UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Sistema de software para apoyar al estudiante en la toma de decisiones durante el proceso de inscripción de asignaturas

FABIÁN ALBERTO OLIVARES ARREDONDO

Profesor Guía: FEDERICO MEZA

Profesor Guía: RUTH GARRIDO

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
Julio, 2018

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2019

Dedicado a Silvia Arredondo,
quien siempre me ha
apoyado en cada paso.
¡Gracias por todo mamá!

AGRADECIMIENTOS

A Ruth Garrido y Federico Meza, por aceptarme como su estudiante memorista, apoyarme durante este proceso y orientar mi trabajo siempre apuntando a obtener el mejor resultado posible.

A Rodrigo Paredes, por aconsejarme y aclarar mis dudas durante el desarrollo de este proyecto.

A Daniel Moreno, por su ayuda en la realización de la encuesta de evaluación.

A mi madre, quien me ha dado todo lo necesario para llegar donde estoy, prestándome su apoyo incondicional, aconsejándome (incluso cuando no escuchaba) y dándome la seguridad de que siempre hay alguna forma de salir adelante y cumplir mis metas. Si hay alguien de quien tengo que estar agradecido toda la vida es de ti.

A mis abuelos, que siempre se han preocupado por mi bienestar, apoyando mis sueños, compartiendo mis victorias y ayudándome a ponerme de pie en las derrotas.

A mis hermanos Andrés, Pablo y Katherine, por su compañía, por aguantarme cuando soy insoportable y por ser parte de los momentos importantes de mi vida.

Al resto de mi familia, que siempre han estado ahí de una u otra forma, alentándome a superar cada nuevo desafío.

Y finalmente, a mis compañeros y amigos cercanos, especialmente a Erik Regla (best co[a|u]ch), Nicolás Pradenas y Diego Aldana (best team), Daniela Paredes, Yorch Sepúlveda y al Centro de Alumnos de Ingeniería Civil en Computación. Cada etapa en la vida tiene personas que dejan su huella y durante esta etapa ustedes han sido parte de esas personas para mí.

RESUMEN

La inscripción de asignaturas, denominadas *módulos* en la Universidad de Talca, es un proceso que se lleva a cabo en cada nuevo período académico en las universidades. Esta instancia es de gran relevancia, ya que los estudiantes deciden el futuro sobre sus estudios, guiando el camino que los llevará a obtener su título universitario.

Al momento de inscribir módulos, los estudiantes deben reconocer y evaluar muchos factores, donde se destacan las *líneas críticas*. Es un proceso complejo e impacta en el tiempo de permanencia de los estudiantes en la Universidad. Una mala inscripción puede tener como consecuencia un retraso en el egreso del estudiante, enfatizando la relevancia de tomar buenas decisiones a la hora de inscribir. Entonces, teniendo tanta importancia ¿cómo pueden los estudiantes estar seguros de que las decisiones tomadas son correctas? Son necesarias las herramientas adecuadas y actualmente los estudiantes no disponen de ellas.

Esta memoria busca ayudar en esta problemática, realizando el diseño y desarrollo de una herramienta que brinda apoyo a los estudiantes en la toma de decisiones durante el proceso de inscripción de módulos. Como parte de la metodología, se detallan los pasos para el diseño e implementación de un algoritmo capaz de sugerir planes de inscripción a los estudiantes. A su vez, teniendo *Scrum* como base de la metodología de desarrollo de software, se implementa una aplicación web para la interacción de los usuarios con dicho algoritmo.

La herramienta desarrollada trabaja con carreras de régimen semestral y fue probada en la carrera de Ingeniería Civil en Computación. Su finalidad es disminuir la carga que involucra decidir correctamente los módulos a inscribir y realizar ese proceso con suficiente información de respaldo.

Los resultados obtenidos en una encuesta aplicada a los potenciales usuarios indican que más del 90 % de los encuestados manifiestan que la herramienta creada es de gran utilidad y se apoyarían en ella durante la inscripción. Se destaca también la aprobación de la herramienta por parte de la profesora *Ruth Garrido*, Directora de Escuela de la carrera mencionada con anterioridad.

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

1. Introducción

Los estudiantes universitarios a lo largo de su estadía en una casa de estudios deben tomar decisiones que pueden aplazar de una u otra forma su egreso. El proceso de inscripción de módulos[1] no es la excepción, ya que es una instancia en la cual el estudiante decide los módulos que en su conjunto definen el avance con respecto al plan de formación.

Los estudiantes muchas veces reprueban módulos, lo que lleva consigo varias consecuencias. En la mayoría de estos casos el egreso es aplazado de manera irremediable, pero es posible reducir el impacto de reprobar al realizar una correcta inscripción de módulos en cada período.

Además, realizar una mala inscripción de módulos puede tener las mismas o peores consecuencias que reprobar.

En este capítulo el lector es introducido a la problemática y la motivación que hay detrás de este proyecto. A su vez se define el contexto del problema, los objetivos a lograr y los alcances del proyecto.

[1] En el contexto de la Universidad de Talca, un módulo comprende un conjunto de aprendizajes. Estas unidades de conocimiento reciben distintas denominaciones dependiendo de la institución en particular, siendo también usual el empleo de los términos curso o asignatura.

1.1. Descripción del contexto

Cada nuevo período en la Universidad de Talca lleva consigo un proceso de inscripción de módulos, tarea que cada estudiante debe completar en orden de avanzar en su respectivo plan de formación y finalmente convertirse en un egresado de esta institución.

A pesar de que es una tarea repetitiva y que cada estudiante debe conocer muy bien, no es tan simple como puede parecer. Una mala decisión a la hora de elegir entre inscribir uno u otro módulo tiene el potencial de aplazar el egreso en uno o más semestres.

Entonces, se puede decir que hay mucho más de lo que se observa a simple vista detrás de la inscripción en un nuevo período y muchas variables que entran en juego a la hora de decidir. En efecto, la inscripción de módulos no es una tarea sencilla después de todo.

Bajo este contexto, este proyecto busca generar una herramienta que ayude a los estudiantes a definir su inscripción de manera eficiente. Se hace notar que el alcance se acota a la carrera de *Ingeniería Civil en Computación* (ICC) de la *Universidad de Talca*.

1.2. Definición del problema

Durante el proceso de inscripción de módulos, los estudiantes deciden el conjunto de ramos que van a cursar en el período, teniendo en cuenta una serie de aspectos, entre ellos:

- Si corresponde que se dicte el módulo en dicho período.
- Prerrequisitos de cada módulo.
- Horarios de clases.

- Créditos y tiempo de dedicación al módulo.
- Dificultad y experiencias previas de otros estudiantes con el módulo.
- Docente a cargo del módulo.

Los estudiantes que reprueban módulos comienzan a tener mayores complicaciones para tomar este tipo de decisiones, debido a que deben cursar a la vez módulos que pertenecen a distintos semestres del plan de formación. Esto se traduce en que hacer una proyección a futuro sobre cuales módulos inscribir cada semestre, hasta finalizar los estudios, es aún difícil, ya que la cantidad de factores a tomar en consideración es grande.

La elección de módulos en la inscripción de cada semestre determina los módulos que podrán ser inscritos en los semestres posteriores, dados los prerrequisitos de cada uno. Adicionalmente, dada la restricción de cantidad máxima de créditos a inscribir por semestre, en la mayoría de los casos un estudiante no puede inscribir todos los módulos habilitados en el semestre. Entonces, por cada semestre se tiene una gran cantidad de conjuntos de módulos que pueden ser inscritos, cuya elección influirá en las inscripciones futuras. En base a lo anterior, para un estudiante que desea tener una selección eficiente de módulos a inscribir cada semestre, esto es, egresar lo antes posible, la combinatoria que involucra llegar a ese resultado es compleja y por sus propios medios, sin ayuda de herramientas, difícilmente podrá visualizar todas las opciones que tiene a la hora de realizar su inscripción. Es aquí donde se genera el problema, porque no existen herramientas para facilitar esta tarea para el caso específico de la Universidad de Talca.

En el caso particular de ICC, los estudiantes tienden a acudir a la Escuela buscando consejo sobre las opciones de inscripción. Se justifica entonces la necesidad de tener una herramienta que asista a los estudiantes en la toma de decisiones en cuanto a la inscripción de módulos. De esta manera es posible obtener un proceso de inscripción más expedito y acertado, tanto para la Escuela como para los mismos

estudiantes. Además, como efecto a largo plazo se puede obtener una mejora en el tiempo en que los estudiantes logran terminar la carrera, ya que es posible evitar errores en la inscripción producto del desconocimiento o un mal plan de inscripción de módulos.

1.3. Objetivos

Los objetivos en este proyecto son clasificados en objetivos generales y objetivos específicos. Estos se detallan a continuación.

1.3.1. Objetivo general

- Desarrollar un sistema de software para dar apoyo en la toma de decisiones durante el proceso de inscripción de módulos.

1.3.2. Objetivos específicos

- Diseñar e implementar un algoritmo para realizar simulaciones de la inscripción de módulos para un estudiante, dando a conocer las posibles opciones de inscripción.
- Diseñar e implementar una interfaz gráfica que permita configurar los parámetros, ejecutar y mostrar los resultados obtenidos del algoritmo.
- Implementar una herramienta para cargar información de los módulos de una malla y de los estudiantes.
- Obtener la aprobación del sistema implementado y retroalimentación por parte de los interesados.

1.4. Alcances

- La plataforma se desarrolla para el uso de los estudiantes de ICC y de la Escuela de la misma carrera.

- El algoritmo a diseñar no trabaja con información relacionada a la asignación de horarios de los módulos.

1.5. Descripción de contenidos

Los resto del contenido de este documento consta de:

- Capítulo 2, Antecedentes: Se abordan los antecedentes necesarios para comprender este proyecto de titulación. Se definen conceptos como plan de formación, malla, módulo, prerrequisito y también se explicita la problemática que se plantea resolver con este proyecto.
- Capítulo 3, Metodología: Se definen las metodologías utilizadas para resolver el problema y llevar a cabo el desarrollo de este proyecto, dando a conocer la forma de organización del trabajo realizado.
- Capítulo 4, Desarrollo del proyecto: Se detalla el trabajo realizado. Particularmente se enfoca en los aspectos de diseño e implementación de las distintas piezas que componen la solución.
- Capítulo 5, Evaluación de la solución y resultados: Se da a conocer los resultados de la evaluación realizada, destacando los puntos más relevantes.
- Capítulo 6, Conclusiones: Se presentan las conclusiones del proyecto en relación con el trabajo realizado y los resultados obtenidos. También se dan a conocer los trabajos futuros a desarrollar.

2. Antecedentes

Este capítulo detalla conceptos como plan de formación, malla, módulo, prerrequisito y líneas críticas, los cuales son relevantes para entender el impacto que tiene el proceso de inscripción y las consecuencias de no planificarlo adecuadamente. Se exponen también los problemas que existen actualmente, junto a los trabajos relacionados desarrollados en otras instituciones al respecto. Finalmente se da a conocer la propuesta de este proyecto para dar solución a los problemas mencionados.

2.1. Conceptos

Esta sección tiene la finalidad de informar al lector sobre los diferentes conceptos que tienen relevancia para el proyecto. Durante el resto del documento se hace amplio uso de éstos por lo que se destaca la importancia de comprenderlos.

2.1.1. Plan de formación

Un plan de formación es la estructura curricular en que se enmarca una carrera [9]. Cada plan de formación detalla un perfil de egreso del estudiante, el cual define las competencias y habilidades esperadas de un estudiante egresado de la carrera que estudió en ese plan específico.

Cada carrera establece el recorrido a efectuar por el estudiante para desarrollar las competencias previamente definidas en el plan de formación. A esto se le conoce

como currículo o simplemente *malla*, la cual comprende el conjunto total de módulos que desarrollan las competencias requeridas según el perfil de egreso de la carrera y plan en particular.

Un plan de formación debe ser aprobado en una resolución de la Universidad de Talca antes de su implementación.

2.1.2. Malla

Una malla corresponde a un conjunto de asignaturas o módulos en orden secuencial, cada uno de ellos valorados en créditos SCT-Chile (**Sección 2.1.5**).

El aprobar completamente los módulos de una malla de un plan de formación le permite al estudiante optar a un grado o título profesional [10].

En la **Figura 2.1** se aprecia parte de la malla del plan 16 de la carrera de ICC. Cada rectángulo representa un módulo (**Sección 2.1.3**), junto a la cantidad de créditos que le corresponde en su esquina superior derecha. Las líneas entre módulos representan dependencias de prerrequisitos (**Sección 2.1.4**). También cabe notar que la malla mencionada está dividida semestralmente, por lo que cada uno de sus módulos es semestral. En caso de requerirse un módulo anual, este se vería representado como un rectángulo que abarca dos semestres. La malla completa del plan 16 de ICC se encuentra en el **Anexo B**.

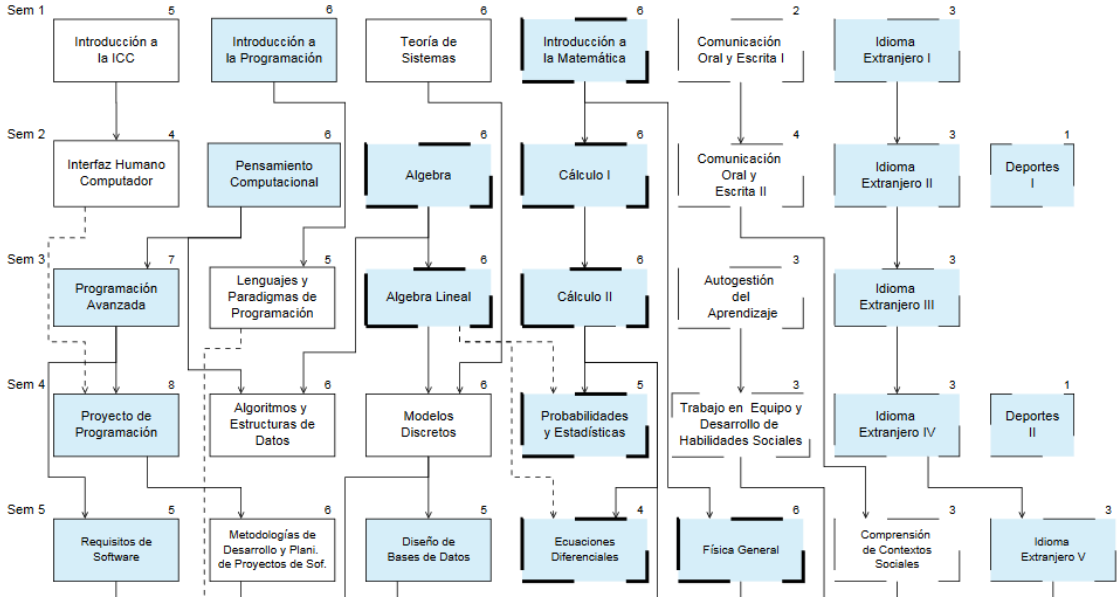


Figura 2.1: Representación de los primeros cinco semestres de la malla del plan 16 de ICC

2.1.3. Módulo

Un módulo es una unidad de trabajo-aprendizaje referida a una competencia o un conjunto de capacidades declaradas en el perfil de egreso del plan de formación [9].

La planificación de un módulo, esto es el contenido de las clases, competencias, créditos SCT-Chile, actividades y evaluaciones, es expresada en su *syllabus* y en un plan de clases [9].

El *syllabus* es un documento institucional que establece el proceso de aprendizaje del alumno a partir de competencias y/o capacidades, considerando el tiempo de

trabajo, nivel de logros, desempeños y productos esperados del estudiante [10].

En la Universidad de Talca existen módulos anuales (dictados en el período de un año) y módulos semestrales (dictados en el período de un semestre). En el caso de los módulos semestrales, éstos son dictados según el semestre al que pertenecen en la malla, esto es, si el módulo pertenece al primer semestre, cada primer semestre del año ese módulo es impartido. Así también ocurre en el caso de ser de segundo semestre. Cabe mencionar que, para el caso particular de ICC, todos los módulos de cada una de sus mallas son semestrales.

Los módulos que se dictan en un período tienen un horario definido, así como también un lugar donde se imparten (salas de clase, laboratorio, entre otros). Esta asignación se define al inicio de cada semestre, y varía entre un período y otro, al igual que los módulos impartidos.

2.1.4. Prerrequisitos de módulo

Dentro de una malla, existen ciertos prerrequisitos para poder optar a la inscripción de un módulo, los que pueden ser de tres tipos: Haber cursado un módulo específico con anterioridad, haber aprobado antes un módulo específico y haber aprobado una cantidad definida de créditos.

Prerrequisito de módulo cursado: Define que, para efectos de inscripción de un módulo, es necesario previamente haber cursado otro módulo de la malla. Un módulo cursado es aquel que fue inscrito en un período anterior y puede o no haber sido aprobado por el estudiante.

Prerrequisito de módulo aprobado: Define que, para efectos de inscripción de un módulo, es necesario previamente haber aprobado otro módulo de la malla. A diferencia del prerrequisito de módulo cursado, en este caso sí es necesario haber aprobado el módulo.

Prerrequisito de créditos aprobados: Define que, para efectos de inscripción de un módulo es necesario tener una cantidad específica de créditos aprobados. Los créditos aprobados se calculan sumando la cantidad de créditos de cada módulo aprobado por el estudiante en el plan de formación al que pertenece actualmente.

2.1.4.1. Línea crítica

Una línea crítica corresponde a una secuencia de módulos que tienen dependencia de prerrequisitos. Este concepto es muy relevante para el proyecto, ya que en sí determina cuánto se retrasa el egreso de un estudiante al reprobar un determinado módulo, según la línea crítica a la que pertenece el módulo.

Por lo general estas líneas abarcan módulos que inician en los primeros semestres de la carrera y otros que terminan en los últimos semestres. Debido a lo anterior al reprobar uno de estos módulos es casi seguro que el egreso del estudiante se verá retrasado a lo menos en un semestre y en algunos casos se puede extender a un año.

En la línea crítica de la **Figura 2.2** se puede observar que ésta abarca varios semestres. Ya que cada módulo depende de haber aprobado el anterior (por prerrequisitos), al reprobar uno de estos inmediatamente se aplaza el resto en un semestre. Un caso especial de esta línea es el módulo *Metodologías de Desarrollo y Planificación de Proyectos de Software*, el cual sólo se imparte los primeros semestres de cada año. En caso de reprobar el módulo antes mencionado o uno de sus prerrequisitos (*Pensamiento Computacional*, *Programación Avanzada* o *Proyecto de Programación*), la línea completa se retrasa en un año. Debido a lo anterior, el último módulo de la línea podría inscribirse el semestre 11, pero tendría que aplazar otro módulo de dicho semestre, ya que sobrepasaría la cantidad máxima de créditos a inscribir, retrasando el egreso en un semestre.

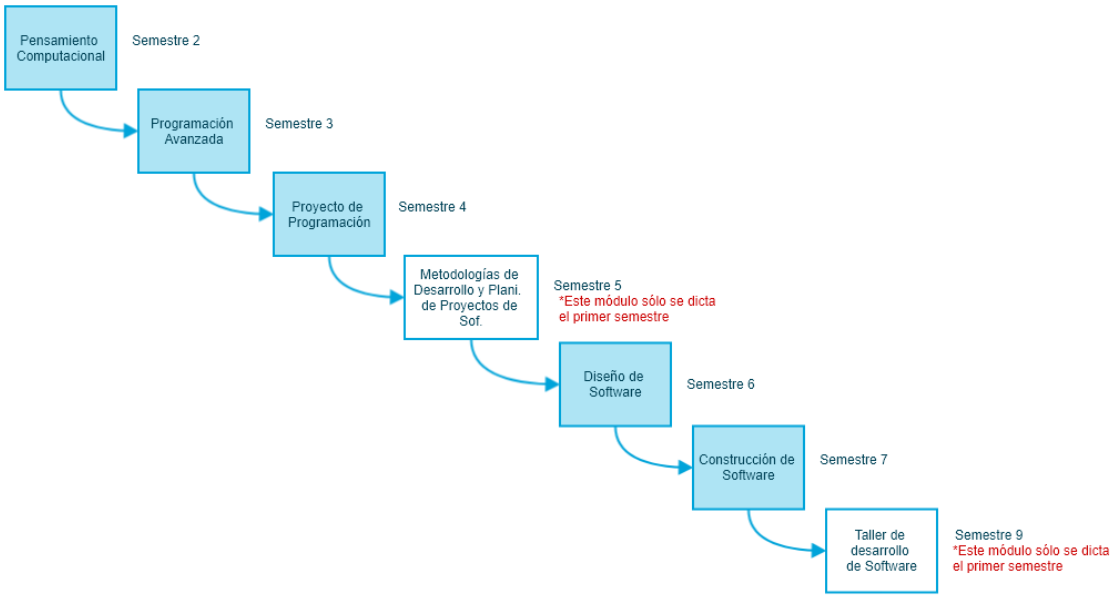


Figura 2.2: Ejemplo de línea crítica de la malla del plan 16 de ICC

2.1.5. Sistema de créditos: SCT-Chile

El SCT-Chile es un sistema de créditos que representan la carga de trabajo que demanda una actividad curricular al estudiante de manera que logre los resultados de aprendizaje esperados [8]. Este sistema es equivalente al Sistema Europeo de Transferencia y Acumulación de Créditos (ECTS) [3].

Como acuerdo del Consejo de Rectores, el trabajo correspondiente al período de un año equivale a 60 créditos SCT-Chile, estimados entre 1440 y 1900 horas de trabajo efectivo. A su vez, un semestre equivale a 30 créditos y un trimestre a 20. En la **Tabla 2.1** se puede apreciar la relación en cantidad de créditos con respecto a la

duración de una carrera. Cabe mencionar que los últimos planes para las carreras de Ingeniería en la Universidad de Talca (incluyendo ICC) contemplan una duración de 11 semestres, es decir, 330 créditos aproximadamente según la tabla.

Duración de carrera (en años)	**Total de créditos (valor aproximado)**
4	240
5	300
5,5	330
6	360
7	420

Tabla 2.1: Relación entre duración de una carrera y la cantidad de créditos SCT-Chile que contemplan

Cada módulo de una malla tiene asignado una cantidad específica de créditos. En caso de que el estudiante apruebe el módulo en cuestión, estos pasan a ser créditos aprobados. Los créditos aprobados tienen relevancia en la inscripción, como se detalló en la **Sección 2.1.4**.

2.2. El proceso de inscripción

Antes del inicio de cada semestre, los estudiantes de la Universidad de Talca realizan un proceso de inscripción de módulos. Este proceso se divide en dos partes: la preinscripción automática de módulos y la inscripción del estudiante. El diagrama de la **Figura 2.3** detalla ambas partes del proceso de inscripción. Cabe señalar que el sistema de inscripción se encarga de que el estudiante sólo pueda inscribir módulos que se dicten el semestre correspondiente.

2.2.1. Preinscripción automática

En esta parte del proceso, los módulos que según el plan de formación del estudiante son del menor nivel (semestre más temprano en la malla) y no han sido aprobados son inscritos de forma automática. Este proceso sólo ocurre cuando el

estudiante está atrasado respecto a su malla, es decir, ha reprobado algún módulo.

Un estudiante no puede desinscribir un módulo inscrito en esta etapa del proceso, ya que su inscripción es de carácter obligatorio. No obstante, un estudiante sí puede hacer uso del artículo 21 del reglamento de régimen de estudios de la Universidad en las fechas establecidas para dejar sin efecto la inscripción de uno de estos módulos [10], siempre que no haya eliminado dicho módulo de esta manera con anterioridad.

2.2.2. Inscripción del estudiante

Posterior a la etapa de preinscripción, el estudiante puede inscribir módulos a elección entre los habilitados del período hasta completar un máximo de créditos, contando también los créditos de los módulos inscritos de forma automática. El máximo de créditos se define a partir de la cantidad total de créditos del semestre de menor nivel no aprobado del estudiante [10]. Cabe mencionar que es posible inscribir por sobre el máximo de créditos teniendo autorización de la Escuela de la carrera.

Para que un módulo esté habilitado para un estudiante, éste debe cumplir con los prerrequisitos del módulo y además el curso debe tener vacantes disponibles.

Existe información relevante que un estudiante puede necesitar para tomar la decisión de inscribir determinado módulo. A continuación, se describen factores relevantes que un estudiante toma en cuenta durante el proceso de inscripción:

- **Horario del módulo:** Existe la posibilidad de topes de horarios entre módulos inscritos, por lo que conocer el horarios es relevante para evitar este tipo de situaciones. El sistema de inscripción de la Universidad alerta sobre esta situación y no permite inscribir en caso de darse.
- **Docente a cargo del módulo:** Muchas veces los estudiantes tienen preferencia de un docente por sobre otro en algunos módulos, por lo que en lo posible inscribirán con el docente de su preferencia.

- **Carga académica:** Conocer los créditos y horas de dedicación a determinado módulo es importante para el alumno. Los estudiantes evitan muchas veces inscribir en un mismo semestre varios módulos con gran carga académica, ya que conlleva mayor riesgo de reprobación.

- **Dificultad del módulo:** Si el módulo tiene una alta tasa de reprobación o si ya ha sido reprobado con anterioridad por el estudiante puede ser un indicio de que no es conveniente inscribirlo si los módulos inscritos en el semestre ya tienen una gran carga académica.

- **Línea crítica:** Este es el más importante de los factores y a su vez es uno de los que más pasa desapercibido por los estudiantes. Como ya fue mencionado en la **Sección 2.1.4.1**, reprobar un módulo en una línea crítica puede llevar consigo un gran retraso en el egreso del estudiante. Además, puede que, si el estudiante no está consciente de este factor, prefiera inscribir un módulo menos relevante, en cuanto a dependencias de prerrequisitos, por sobre uno que esté en una línea crítica y finalmente se retrase de igual manera.

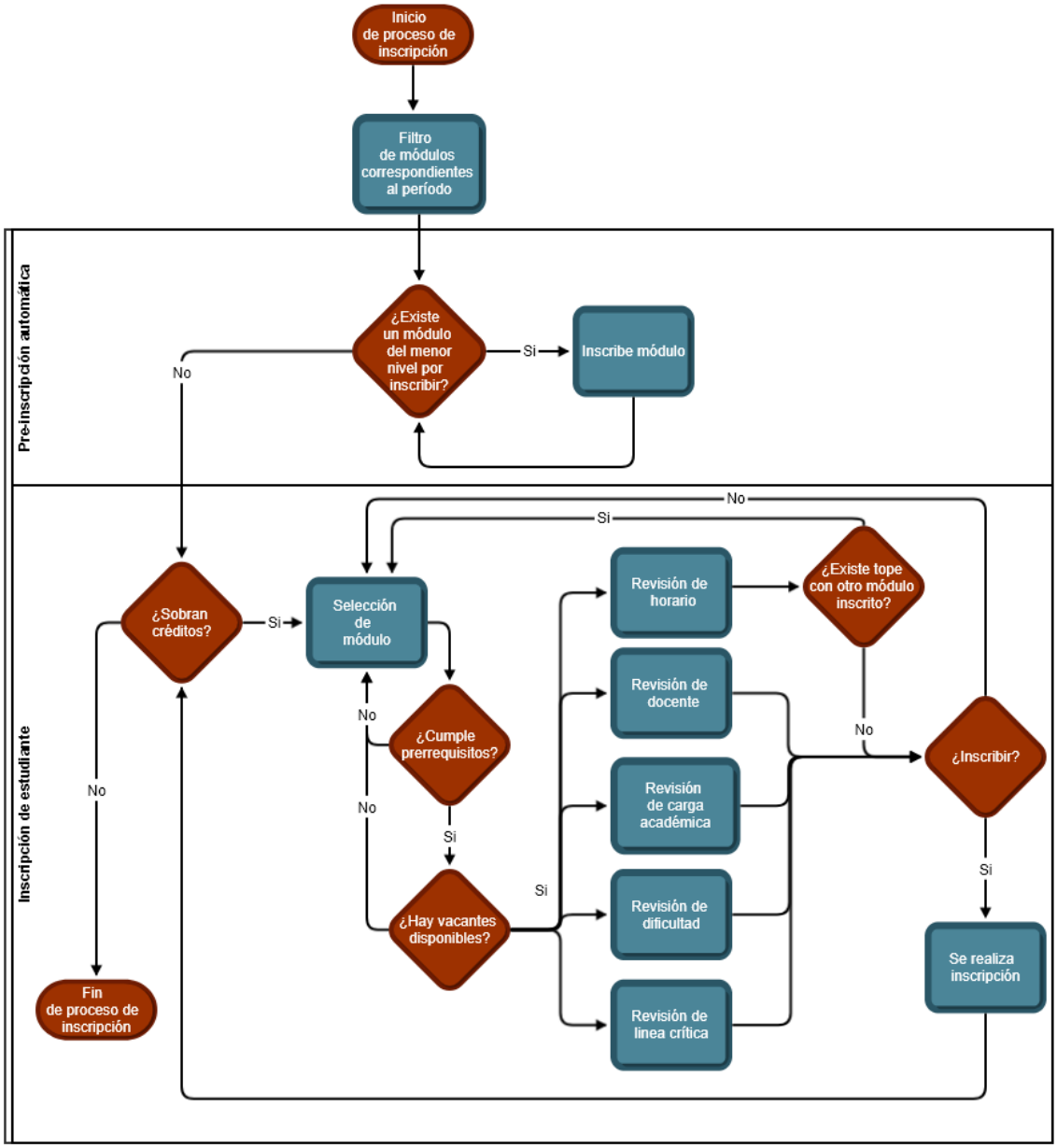


Figura 2.3: Diagrama que detalla el proceso de inscripción de módulos de un estudiante

2.3. Problemática

Es sabido que un retraso en el egreso, por mínimo que sea, afecta de muchas maneras a los estudiantes y su entorno. Esto es aún más importante si se tiene en cuenta la cantidad de estudiantes que reprueban módulos cada semestre.

Si bien los estudiantes que nunca han reprobado un módulo sólo inscriben los módulos del semestre que les corresponde, los que han reprobado tienen muchas posibles combinaciones de módulos a inscribir cada semestre. Debido a la gran cantidad de factores a tomar en consideración, la decisión se hace compleja.

Cómo se detalló en secciones anteriores (**2.1.4.1 y 2.2.2**) el factor de las líneas críticas es uno de los más difíciles de visualizar para los estudiantes. Por otro lado, los estudiantes no cuentan con herramientas para tomar decisiones teniendo conciencia de la gran cantidad de posibilidades que existen a la hora de inscribir. Como se mencionó anteriormente, la cantidad de opciones de inscripción que tiene un estudiante durante los semestres que restan de sus estudios es alta y la tarea de determinar planes de inscripción eficientes implica probar una gran cantidad de combinaciones. Es necesario simular para cada semestre las distintas opciones de inscripción, analizando prerrequisitos, *líneas críticas* y créditos de los módulos, entre otros factores. Además, las decisiones de inscripción en un semestre determinado influyen en las inscripciones de los semestres siguientes. Entonces, una persona por si misma y sin herramientas, difícilmente puede obtener los mejores resultados al realizar dicha tarea. Es necesario entonces solventar esa falta de herramientas, siendo este déficit el problema que busca solucionar este proyecto. Se tiene como uno de los focos principales evitar los problemas relacionados a las *líneas críticas* de las mallas.

2.3.1. Trabajos relacionados

Lamentablemente no existe mucha información al respecto del problema específico a resolver. Una de las principales razones para esto corresponde a las diferencias

existentes entre el sistema educativo chileno y los sistemas educativos en el resto del mundo. En otras universidades del mundo los estudiantes pueden elegir un conjunto de módulos diferentes unos de otros y sin embargo egresar de la misma carrera debido al sistema curricular flexible que poseen. Las carreras de universidades chilenas poseen un currículum rígido, esto quiere decir que el camino para obtener el título está muy delimitado. En el caso de la Universidad de Talca esto se aprecia claramente en las mallas de los planes de formación. Cada malla detalla, de manera precisa y sin mucho espacio para elegir, los módulos que el estudiante debe aprobar para llegar a ser un egresado de la carrera.

Dentro de los trabajos recopilados se da cuenta de dos que tienen una mayor relación con la problemática. El primero corresponde a una herramienta llamada *STOPS*, la cual permite a los estudiantes universitarios crear planes de estudios personales. La herramienta organiza a su vez los recursos necesarios para mantener y desarrollar los contenidos de los cursos y la estructura curricular para dichos planes [6]. El sistema fue utilizado en 2013 en la renovación del programa de licenciaturas de la *Escuela de Ingeniería* en la *Universidad Aalto*, en Finlandia.

El otro trabajo relacionado [7] busca, por medio de técnicas de Machine Learning, mejorar la eficiencia de los planes de estudio personales de los estudiantes en la Universidad *Sakon Nakhon Rajabhat* en Tailandia. En ese trabajo se generó un modelo de predicción de CGPA[1] (Cumulative Grade Point Average) por medio de redes neuronales, el cual tuvo un alto rendimiento en la predicción. Los resultados experimentales demuestran una mejora en los logros de aprendizaje de los estudiantes, llegando gran parte de estos a su CGPA ideal antes de la graduación.

Ambos trabajos están enfocados a planes curriculares flexibles y por lo tanto no es posible su aplicación directa al problema que aborda este proyecto.

[1] **Cumulative Grade Point Average** es una medida de rendimiento académico obtenida de sumar las ponderaciones de todas las notas obtenidas por el estudiante con respecto a la cantidad de créditos respectivos de los módulos.

2.3.2. Propuesta de solución

La solución propuesta por este proyecto consta de:

- El desarrollo de un algoritmo para determinar planes de inscripción para los estudiantes. Lo anterior tiene la finalidad de que éstos conozcan las rutas de inscripción que aseguran un egreso en el menor tiempo posible.
- Un sistema con el cual los estudiantes puedan hacer uso del algoritmo y probar distintos escenarios. De esta forma podrán realizar su inscripción al inicio de cada semestre conociendo las distintas posibilidades que tienen y buscando egresar lo antes posible.
- Es importante también que la Escuela conozca los planes de inscripción de los estudiantes, ya que de esta manera se puede preparar la oferta de módulos de cada semestre con más información. Entonces, como parte del sistema a implementar, se pretende la creación de un módulo para visualizar estadísticas relacionadas a los planes de inscripción de los estudiantes.

Como fue mencionado en los alcances (**Sección 1.4**), la asignación de horarios así como la distribución de las salas de clase es un factor que se ha dejado fuera de la solución del proyecto. La razón radica en que estos factores agregan una dificultad tal al problema, que los recursos necesarios para generar una solución que los incluya sobrepasaría al de un proyecto de titulación.

3. Metodología

Este capítulo detalla la metodología utilizada para elaborar la solución al problema en cada etapa del proyecto. Adicionalmente se da a conocer la planificación y organización del trabajo, trazando el camino para cumplir los objetivos propuestos (**Sección 1.3**).

En primera instancia se define la secuencia de pasos para diseñar un algoritmo que, teniendo el detalle de la malla y la situación de un estudiante en particular, determina planes de inscripción válidos para ese estudiante. Posteriormente se explica SCRUM y cómo se aplica al desarrollo del sistema, el cual actúa de intermediario entre los usuarios, el algoritmo y los datos obtenidos de éste. Se detalla también la forma de evaluación del proyecto en base a las pruebas a realizar.

3.1. Exploración y diseño del algoritmo de solución

Entre los objetivos planteados está el de diseñar e implementar un algoritmo capaz de generar inscripciones válidas para los estudiantes y conocer el tiempo estimado que tienen para egresar. Inicialmente no se tenía una idea clara de cómo lograr esto, razón por la cual se opta por seguir los siguientes pasos:

1. Investigar trabajos realizados con problemas similares y sus soluciones.
2. Definir objetivos y restricciones del algoritmo.

3. Diseñar, implementar y probar un algoritmo de fuerza bruta (búsqueda completa de soluciones).

4. Definir formas de mejorar el algoritmo.

5. Implementar mejoras al algoritmo y realizar pruebas nuevamente.

6. Si el paso anterior no arroja resultados correctos o no trabaja en rangos de tiempo razonables, volver al paso 4.

Los algoritmos implementados inicialmente buscan dar soluciones de manera determinista. Se tiene la posibilidad de, en caso de no cumplir con entregar soluciones válidas o no trabajar en tiempos razonables, buscar soluciones de forma no determinista utilizando heurísticas o *machine learning*.

3.2. Metodología de desarrollo de Software

Con la finalidad de llevar el desarrollo del sistema a implementar de la mejor forma y teniendo flexibilidad suficiente para afrontar cambios en las funcionalidades requeridas se escoge *Scrum* [13] como la metodología base en el desarrollo del software. La decisión de la metodología toma en cuenta la experiencia del autor, el cual la ha utilizado con anterioridad.

3.2.1. Scrum

Scrum es un marco de trabajo para el desarrollo, entrega y mantenimiento de productos. Es utilizado para resolver problemas en que es necesario adaptarse, manteniendo a su vez la productividad y entregando productos con el mayor valor posible [14].

Scrum emplea un enfoque iterativo e incremental. Esto quiere decir que en cada iteración se entrega un producto funcional con nuevo valor agregado con respecto a la iteración anterior.

A continuación, se dan a conocer los conceptos más relevantes de la metodología y cómo se adapta a las condiciones de desarrollo del presente proyecto.

3.2.1.1. Backlog

El *Backlog* es una lista ordenada de todo lo que va a necesitar el producto. Cada característica, funcionalidad, requisito, mejora y reparaciones que serán parte del producto en futuras iteraciones debe estar en el *Backlog*. Este es manejado únicamente por el *Product Owner* (**Sección 3.2.1.2**).

Existen dos tipos de *Backlog* en Scrum: *Product Backlog* (*backlog* del producto) y *Sprint Backlog* (*backlog* del *sprint*).

Product Backlog: está constituido por todos los elementos que han de ser implementados en el producto.

Sprint Backlog: está constituido por todas los elementos que serán implementados durante un sprint en particular.

El *Product Backlog* y detalle de los *sprint* del proyecto pueden ser revisados en la **Sección 3.2.2**.

Los elementos en el *Backlog* pueden ser escritos en forma de historias de usuario. Una **historia de usuario** corresponde a una definición de alto nivel de una funcionalidad requerida, es decir, descrita con lenguaje natural. Además, debe contar con la información suficiente de forma que el desarrollador pueda hacer una estimación razonable del esfuerzo necesario para implementarla. Además, con la finalidad de poder medir los avances realizados y tomar decisiones, cada historia de usuario tiene asignada una cantidad de puntos, llamados *puntos de historias de usuario*.

Los **puntos de historias de usuario** son una medida del esfuerzo necesario para implementar una historia y cada equipo de Scrum tiene su propia forma de eva-

luar y asignar estos puntos. En el caso de este proyecto los puntos de cada historia fueron asignados en las planificaciones de *sprint* respectivas según su complejidad, esto es, se estima qué tan compleja es una historia de usuario con respecto a las demás historias del *Sprint Backlog*.

3.2.1.2. Roles de Scrum

Un equipo de Scrum está compuesto por un *Product Owner*, un *Scrum Master* y el *Equipo de Desarrollo*. Estos roles son detallados a continuación:

- **Product Owner:** es el responsable de maximizar el valor del producto en cada iteración y es la persona a cargo de manejar el *Product Backlog*. Entre las tareas que realiza están: expresar claramente cada elemento en el *Backlog* para la comprensión del equipo, priorizar las historias de usuario en el *Backlog* y definir las historias a cumplir en cada *sprint*. Además, es quien se relaciona con los interesados con la finalidad de guiar el producto por el camino deseado y generar la mayor cantidad de valor posible.
- **Scrum Master:** es la persona que vela por el cumplimiento de Scrum en el equipo y se encarga de dar soporte, ayudando a los demás a entender la teoría, prácticas, reglas y valores de la metodología. Tiene además un rol facilitador, removiendo impedimentos que afecten la productividad del equipo.
- **Equipo de desarrollo:** corresponde a todas aquellas personas que hacen el trabajo de entregar un nuevo incremento del producto en cada *sprint*. Las principales características de los miembros del equipo es su capacidad de auto-organización y el tener en su conjunto todas las habilidades necesarias para crear un nuevo incremento del producto.

Dada las características de este proyecto, los diversos roles de Scrum son cubiertos por el autor, encargándose de:

- Mantener comunicación con los interesados.

- Definir las historias de usuario a completar en cada *sprint*.
- Implementar las funcionalidades necesarias para satisfacer las historias de usuario.

Los **interesados** para este proyecto corresponden a la Escuela de Ingeniería Civil en Computación y los estudiantes de dicha carrera.

3.2.1.3. Eventos de Scrum

En Scrum existen cinco eventos cuyo propósito es mantener la transparencia, inspeccionar avances y adaptarse a los cambios. A continuación, se detalla cada uno de ellos:

- **Sprint**

  Es el principal evento de Scrum y contiene a los otros cuatro. Un *sprint* corresponde a un período de tiempo, generalmente entre dos y cuatro semanas, en que se genera un incremento del producto. Cada *sprint* tiene un objetivo con respecto al producto, definido por el *Sprint Backlog*. Durante el *sprint* no se realizan cambios significativos de planificación que puedan afectar su objetivo. Una vez iniciado, la duración y objetivo del *sprint* no pueden ser modificados.

- **Sprint Planning**

  Al inicio de cada *sprint* se realiza una reunión en la que se lleva a cabo su planificación. La planificación es un trabajo colaborativo, involucrando a todos los miembros del equipo. Se fija un objetivo para el *sprint* y se hace la correspondencia con el conjunto de historias de usuario que lo cumplen, generando así el *Sprint Backlog*.

- **Daily Meeting**

  Una parte importante de Scrum son las reuniones diarias de los miembros

del equipo. Como su nombre lo indica, esta reunión se realiza una vez al día y su propósito es mantener al equipo informado sobre el avance que llevan, así como conocer los obstáculos que existen. La duración no debe superar los 15 minutos y cada miembro del equipo tiene que responder tres preguntas:

- ¿Qué hice ayer para ayudar al equipo a avanzar a favor del objetivo del *sprint*?
- ¿Qué haré hoy para ayudar al equipo a avanzar a favor del objetivo del *sprint*?
- ¿Veo algún impedimento que evita que yo o el equipo pueda cumplir con el objetivo del *sprint*?

El ***scrum master*** Es el responsable de asegurar que la reunión diaria se realice, pero el equipo es responsable de dirigirla. En caso de que algún miembro del equipo reporte un impedimento, es trabajo del equipo dar solución a la situación.

- **Sprint Review**

  Al final del *sprint* se realiza una reunión de revisión a la que asiste el equipo de Scrum y los interesados. Se inspecciona el incremento del producto y se adapta el *Product Backlog* si es necesario. Es importante notar que esta es una reunión informal y no una de estatus, teniendo la finalidad de acordar la dirección del desarrollo y los elementos que serán trabajados en los *sprint* siguientes. El resultado final de esta reunión es un *product Backlog* revisado, que define las historias que probablemente se trabajarán en el siguiente *sprint*.

- **Sprint Retrospective**

  Esta reunión también es realizada al finalizar un *sprint* y es una instancia donde participa todo el equipo de Scrum. El equipo realiza una introspección y busca generar un plan para mejorar en el siguiente *sprint*. Se define todo

aquello que se debe mantener, las cosas nuevas a intentar (pensando siempre en mejorar) y las acciones a realizar para cumplir con los puntos anteriores. A cada acción acordada se le asigna un responsable dentro del equipo, el cual debe asegurar que se cumpla.

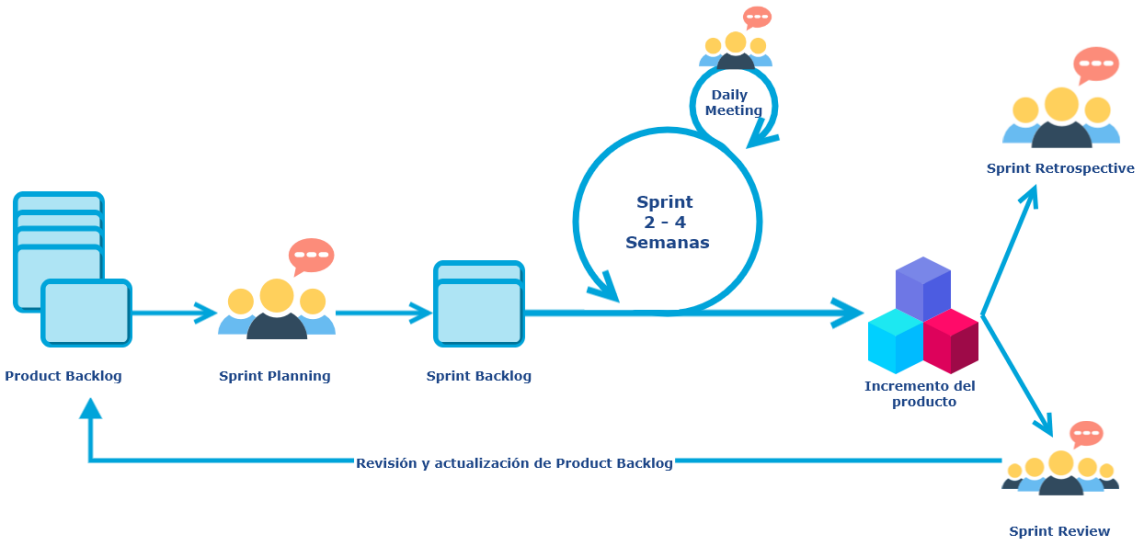


Figura 3.1: Resumen de Scrum

La **Figura 3.1** muestra un resumen de todo el proceso de Scrum. Durante el *Sprint Planning* se seleccionan las *historias de usuario* del *Product Backlog* que serán implementadas, pasando estas a ser parte del *Sprint Backlog*. Posteriormente se da inicio al período de implementación, realizando las *Daily Meetings* cada día del *sprint*. Finalizando el *sprint*, se tiene un incremento del producto y se realizan las reuniones finales: *Sprint Review* (donde se revisa y actualiza el *Product Backlog*) y *Sprint Retrospective*; dando paso a un nuevo *sprint*.

3.2.2. *Sprints* y *Backlog* del proyecto

Para el desarrollo del sistema en este proyecto se define que:

- Se realizarán **tres *sprints***.
- La duración de cada *sprint* será de **dos semanas laborales** (14 días laborales).
- En caso de ser necesario, se evaluá la realización de un cuarto *sprint*. Finalmente, dicho *sprint* no fue necesario, ya que se completaron todas las historias de usuario durante los tres primeros *sprint*.

Este proyecto tiene una **etapa previa** al inicio del primer *sprint*, en la cual se busca obtener las historias de usuario iniciales. Además, se realiza un diseño del sistema a construir a partir de las historias obtenidas. Posterior a esto se da inicio a la etapa de construcción.

El *Backlog* del proyecto fue definido en conjunto con los interesados. El detalle del *Backlog* de cada *sprint* corresponde a:

Sprint 1

- Vista de bienvenida y autenticación del usuario.
- Vista de cuenta tipo administrador.
- Permitir a un administrador crear una nueva malla.
- Permitir a un administrador crear un nuevo módulo.
- Permitir a un administrador crear un nuevo prerrequisito de un módulo.
- Permitir a un administrador modificar una malla.
- Permitir a un administrador modificar un módulo.

- Permitir a un administrador modificar un prerrequisito de un módulo.
- Permitir a un administrador eliminar una malla.
- Permitir a un administrador eliminar un módulo.
- Permitir a un administrador eliminar un prerrequisito de un módulo.

Sprint 2

- Modificar el algoritmo para permitir prerrequisitos de tipo *módulo cursado*.
- Permitir a un administrador crear nuevos usuarios de tipo administrador.
- Permitir a un administrador crear nuevos usuarios de tipo estudiante.
- Permitir a un administrador crear múltiples usuarios de tipo estudiante al subir una planilla de Microsoft Excel.
- Vista de cuenta tipo estudiante.
- Permitir a un usuario cambiar su correo.
- Permitir a un usuario cambiar su nombre.
- Permitir a un usuario cambiar su contraseña.
- Permitir a un estudiante editar y guardar detalles específicos de una cuenta estudiante (malla y parámetros de simulación).

Sprint 3

- Permitir a un estudiante seleccionar los módulos que ha aprobado y guardar esta selección.

- Permitir a un estudiante seleccionar los módulos que ha reprobado y guardar esta selección.
- Permitir a un estudiante realizar simulaciones utilizando el algoritmo y generar una vista de resultados.
- Permitir a un estudiante seleccionar un plan de inscripción entre los resultados obtenidos de la simulación y guardar esta selección.
- Permitir a un administrador realizar solicitudes y ver estadísticas de los planes de inscripción de los estudiantes.
- Vista de recuperación de cuenta y permitir a un usuario recuperar sus credenciales.
- Implementar vistas de estudiante para que sean responsivas al tamaño de la pantalla (Diseño Web Responsive[1]).

Notar que los *sprints* no toman en cuenta el trabajo previo con el algoritmo, ya que éste no se realiza dentro del marco del desarrollo de software del sistema, pero sí toma en cuenta trabajos posteriores y modificaciones con el fin de satisfacer necesidades del sistema.

3.3. Metodología de evaluación

Con la finalidad de evaluar el nivel de aceptación y futura adopción del sistema implementado se definen dos instancias de evaluación. La primera corresponde a evaluar la percepción del sistema que tienen los **estudiantes** según:

- Qué tan útil es el sistema para ellos en cuanto a la inscripción de módulos.
- Qué tan completo es el sistema en cuanto a funcionalidades. ¿Son suficientes para que sea útil?

[1]**Diseño Web Responsive** es un diseño web capaz de redistribuir los elementos en pantalla para que se adapten a todo tipo de dispositivos.

- Qué tan usable es para ellos la interfaz de usuario.

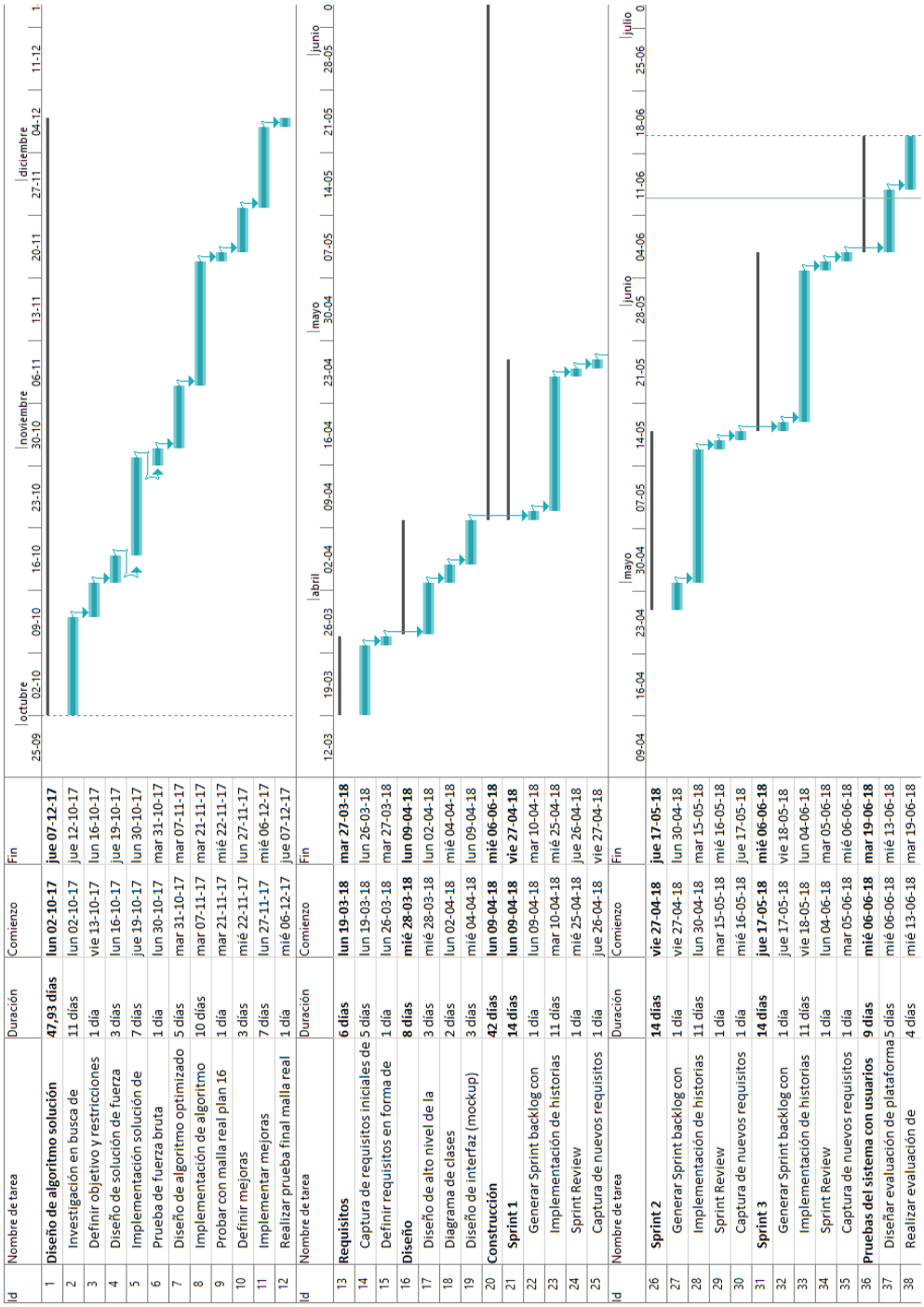
Esta evaluación se realiza con una encuesta, previa demostración del sistema a los estudiantes. En la demostración los estudiantes hacen uso del sistema y prueban sus funcionalidades.

La segunda instancia de evaluación corresponde a la aprobación del sistema por parte de la **Dirección de Escuela** de la carrera. Al igual que con los estudiantes, se realiza una demostración del sistema, incluyendo las funcionalidades de administrador.

3.4. Planificación

La planificación del proyecto comprende varias etapas. La primera es la de diseño del algoritmo, donde se utiliza la metodología descrita en la **Sección 3.1**. La segunda corresponde al desarrollo del sistema, donde se utiliza la metodología descrita en la **Sección 3.2**. Finalmente la tercera etapa corresponde a la evaluación del sistema construido, donde se utiliza la metodología descrita en la **Sección 3.3**.

La **Figura 3.2** detalla la planificación global del proyecto. Es posible apreciar las diferentes etapas y los tiempos dedicados a cada una de ellas. La duración en días toma en cuenta **sólo días laborales**.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Diseño de algoritmo solución**	**47,93 días**	**lun 02-10-17**	**jue 07-12-17**
2	Investigación en busca de	11 días	lun 02-10-17	jue 12-10-17
3	Definir objetivo y restricciones	1 día	vie 13-10-17	lun 16-10-17
4	Diseño de solución de fuerza	3 días	lun 16-10-17	jue 19-10-17
5	Implementación solución de	7 días	jue 19-10-17	lun 30-10-17
6	Prueba de fuerza bruta	1 día	lun 30-10-17	mar 31-10-17
7	Diseño de algoritmo optimizado	5 días	mar 31-10-17	mar 07-11-17
8	Implementación de algoritmo	10 días	mar 07-11-17	mar 21-11-17
9	Probar con malla real plan 16	1 día	mar 21-11-17	mié 22-11-17
10	Definir mejoras	3 días	mié 22-11-17	lun 27-11-17
11	Implementar mejoras	7 días	lun 27-11-17	mié 06-12-17
12	Realizar prueba final malla real	1 día	mié 06-12-17	jue 07-12-17
Id	Nombre de tarea	Duración	Comienzo	Fin
13	**Requisitos**	**6 días**	**lun 19-03-18**	**mar 27-03-18**
14	Captura de requisitos iniciales de	5 días	lun 19-03-18	lun 26-03-18
15	Definir requisitos en forma de	1 día	lun 26-03-18	mar 27-03-18
16	**Diseño**	**8 días**	**mié 28-03-18**	**lun 09-04-18**
17	Diseño de alto nivel de la	3 días	mié 28-03-18	lun 02-04-18
18	Diagrama de clases	2 días	lun 02-04-18	mié 04-04-18
19	Diseño de interfaz (mockup)	3 días	mié 04-04-18	lun 09-04-18
20	**Construcción**	**42 días**	**lun 09-04-18**	**mié 06-06-18**
21	**Sprint 1**	**14 días**	**lun 09-04-18**	**vie 27-04-18**
22	Generar Sprint backlog con	1 día	lun 09-04-18	mar 10-04-18
23	Implementación de historias	11 días	mar 10-04-18	mié 25-04-18
24	Sprint Review	1 día	mié 25-04-18	jue 26-04-18
25	Captura de nuevos requisitos	1 día	jue 26-04-18	vie 27-04-18
Id	Nombre de tarea	Duración	Comienzo	Fin
26	**Sprint 2**	**14 días**	**vie 27-04-18**	**jue 17-05-18**
27	Generar Sprint backlog con	1 día	vie 27-04-18	lun 30-04-18
28	Implementación de historias	11 días	lun 30-04-18	mar 15-05-18
29	Sprint Review	1 día	mar 15-05-18	mié 16-05-18
30	Captura de nuevos requisitos	1 día	mié 16-05-18	jue 17-05-18
31	**Sprint 3**	**14 días**	**jue 17-05-18**	**mié 06-06-18**
32	Generar Sprint backlog con	1 día	jue 17-05-18	vie 18-05-18
33	Implementación de historias	11 días	vie 18-05-18	lun 04-06-18
34	Sprint Review	1 día	lun 04-06-18	mar 05-06-18
35	Captura de nuevos requisitos	1 día	mar 05-06-18	mié 06-06-18
36	**Pruebas del sistema con usuarios**	**9 días**	**mié 06-06-18**	**mar 19-06-18**
37	Diseñar evaluación de plataforma	5 días	mié 06-06-18	mié 13-06-18
38	Realizar evaluación de	4 días	mié 13-06-18	mar 19-06-18

Figura 3.2: Planificación del proyecto

4. Desarrollo del proyecto

El contenido de este capítulo busca dar a conocer al lector el desarrollo de las distintas piezas que componen la solución.

En primera instancia se explica la abstracción del problema, detallando el modelo de datos y la arquitectura de software de la solución. Luego se detalla el trabajo relacionado al algoritmo, destacando los diseños realizados y las mejoras implementadas. Finalmente se describe la construcción del sistema, especificando las diferentes partes que lo componen y su interacción.

4.1. Modelo de datos

Parte de la abstracción del problema implica reconocer los diferentes elementos que forman parte de él (**Sección 2.1**). Teniendo en consideración cada uno de estos elementos de la problemática y sus relaciones se utiliza un diagrama de clases (**Figura 4.1**) para representar el modelo de datos.

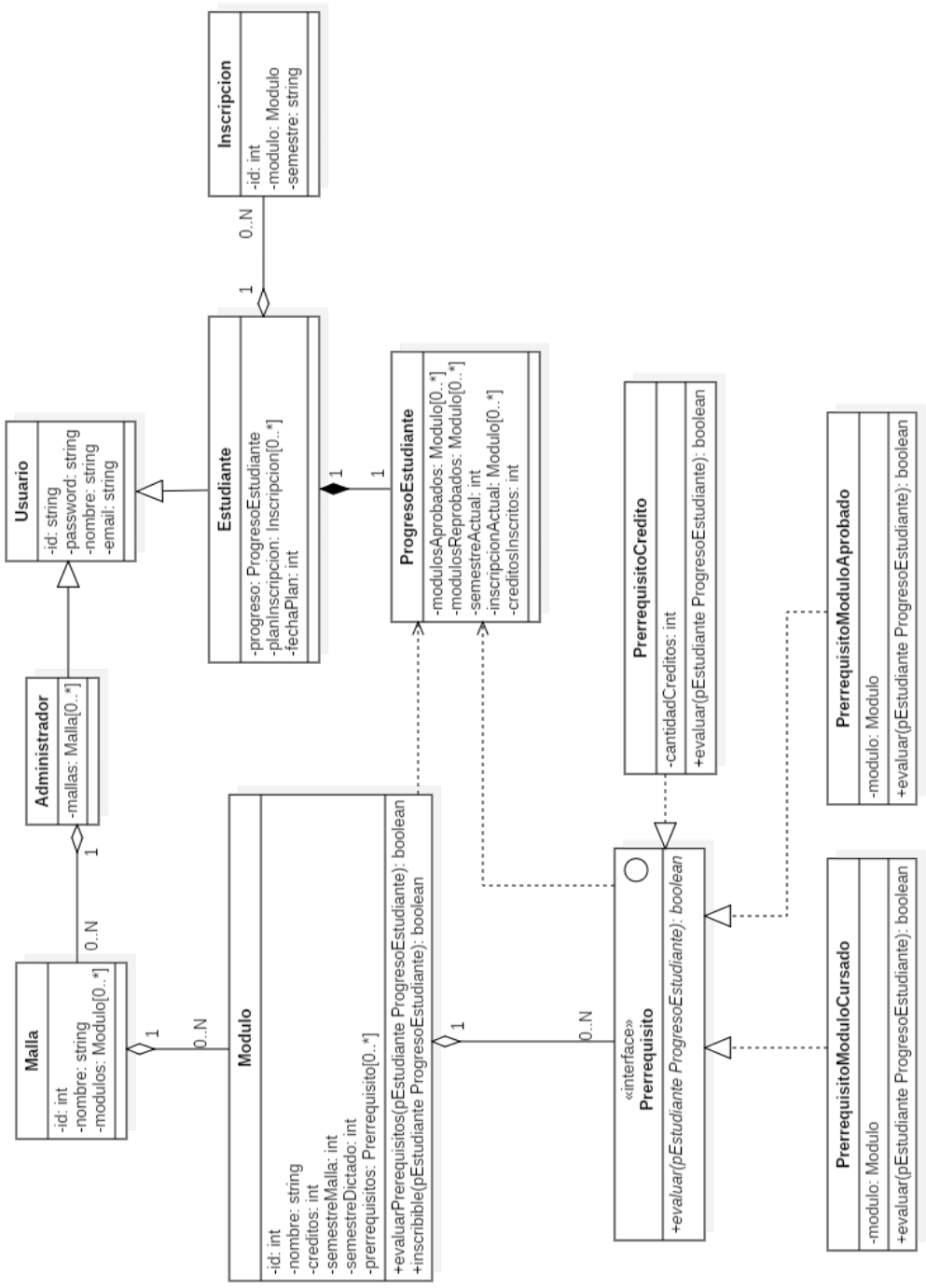


Figura 4.1: Diagrama de clases que detalla el modelo de datos

Parte de la propuesta de solución corresponde a desarrollar un sistema a través del cual los estudiantes puedan utilizar el algoritmo y conocer cuáles son sus opciones respecto a la inscripción de módulos. Con respecto a esto es necesario aclarar que algunas de las clases del diagrama presentan utilidad para el sistema, pero no para el algoritmo, las clases a las que se hace alusión son: Usuario, Administrador, Estudiante (sólo es necesario el progreso para el algoritmo) e Inscripción.

Cada clase del diagrama tiene un objetivo específico, los cuales son detallados a continuación:

- **Usuario:** Representa a un usuario del sistema de forma general. Presenta atributos para identificarlo, así como otros datos de interés.
- **Administrador:** Representa a un usuario de tipo administrador. Esta clase hereda de la clase **Usuario**, por lo que también tiene los atributos que identifican a un usuario. Adicionalmente tiene una lista de **mallas**, ya que son los administradores los que crean y definen las mallas. Un administrador puede ser, por ejemplo, un miembro de la Escuela de ICC.
- **Estudiante:** Representa a un usuario de tipo estudiante. Esta clase hereda de la clase **Usuario**, por lo que también tiene los atributos que identifican a un usuario. Como su nombre lo indica, tiene atributos específicos de un estudiante en el sistema, como lo son su **progreso**, el **plan de inscripción** que está siguiendo, y la fecha en que actualizó por última vez su plan de inscripción.
- **ProgresoEstudiante:** Representa el progreso de un estudiante del sistema. El progreso está definido por los módulos que ha **aprobado** y los que ha **reprobado** el estudiante (estos últimos sirven para verificar el requisito de módulo cursado). Además contiene los **módulos inscritos** y el semestre en que se encuentra, los cuales son útiles para realizar simulaciones.
- **Inscripcion:** Representa la inscripción de un módulo por parte de un estudiante, para un período en particular. Posee un atributo identificador. Es utilizado para definir un plan de inscripción de un estudiante.

- **Malla:** Representa a una malla. Esta representación se realiza por medio de una **lista de módulos** que componen dicha malla. Posee también atributos de utilidad, como un identificador y un nombre.
- **Módulo:** Representa un módulo de una malla. Contiene atributos relevantes de un módulo: **cantidad de créditos** asignados, el semestre en que se ubica en la malla, el o los semestres en que se imparte y los **prerrequisitos** que tiene para ser inscrito. Al igual que otras clases, posee un atributo identificador y un nombre. Tiene también métodos para comprobar si se puede inscribir el módulo según el progreso de un estudiante.
- **Prerrequisito:** Corresponde a una interfaz que representa de forma general varios tipos de prerrequisitos. Se destaca su método evaluar, que según su implementación, indica si se cumple o no el prerrequisito.
- **PrerrequisitoModuloCursado:** Representa a un prerrequisito de tipo módulo cursado. El método evaluar indica si, según el progreso de un estudiante, el módulo (atributo) fue cursado.
- **PrerrequisitoModuloAprobado:** Representa a un prerrequisito de tipo módulo aprobado. El método evaluar indica si, según el progreso de un estudiante, el módulo (atributo) fue aprobado.
- **PrerrequisitoCredito:** Representa a un prerrequisito de tipo crédito. El método evaluar indica si, según el progreso de un estudiante, por lo menos ha aprobado una cierta cantidad de créditos (atributo).

4.2. Arquitectura

Dadas las necesidades del sistema y pensando en el potencial uso por parte de los estudiantes, se selecciona una arquitectura de **tres capas: cliente, servidor, base de datos**. La idea principal es proveer el servicio por red (Internet o institucional) para que de esta forma pueda llegar a los usuarios finales de mejor forma.

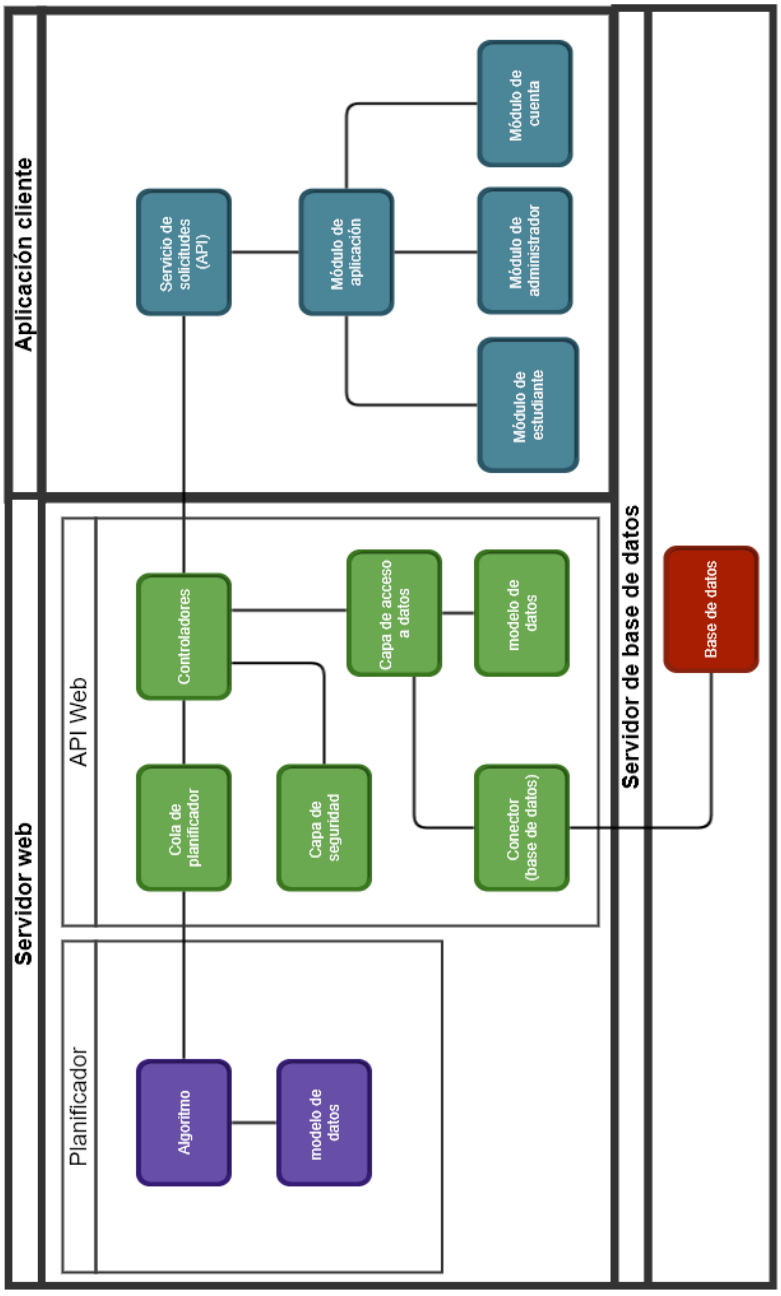


Figura 4.2: Diagrama que detalla la arquitectura de la solución

La **Figura 4.2** da a conocer la arquitectura escogida y las diferentes partes que la componen, destacando las tres capas mencionadas. A continuación se describe cada uno de estos elementos.

Servidor web

El servidor web está compuesto por dos elementos principales: la API[1] web y el planificador.

- **API web:** Proporciona las funcionalidades para trabajar los datos y **encolar** los trabajos de simulación a ser resueltos por el **planificador**. Tiene una capa de seguridad, que se encarga de validar a los usuarios y administrar el acceso a los datos según el tipo de usuario. La capa de acceso a datos, como su nombre lo indica, proporciona el acceso a los datos almacenados en la base de datos. Cuenta también con un componente que encola peticiones para el planificador. La necesidad de encolar radica en tener la capacidad de administrar los recursos al realizar simulaciones y evitar situaciones en que se generen muchas simulaciones simultáneas, poniendo en riesgo el correcto funcionamiento de la máquina física donde se encuentra alojado el sistema.
- **Planificador:** Componente de software que implementa el algoritmo de simulación de inscripción. Una instancia del planificador simula y retorna los planes de inscripción para un conjunto de datos específicos recibidos como entrada.

Servidor de base de datos

Servidor que aloja y proporciona acceso a la base de datos del sistema y proporciona el acceso a ésta. Los detalles de las herramientas utilizadas en este servidor se encuentran en la **Sección 4.2.1**.

[1] **Application Programming Interface** corresponde a un conjunto de subrutinas, funciones y métodos que proporcionan una capa abstracta de comunicación entre componentes de software.

Aplicación de cliente

La aplicación de cliente es aquélla que permite a los usuarios hacer uso el sistema, siendo su "cara visible". Se decide que el cliente será una aplicación web (en navegador), ya que de esta manera los usuarios pueden acceder fácilmente sin necesidad de obtener el software por otros medios. Se destacan cuatro partes de la aplicación:

- **Módulo de aplicación:** Es el módulo principal de la aplicación, el cual contiene a los demás módulos y servicios.
- **Módulo de administrador:** Es el módulo que implementa las funcionalidades de un administrador. Permite a un usuario ver, editar y eliminar mallas, módulos y prerrequisitos. También es responsable de la administración de usuarios, por lo que tiene las funcionalidades de crear nuevos usuarios o eliminarlos.
- **Módulo de estudiante:** Es el módulo que implementa las funcionalidades de un estudiante. Permite a un usuario realizar simulaciones y revisar las alternativas de inscripción que tiene según la configuración de parámetros ingresada.
- **Módulo de cuenta:** Es el módulo que implementa las funcionalidades de una cuenta. Permite a un usuario ver y editar datos de su cuenta.
- **Servicio de solicitudes:** Es el servicio encargado de comunicarse con la API del servidor para realizar las diferentes solicitudes.

4.2.1. Stack de software

El **stack de software** define el conjunto de herramientas de software que se utilizan para lograr un objetivo común. En el caso de este proyecto ese objetivo corresponde al sistema a implementar [12].

A continuación, se detalla el *stack de software* seleccionado para el desarrollo del sistema, dando a conocer también las decisiones detrás de cada elección.

- **Spring Framework**

  Es un *framework* de desarrollo de aplicaciones para la plataforma **Java**, teniendo como principal característica su contenedor de *inversión de control* (IoC)[2] por medio de inyección de dependencias. Las tareas de instanciar, inicializar y conectar objetos son llevadas a cabo por el *framework*.

  Spring se enfoca en realizar todas las acciones necesarias para levantar una aplicación empresarial, mientras que los desarrolladores pueden enfocarse en desarrollar la lógica de negocio [4].

  Existen una serie de *frameworks* asociados a Spring (son subproyectos del framework), de los cuales se seleccionan aquellos que presentan utilidad para este proyecto:

  - Spring Boot: Es un framework para aplicaciones de Spring que se encarga de auto-configurar la mayoría de los distintos módulos y frameworks de Spring. Dentro de sus principales características están:
    - Crear aplicaciones independientes (no requieren de dependencias externas para ser ejecutadas).
    - Tomcat Embebido (no requiere de un servidor web instalado en la máquina, ya que el ejecutable generado trae uno).
    - Facilitar el desarrollo de aplicaciones basadas en Spring en general.
  - Spring Security: Es un framework para la autenticación y el control de acceso en aplicaciones basadas en Spring. Es muy útil a la hora de agregar seguridad a la aplicación desarrollada.

[2] **Inversion de Control** es un patrón de diseño de software en que es el framework quien toma el control del flujo del código para tareas de alto nivel, mientras que las tareas de bajo nivel siguen siendo implementadas por el código del usuario [5].

- Spring Data: es un framework cuya misión es proveer de un modelo de programación basado en Spring para el acceso a datos. Este framework provee acceso a datos de forma transparente para el usuario y es posible configurar una gran variedad de servidores de bases de datos.

Debido a que Spring Framework y sus asociados de por sí realizan o facilitan varias de las características que requiere el sistema a desarrollar, la elección de la herramienta es clara. También es necesario destacar que el autor tiene experiencia previa con este Framework.

- **Angular**

Es un *framework* para desarrollo de aplicaciones web de una sola página. Se centra en mantener parte de la lógica de negocio en la aplicación del cliente, utilizando así sus recursos y dejando al servidor con recursos para cumplir otras tareas. Esto último es muy útil si el servidor de la aplicación no tiene gran cantidad de recursos.

Dentro de las principales características se encuentran [1]:

- Optimización de código: convierte las plantillas en código altamente optimizado para las máquinas virtuales de Javascript.
- Separación de código: sólo renderiza lo que requiere la vista a la que está accediendo el usuario.
- Se puede montar en cualquier servidor web, incluyendo el Tomcat embebido de Spring Boot.

La elección de esta herramienta se basa principalmente en la capacidad de utilizar los recursos del cliente, dejando libres los limitados recursos del servidor para otras tareas. Además, esta herramienta tiene la capacidad de trabajar en conjunto con Spring, otra herramienta seleccionada. Sin embargo, también se

toman en cuenta las otras características mencionadas, ya que son importantes para el producto final a desarrollar.

- **MySQL**

  Es un sistema de base de datos relacional de código abierto [2]. Es ideal para aplicaciones web, donde la lectura de datos es intensiva en comparación con la escritura. Esto último se debe a que su motor no transaccional MyISAM tiene buen rendimiento en esos casos.

  La elección de esta herramienta se basa en su uso previo por parte del autor y el hecho de que es soportada por Spring Data.

4.3. Algoritmo de solución

Antes de iniciar con el diseño de un algoritmo, se realizó una investigación de trabajos previos sobre problemas similares. Los trabajos revisados son mencionados en la **Sección 2.3.1**. Los resultados de esta investigación dejan en claro que no es posible utilizar directamente las soluciones propuestas en estos trabajos. Más aún, las soluciones abarcan mucho más de lo necesario para este proyecto, dada las diferencias en los sistemas de educación de cada país. Por los motivos anteriores se decide realizar el diseño del algoritmo desde cero.

Como parte del diseño del algoritmo se definen sus **objetivos**:

- **Obtener el mínimo número de semestres para el egreso de un estudiante.**
- **Obtener uno o varios planes de inscripción de módulos, semestre a semestre, que lleven a egresar en ese mínimo de tiempo.**

asimismo también se definen las **restricciones**:

- **Carga obligatoria:** Antes de que el estudiante inicie su inscripción, hay módulos que se cargan de forma automática al inicio del semestre (los de menor nivel).
- **Máximo número de créditos por semestre:** Durante un semestre específico, existe una cantidad máxima de créditos que un estudiante puede inscribir.
- **Máximo número de módulos por semestre:** Algunos estudiantes prefieren inscribir hasta cierta cantidad de módulos, que puede ser menor a la máxima que podría inscribirse.
- **Prerrequisitos de malla:** Los módulos en la malla de un plan pueden requerir tener aprobados/cursados otros módulos anteriores o haber pasado cierta cantidad de créditos aprobados antes de ser inscritos.
- **Semestre en que se dictan los módulos:** Los módulos se dictan sólo en el primer semestre, otros sólo en el segundo semestre y algunos en ambos.
- **No queden créditos sobrantes:** Se debe tomar la cantidad de créditos de tal forma que no quede un módulo fuera de la inscripción en el semestre, que sí podría haber sido inscrito.

Los objetivos, así como también las restricciones, buscan generar un algoritmo que sea capaz de entregar información útil y correcta a un estudiante. Entonces, los resultados de un algoritmo que cumpla con lo anterior estarán libres de datos innecesarios (planes de inscripción ineficientes o incorrectos).

Cabe destacar que el algoritmo trabaja bajo el supuesto de que siempre habrá vacantes en los módulos. Asimismo, trabaja con un número fijo de créditos máximos por semestre definido por un parámetro, ya que se supone que en caso de ser necesario, se puede pedir autorización a dirección de Escuela para exceder el máximo de créditos real (Siempre que la razón sea suficientemente buena y minimizar los semestres para el egreso se considera una buena razón). Finalmente, se vuelve a mencionar que la

solución no trabaja con la asignación de horarios, ya que está fuera del alcance del proyecto.

4.3.1. Diseño de fuerza bruta

Como primer acercamiento al algoritmo que soluciona el problema, se propone diseñar e implementar un algoritmo de fuerza bruta que cumpla con los objetivos y restricciones.

El algoritmo mencionado realiza una búsqueda completa, por medio de simulación de semestres e inscripciones, generando combinaciones posibles de inscripción en que finalmente se llega a una situación de egreso (todos los módulos aprobados). Además, toma en cuenta las restricciones, así como también va actualizando sus mejores resultados encontrados (guardando cantidad de semestres y planes de inscripción), cumpliendo de esta manera con los objetivos.

La función `simularSemestre` (**Pseudocódigo 1**) se encarga en primera instancia de revisar si las condiciones de término se cumplen (haber aprobado todos los módulos). En caso de que se cumplan, se procede a revisar si es igual o mejor, con respecto a la cantidad de semestres, que las soluciones actuales. En caso de que sea una mejor solución se guarda un nuevo conjunto de soluciones, desechando el anterior (sólo se mantienen las mejores soluciones en cuanto a número de semestres).

Posteriormente se realiza un filtro de módulos de la malla, para trabajar sólo con los que no han sido aprobados. De la misma forma se realiza la inscripción automática (nivel más bajo no aprobado por el estudiante). Se remueven, además, los módulos inscritos del conjunto de no aprobados, para que la simulación de inscripción de ese semestre no tome en cuenta los módulos que ya fueron inscritos de manera automática. Finalmente se llama a la función `simularInscripciones` (**Pseudocódigo 2**) utilizando el conjunto filtrado de módulos.

Pseudocódigo 1: función que simula un semestre

```
M = conjunto de módulos que conforman una malla
mins = mínimo número de semestres (la mejor solución encontrada)
S = conjunto de soluciones
function simularSemestre(P, count)
    input : P = progreso de un estudiante (Módulos aprobados, reprobados,
            inscritos)
            count = contador de semestres
    /* Se han aprobado todos los módulos                                  */
    if |P.aprobados| = |M| then
        /* si el tiempo de egreso coincide con el menor encontrado        */
        if mins = count then
            S = S ∪ {P};                          // se agrega la solución
        else if mins > count then                 // si hay un nuevo mínimo
            mins ← count;
            /* descarta S e inicia a un nuevo conjunto de mejores
               soluciones                                                 */
            S = {P};
        return;
    end if
    M' ← M \ P.aprobados;                  // filtro: módulos no aprobados
    /* carga obligatoria de semestre                                      */
    minSemestre ← semestreMinimo(M');    // mínimo semestre no aprobado
    foreach modulo ∈ M' do
        if modulo.semestreMalla = minSemestre and modulo.inscribible(P)
         then
            P.inscribir(modulo)
        end if
    end foreach
    M' ← M' \ P.inscritos;                                // remueve inscritos
    simularInscripciones(P,M',0,count,∞)
end function
```

La función `simularInscripciones` es el corazón del algoritmo. Mientras que la función `simularSemestre` se encarga de las soluciones encontradas y filtrar los módulos, `simularInscripciones` se encarga de generar todas las combinaciones de módulos posibles, definiendo las inscripciones para un semestre. Esto lo realiza ejecutando una búsqueda completa y podando las ramas que no sirven según las restricciones del algoritmo.

En primer lugar, se revisa si la condición de término de simulación se cumple, la cual corresponde a no poder inscribir más módulos (llegar al índice final para el conjunto de módulos). En caso de que se cumpla la condición de término, se procede a comprobar si los módulos se inscribieron de forma óptima, esto es, sin créditos sobrantes y con la mayor cantidad de módulos posible, respetando la restricción de cantidad máxima de módulos. Si hubo una inscripción óptima, se actualizan los datos del progreso de estudiante para iniciar un nuevo semestre con la función `simularSemestre`. Si la inscripción no es óptima, ésta es desechada.

Notar que `simularInscripciones` es una función recursiva que se encarga de realizar la simulación de un sólo módulo por llamada. La recursividad es utilizada para simular el caso en que el módulo **no es inscrito** y el caso en que **sí es inscrito**, siempre revisando si es posible la inscripción según los prerrequisitos y restricciones.

En el caso de no inscripción de un módulo, antes de hacer la llamada recursiva, se actualiza la variable *mínima cantidad de créditos no inscritos*. Esta variable es útil para revisar si la inscripción es óptima en el primer paso, ya que si aún quedan créditos para inscribir el módulo con menor cantidad de créditos que no fue inscrito, la inscripción está violando la restricción de que no queden créditos sobrantes.

Es necesario mencionar que la variable de **progreso de un estudiante** es importante para ambas funciones, ya que es la que define el estado del estudiante con respecto a su avance en cada paso de la simulación. Por lo anterior, es parte de los datos de entrada en las dos funciones descritas.

Pseudocódigo 2: función que simula inscripciones para un semestre

```
maxCreditos = límite de créditos por semestre
maxInscripciones = límite de inscripciones por semestre
function simularInscripciones(P, M, i, count, minc)
    input : P = progreso de un estudiante (Módulos aprobados, reprobados, inscritos)
            M = conjunto de los módulos no aprobados por el estudiante
            i = índice actual de la simulación para M
            count = contador de semestres
            minc = Mínima cantidad de creditos no inscritos
    if i ≥ |M| then
        if P.creditosInscritos + minc > maxCreditos or |P.inscritos| = maxInscripciones then
            P' ← P;                                          // copia contenido a nueva variable
            P'.semestreActual ← P'.semestreActual + 1;
            P'.aprobados ← P'.aprobados ∪ P'.inscritos;
            P'.reprobados ← P'.reprobados \ P'.inscritos;
            P'.inscritos ← ∅;
            simularSemestre(P', count + 1);                  // inicia nuevo semestre
        end if
        return;
    end if
    minc' ← minc
    if M[i].inscribible(P) then
        minc' ← min{minc', M[i].creditos};
    end if
    /* caso en que no se inscribe el módulo                                          */
    simularInscripciones(P, M, i + 1, count, minc');
    /* si no se puede inscribir el modulo (por exceso de créditos o modulos inscritos) se
       retorna                                                                       */
    if P.creditosInscritos + M[i].creditos > maxCreditos or |P.inscritos| > maxInscritos then
        return;
    end if
    /* caso en que se inscribe el módulo (si es posible)                             */
    P' ← P;
    if M[i].inscribible(P) then
        P.inscribir(M[i]);
        simularInscripciones(P, M, i + 1, count, minc);
    end if
end function
```

Pruebas de la implementación

La implementación del algoritmo fue realizada en el lenguaje de programación *Java* (versión 8). Los detalles del sistema en que se realizaron las pruebas son:

- **Sistema operativo:** Windows 10 64-bit
- **Procesador:** Intel® Core™ i3-6100U con gráficos Intel HD 520 (2,3 GHz, 3 MB de caché, 2 núcleos)
- **Memoria RAM:** DDR3L 4 GB (1600 Mhz).

Se realizaron pruebas midiendo tiempo de ejecución del algoritmo para casos de **estudiantes desde primer año hasta sexto año**. El detalle de las pruebas realizadas puede ser encontrado en el **Anexo A**.

Un problema notorio de la implementación corresponde a la necesidad de comprobar que las soluciones no se repitan, ya que esto no lo realizan las funciones de simulación.

Caso de estudiante	Tiempo de ejecución (segundos)
1^{er} año	Error
2^{do} año	95,044
3^{er} año	6,45
4^{to} año	0,13
5^{to} año	0,07
6^{to} año	0,06

Tabla 4.1: Pruebas de algoritmo de fuerza bruta

La **Tabla 4.1** muestra los resultados de las pruebas realizadas. Si bien el algoritmo funciona para alumnos desde el 2^{do} hasta el 6^{to} semestre, es importante notar los tiempos de ejecución. En el caso de 2^{do} semestre el tiempo que tarda el algoritmo es de 95 segundos, un tiempo no despreciable. La situación es peor para la prueba de 1^{er} semestre, donde no se logró terminar la ejecución debido a que la cantidad de

llamadas recursivas provocó un error de ejecución al consumir una gran cantidad de memoria en el sistema.

Para un sistema que podría hacer uso de este algoritmo, con varios usuarios realizando simulaciones a la vez, no es deseable que el algoritmo en ciertos casos tenga:

- Un uso excesivo de recursos.
- Grandes tiempos de espera.
- Errores en la ejecución.

Se plantea la necesidad de seguir mejorando el diseño. Dado que en varios casos funciona como se espera, **se decide conservar la idea detrás del diseño buscando formas de mejorarlo**.

4.3.2. Optimización utilizando memoización

Dentro de las posibles mejoras aplicables a la solución se plantea el uso de *programación dinámica*(PD). La programación dinámica es un paradigma para abordar problemas que se basa en transformar un problema complejo en una secuencia de sub-problemas, teniendo la posibilidad de separar éstos últimos en otros sub-problemas y así sucesivamente. Para que un problema se pueda resolver utilizando PD se tienen que cumplir dos prerrequisitos [11]:

- **El problema tiene sub-estructuras óptimas:** La solución del sub-problema es parte de la solución del problema original.
- **El problema tiene sub-problemas solapados:** Los mismos sub-problemas se repiten en las distintas ramas de la búsqueda de soluciones.

El problema de las inscripciones posee los prerrequisitos para utilizar PD. Cada módulo inscrito forma parte de la solución de un sub-problema, que en su conjunto

resuelven un problema mayor: llegar a un plan de inscripción. Además, la misma combinación de módulos se repite en distintas inscripciones de semestres que se encuentran en distintas ramas de la simulación.

Para que la programación dinámica sea efectiva, es necesario identificar **los estados del problema y sus transiciones**. Un **estado** corresponde al conjunto de parámetros que definen de manera única un sub-problema. Una **transición** define el cambio que lleva de un estado al otro.

En el caso del algoritmo a mejorar se tiene que:

- **Estados:** Son definidos por el conjunto de módulos que ha aprobado y reprobado el estudiante junto con el semestre en que se encuentra en la simulación. Esto determina de forma única una situación en que se encuentra el alumno con respecto a su progreso.
- **Transiciones:** Corresponden a las inscripciones realizadas (ya que estas generan un nuevo conjunto de módulos aprobados al final del semestre) y el cambio de semestre.

Existe una forma de implementación de PD llamada **Memoización** [11]. La memoización se basa en un concepto simple: *recordar*. Al utilizar memoización se recuerdan los estados que ya han sido resueltos para evitar resolverlos nuevamente. La forma típica de implementación involucra una tabla de búsqueda, donde se almacenan los estados, generalmente representados por un valor numérico obtenido al utilizar alguna función de hash.

En el algoritmo **la memoización se aplica antes** de una nueva simulación de inscripción, ya que en esa función es donde se genera la transición de un estado a otro.

Pseudocódigo 3: memoización en función `simularSemestre`

```
M = conjunto de módulos que conforman una malla
mins = mínimo número de semestres (la mejor solución encontrada)
S = conjunto de soluciones
memo = conjunto de estados                                   // nueva variable memo
function simularSemestre(P, count)
    input : P = progreso de un estudiante (Módulos aprobados, reprobados, inscritos)
            count = contador de semestres
    /* Se han aprobado todos los módulos                                          */
    if |P.aprobados| = |M| then
        /* si el tiempo de egreso coincide con el menor encontrado                */
        if mins = count then
            S = S ∪ {P};                                     // se agrega la solución
        else if mins > count then                            // si hay un nuevo mínimo
            mins ← count;
            /* descarta S e inicia a un nuevo conjunto de mejores soluciones      */
            S = {P};
        return;
    end if
    M' ← M \ P.aprobados;                                    // filtro: módulos no aprobados
    /* carga obligatoria de semestre                                              */
    minSemestre ← semestreMinimo(M');                        // mínimo semestre no aprobado
    foreach modulo ∈ M' do
        if modulo.semestreMalla = minSemestre and modulo.inscribible(P) then
            P.inscribir(modulo)
        end if
    end foreach
    M' ← M' \ P.inscritos;                                   // remueve inscritos
    /* revisa si el estado ya fue resuelto con anterioridad                       */
    if P ∉ memo then
        /* si no ha sido resuelto, se guarda el estado                            */
        memo ← memo ∪ {P};
        simularInscripciones(P,M',0,count,∞)
    end if
end function
```

El **Pseudocódigo 3** muestra los cambios realizados a la función `simularSemestre` para implementar la memoización. Al revisar si ya se ha ejecutado un estado se evita volver a ejecutar el mismo caso. Es necesario notar que se almacena la variable de progreso P en el conjunto de estados, pero en una implementación real es necesario utilizar el conjunto de módulos aprobados y reprobados, así como el valor de semestre, ya que son éstos los que realmente definen un estado.

Pruebas de la implementación con memoización

Las pruebas de la implementación se realizaron de igual manera que para el algoritmo de fuerza bruta. Para la implementación de memoización se utilizó una tabla hash en la que se almacena el valor de hash calculado a partir del semestre, los módulos aprobados y los módulos reprobados de la clase ProgresoEstudiante.

Inicialmente para definir los estados de la tabla de búsqueda se utilizó una implementación que consistía en ordenar las listas de módulos aprobados y reprobados, almacenar los nombres de éstos en una variable de tipo String junto al semestre y utilizar el método `hashCode` de Java, que implementa el cálculo de un valor hash a partir de una cadena de caracteres. Realizando pruebas se obtuvieron resultados que no son favorables (muchos peores incluso a la versión de fuerza bruta) por lo que finalmente se decidió cambiar esta implementación.

La implementación final consiste en calcular el valor de hash de ambas listas de módulos a partir de los identificadores de cada módulo y adicionalmente el valor hash de un semestre, juntando finalmente todos en una sola variable.

Se define, entonces, el valor de hash de un conjunto de módulos como la suma sus números identificadores. Además, se define el valor de hash de un semestre como el mismo valor asignado a la variable (la variable siempre aumenta al pasar de un semestre a otro). Entonces el valor total de hash para un estado corresponde a:

$$valor\_hash \quad = \quad 31^3 \cdot semestre + 31^2 \cdot aprobados + 31 \cdot reprobados$$

Para efectos de la explicación se usa el valor 31, siendo éste un número primo, teniendo así una menor probabilidad de colisiones en los valores de hash calculados. Es importante destacar que en la implementación se hace uso de los métodos de Java para realizar las operaciones de hashing y manejo de colisiones en una tabla hash.

Caso de estudiante	**Tiempo de ejecución (segundos)**
1^{er} año	21,79
2^{do} año	1,69
3^{er} año	1,13
4^{to} año	0,11
5^{to} año	0,09
6^{to} año	0,07

Tabla 4.2: Pruebas de algoritmo con memoización

Los resultados de las pruebas se pueden observar en la **Tabla 4.2**. A simple vista se puede apreciar que los tiempos son, en la mayoría de los casos, menores que en la implementación de fuerza bruta. En los casos de estudiantes que tienen mayor avance en sus estudios(5^{to} y 6^{to} año) el algoritmo de fuerza bruta tiene mejor rendimiento en cuanto a tiempo, aunque la diferencia no es realmente significativa.

Otra mejora de esta implementación es el hecho de que la memoización evita que se repitan soluciones, evitando tener que realizar un filtrado posterior.

La **Figura 4.3** muestra una gráfica comparativa de los dos algoritmos. Es notoria la reducción de tiempo que otorga la técnica de memoización al algoritmo de solución.

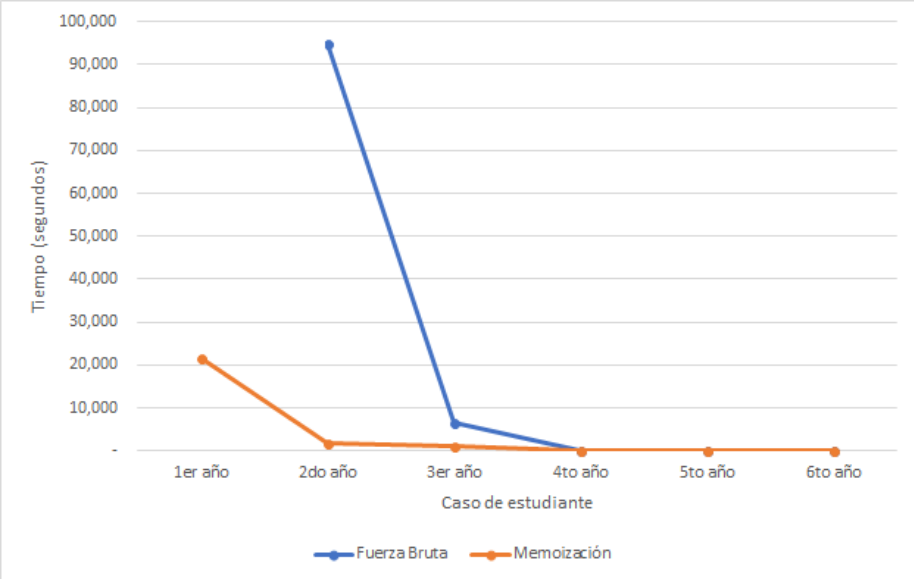


Figura 4.3: Gráfico comparativo de algoritmos

La **Figura 4.4** muestra la comparación logarítmica de los tiempos de ejecución de ambos algoritmos, donde es posible observar que para los casos de quinto y sexto año, la solución de fuerza bruta tiene mejor rendimiento. Esto se debe a la sobrecarga que causan las búsquedas en la tabla hash teniendo pocos módulos restantes (caso que la solución de fuerza bruta soluciona sin problemas).

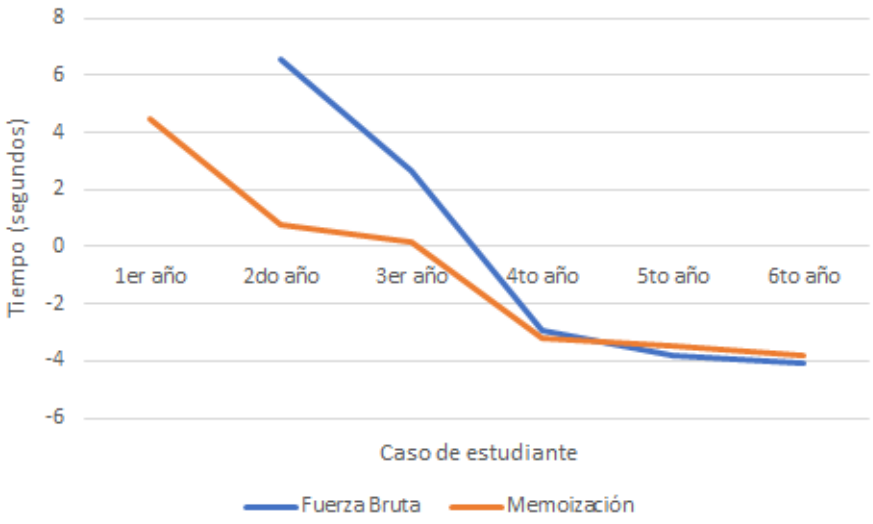


Figura 4.4: Gráfico comparativo de algoritmos (escala logarítmica)

Se determina que los resultados de la implementación con memoización son suficientemente buenos como para ser utilizada en el planificador del sistema.

4.4. Implementación: Servidor web

La implementación del servidor web hace uso de las diferentes tecnologías relacionadas a Spring Framework.

A modo general, las partes del servidor interactúan de la siguiente forma:

- Los controladores de API reciben solicitudes desde los clientes, previa autorización de la **capa de seguridad**.
- El procesamiento de las solicitudes implica obtener datos a través de la **capa**

de acceso a datos y, si es el caso, el **planificador**.

- Para el uso del planificador se realiza un **encolamiento de solicitudes**.

A continuación, se describen las partes del servidor en mayor detalle.

4.4.1. Controladores de API

La implementación de la API hace uso de los **controladores REST**[3] **de Spring**. Cada controlador define el formato de entrada, los procesos necesarios y el formato de respuesta para un tipo de consulta en particular.

En el proyecto se definen los siguientes controladores para la API:

- **Controlador de autenticación**

  Se encarga de las solicitudes relacionadas a la identificación y autenticación de un usuario en el sistema. Dentro de las solicitudes que define están:

  - Validar credenciales para un usuario.
  - Recuperar el acceso a una cuenta.

  Al validar las credenciales de un usuario, las registra en la capa de seguridad para futuras solicitudes (**Sección 4.4.4**).

- **Controlador de usuario:**

  Se encarga de las solicitudes relacionadas a los datos de los usuarios. Dentro de las solicitudes que define están:

[3]**REST**, o REpresentational State Transfer, es un estilo de arquitectura que provee estándares de comunicación entre sistemas computacionales en una red. Se usa particularmente en arquitecturas de tipo cliente-servidor, donde el cliente realiza solicitudes al servidor y este último procesa y responde dichas solicitudes.

- Obtener una lista de usuarios del sistema con su información general.
- Obtener la información del usuario que realiza la consulta. La información contiene datos específicos según el tipo de usuario (administrador o estudiante).
- Cambiar la contraseña del usuario que realiza la consulta.
- Eliminar un usuario.

Notar que las solicitudes son de tipo general. Si se requieren solicitudes más específicas, se pueden encontrar en los controladores de administrador o de estudiante.

- **Controlador de administrador**

  Se encarga de las solicitudes relacionadas a los datos de los administradores. Dentro de las solicitudes que define están:

  - Creación de un usuario de tipo administrador.
  - Guardar la información de un administrador.
  - Obtener estadísticas de inscripción de los estudiantes.

- **Controlador de estudiante**

  Se encarga de las solicitudes relacionadas a los datos de los estudiantes. Dentro de las solicitudes que define están:

  - Creación de un usuario de tipo estudiante.
  - Creación de múltiples usuarios de tipo estudiante a partir de documentos de Microsoft Excel.
  - Guardar la información de un estudiante.
  - Guardar el plan de inscripción de un estudiante.

- **Controlador de malla**

  Se encarga de las solicitudes relacionadas a los datos de las mallas. Dentro de las solicitudes que define están:

  - Obtener una lista de todas las mallas del sistema. Retorna información general de cada malla.
  - Obtener una malla específica, junto a la información de sus módulos.
  - Crear una malla.
  - Guardar una malla.
  - Eliminar una malla.

- **Controlador de módulo**

  Se encarga de las solicitudes relacionadas a los datos de los módulos. Dentro de las solicitudes que define están:

  - Crear un módulo.
  - Obtener información de un módulo y sus prerrequisitos.
  - Guardar un módulo.
  - Eliminar un módulo.

- **Controlador de prerrequisito**

  Se encarga de las solicitudes relacionadas a los datos de los prerrequisitos. Dentro de las solicitudes que define están:

  - Crear un prerrequisito y asociarlo a un módulo.
  - Obtener información de un prerrequisito.

- Guardar un prerrequisito.
- Eliminar un prerrequisito.

- **Controlador de simulación**

  Se encarga de las solicitudes relacionadas a las simulaciones. Define una única solicitud:

  - Ejecutar simulación (encola la solicitud para ser procesada en el planificador).

4.4.2. Cola de planificador

El **planificador** es un archivo ejecutable independiente que implementa el algoritmo de resolución de planes de inscripción. Dicho ejecutable es utilizado por el servidor para retornar resultados de las simulaciones a los clientes.

Para asegurar el correcto funcionamiento del servidor se realiza un encolamiento de solicitudes para el planificador. La razón detrás de esta decisión se basa en el uso de recursos que involucra simular las inscripciones por parte del algoritmo, pudiendo ocasionar un gran uso de memoria si se realizan varias simulaciones a la vez, lo que es perjudicial para la estabilidad de la máquina física que aloja al servidor.

Para implementar el encolamiento se aprovecha la **API de concurrencia de Java**. Dado que los controladores REST de Spring utilizan un hilo de ejecución independiente por cada solicitud recibida, para implementar una cola simple, en que se procesa una sola tarea a la vez, basta con hacer un **método sincronizado**.

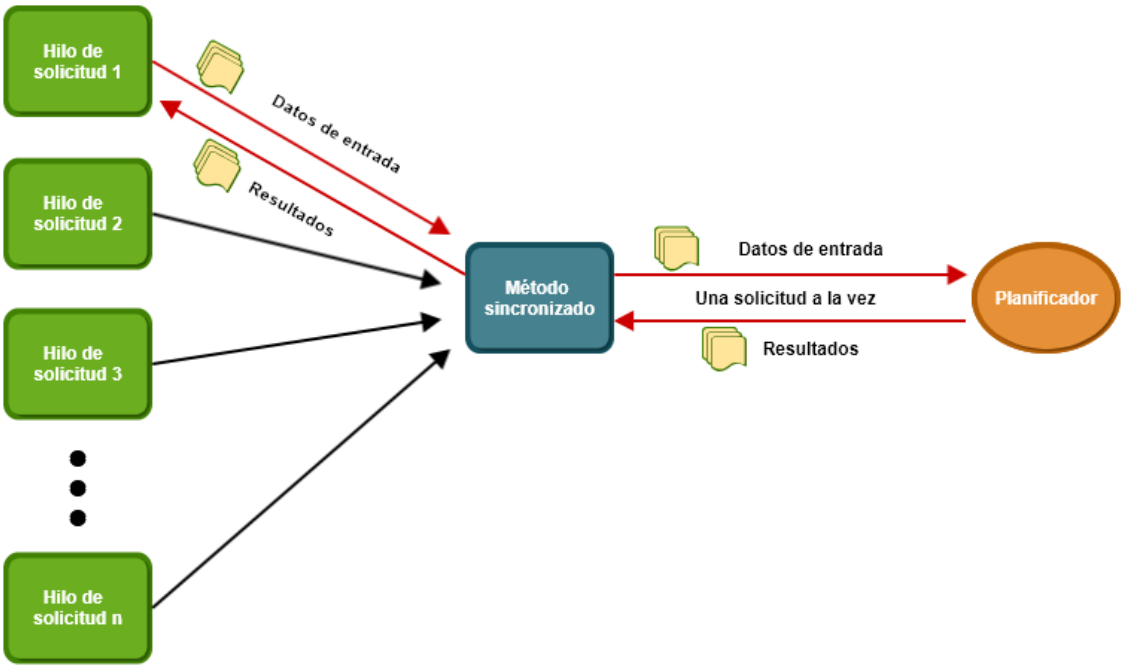


Figura 4.5: Proceso de encolamiento para el planificador

Un **método sincronizado** es un método en lenguaje Java, que se define utilizando la palabra reservada `syncronized`. Lo anterior asegura que un único hilo pueda acceder a utilizar dicho método a la vez, generando así una cola de tareas que se ejecutan una detrás de otra y nunca al mismo tiempo (ver **Figura 4.5**).

Una parte importante es asegurar que se utiliza la misma instancia de la clase que implementa el método sincronizado. Esto se realiza por medio del patrón de diseño **singleton**, es decir, se mantiene una sola instancia de la clase para toda la aplicación. Spring tiene una anotación `Service` que por defecto implementa dicho patrón, por lo que no es necesario realizar la implementación en sí, sólo basta utilizar

dicha anotación en la clase que contiene el método sincronizado.

4.4.3. Capa de acceso a datos

La capa de acceso a datos se encarga de toda la interacción entre la API y la base de datos. Para el desarrollo de esta capa se utiliza el framework **Spring Data**, particularmente las clases de **entidad** e interfaces de **repositorios**.

Una clase de **entidad** es una clase que modela una entidad. Se utiliza la anotación `Entity` para definir que una clase es de este tipo. Una entidad debe definir la forma en que los atributos van a ser tratados para ser almacenados en la base de datos, así como las respectivas relaciones entre clases (desde el punto de vista de una base de datos relacional). Entonces, implementar una clase entidad corresponde a definir todas las reglas que tendría la tabla respectiva en la base de datos.

A través de las clases de entidad, Spring Data se encarga de generar las tablas de forma automática en la base de datos correspondiente (definida en un archivo de configuración). Está funcionalidad del *framework* realiza una revisión previa en busca de las tablas correspondientes, para evitar volver a generarlas.

Un **repositorio** en Spring Data es una interfaz que declara los métodos que implementan las consultas a la base de datos. La principal característica es que el desarrollador puede despreocuparse de la implementación de la consulta en sí, ya que el framework, a partir del nombre del método y la clase entidad definida para el repositorio, define todas las consultas necesarias para retornar los datos esperados por el método. Cabe notar que Spring Data utiliza la **API de persistencia** de *Java* para la implementación que realiza en los métodos de repositorios.

La **Figura 4.6** muestra un ejemplo de implementación de un repositorio. Se tiene un repositorio base que implementa métodos para obtener una entidad a partir de un identificador y guardarla en la base de datos. También se tiene un repositorio

específico de usuario, donde se declara el método **findByEmailAddress**, es decir, obtener un usuario a partir de su correo electrónico.

```
interface MyBaseRepository<T, ID extends Serializable> extends Repository<T, ID> {
  T findOne(ID id);
  T save(T entity);
}

interface UserRepository extends MyBaseRepository<User, Long> {

  User findByEmailAddress(EmailAddress emailAddress);
}
```

Figura 4.6: Ejemplo de implementación de repositorio

Como parte de la implementación de esta capa, se generan todas las clases del **modelo de datos** en forma de entidades. Posteriormente se genera una interfaz de repositorio por cada clase entidad.

4.4.4. Capa de seguridad

La capa de seguridad es implementada utilizando el framework **Spring Security**. Dicho framework facilita los aspectos de seguridad de la aplicación, ya que basta con realizar las configuraciones respectivas, sin necesidad de realizar la implementación específica.

Principalmente se configuran los niveles de acceso para las diferentes solicitudes. Se destacan tres niveles de acceso: usuario no identificado, usuario de tipo estudiante y usuario de tipo administrador.

Un **usuario no identificado** es aquel que no se ha autenticado en el sistema. Dado que es un usuario sin identificación sólo tiene permisos para:

- Realizar el proceso de autenticación.

- Solicitar recuperar una cuenta.

Un **usuario de tipo estudiante** es aquel que se ha autenticado en el sistema con una cuenta de dicho tipo. Este usuario tiene permisos para:

- Obtener sus datos de cuenta.
- Obtener sus datos específicos de estudiante (parámetros de simulación, progreso, plan de inscripción y su fecha de modificación correspondiente).
- Editar sus datos de cuenta, incluyendo contraseña.
- Editar sus datos específicos de usuario estudiante.
- Realizar simulaciones.

Un **usuario de tipo administrador** es aquél que se ha autenticado en el sistema con una cuenta de dicho tipo. Este usuario tiene permisos para realizar **todos los tipos de solicitudes disponibles en la API**.

Control de acceso con JWT

Para controlar el acceso de los distintos tipos de usuario, se utilizan *tokens* de acceso[4]. **Json Web Token (JWT)** es un estándar basado en Json[5] para la creación de *tokens* de acceso.

La forma de trabajar con JWT corresponde a generar un *token* y relacionarlo a un usuario que se haya autenticado en el sistema. De esta forma, el cliente utiliza el *token* en cada solicitud en lugar de tener que enviar los datos de acceso del usuario todo el tiempo.

[4]**Token de Acceso** es una credencial que puede ser utilizada por una aplicación para tener acceso a ciertos recursos informáticos, como por ejemplo, una *API*.

[5]**Json** es un formato de notación de objetos para javascript. Es un formato de texto ligero para el intercambio de datos.

La relación entre el usuario y el *token* queda almacenada en el servidor, el cual tiene la capacidad de evaluar el *token* enviado en las solicitudes y determinar el usuario que la envió.

4.5. Implementación: Aplicación de cliente

La aplicación de cliente es desarrollada utilizando **Angular**. La forma de trabajar con dicho *framework* es la siguiente:

- Se componen **plantillas** HTML[6] utilizando el marcado de angular.
- Se implementan **componentes** que gestionan las plantillas, procesando los datos utilizados en éstas y haciendo uso de los diferentes servicios de la aplicación.
- Se encapsula la lógica de la aplicación en **servicios**, a los cuales se puede acceder desde cualquier punto de la aplicación.
- Se definen **módulos** que gestionan las dependencias y el orden jerárquico de la aplicación. Existe un **módulo principal** que gestiona toda la aplicación.

En el caso del proyecto, se crean componentes y plantillas para las diferentes funcionalidades a implementar. A su vez se implementa un servicio para realizar las solicitudes a la API del servidor.

La aplicación de cliente se encuentra separada en tres módulos: módulo de cuentas, módulo de administrador y módulo de estudiantes.

4.5.1. Módulo de cuentas

El módulo de cuentas contiene la lógica y vistas de todas las operaciones relacionadas a la información general y de acceso de una cuenta.

Las funcionalidades que implementa se separan en:

[6] **HyperText Markup Language** es un lenguaje de marcado para elaboración de páginas web. Está basado en XML.

- **Autenticación de un usuario en el sistema**

  La autenticación de un usuario consta de solicitar acceso utilizando las credenciales ingresadas por el usuario (ver **Figura 4.7**). Si las credenciales son válidas, se obtiene un *token* de acceso (**Sección 4.4.4**) que puede ser utilizado en las demás solicitudes.

  Una vez obtenido el acceso, se hace una redirección al módulo de administrador o estudiante según el tipo de usuario que accedió.

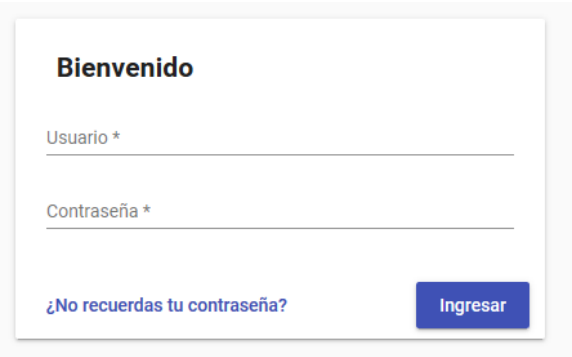


Figura 4.7: Vista de ingreso al sistema

- **Recuperación de acceso a una cuenta**

  La recuperación de una cuenta es el proceso en que un usuario, que ha perdido sus credenciales de acceso, ingresa información que lo identifica para volver a obtener acceso a su cuenta (ver **Figura 4.8**). Una vez ingresada la información se solicita la recuperación de cuenta al servidor, el cual envía un correo con nuevos datos de acceso al usuario.

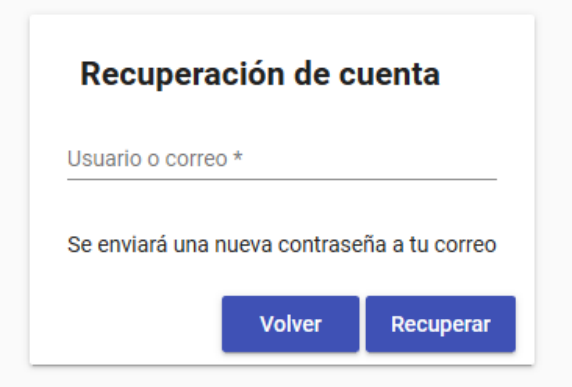


Figura 4.8: Vista de recuperación de cuenta

- **Edición de datos de una cuenta**

  Corresponde a la edición de datos generales de la cuenta por parte del usuario (ver **Figura 4.9**). Los nuevos datos son enviados mediante solicitud al servidor, el cual los actualiza en la base de datos.

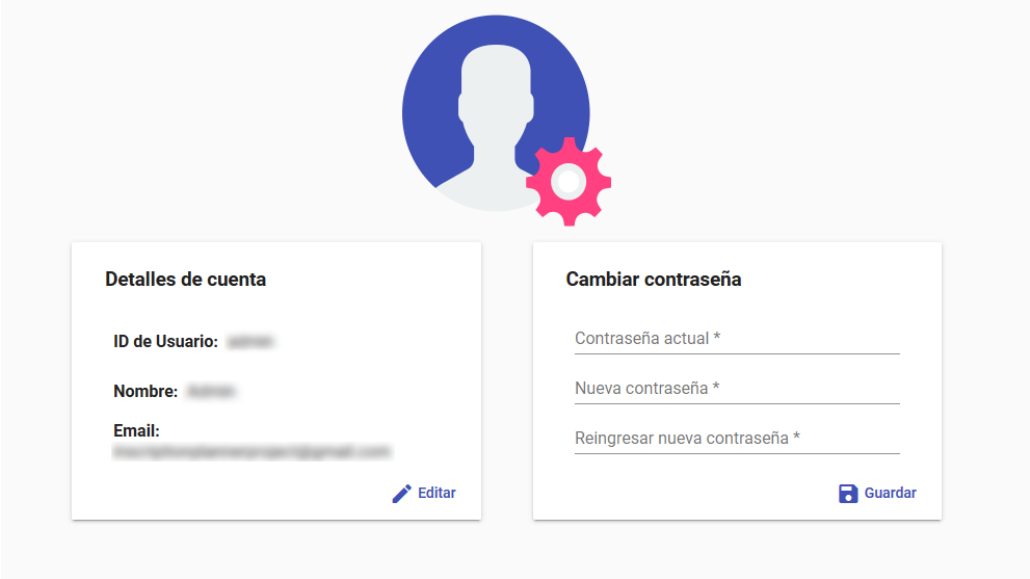


Figura 4.9: Vista de edición de cuenta

4.5.2. Módulo de administrador

El módulo de administrador contiene la lógica y vistas de todas las operaciones que realiza un administrador del sistema.

Las funcionalidades que implementa se separan en:

- **Gestión de mallas, módulos y prerrequisitos**

  La gestión de mallas, módulos y prerrequisitos consta de la creación, edición y eliminación de los datos referentes a dichas entidades (ver **Figuras 4.10** y **4.11**).

Las operaciones realizan las solicitudes apropiadas al servidor para que actualice la información.

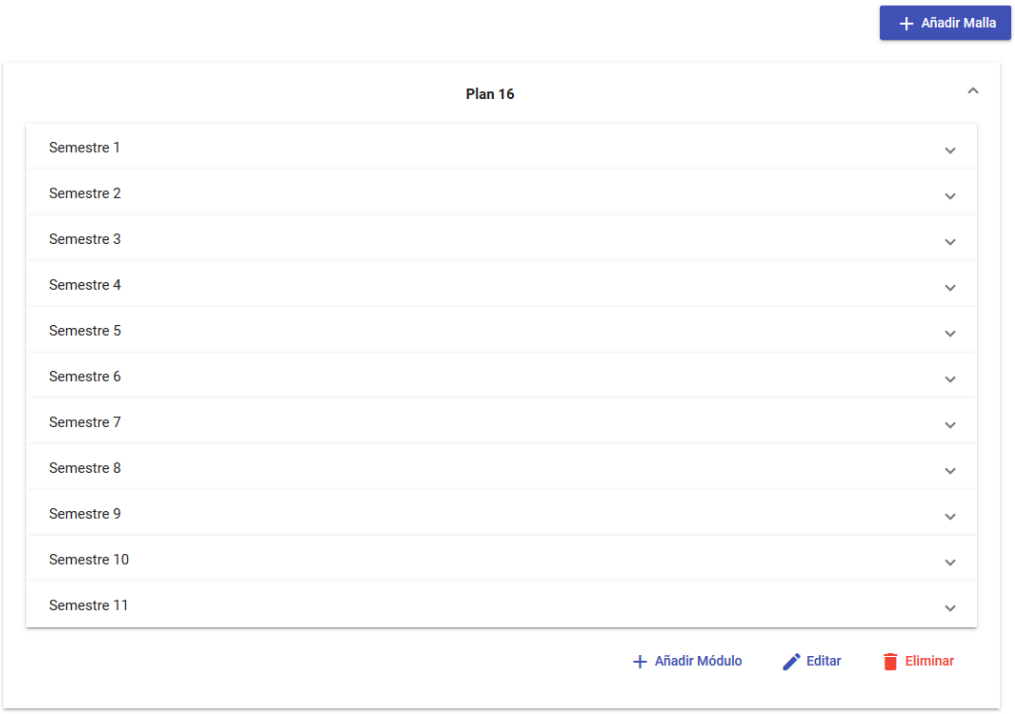


Figura 4.10: Vista de gestión de mallas

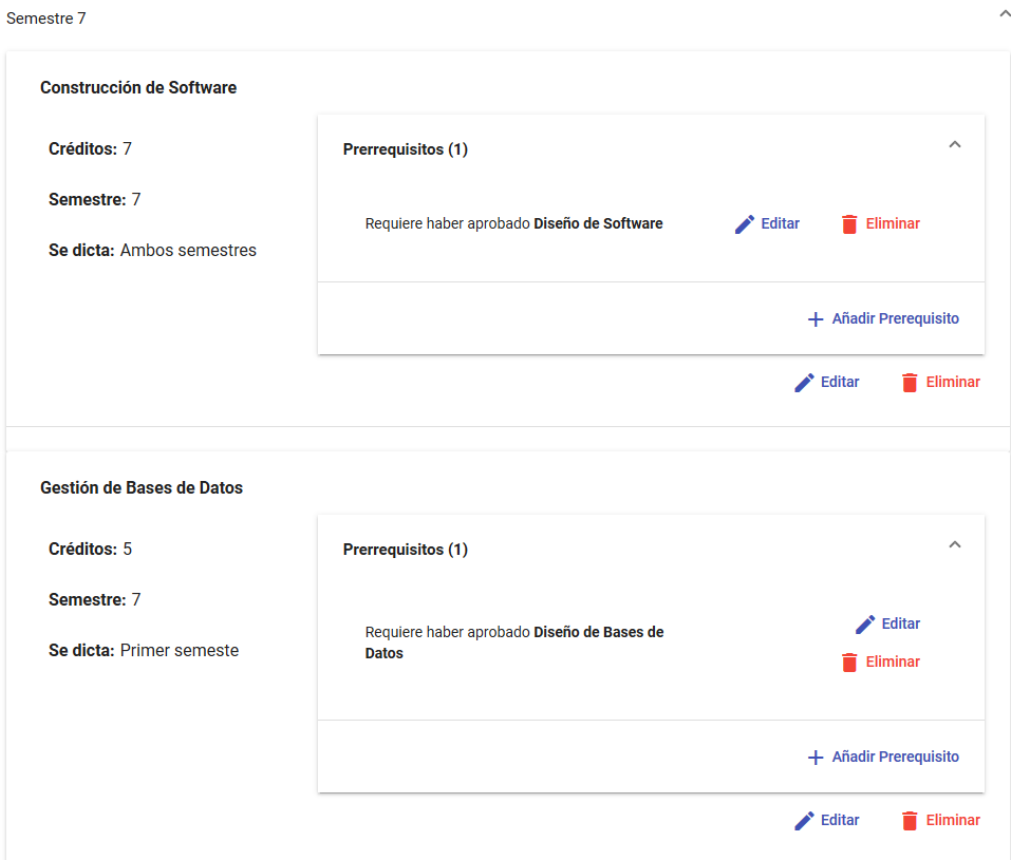


Figura 4.11: Vista de gestión de módulos y prerrequisitos

- **Gestión de usuarios**

  La gestión de usuarios permite la creación y eliminación de usuarios en el sistema. Las funcionalidades que cubre incluyen la creación y eliminación de administradores y estudiantes, creación de estudiantes a través de documentos de Microsoft Excel (ver **Figura 4.12**) y visualización de todos los usuarios del sistema (ver **Figura 4.13**).

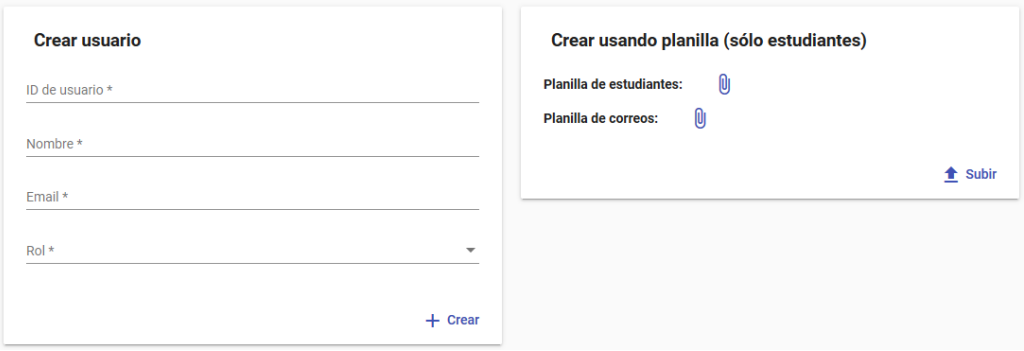


Figura 4.12: Vista de gestión de usuarios

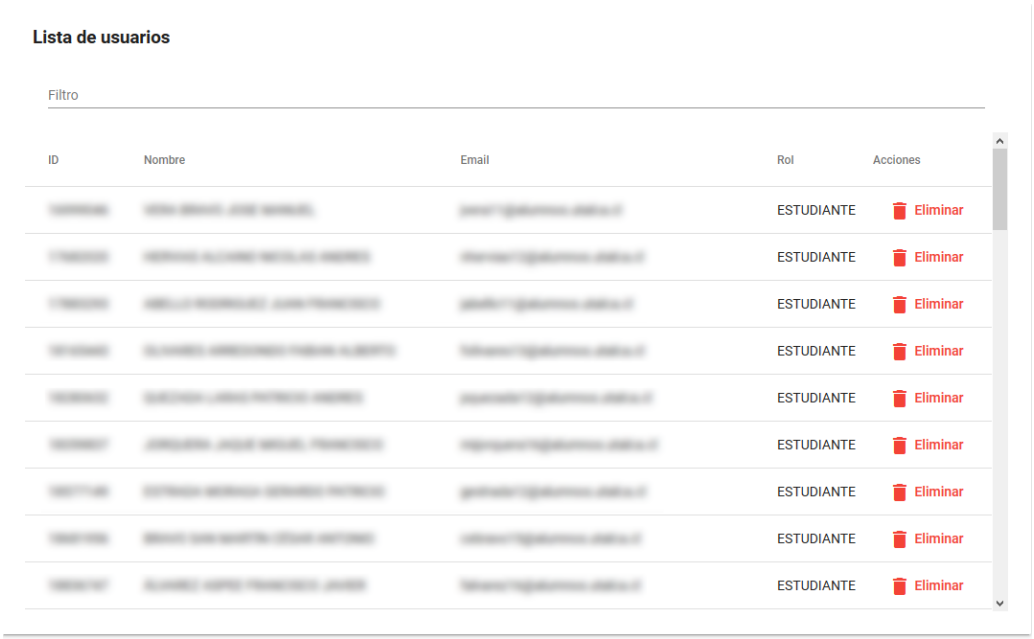


Figura 4.13: Vista de gestión de usuarios - lista de usuarios del sistema

- **Estadísticas de inscripción**

  Las estadísticas de inscripción dan a conocer las inscripciones que planean realizar los estudiantes para un período y malla en particular (ver **Figura 4.14**). Por cada módulo se representa en una barra la cantidad de estudiantes que planean inscribirlo.

  Permite filtrar datos que no hayan sido actualizados por los estudiantes permitiendo la selección de una fecha. Sólo a partir de dicha fecha los datos guardados por estudiantes serán tomados en cuenta.

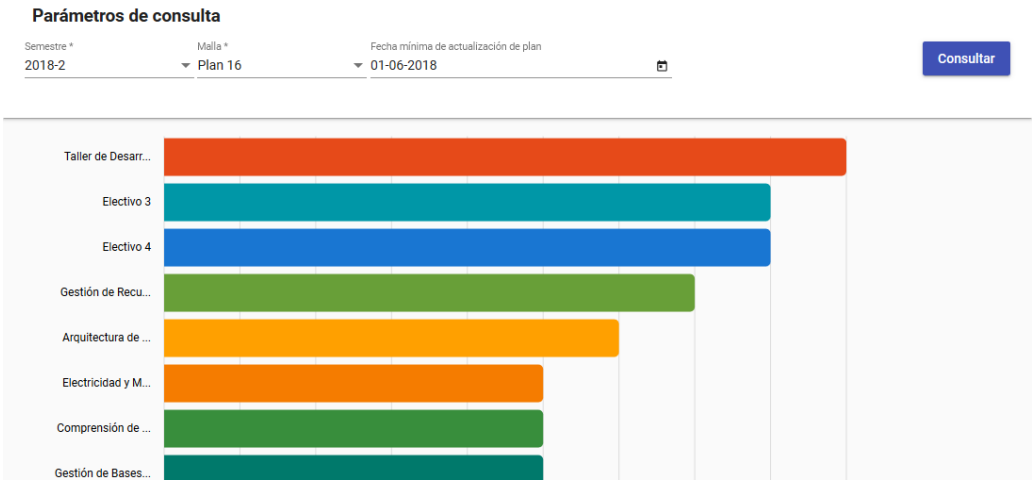


Figura 4.14: Vista de estadísticas

4.5.3. Módulo de estudiantes

El módulo de administrador contiene la lógica y vistas de todas las operaciones que realiza un estudiante en el sistema.

Las funcionalidades que implementa se separan en:

- **Configuración de parámetros de simulación**

  Permite a los estudiantes seleccionar los parámetros a utilizar en la simulación (ver **Figura 4.15**). Los parámetros corresponden a:

  - **Malla** a utilizar en la simulación.
  - **Cantidad máxima de créditos** a permitir por semestre.
  - **Cantidad máxima de módulos** a permitir por semestre.

- **Semestre actual** en que se encuentra el estudiante.

Estos parámetros corresponden con los datos de entrada que necesita el algoritmo para realizar las simulaciones.

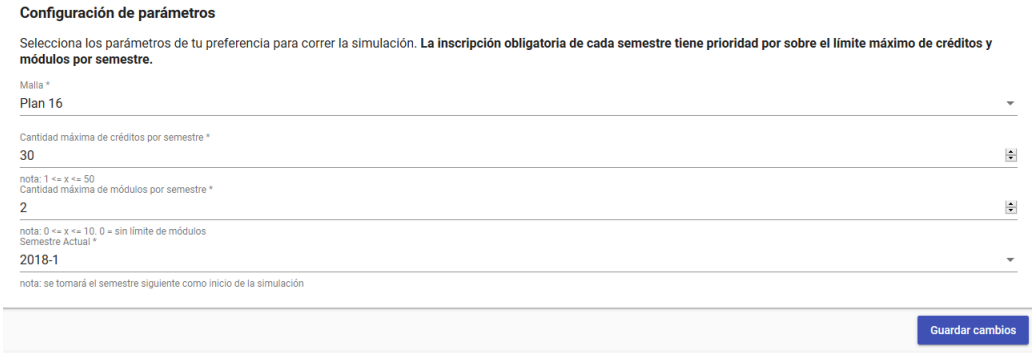


Figura 4.15: Vista de configuración de parámetros de simulación

- **Selección de módulos aprobados y reprobados**

Para realizar la simulación es necesario que los estudiantes ingresen su progreso con respecto a sus estudios. Este progreso se refleja como el conjunto de módulos que ha aprobado y reprobado un estudiante.

La **Figura 4.16** muestra la vista a través de la cual los estudiantes pueden ingresar dichos datos.

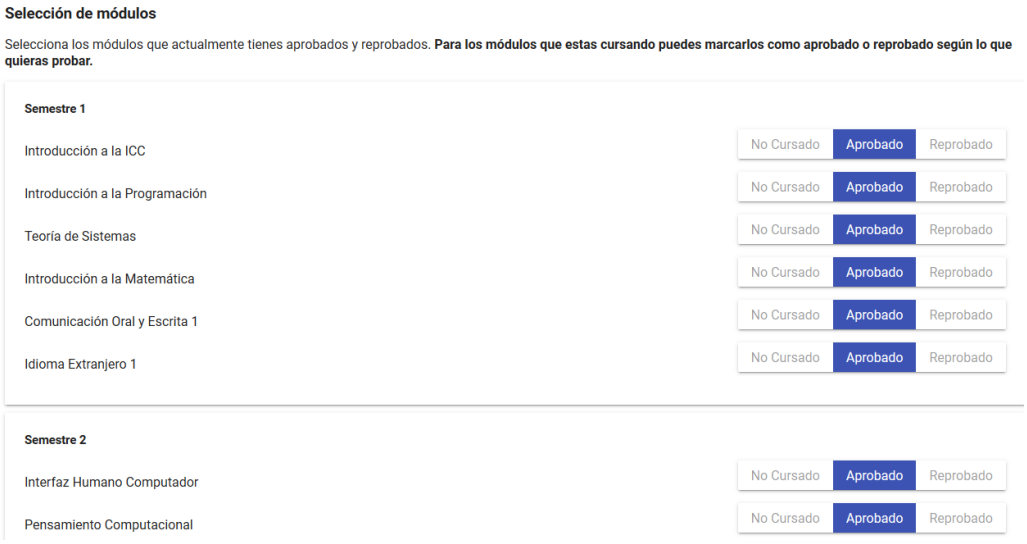


Figura 4.16: Vista de selección de módulos

- **Ejecutar simulación y mostrar resultados**

  Permite al estudiante, una vez que haya configurado los parámetros y actualizado su progreso, realizar simulaciones. La **Figura 4.17** muestra los resultados de una simulación, en la cual se utilizaron los parámetros de la **Figura 4.15**.

  Se envía una solicitud con los datos al servidor, el que encola la petición para el planificador. Una vez que el servidor obtiene respuesta desde el planificador, retorna la respuesta al cliente.

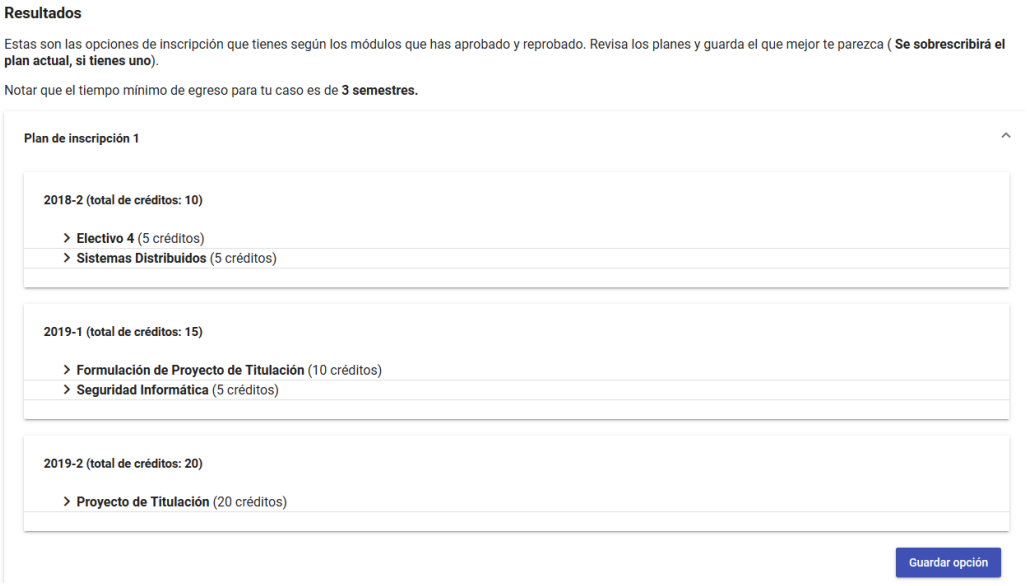


Figura 4.17: Vista de resultados de simulación.

4.5.4. Servicio de solicitudes

La finalidad del servicio de solicitudes de la aplicación cliente es permitir la comunicación entre ésta y el servidor. Dicha comunicación se realiza en forma de solicitudes REST.

El servicio de solicitudes se implementa como servicio de Angular. La particularidad de un **servicio** de Angular radica en la capacidad de acceder a él desde cualquier parte de la aplicación, a diferencia de los **componentes**, los que sólo pueden ser utilizados en un **módulo** en particular.

Angular provee la implementación de un módulo para solicitudes REST mediante los protocolos http/https, por lo que se utiliza para realizar las diferentes llamadas

a los métodos de controlador en el servidor. Particularmente, se implementa una función de consulta para cada método de los controladores de API del servidor, por lo que, si un componente necesita solicitar al servidor cualquier dato, hace uso de dichas funciones.

Con la finalidad de validar el nivel de acceso que tiene un usuario, se utilizan los *tokens* mencionados en la **Sección 4.4.4**. Un **interceptor** de Angular permite modificar datos de una solicitud antes de que sea enviada. En el caso de la aplicación de cliente, se modifica la cabecera de cada solicitud para añadir el *token* de acceso del usuario, de esta forma el servidor puede verificar y aceptar solicitudes.

5. Evaluación y resultados

Una parte fundamental del proyecto es evaluar su capacidad de resolver el problema y el cumplimiento de los objetivos planteados. Es por esto que se definen dos instancias de evaluación, las cuales buscan conocer la eficacia de la solución a través de los interesados.

Este capítulo detalla los resultados obtenidos al evaluar el sistema en dichas instancias. En primer lugar, se expone el caso real de un estudiante al utilizar la herramienta, con el motivo de mostrar al lector su funcionamiento. Posteriormente, se dan a conocer los resultados de la primera instancia de evaluación del sistema, la cual fue realizada por estudiantes de la carrera de ICC a través de una encuesta. Para finalizar, se evidencia la segunda instancia, que corresponde a la aprobación de la solución por parte de la Escuela de dicha carrera.

5.1. Demostración de caso real

Con la finalidad de mostrar al lector el funcionamiento de la herramienta y cómo ayuda a los estudiantes a decidir su inscripción, se realiza a continuación la demostración de un caso real de un estudiante de ingreso 2016 que actualmente pertenece al Plan 16 de ICC.

La **Figura 5.1** muestra la configuración utilizada por el estudiante al realizar su simulación. Dicha configuración limita al algoritmo a simular inscripciones de hasta 31 créditos y con un máximo de 7 módulos por semestre.

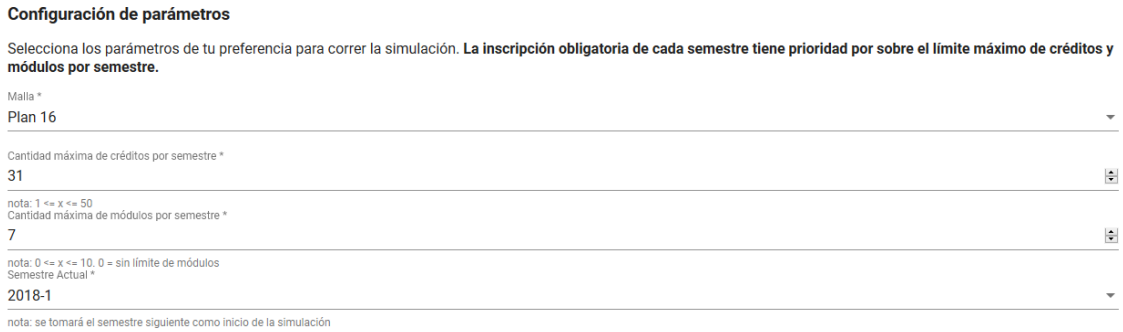


Figura 5.1: Configuración utilizada por el estudiante

La situación del estudiante con respecto a sus módulos se muestran en la **Figura 5.2**, donde los módulos aprobados aparecen con una marca verde. El estudiante ingresó la información en la herramienta, utilizando la vista de *Selección de módulos aprobados y reprobados*. Cabe destacar que el estudiante tiene la opción de marcar los módulos que está cursando actualmente como aprobados o reprobados, con la finalidad de que pueda realizar simulaciones en los distintos casos que se pueden dar al finalizar el semestre.

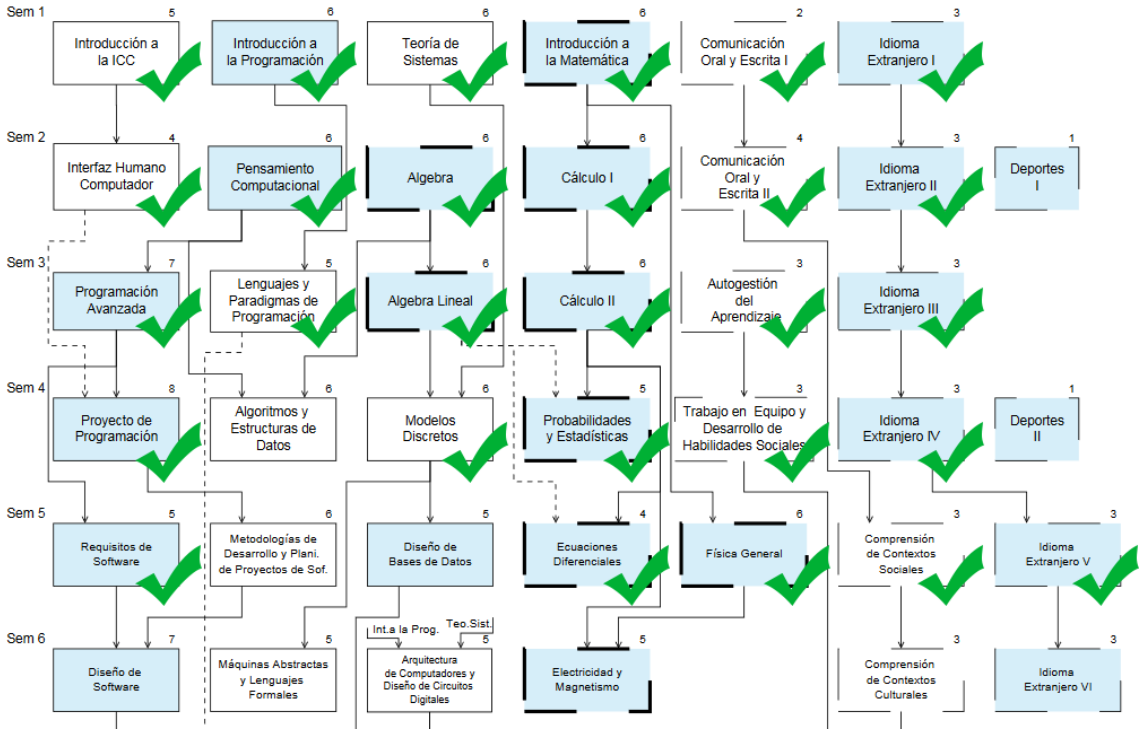


Figura 5.2: Situación del estudiante con respecto a sus módulos

En este caso se da la particularidad de que el módulo *Metodologías de Desarrollo y Planificación de Proyectos de Software* no ha sido inscrito, siendo que corresponde al quinto semestre de la carrera, período actual que cursa el estudiante. En la **Sección 2.1.4.1** se da a conocer un ejemplo, donde el módulo mencionado ocasiona un retraso de un semestre en caso de no ser aprobado en el semestre en que corresponde. Esto se refleja en los resultados de la simulación del estudiante (ver **Figura 5.3**).

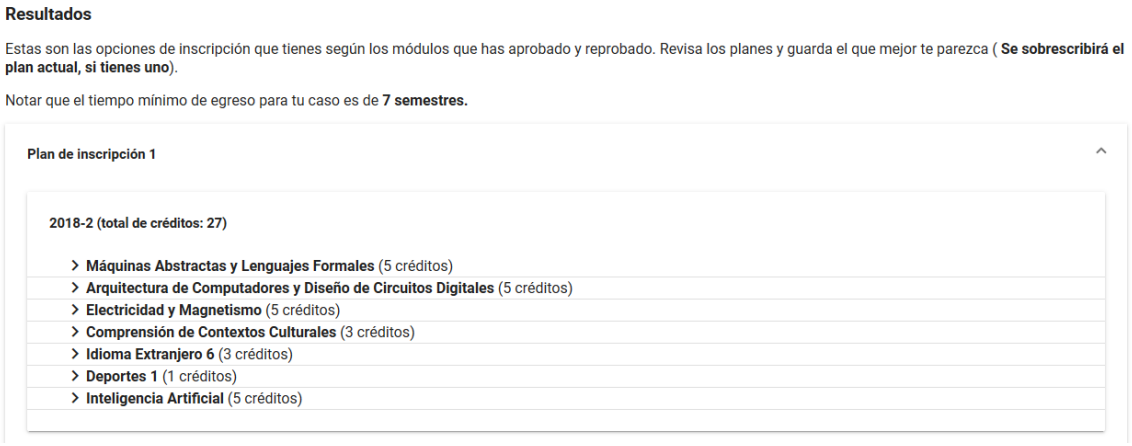


Figura 5.3: Resultados de la simulación

El tiempo mínimo de egreso para el caso del estudiante es de **siete semestres** a partir del segundo semestre del presente año (2018-2), es decir, estaría en condiciones de egresar el **segundo semestre del año 2021**. Lo anterior equivale efectivamente a un retraso de un semestre en el egreso.

Este caso destaca la importancia para los estudiantes de conocer y evitar reprobar o atrasar la inscripción de módulos que se encuentran en las líneas críticas, ya que en muchos casos el retraso es inevitable.

La herramienta entregó al estudiante siete planes de inscripción, que lo llevan a egresar el segundo semestre del año 2021, el menor tiempo de egreso posible. Las **Figuras 5.4**, **5.5** y **5.6** muestran la primera sugerencia de inscripción, semestre a semestre, dada por la herramienta al estudiante.

Figura 5.4: Sugerencia de inscripción (parte 1 de 3)

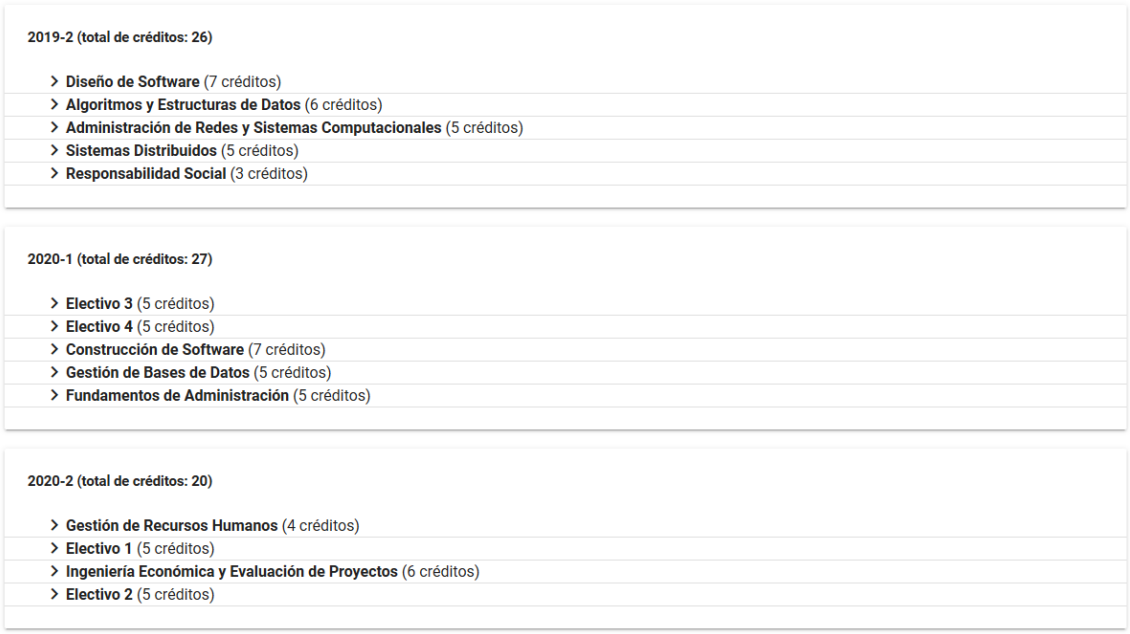


Figura 5.5: Sugerencia de inscripción (parte 2 de 3)

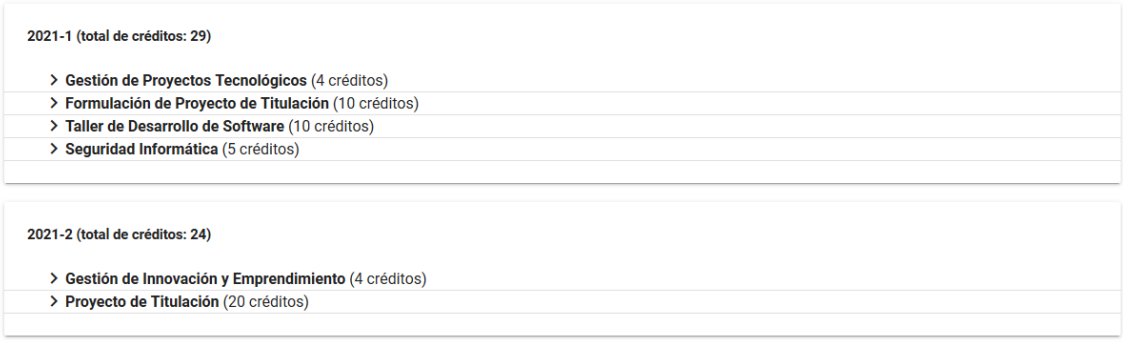


Figura 5.6: Sugerencia de inscripción (parte 3 de 3)

5.2. Resultados de la encuesta realizada a estudiantes

La encuesta fue realizada a una muestra de 28 estudiantes, los que equivalen aproximadamente al 10 % de los estudiantes matriculados de ICC, todos pertenecientes al plan 16, distribuidos entre los módulos de *Requisitos de Software* y *Construcción de Software*. La selección de esta muestra se debe a varias razones:

- Los módulos mencionados están en los semestres 5 y 7 respectivamente en la malla, por lo que se puede esperar que los estudiantes de dichos módulos presenten una mayor variación en cuanto a sus situaciones académicas (progreso en la malla).
- Para fines de evaluación, no es bueno utilizar como muestra a estudiantes en los primeros años de carrera, ya que no presentan situaciones muy diferentes unos de otros en cuanto a avance curricular. Lo anterior evita que se pruebe el algoritmo de forma extensiva y con casos variados.

- A su vez la situación de estudiantes en sus últimos años es, por lo general, simple de visualizar (les quedan pocos módulos por aprobar), lo que hace menos necesario para ellos un sistema como éste.

Previo a realizar la encuesta, los estudiantes tuvieron la oportunidad de probar el sistema, como un usuario de tipo estudiante.

La encuesta realizada cuenta con tres secciones. A continuación, se da a conocer el resumen de los resultados de cada una de ellas.

- **Sección de preguntas generales sobre el sistema**

  Esta sección comprende preguntas generales sobre el sistema, enfocándose en el valor que le otorgan los estudiantes a la solución.

  Los estudiantes expresan que existe la necesidad de una herramienta que los ayude a decidir la inscripción de sus módulos. Asimismo expresan que la solución implementada ayuda a cumplir con ese objetivo. Además, hay cierta tendencia en las respuestas que indican que el sistema podría ayudar a reducir el tiempo de egreso de los estudiantes que la utilicen, ya que tomarán decisiones estando informados.

  Más de un 90 % de los alumnos encuestados afirma que se apoyarían en la aplicación para ayudarlos a decidir la inscripción de módulos. Finalmente, los resultados expresan que el sistema puede tener entre un mediano a un gran impacto en las decisiones tomadas por los estudiantes, respecto a la inscripción.

- **Sección de preguntas sobre las funcionalidades del sistema**

  Esta sección comprende preguntas respecto a las funcionalidades del sistema.

Alrededor del 85 % de los encuestados afirma que las funcionalidades son suficientes para que la aplicación sea de utilidad para ellos. No obstante, también dan a conocer que los resultados no siempre son fiables. Con respecto a esto último es necesario destacar que al momento de las pruebas el algoritmo tenía algunos errores en su funcionamiento, los cuales fueron resueltos posteriormente. Los estudiantes que probaron el sistema en una fecha posterior a los arreglos no presentaron problemas con sus resultados.

Los resultados también dan a conocer que los estudiantes, mientras realizaban las pruebas de la aplicación encontraron no más de tres errores en el sistema cada uno. Los errores encontrados fueron reportados por los mismos estudiantes y solucionados posteriormente.

- **Sección de preguntas sobre usabilidad e interfaz de usuario**

  Esta sección comprende preguntas sobre la percepción de los estudiantes sobre la usabilidad de la aplicación y su interfaz de usuario.

  El 85 % de los estudiantes afirman que la aplicación es fácil de usar y no requiere instrucciones previas. También dan a conocer que están de acuerdo con la distribución de la mayoría de los elementos en pantalla.

  Más de un 90 % de los encuestados afirman que los controles para seleccionar sus módulos aprobados y reprobados, así como también la forma de realizar y mostrar los resultados de las simulaciones es adecuada y fácil de entender.

Cabe señalar que además de las preguntas fijas, se dio espacio en la encuesta para que los estudiantes expresaran su opinión y dieran ideas. Muchas de estas ideas fueron tomadas en consideración y serán posibles trabajos futuros de la aplicación.

Adicionalmente, las opiniones recibidas fueron positivas y refuerzan la idea de que herramientas como ésta, que apoyen a los estudiantes en diferentes ámbitos durante su paso por la casa de estudio, son extremadamente necesarias.

Si el lector desea conocer los resultados en mayor detalle, puede consultar el **Anexo C**.

5.3. Aprobación por parte de la Dirección de Escuela de ICC

Dada la metodología de desarrollo de software utilizada, durante el desarrollo se realizaron demostraciones del sistema a medida que se avanzaba en su construcción, donde la profesora *Ruth Garrido*, Directora de Escuela de la carrera de ICC, en su calidad de interesada, pudo observar el funcionamiento del sistema, tanto en las funcionalidades de administrador como de estudiante.

En cada demostración (*Sprint Review*), la profesora se mostró conforme con los resultados obtenidos, destacando la utilidad del sistema para los estudiantes y realizando también observaciones. A partir de estas observaciones se generaron nuevos items para el *Backlog* del proyecto, formando parte de la implementación final.

Por medio de un documento, la profesora *Ruth Garrido* da a conocer su aprobación del sistema implementado. El documento puede ser consultado en el **Anexo D**.

5.4. Resultados finales del proyecto

El resultado global del proyecto se mide en función del cumplimiento de cada uno de los objetivos propuestos, donde los objetivos específicos definen el cumplimiento del objetivo general. En relación con lo anterior se tienen los siguientes resultados para los objetivos específicos:

- **Diseñar e implementar un algoritmo para realizar simulaciones de la inscripción de módulos para un estudiante, dando a conocer las posibles opciones de inscripción**

  Como se detalla en la **Sección 4.3**, se realizó el diseño e implementación del algoritmo, realizando las pruebas pertinentes y verificando los resultados obtenidos de las ejecuciones, tanto por el autor como por los interesados.

- **Diseñar e implementar una interfaz gráfica que permita configurar los parámetros, ejecutar y mostrar los resultados obtenidos del algoritmo**

  Como se detalla en las **Secciones 4.2, 4.4 y 4.5**, se realizó el diseño e implementación de la interfaz gráfica por medio de la cual los estudiantes pueden hacer uso del algoritmo y obtener sus resultados. La verificación de la interfaz fue realizada por los mismos estudiantes y la profesora *Ruth Garrido*, como ya se mencionó.

- **Implementar una herramienta para cargar información de los módulos de una malla y de los estudiantes**

  Como se detalla en las **Secciones 4.2, 4.4 y 4.5**, la implementación de la herramienta fue realizada en conjunto con la interfaz gráfica del objetivo anterior. La herramienta permite a un administrador del sistema ingresar todos los datos requeridos con respecto a una malla, incluyendo módulos y prerrequisitos. A su vez, los estudiantes pueden hacer ingreso de su avance académico (módulos aprobados y reprobados) en el módulo de estudiantes de la aplicación. La verificación del objetivo y correcto funcionamiento de la herramienta fue realizada por los mismos estudiantes y la profesora *Ruth Garrido*.

- **Obtener la aprobación del sistema implementado y retroalimentación por parte de los interesados**

  Como se detalla en las secciones anteriores del presente capítulo, se realizó una encuesta a una muestra de estudiantes de la carrera de ICC. En dicha encuesta se expresa la aprobación de la aplicación por parte de la gran mayoría de los encuestados, obteniendo también retroalimentación de su parte. Además, Se destaca, a través de un documento firmado, la aprobación del sistema por parte de la profesora *Ruth Garrido*, en su calidad de Directora de Escuela de la carrera de ICC.

Conociendo el cumplimiento de cada uno de los objetivos específicos, se puede entonces inferir el cumplimiento del objetivo general: **Desarrollar un sistema de software para dar apoyo en la toma de decisiones durante el proceso de inscripción de módulos**.

6. Conclusiones

Por medio de este documento se dan a conocer los detalles del proceso completo que involucra el desarrollo de este *Proyecto de Titulación*. En primera instancia se reconocen los conceptos y variables que entran en juego a la hora de inscribir módulos para los estudiantes, determinando así la problemática que busca resolver este proyecto: **los estudiantes no poseen herramientas en las que apoyarse para tomar buenas decisiones en la inscripción de módulos**. La dificultad para determinar las líneas críticas es uno de los principales motivos de esta necesidad.

Ante la problemática, se propone una solución de carácter informático, detallando el diseño de un algoritmo capaz de determinar planes de inscripción que reducen el riesgo de las líneas críticas. Asimismo se especifica el diseño e implementación de un sistema que hace uso de dicho algoritmo, siendo el sistema el enlace entre el algoritmo y los usuarios.

Para los estudiantes de la carrera de Ingeniería Civil en Computación, la herramienta puede servir de guía y ayudar a evitar errores durante el proceso de inscripción. Dentro de las consecuencias de equivocarse en la inscripción se encuentra el posible retraso en el egreso de los estudiantes, lo que lleva consigo un alto gasto económico para el afectado y su familia, además del riesgo de perder beneficios, si es el caso.

La evaluación realizada por los estudiantes indica que están en conocimiento de la necesidad de herramientas de este tipo. A su vez deja en manifiesto el éxito del sistema, destacando entre otras cosas, la utilidad e impacto que tiene para las decisiones que se toman con respecto a la inscripción de módulos.

Para la Escuela de la carrera de Ingeniería Civil en Computación, la herramienta puede ser utilizada en la planificación de semestres, conociendo de esta manera las tendencias de los estudiantes en cuanto a los módulos que planean inscribir en cada período. Es posible, entre otras cosas, asegurar los cupos de cada módulo, teniendo datos que respaldan las decisiones tomadas. En este ámbito, se destaca la aprobación del sistema por parte de la Directora de Escuela de dicha carrera.

Para la Universidad, la implementación de esta herramienta es una oportunidad de crecer en un ámbito que los estudiantes muchas veces sienten que se ha dejado de lado: mejorar y/o aumentar la oferta de herramientas para simplificar las tareas involucradas en los procesos de la vida universitaria.

Se destaca además, el cumplimiento de los objetivos propuestos, a través de:

- El diseño e implementación del algoritmo que simula y da a conocer opciones de inscripción al estudiante.
- Un sistema cuya interfaz permite configurar parámetros y ejecutar simulaciones para un estudiante.
- La carga de información de mallas y progreso académico de un estudiante utilizando el sistema antes descrito.
- La aprobación del sistema de parte de los interesados.

Para finalizar, este proyecto entrega un sistema funcional que puede ser utilizado tanto por estudiantes como por miembros de la Escuela de ICC. En este marco se destaca el trabajo a futuro que es posible realizar con este software:

- Implementar algunas de las ideas sugeridas por los estudiantes en la encuesta realizada.
- Lanzar el sistema como una herramienta oficial de la carrera.
- Extender el sistema para que sea funcional a nivel de Universidad, esto es, utilizable por cualquier carrera.
- Añadir alguna forma de mantener la tabla de hash del algoritmo entre ejecuciones, lo que mejoraría aún más los tiempos de ejecución.
- Modificar el algoritmo para que soporte períodos anuales y trimestrales. Si bien la implementación actual sirve para la carrera de ICC, no funciona para casos de mallas con módulos anuales o trimestrales, los cuales pueden existir en otras carreras.
- Extender el algoritmo o diseñar uno nuevo que trabaje con la asignación de horarios. Esto sería un gran complemento para la actual implementación.
- El software implementado o parte de éste puede ser integrado a otro sistema más complejo.
- Existiendo muchos caminos a través de los cuales el software puede crecer, éste puede ser tomado como base para futuros proyectos de título.

Bibliografía

[1] Angular Features. `https://angular.io/features`. Online. Consultado el 16 de junio de 2018.

[2] MySQL: The most popular Open-Source Database. `https://www.oracle.com/mysql/index.html`. Online. Consultado el 14 de julio de 2018.

[3] Sistema Europeo de Transferencia y Acumulación de Créditos (ECTS). `https://ec.europa.eu/education/resources/european-credit-transfer-accumulation-system_es`. Online. Consultado el 10 de septiembre de 2017.

[4] Spring Framework. `https://spring.io/projects/spring-framework`. Online. Consultado el 16 de junio de 2018.

[5] The IoC container. `https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/beans.html`. Online. Consultado el 16 de julio de 2018.

[6] Tapio Auvinen, Juha Paavola, and Juha Hartikainen. STOPS: A Graph-based Study Planning and Curriculum Development Tool. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, Koli Calling '14, pages 25–34, New York, NY, USA, 2014. ACM.

[7] Nipaporn Chanamarn and Kreangsak Tamee. Enhancing Efficient Study Plan for Student with Machine Learning Techniques. *International Journal of Modern Education and Computer Science(IJMECS)*, 9(3):1–9, 2017.

[8] CRUCH. Guía practica para la instalación del SCT-Chile. `http://sct-chile.consejoderectores.cl/documentos_WEB/Sistema_de_creditos_transferibles/documentos_generales/1.Guia_Practica_SCT_Chile.pdf`. Online. Consultado el 18 de mayo de 2018.

[9] Vicerrectoría de Docencia de Pregrado de la Universidad de Talca. Modelo Educativo de la Universidad de Talca. `http://fen.utalca.cl/pdf/Modelo_Educativo_Universidad_de_Talca.pdf`. Online. Consultado el 10 de septiembre de 2017.

[10] Universidad de Talca. Reglamento de régimen de estudios Universidad de Talca. `http://www.utalca.cl/medios/utalca2010/alumnos/RU-072-2017_RegimenEstudios.pdf`. Online. Consultado el 16 de diciembre de 2017.

[11] Steven Halim and Felix Halim. *Competitive Programming 3: The New Lower Bound of Programming Contests*. Jun 2013.

[12] Aswin Pranam. *Product Management Essentials: Tools and Techniques for Becoming an Effective Technical Product Manager*. Apress, 1 edition, 2018.

[13] L. Rising and N. S. Janoff. The Scrum software development process for small teams. *IEEE Software*, 17(4):26–32, Jul 2000.

[14] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. Nov 2017.

Glosario

Algoritmo Heurístico: Es un algoritmo no determinista que sacrifica uno o varios factores para ganar en otros. Por ejemplo, puede tener una menor precisión en sus soluciones, pero reduce el tiempo que toma calcularlas.

API: Corresponde a un conjunto de subrutinas, funciones y métodos que proporcionan una capa abstracta de comunicación entre componentes de software.

Concurrencia: Es una propiedad de los sistemas en la cual los procesos de un cómputo se hacen simultáneamente, y pueden interactuar entre ellos.

Cumulative Grade Point Average: Es una medida de rendimiento académico obtenida de sumar las ponderaciones de todas las notas obtenidas por el estudiante con respecto a la cantidad de créditos respectivos de los módulos.

Diagrama de clases: Es un tipo de diagrama UML (Lenguaje Unificado de Modelado) que describe la estructura de un sistema mostrando las clases de éste, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

Egreso: Finalización de los estudios medios o superiores de una persona, y su titulación.

Hilo de ejecución: Es una secuencia de tareas encadenadas muy pequeña que puede ser ejecutada por un sistema operativo. Se destaca el hecho de poder ser ejecutados de forma simultánea o intercalada (dependiendo del procesador) con otros hilos.

Html: Es un lenguaje de marcado para elaboración de páginas web. Ésta basado en XML.

ICC: Abreviación de *Ingeniería Civil en Computación.*

Interesados: Corresponde a la parte interesada y se refiere a todas aquellas personas u organizaciones que tienen derechos, acciones o interés con respecto al sistema o sus propiedades, satisfaciendo este último las necesidades o expectativas de los primeros.

Inversion de control: Es un patrón de diseño de software en que es el framework quien toma el control del flujo del código para tareas de alto nivel, mientras que las tareas de bajo nivel siguen siendo implementadas por el código del usuario.

Json: Es un formato de notación de objetos para javascript. Es un formato de texto ligero para el intercambio de datos.

Machine Learning: Es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente.

Microsoft Excel: Es una aplicación de hojas de cálculo que forma parte de la suite de oficina Microsoft Office, desarrollada por la empresa Microsoft. Es una aplicación utilizada en tareas financieras y contables, con fórmulas, gráficos y un lenguaje de programación.

Plan de inscripción: Plan semestre a semestre de los módulos a inscribir. Este plan abarca desde el semestre actual de carrera en que se encuentra el estudiante hasta el egreso de este.

REST: REpresentational State Transfer, es un estilo de arquitectura que provee estándares de comunicación entre sistemas computacionales en una red. Se usa particularmente en arquitecturas de tipo cliente-servidor, donde el cliente realiza solicitudes al servidor y éste último procesa y responde dichas solicitudes.

Sistema curricular: Conjunto de instrumentos curriculares que actúan de manera articulada y sistemática para facilitar la enseñanza.

Token de Acceso: Es una credencial que puede ser utilizada por una aplicación para tener acceso a ciertos recursos informáticos, como por ejemplo, una *API*.

Web Responsive Design: Es un diseño web capaz de redistribuir los elementos en pantalla para que se adapten a todo tipo de dispositivos.

ANEXOS

A. Pruebas del algoritmo

Casos de prueba para algoritmos utilizando la malla del plan 16 (**Anexo B**).

Estudiante de 1^{er} año

- **Semestre actual:** 2018-1
- **Módulos aprobados:**
  - Introducción a la ICC
  - Teoría de Sistemas
  - Comunicación Oral y Escrita 1
  - Idioma Extranjero 1
- **Módulos reprobados:**
  - Introducción a la Programación
  - Introducción a la Matemática

Estudiante de 2^{do} año

- **Semestre actual:** 2018-1
- **Módulos aprobados:**

- Primer semestre completo
- Interfaz Humano Computador
- Pensamiento Computacional
- Algebra
- Comunicación Oral y Escrita 2
- Idioma Extranjero 2
- Deportes 1
- Programación Avanzada
- Lenguajes y Paradigmas de Programación
- Autogestión del Aprendizaje
- Idioma Extranjero 3

▪ **Módulos reprobados:**

- Cálculo 1
- Algebra Lineal

Estudiante de 3^{er} año

▪ **Semestre actual:** 2018-1

▪ **Módulos aprobados:**

- Hasta tercer semestre completo
- Proyecto de Programación
- Algoritmos y Estructuras de Datos
- Modelos Discretos
- Trabajo en Equipo y Desarrollo de Habilidades Sociales
- Idioma Extranjero 4

- Deportes 2
- Requisitos de Software
- Metodologías de Desarrollo y Planif. de Proyectos de Sof.
- Diseño de Bases de Datos
- Comprensión de Contextos Sociales

- **Módulos reprobados:**
  - Probabilidades y Estadísticas
  - Ecuaciones Diferenciales
  - Física General

Estudiante de 4^{to} año

- **Semestre actual:** 2018-1
- **Módulos aprobados:**
  - Hasta quinto semestre completo
  - Diseño de Software
  - Arquitectura de Computadores y Diseño de Circuitos Digitales
  - Comprensión de Contextos Culturales
  - Idioma Extranjero 6
  - Gestión de Bases de Datos
  - Redes de Computadores
  - Fundamentos de Administración
  - Etica y Responsabilidad Social
- **Módulos reprobados:**
  - Máquinas Abstractas y Lenguajes Formales

- Electricidad y Magnetismo
- Sistemas Operativo

Estudiante de 5^{to} año

- **Semestre actual:** 2018-1
- **Módulos aprobados:**
  - Hasta octavo semestre completo
  - Gestión de Recursos Humanos
- **Módulos reprobados:**
  - Taller de Desarrollo de Software
  - Gestión de Proyectos Tecnológicos

Estudiante de 6^{to} año

- **Semestre actual:** 2018-2
- **Módulos aprobados:**
  - Hasta quinto semestre completo
  - Diseño de Software
  - Máquinas Abstractas y Lenguajes Formales
  - Arquitectura de Computadores y Diseño de Circuitos Digitales
  - Comprensión de Contextos Culturales
  - Idioma Extranjero 6
  - Construcción de Software
  - Gestión de Bases de Datos
  - Sistemas Operativos

- Redes de Computadores
- Fundamentos de Administración
- Etica y Responsabilidad Social
- Electivo 1
- Inteligencia Artificial
- Administración de Redes y Sistemas Computacionales
- Sistemas Distribuidos
- Ingeniería Económica y Evaluación de Proyectos
- Responsabilidad Social
- Taller de Desarrollo de Software
- Gestión de Recursos Humanos
- Gestión de Innovación y Emprendimiento

- **Módulos reprobados:**
  - Electricidad y Magnetismo
  - Gestión de Proyectos Tecnológicos
  - Formulación de Proyecto de Titulación

B. Malla de ICC Plan 16

La **Figura B.1** muestra una representación de la malla del Plan 16 de la carrera de Ingeniería Civil en Computación.

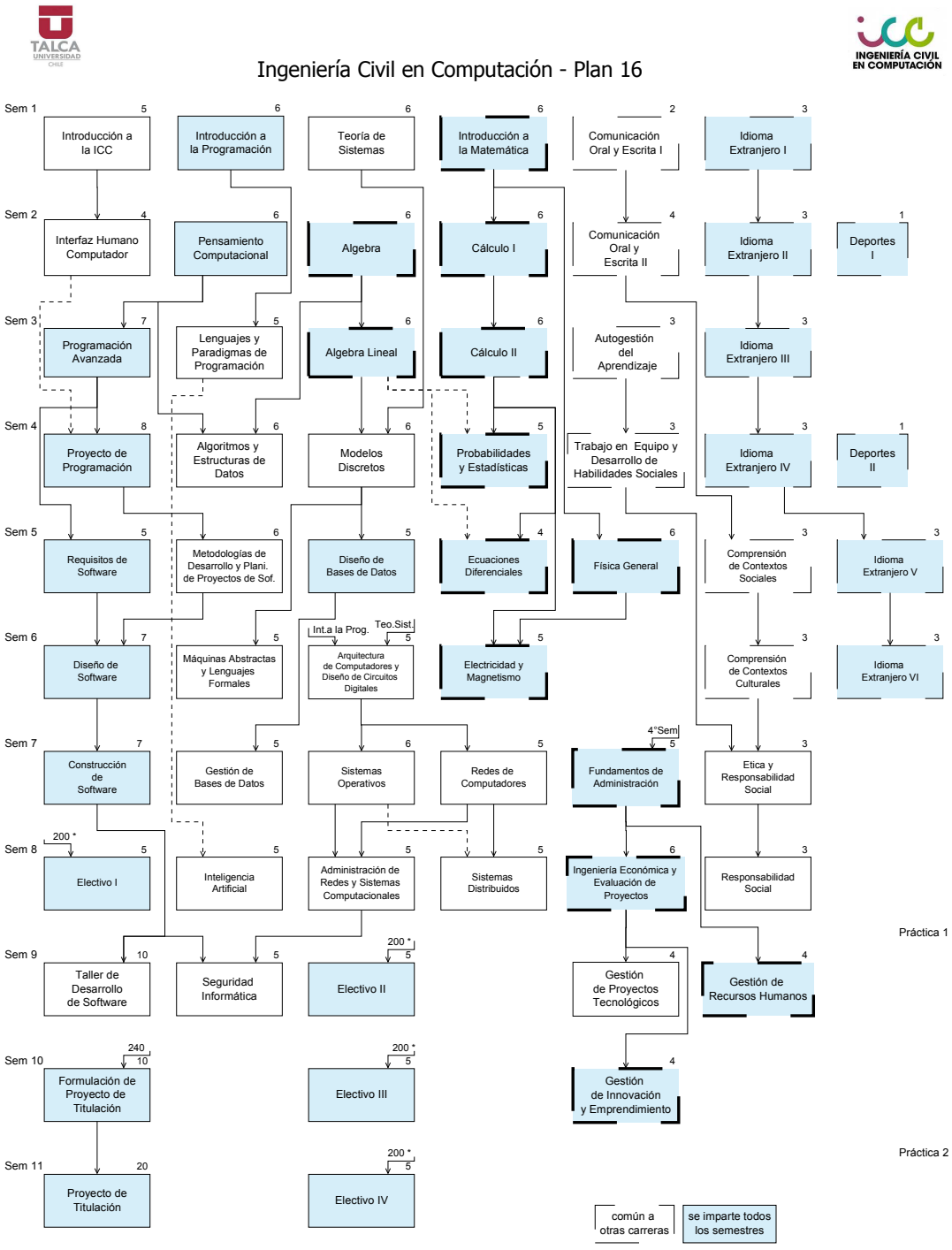


Figura B.1: Malla de plan 16 de ICC

C. Encuesta a estudiantes

Este anexo da a conocer el detalle de las respuestas de la encuesta realizada a 28 estudiantes en los módulos de *Requisitos de Software* y *Construcción de Software*.

Sección de preguntas generales sobre el sistema

Marca la opción que mejor corresponde a tu opinión en cada pregunta

1. **¿Crees que es necesaria una herramienta que ayude a los estudiantes a decidir su inscripción de módulos?**

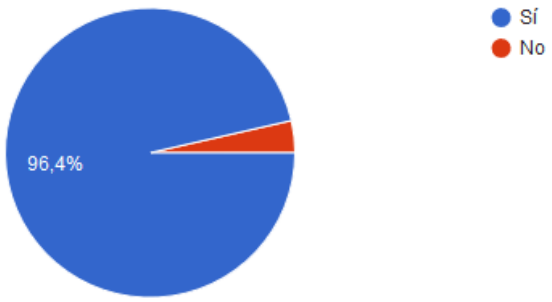


Figura C.1: Resultados de la Pregunta 1

2. **¿Qué tanto crees que puede ayudar esta herramienta a tomar mejores decisiones en la inscripción?**

   En escala de 1 (No es de ayuda) a 5 (Es de mucha ayuda).

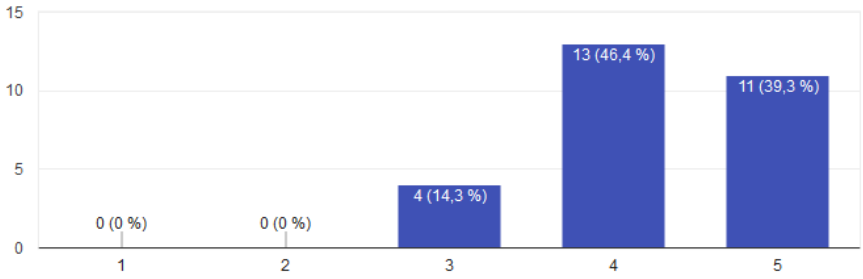


Figura C.2: Resultados de la Pregunta 2

3. **¿Qué tanto crees que puede ayudar esta herramienta a reducir el tiempo de egreso de los estudiantes que la utilizan?**

   En escala de 1 (No es de ayuda) a 5 (Es de mucha ayuda).

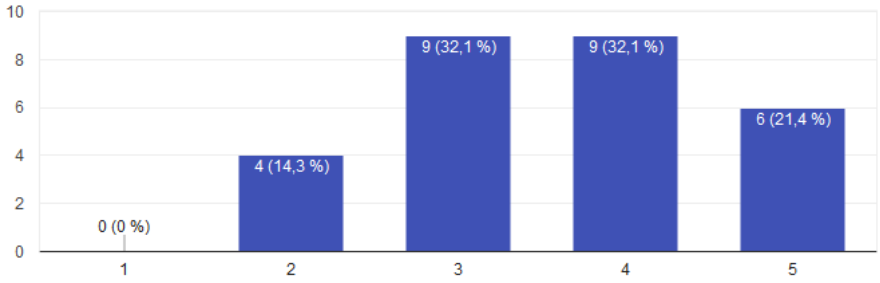


Figura C.3: Resultados de la Pregunta 3

4. **¿Te apoyarías en esta aplicación para decidir tu inscripción de módulos?**

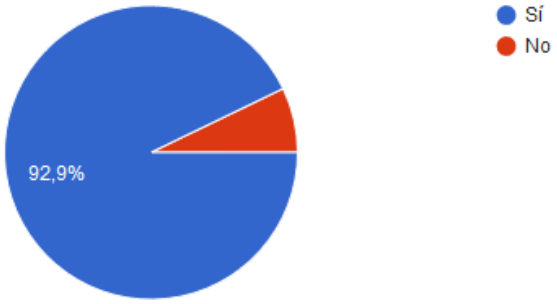


Figura C.4: Resultados de la Pregunta 4

5. **¿Qué tanto impacto en tus decisiones crees que puede tener el uso de esta herramienta en tu próximo proceso de inscripción?**

   En escala de 1 (No tiene impacto) a 5 (Tiene un gran impacto).

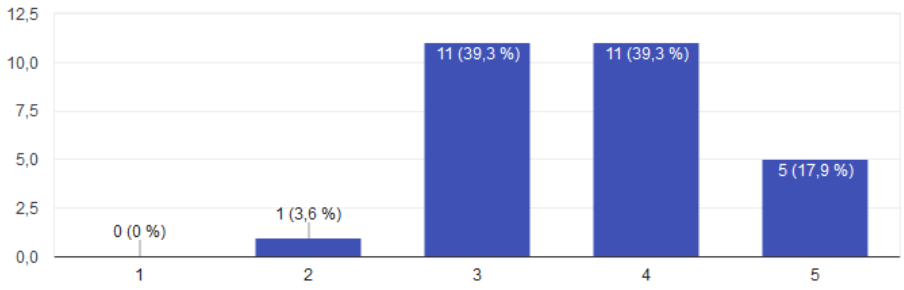


Figura C.5: Resultados de la Pregunta 5

Sección de preguntas sobre las funcionalidades

6. **¿Las funcionalidades son suficientes para que la aplicación sea útil?**

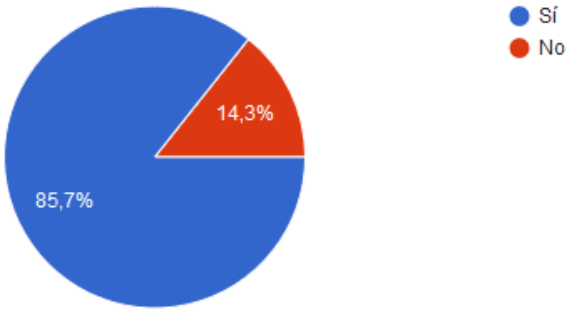


Figura C.6: Resultados de la Pregunta 6

7. **¿Los resultados que entrega el sistema son fiables?**

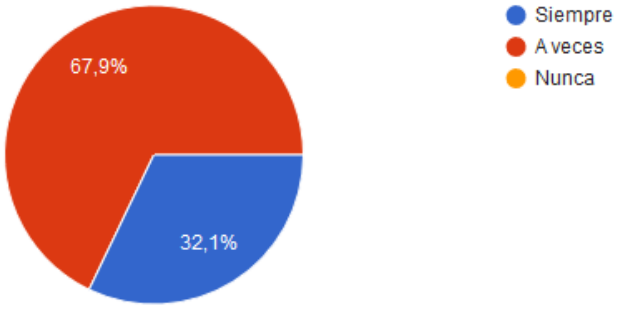


Figura C.7: Resultados de la Pregunta 7

8. **¿Cuántos errores encontraste durante el uso de la herramienta?**

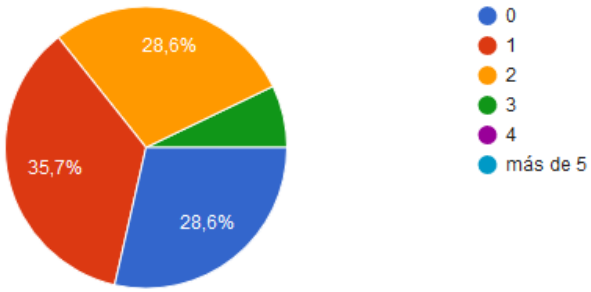


Figura C.8: Resultados de la Pregunta 8

Sección de preguntas sobre la usabilidad e interfaz de usuario

Marca en la escala qué tan de acuerdo estas con cada afirmación. 1 corresponde a **En desacuerdo** y 5 a **De acuerdo**.

9. **La aplicación es fácil de usar**

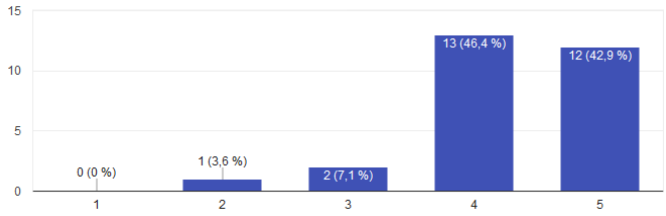


Figura C.9: Resultados de la Pregunta 9

10. **Se entiende cómo utilizar la aplicación sin necesidad de instrucciones**

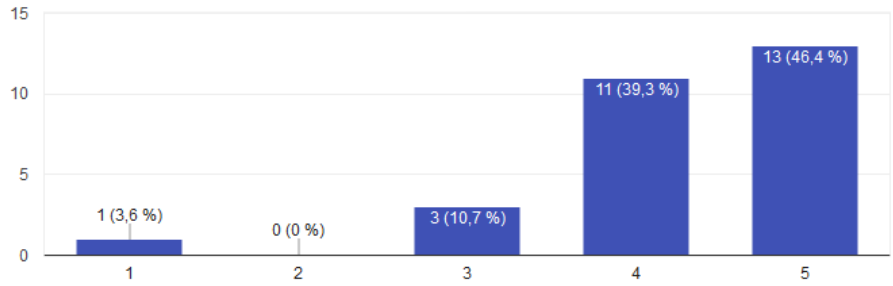


Figura C.10: Resultados de la Pregunta 10

11. **El tiempo de espera de las operaciones es razonable**

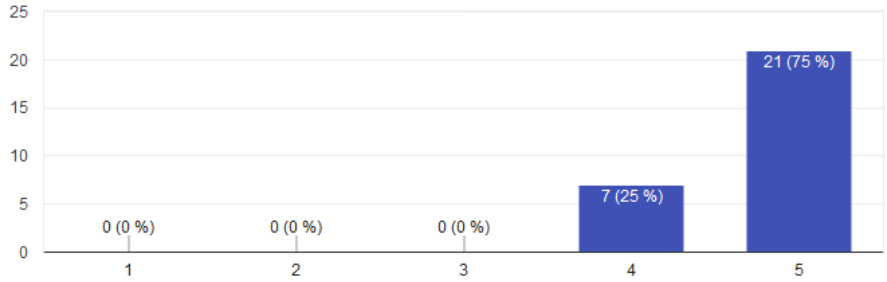


Figura C.11: Resultados de la Pregunta 11

12. **Cada notificación (mensaje informativo) de la aplicación corresponde con las acciones realizadas**

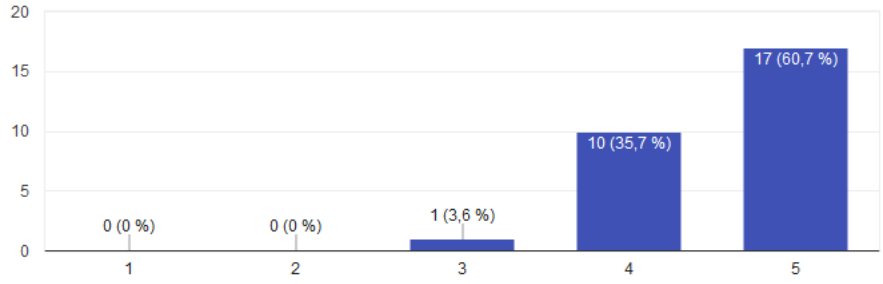


Figura C.12: Resultados de la Pregunta 12

13. **La distribución de los elementos en la pantalla es la adecuada**

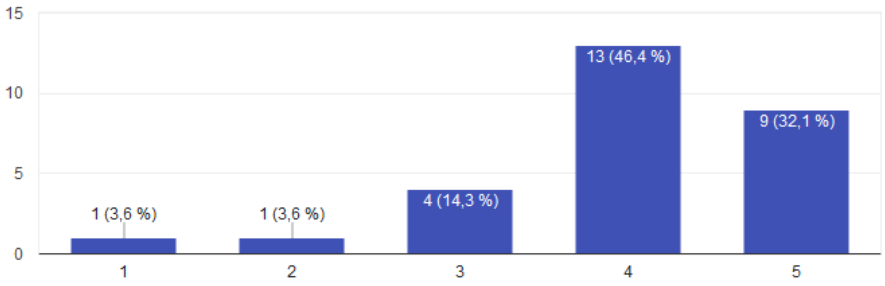


Figura C.13: Resultados de la Pregunta 13

14. **La navegación a través de la aplicación es predecible (puedo saber de antemano a donde me lleva cada acción)**

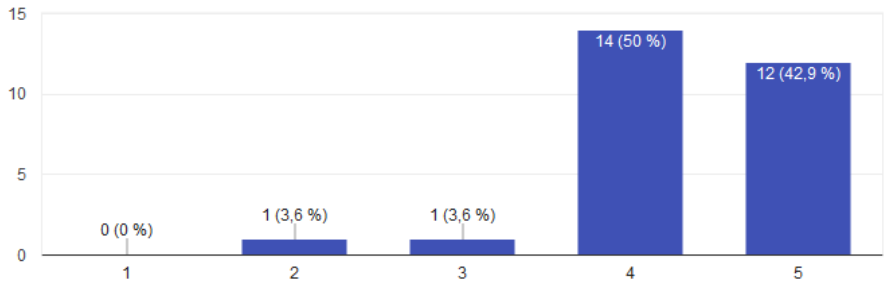


Figura C.14: Resultados de la Pregunta 14

15. **Existen todas las opciones necesarias para navegar por la aplicación**

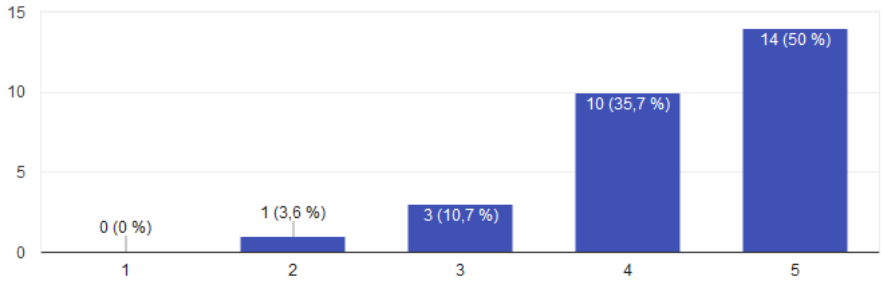


Figura C.15: Resultados de la Pregunta 15

16. **El proceso completo de realizar la simulación y de escoger un plan de inscripción es claro**

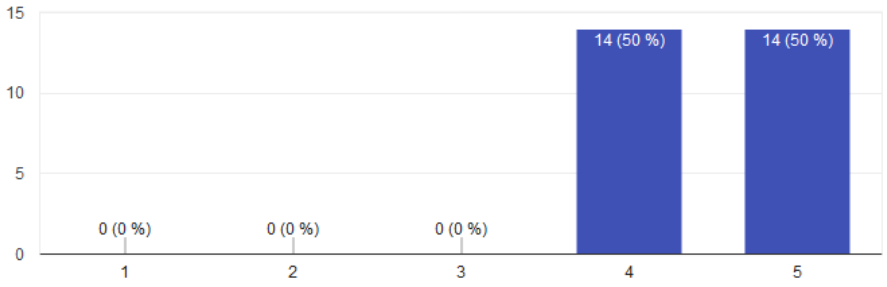


Figura C.16: Resultados de la Pregunta 16

17. **Entiendo todos los textos y terminología en la aplicación**

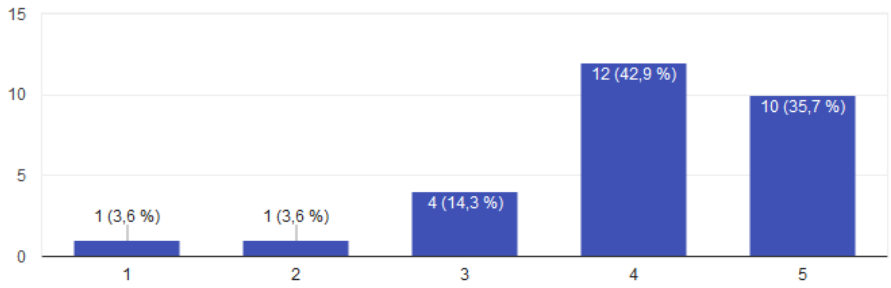


Figura C.17: Resultados de la Pregunta 17

18. **Existen todas las opciones necesarias para navegar por la aplicación**

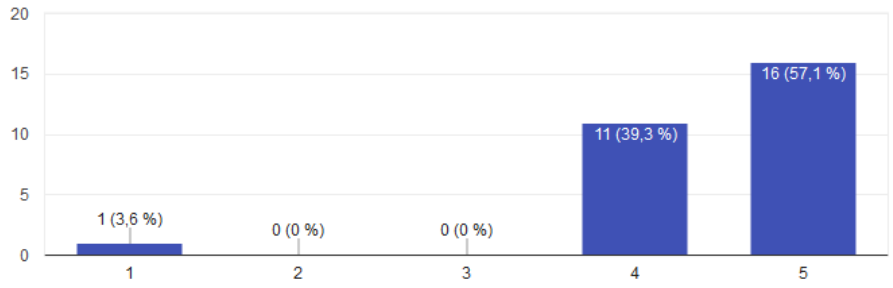


Figura C.18: Resultados de la Pregunta 18

19. **Realizar simulaciones para conocer planes de inscripción es fácil**

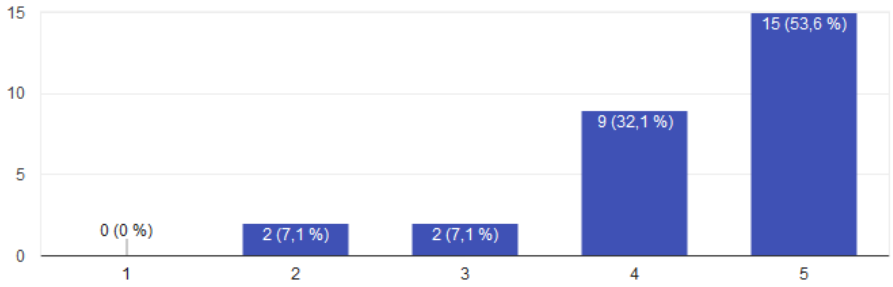


Figura C.19: Resultados de la Pregunta 19

20. **Los controles para marcar mi situación respecto a los módulos (ej: aprobado) son los adecuados**

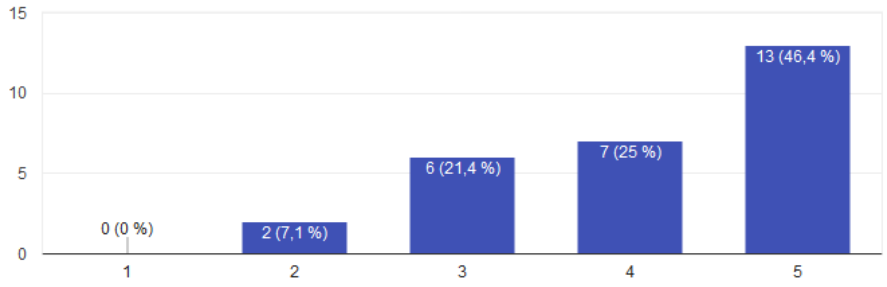


Figura C.20: Resultados de la Pregunta 20

21. **La presentación de los planes de inscripción y sus módulos los hace fácil de entender**

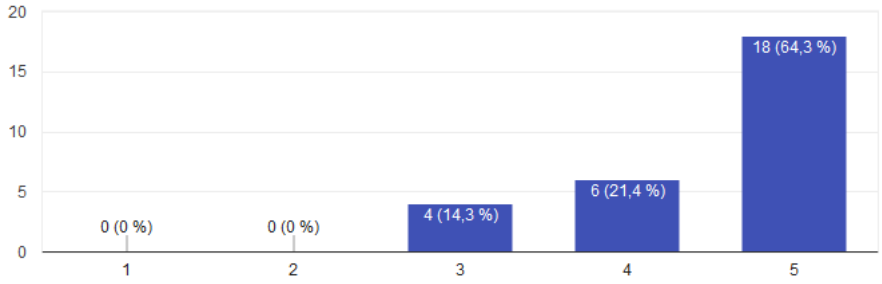


Figura C.21: Resultados de la Pregunta 21

D. Documento emitido por la profesora *Ruth Garrido*

En la página siguiente se encuentra el documento emitido por la profesora *Ruth Garrido*, Directora de Escuela de la carrera de Ingeniería Civil en Computación.

Dicho documento expresa su aprobación al sistema implementado en este *Proyecto de Titulación*.

TALCA UNIVERSIDAD CHILE

INGENIERÍA CIVIL EN COMPUTACIÓN

CONSTANCIA

En Curicó, Chile, a 15 de Junio de 2018, la Sra. ***Ruth Garrido Orrego****, Directora de Escuela de la Carrera de Ingeniería Civil en Computación de la Universidad de Talca, deja constancia que:*

El Sr. FABIAN ALBERTO OLIVARES ARREDONDO rut 18,165,443-0, matrícula 2013407001, ha presentado a total satisfacción la implementación de un sistema de software para apoyar al estudiante en la toma de decisiones durante el proceso de inscripción de asignaturas.

El mencionado sistema pasará a formar parte de las aplicaciones con que cuenta la escuela, para lo cual quedará disponible en el servidor indicado por esta, antes de que finalice el módulo Proyecto de Titulación, curso en el cual el estudiante ha comprometido este proyecto.

Se extiende la presente constancia a solicitud del estudiante, para ser incluido en el informe de Proyecto de titulación semestre 2018-1.

UNIVERSIDAD DE TALCA * DIRECTOR ESCUELA * INGENIERÍA CIVIL EN COMPUTACIÓN

Ruth Garrido Orrego
Directora de Escuela
Ingeniería Civil en Computación
Universidad de Talca