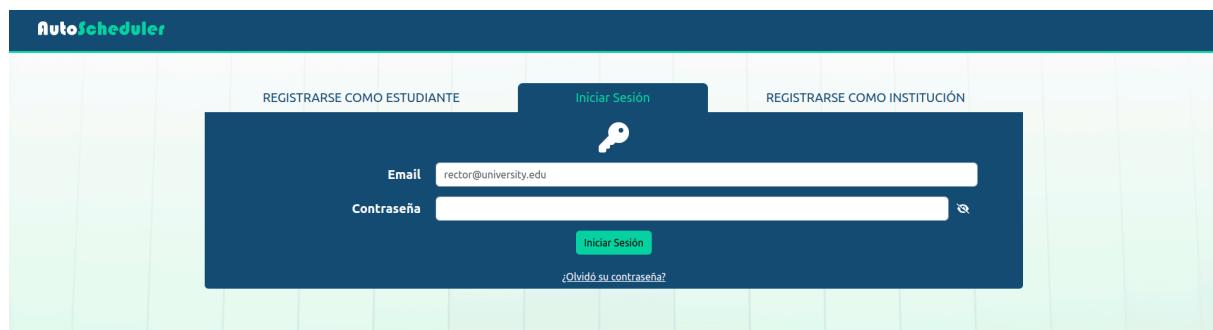


**Figura 34:** Vista del Landing Page

## Vistas de Autenticación

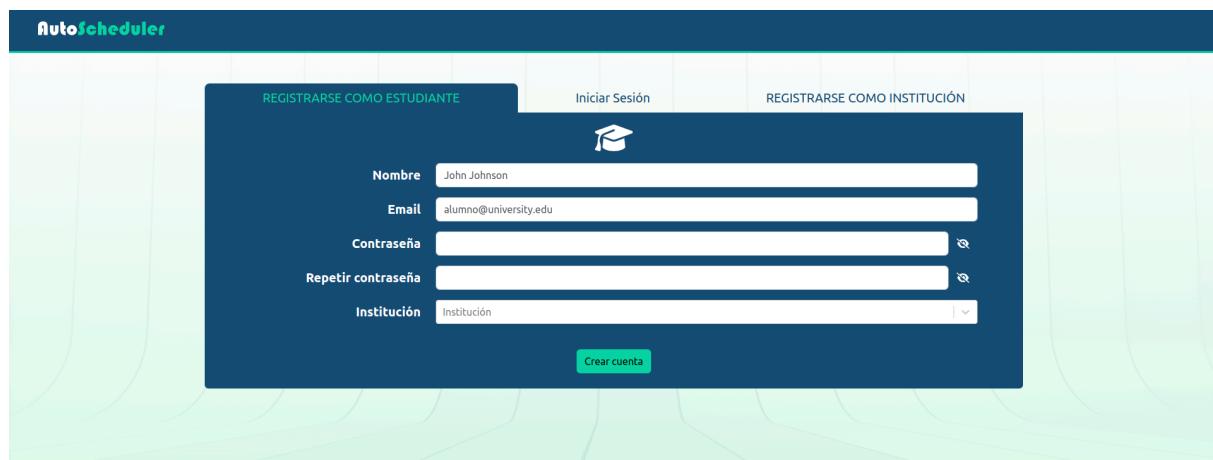
Antes de decidir añadir una landing page, se optó por rediseñar la vista de registro presente en el diseño original al observar que el botón de *Iniciar de Sesión* se encontraba en una posición fuera de foco, lo que obligaría al usuario a ponerse a buscar el botón y posiblemente no encontrarlo.

Esto resultó ser una oportunidad para reforzar el concepto de las pestañas como patrón recurrente dentro de la aplicación. Sin embargo, al ofrecer redirección a cada pestaña específica desde el landing page, la existencia de dichas pestañas en esta vista pasa a ser redundante, aunque siguen siendo un atajo en caso de que el usuario quiera ir a una de las otras opciones que se le habían ofrecido en la pantalla anterior.



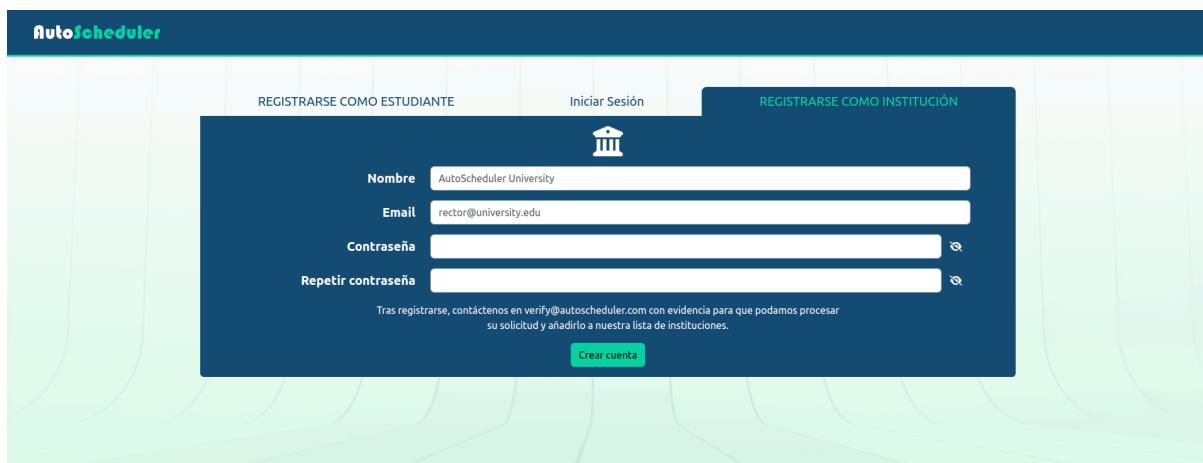
**Figura 35:** Vista de inicio de sesión

Para destacar el cambio de pestañas, se utilizó un ícono distinto dentro de cada formulario; un gorro universitario para el formulario de creación de alumno, una llave para el inicio de sesión, y un edificio de arquitectura clásica para la creación de universidades. Los mismos pueden apreciarse en las figuras 36 y 37.



**Figura 36:** Vista de registro de estudiante

Al intentar crear un estudiante, se le pedirá al usuario que, además de ingresar su nombre, email y contraseña, escoja su universidad y programa de una lista, la cual puede filtrar mediante ingreso de texto.



**Figura 37:** Vista de registro de universidad

Al crear una universidad, se le advierte al usuario que para poder figurar en la lista pública de universidades que ven los estudiantes al registrarse, deberá ponerse en contacto con el equipo que administra AutoScheduler para poder completar un proceso de verificación que garantice que la institución está siendo representada por un representante legítimo de dicha organización.

#### Página de Inicio para Universidades

A screenshot of the AutoScheduler website's homepage for university users. The header includes the 'AutoScheduler' logo, the user's name 'Newman University', the email 'newman@newman.edu.ar', and a 'Cerrar sesión' button. A pink message box at the top states: 'Su institución aún no ha sido verificada. Si aún no inició la solicitud, no olvide contactarnos en sterenziani@itba.edu.ar con pruebas de su vínculo con Newman University para que podamos verificar su cuenta y añadir la institución a nuestra lista.' Below this, another message says: 'Puede continuar usando AutoScheduler con normalidad para cargar información sobre sus sedes, períodos próximos, materias y carreras. Sin embargo, los estudiantes no podrán registrarse como alumnos de su institución hasta completar el proceso de verificación.' Below the message box is a search bar with the placeholder 'Búsqueda'. Below the search bar are four tabs: 'SEDES', 'PERÍODOS', 'MATERIAS' (which is active, shown in white text), and 'CARRERAS'. Under the 'MATERIAS' tab, there is a table with one row showing 'No hay elementos en esta lista.' with a '+' button below it. Navigation arrows are on either side of the table.

**Figura 38:** Vista de la página de inicio para usuarios de tipo Universidad

Una vez autenticados, los usuarios, sin importar su rol, podrán corroborar que ingresaron con los datos correctos en la esquina superior derecha de la pantalla, donde se les recordará el nombre y email asociado a la cuenta activa, junto a un botón para salir.

Al iniciar sesión, un usuario *universidad* será bienvenido con una alerta recordándole que su verificación se encuentra incompleta, si es este el caso. Debajo de la misma, encontrará cuatro pestañas dentro de las cuales se listan sus diferentes recursos.

Las pestañas se ordenaron de forma tal que agilicen el proceso de carga de datos en caso de un usuario que está viendo esta página por primera vez. Al no depender de otras entidades, las pestañas de sedes y períodos fueron ubicadas a la izquierda, seguido por las materias (las cuales opcionalmente dependen de carreras), y por último, las carreras (las cuales dependen de la existencia de materias para estar completas).

Originalmente se consideró colocar las pestañas de carreras y materias en el centro, a modo de facilitar el acceso a las mismas (ya que facilitarían el acceso directo a la página de un curso, sea desde la lista de éstas, o desde el listado de las que conforman una carrera). Sin embargo, esto fue cambiado posteriormente cuando se observó que los usuarios de prueba tendían a seguir un orden de izquierda a derecha al explorar la aplicación. Esto llevaba a casos donde intentaban crear una carrera antes de que hubiese materias definidas (por lo que la misma quedaba vacía), o comisiones para una materia antes de definir períodos (lo cual no está permitido).

Bajo este nuevo orden, el único escenario recurrente donde el usuario debería regresar para editar una entidad ya creada es para añadir correlativas a un curso, para lo cual se necesita una carrera con todas sus materias definidas. Sin embargo, este proceso se simplificará en una sección posterior, acompañado de otro reordenamiento de las pestañas.

Como pestaña predeterminada, se escogió la de materias con el fin de, una vez más, captar la atención del usuario con una pestaña asimétrica (al ser esta la anteúltima en la fila), redirigiendo su atención a las demás opciones disponibles. Si bien la pestaña de carreras se consideró como alternativa debido a que permitiría acceder a listas de asignaturas agrupadas por plan de estudios, el hecho de que esta pestaña fuese la última en la fila corre el riesgo de pasar desapercibida a diferencia de una solapa más cercana al centro.

## Páginas de Sedes

The screenshot shows the main dashboard of the AutoScheduler application. At the top, there's a header with the logo 'AutoScheduler' on the left, the text 'ITBA rector@itba.edu.ar' in the center, and a 'Cerrar sesión' button on the right. Below the header, there are four tabs: 'SEDES' (which is active and highlighted in blue), 'PERÍODOS', 'MATERIAS', and 'CARRERAS'. The 'SEDES' tab displays a list of university locations with their internal codes and names: 'Rectorado' (Sede Rectorado), 'SDF' (Sede Distrito Financiero), 'SDT' (Sede Distrito Tecnológico), and 'Virtual' (Aula Virtual). Each entry has edit and delete icons to its right. Below the list is a navigation bar with a left arrow, the text 'Página 1', and a right arrow.

**Figura 39:** Vista de la pestaña Sedes en la página de inicio para universidades

La primer pestaña lista las sedes definidas por la universidad. Si bien en el diseño original estas estaban representadas por tarjetas que listaban la distancia a otras sedes, se decidió descartar la idea debido a que el listado de distancias requeriría otro llamado a la API para mostrar información en parte redundante, y poco agradable a la vista en casos donde existan muchas sedes.

Dentro de esta lista se muestran las sedes y su código interno, ordenadas alfabéticamente por nombre. Cada una ofrece un botón para editar o eliminar sus datos.

Al final de la lista se encuentra el control de paginación. En caso de que haya resultados que no entren en la página (la cual tiene un tamaño de 20 elementos), se habilitarán las flechas a cada lado del número para permitir la navegación entre páginas.

Debajo de este componente se encuentra un gran botón con el símbolo (+) que permite añadir una materia, redirigiendo al usuario a la página de creación de ésta.

The screenshot shows the 'Crear Sede' (Create Site) form in the AutoScheduler application. At the top, there are fields for 'Código/ID de sede' (EZ) and 'Nombre de la sede' (Edificio Z). Below these, under the heading 'Distancia a otras sedes', are four entries: 'Rectorado' (45 mins.), 'SDF' (45 mins.), 'SDT' (45 mins.), and 'Virtual' (45 mins.). A green 'Guardar' (Save) button is located at the bottom right of the form.

**Figura 40:** Vista del formulario para crear o editar una sede

Al crear o editar una sede, se le permitirá al usuario cambiar el nombre y el código interno por el cual se refiere a la misma, así como definir la distancia entre ella y las demás sedes. La distancia es un atributo de la relación bilateral, por lo que al guardar los cambios, cualquier cambio a una distancia se reflejará también al editar la otra sede.

Las distancias están definidas en minutos, y por defecto toman el valor de 45. El mismo se eligió de modo tal que si una universidad olvidase definir la distancia, no se recomiendan a estudiantes cronogramas que requieran desplazarse de una sede a otra instantáneamente. En el caso donde estas dos resulten ser contiguas o muy cercanas entre sí, simplemente se le estaría dando al alumno un bloque de tiempo libre entre clases, que podría incluso resultar útil si ocurre en un horario donde le permita almorzar.

Por otro lado, si existe una distancia considerable entre estas sedes, 45 minutos debería ser suficiente para trasladarse en transporte público o un vehículo particular entre dos puntos de una ciudad grande como Buenos Aires. De esta forma, se asegura un valor de compromiso que no pondría en riesgo la puntualidad del estudiante, ni tampoco le dejaría demasiado tiempo muerto.

Al crear la primer sede, el campo de distancias será omitido, ya que si una universidad tiene un único edificio, no debe conocer ni preocuparse por el concepto de distancias entre sedes.

## Páginas de Períodos



The screenshot shows the 'Periodos' tab selected in the navigation bar. Below it, two periods are listed: '2024-C1' (1er Cuatrimestre 2024) and '2023-Q2' (2do Cuatrimestre 2023). Each period entry includes a date ('1 de marzo de 2024' and '2 de agosto de 2023'), edit and delete icons, and a 'Publicar' (Publish) or 'Ocultar' (Hide) button. At the bottom, there are navigation arrows for 'Página 1' and a '+' icon.

**Figura 41:** Vista de la pestaña Períodos en la página de inicio para universidades

Como siguiente recurso a cargar, se encuentran los períodos. Estos se listarán según su fecha de inicio, de más reciente (o futuro) a más antiguo. De esta manera, se mantienen a fácil alcance las entidades que el usuario querrá modificar con mayor probabilidad.

En la lista, junto a los botones para editar y eliminar, se encuentra un botón para publicar y ocultar un período. Se optó por un color llamativo cuando el mismo no está publicado, con el fin de captar rápidamente la atención del usuario en caso de que haya olvidado publicarlo una vez terminada la carga de datos para el mismo.

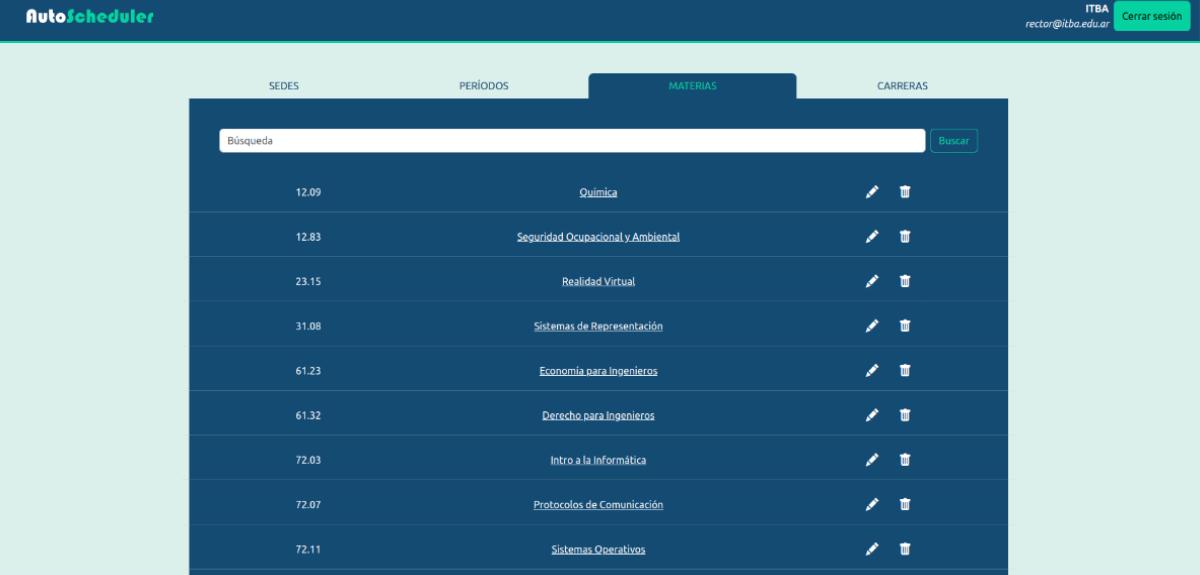
El propósito de este botón es ocultar los períodos cuya información aún está siendo cargada, y por lo tanto, no deberían ser visibles para los estudiantes hasta que la universidad decida hacer estos datos públicos. En un futuro, este cambio podría disparar el envío de una notificación para anunciar que las inscripciones se han habilitado para este período entrante.



The screenshot shows a modal window titled 'Crear Período'. It contains three input fields: 'Código/ID del período' (2024-C1), 'Nombre del período' (Semestre X, Año Y), and 'Fecha de inicio' (10/24/2023). Below the fields is a 'Guardar' (Save) button.

**Figura 42:** Vista del formulario para crear o editar un período

## Páginas de Materias y Comisiones

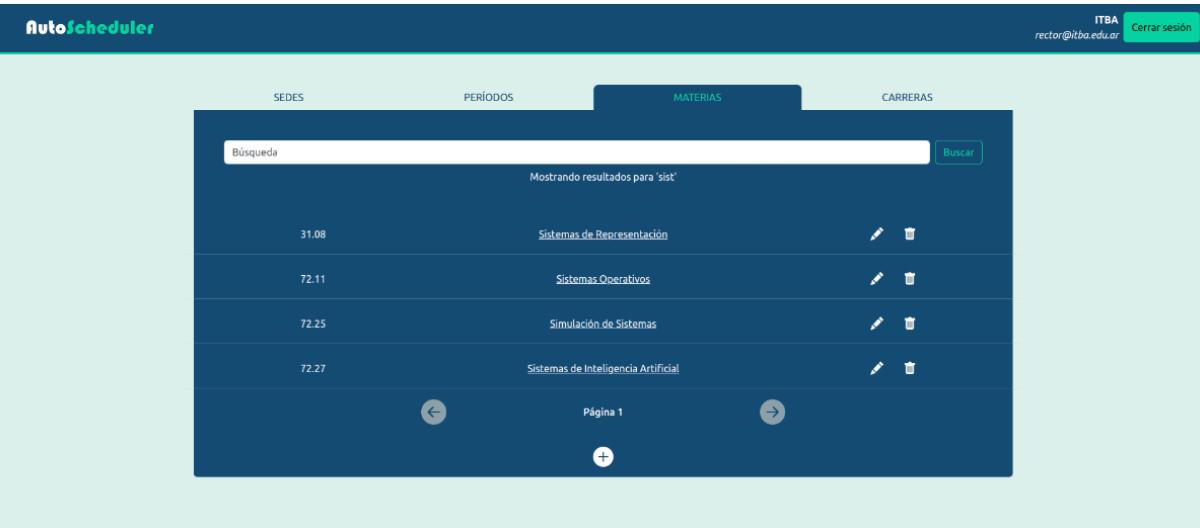


The screenshot shows the 'AutoScheduler' application interface. At the top, there are tabs for 'SEDES', 'PERÍODOS', 'MATERIAS' (which is the active tab), and 'CARRERAS'. Below the tabs is a search bar labeled 'Búsqueda' with a 'Buscar' button. The main content area displays a list of courses with their respective codes and names, each accompanied by edit and delete icons. The courses listed are:

Código	Materia	Acciones
12.09	Química	edit, delete
12.83	Seguridad Ocupacional y Ambiental	edit, delete
23.15	Realidad Virtual	edit, delete
31.08	Sistemas de Representación	edit, delete
61.23	Economía para Ingenieros	edit, delete
61.32	Derecho para Ingenieros	edit, delete
72.03	Intro a la Informática	edit, delete
72.07	Protocolos de Comunicación	edit, delete
72.11	Sistemas Operativos	edit, delete

**Figura 43:** Vista de la pestaña Materias en la página de inicio para universidades

La lista de materias ofrece, además de la misma información básica que las demás pestañas, un enlace sobre el nombre de cada elemento en la lista. Esto llevará a la página de curso, donde se podrá actualizar la información relacionada a sus comisiones y ver la lista de correlativas.



The screenshot shows the 'AutoScheduler' application interface, similar to Figure 43, but with a search filter applied. The 'MATERIAS' tab is active, and a search bar contains the text 'sist'. Below the search bar, a message says 'Mostrando resultados para "sist"'. The list of courses now includes only those containing the search term 'sist':

Código	Materia	Acciones
31.08	Sistemas de Representación	edit, delete
72.11	Sistemas Operativos	edit, delete
72.25	Simulación de Sistemas	edit, delete
72.27	Sistemas de Inteligencia Artificial	edit, delete

At the bottom of the list, there are navigation arrows for 'Página 1' and a '+' button.

**Figura 44:** Vista de la pestaña Materias en la página de inicio para universidades utilizando la búsqueda por texto

Considerando la larga lista de materias que componen una carrera, multiplicado por la cantidad de éstas, es esperable que el número de elementos listados en esta pestaña supere ampliamente la de las demás. Es por esto que se incluyó una barra de búsqueda para filtrar la

lista de cursos, facilitando la búsqueda de una materia particular sin tener que recorrer página por página.



**Figura 45:** Vista de la pestaña Correlativas dentro de la página de una materia

Al hacer click sobre el nombre de una materia, el usuario será redirigido a la página del curso. La misma consta de dos pestañas, y un botón que facilita el regreso a la página de inicio debajo del contenido.

La pestaña izquierda dentro de esta página listará las correlativas definidas para una carrera seleccionada entre las cuales forma parte este curso.

En caso de que el usuario encuentre un error entre esta información o quiera realizar modificaciones, tendrá un botón debajo de la lista que lo redirigirá a la página para editar la materia en sí.

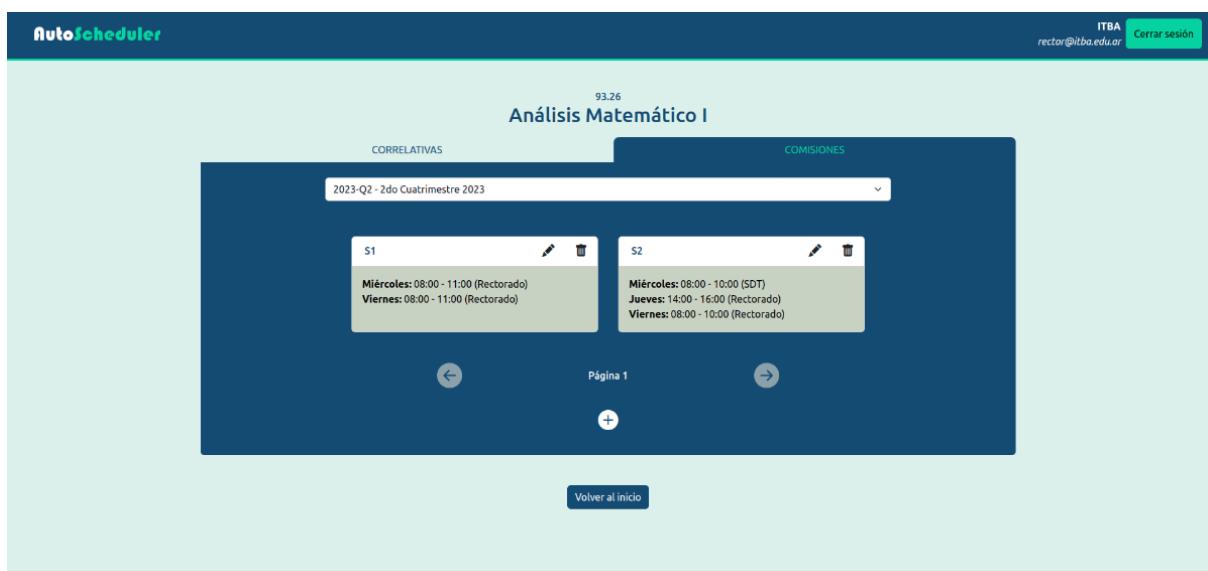


**Figura 46:** Vista del formulario para crear o editar una materia

Al editar una materia, además del código y nombre de la misma, se podrán definir la cantidad de créditos otorgados al alumno al aprobarla, y la lista de correlativas para cada carrera de la cual esta es parte.

Si se está creando una materia, este último campo será reemplazado por un mensaje haciéndole saber al usuario que luego podrá volver a esta página para actualizar sus correlativas una vez que la nueva asignatura forme parte de una carrera.

Adicionalmente, se decidió agregar un enlace a la lista de comisiones correspondientes al curso en este formulario en caso de que alguien pretenda acceder o modificar esa información desde la página de edición de la materia.



**Figura 47:** Vista de la pestaña Comisiones dentro de la página una materia

Volviendo a la página de la materia, la otra pestaña muestra las comisiones definidas para cada período y detalles sobre cada clase que la compone. A diferencia de la distancia entre sedes, se consideró que realizar llamados adicionales a la API para obtener los horarios de cada comisión estaba justificado, para poder identificarla en la lista por algo más que sólo su nombre.

Las comisiones pueden, como elementos de cualquier otra lista, editarse, eliminarse, y crearse desde esta vista.

**Figura 48:** Vista del formulario para crear o editar una comisión

Una comisión está definida por la materia a la que pertenece, el período durante el cual existe, su nombre (el cual actúa como identificador dentro de esa combinación curso-período) y sus clases. Cada clase tiene asociada alguna de las sedes, y pueden añadirse cuantas hagan falta al clickear el botón (+).

Una vez creada, la comisión puede editarse, pero su curso deberá permanecer fijo. Esto es para evitar errores accidentales donde el usuario no reconozca cuál comisión está editando, haciendo más difícil revertir el cambio si no se identifica el error de inmediato.

## Páginas de Carreras

SEDES	MATERIAS	CARRERAS
E11		Ingeniería Electricista
K07		Ingeniería Electrónica
P22		Ingeniería en Petróleo
S10		Ingeniería Informática
Q03		Ingeniería Química

**Figura 49:** Vista de la pestaña Carreras en la página de inicio para universidades

De vuelta a la página de inicio, la última pestaña lista las carreras existentes y ofrece la posibilidad de crear una nueva.

**Editar Carrera**

Código/ID del plan: S10

Nombre del plan: Ingeniería Informática

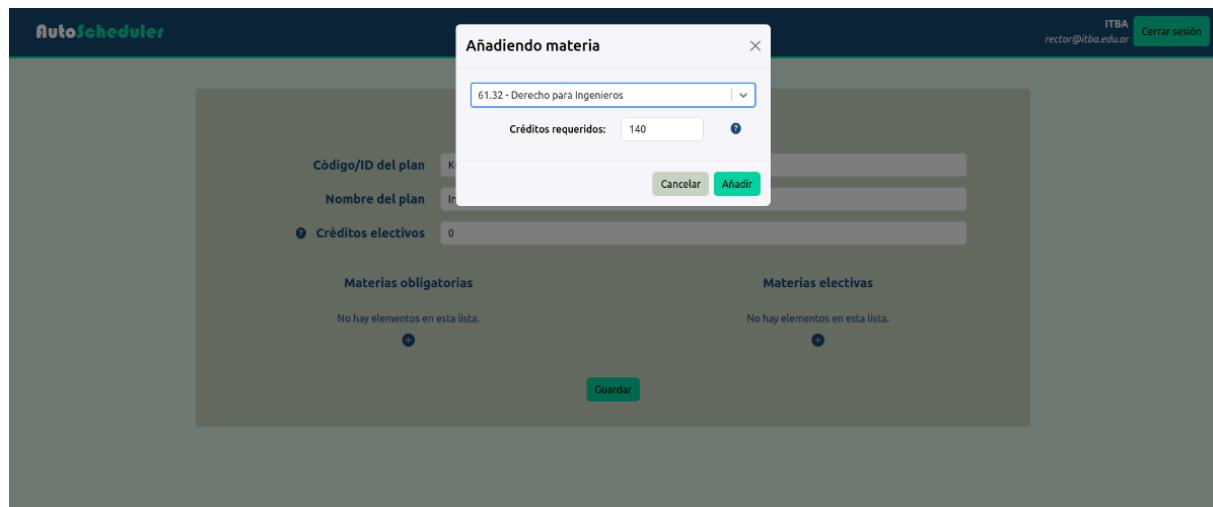
Créditos electivos: 27

Materias obligatorias		Materias electivas	
31.08	Sistemas de Representación	23.15	Realidad Virtual
72.27	Sistemas de Inteligencia Artificial (Requiere 140 créditos)	72.75	Machine Learning
72.31	Programación Imperativa	72.97	Introducción a la Programación de Videojuegos
72.45	Proyecto Final (Requiere 160 créditos)	82.08	Cloud Computing

**Guardar**

**Figura 50:** Vista del formulario para crear o editar una carrera

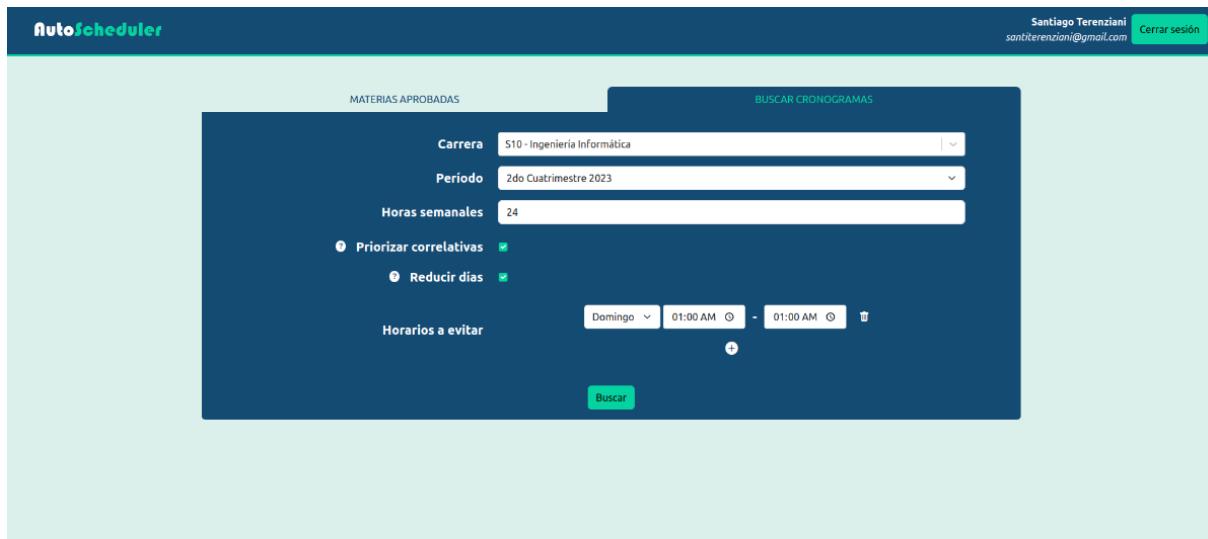
Al crear o editar una carrera, se puede definir nombre, código interno, la cantidad de créditos de electivas necesitados para finalizarla, y cuáles serán las materias obligatorias y las electivas bajo el programa de estudios.



**Figura 51:** Vista del modal para añadir una materia a una lista de materias de una carrera

Al cargar una materia a alguna de las listas, el usuario tiene la posibilidad de definir la cantidad mínima de créditos que un estudiante debe haber aprobado antes de inscribirse en ella.

## Página de Inicio para Estudiantes



**Figura 52:** Vista de la página de inicio para usuarios de tipo Estudiante, en la pestaña de búsqueda

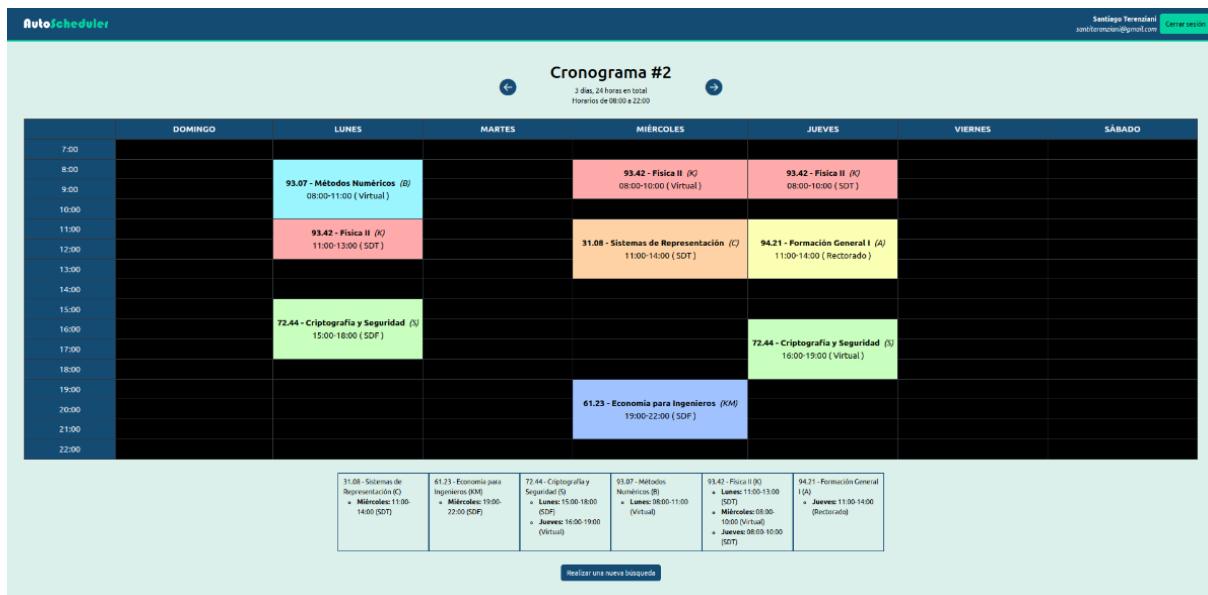
Al iniciar sesión, el estudiante se encontrará con el formulario para realizar su búsqueda. En él podrá seleccionar la carrera y período para los cuales buscar cronogramas, tildar casilleros para elegir los valores booleanos, elegir el número de horas semanales, y definir las franjas horarias entre las cuales no quiere cursar, definiendo cuantas franjas quiera a través del botón (+) y los íconos de cesto de basura.

Una vez definidas sus preferencias, solo debe clickear el botón para buscar resultados.



**Figura 53:** Vista de resultados de búsqueda antes de recibir respuesta del servidor

Mientras se espera la respuesta del servidor, se le solicitará al estudiante que aguarde unos minutos hasta que se tengan los cronogramas a sugerir.



**Figura 54:** Vista de resultados de búsqueda tras recibir respuesta del servidor

Una vez recibidos, se generarán componentes de tipo tabla para dibujar los distintos cronogramas en formato de agenda, tomando en cuenta la franja horaria entre las cuales hay clases (para no desperdiciar espacio en la pantalla antes de la clase más temprana y después de la última) y asignando un color a cada materia para facilitar la comparación visual al saltar de una propuesta de horarios a otra.

Debajo de la tabla horaria se muestra un resumen textual de las comisiones sugeridas, a modo de refuerzo en caso de que el redondeo de horarios a la hora de dibujar el cronograma haya distorsionado la distribución real, ya que la tabla redondea a la hora más cercana. En pantallas con resolución baja, este recuadro es lo único que el usuario podrá ver, ya que la tabla sería difícil de representar y leer adecuadamente con limitado espacio.

Si el estudiante no está convencido por el primer cronograma, por encima del mismo tendrá un resumen de la agenda actual y botones para navegar por la galería de opciones. Al pasar de un cronograma al otro, debería resultar fácil para el usuario seguir con la vista qué comisiones permanecieron fijas, cuáles cambiaron de lugar, cuáles desaparecieron, y cuáles son nuevas gracias al mapeo entre materias y colores que los identifiquen.

Si ninguno de los resultados fue suficientemente convincente, se ofrece la opción de regresar a la pantalla inicial con el botón debajo de cada cronograma, donde el usuario será capaz de ajustar sus parámetros de búsqueda.



**Figura 55:** Vista de la página de inicio para usuarios de tipo Estudiante, en la pestaña Materias Aprobadas

Una vez que un estudiante apruebe una materia, debería utilizar la pestaña alternativa en su página inicial para registrar la aprobación de la misma. De esa forma, la próxima búsqueda no tendrá en cuenta cronogramas que contengan esa asignatura, sino las correlativas siguientes a ella que el usuario esté habilitado para cursar.

#### Página de Inicio para Administradores

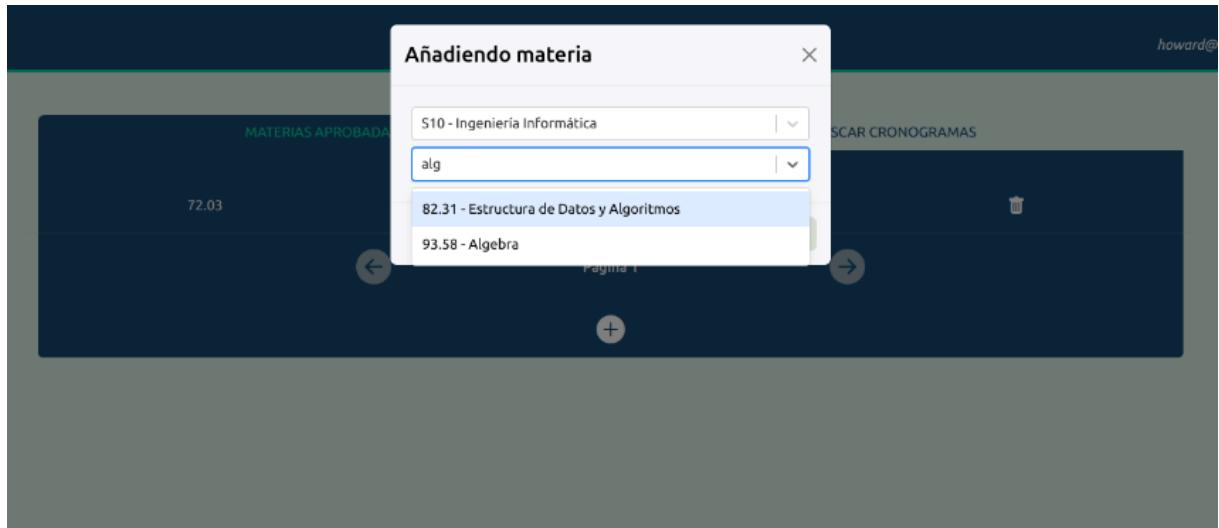
A screenshot of the AutoScheduler administrator dashboard. At the top, there's a header with the logo 'AutoScheduler', the role 'ADMIN', the user name 'admin', the email 'admin@autoscheduler.com', and a 'Cerrar sesión' (Logout) button. Below the header, there's a search bar labeled 'Busque una institución'. Underneath, there's a table listing four universities with their IDs and names, each with a 'Marcar como verificado' (Mark as Verified) or 'Revocar verificación' (Revoke Verification) button. The table includes columns for ID, Name, and Action. The background is light blue.

**Figura 56:** Vista de la página de inicio para usuarios de tipo Administrador

Cuando un usuario con el rol *administrador* inicie sesión, se encontrará con una lista de todas las universidades registradas en el sistema. Al lado de cada una, tendrá un botón para verificarla o revocar su estado como verificada una vez completado el proceso de verificación.

## Decisiones para Mejora de Experiencia del Usuario

A continuación, se ejemplifican algunas herramientas que el equipo implementó para mejorar la experiencia del usuario durante el uso de la aplicación.



**Figura 57:** Vista del modal para añadir materias a la lista de materias aprobadas de un estudiante

En muchas ocasiones, se ofrece al usuario un selector con una cantidad alta de opciones. Para facilitar la búsqueda dentro de ellos, se permite escribir texto para filtrar la búsqueda.

A screenshot of a web application titled 'AutoScheduler'. The top navigation bar includes the logo 'AutoScheduler', the text 'ITBA rector@itba.edu.ar', and a 'Cerrar sesión' button. The main content area is titled 'Editar Carrera' and contains a message: 'Ya existe un plan de carrera con ese código.' Below this, there are several input fields and validation messages:

- Código/ID del plan: 'S10' (no error)
- Nombre del plan: 'In' (validation message: 'El nombre del plan de carrera debe tener al menos 3 caracteres')
- Créditos electivos: '-6' (validation message: 'La cantidad de créditos de materias electivas requeridos debe ser un número igual o mayor que 0.')
- Materias obligatorias: 'No hay elementos en esta lista.' (with a '+' button)
- Materias electivas: 'No hay elementos en esta lista.' (with a '+' button)

At the bottom center is a green 'Guardar' button.

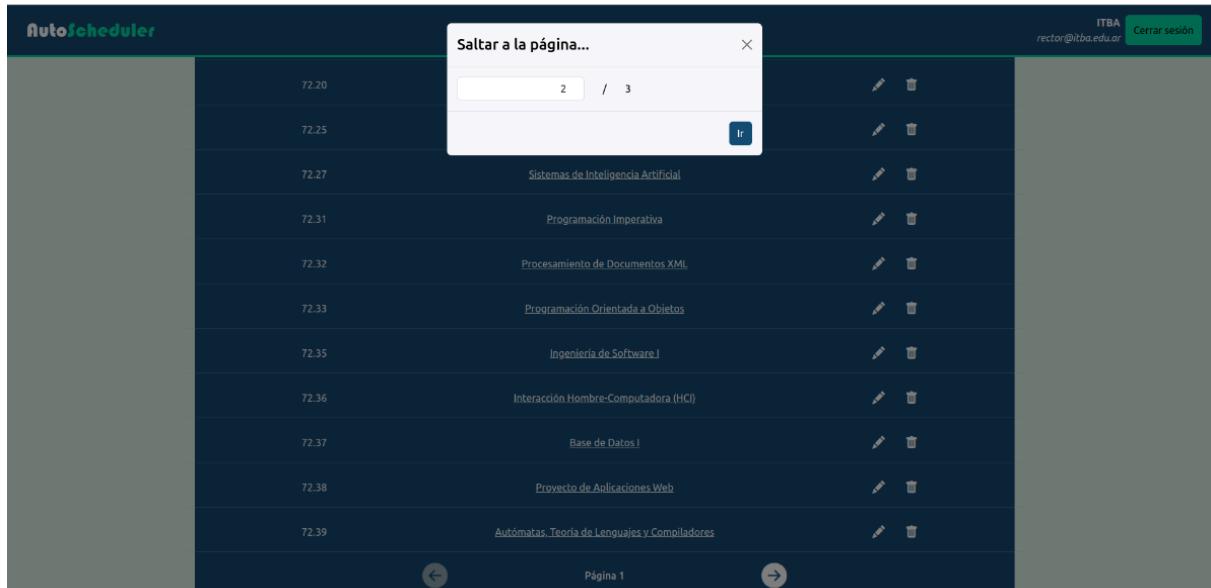
**Figura 58:** Vista de un formulario con errores

Cuando un valor en un formulario sea rechazado, se le notificará al usuario lo antes posible para que pueda corregir su error. Algunos errores, principalmente aquellos relacionados

con formato de texto, pueden detectarse antes de enviar el formulario, mientras que otros serán detectados al recibir una respuesta del servidor. Por ejemplo, si un código único ya está en uso.

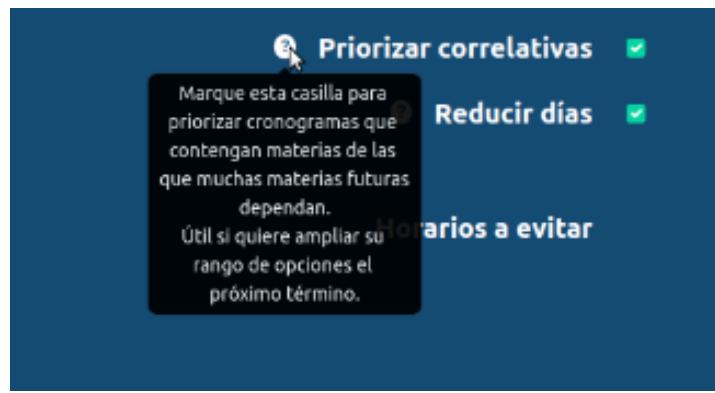


**Figura 59:** Vista del tooltip al mover cursor sobre el número de página



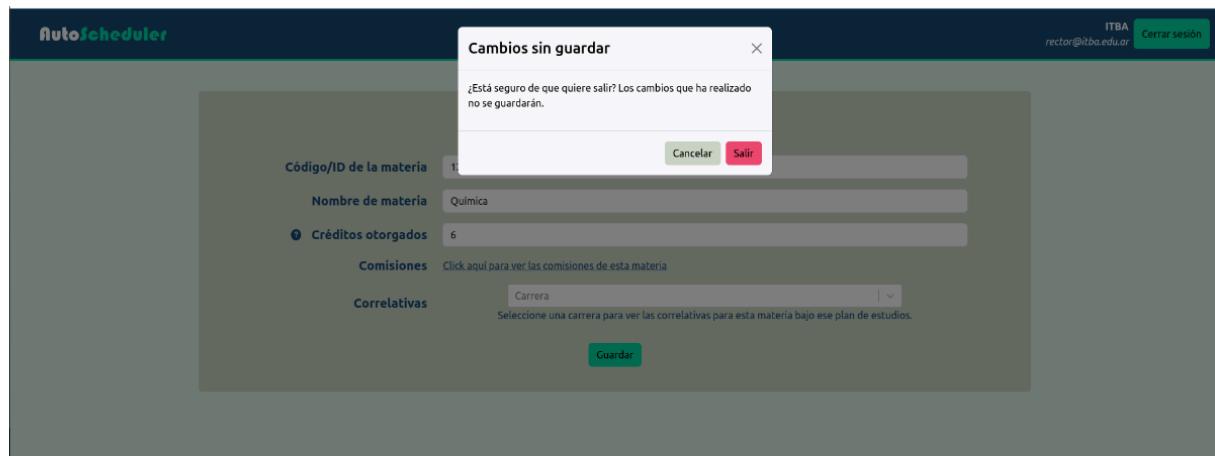
**Figura 60:** Vista del modal para saltar a una página específica

En el componente de paginación, para que el usuario no tenga que clickear el botón de flecha múltiples veces con el fin de alcanzar un número alto de página, se añadió un diálogo al pasar el cursor sobre el número de página, dejando en claro que se puede hacer click ahí para seleccionar un número de página específico en lugar de desplazarse página por página.



**Figura 61:** Vista de un tooltip al mover cursor sobre el signo de interrogación

En campos de formulario donde las definiciones pueden resultar ambiguas, se implementó un Tooltip, representado por un signo de interrogación dentro de un círculo como aparece en la figura 61, que muestre información adicional al deslizar el cursor sobre el mismo.



**Figura 62:** Vista del modal de confirmación de pérdida de cambios

Por último, al haber modificado un formulario, si el usuario intenta salir de la página se mostrará una alerta para detenerlo y recordarle que debe guardar sus cambios primero.

## Pruebas de Usuario Iniciales

Una vez que se contaba con un prototipo funcional, se realizó una pequeña prueba de usuario con dos adultos de 50 y 53 años, con experiencia moderada utilizando páginas web adquirido a través del uso casual en su vida cotidiana y en su puesto de trabajo. El objetivo de este experimento fue recibir feedback proveniente de un usuario poco experimentado para hallar puntos de mejora en la versión existente en ese entonces, antes de pasar a una fase de prueba más masiva.

Se les presentó a estos dos sujetos el prototipo en la pantalla inicial, con el formulario de inicio de sesión, y se les pidió que crearan una universidad y cargaran datos. A pesar de que uno de ellos interpretó los campos como un formulario de registro, al no poder iniciar sesión se dio cuenta de su error y comenzó a buscar dónde registrarse.

Una vez creada la cuenta y redirigidos al Home para universidades, se recibieron críticas acerca del mensaje de verificación incompleta. El mensaje utilizado en ese entonces era considerado ambiguo y poco claro, dándole al usuario la impresión de que no podría utilizar la aplicación hasta completar el proceso. Por ello, se redactó el mensaje de forma más clara, haciendo énfasis en el hecho de que el representante de la universidad puede comenzar a cargar datos mientras el proceso de verificación está en trámite, y qué implica exactamente el estar verificado.

Esta queja se extendió además a conceptos con los que los sujetos no estaban tan familiarizados. Ya que en su formación académica no se utilizaba el concepto de créditos, mostraron confusión al ver campos que utilizaban este término. Por ello se agregaron más tooltips con un signo de interrogación para que un usuario, ante la duda, pueda recibir una descripción más detallada de qué valor se debería ingresar en dicho campo.

En una nota similar, la pestaña de Correlativas dentro de la página de una materia resultó confusa al no contener más que un selector de carreras. Para despejar dudas, se agregaron explicaciones de qué puede esperar encontrar el usuario dentro de ese componente y aclaraciones de cómo realizar determinadas acciones en caso de que no les parezcan obvias.

**Editar Carrera**

Código/ID del plan	S10
Nombre del plan	Ingeniería Informática
Créditos electivos	27

Para definir correlatividades, haga click sobre el nombre de la materia para la cual quiere definir correlativas.

Materias obligatorias			Materias electivas		
93.58	Algebra	delete	82.08	Cloud Computing	delete
93.26	Análisis Matemático I	delete	72.97	Introducción a la Programación de Videjuegos	delete
93.28	Análisis Matemático II	delete	72.75	Machine Learning	delete

**Figura 63:** Tip para definir correlativas desde la vista de una carrera

Otro inconveniente surgió del orden de las pestañas. Como se mencionó anteriormente, la idea original era tener las pestañas de Materias y Carreras en el centro. Sin embargo, ambos usuarios exploraron la página pestaña por pestaña, yendo de izquierda a derecha. Esto marcó un importante error en la distribución de pestañas, ya que en más de una ocasión el sujeto vio limitadas sus opciones. Al crear una carrera, no contaba con materias disponibles para agregar. Y al crear una materia, no contaba con períodos para crear una comisión. Por este motivo, se decidió reordenar las pestañas para permitir un flujo más natural de la carga de datos.

El comentario más inesperado provino del componente selector con filtro de texto. Al descubrir que era posible escribir dentro del campo, el usuario se sentía frustrado al intentar seleccionar el texto dentro del recuadro de input (el cual mostraba el valor anteriormente seleccionado) para poder borrarlo. Éste fue un comportamiento imprevisto por parte del equipo, ya que el mismo estaba familiarizado con el concepto de presionar una tecla para saltar a las opciones con esa inicial. En consecuencia, se modificó el componente de forma tal que el texto correspondiente a la opción seleccionada se oculte al hacer click sobre el buscador.

En una nota similar, el otro usuario se vio confundido al ingresar a la página del curso y no encontrar la forma de regresar a la página de inicio. En ningún momento consideró presionar el botón de regreso en el navegador, ni el logo de *AutoScheduler* en el navbar. Es por esto que se añadieron botones ofreciendo volver al Home en estas páginas.

## Resultados

Para confirmar que la solución propuesta facilita el proceso de búsqueda de cronogramas para el estudiante, es fundamental que el proceso de buscar los mismos sea rápido e intuitivo. De otro modo, sólo se le estaría ofreciendo otra alternativa de similar o mayor complejidad a la metodología que ya utiliza.

La aplicación debe ser igualmente intuitiva y cómoda para los usuarios de tipo universidad, ya que son éstos los responsables de cargar los datos relevantes para que sus estudiantes puedan utilizar la herramienta en primer lugar.

## Metodología

Para validar que la usabilidad de la aplicación sea lo suficientemente buena para garantizar el uso correcto por parte de usuarios inexpertos, se definió una serie de tareas que usuarios de tipo universidad y estudiantes deberían poder completar sin asistencia, y se realizaron experimentos con diez sujetos de prueba de diferentes rangos etarios, ocupaciones y niveles de alfabetismo digital.

Considerando que esta herramienta sería utilizada principalmente por administrativos y oficinistas de una universidad, se buscó que al menos la mitad de los usuarios de prueba vinieran de ámbitos educativos. Entre los 10 sujetos, una es la directora de una escuela de idiomas, dos eran estudiantes trabajando como secretarias en dicha escuela, dos son docentes en la Universidad Provincial de Ezeiza, y uno había ejercido como docente en la Universidad de Palermo antes de jubilarse. Adicionalmente, uno de los testers es un desarrollador front-end que aportó sugerencias para mejorar aspectos del diseño de la interfaz.

En cada paso se definió un objetivo, y el sujeto de prueba fue dado libertad de encarar el problema con mínimas intervenciones por parte del equipo, el cual fue cronometrando el tiempo transcurrido hasta cumplirlo y observando el comportamiento del usuario para detectar comportamientos imprevistos o dificultades a la hora de cumplir una subtarea.

Las pruebas se llevaron a cabo en entornos silenciosos y sobre un escritorio, permitiéndole al sujeto de prueba utilizar un mouse o touchpad según su preferencia. Según su disponibilidad, estos entornos incluyeron habitaciones de su domicilio y aulas vacías u oficinas personales de su lugar de trabajo.

Una vez finalizadas las pruebas con cada usuario, se le solicitó su opinión sobre la interfaz y una opinión concisa acerca de su usabilidad.

## Paso 1

Al iniciar la prueba, se sentó al sujeto frente a una computadora corriendo la aplicación en la vista de la Landing Page. A partir de ahí, se le pidió que cumpla con el primer objetivo:

- Registrarse como usuario de tipo Universidad

## Paso 2

Una vez que el usuario hubiera creado su cuenta de universidad sin datos cargados, se le dio un objetivo mucho más amplio, con el fin de incentivarlo a explorar la aplicación en búsqueda de herramientas que ayuden a su caso de uso.

- Crear una carrera Ingeniería
  - Esta carrera tiene 3 créditos de electivas
  - El plan de esta carrera incluye las materias:
    - Matemática (otorga 9 créditos)
    - Física I (otorga 6 créditos)
    - Física II (otorga 6 créditos)
    - Física III (otorga 6 créditos, correlativa con Física I)
    - Química I (otorga 3 créditos)
    - Química II (otorga 3 créditos, correlativa con Química I)
    - Proyecto Final (otorga 12 créditos, requiere 24 para cursarse)
    - Economía (otorga 3 créditos, electiva)
    - Historia (otorga 3 créditos, electiva)
- Crear una carrera Exactas
  - Esta carrera tiene 3 créditos de electivas
  - El plan de esta carrera incluye las materias:
    - Matemática (otorga 9 créditos)
    - Física I (otorga 6 créditos)
    - Física II (otorga 6 créditos)
    - Física III (otorga 6 créditos, correlativa con Física I)
    - Economía (otorga 3 créditos)
    - Proyecto Final (otorga 12 créditos, requiere 18 para cursarse)
    - Química I (otorga 3 créditos, electiva)
    - Historia (otorga 3 créditos, electiva)

### Paso 3

Con las carreras y sus materias cargadas, se estudió la forma en la que el usuario intentaba crear comisiones:

- Crear una comisión de Matemática para el 1º Cuatrimestre de 2024 con los horarios:
  - Martes 9-13 en Lanús
  - Jueves 9-14 en Lanús
- Crear otra comisión de Matemática para el 1º Cuatrimestre de 2024 con los horarios:
  - Martes 17-22 en Lanús
  - Jueves 18-22 en Lanús
- Crear una comisión de Física I para el 1º Cuatrimestre de 2024 con los horarios:
  - Lunes 16-18 en Lomas de Zamora
  - Martes 14-16 en Lanús
  - Viernes 16-18 en Lomas de Zamora

### Paso 4

Para cerrar con esta sección, se dio una última tarea clave para cualquier usuario de tipo universidad.

- Publicar el período “1º Cuatrimestre de 2024”

### Paso 5

Evaluando la habilidad del sujeto para entender cuándo y cómo debe cerrar su sesión, se le dio un nuevo objetivo. A partir de este paso, se utilizó una cuenta de usuario universidad con datos reales del ITBA, obtenidos desde su Sistema de Gestión Académica, mencionado anteriormente:

- Registrarse como un estudiante de Ingeniería Informática del ITBA

## Paso 6

Una vez registrados como estudiantes, se estudió la capacidad de encontrar la pestaña de materias aprobadas y cómo añadir registros:

- Registrar las siguientes materias como aprobadas:
  - Algebra
  - Análisis Matemático
  - Sistemas de Representación
  - Metodología del Aprendizaje
  - Introducción a la Informática
  - Programación Imperativa

## Paso 7

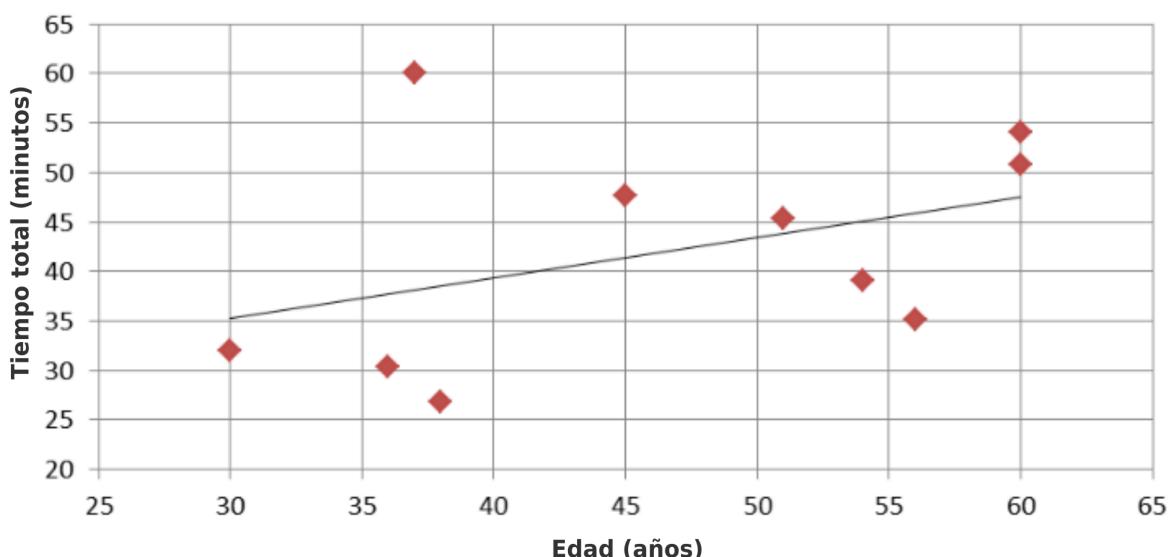
Finalmente, se le dio al usuario una lista de restricciones para guiarlo en el relleno del formulario para buscar cronogramas, de forma tal que pueda ver el listado de combinaciones resultante y cómo se ajustó a las preferencias definidas:

- Buscar cronogramas considerando los siguientes comentarios:
  - Quiero buscar cronogramas para el primer cuatrimestre de 2024.
  - Conseguí un trabajo los lunes, miércoles y viernes de 8 a 13.
  - Prefiero cursar materias que, en caso de aprobarlas, habiliten mayor cantidad de materias para las cuales son correlativas.
  - Preferiría cursar alrededor de 20 horas por semana.
  - Creo que es mejor tener un día largo que varios días cortos.

## Observaciones

Las pruebas resultaron exitosas, alcanzando todos los testers los objetivos que se les definieron en un tiempo que se consideraría razonable para un usuario que nunca antes había visto la interfaz ni estaba familiarizado con los modelos definidos y gran parte de la terminología utilizada. En promedio, los sujetos de prueba demoraron 42 minutos en completar todos los pasos, con los usuarios más jóvenes lográndolo en menor tiempo, con la excepción de un caso outlier.

Una observación interesante es que el tiempo demorado en el paso 2 tendía a ser inversamente proporcional al utilizado en el paso 3, ya que aquellos usuarios que exploraron el sitio web libremente durante el paso 2 acabaron familiarizándose con las diferentes vistas y secciones y por ende tenían una visión más general del funcionamiento de la aplicación a la hora de encarar las tareas del punto 3.



**Figura 64:** Tiempo total para completar los objetivos según edad

La mayoría de los pasos se completaron sin problemas, donde la mayor parte del tiempo transcurrido se invirtió en personalizar campos de ingreso de texto. Esto fue especialmente notable en los pasos 1 y 4 donde los usuarios ingresaban sus direcciones de email reales e intentaban crear una contraseña que cumpliera los requisitos a medida que estos se hacían conocidos mediante el mensaje de error.

Por otro lado, en los puntos 2 y 3, al ser objetivos amplios que no especifican una secuencia concreta de pasos, los usuarios se vieron obligados a explorar la aplicación, a veces siguiendo caminos incorrectos o entrando en un ciclo yendo de una vista a la otra.

Esto no fue inesperado, ya que el paso 2 buscaba analizar el comportamiento de los usuarios y qué tan fácilmente lograban comprender la relación entre materias y carreras, al igual que el concepto de correlativas dentro de estas últimas. No hay carrera sin cursos, y un curso no tiene correlativas sin carrera. Si bien existe un camino óptimo, que es crear las materias primero, luego la carrera, y finalmente definir las correlatividades, es posible y esperable que un usuario ponga el foco en cumplir todos los requerimientos de un curso ni bien lo crea, generando un ciclo repetitivo de añadir materias individualmente a la carrera y definir las correlativas en caso de ser necesario.

Si bien algunos sujetos captaron la idea y desarrollaron una estrategia acorde rápidamente, otros tardaron más tiempo en entender que los cursos son entidades independientes de la carrera, y que la misma no es solo un listado sino también un intermediario para definir relaciones entre asignaturas.

Un problema que se observó con regularidad fue el esperado hecho de que el usuario promedio no se molesta en leer detenidamente el texto frente a él excepto como último recurso. En todos los casos donde el sujeto intentó sin éxito recorrer la página hasta encontrar la forma de definir correlatividades, después del tercer o cuarto intento fallido decidió leer detenidamente los breves mensajes en pantalla que explicaban qué pasos debía seguir para cumplir ese objetivo pero no les había prestado atención anteriormente.

Si bien no es ideal que el usuario no logre encontrar lo que busca en su primer intento, se considera que dada la complejidad del caso de uso, el hecho de que hayan podido reflexionar sobre sus intentos fallidos y prestar atención a la información frente a ellos sin intervención del equipo es una buena señal.

## Feedback

En cuanto a la opinión de los usuarios, el consenso fue fuertemente positivo, aplaudiendo el concepto de la herramienta desde la perspectiva de un posible estudiante, siendo “Amigable” y “Simple” las dos palabras más comunes utilizadas por los sujetos de prueba para describir la usabilidad. En repetidas ocasiones, éstos remarcaron que si bien tuvieron algo de dificultad para orientarse durante el paso 2, después de superar ese obstáculo y entender dónde podían encontrar cada elemento, el navegar por la aplicación les resultó intuitivo.

Las únicas observaciones negativas fueron relacionadas al diseño sencillo y plano de la interfaz (estas dos viniendo de usuarios especialistas en ventas y marketing). Si bien son críticas válidas, no se consideró un punto de mejora urgente dado el alcance de la solución como un prototipo experimental de una herramienta de gestión de datos o de búsqueda rápida. En caso de que se intentara vender el producto a las universidades, indudablemente se recurriría a un especialista en diseño gráfico para hacer la interfaz más atractiva sin sacrificar su funcionalidad, además de considerar un rebranding para que el nombre del producto resulte más universal y fácil de pronunciar en distintos idiomas.

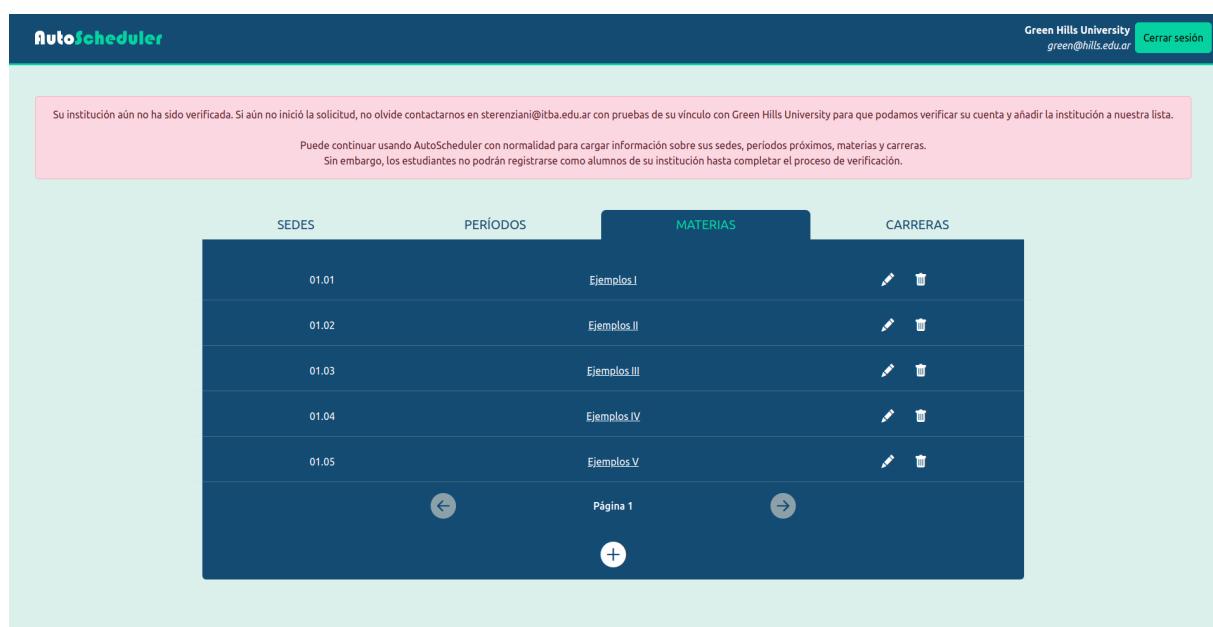
## Ajustes

Al observar errores cometidos por varios de los sujetos de prueba, se realizaron pequeñas modificaciones a la interfaz para atenuar estos errores y dificultar su ocurrencia a futuro.

### Barra de Búsqueda

Un error que cometió la mayor parte de los usuarios fue utilizar la barra de búsqueda disponible en la pestaña de Materias para crear una ingresando su nombre. Incluso después de haber encontrado la manera de crear un curso, varios de los sujetos de prueba repitieron este error al regresar a la pestaña.

En consecuencia, se decidió que la barra de búsqueda sólo debería mostrarse una vez que hubiese suficientes cursos para justificar su existencia. El equipo determinó que el mínimo requerido para mostrar la barra sea de 5. De esta manera, un nuevo usuario ya estaría acostumbrado a crear un curso haciendo click sobre el botón correspondiente para cuando vea la barra de búsqueda.



The screenshot shows the AutoScheduler application interface. At the top, there is a dark header bar with the 'AutoScheduler' logo on the left and 'Green Hills University green@hills.edu.ar' and a 'Cerrar sesión' button on the right. Below the header, a pink message box displays text about institution verification. The main content area has a light blue background and features four tabs: 'SEDES', 'PERÍODOS', 'MATERIAS' (which is highlighted in blue), and 'CARRERAS'. Under each tab, there is a table with two columns: 'SEDES' and 'MATERIAS'. The 'MATERIAS' column contains five rows labeled 'Ejemplos I' through 'Ejemplos V', each with edit and delete icons. At the bottom of the table, there are navigation arrows and a page number 'Página 1'. A large '+' button is located at the bottom center of the table area.

**Figura 65:** Vista de la pestaña de Materias sin barra de búsqueda y con botón (+) agrandado

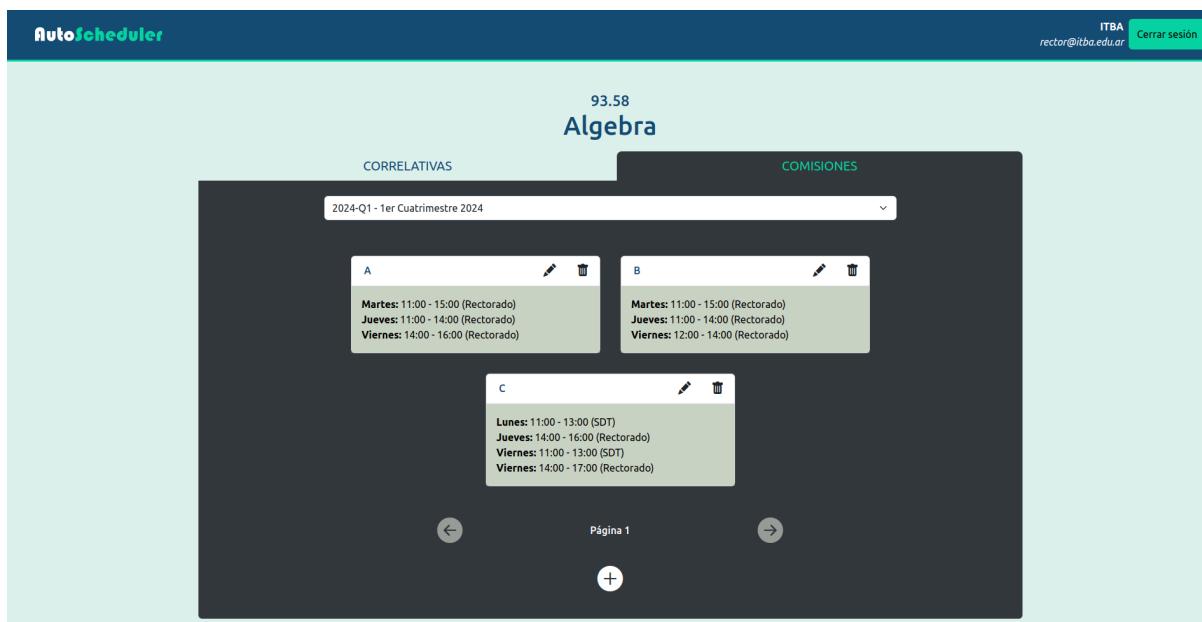
## Botón de Creación

En algunas pruebas, el botón (+) para crear o añadir elementos a una lista pasó desapercibido. Sea por la falta de familiaridad del usuario con el concepto de este tipo de botones o porque el mismo no llamaba lo suficiente la atención, esto no debería ocurrir.

Para atenuar este problema, se incrementó el tamaño de todos los botones de este tipo. Se espera que su tamaño capte la atención del usuario perdido con mayor facilidad, y deje en claro que tiene una importancia mayor que las flechas de paginación que aparecen cerca del mismo.

## Página del Curso

Muchos usuarios presentaron señales de confusión al guardar un curso y ser redirigidos a una página familiar pero que no reconocían. Para diferenciar claramente la página de una materia de la página Home, se cambió el color del componente principal por un tono negro, que se diferencie del azul de la página de inicio. Además, se incrementó el tamaño de los títulos para que el nombre de la asignatura no pase desapercibido, y las pestañas sean más fáciles de leer y encontrar.



**Figura 66:** Vista de la nueva página de curso

## Títulos de Formularios

A pesar de que para crear o editar una entidad (por ejemplo, una materia) el usuario debe acceder a una pestaña con el nombre del tipo de dicha entidad ("Materias") y una lista de entidades similares (la lista de materias), más de uno se perdía fácilmente y no comprendía por qué el formulario al que había accedido tenía campos diferentes de los que había completado para otro tipo de entidad (por ejemplo, sedes).

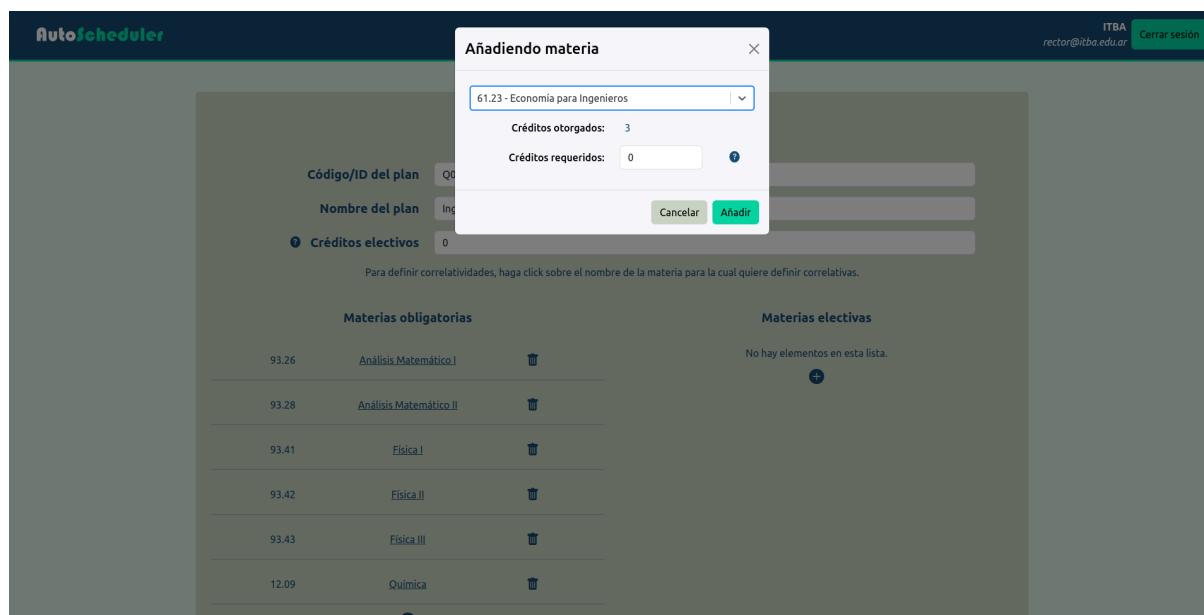
Como refuerzo, se incrementó el tamaño de los títulos de los formularios, esperando que el tipo de la entidad a crear o editar sea más legible para el usuario.

## Créditos

Sin duda el error más común durante las pruebas realizadas fue la diferenciación entre créditos otorgados por una materia y los requeridos para cursarla en una carrera. Si bien esto puede deberse a que la mayoría de los usuarios no estaban familiarizados con el concepto de créditos utilizado en algunas universidades, no es motivo para ignorar su problema.

Dado que el enunciado que se les dio a los sujetos de prueba definía los créditos otorgados por cada curso, dejando los créditos requeridos implícitos a menos que sean mayores a 0, muchos cometían el error de introducir el valor dos veces: Una vez al crear la materia (que sería lo correcto) y al cargarlo a la carrera (donde deberían especificarse los créditos requeridos, no los otorgados).

Para dejar en claro que estos no son intercambiables, se agregó al modal utilizado para añadir una materia a una carrera el dato de cuántos créditos otorga la asignatura seleccionada. De esta forma, se evita la confusión entre los dos campos.



**Figura 67:** Información de créditos al añadir una materia a una carrera

## Horarios Predeterminados

Algunos sujetos de prueba tuvieron dificultades al definir horarios de clase al crear comisiones. A pesar de que el enunciado aclaraba que se necesitaban dos de ellas, y que los desarrolladores presentes en las pruebas remarcaban cuándo el objetivo no había sido cumplido y por qué, muchos interpretaban que cada comisión podía tener una única franja horaria.

Para solucionar esto, se incrementó el tamaño del botón que agrega más franjas al formulario, y además se definieron dos franjas predeterminadas que estarán presentes al crear la comisión. De este modo, quedará claro que una comisión puede tener dos franjas horarias. El botón (+) debería ahora ser suficientemente grande para llamar su atención, y en caso de que solo necesite una franja horaria, debería notar el ícono de cesto de basura junto a cada franja.

En caso de que esto último no ocurra, las franjas contienen horarios consecutivos, por lo que una solución rebuscada sería dividir su horario deseado a la mitad, efectivamente definiendo un diminuto recreo entre ambas franjas.

The screenshot shows the 'Crear comisión' (Create Commission) interface. At the top, there are dropdown menus for 'Materia' (93.58 - Algebra) and 'Período' (2024-Q2 - 2do Cuatrimestre 2024). Below these, a field 'Nombre de comisión' contains 'Grupo X'. Under the heading 'Horarios', there are two sets of time inputs: 'Domingo 06:00 AM - 07:00 AM Virtual' and 'Domingo 07:00 AM - 08:00 AM Virtual'. A large green '+' button is positioned between the two time slots. At the bottom right of the form is a 'Guardar' (Save) button.

**Figura 68:** Doble franja horaria por defecto al crear una comisión

## Auto-Selección de Carrera

Dentro de la página de una materia, al ver el listado de correlativas definidas para una carrera se ofrece al usuario un botón “Editar” con el fin de darle un atajo a la página de edición donde puede, entre otras cosas, modificar la lista de correlativas para cualquier carrera.

Tras ver a un sujeto de prueba usar este método como el único acceso a la lista editable de correlativas, se decidió implementar una mejora a este loop. El botón originalmente redirigía al formulario de edición y pretendía que el usuario seleccione la carrera a editar. En la mayoría de los casos, esto implicaría realizar la selección dos veces, lo que resultaría frustrante.

La solución fue pasar la carrera que había sido seleccionada en la página del curso como parámetro de la URL para abrirla automáticamente al cargar el formulario y ahorrarle un paso extra al usuario.

### Asociación de Carreras al Crear Curso

Dada la complejidad de vincular materias y carreras (ya que la carrera necesita materias, y estas últimas pueden necesitar carreras bajo las cuales definir correlativas), fue esperable que los usuarios tuvieran problemas al cumplir esa meta.

El proceso de asociación se había ideado para realizarse desde el formulario de creación/edición de carrera, ya que se consideró al curso como más independiente de la carrera que de la forma inversa. Sin embargo, el proceso para vincularlos era tedioso, obligando a los usuarios a moverse de un formulario de edición a otro para definir las relaciones.

Tras discutir la manera más conveniente de simplificar este proceso, se llegó a la conclusión de que lo más razonable sería permitir que el usuario cree la carrera primero, asegurándole que podrá asociar materias más tarde. Luego, una vez que intente crear una, se le mostrará una lista con sus carreras creadas, y checkboxes para seleccionar si quiere asociar el nuevo curso con ellas.

The screenshot shows the AutoScheduler application interface. At the top, there is a navigation bar with the logo 'AutoScheduler', the name 'Instituto Tecnológico de Buenos Aires', the email 'rector@itba.edu.ar', and a 'Cerrar sesión' button. Below the navigation bar, there are four tabs: 'SEDES', 'PERÍODOS', 'CARRERAS', and 'MATERIAS'. The 'MATERIAS' tab is currently selected and highlighted in blue. Under the 'MATERIAS' tab, there is a search bar labeled 'Búsqueda' with a 'Buscar' button. Below the search bar, there is a table listing three courses:

Carrera	Créditos
Algebra	93.58 9 créditos
Algebra Lineal	93.18 6 créditos
Análisis de Coyuntura Económica	61.27 3 créditos

Each course row has two small icons on the right side: a pencil icon for editing and a trash can icon for deleting.

**Figura 69:** Reordenamiento de pestañas para incentivar que la carrera se cree primero

**Crear Materia**

Código/ID de la materia	01.01
Nombre de materia	Ejemplos I
Créditos otorgados	0

Nombre del plan	Parte del plan	Materia electiva	Créditos requeridos
BIO22 - Bioingeniería	<input type="checkbox"/>	<input type="checkbox"/>	0
S10 - Ingeniería Informática	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0

**Cancelar** **Guardar**

**Figura 70:** Nuevo componente para seleccionar carreras al crear una materia

Al guardar la nueva materia, se le notificará al usuario que los cambios han sido guardados, y se le presentará la herramienta para definir las correlativas para las carreras seleccionadas. En caso de que no quiera hacer esto, podrá volver al Home con el botón que anteriormente decía “Cancelar”.

**Editar Materia**

¡Su materia "01.01 - Ejemplos I" ha sido guardada!

Ahora puede definir las materias correlativas bajo las carreras a las cuales las agregó, o regresar al inicio.

Correlativas		
82.31	<a href="#">Estructura de Datos y Algoritmos</a>	
97.35	<a href="#">Arquitectura de Computadoras</a>	

**+**

**Volver al inicio** **Guardar**

**Figura 71:** Paso intermedio para definir correlativas inmediatamente después de crear una materia

Una vez guardados los nuevos cambios, se redireccionará al usuario a la página del curso, donde podrá crear las comisiones.

Fuera de este caso de uso donde se está creando el curso, el flujo del sitio se mantuvo igual. Al editar una materia, podrá editar las correlativas para las carreras de las que forma parte, pero no podrá editar a qué carreras pertenece. Esta fue una de las preocupaciones del equipo durante el diseño de esta funcionalidad, ya que no se quería confundir al usuario en el caso de arrepentirse y querer realizar modificaciones. Sin embargo, se llegó al acuerdo de que el ofrecer el atajo para vincular materias inicialmente sumaba demasiado valor como para no añadirlo, y que sería mucho más común el caso donde se quiere crear un curso nuevo y definir sus relaciones ágilmente que el caso donde un curso se dé de baja en un único plan de estudios.

### Mayor Información en Lista de Cursos

Una funcionalidad que no fue demandada por los usuarios, pero el equipo consideró que aportaría valor fue mostrar los créditos asignados de un curso debajo de su código, dentro de la lista desde la página de inicio, como puede observarse en la figura 69.

### Cambio de Flujo para Estudiantes

Hacia fines del desarrollo, se analizó nuevamente el flujo de un estudiante que accede a la aplicación, y posibles dificultades que podría encontrar. Si bien los usuarios de prueba no mostraron dificultades para encontrar la pestaña para registrar materias aprobadas, se consideró que el convertirla en la pestaña secundaria frente al formulario de búsqueda era impráctico, dado el contexto del caso de uso principal.

En el caso de un estudiante que acaba de registrarse, lo primero que debería hacer es registrar los cursos que tiene aprobados para actualizar su información en la base de datos. Asimismo, cuando un estudiante entra a buscar cronogramas para un próximo período, no debería olvidar actualizar esta lista antes de realizar una búsqueda.

Por ello se decidió redirigir a la pestaña de Materias Aprobadas al iniciar sesión o registrarse. De esta forma, el texto mostrado en lugar de la lista vacía le explicará al nuevo usuario qué hacer y cómo buscar cronogramas.

## Trabajo Futuro

Debido a la naturaleza del proyecto como un prototipo minimalista, existen múltiples funcionalidades y opciones para extensión que no fueron llevadas a cabo debido a que su valor agregado era secundario comparado con las funcionalidades que fueron implementadas. Algunas posibilidades de extensión que el equipo considera importantes para mejorar el producto se describen a continuación.

### Soporte para Carga de Datos Alternativa

Como se mencionó anteriormente, al inicio del desarrollo se consideró la posibilidad de ofrecer a las universidades simplificar la carga de datos mediante un archivo CSV, pero eventualmente se descartó la idea para priorizar un robusto sistema genérico que permita fácilmente realizar ajustes a los datos cargados si eventualmente se implementara tal funcionalidad.

Además de permitir la carga de archivos en formato CSV podrían ofrecerse, como un servicio extra, integraciones con sistemas internos de universidades. De esta forma, la información se mantendría actualizada sin necesidad de interacción humana, y los alumnos podrían seleccionar las materias a las cuales inscribirse desde la vista de cronogramas.

### Filtros Adicionales

Actualmente la búsqueda de cronogramas se limita a filtrar horarios indeseados, ajustar la función de puntaje según la existencia de días libres o presencia de materias importantes para la carrera en la combinación propuesta, y la definición de la cantidad ideal de horas semanales.

Si hubiese que agregar un filtro adicional, sin duda sería la habilidad de especificar asignaturas específicas a incluir o evitar en los cronogramas a encontrar. Esta es una de las ventajas que tienen sistemas como el Combinador de Horarios por sobre *AutoScheduler*, ya que a veces un estudiante puede tener una preferencia por una materia específica que preferiría cursar, pero desconoce o le es indistinto qué otras puede cursar en simultáneo.

Otro filtro que sumaría valor es la posibilidad de restar puntos si el día académico inicia, termina o contiene un cambio de sedes lejanas cerca de un horario pico, donde el tráfico y el transporte público estarían más congestionados de lo normal.

## Calificación de Dificultad

Como funcionalidad adicional, los estudiantes podrían evaluar materias según su exigencia o dificultad, permitiendo llevar un registro de la dificultad en períodos recientes para calcular un balance y no ofrecer a los estudiantes cronogramas demasiado fáciles ni demasiado exigentes a menos que lo soliciten explícitamente en el formulario de búsqueda.

## Refinamiento de la función de puntaje

El método de puntaje utilizado se basa en la asignación de un peso a distintas métricas del cronograma dentro de una fórmula. En el diseño actual, el peso de cada métrica es constante y se define basado en los atributos que se consideraron con mayor relevancia para el equipo de desarrollo. En caso de que el usuario considere una variable más importante que otra, podría contar con la opción de ajustar los pesos en una escala de “poco importante” a “muy importante”.

## Portabilidad

Si bien ésta no es una herramienta que se utilice fuera de los pocos momentos en el año donde los alumnos necesiten investigar sus opciones para un período próximo, por motivos de conveniencia podrían preferir la existencia de una aplicación para dispositivos Android o iOS que les notifique cuando se publique un nuevo período.

## Conclusiones

El producto desarrollado tenía la finalidad de proponer una solución a una problemática común entre estudiantes universitarios; encontrar las mejores combinaciones de materias y comisiones habilitadas, de una manera diferente a la que pueden ofrecer las alternativas existentes hasta el momento.

Los objetivos funcionales definidos al inicio del proyecto fueron alcanzados, y si bien existe lugar para mejora, la solución desarrollada ha recibido feedback positivo por parte de los sujetos de prueba. Teniendo esto en cuenta, sumado al hecho de que la obtención de materias habilitadas para el estudiante es automática, reduciendo al mínimo el ingreso de datos requerido por parte de este tipo de usuario, es posible afirmar que el producto desarrollado brinda una solución con ventajas por sobre sus competidores, destacando por su simplicidad y reducción de input a ser ingresado cada vez que se intenta buscar combinaciones.

El diseño de la arquitectura, además de ofrecer un funcionamiento fluido, permite no solo la escalabilidad de los servicios existentes, sino que también facilita la modificación modular de la misma en un futuro, en caso de considerarse necesario. Junto al foco en la separación de responsabilidades en el back-end, se garantiza una alta adaptabilidad a la hora de modificar los componentes del proyecto, según sea demandado, con un mínimo impacto sobre el resto de los componentes.

Con estas consideraciones, el equipo de desarrollo considera que los resultados obtenidos dados los recursos limitados resultan satisfactorios y se alinean con las expectativas planteadas al principio del proyecto. En caso de disponibilizar el prototipo para el uso general, podrían recolectarse con mayor facilidad datos que permitan evaluar necesidades de usuario incumplidas y puntos de mejora para desarrollo futuro.

## Manual de Uso

El código fuente del proyecto puede ser descargado y ejecutado localmente desde el repositorio en BitBucket (<https://bitbucket.org/itba/pf-2022-auto-scheduler-cuatrimestral>). Se recomienda seguir las instrucciones detalladas en el archivo README del mismo para configurar los contenedores correctamente.

Para replicar los experimentos realizados utilizando información del ITBA, se encuentran dentro del repositorio archivos para inicializar la base de datos Neo4J y un dump de la base utilizada para obtener las métricas.

## Referencias

- [1] Combinador de Horarios. <https://testlogin-97e9b.firebaseio.com>. Se consultó el 09 feb. 2024
- [2] Combinatrix. <https://github.com/soypat/combinatrix/tree/v1.1.0>. Se consultó el 09 feb. 2024
- [3] Coursicle. <https://www.coursicle.com>. Se consultó el 09 feb. 2024
- [4] Semester.ly. <https://semester.ly>. Se consultó 09 feb. 2024
- [5] Utah State University, College Scheduler.  
<https://www.usu.edu/registrar/registration/college-scheduler/step-by-step-guide>. Se consultó 09 feb. 2024
- [6] University of North Carolina, Using Your Schedule Planner.  
<https://registrar.unc.edu/guide/planning-for-registration/plan-your-class-schedule/schedule-planner-how-to-guide/using-your-schedule-planner>. Se consultó 09 feb. 2024
- [7] Bullet Education Scheduling and Timetabling.  
<https://www.bulletsolutions.com/es/best-bullet-education-scheduling>. Se consultó 29 oct 2023.
- [8] Cousens R., Patel S.H., & Sobh T.M. (2007). Course Scheduler: An Automated Schedule Generator. *Journal of Advances in Computer Science and Engineering*, 1, 25-46. Recuperado de [https://www.researchgate.net/publication/256456000\\_Course\\_Scheduler\\_An\\_Automated\\_Schedule\\_Generator](https://www.researchgate.net/publication/256456000_Course_Scheduler_An_Automated_Schedule_Generator). Se consultó 09 feb. 2024
- [9] Con AWS, el ITBA ofrece educación en línea y una mejor experiencia a sus estudiantes. AWS. Recuperado de <https://aws.amazon.com/es/solutions/case-studies/itba/>. Se consultó 09 feb. 2024
- [10] AWS, sitio oficial. <https://aws.amazon.com/>. Se consultó 09 feb. de 2024.
- [11] ECS, sitio oficial. <https://aws.amazon.com/route53/>. Se consultó 09 feb. de 2024.
- [12] S3, sitio oficial. <https://aws.amazon.com/s3/>. Se consultó 09 feb. 2024
- [13] DocumentDB, sitio oficial. <https://aws.amazon.com/documentdb/>. Se consultó 09 feb. 2024
- [14] Neptune, sitio oficial. <https://aws.amazon.com/neptune/>. Se consultó 09 feb. 2024
- [15] Route 53, sitio oficial. <https://aws.amazon.com/route53/>. Se consultó 09 feb. 2024
- [16] EC2, sitio oficial. <https://aws.amazon.com/ec2/>. Se consultó 09 feb. 2024
- [17] Internet Gateway. [https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Internet\\_Gateway.html](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html). Se consultó 09 feb. 2024
- [18] Mongo, sitio oficial. <https://www.mongodb.com/>. Se consultó 09 feb. 2024

[19] Neo4J, sitio oficial. <https://neo4j.com/>. Se consultó 09 feb. 2024

[20] NodeJS, sitio oficial. <https://nodejs.org/en>. Se consultó 09 feb. 2024

[21] JavaScript. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. Se consultó 09 feb. 2024

[22] Express, sitio oficial. <https://expressjs.com/>. Se consultó 09 feb. 2024

[23] TypeScript, sitio oficial. <https://www.typescriptlang.org/>. Se consultó 09 feb. 2024

[24] API REST. <https://restfulapi.net/>. Se consultó 09 feb. 2024

[25] Viescinski A. (2023). Tournament Selection in Genetic Algorithms. *Baeldung*. Recuperado de <https://www.baeldung.com/cs/ga-tournament-selection>. Se consultó 09 feb. 2024

[26] ReactJS, sitio oficial. <https://react.dev/>. Se consultó 09 feb. 2024

[27] Angular, sitio oficial. <https://angular.io/>. Se consultó 09 feb. 2024

[28] VueJS, sitio oficial. <https://vuejs.org/>. Se consultó 09 feb. 2024

[29] Bootstrap, sitio oficial. <https://getbootstrap.com/>. Se consultó 09 feb. 2024

[30] ReactBootstrap, sitio oficial. <https://react-bootstrap.github.io/>. Se consultó 09 feb. 2024

[31] React-i18Next, sitio oficial. <https://react.i18next.com/>. Se consultó 09 feb. 2024

[32] JSON Web Token, RFC 7519, 2015. Recuperado de <https://www.rfc-editor.org/rfc/rfc7519>. Se consultó 09 feb. 2024

[33] Bauhaus 93. <https://learn.microsoft.com/en-us/typography/font-list/bauhaus-93>. Se consultó 09 feb. 2024

[34] Fondo utilizado en Landing Page.

<https://wallup.net/abstract-3d-lines-white-bright-digital-art-artwork/>. Se consultó 09 feb. 2024

## Anexo

# DE TESTER	EDAD	OCCUPACIÓN	ALFABETISMO DIGITAL	COMENTARIOS SOBRE USABILIDAD
1	30	Secretaria	Alto	Simple
2	36	Programador	Alto	Intuitiva
3	37	Secretaria	Alto	Data entry complicado
4	38	Docente	Alto	Útil y amigable
5	45	Vendedora	Bajo	Buena
6	51	Directora Académica	Medio	Accesible, fluido y claro
7	54	Empresario	Medio	Simple
8	56	Gerente	Alto	Claro
9	60	Docente	Medio	Progresivamente amigable
10	60	Publicista, Docente	Medio	Relativamente amigable

*Tabla 1: Sujetos de prueba de la aplicación, ordenados por edad.*

# DE TESTER	PASO 1	PASO 2	PASO 3	PASO 4	PASO 5	PASO 6	PASO 7	TOTAL
1	90 s	763 s	832 s	8 s	43 s	56 s	131 s	32.1 mins
2	35 s	1078 s	459 s	10 s	76 s	59 s	104 s	30.4 mins
3	64 s	1464 s	1024 s	840 s	56 s	70 s	86 s	60.1 mins
4	77 s	840 s	478 s	10 s	41 s	62 s	102 s	26.8 mins
5	119 s	1339 s	738 s	23 s	140 s	200 s	300 s	47.7 mins
6	125 s	1854 s	344 s	8 s	86 s	160 s	145 s	45.4 mins
7	144 s	1356 s	510 s	19 s	91 s	105 s	118 s	39.1 mins
8	69 s	1425 s	402 s	18 s	60 s	69 s	66 s	35.2 mins
9	103 s	2600 s	189 s	13 s	112 s	117 s	106 s	54.0 mins
10	156 s	1940 s	515 s	36 s	145 s	79 s	173 s	50.7 mins

*Tabla 2: Tiempo que demoró cada sujeto en completar el objetivo asignado.*