UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

OPTIMIZACIÓN DE LA CONFORMACIÓN DE EQUIPOS DE PROYECTO DE SOFTWARE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

DIEGO GASPAR DÍAZ OSORIO

PROFESORA GUÍA:
MARÍA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
JOSÉ PINO URTUBIA
HUGO MORA RIQUELME

SANTIAGO DE CHILE
2020

Resumen

Un desafío recurrente dentro de la carrera de computación es aquella de trabajar en equipo para dar solución a la(s) preocupación(es) de un cliente. Para ello, se cuenta con las instancias de práctica profesional, en la que se busca que el alumno tenga una experiencia cercana a lo que vivirá en su vida profesional una vez egresado. La culminación de este proceso ocurre en el curso de *Proyecto de Software*, donde se trabaja con un cliente real con problemas reales. Dicho esto, se busca que la experiencia vivida por el alumno y su equipo sea una de aprendizaje, donde idealmente puedan desarrollar un producto que cumpla con lo pedido por el cliente, pero que por sobretodo, se logre trabajar en equipo de buena manera, creando un espacio para compartir conocimiento y de respeto entre todos.

Actualmente, la conformación de los equipos de trabajo en *Proyecto de Software* se ha hecho de manera manual, pues la demanda del curso ha permitido que sea factible y no muy complejo de lograr. Sin embargo, lo anterior no implica que el proceso de conformación de equipos sea trivial. Al pasar del tiempo este proceso se ha tornado aún más complejo cuando se toma en cuenta que la cantidad de alumnos que necesitan tomar el curso aumenta cada vez más año tras año, por lo que conformar los equipos de manera balanceada y justa se hace una tarea cada vez más difícil de abordar.

Con el objetivo de facilitar la conformación de equipos y a la vez mantener la calidad de estos, se ha creado un algoritmo y un sistema que trabajan de la mano para lograr este fin. El algoritmo creado genera dos valores, denominados *armonía local (AL)* y *armonía global (AG)*. El primero mide qué tan bien armado quedó un equipo en base a la cantidad de miembros que logran trabajar de manera presencial en la semana. El segundo mide qué tan parejo o justo quedó el curso a nivel general en base a los valores de AL obtenidos.

Para corroborar la eficacia del sistema se llevó a cabo una prueba donde se utilizó la información de alumnos de un semestre anterior y la configuración manual de los equipos conformados en ese entonces. Se comparó si los equipos conformados de manera automática por el sistema arrojaban mejores valores de AL y AG que en el caso en que se forzaba a la configuración manual realizada anteriormente. Los resultados indicaron que en promedio el sistema cumple con llevar a cabo un proceso de conformación de equipos de manera rápida y sin perder calidad en la conformación, es más, en algunos casos es incluso mejor a la configuración manual.

Así, se espera que las futuras iteraciones de *Proyecto de Software* se realicen en un ambiente lo más justo posible, asegurando la calidad de aprendizaje de cada uno de sus miembros y esperando que así se aproveche al máximo la experiencia académica que ofrece éste.

A mi familia, por el apoyo incondicional en este largo proceso.

A todas las amistades que estuvieron y están, pues ayudaron en los momentos bajos y acompañaron en los altos.

A Nati, por su cariño, ayuda y paciencia en esta etapa final.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

Dentro de la carrera de computación se imparten una variedad de enseñanzas con respecto a ésta y el mundo de la programación. Metodologías de diseño, algoritmos, bases de datos, arquitectura de computadores, teoría, etc; son todas habilidades que forman parte fundamental de un ingeniero al momento de adentrarse al mundo laboral. Sin embargo, hay una cualidad que al pasar del tiempo se ha vuelto también muy relevante e importante de tener: habilidades blandas, en particular el saber comunicarse con un equipo y con el cliente. Una vez completadas las prácticas profesionales, la última instancia en la que se debe trabajar en equipo situado en un ambiente real con clientes y problemas reales, es en el curso de *Proyecto de Software*.

Existen aspectos que hacen que *Proyecto de Software* se diferencie de los cursos de *Ingeniería de Software* que lo anteceden y las prácticas profesionales, lo cual conlleva a que se requiera de mayor eficacia al momento de conformar los equipos que trabajarán en los distintos proyectos del semestre. En primer lugar, a diferencia de los cursos de *Ingeniería de Software*, los clientes de *Proyecto de Software* suelen ser de departamentos externos al de computación, y en ocasiones, externos a la universidad completamente, por lo que se hace necesario que el equipo trabaje en las dependencias del cliente una cantidad fija de horas semanales. Adicionalmente, a diferencia de las prácticas profesionales que suelen llevarse a cabo en el periodo de verano, sin clases de por medio, en *Proyecto de Software* se debe llevar un equilibrio entre el trabajo y las otras clases que estén cursando los alumnos. Lo anterior implica que los equipos conformados han de tener una disponibilidad horaria similar entre todos sus miembros para así trabajar de mejor manera entre sí.

En la actualidad, los equipos de trabajo del curso *Proyecto de Software* se conforman de modo tal que ningún alumno se quede trabajando solo mientras cumple sus horas de trabajo. Esto, con la idea de lograr comunicación entre los miembros del equipo de manera presencial, lo cual fomenta la buena gestión del proyecto. Es de suma importancia que lo anterior se cumpla, pues de lo contrario resulta complejo lograr un nivel apropiado de productividad. Lo anterior ha sido estudiado y observado en el trabajo de Hayward [1]. En su estudio compara el nivel de productividad en equipos de desarrollo de software que trabajan de manera presencial, con aquellos que lo hacen de manera virtual. Concluyó que se logra un mayor nivel de calidad en la interacción del equipo gracias al contacto visual y lenguaje corporal. Tomando lo anterior como base, si bien es necesario que el trabajo en equipo ocurra de manera presencial, se debe analizar también que la manera en que se conforma éste sea lo más óptimo posible.

La problemática de lograr optimizar el desempeño de equipos de desarrollo de software no es reciente, existen múltiples trabajos que han investigado en profundidad dicho tema, identificando categorías bajo las cuales medir qué tan productivo es un equipo. En particular, uno de los aspectos que suele ser más crítico al momento de definir si éste será o no óptimo, corresponde al factor del tamaño del equipo. De acuerdo al trabajo realizado por Putnam [14], existe un tamaño de equipo óptimo que permite un alto nivel de productividad para un periodo acotado, con un bajo costo y sin que afecte el resultado final del proyecto. Putnam (de acuerdo a lo citado por Rodríguez [16]) menciona que la cantidad de miembros del equipo depende de múltiples variables, dentro de las cuales se encuentran: (i) la cantidad de código a desarrollar y reutilizar, (ii) la complejidad del proyecto y (iii) el grado en el cual el tiempo planificado o el costo son las variables de restricción predominantes. Como producto de una serie de análisis, Putnam concluye que el tamaño óptimo de un equipo está en el rango de 3-5 personas, obteniendo resultados similares en el rango de 5-7, y notando que en un equipo mayor a 9 personas se comienza a observar un crecimiento exponencial en términos de esfuerzo y costo.

Justificación del trabajo

La temática de la conformación de equipos de desarrollo de software no es algo nuevo en el DCC, existe un trabajo de tesis realizado por Luis Silvestre el año 2012 [18]. En dicha tesis, se realizó un trabajo exhaustivo donde se midieron diversas áreas de personalidad y psicología para lograr conformar equipos que tuvieran un buen nivel de cohesión entre ellos, y por ende, que lograran más productividad durante el semestre. En resumen, el trabajo consistió en desarrollar una heurística para el diseño de equipos cohesivos, el cual es utilizado actualmente en el curso de *Ingeniería de Software II*. Sin embargo, si bien el sistema está enfocado en formar equipos de acuerdo a factores personales, no considera la disponibilidad horaria de sus integrantes.

Actualmente, la conformación de los equipos en *Proyecto de Software* se mantiene de manera manual, pues la demanda del curso ha permitido que sea factible y no muy complejo de lograr. Sin embargo, dado que el DCC ha incrementado su cantidad de alumnos año tras año, dicha demanda comienza a ser cada vez más alta y en consecuencia la conformación manual se hace más y más compleja de realizar, debido a la alta afluencia de alumnos que necesitan tomar el curso. Solucionar el problema de equilibrio en los equipos de modo que puedan trabajar la mayor cantidad de sus miembros de manera simultánea, se ha vuelto cada vez más difícil de llevar a cabo. Esto debido a que no se desea crear escenarios de desventaja donde por ejemplo dos alumnos trabajen siempre juntos pero sin interactuar con el resto del equipo. Es por ello que lo que se busca construir en este trabajo de memoria es facilitar y automatizar, mediante la creación de una aplicación web y una fórmula de "armonía", la conformación de equipos de trabajo, con la finalidad que todos tengan una oportunidad equivalente de poder llevar a cabo sus proyectos con el mayor nivel de productividad posible, vale decir, trabajando juntos de manera presencial.

Cabe mencionar que existe una primera versión de acercamiento a la resolución de la conformación de equipos para *Proyecto de Software*, diseñada por un equipo del ramo *Ingeniería de Software II* el semestre de primavera 2017. Sin embargo, dicho software calcula todas las combinaciones posibles de equipos y por ende no está pensado para entregar los mejores

equipos en base a los horarios de sus integrantes; adicionalmente, no permite configurar los equipos una vez que estos han sido armados.

Objetivos

Objetivo General

El objetivo general de este trabajo de memoria es lograr idear e implementar un nuevo sistema de conformación de equipos de proyecto de software, que toma como inspiración el sistema ya existente, pero que busca optimizar la creación de los equipos a nivel global en base a una fórmula de "armonía". La utilización de dicha fórmula dependerá de los valores locales de "armonía" asociados a cada equipo. Este valor a nivel local indicará qué tan ideal es la conformación de un equipo en términos de la disponibilidad horaria de sus miembros, siendo éste más alto a medida que coincidan la mayor cantidad de miembros del equipo en un mismo bloque horario y en la mayor cantidad de bloques a la semana. A nivel global, este valor indicará qué tan equilibrados están todos los grupos entre sí, buscando alcanzar el mayor porcentaje posible. El sistema ha de ser intuitivo de usar y permitir visualizar a los miembros de los equipos conformados, además de sus respectivos valores de "armonía", tanto locales como el global. Finalmente, debe permitir la modificación de los miembros del equipo y posteriormente recalcular todos los valores pertinentes.

Objetivos Específicos

1. Crear un módulo para consultar al alumno sus horarios disponibles para cada día de la semana, para el semestre en curso.
2. Generar una lista con las permutaciones de alumnos inscritos, desde el cual se obtengan cantidad de horas en común, además de bloque horario al que corresponde.
3. Crear un algoritmo de cálculo del valor de "armonía" local (desde ahora AL) de un equipo.
4. Crear un algoritmo de "armonía" global (desde ahora AG) que optimiza la distribución de equipos en base a los valores de AL.
5. Crear una herramienta web de apoyo a la conformación de equipos, utilizando los valores de AL y AG.

Capítulo 1

Marco teórico

Como ya fue mencionado, la problemática de formar equipos no es nueva, pero suele abordarse desde diferentes perspectivas. Dentro del contexto a la solución que se desea obtener, está la creación de dos tipos de algoritmos que irán de la mano en lograr conformar equipos que sean lo más balanceados en términos de coincidencia de horario para trabajar en conjunto, para llegar así a un escenario que sea lo más justo para todos. A continuación se detallará el objetivo de cada uno de estos algoritmos, así como las alternativas existentes al momento de intentar resolver el problema planteado.

1.1. Algoritmo de armonía local, AL

El propósito de este algoritmo está enfocado, tal como dice en su nombre, en valorar la calidad de cada uno de los equipos que se conformen, sin importar los otros grupos que existan. En principio, lo que se desea es que todos los integrantes del grupo de trabajo estén disponibles para trabajar en conjunto y de manera presencial el mayor tiempo posible. Ante esta condición, surge la interrogante del ¿por qué es tan fundamental que los miembros del equipo trabajen juntos presencialmente? En parte su importancia ya fue mencionada en el capítulo introductorio de esta memoria, haciendo alusión al trabajo realizado por Hayward [1] con respecto a la importancia del trabajo presencial en equipos de desarrollo de software, pero existe además otro referente ampliamente popular desde el cual se hace hincapié a este mismo principio: el manifiesto ágil y sus 12 principios.

Con el fin de contextualizar lo anterior, es sabido que dentro del ámbito de la computación y el desarrollo de software, una de las metodologías de trabajo más usadas a través de los años es aquella descrita por Winston Royce [17] en el año 1970, la metodología cascada. A grandes rasgos, la metodología sigue los siguientes pasos: toma de requisitos, análisis, diseño, programación, testeo y operación. Sin querer adentrarse demasiado a si el modelo cascada es efectivo o no (pues no es el enfoque de esta memoria), sí se puede observar que dado el modelo de trabajo, la necesidad de trabajar en equipo de manera presencial para resolver la necesidad de un cliente, no era primordial. Muchos desarrolladores se convirtieron en lo que se conoce coloquialmente como "tomadores de pedido" y simplemente desarrollaban lo que se requería con interacción limitada tanto con el cliente como con sus compañeros de equipo.

Tomando este modelo como base, surge la necesidad de cambiar la metodología con la que se trabaja en el área de desarrollo de software, por una que sea más inmediata, de mayor comunicación tanto con el cliente y con los integrantes de un grupo de trabajo. Es con lo anterior en mente que el año 2001 se crea el manifiesto ágil [3] y con él, 12 principios que describen la esencia de lo declarado. En particular cabe destacar el principio número seis: *"El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara"*. Y es que con ello, se pueden desarrollar proyectos de manera más rápida, abordando de mejor manera las necesidades de un cliente y logrando abordar problemas que ocurran en el camino rápida y eficazmente. Lo anterior justifica entonces que sea relevante valorar el nivel de calidad de un equipo en base a la cantidad de tiempo que logran trabajar la mayor cantidad de alumnos de manera presencial.

1.2. Algoritmo de armonía global, AG

En contraste al algoritmo recién descrito, el propósito de la armonía global recae en que a nivel de curso, los grupos formados estén lo más balanceados entre sí. La justificación de obtener este valor se basa principalmente en el hecho que *Proyecto de Software* es un curso académico y que por tanto se busca que los alumnos estén en una situación equitativa a lo largo del semestre, sin crear desigualdad de condiciones entre un alumno y otro. Esto apunta a que los alumnos posean las herramientas para que, dentro del contexto del curso, ningún alumno posea más ventaja que otro, vale decir, que las oportunidades del curso sean justas para todos.

Cuando se habla de justicia en la sala de clases, se hace referencia a lo que la Dra. Rebecca Chory [4] menciona en su artículo del año 2002, que hace referencia al tema en cuestión, donde describe el concepto de *Justicia Organizacional*. El término que originalmente fue explicado por Cropanzano & Greenberg [5], se refiere a la percepción de igualdad y las evaluaciones que conciernan qué tan apropiado es un lugar de trabajo en cuanto a lo que se obtiene de él y los procesos que se llevan a cabo dentro de él. De este término se desprenden dos conceptos: *Justicia Distributiva* y *Justicia de Procedimiento*. El primero, explicado por Deutsch [6] y luego Kabanoff [10], se centra en la igualdad de la distribución de condiciones y bienes que afectan a una persona, grupo o bienestar de una comunidad y considera aspectos de índole psicológico, fisiológico, económico y social. En cuanto al segundo término, Leventhal [11] lo describe como "la percepción que tiene un individuo ante la igualdad en los procesos que componen el sistema social en la regulación de distribución de recursos", lo cual visto desde la perspectiva de una sala de clases, sería la consistencia tanto en la distribución de notas, así como las condiciones necesarias para obtenerlas.

Con ese principio de justicia como base, se busca que el algoritmo logre generar un valor a nivel de curso que refleje el factor de igualdad y equilibrio que existe entre los grupos formados.

1.3. ¿AL o AG?

Habiendo explicado el razonamiento detrás de los dos algoritmos a utilizar en la solución al problema, surge la última pregunta a considerar con respecto a ellos, ¿cuál tiene mayor

importancia al momento de conformar los equipos? Para poder responder esta pregunta, se describirán dos casos: en el primero la importancia la tendrá el valor de AL y en segundo el de AG.

1.3.1. Caso uno: alto nivel de AL y bajo nivel de AG

Es natural que a medida que se realice el proceso de conformación de equipos y vayan quedando menos alumnos disponibles sin uno, la flexibilidad de encontrar horario en común con otros integrantes del curso disminuirá; esto inevitablemente hará que del total de equipos que se formen, habrán uno o dos con altos niveles de AL mientras que los otros equipos tendrán valores menores. Si bien todos los valores de AL serán altos dentro de lo posible para cada equipo, habrán algunos que estarán mejor conformados y por ende hará que la sensación de igualdad y equilibrio entre equipos se vea alterada, en otras palabras, un bajo AG, lo cual si no se soluciona puede resultar en un mal trabajo semestral de múltiples equipos.

1.3.2. Caso dos: alto nivel de AG y bajo nivel de AL

Por otro lado, suponiendo que se realiza la conformación teniendo en mente que el valor de AG debe ser lo más alto posible, implicará que los valores de AL han de estar lo más nivelados posibles entre sí. Esto puede provocar dos posibles escenarios para un equipo; por un lado tendrán un nivel de AL más bajo del óptimo en caso que se hubiese dado prioridad a ese valor, pero por otro, algunos equipos tendrán un mejor valor de AL en comparación al que hubiesen tenido en el caso uno. Si bien es cierto que aquellos equipos que se vieron perjudicados por el nivelamiento no lograrán el mismo nivel de trabajo en equipo que el caso uno, también se puede decir que no será el peor, sino más bien un caso intermedio.

Por consecuencia, considerando que el contexto de la conformación de equipos se produce en un curso académico y que tal como fue explicado se busca generar una sensación de igualdad entre todos los alumnos del curso, se dará mayor importancia en lograr que el valor de AG sea alto. Lo anterior implica que, si bien se hará un esfuerzo por buscar los mejores valores posibles de AL para cada equipo, ante cualquier cambio que se deba realizar entre miembros de diferentes equipos, tendrá más peso e importancia que el valor de AG no se vea impactado negativamente.

1.4. Soluciones existentes

1.4.1. Woven

Woven [20] crea encuestas para coordinar horarios en común entre personas y a su vez agrega múltiples funcionalidades de coordinación de equipo. Permite la sincronización de múltiples calendarios, posee templates de reuniones tipo (salida a un café, reunión de equipo, reunión con cliente, etc), e incluso ofrece opciones de planificación de viajes antes de asistir a una reunión de modo que se disminuya el riesgo de atrasos. En general, es una aplicación que resulta ser muy útil para coordinar a miembros de un equipo o para gestionar reuniones con un cliente, pero tal como ocurre con otras herramientas de coordinación de horarios, no ofrece las funcionalidades de conformación de equipos que se busca resolver. En términos de conceptos utilizados en esta memoria, se puede decir que si bien puede lograr conformar

equipos que posiblemente tengan buen AL, no existe ninguna manera de medir ni conocer cómo está el AG de los equipos como un todo.

1.4.2. TeamBuilder

Si bien el enfoque de TeamBuilder [19] posee similitudes a aquellas descritas para Woven, TeamBuilder tiene la particularidad que su foco está en la conformación de un equipo de trabajo. Para lograr esto, posee perfiles de empleador y empleado, donde el primero crea una búsqueda de un perfil en particular, mientras que el segundo se encarga de rellenar su perfil de la manera más completa posible. Una vez que el empleador ingrese los parámetros de su búsqueda, se le entregará un listado con todos los empleados que calzan con su petición, de manera anónima. La elección de un empleado se basa en el conocimiento que éste posee, pero también puede incluir otros parámetros acorde a lo que busca el empleador, ya sea disponibilidad horaria, costo del empleado, experiencia, etc. Si bien esta herramienta sí cumple con el factor de conformar un equipo y es posible que uno de sus parámetros sea por disponibilidad horaria, aún sigue siendo un proceso bastante manual y sin la posibilidad de aplicarlo de manera automática a un amplio grupo de personas. Al igual que con la solución anterior, si bien es posible lograr un alto valor de AL, no hay manera de poder medir qué tan bien quedan estos grupos en un contexto general (en este caso a nivel de empresa o departamento).

1.4.3. Carencias de las alternativas

El problema de estas alternativas recae en lo mencionado al comienzo de este capítulo, la perspectiva desde la cual se aborda el evento de coordinar a una cantidad de personas en una fecha determinada. Si el enfoque se centrara en encontrar una herramienta que logre encontrar los bloques en que un grupo de personas pueda coincidir para trabajar en conjunto, entonces la oferta es amplia; pero al hacer el ejercicio de buscar una herramienta que no tan solo logre agrupar a personas en bloques determinados dentro de una semana, sino que además permita modificaciones de dichos grupos y otorgue un factor que indique qué tan "armoniosos" o balanceados han quedado estos grupos entre sí y a nivel general, entonces es fácil darse cuenta que no existe algo apropiado que cumpla con estos requisitos en el mercado. Por otro lado, aquellas alternativas en las que se puede llegar a un resultado cercano al deseado, creando equipos con disponibilidad horaria en mente, no tienen la facultad de poder realizar este proceso de manera automática para una amplia cantidad de personas. Dado estos antecedentes, se tomó la decisión de crear una aplicación web desde cero, que cumpla con las necesidades del problema.

Capítulo 2

Concepción de la solución

2.1. Perfiles

Al momento de pensar en cómo idear una solución que aborde el problema dado, es importante distinguir en primera instancia cuáles son los tipos de usuarios que harán uso del sistema. Para este caso en particular se distinguen dos tipos de usuarios de manera clara: *Docente* y *Alumno*. Esta distinción ocurre de manera natural, ya que son los actores involucrados en el proceso de conformación de equipos y tienen necesidades específicas que han de ser cumplidas. Adicional a los perfiles mencionados, surge la necesidad como en cualquier sistema de software, de contar con un perfil *Administrador*.

2.1.1. Docente

Al hablar de *Docente* se hace referencia no tan solo al profesor de cátedra encargado del curso, sino que también a profesores auxiliares y/o ayudantes según sea el caso. La decisión de englobar a todos los roles del cuerpo docente en un solo perfil tiene como propósito entregar flexibilidad al ente a cargo de cumplir con las acciones correspondientes del rol. Lo anterior se pensó así ya que si sólo se dejara la responsabilidad al profesor de cátedra y ocurriera una situación donde se requiera la acción de este perfil sin estar éste disponible, produciría una situación de "cuello de botella" que perjudicaría el funcionamiento rápido y eficiente del sistema. El acceso de este usuario es amplio y debe ser capaz de interactuar con todas las secciones del curso que tiene a cargo.

2.1.2. Alumno

Por otro lado, en el perfil de *Alumno* no existen ambigüedades. Este perfil ha de tener un acceso limitado al sistema y cumplir con un set de acciones limitadas de modo que no perturbe los elementos del sistema a los que tiene acceso el perfil *Docente*.

2.1.3. Administrador

Finalmente, el perfil de *Administrador* debe encargarse de ejecutar procesos que afectan el funcionamiento del sistema propiamente tal. No es capaz de realizar acciones de los otros

perfiles mencionados.

2.2. Requisitos de la solución

A continuación se listarán los requisitos que se deben cumplir en la solución al problema planteado. Se listarán aquellos que son más relevantes y se obviará el requisito de crear un sistema de login que diferencie entre los perfiles ya mencionados, ya que este proceso es relativamente estándar en cualquier desarrollo de un sistema web.

2.2.1. Acciones de perfil administrador

La acción del perfil administrador es acotada y consiste en lo siguiente:

- **Cargar información del curso a la base de datos**.

  El administrador debe ser capaz de indicar información del curso incluyendo año, semestre, sección, además de la lista de integrantes de éste y cargarla en la base de datos del sistema.

2.2.2. Acciones de perfil docente

Dentro de las acciones que ha de poder realizar este perfil se identifican las siguientes:

- **Visualizar listado con todas las secciones del curso para el semestre actual**.

  Si bien es cierto que en el último tiempo el curso de *Proyecto de Software* se ha realizado con solo una sección, siempre existe la posibilidad que dado el incremento en la demanda de la carrera la creación de más secciones sea inevitable. Ante una situación así es necesario que el sistema sea capaz de distinguir entre secciones y tratarlas como entes independientes, sin errores de cruce de información entre ellos.
- **Capacidad de iniciar un proceso donde los alumnos puedan elegir los horarios en los que estarán disponibles para trabajar durante el semestre**.

  Un aspecto fundamental para dar inicio al curso de *Proyecto de Software* y en consecuencia uno de los aspectos que se debe considerar en la solución al problema planteado, es el proceso de elección de horarios de trabajo. Actualmente se soluciona mediante una tarea en la cual el alumno entrega un documento donde indica los horarios que tendrá disponible a la semana. Lo que se desea es digitalizar y automatizar ese proceso, por lo que se hace necesario dar la oportunidad de que el alumno elija horarios dentro de un plazo dado.
- **Posibilidad de modificar rango de tiempo en que proceso de elección de horas estará abierto**.

  Como complemento al requerimiento anterior, puede ocurrir que el plazo inicial de tiempo considerado para el proceso de selección de horarios no sea suficiente y que por tanto haya que extenderlo.
- **Revisión del estado del proceso de selección de horas**.

  De cierta forma ligado a la acción anterior, lo que se desea identificar en esta acción es el estado de cada alumno durante el proceso de selección de horas. El propósito consiste

en saber quiénes son aquellos que ya contestaron y quiénes no para así poder tomar las medidas correspondientes y extender el periodo en caso de ser necesario.

- **Inicio de proceso de conformación de equipos**.

  En base a los resultados del proceso de selección de horas, debe existir la posibilidad para que el docente inicie de manera manual el proceso de conformación de equipos. La necesidad nace a partir del hecho que si bien el proceso de selección de horas puede finalizar en la fecha y hora estipulada con anterioridad, pueden surgir casos bordes de alumnos que no registraron sus horas en el periodo establecido y que por tanto se debe volver a habilitar el proceso de manera extraordinaria para que dichos alumnos puedan contestar. Es importante entonces que la conformación de equipos no se lleve a cabo hasta que la totalidad de los alumnos haya seleccionado sus horarios disponibles.

- **Visualización de equipos conformados**.

  Una de las acciones principales que debería poder realizar un docente es visualizar los resultados de la conformación de equipos una vez que el proceso de selección de horarios ha finalizado. Mediante la visualización ha de ser capaz de ver si la distribución de los integrantes del curso quedó balanceada en cuanto a cantidad de integrantes por equipo y por sobre todo, revisar los valores de AL y AG obtenidos.

- **Modificación de miembros de equipo**.

  En base al punto anterior, debe existir la posibilidad de modificar los miembros de los equipos y que al hacerlo, se vea reflejado el efecto de los cambios en los valores de AL y AG. Esto debido a que por circunstancias extraordinarias se pueden dar situaciones donde ciertos alumnos no pueden quedar en el mismo grupo y han de ser cambiados (ej: incompatibilidades personales, pares de hermanos, entre otros).

2.2.3. Acciones de perfil alumno

Dentro de las acciones que ha de poder realizar este perfil se identifican las siguientes:

- **Que se liste sección correcta en la que está inscrito el usuario**.

  Como ya fue mencionado en el perfil de docente, existe la posibilidad que en el futuro haya más secciones del curso. Es importante que un alumno al momento de entrar al sistema, tenga disponible la sección del curso apropiado al que pertenece y no otro.

- **Capacidad de comenzar proceso de elección de horarios**.

  Un alumno debe ser capaz de ingresar al proceso de elección de horarios si es que este se encuentra abierto. Es fundamental que alumno sea capaz de pasar por este proceso, ya que de lo contrario no tendrá un equipo para el semestre.

- **Capacidad de seleccionar los bloques de horarios que el usuario estime conveniente**.

  Como complemento al punto anterior, un alumno debe ser capaz de elegir los horarios que él o ella estime conveniente. El sistema ha de distinguir entre cuáles fueron los bloques seleccionados y cuáles no, además de indicar al alumno cantidad de horas que lleva seleccionadas. Finalmente, el sistema hará un chequeo verificando que la cantidad escogida sea al menos la mínima necesaria para poder finalizar el proceso (1 bloque es equivalente 4 horas, por lo tanto el mínimo de bloques a elegir es 4).

- **Modificación de horarios elegidos**.

  Si un proceso de selección de horas sigue abierto y el alumno ya hizo una elección previa, ha de tener la opción de modificar sus horarios elegidos. Esta necesidad nace debido a que los horarios suelen elegirse en las primeras semanas del semestre, donde aún existe una cuota de incertidumbre en cuanto a cuál será el horario final con el que contará el alumno para el resto del semestre.

- **Visualización de equipo para el semestre**.

  Al finalizar el proceso de selección de horarios, y siempre y cuando el docente ya haya realizado el proceso de conformación de equipos, el alumno debe ser capaz de visualizar a cuál grupo fue asignado. Esta necesidad nace a partir de querer saber cuanto antes quiénes serán el resto de los integrantes del equipo y así saber si se debe presentar alguna objeción al respecto o no.

2.3. Procesos

A continuación se detallan los procesos por los que pasan los perfiles del sistema.

2.3.1. Flujo administrador

El perfil del administrador se conforma del siguiente flujo, el cual cubre la acción anteriormente descrita.

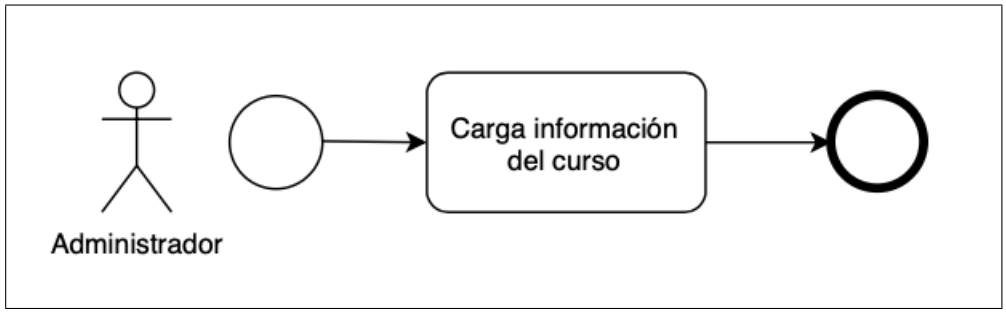

Figura 2.1: Flujo del administrador

2.3.2. Flujo docente

El perfil del docente se conforma del siguiente flujo, el cual cubre todas las acciones anteriormente descritas.

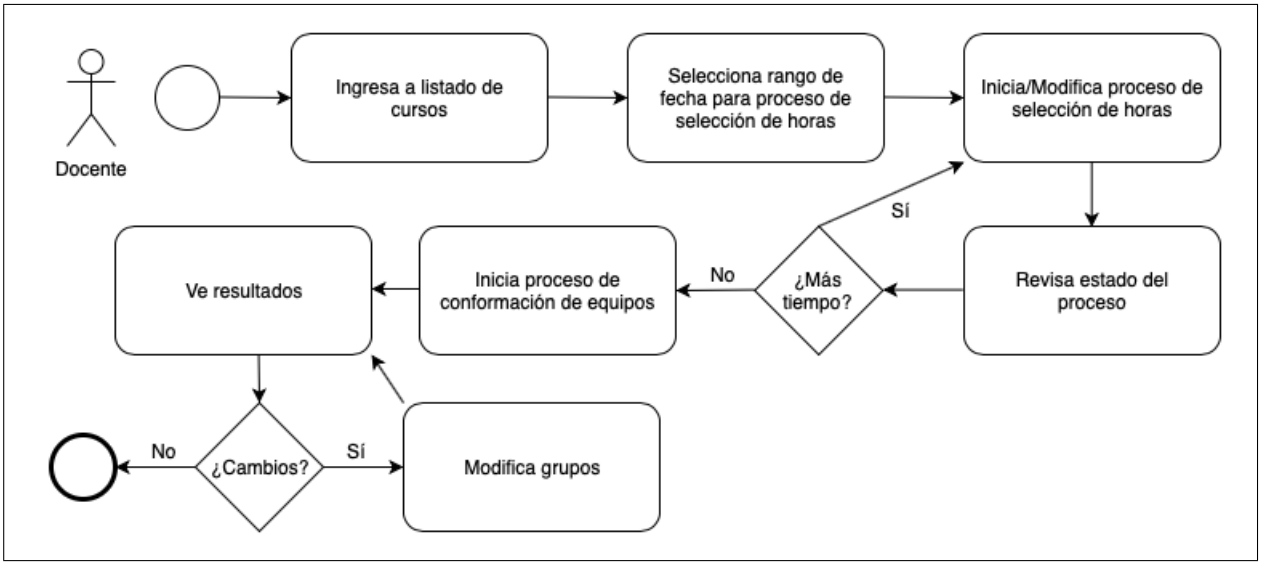

Figura 2.2: Flujo del docente

2.3.3. Flujo alumno

El perfil del alumno se conforma del siguiente flujo el cual cubre todas las acciones anteriormente descritas.

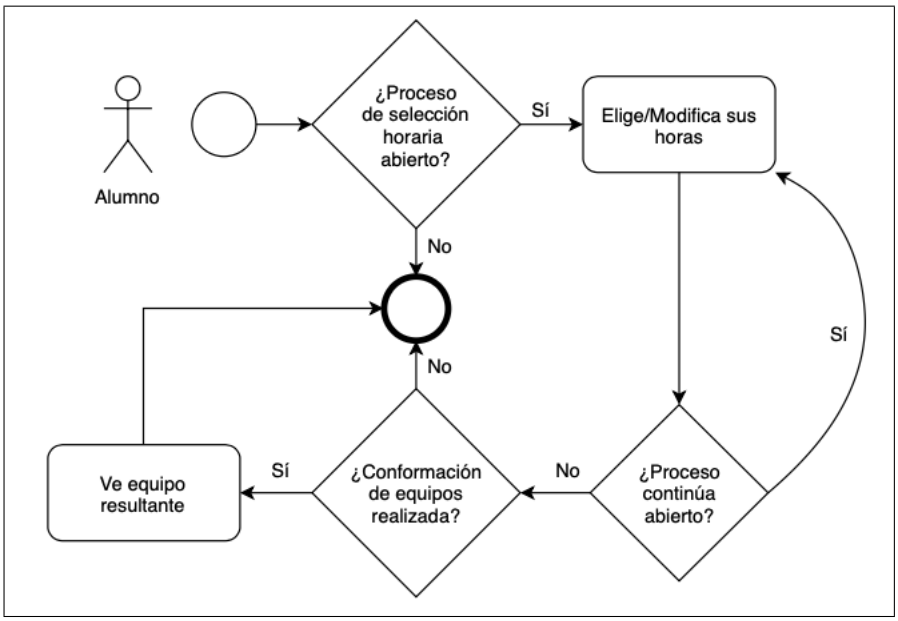

Figura 2.3: Flujo del alumno

Capítulo 3

Descripción de la solución

Ahora que se tiene más claro el contexto y la teoría de cómo abordar la solución, se procederá a describir los componentes que conformarán la solución tecnológica a implementar.

3.1. Arquitectura física

El requerimiento técnico del sistema a desarrollar resulta no ser demasiado demandante, puesto que en esencia lo que se necesita del sistema es un proceso de autenticación, para luego llevar a cabo un registro de horarios o bien iniciar un proceso de conformación de equipos. Esto quiere decir que no se necesitará de una infraestructura compleja ni de una multitud de servidores para abordar la solución. En base a lo recién descrito, se tiene el siguiente esquema de arquitectura física a nivel general.

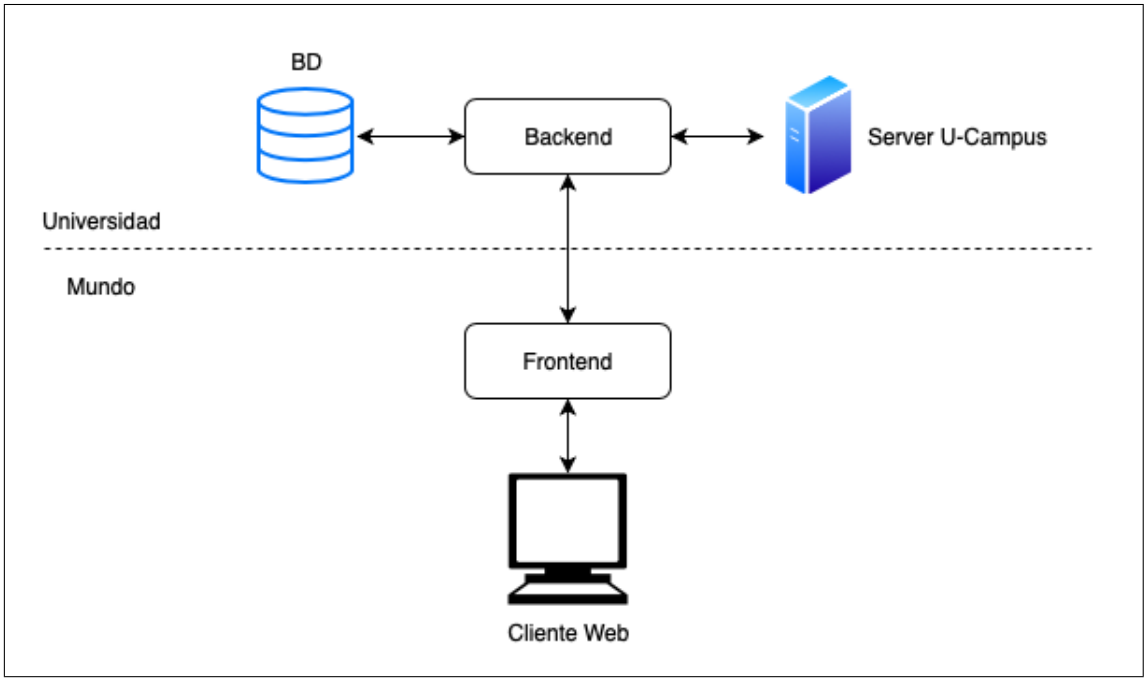

Figura 3.1: Arquitectura Física

Como se puede apreciar, la arquitectura del sistema es estándar, teniendo como única componente adicional el hecho que debe comunicarse con un server. Esta comunicación se utilizará con el único fin de autentificar al usuario.

3.2. Arquitectura lógica

Siguiendo el estándar de la mayoría de aplicaciones web del día de hoy, se seguirá un patrón MVC (Modelo - Vista - Controlador). Con la utilización de este patrón, se podrá mantener independiente el componente que maneja información (Modelo) con aquél que maneja la interfaz de usuario (Vista); todo mediante otro componente encargado del flujo de información de un componente al otro (Controlador). Para facilitar el uso del sistema en cualquier entorno y sin tener preocupaciones de incompatibilidades entre sistemas operativos, el backend, frontend y base de datos del sistema se levantarán dentro de contenedores.

3.3. Modelo de datos

Dada la naturaleza del problema y el enfoque de la solución, se ha decidido utilizar una base de datos de tipo relacional en lugar de una no relacional. Lo anterior se justifica considerando que se estará manejando una cantidad acotada de datos y por consecuencia las consultas a realizar no serán de alta complejidad; por ende, una base de datos relacional otorgará factibilidad y simplicidad al sistema. Dentro de las alternativas de bases de datos relacionales, se ha decidido utilizar PostgreSQL, ya que es un sistema robusto y que otorga múltiples funcionalidades y facilidades.

A continuación se tiene un diagrama general de la base de datos del sistema, omitiendo aquellas tablas de autenticación que son creadas por defecto al utilizar Django.

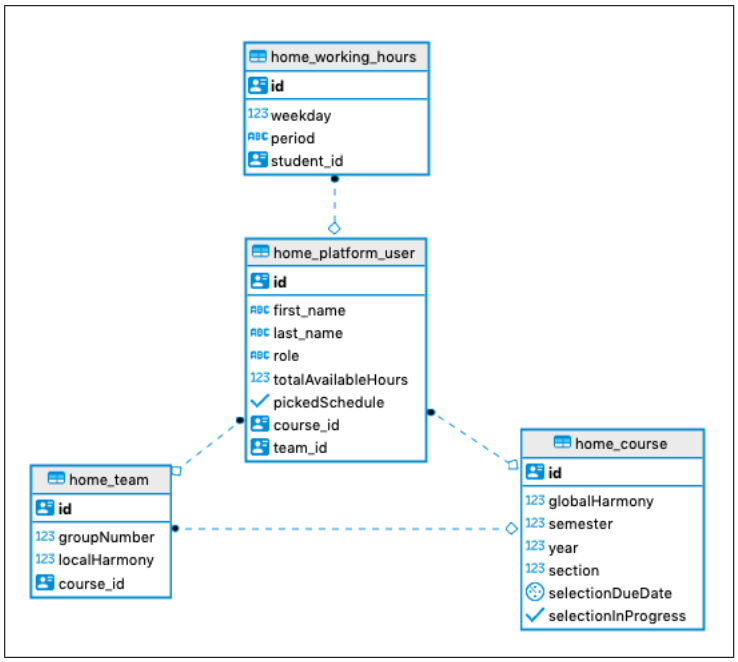

Figura 3.2: Modelo de datos

Como se puede observar, el modelo de datos es bastante compacto y sencillo debido a lo acotado de las acciones que se deben llevar a cabo en el sistema. A continuación se dará una breve descripción de la utilidad de cada una de las tablas del modelo.

1. **Platform User**

   Tabla que representa al usuario que ingresa a la plataforma. Posee información básica como nombre, curso, rol y equipo asociado. Para el caso particular de usuarios con rol alumno, posee campos que indican si ya registró horarios en el sistema y cantidad máxima de bloques horarios disponibles a la semana.

2. **Course**

   Tabla que representa al curso de *Proyecto de Software* como tal. En conjunto a la información básica de éste, indica si se está llevando a cabo un proceso de selección de horas y fecha de término de éste. Finalmente posee un parámetro que indica el valor de AG del curso.

3. **Working hours**

   Tabla que representa los horarios disponibles a la semana para un alumno en particular. Se distingue el día de la semana además del periodo disponible dentro de ese día (AM o PM).

4. **Team**

   Tabla que representa al equipo conformado para un curso en particular. Indica número de grupo además de valor de AL que posee dicho equipo.

3.4. Algoritmos de AL y AG

Para lograr obtener los valores de AL y AG, es necesario identificar las variables relevantes para formular un problema de optimización que cumpla con ciertos requisitos. Cabe destacar que estos parámetros son independientes para cada sección del curso de *Proyecto de Software* que se imparta durante un semestre. Dentro de los parámetros a considerar se identifica lo siguiente:

1. Cantidad de alumnos inscritos en el curso.
2. Cantidad de horas libres disponibles por alumno para cada día de la semana, además de bloque horario al que corresponde (AM o PM).

Dentro de las restricciones, se identifica lo siguiente:

1. No puede haber un alumno solo en ningún bloque de trabajo.
2. Cantidad de horas de trabajo para cada alumno es de 16 horas semanales.
3. Bloques mínimos de trabajo de 4 horas consecutivas.
4. Se considera horario válido de trabajo de 9-17 hrs.
5. Cantidad de alumnos por grupo debe ser idealmente de 6, con un margen de diferencia de +- 1, dependiendo de la cantidad de alumnos que hay a nivel de curso.

En base a la restricción de bloques mínimos de 4 horas de trabajo, el primer paso a considerar es formular una tabla que indique la cantidad de bloques disponibles por día para

cada alumno. Lo anterior se cuantificará entonces de la siguiente manera:

1. Si el alumno tiene un bloque de 4 horas consecutivas disponible, cuenta como 1.
2. Si el alumno cuenta con dos de los bloques anteriormente mencionados disponible, cuenta como 2.

Lo anterior se puede esclarecer aún más tomando en cuenta la restricción de horario laboral de 9-17 hrs. Con ello, se puede decir que los alumnos poseen dos posibles bloques a elegir: AM o PM.

3.4.1. Algoritmo de registro de bloques libres por alumno

En base a la información expuesta anteriormente, se procede a la construcción de la primera matriz a utilizar, es simplificada de manera tal que indique si el alumno tiene el bloque AM, PM o ambos disponible.

Se tiene entonces que:

Datos: Cantidad de alumnos en curso
Resultado: Cantidad de bloques disponibles por día para cada alumno del curso
inicializar matriz;
para cada *alumno al en curso C* **hacer**
  para cada *día* d *en semana W* **hacer**
    matriz[al][d] = bloque(s) seleccionados por alumno
  fin
fin

Algoritmo 1: Registro de bloques libres por alumno

Finalizando el proceso anterior, se tendrá un registro de todos los alumnos del curso, indicando todos los bloques disponibles para cada día de la semana. A modo de ejemplo, y para poder visualizar de mejor manera el resultado, se tendrá lo siguiente:

Nombre	Lunes	Martes	Miércoles	Jueves	Viernes
Diego	-	AM	AM	PM	[AM,PM]
Pedro	AM	AM	PM	PM	-

Tabla 3.1: Ejemplo de output generado por proceso anterior

Así como se muestra en el ejemplo, se tendrá un listado de todos los alumnos con sus respectivos bloques disponibles.

3.4.2. Algoritmo de conformación inicial de equipos

Al término del algoritmo anterior se tendrá la disponibilidad horaria para cada alumno, por tanto se puede proceder a la siguiente etapa del proceso. Se necesita realizar una serie de permutaciones para formar grupos en base a los bloques disponibles del proceso anterior. La idea principal del segundo algoritmo es ir formando grupos hasta que no quede ningún

alumno sin uno. El enfoque de esta solución será en primera instancia de fuerza bruta, ya que lo primero que se desea es llegar a una solución, a pesar que ésta no sea óptima en las primeras iteraciones. El esquema general del segundo análisis se compone a grandes rasgos de la siguiente manera:

Datos: Cantidad de bloques disponibles por día para cada alumno del curso
Resultado: Matriz con conformación inicial de equipos realizada
seleccionar seis alumnos al azar para comparar horarios;
de ellos, elegir a uno para comenzar proceso;
para cada *alumno al en subgrupo SG* **hacer**
  para cada *día* d *en semana W* **hacer**
    si *bloque coincide con bloque de alumno seleccionado* **entonces**
      matriz[d][subgrupo] = [combinación alumnos comparados, hora del día];
    fin
  fin
fin

Algoritmo 2: Conformación inicial de equipos

Continuando con el ejemplo del *Algoritmo 1*, el output generado por el *Algoritmo 2* sería el siguiente:

Equipo	Lunes	Martes	Miércoles	Jueves	Viernes
A	-	[(Diego, Pedro), AM]	-	[(Diego, Pedro), PM]	-

Tabla 3.2: Combinaciones posibles entre Diego y Pedro durante la semana

En base a esta primera combinación, se tiene que Diego y Pedro coinciden para trabajar los días martes y jueves. A pesar que ambos tienen bloques disponibles el día miércoles, no coinciden en el horario del día y por tanto el output es vacío.

Para poder distinguir entre aquellos alumnos que han tenido problemas en unirse a grupos y aquellos que aún no han sido seleccionados, se tendrán dos conjuntos de alumnos: los **FA** (Failed Attempt) y los **NG** (No Group), donde el primero corresponde al conjunto de alumnos a los que se les ha intentado unir a un grupo pero se ha fallado y el segundo a aquellos que aún no han pasado por el proceso de intentar unirse a un grupo.

3.4.3. Algoritmo de conformación final de equipos en base a disponibilidad

Con el algoritmo anterior se busca obtener una combinación de pares de alumno por día, separados en bloques AM o PM. De esta manera se cumple automáticamente con la restricción que un alumno no quede solo en algún bloque, ya que la cantidad mínima de alumnos que se encontrarán disponibles para un bloque dado es dos. Una vez terminado este proceso, lo que se tendrá es una lista de combinaciones de alumnos para todos los días de la semana. Es en esta fase donde se debe hacer una corroboración: cada alumno del subgrupo elegido debe estar presente en al menos cuatro bloques de la semana. Si tiene más de cuatro bloques se hará una nota de ello, pues ese alumno tiene más flexibilidad para ser cambiado

en caso de necesitarlo. Sin embargo, si el alumno cuenta con menos de cuatro bloques de trabajo disponibles para este subgrupo, entonces está claro que no es apto para formar parte de ese equipo y se debe buscar otro grupo para él.

En cuanto a su reemplazo, se buscará en primera instancia dentro del grupo FA, siempre y cuando no se haga la comparación con alguien que ya no haya calzado previamente, de lo contrario se buscará en el grupo NG.

Datos: Matriz con conformación inicial de equipos realizada
Resultado: Matriz con conformación final de equipos realizada
para cada *equipo* e *en clase* C **hacer**
  para cada *alumno al en equipo* E **hacer**
    si *bloques en semana* $>= 4$ **entonces**
      alumno calza dentro del equipo y además puede ser cambiado;
    en otro caso
      $nuevoMiembro = buscarIntegrante(FA, NG)$
    fin
  fin
fin

Algoritmo 3: Conformación final de equipos en base a disponibilidad por bloques

3.4.4. Cálculo de AL

Una vez finalizado el proceso anterior, se tendrán armados todos los equipos del curso. Sin embargo, para lograr obtener un valor de AL no basta con que todos los grupos estén armados, es necesario saber también qué tan bien formado quedó el equipo. Para ello se considerarán los siguientes aspectos de cada uno:

- Cantidad de alumnos que trabajan a la vez en un mismo bloque horario.
- Cantidad de combinaciones que hay entre los alumnos a lo largo de la semana.

El primer punto es importante por lo que se ha mencionado con anterioridad, mientras más miembros del grupo estén trabajando a la vez, mayor productividad se tendrá en ese bloque. Ahora bien, sin desmerecer lo anterior, existirán instancias en las que simplemente no se podrá tener a una alta cantidad de alumnos trabajando a la vez; es aquí donde es importante el segundo punto, ya que al menos se debe buscar producir la mayor cantidad de interacción entre los de miembros del equipo a la semana. El escenario que se busca evitar es que un alumno A y otro alumno B siempre trabajen juntos sin interactuar con otros miembros del equipo, ya que se estaría produciendo un escenario desventajoso de compartimiento de conocimiento dentro del equipo.

Lo que se hará entonces es asignar un puntaje a cada uno de los equipos en base a las dos condiciones recién mencionadas. A lo que se apunta es que todo el equipo trabaje en los mismos bloques y en consecuencia eso hará que haya un máximo de interacción entre todos los miembros del equipo en la semana. Si N corresponde a la cantidad de miembros que debería tener un equipo, y k es la cantidad real de miembros que pueden trabajar a la vez, entonces el ideal que se busca es:

- Cantidad de alumnos trabajando en un mismo bloque = N
- Cantidad de combinaciones entre alumnos = $\underset{2 \leq k \leq N}{max} \binom{N}{k}$

En base a lo anterior, se definen rangos de valor para cada condición. En el caso de la primera, se calculará el porcentaje de $\frac{k}{N}$. Mientras tanto, para la segunda se utilizará una escala con factor de 10, donde cada escalón de combinaciones en base a k variará en 10 %.

Ejemplo cálculo de AL

Para lograr entender de mejor manera cómo se calcula el valor de AL en base a las condiciones recién mencionadas, se dará un ejemplo. Suponiendo que en el grupo formado los seis alumnos pueden trabajar a la vez, se está frente a un caso fácil e ideal donde el valor de AL será igual al 100 %. El caso en que no ocurre lo anterior es más interesante. A modo de preparación, se hará el cálculo para saber cuál k da el máximo número de combinaciones para el N dado. En este escenario, ese k resulta ser k=*3*, por lo tanto se tiene la siguiente escala de puntajes:

k	5	4	3	2	1
%	0	80	90	80	0

Tabla 3.3: Escala de porcentajes asignados a combinaciones en base a k con factor del 10 %

Como se puede apreciar, la combinación de 5 y 1 tiene un valor asignado de cero, esto se decidió así ya que no se puede tener a un alumno trabajando de manera individual. Adicionalmente, se mencionó que el AL de los seis miembros juntos es 100 %, por lo que la siguiente mejor opción de tres miembros juntos tendrá un valor de 90 %, la siguiente de 80 % y así sucesivamente.

Suponiendo entonces que los alumnos del grupo solo pueden trabajar en grupos de a tres para un bloque dado, el cálculo a realizar es el siguiente:

$$AL = prom[(\frac{3}{6} * 100) + escala[\binom{6}{3}]] = prom[50 + 90] = 70\,\% \tag{3.1}$$

Lo anterior entrega un valor de AL para un periodo de la semana en particular (ej: lunes en horario AM), por lo que este proceso se debe repetir para cada periodo de la semana y así poder calcular el promedio de todos estos valores para la semana completa. Habiendo terminado el proceso se tendrá entonces el valor de AL para el equipo.

3.4.5. Cálculo de AG

Habiendo finalizado lo anterior, se tendrán armados todos los equipos del curso con su respectivo AL y por ende se puede proceder a la parte final de la conformación: obtener el valor de AG. El objetivo principal de este valor, es minimizar la desviación estándar de los

equipos en base a sus valores de AL. Sin embargo, calcular la desviación no siempre será suficiente, ya que pueden ocurrir casos donde todos los grupos tienen armonías similares y solo uno difiere, como también puede suceder que todos los grupos tienen armonías lo suficientemente separadas, donde el valor de desviación sería similar en ambos casos. Es por ello que se requiere de un proceso adicional para cerciorar que se está llegando al mejor valor posible. Por tanto, se procederá a hacer comparaciones entre grupos para mejorar el valor de AG, independiente si estos cambios perjudican los valores de AL.

Lo que se hará es comparar al mejor grupo con el peor grupo (en términos de AL), haciendo un cambio entre el alumno menos flexible del peor grupo, con el más flexible del mejor grupo. Este cambio afecta directamente al AL de ambos equipos, pero dada la naturaleza de los alumnos intercambiados, debiera lograrse una conformación que sea más justa a nivel de curso (que como ya se mencionó anteriormente, es prioridad en comparación a una buena conformación local de equipo). Para acotar la necesidad de llevar a cabo el proceso anterior, éste se realizará de manera continua siempre y cuando el valor de la desviación estándar, una vez que se haya realizado el cambio, sea mayor a 5. Habiendo llegado a la desviación estándar mencionada, el valor de AG final para el proceso corresponderá al promedio de todos los AL del curso. Finalmente, cabe mencionar que este es un proceso que opera por fuerza bruta y que si bien no es un proceso óptimo, de todos modos ayudará a emparejar a los equipos del curso.

Capítulo 4

Implementación de la solución

4.1. Tecnologías consideradas

Al definir que se trabajará en el desarrollo de una aplicación web, es importante considerar las tecnologías que lograrán resolver la totalidad del problema de la mejor manera posible.

4.1.1. Flask

Cuando se trata de desarrollo de aplicaciones web, es inevitable encontrarse con Flask [9] como un fuerte contendiente a utilizar, sobretodo si se desea usar Python como lenguaje de programación en el lado del servidor. Esto debido a que es un framework liviano y fácilmente expandible, el cual tiene como objetivo ser minimalista, dejando a criterio del usuario gran parte de las decisiones a tomar cuando se trata de qué librerías incluir en el desarrollo del sistema. Es por ello que una de las principales ventajas de Flask recae sobre su flexibilidad y posibilidad de customización.

4.1.2. Django

Dicho lo anterior, siempre que se consulte sobre los mejores frameworks de desarrollo web en Python, en conjunto a Flask aparecerá Django [7]. Una diferencia importante entre ambos es que Django es un framework web full-stack, vale decir, su utilidad va más allá del lado del servidor y puede usarse para desarrollo del lado del cliente igualmente. Lo anterior implica que Django posee una variedad de características que vienen incluidas desde un comienzo y que en Flask se deben desarrollar de manera aparte. Uno de los grandes atractivos de Django está en su módulo de administración, el cual tiene una interfaz propia y viene listo para usar desde el momento de instalación. Por ende, se considera como una de las alternativas más populares con la cual realizar un proyecto web. Además, hace uso de prácticas ágiles de trabajo sobre una arquitectura avanzada y escalable.

4.1.3. Node

Mientras que las opciones anteriores se centran en el lenguaje Python, existe también una alternativa potente cuando se trata de JavaScript: NodeJS [13]. Está diseñado para construir

aplicaciones de redes escalables, agregando robustez y poder al momento de desarrollar un servidor. A diferencia de Django no viene con tantas funcionalidades listas desde ya, pero similar a Flask es flexible en cuanto a customización y escalabilidad.

4.1.4. ReactJS

Si bien Node otorga todo lo necesario para poder desarrollar el backend de una aplicación, cuando se trata de una alternativa frontend hecha en JavaScript, ReactJS [15] es una opción altamente popular. Es una librería utilizada en la creación de interfaces; fue creada por Facebook y es usada en todos sus proyectos, por lo que cuenta con un alto nivel de soporte, comunidad y aceptación a nivel mundial. Los elementos son altamente reutilizables, lo cual minimiza la cantidad de código repetido y por ende disminuye la complejidad para futuras mejoras.

4.1.5. Material Design

Dentro de los frameworks de React a utilizar en el desarrollo frontend de una aplicación, se encuentra Material Design [12], uno de los más usados por desarrolladores. Al momento de utilizar React, Material se utiliza casi por defecto junto a él. Dentro de sus ventajas se encuentra el que es ampliamente customizable, tiene una amplia comunidad y por tanto amplia documentación, además de múltiples ejemplos de variada dificultad que utilizan librerías externas para cumplir con una multitud de necesidades.

4.1.6. Antd

A pesar que Material entrega una amplia variedad de ejemplos y alternativas de diseño de componentes, existe otra alternativa igualmente popular que proporciona un estilo más empresarial al diseño, Antd [2]. Dentro de las ventajas de este framework se encuentra su facilidad de uso, variedad de componentes, amplia comunidad y constante soporte.

4.1.7. Docker

Es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker [8] empaqueta software en unidades estandarizadas llamadas *contenedores* que incluyen todo lo necesario para que el software se ejecute, esto incluye librerías, herramientas de sistema, código y tiempo de ejecución. Al utilizar Docker, se puede implementar y ajustar la escala de aplicaciones rápidamente, con la certeza de saber que el código se ejecutará independiente del ambiente en el que se encuentre.

4.2. Tecnologías escogidas

Habiendo listado las tecnologías que se analizaron para el desarrollo de la aplicación, a continuación se explicará cuáles fueron elegidas y el por qué de la decisión.

4.2.1. Backend

Para el desarrollo de backend se optó por trabajar con Django, dado que viene por defecto con una gran cantidad de características que facilitan el desarrollo del sistema, además de contar con un robusto módulo admin desde el primer momento.

4.2.2. Frontend

Para el desarrollo de frontend, si bien se podría haber utilizado el mismo framework para cliente de Django, se optó por utilizar ReactJS en conjunto a Material Design. El hecho que ReactJS sea una herramienta utilizada ampliamente en el mundo tecnológico e impulsada por Facebook, entrega una sensación de confianza y seguridad de soporte constante debido a la amplia comunidad de desarrolladores que lo utilizan.

Finalmente, cabe señalar que tanto backend como frontend serán montados en contenedores Docker, para así garantizar la funcionalidad de ambos lados sin importar el entorno en el que se monte.

4.3. Interfaces y procesos

A continuación se detallarán las distintas interfaces del sistema, dando a conocer las que existen además de explicar los procesos que se pueden llevar a cabo en ellas, dependiendo del perfil de usuario que se encuentre logueado. Debido a que el login al sistema es a través de la utilización de U-Pasaporte, la interfaz es aquella estándar para todas los sistemas que utilizan este módulo y no se incluirá en el listado de vistas. Adicionalmente, para poder visualizar los elementos de las vistas de mejor manera, se excluirá el header y footer de las imágenes.

4.3.1. Carga de información de curso(s)

Esta vista es única para el perfil administrador y en ella se puede cargar la información de un curso de *Proyecto de Software* en particular.

Para cargar la información de un curso, se debe ingresar la siguiente información:

- Año
- Semestre
- Sección

Adicionalmente, se debe cargar un archivo Excel que contenga la información de los integrantes del curso. Es importante destacar que este archivo Excel debe poseer el formato del archivo de integrantes que se descarga desde la plataforma de U-Cursos. La carga exitosa o errada de información del curso es informada al usuario mediante notificaciones.

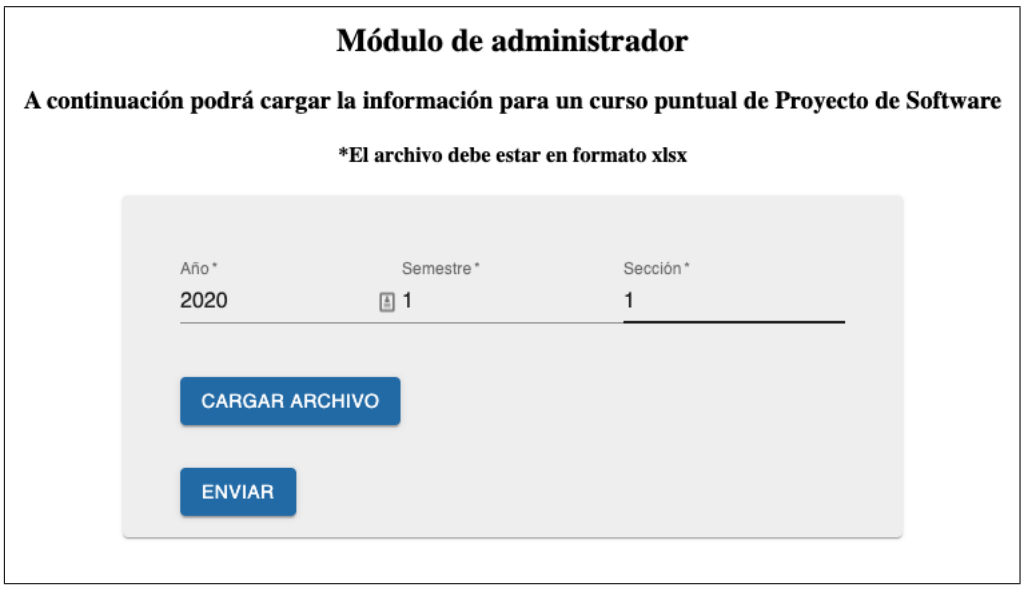

Figura 4.1: Vista de módulo administrador

4.3.2. Listado de cursos (Home)

Una vez que el usuario ingrese al sistema, sea éste de perfil alumno o docente, se encontrará con el listado de cursos a los cuales está asociado para el semestre en curso.

Perfil docente

Para el perfil docente se tendrá la siguiente vista:

Figura 4.2: Vista de cursos para el perfil docente

Como se puede observar, se detalla un listado de todos los cursos disponibles para el año y semestre en curso. Para cada una de las secciones disponibles se puede dar inicio al proceso de selección de horas tomando como fecha de término de éste, la fecha seleccionada en **Duración Proceso**.

Una vez iniciado el proceso, se puede optar a modificar la fecha de término de éste o bien ver el estado del proceso en curso.

Perfil alumno

Para el caso del alumno se tiene una vista similar a la anterior, en la cual se detallará el curso en el que se encuentra actualmente, además de información indicando si el proceso de selección de horas se ha iniciado o no. En caso que éste esté iniciado, se informará la cantidad de días que quedan del proceso además de la opción de poder elegir su horario en caso que aún no lo haya hecho o modificarlo en caso que ya haya registrado horas. Finalmente, si el proceso se encuentra cerrado y la conformación de grupos ya fue realizada, aparecerá la opción para ver el grupo al que el alumno fue asignado.

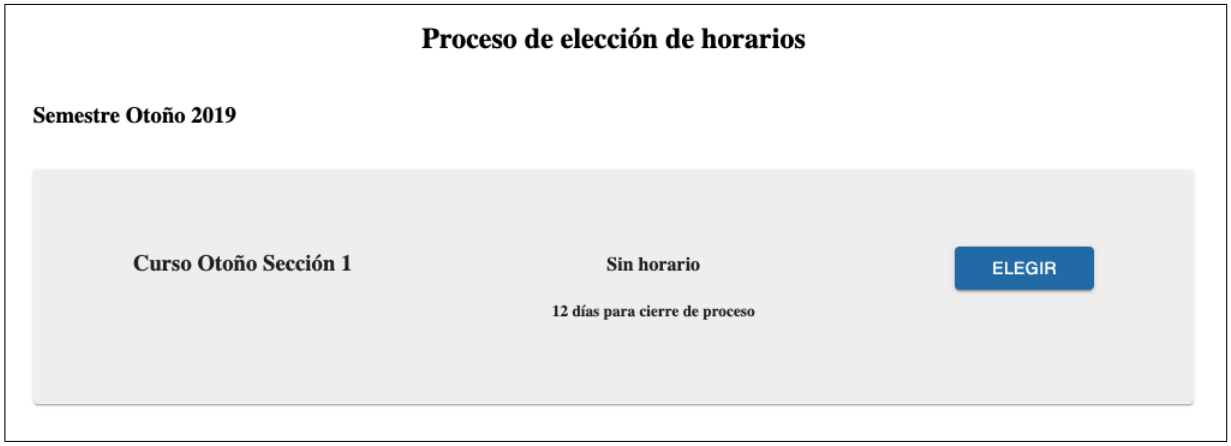

Figura 4.3: Vista de cursos para perfil de alumno sin selección de horas

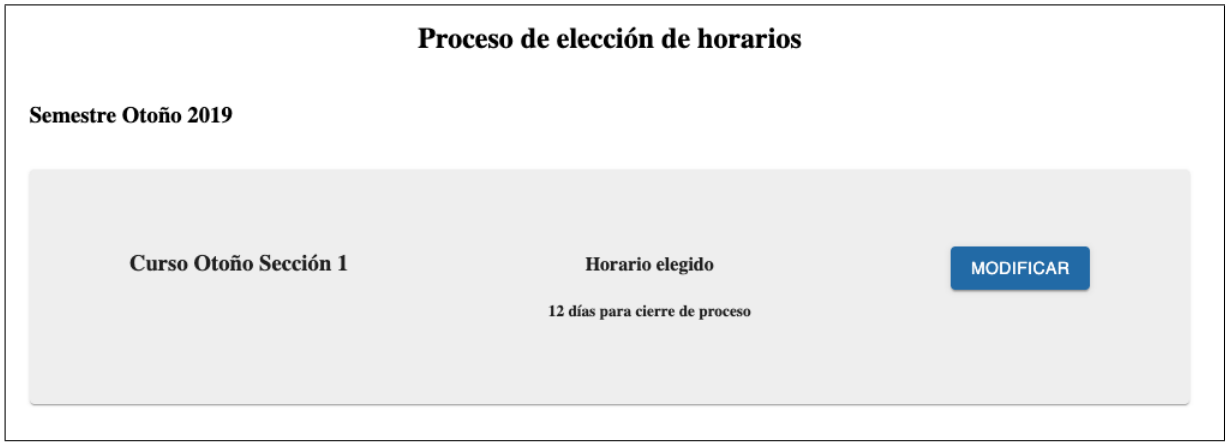

Figura 4.4: Vista de cursos para perfil de alumno con selección de horas y proceso aún abierto

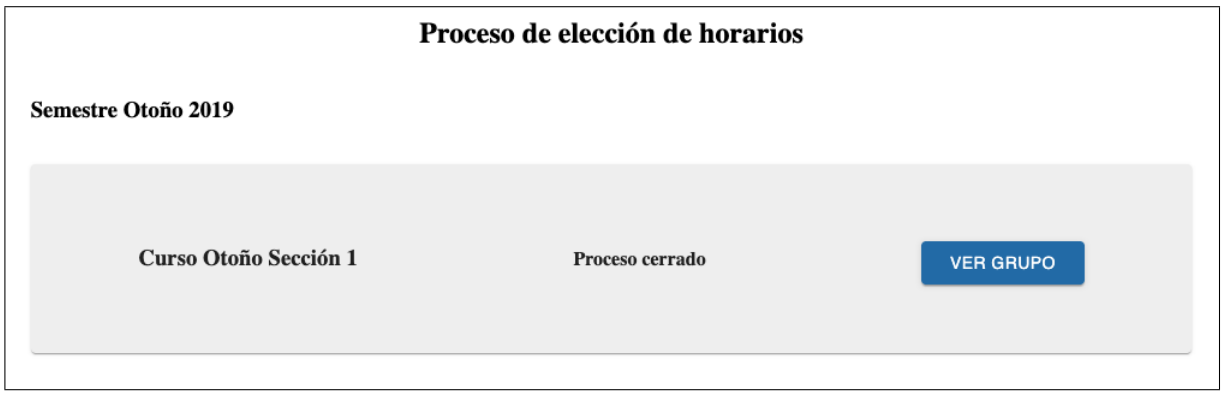

Figura 4.5: Vista de cursos para perfil de alumno con conformación de equipo disponible

4.3.3. Elección de horarios

Esta viste corresponde solo al perfil de alumno y permite que éste pueda seleccionar los bloques de horario que tendrá disponibles para trabajar en la semana.

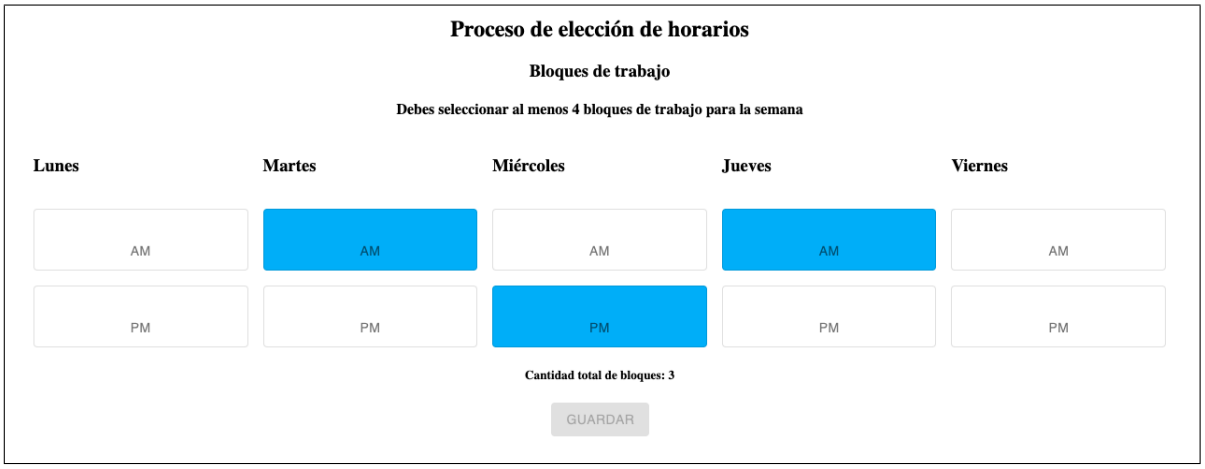

Figura 4.6: Vista de selección de bloques horarios con horario insuficiente y guardado bloqueado

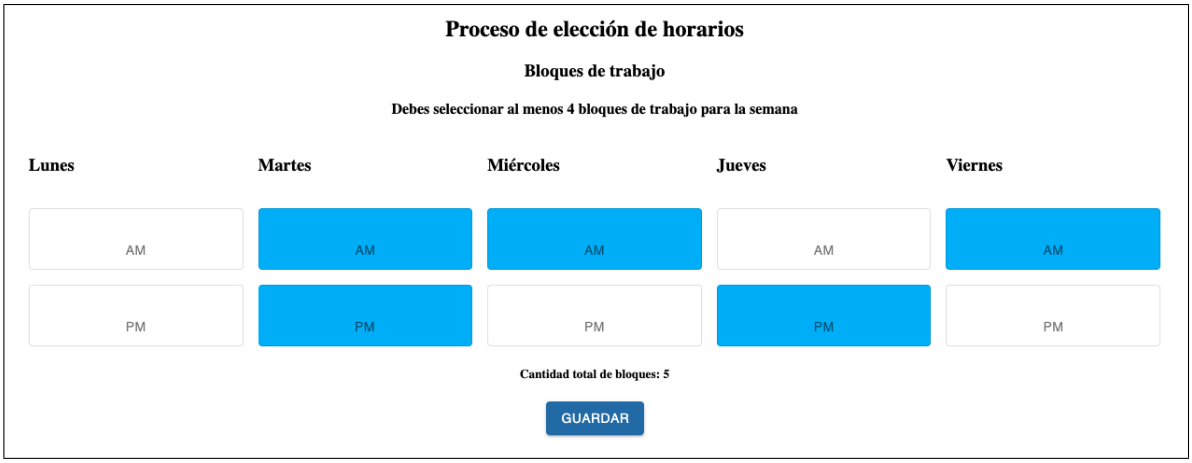

Figura 4.7: Vista de selección de bloques horarios con cantidad suficiente seleccionada

Como se puede apreciar en la figura, la interfaz deja claro al usuario cuando éste selecciona un bloque, indicando además cuántos han sido seleccionados hasta el momento. El botón para guardar la selección horaria se mantiene desactivado mientras no se registre la condición mínima de tener 4 bloques seleccionados mínimo en la semana.

4.3.4. Listado de progreso

Esta vista es exclusiva del perfil docente y en ella se puede ver el listado de todos los alumnos del curso, indicando el estado en el que se encuentran dentro del proceso de elección de horas. El estado puede ser *Contestado* o *No Contestado*.

Adicional a lo anterior, esta vista posee dos botones:

- **Finalizar proceso**

  Consiste en dar cierre al proceso de selección de horas de manera manual, sin haber llegado a la fecha de término estipulada al momento de haber iniciado el proceso.
- **Iniciar conformación**

  Consiste en ejecutar el algoritmo de conformación de equipos. Para ejecutar esta acción es necesario que el proceso de selección de horas haya finalizado.

Figura 4.8: Vista de progreso donde se indica si alumno ha registrado horarios o no

4.3.5. Equipos conformados

Vista exclusiva del perfil docente en la cual se despliegan los equipos conformados una vez que haya finalizado la ejecución del algoritmo de conformación de equipos.

En la interfaz se pueden apreciar todos los equipos conformados así como el valor de AL de cada uno y los integrantes que los componen. Adicionalmente, se puede apreciar el valor de AG del curso de acuerdo a los valores de AL generados. Finalmente, existe la posibilidad de modificar grupos de manera manual, donde el cambio de un integrante de un equipo a otro se puede hacer "arrastrandolo" (Drag n Drop). Los cambios de integrantes afectan los valores de AL y AG en tiempo real.

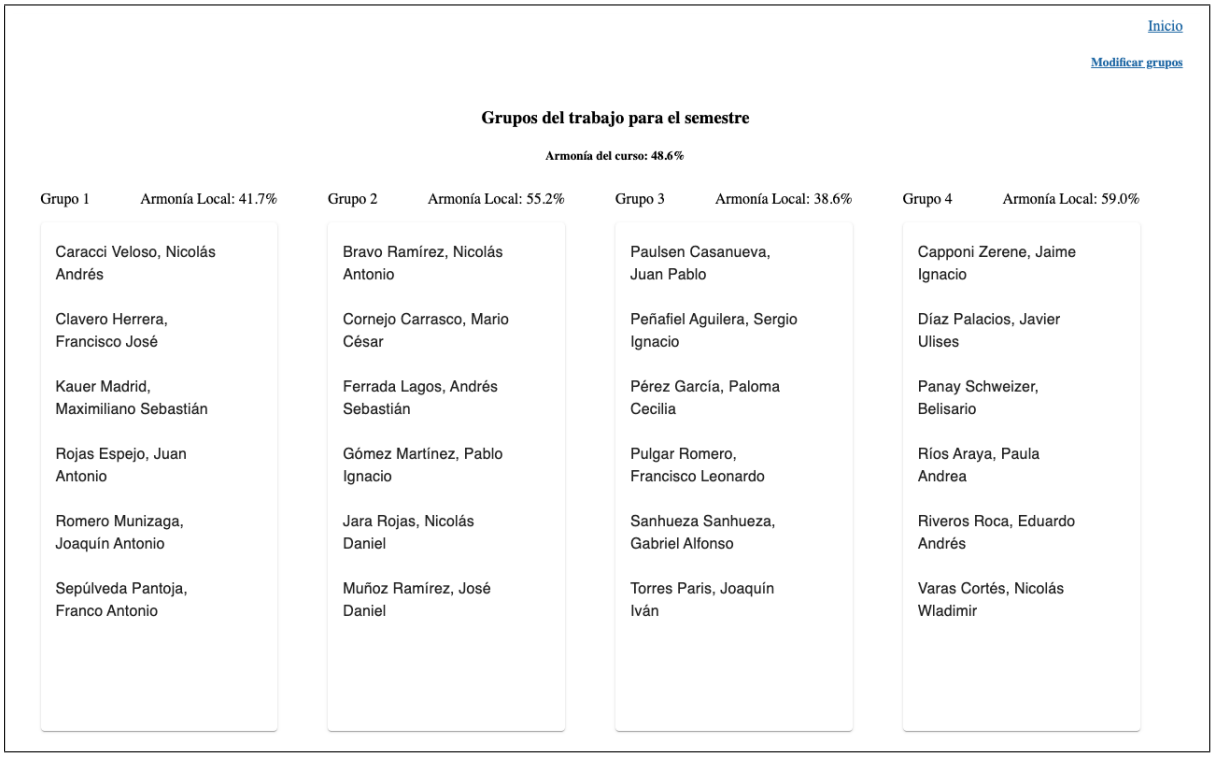

Figura 4.9: Vista de grupos conformados con sus respectivos valores de AL y AG

4.3.6. Equipo asignado

Finalmente, como vista exclusiva del perfil de alumno, éste podrá visualizar a cuál grupo fue asignado una vez finalizado el proceso de conformación de equipos.

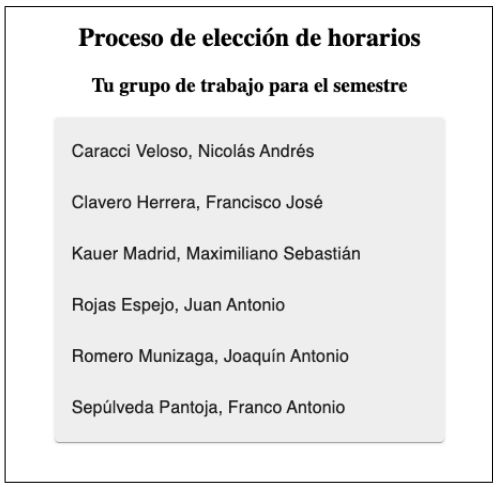

Figura 4.10: Vista de grupo al que alumno fue asignado

Capítulo 5

Validación

La elección de prueba de validación es una a la que se llegó a acuerdo en conjunto a la profesora Cecilia Bastarrica, quien es uno de los usuarios finales que utilizará el sistema. Como fue explicado al comienzo de esta memoria, el proceso para llevar a cabo la conformación de equipos se hace de manera manual y queda a carga de la profesora Bastarrica. Ella recibe la disponibilidad horaria de los alumnos a través de documentos de texto y se encarga de compilar estos horarios en una hoja excel, para luego analizar "al ojo" cuál es la mejor conformación a la que se puede llegar en base a los horarios entregados.

Tomando lo anterior en consideración, se ingresó al sistema a los alumnos de una realización anterior del curso. Una vez ingresados, se registraron los horarios que en su momento fueron compilados en el excel de disponibilidad horaria. Habiendo ingresado los horarios, se procedió a ejecutar el algoritmo de conformación de equipos para analizar los grupos formados y ver qué tan buenos fueron sus valores de AL y AG. Se hizo una nota de los grupos conformados además de los valores obtenidos.

Teniendo el registro de los grupos conformados por el sistema de manera automática, se procedió a "forzar" la configuración real que se llevó a cabo en esa oportunidad (la configuración realizada por la profesora). Al forzar esta configuración en el sistema, éste entregó los valores de AL de cada equipo y el valor AG a nivel de curso. Una vez obtenido estos resultados, se comparó con aquellos realizados por el sistema para analizar cuál era mejor.

Los resultados obtenidos se muestran a continuación:

	Resultado	
Tipo Config	σ	$prom(\%)$
manual	4.04	42.1
sistema 1	3.07	58
sistema 2	4.83	45.2
sistema 3	4.54	44.4

Tabla 5.1: Resultados de conformación de equipos, tomando en consideración configuración manual

Como se puede apreciar, resulta ser que la configuración manual realizada en aquél entonces estaba dentro del rango de configuraciones consideradas "justas" por el sistema. En cuanto a la conformación automática, esta se realizó tres veces para el mismo grupo de alumnos, con el propósito de lograr obtener una mejor noción de la calidad de equipos que conforma el sistema en comparación a aquella que se realizó de manera manual. Por lo tanto se puede concluir que los resultados de la conformación automática de equipos son en promedio iguales y en algunos casos mejores que aquellos obtenidos de la conformación manual. Los registros de las configuraciones obtenidas en este proceso de validación se pueden encontrar en la sección de anexos.

Capítulo 6

Conclusiones y trabajo futuro

Actualmente la conformación de equipos para el ramo de *Proyecto de Software* se realiza de manera manual. Lo anterior se ha sustentado a través del tiempo debido a que la cantidad de alumnos que debían tomar el ramo, solía ser de menor escala. Hoy en día, esa cantidad de alumnos ha incrementado de sobremanera y es probable que siga ese aumento en los años que vienen. Es por ello que se requiere de un sistema que automatice la conformación de equipos para agilizar proceso, sin perjudicar la calidad de los equipos formados, a modo que estos sean lo más justos a nivel de curso.

Con dicho propósito en mente, se ha desarrollado un sistema que conforma equipos utilizando como variable la disponibilidad horaria de sus alumnos. La base de este sistema funciona en torno a la creación de un algoritmo que genera dos valores claves: la *armonía local (AL)* y la *armonía global (AG)*. La primera mide qué tan bien formado quedó un grupo a nivel local de equipo, tomando en cuenta la cantidad de personas que pueden trabajar a la vez en los bloques de trabajo de la semana. La segunda mide qué tan dispersos y balanceados quedaron dichos grupos a nivel de curso.

Para corroborar la calidad de los grupos formados se llevó a cabo una prueba tomando a los integrantes de un curso anterior, junto a los horarios de disponibilidad que indicaron en su momento, e ingresándolos al sistema para aplicar el algoritmo elaborado y así obtener los valores de AL y AG correspondientes. Adicional a lo anterior, se forzó la conformación de equipos a la configuración realizada de manera manual en esa oportunidad, para así medir si los valores de AL y AG obtenidos fueron mejor o peor que aquellos realizados de manera automática en el sistema. Los resultados obtenidos indicaron que la configuración manual realizada en aquella oportunidad fue de buena calidad. Adicionalmente, se observó que los equipos que el sistema conforma de manera automática son en promedio igual de buenos o incluso mejores a la configuración manual y evidentemente más rápidos de lograr, demorando menos de un minuto en realizar la configuración.

Tomando en cuenta lo anterior, se ha creado un sistema que utiliza un algoritmo enfocado en la disponibilidad horaria de sus alumnos que cumple lo siguiente:

- Automatiza la conformación de equipos.

- Dentro de la conformación misma, asegura que los integrantes de éste no trabajen solos y siempre estén acompañados de al menos otro integrante.
- Asigna un valor al equipo (AL) para así poder medir qué tan bien se llevó a cabo el proceso de conformación.
- Asigna un valor al curso (AG) que indica el nivel de justicia entre los equipos formados.
- Permite la modificación de los equipos creados en primera instancia, recalculando también los valores de AL y AG.

Si bien el sistema cumple con el objetivo central que se planteó al comienzo, quedan ciertos aspectos donde es posible mejorar.

1. Mejoras de usabilidad

   Las interfaces del sistema fueron creadas con la visión de alumno del memorista (habiendo pasado por el curso en su momento) y la profesora Bastarrica, quien realiza la conformación. Queda pendiente entonces la realización de una posible ronda de encuestas preguntando a usuarios de los distintos perfiles si están de acuerdo con lo diseñado y si tienen sugerencias de cómo mejorar el sistema.
2. Mejoras al algoritmo de AL y AG

   El algoritmo que genera los valores mencionados contiene decisiones que fueron consideradas óptimas por el memorista en base a los datos disponibles. Queda abierta la posibilidad de mejorar dicho algoritmo mediante la utilización de otros métodos matemáticas de optimización y considerando quizás la inclusión de otras variables (nivel de conocimiento del alumno, carga académica del semestre, entre otros).
3. Extensión del sistema a otros cursos

   Si bien el enfoque del sistema estaba centrado en la conformación de equipos para el curso de *Proyecto de Software*, existe también la posibilidad de expandir su uso a otros ramos que requieran conformar equipos. Se deja abierta la posibilidad entonces a que la utilización del algoritmo generador de AL y AG sea utilizado en otro contexto y utilizando otras variables, como se mencionó en el punto anterior.

Bibliografía

[1] Hayward P. Andres. A comparison of face-to-face and virtual software development teams. *Team Performance Management: An International Journal*, 8(1/2):39–48, 2002.

[2] Antd. Ant design - the world's second most popular react ui framework. `https://ant.design/`, (2019). [En línea; última visita 17 de Diciembre 2019].

[3] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for agile software development, 2001.

[4] Rebecca M. Chory-Assad. Classroom justice: Perceptions of fairness as a predictor of student motivation, learning, and aggression. *Communication Quarterly*, 50(1):58–77, 2002.

[5] Russell Cropanzano and Jerald Greenberg. *Progress in Organizational Justice: Tunneling Through the Maze*, volume 12, pages 317–372. 01 1997.

[6] Morton Deutsch. *Distributive justice : a social-psychological perspective / Morton Deutsch*. Yale University Press New Haven, 1985.

[7] Django. Django: The web framework for perfectionists with deadlines. `https://www.djangoproject.com/`, (2019). [En línea; última visita 17 de Diciembre 2019].

[8] Docker. Docker: Enterprise container platform. `https://www.docker.com/`, (2019). [En línea; última visita 17 de Diciembre 2019].

[9] Flask. Flask | the pallets projects. `https://www.palletsprojects.com/p/flask/`, (2019). [En línea; última visita 17 de Diciembre 2019].

[10] Boris Kabanoff. Equity, equality, power, and conflict. *Academy of Management Review*, 16(2):416–441, 1991.

[11] Gerald S. Leventhal. *What Should Be Done with Equity Theory?*, pages 27–55. Springer US, Boston, MA, 1980.

[12] MaterialUI. Build beautiful products, faster. `https://material.io/`, (2019). [En línea; última visita 17 de Diciembre 2019].

[13] Node. Node.js. https://nodejs.org/en/, (2019). [En línea; última visita 17 de Diciembre 2019].

[14] L. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, 4(04):345–361, jul 1978.

[15] ReactJS. React – a javascript library for building user interfaces. https://reactjs.org/, (2019). [En línea; última visita 17 de Diciembre 2019].

[16] Daniel Rodriguez, M Sicilia, Elena Barriocanal, and Rachel Harrison. Empirical findings on team size and productivity in software development. *Journal of Systems and Software - JSS*, 85, 03 2012.

[17] W. W. Royce. Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press.

[18] Luis Silvestre, Sergio F. Ochoa, and Maira Marques. Understanding the design of software development teams for academic scenarios. In *34th International Conference of the Chilean Computer Science Society, SCCC 2015, Santiago, Chile, November 9-13, 2015*, pages 1–6, 2015.

[19] TeamBuilder. Project teambuilder | source and deploy employee project teams instantly. https://teambuilder.symbasync.com/, (2019). [En línea; última visita 22 de Diciembre 2019].

[20] Woven. Woven - the best calendar app for busy professionals. https://woven.com/, (2019). [En línea; última visita 17 de Diciembre 2019].

Anexos

Anexo A - Resultados de prueba de validación

A continuación se adjuntan las imágenes de los grupos conformados por el sistema y sus respectivos valores de AL y AG.

Figura 6.1: Valores de AL y AG obtenidos al realizar la configuración manual de la profesora

Para esta configuración se obtuvo una desviación estándar de 4.04 y un AG promedio de 42.1 %.

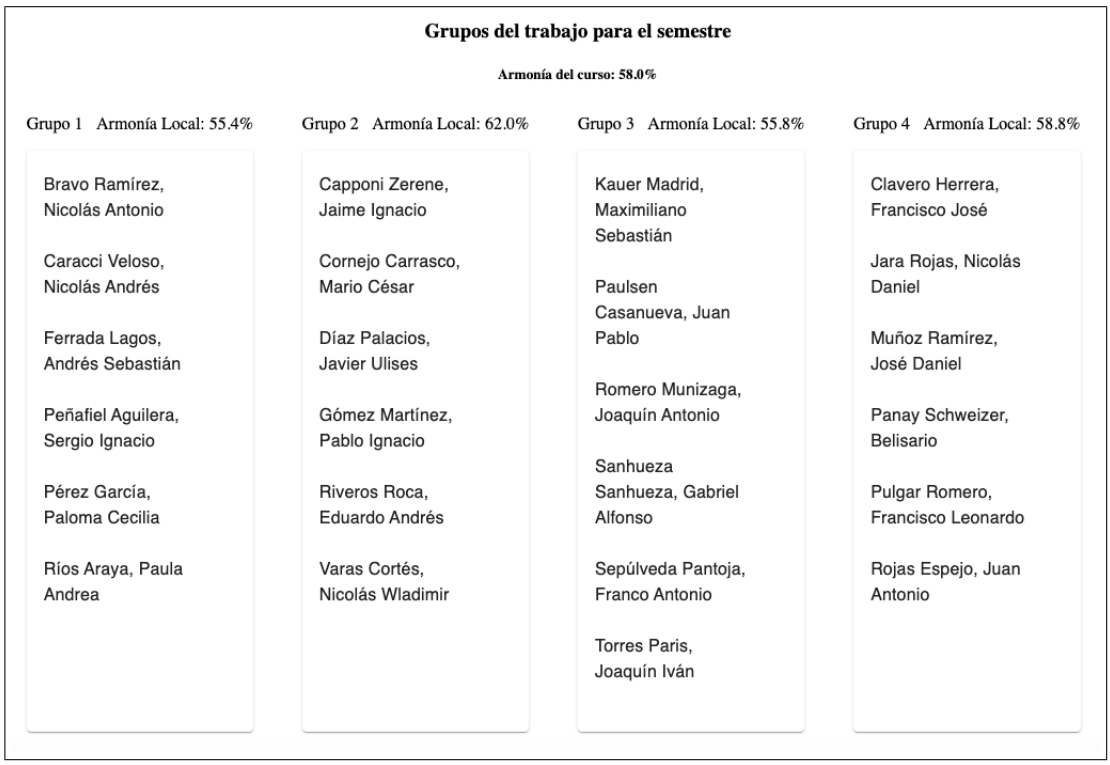

Figura 6.2: Valores de AL y AG obtenidos al realizar la configuración automática por primera vez

La primera configuración realizada por el sistema arrojó valores bastante positivos, con una desviación estándar de 3.07 y un AG promedio de 58 %.

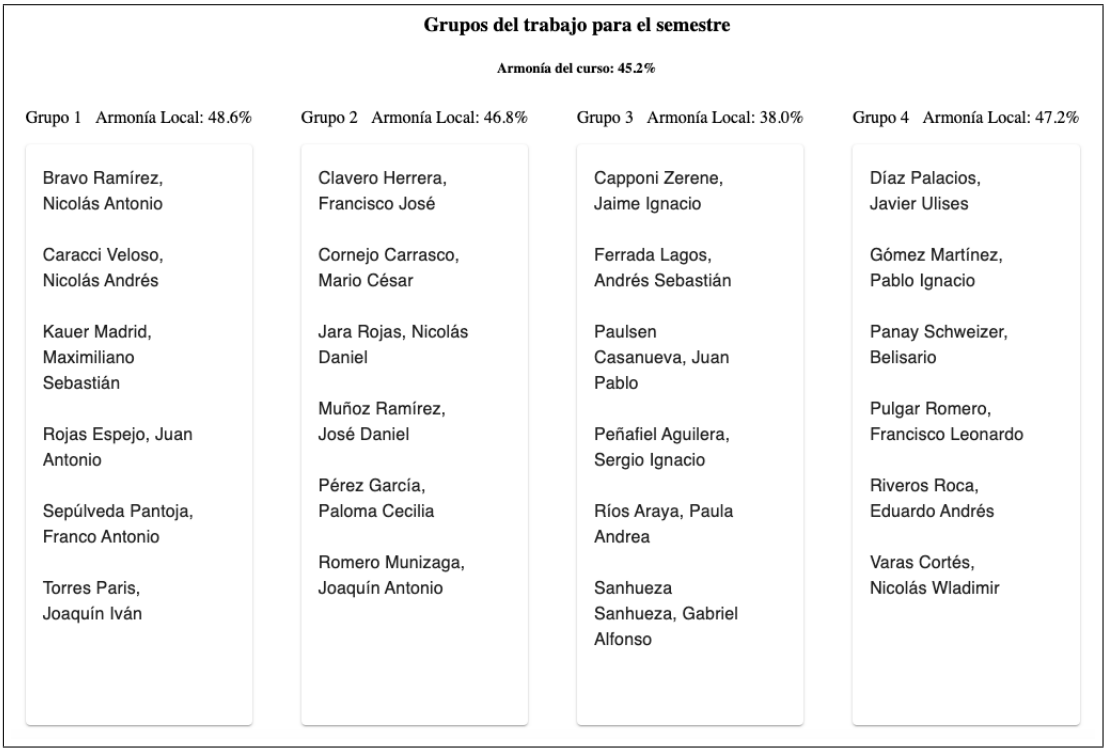

Figura 6.3: Valores de AL y AG obtenidos al realizar la configuración automática por segunda vez

La segunda configuración realizada por el sistema, si bien no arrojó valores tan buenos como el caso anterior, de igual manera se obtuvo una desviación estándar de 4.83 y un AG promedio de 45.2 %, un caso levemente peor a la configuración manual.

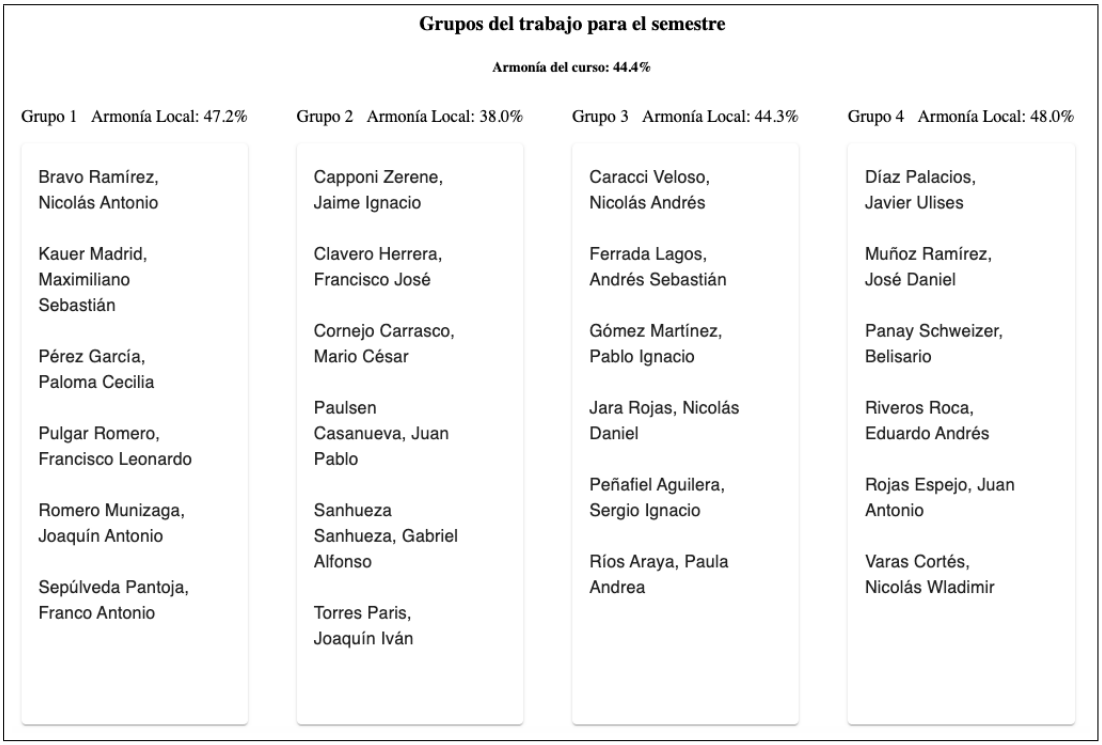

Figura 6.4: Valores de AL y AG obtenidos al realizar la configuración automática por tercera vez

Finalmente, la tercera configuración realizada por el sistema, arrojó una desviación estándar de 4.54 y un AG promedio de 44.4 %, un caso promedio de los anteriores.