ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

UNA HERRAMIENTA WEB PARA LA PLANIFICACIÓN DE HORARIOS Y DE ASIGNACIONES DE AULAS

A WEB-BASED TOOL FOR TIMETABLE SCHEDULING AND CLASSROOM ALLOCATION

Realizado por
Mario Padilla Carrión
Tutorizado por
Eduardo Guzmán de los Riscos
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, Diciembre de 2015

Fecha defensa:
El Secretario del Tribunal

Resumen

Este proyecto consiste en el desarrollo de una aplicación web para cubrir una necesidad del centro universitario: la necesidad de gestionar los horarios y la asignación de aulas cada inicio de curso de una forma automática e inexistente en la actualidad.

El objetivo principal ha sido que la aplicación fuera lo más intuitiva posible y que tenga una facilidad de uso que motivase su utilización. Para ello nos hemos decantado por un diseño moderno y una interfaz amigable, con una serie de características tanto visuales (distribución de la información, códigos de colores), como de uso (sistema Drag&Drop).

La aplicación tiene cuatro módulos principales:

Un primer módulo para el manejo de los horarios, el cual nos permite la construcción de un horario para un grupo determinado evitando cualquier tipo de conflicto.

El segundo módulo para la asignación de grupo a aulas, de forma que nos permite tener un control de los espacios de la facultad.

El tercer módulo, con una fuerte relación con el segundo, nos permite la asignación de asignaturas optativas a aulas.

Por último, el cuarto módulo es administrativo y nos permite gestionar el manejo de datos de la aplicación.

Además, la aplicación cuenta con diversas páginas de información.

Una de las ventajas más importantes que tiene esta aplicación es la automatización a la hora de realizar todas las comprobaciones necesarias para hacer cualquier asignación.

Debido a la envergadura del proyecto, optamos por realizar este proyecto conjuntamente entre tres personas para ser capaces de ofrecer un producto lo más completo posible sin salirnos de los límites de este trabajo de fin de grado.

Palabras claves

Gestión, planificador, horarios, aulas, asignaturas, aplicación web, Materialize, SCRUM, XP (eXtreme Programming), Python, Django, JQuery, Backbone, Marionette.

Abstract

This project consist of the development of a web application to fulfil a need of the university: the need to manage schedules and assigning classrooms to the beginning of each course in an automatic way and currently inexistent.

Our main target have been that the application as intuitive as possible and providing it of an ease way to use, motivating its usage. Therefore we have opted for a modern design and friendly interface, with a series of features, both visual (information distribution, colour fonts) like of use (Drag & Drop system).

The application has three main modules:

A first module for managing schedules, which allows us to build a schedule for a particular group avoiding any conflict.

The second module is about the groups and classrooms assignment, which allows us to have a control spaces of the faculty.

The third module, with a strong relationship with the second, allows the assignment of optional subjects to classrooms.

Finally, the fourth module is an administration site that allows us the data handling

In addition, the application has several pages of information.

One of the most important advantages of this application is the automation when carrying out any checks required for any assignment.

Due to the size of the project, we chose to carry out this project jointly between three people to be able to offer a complete product as were possible without leaving the limits of this work of end of grade.

Keywords

Management, scheduler, timetable, subjects, web application, Materialize, SCRUM, XP (eXtreme Programming), Python, Django, JQuery, Backbone, Marionette.

Contenido

1. Introducción

Motivación

En la actualidad el proceso que se sigue a la hora de planificar un nuevo curso, es decir la confección de horarios y la asignación de la ocupación de las aulas, se realiza de una forma poco automatizada, de forma que todas las restricciones que es necesario controlar a la hora de realizar estas acciones se controlan de una manera estrictamente manual. Esto resulta una tarea excesivamente pesada.

De ahí surge la necesidad y motivación de una herramienta que automatizara todos estos procesos y facilitara la realización de estas tareas.

Objetivos del Proyecto

Tras un arduo trabajo se ha conseguido desarrollar un proyecto que integra las nuevas tecnologías y el desarrollo de una aplicación web para la administración y organización interna de aulas, horarios, grupos, asignaturas y bloques, adaptado a las necesidades de la Universidad de Málaga. Se pueden distinguir cuatro módulos principales y una serie de pequeñas funcionalidades y páginas de información que expanden los módulos.

Se han cumplido todos los objetivos individuales, generalmente:

- Sección de acceso e identificación.
- Módulo de administrador para crear, borrar, editar y consultar los datos.
- Sección para asignar/desasignar asignaturas optativas a aulas.
- Sección para consultar la información referente a los bloques (aula, asignaturas y horarios) y exportar dicha información a PDF.

Es cierto que por el camino se han redefinido algunos detalles planteados inicialmente. Por ejemplo, el hecho de hacer que la aplicación fuera totalmente adptativa “responsive”. Aunque en un principio era esta la idea, a lo largo del desarrollo nos dimos cuenta de que no tenía sentido utilizar la aplicación en dispositivos de pequeño tamaño como puede ser un móvil, porque debido a la forma en la que está estructurada la funcionalidad no resultaría práctica ni fácil de usar. Por ello la aplicación no está preparada para un correcto visualizado en estos dispositivos. Si bien es cierto que la aplicación sí está preparada para que funcionalmente no presente ningún tipo de problemas en dispositivos móviles, por lo que la aplicación se puede utilizar perfectamente en una tablet o dispositivo de mayor tamaño a un móvil convencional.

Estructura de la memoria

La memoria se encuentra dividida en siete apartados principales:

- Un primer apartado de introducción que permite al lector hacerse una idea general sobre el proyecto. En esta primera parte se describe brevemente que motivó y con qué objetivo se realizó el proyecto. De esta forma cualquier persona que lea este apartado puede decidir si está interesado en aprender más sobre él o no.

- El segundo apartado hace un repaso rápido acerca de las diferentes tecnologías que se han utilizado a lo largo del desarrollo, lo cual podríamos incluirlo también como una parte introductoria.

- El tercer apartado entra más en profundidad en el proyecto en sí. Al comienzo se realiza una descripción general de todo el proyecto, de forma que el lector pueda tener una visión global de la interfaz de trabajo y su funcionamiento al interactuar con la aplicación, explicando las diferentes funcionalidades que la conforman y aportando algunos detalles que son considerados importantes, sin entrar en detalles técnicos.
  Tras ello, se explica la metodología de desarrollo que se ha seguido durante el transcurso del proyecto; dándose detalles del funcionamiento y de los puntos positivos y negativos que han surgido de su utilización.

- El cuarto apartado trata sobre la especificación y análisis de requisitos, explicando cómo se ha realizado la formalización del proyecto de forma grupal.

- El quinto apartado explica la parte de diseño del sistema. En concreto el diseño de clases del sistema, el modelo de datos y del proyecto; así como su estructura. Seguidamente, se explica gráficamente parte del funcionamiento del sistema mediante algunos esbozos de los diagramas de secuencia y actividad.

- El sexto apartado, trata el desarrollo realizado, explicando los pasos que se han seguido desde el principio de la implementación hasta la finalización de la aplicación, visto desde la perspectiva individual del proyecto y abarcando algunos temas generales. Para terminar se mencionan las pruebas realizadas.

- Por último, el séptimo apartado trata las conclusiones del proyecto, los objetivos cumplidos y las dificultades encontradas. Para finalizar, se analizan las líneas futuras, sus posibles ampliaciones y las mejoras que podrían aplicarse en un futuro.

2. Tecnologías y Herramientas Utilizadas

Herramientas y aplicaciones para el desarrollo

VirtualBox

Es un software de virtualización, bajo el cual se ha instalado el sistema operativo, Ubuntu, empleado a lo largo del proyecto

Sistema operativo Ubuntu

Se ha trabajado con este sistema operativo debido a la buena compatibilidad y facilidad de uso en conjunto con las tecnologías posteriormente descritas.

Pip

Pip (Pip Install Package) es un gestor de paquetes que se utiliza para instalar y manejar paquetes escritos en lenguaje Python. Se ha utilizado fundamentalmente para instalar el framework principal sobre el que se ha trabajado y diferentes librerías que han sido necesarias durante el desarrollo.

Virtualenv

Es una herramienta que permite crear entornos aislados de Python. Esto permite que sea posible instalar una aplicación en un entorno virtual propio aislado del resto del entorno. Una ventaja que tiene esto es que evita posibles conflictos entre las dependencias de esta aplicación con los de otra, ya que existe la posibilidad de que diferentes aplicaciones requieran de diferentes versiones de las mismas dependencias.

VirtualenvWrapper

Es un conjunto de extensiones para Virtualenv. Proporciona una serie de herramientas y utilidades para administrar todos los virtualenvs creados en el sistema. Facilita mucho el manejo de virtualenvs desde la consola.

Pycharm

Es el entorno de desarrollo (IDE) sobre el que se ha realizado el desarrollo. Se trata de un IDE multiplataforma que se utiliza para programar en Python. Incluye integración con sistemas de control de versiones y soporta desarrollo web con Django. Que son algunos de los motivos por los que nos decantamos por utilizar este IDE.

Git

Es un sistema de control de versiones. Debido a su fácil integración con el IDE y a que todos los integrantes del grupo estábamos familiarizados con su uso nos decidimos por él.

MagicDraw:

MagicDraw es una herramienta de modelado de software que permite construir una gran variedad de diagramas de diferente tipo que resultan fundamentales a la hora de diseñar y planificar el diseño de una aplicación.

Asana

Es una herramienta de gestión y organización de equipos. Proporciona una serie de herramientas que facilitan la ejecución de diferentes metodologías de desarrollo.

Backend

Python

Python es un lenguaje interpretado que utiliza tipado dinámico. Se trata de un lenguaje multiparadigma, esto quiere decir que soporta varios paradigmas de programación como programación orientada a objetos, programación imperativa y programación funcional. Una de las filosofías fundamentales de Python es que la sintaxis favorezca un

código legible y en ello contribuyen algunas de las características de este. La elección por este lenguaje está basada en estas características ya comentadas y también en el hecho de que como grupo queríamos aprovechar la oportunidad que nos brindaba este proyecto para aprender a manejar tecnologías que durante el transcurso de la carrera no hemos tenido la suerte de aprender y que forman parte de la actualidad.

Django

Es un framework de Python que se utiliza para desarrollo web. Django se basa en el patrón de diseño Modelo-Vista-Controlador (MVC) aunque en este caso recibe el nombre Modelo-Plantilla-Vista (MTV como en Model-Template-View) debido a que en el caso de Python la 'C' correspondiente a Controlador es manejada por el mismo framework y realizan una interpretación distinta del patrón de diseño.

Es uno de los frameworks de desarrollo web de código abierto más completos de Python y además posee una amplia biblioteca de librerías y una extensa documentación que facilita en gran medida el trabajo a desarrollar.

SQLite

SQLite es un sistema de gestión de bases de datos relacional, contenido en una biblioteca escrita en C.

La biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones, reduciendo la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos.

Tastypie

Es un servicio web para Django, que permite crear una API para Django. Proporciona una abstracción conveniente y altamente personalizable para crear interfaces estilo REST. Para poder realizar estos accesos es necesario definir los recursos que van asociados a estos modelos, con un control total sobre lo que se expone o se accede, por ejemplo mediante filtros, lo cual permite abstraer la base de datos tanto como sea necesario.

Xhtml2pdf

Librería escrita en Python que permite transformar documentos HTML en archivos PDFs de forma sencilla y práctica.

Frontend

Jquery

Jquery es una librería de JavaScript diseñada para simplificar la interacción con los documentos HTML y los elementos del DOM desde el lado del cliente. También incluye funcionalidades un poco más complejas como la creación de animaciones o la posibilidad de utilizar fácilmente AJAX. Es una de las librerías de JavaScript más populares y más usadas del mundo.

Jquery UI

Es una librería de interfaces de usuario oficial de Jquery. Esta añade un conjunto de *widgets*, *plugins* y efectos visuales que pueden utilizar durante el desarrollo web para crear aplicaciones webs más completas e interactivas.

Jquery UI Touch Punch

Es una pequeña librería que permite que aprovechar las ventajas que proporciona Jquery UI en dispositivos móviles. Ya que a día de hoy estos dispositivos no detectan bien los eventos de toque lo cual no permite aprovechar Jquery UI en su máximo potencial.

Backbone

Backbone es una librería JavaScript que se basa en el paradigma Modelo-Vista-Presentación (MVP) que es una variación del Modelo-Vista-Controlador (MVC) e integra

este mismo en la vista de una aplicación web. Esta librería permite crear modelos que se corresponden a los que existen en la base de datos y colecciones de estos datos para poder trabajar con objetos que adquirimos desde la base de datos de una forma más sencilla.

Underscore

Es una librería JavaScript que proporciona una gran cantidad de funcionalidades. Es una dependencia de Backbone. Además permite renderizar contenido de manera dinámica en plantillas HTML.

Backbone-Tastypie

Backbone-Tastypie como su propio nombre indica es una librería que integra Backbone con Tastypie de forma que sincroniza los modelos de ambas partes. Esta librería sobrescribe varios métodos de Backbone de forma que podamos realizar llamadas a los recursos de Tastypie y encapsular los resultados que obtenemos en los modelos de Backbone. Y de esta forma podemos tener los datos sincronizados entre el servidor y el cliente en todo momento.

Marionette

En una librería que complementa y trabaja sobre Backbone. Proporciona una variedad de funciones y utilidades para completar aquellas partes en las que Backbone flojea un poco. Permite organizar el código de una forma más intuitiva y construir estructuras más ricas y formadas por pequeños componentes. Es muy potente para renderizar el contenido de forma dinámica en nuestras páginas web.

Materialize

Es un *framework* CSS dinámico (*responsive)* de reciente creación que proporciona un conjunto de componentes y animaciones basados en *Material Design* de Google.

3. Metodología y Desarrollo

Descripción General

Tal y como se introdujo al principio, el objetivo de este proyecto es satisfacer una necesidad presente en la Escuela, y para ello se planteó una aplicación web que hiciera más fácil una tarea que en este momento puede llegar a ser bastante tediosa y complicada.

Por ello esta aplicación web debe ser capaz de realizar algunas de esas tareas necesarias cada vez que se realiza la planificación de un nuevo curso académico.

Antes de nada, para familiarizarnos con la terminología empleada en el proyecto dado que dichos términos puede llegar a resultar un tanto confusos, y además, se usa su terminología en inglés, procedemos a definir algunos de los términos con los que vamos a trabajar a lo largo del proyecto y a definir sus traducciones de la siguiente forma:

Ejemplo: - **Término en español** (término en inglés). Explicación del término en el contexto actual

- **Grado** (Degree). Son las diferentes titulaciones de las cuales consta la Escuela o centro.
- **Mención** (Mention). Son las menciones que hay por cada Grado de la Escuela.
- **Curso** (Course). Son los cursos que hay por cada mención, diferenciados por años académicos.
- **Grupo** (Group). Son los grupos que contiene cada curso y están diferenciados por un código identificativo y el turno, que puede ser mañana o tarde.
- **Horario** o esquema (Schedule). Los horarios son diferenciados por semestre y cada grupo contiene dos horarios, uno por cada semestre.
- **Franjas horaria** (Shift). Son las diferentes franjas horarias que contiene nuestro horario, diferenciadas por día y hora. Cada horario consta de tres franjas horarias y cinco días lo que hace un total de quince “huecos horarios” donde asignar nuestras asignaturas.
- **Aula** (Classroom). Son las aulas en las cuales se imparten las clases a los grupos asignados y/o las asignaturas optativas o compartidas, ambas explicadas posteriormente.
- **Asignatura Optativa** (Optional Subject). A nivel organizativo, las asignaturas optativas están englobadas en bloques de optatividad, posteriormente explicado. Todas las asignaturas optativas que se engloban en un mismo bloque comparten un horario en común a diferencia de sus aulas, que debe ser obligatoriamente diferentes.
- **Bloque** (Block): son bloques que contienen asignaturas optativas. Hay dos tipos de bloques: los específicos, los cuales solo pueden asignarse a menciones específicas, y comunes, los cuales son comunes a todos los Grados.
- **Asignatura** (Subject). Una asignatura contiene la información sobre la materia que se imparte en ella. Técnicamente, a nivel organizativo, encontramos dos tipos especiales de asignaturas: asignaturas compartidas, las cuales comparten

obligatoriamente el mismo horario y asignaturas ligadas, las cuales comparten obligatoriamente tanto el horario como el aula.

Una vez familiarizados con la terminología pasamos a describir el proyecto de forma general.

Como ya se ha dicho con anterioridad, el proyecto consta de 4 módulos principales para cumplir con este propósito.

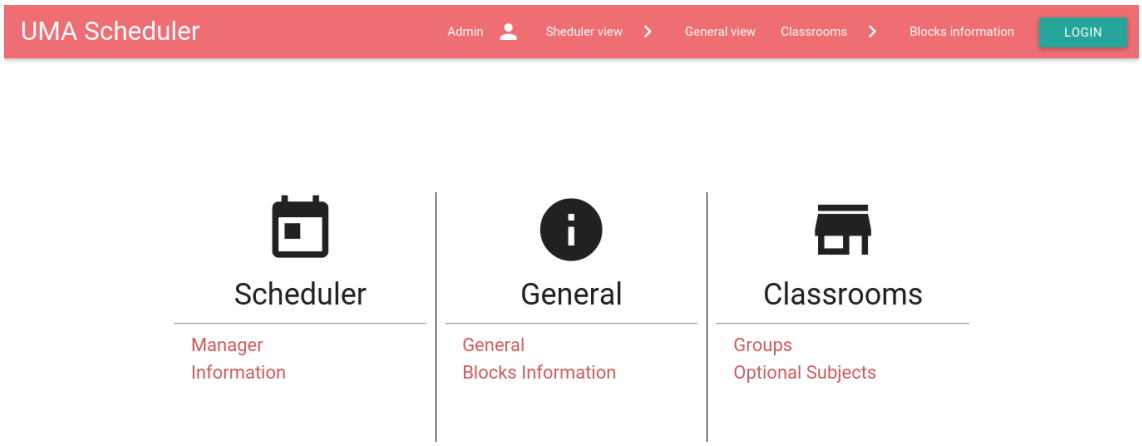

Ilustración 1. PÁGINA DE INICIO

En primer lugar tenemos el módulo para el manejo de horarios. En este módulo será posible crear horarios para los grupos que existen en el sistema. A través de un sistema Drag&Drop podremos ir asignando las diferentes asignaturas a las franjas horarias que creamos convenientes. La aplicación se encargará de realizar las comprobaciones pertinentes entre las que se encuentran el no poder asignar dos veces la misma asignatura al mismo día, o no asignar más franjas de las necesarias a una asignatura, lo cual dependerá del número de créditos de esa asignatura. Ya dependiendo del tipo de asignatura se comprobarán otros tipos de restricciones. Ya que en el caso de tratarse de asignaturas ligadas, asignaturas compartidas o asignaturas optativas, su asignación afecta a las asignaturas de otros grupos, por lo que hay que asegurarse de que todas ellas cumplen los requisitos. También seremos capaces de intercambiar asignaturas de forma que las restricciones se cumplan para ambas asignaturas y podremos eliminar asignaciones realizadas previamente. Por último, la aplicación nos permite seleccionar cuál de las franjas asignadas a una asignatura es la franja de “grupo grande”, que se corresponde a la franja en cuyas horas se impartirá la clase teórica o magistral de dicha asignatura.

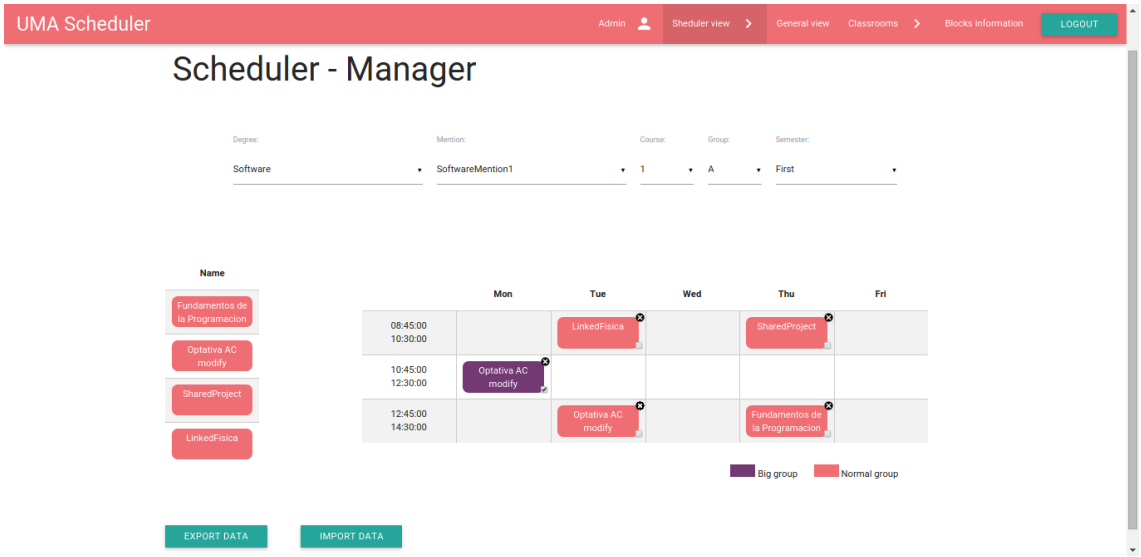

Ilustración 2. PRIMER MÓDULO

En un segundo módulo podremos tener un control sobre los espacios del centro en relación a los grupos, es decir, podremos asignar los distintos grupos a las aulas existentes, teniendo en cuenta tanto la capacidad del aula como la ocupación de la misma. De esta forma evitamos conflictos en los que se imparta más de una asignatura a la vez en la misma aula. En este caso aunque estemos asignando grupos a aulas también tenemos en cuenta que las aulas pueden tener asignaturas optativas asociadas y evitamos que se generen conflictos también de esta forma. En esta vista nos basamos en un código de colores para transmitir la información necesaria para un uso eficiente del módulo. De esta forma indicamos con un color cuándo un aula se encuentra vacía, otro color cuando se encuentra ocupada por otro grupo y un último color para cuando solo están ocupada por otras asignaturas optativas. Igual que en el anterior módulo, utilizamos un sistema Drag&Drop a la hora de realizar las asignaciones. Y ya a nivel interno la aplicación se encarga de comprobar que todas las restricciones son satisfechas antes de consumar la asignación.

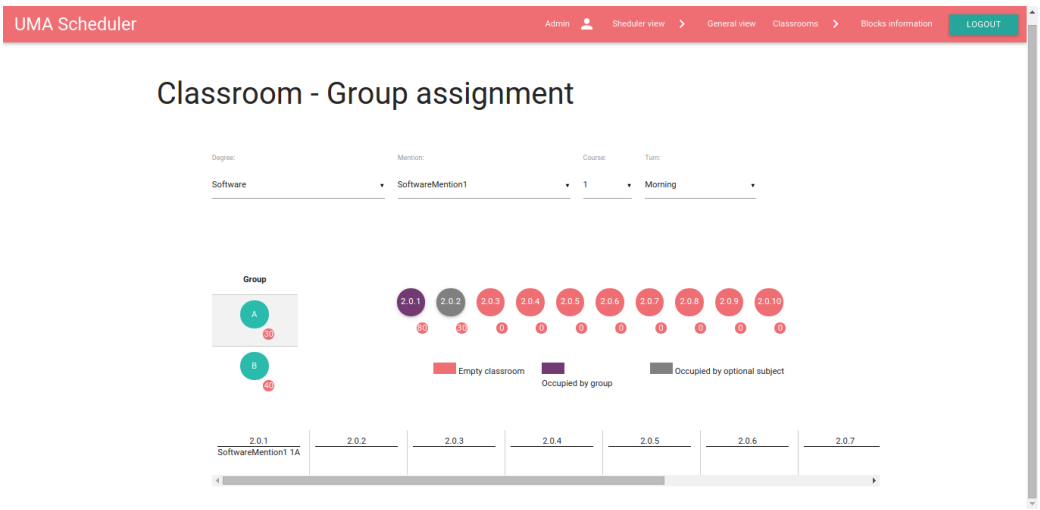

Ilustración 3. SEGUNDO MÓDULO

El tercer módulo nos proporciona un control sobre la distribución de las asignaturas optativas, distribuidas por bloques, en relación con las aulas del centro. Dicho de otra forma, nos permite asignar las asignaturas optativas a las aulas/espacios de nuestra Escuela. Se contemplan varias restricciones que serían tanto la capacidad del aula como las asignaturas optativas previamente asignadas a las aulas debido a la restricción de que un aula no puede albergar más de una optativa del mismo bloque, por incompatibilidad de horario. Mediante los filtros obtenemos los bloques los cuales pueden ser seleccionados mostrando las asignaturas optativas pertenecientes a dicho bloque.

Para seguir el axioma de obtener una interfaz amigable, la vista sigue un código de colores de forma que al seleccionar un bloque, este, se colorea y muestra las asignaturas optativas que alberga. Dichas asignaturas siguen un patrón de colores, amarillo cuando no están asignadas a un aula y verde en caso contrario. Al deslizar el puntero sobre la asignatura se despliega información sobre el estado de la asignatura, su capacidad en términos de alumnado y, si procede, el aula al que pertenece y la opción de desvincularla de la misma. Al pulsar sobre dicha asignatura se realizan dos acciones: por un lado, las posibles aulas tornan en diferentes colores indicando visualmente si puede asignarse a ella, si es demasiado pequeña o si ya contiene una asignatura del mismo bloque; y por otro lado, la asignatura seleccionada se transforma en una bola que puede arrastrarse para ser asignada a las diferentes aulas; posteriormente se muestra un mensaje temporal que nos indica lo sucedido.

Igual que en el anterior módulo, utilizamos un sistema Drag&Drop a la hora de realizar las asignaciones. Y ya a nivel interno la aplicación se encarga de comprobar que todas las restricciones son satisfechas antes de consumar la asignación.

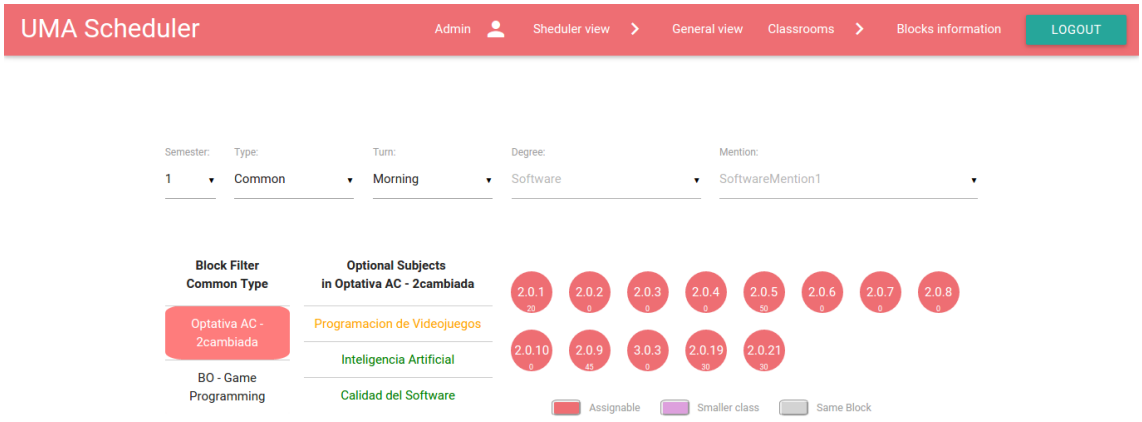

Ilustración 4. TERCER MÓDULO

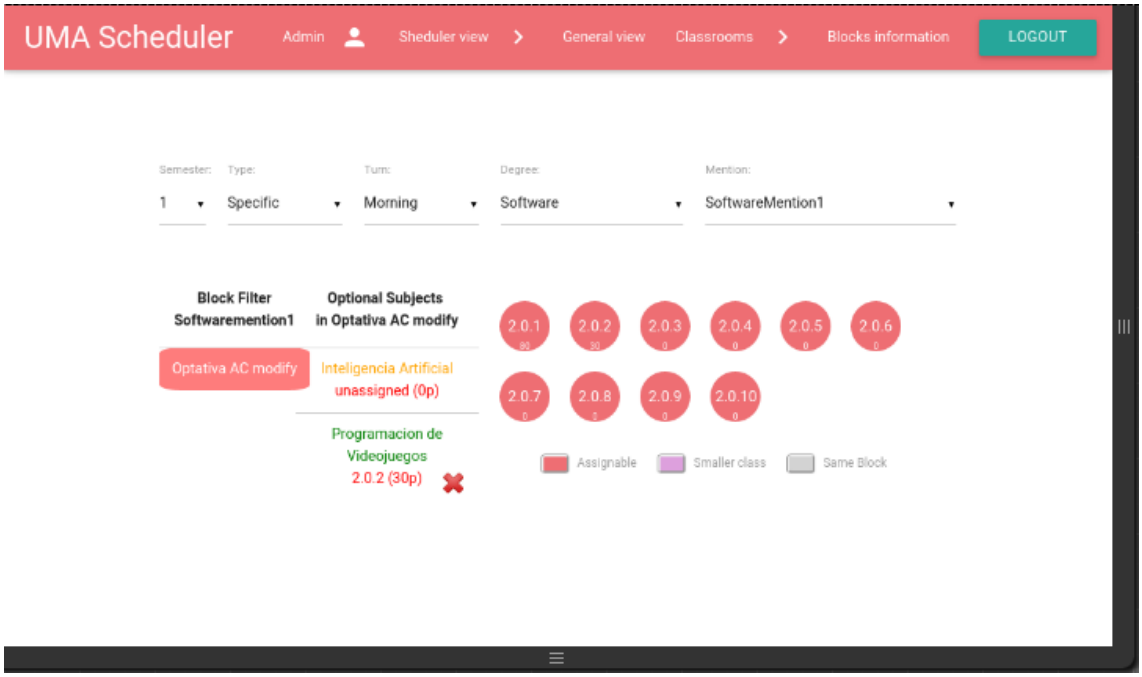

Ilustración 5. TERCER MÓDULO (INTERFAZ IPAD)

Para terminar, nuestro cuarto y último módulo permite tener un control completo para leer, crear, borrar y editar todos los datos de nuestra aplicación. Para este módulo se mantuvo un diseño robusto que no comparte la filosofía de la aplicación pero que en términos de tiempo era el menos prioritario focalizándose más en la funcionalidad del mismo o parte lógica (*backend*). Cada "objeto" o clase requiere una lógica muy específica o con unas restricciones muy detalladas, por ello debemos evitar inconsistencias para permitir la correcta ejecución de la aplicación. Posteriormente, se explicará técnicamente el esfuerzo y complejidad que supuso y los problemas que surgieron.

Ilustración 6. CUARTO MÓDULO (ADMINISTRACIÓN)

Aparte de estos cuatro módulos tenemos una serie de vistas de información para complementar a estos módulos.

Una vista de información sobre los horarios, en esta vista se podrá consultar los horarios construidos en el primer módulo comentado anteriormente. A través de los selectores es posible filtrar para ir viendo los horarios de cada grupo dividido por semestres. En esta vista también se incluye una opción para poder exportar el conjunto de todos los horarios a pdf, de forma que se pueda generar un documento con toda la información para poder imprimirla.

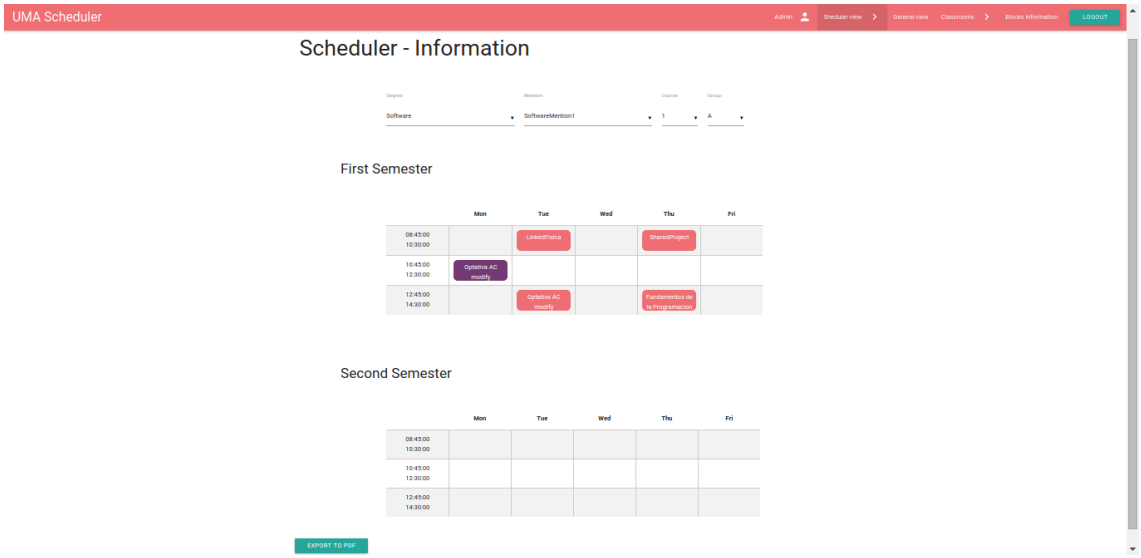

Ilustración 7. INFORMACIÓN DE HORARIOS

Otra vista llamada Visión General, en la cual se puede visualizar la ocupación en todo momento de las aulas. Para ello se tiene la capacidad de filtrar por el semestre, el día de la semana y el turno (mañana o tarde) y se mostrará una tabla en la que podremos ver qué hay en cada aula dividido por horas.

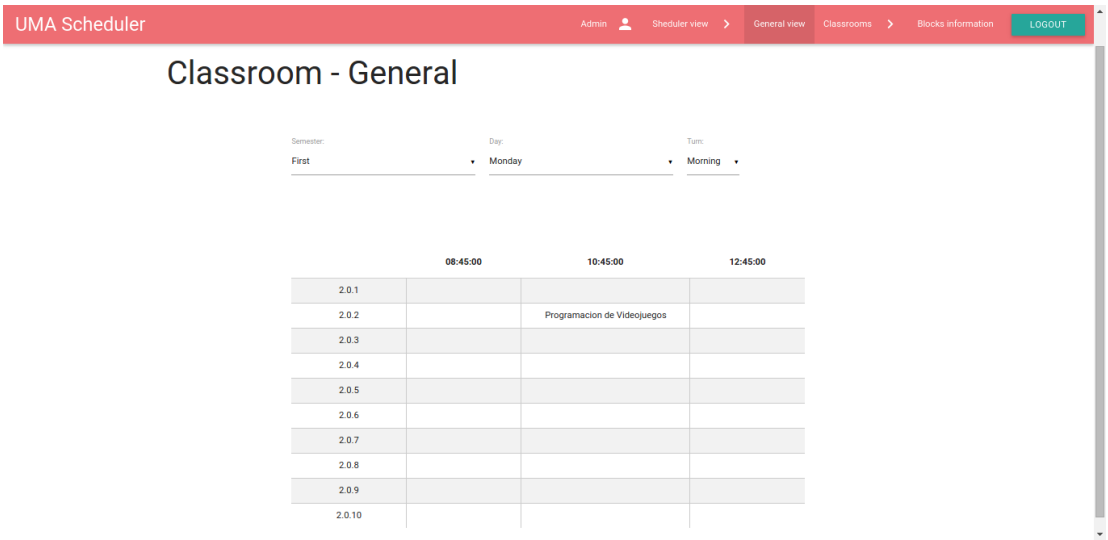

Ilustración 8. VISIÓN GENERAL

Una última vista que permitirá visualizar la información de los bloques de optativas, se mostraran divididos por bloques de optativas comunes y específicas, pudiendo filtrarse los bloques específicos por Grado y Mención a través de los selectores.

La información a su vez se dividirá por semestres y mostrara tanto el bloque de optatividad, junto con el horario y días en los que se imparte, así como las asignaturas que contiene y el aula al que han sido asignadas cada una de ellas.

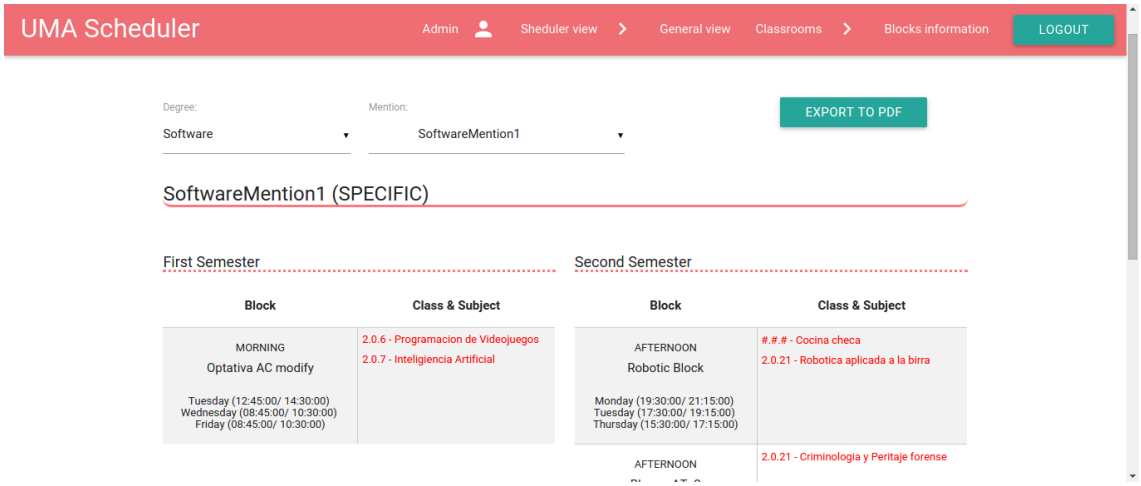

Ilustración 9. INFORMACIÓN BLOQUES DE OPTATIVIDAD

Todo ello puede ser exportado a PDF, el cual ha sido generado manualmente para respetar un estilo concreto.

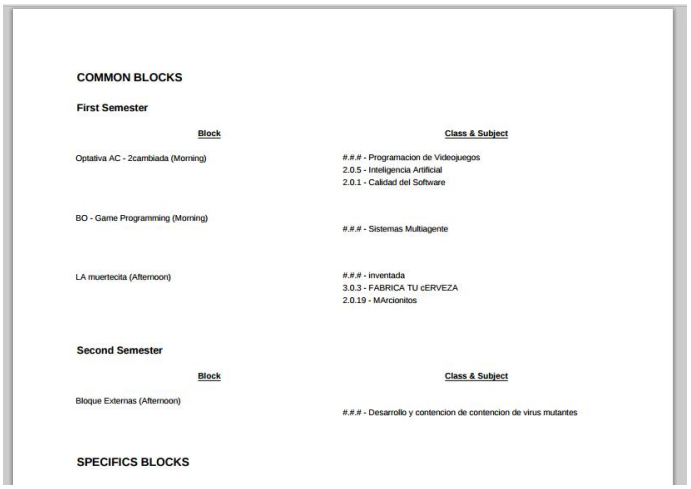

Ilustración 10. EJEMPLO PDF GENERADO

La última vista, no por ello menos importante, se trata del formulario de acceso a la aplicación que sigue un diseño adaptativo (*responsive)*, moderno y fiel a las nuevas tecnologías.

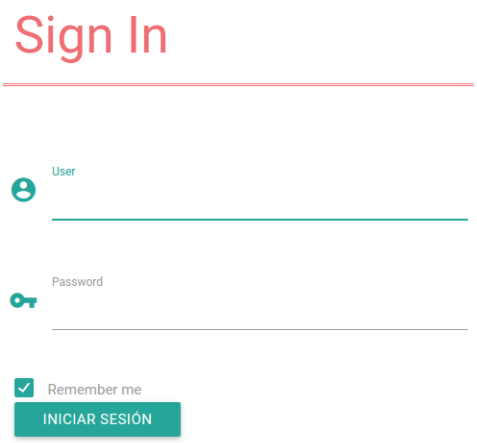

Ilustración 11. PÁGINA DE INGRESO

Organización y Metodología

La elección de la metodología de trabajo fue una de las más importantes de la fase inicial del trabajo. Esto se debe a que al tratarse de un proyecto en grupo la forma de organizarse tiene una importancia superior a un proyecto individual.

El proyecto se ha desarrollado siguiendo una metodología de trabajo ágil, *SCRUM* (nombre con el que se denomina a los marcos de desarrollo ágiles), con un *sprint* o periodo de dos semanas.

El motivo por el que se ha seguido un desarrollo *SCRUM* es una de sus claves principales: el reconocimiento de que durante un proyecto los clientes pueden cambiar de idea sobre lo que quieren y necesitan, y que los desafíos impredecibles no pueden ser fácilmente enfrentados de una forma predictiva y planificada. Por otro lado, ha sido elegido en base a experiencias previas obtenidas a través de las prácticas realizadas a lo largo de la carrera. Y también porque consideramos que tener objetivos claros a corto plazo motiva en gran medida el esfuerzo y permite que el proyecto avance de una manera más fluida.

El rol de *Stakeholder* o cliente ha sido cubierto por nuestro tutor del proyecto, aunque realmente era el nexo de unión con el cliente, mediante el cual se han ido especificando los requisitos y realizando las revisiones en sucesivas reuniones.

El conjunto de tareas que han formado parte de cada sprint se determinan durante la reunión de *Sprint Planning*. Durante esta reunión se han decidido las tareas que querían verse completadas durante el siguiente *sprint*. Dichas reuniones se hicieron generalmente por videoconferencia debido a que uno de los integrantes del grupo se encontraba estudiando en el extranjero, por ello se podrán distinguir dos equipos de desarrollo distintos. Las reuniones eran celebradas al comienzo de cada *Sprint*, en las cuales se hacía una evaluación muy breve del *sprint* que acababa de terminar y se

definían las tareas del nuevo *sprint*. Luego, durante la semana intermedia se hacía un leve repaso para saber cómo iban avanzando las tareas.

Para gestionar de una forma más eficaz todo este procedimiento se utilizó la herramienta Asana. Esta herramienta permite crear un proyecto e ir añadiéndole tareas con una fecha de inicio y fin. De esta manera se puede llevar una monitorización de la evolución de los *sprints* y del proyecto en general. Para ello, Asana proporciona algunas estadísticas muy útiles.

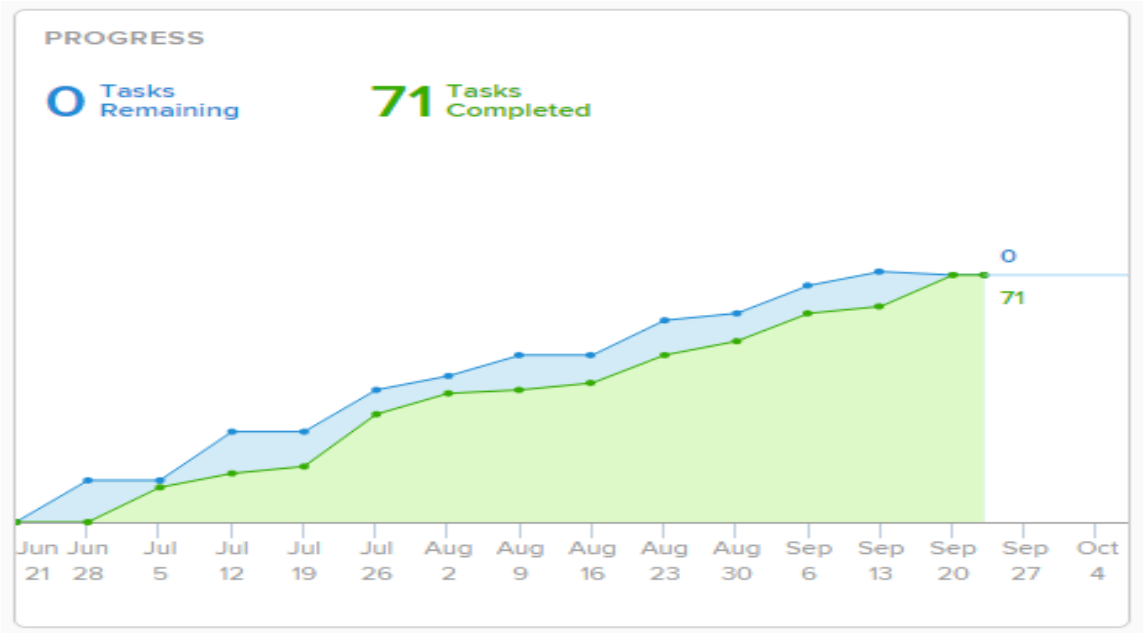

Ilustración 12. GRÁFICO DE SPRINTS DEL PROYECTO (DESARROLLO PARCIAL)

En la gráfica podemos observar la evolución durante todo el proceso de desarrollo anterior al 2 de Octubre de 2015, donde se produjo la entrega parcial del proyecto por parte de uno de los equipos de desarrollo. En azul, se contemplan las tareas completadas pendientes y en color verde aquellas que ya fueron completadas. Lo ideal sería que cada dos semanas la línea azul y la verde intersecaran lo que significaría que todas las tareas fueron realizadas pero debido a una mala estimación se crearon algunas tareas de una envergadura demasiado grande para la duración del sprint y eso se ha ido arrastrando a lo largo del proyecto.

Este seguimiento ha resultado muy útil porque permitía tener una idea global de la velocidad a la que avanzaba el proyecto. Además conforme pasaban los *sprints* las estimaciones de las tareas se realizaban con más precisión debido a la experiencia obtenida. Aunque por otro lado, el tener *sprints* en teoría tan cortos a veces producía problemas, porque cualquier imprevisto hacía que se complicara mucho cumplir los plazos de los *sprints*.

A continuación se puede observar el gráfico definitivo de la evolución completa y total del proyecto en conjunto:

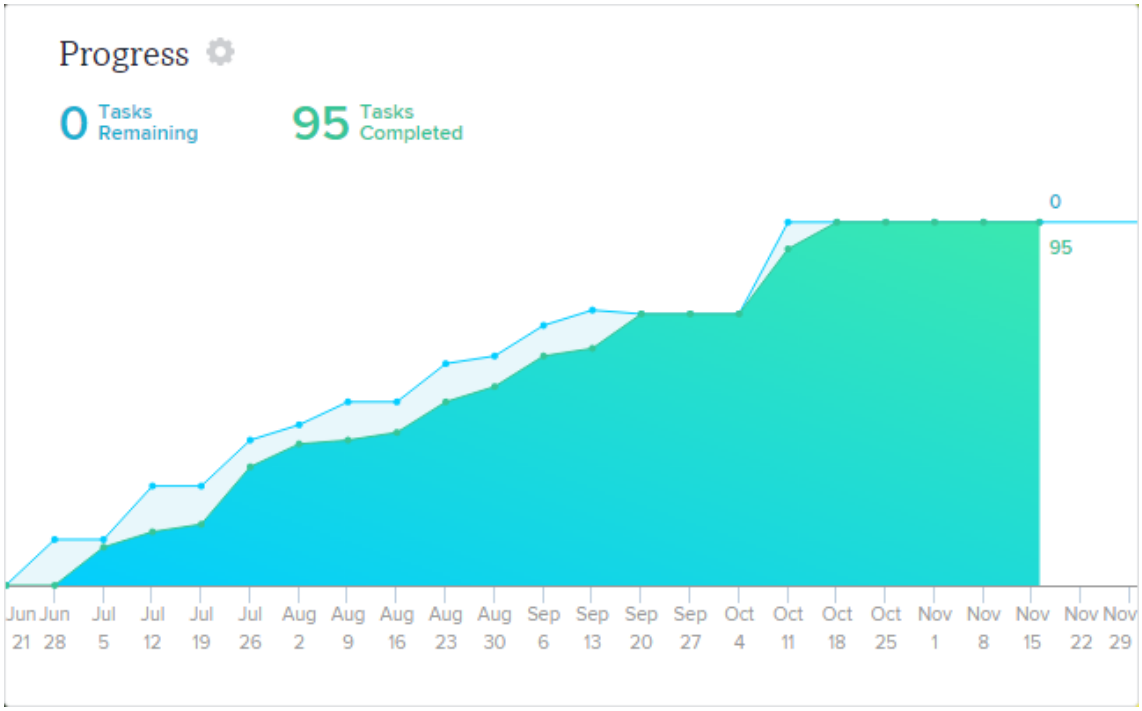

Ilustración 13. GRÁFICO DE SPRINTS DEL PROYECTO (DESARROLLO COMPLETO)

En este gráfico podremos observar un *sprint* o tramo llano de dos semanas, en el cual se observa una completitud de las tareas asignadas. Esto se debió a que se estaba realizando la memoria del proyecto. Posteriormente, se puede apreciar que se retoma el proyecto con un sprint más (2 semanas), en dicho punto, uno de los equipos de desarrollo del proyecto ya había realizado la entrega parcial del mismo. Al cabo de las dos semanas siguientes, se puede apreciar la finalización de todas las tareas y la finalización del proyecto, a excepción de la memoria.

Por otra parte decidimos incluir también características de la metodología eXtreme Programming a nuestra forma de trabajar. Esto es debido a que una de las principales características de esta metodología es el énfasis que pone en la adaptabilidad durante el proceso de desarrollo. Debido a las características del proyecto y las necesidades de esto nos apoyamos en algunas de las características de esta metodología como el desarrollo iterativo e incremental, la refactorización del código o la programación por parejas. En concreto esto último se ha realizado para aquellas partes de la aplicación que poseían una complejidad mayor, de esta forma resultaba más sencillo programar entre dos.

4. Especificación y Análisis

En base a las primeras fases del proyecto, de forma grupal, se fue desarrollando las fases de análisis y especificación.

Las primeras fases abarcaron desde la obtención de información por parte del cliente para realizar análisis de la especificación del proyecto hasta la extracción para definir que tareas debían realizarse. Para ello tuvimos varias reuniones tanto entre nosotros como con nuestro tutor, ya que él era el nexo de unión con el cliente que podía aclararnos todas aquellas dudas que nos surgieron al principio, puesto que consideramos que la especificación inicial que nos fue proporcionada no era la suficientemente clara y descriptiva en alguno puntos.

Tras todas estas reuniones nos hicimos una idea más clara del proyecto que teníamos que llevar a cabo y procedimos a plasmar dichas ideas en una lista formalizada de requisitos.

Conceptos Comunes

Este apartado aclara los diferentes términos empleados lo largo de la especificación y su posterior implantación en el diagrama de Clases.

Degree: son los Grados; forman la clase superior de nuestro sistema.

Mention: son las menciones que van asociadas a un Grado

Course: son los Cursos, un curso se distingue por año y mención

Group: son los grupos que están asociados a los cursos (por año y mención)

Subject: son las asignaturas; las asignaturas están asignadas a los grupos

LinkedSubject: son asignaturas enlazadas y estas comparten el horario. Las asignaturas pueden estar enlazadas independientemente del Grado, solamente debe ser del mismo turno, es decir, mañana o tarde, por motivos de horario.

SharedSubject: son asignaturas que comparten el aula y el horario y se pueden compartir entre grupos dentro de una misma mención.

Schedule: son los esquemas, se crean dos por cada grupo, uno por cada semestre.

Shifts: son las franjas horarias y cada esquema tiene quince franjas horarias, tres franjas horarias por cinco días de la semana.

OptionalSubject: son las asignaturas optativas, y estas están asignadas a los bloques.

Block: son los bloques, que pueden ser comunes o específicos, básicamente se comparten para todos los Grados o para una Mención específica.

Classroom: es el aula a la cual se asignan los grupos, asignaturas optativas o asignaturas compartidas por temas de especificación del modelo

Captura de Requisitos

Los requisitos que se muestran a continuación, han sido definidos al comienzo del proyecto tras varias entrevistas con el cliente, en este caso el tutor del TFG (Trabajo de Fin de Grado).

Actores Principales

Se dispondrá de un tipo de usuario. A continuación se describe al usuario, así como las actividades que realiza respectivamente.

Actor 1

Es el usuario principal, cuenta con todos los privilegios.

Las actividades pueden verse reflejadas en los diagramas de Casos de Uso, descritos posteriormente.

Requisitos Funcionales

RF01: El modelo contiene varios grados.

RF02: Los grados están relacionados con varias menciones.

RF03: Las menciones están relacionadas con varios cursos.

RF04: Los cursos están relacionados con uno o más grupos.

RF05: El modelo contiene varias asignaturas por cada grupo, curso y grado.

RF06: El modelo contiene asignaturas de tipo "ligada".

RF07: Las asignaturas "ligadas" pueden estar ligadas a otras asignaturas del mismo curso en el mismo u otros grados y deben tener el mismo horario.

RF08: El modelo contiene asignaturas de tipo "compartida".

RF09: Las asignaturas "compartidas" pueden estar relacionadas con otras asignaturas idénticas a estas de uno o más grupos y titulaciones, compartiendo el horario y el aula en el que se imparten.

RF10: El modelo contiene varios bloques.

RF11: Los bloques están compuestos de asignaturas del tipo "optativa común" o del tipo "optativa de mención", no mezclando ambos tipos en un mismo bloque.

RF12: Las asignaturas de tipo "optativa común" deben ser comunes a varias menciones.

RF13: Las asignaturas de tipo "optativa de mención" deben ser propias de una mención.

RF14: Las asignaturas que estén en un mismo bloque deben tener el mismo horario.

RF15: Las asignaturas de seis créditos tienen asignadas tres franjas horarias en un mismo horario.

RF16: Las asignaturas de cuatro créditos y medio tienen asignadas dos franjas horarias en un mismo horario.

RF17: Las asignaturas no podrán tener dos o más franjas asignadas un mismo día de la semana en un mismo horario.

RF18: Una de las franjas asociada a una asignatura en un mismo horario se podrá marcar como "grupo grande".

RF19: Es necesaria una página en la que configurar las asignaturas y franjas horarias, eligiendo el horario y asignaturas según el grado, mención, curso, grupo y semestre.

RF20: Asignar una asignatura a una franja horaria en la página correspondiente al RF19.

RF21: Desasignar una asignatura de una franja horaria en la página correspondiente al RF19.

RF22: Intercambiar la posición en el horario de dos asignaturas en la página correspondiente al RF19.

RF23: Marcar como "grupo grande" una franja horaria con una asignatura asignada en la página correspondiente al RF 19.

RF24: Es necesario poder exportar/importar un estado de la base de datos para trabajar con varias configuraciones.

RF25: Es necesaria una página en la que consultar la información configurable en el requisito RF19.

RF26: Es necesario poder exportar en formato pdf la información representada en el requisito RF25 por cada una de las menciones de la base de datos.

RF27: Es necesaria una página en la que consultar las asignaturas asignadas a un aula en cada una de las franjas horarias del horario de un semestre de un grupo, curso y mención dados.

RF28: Es necesaria una página en la que poder configurar las aulas y los grupos.

RF29: Asignar un grupo a un aula.

RF30: Desasignar un grupo de un aula.

RF31: Consultar las relaciones entre aulas y grupos.

RF32: Es necesaria una página en la que configurar las asignaturas optativas y las aulas.

RF33: Asignar una asignatura optativa a un aula.

RF34: Desasignar una asignatura optativa a un aula.

RF35: Es necesaria una página para consultar la información referente a los bloques (asignaturas contenidas, aula y horario).

RF36: Es necesario poder exportar en formato pdf la información representada en el requisito RF35.

RF37: Es necesaria una página que permita la creación de los modelos recogidos en los requisitos RF1, RF2, RF3, RF4, RF5, RF6, RF8, RF10 y RF11.

Requisitos No Funcionales

RNF01: El sistema podrá desplegarse en computadoras con un sistema operativo de distribución Linux.

RNF02. La interfaz deberá ser intuitiva y amigable.

RNF03. La aplicación será de estilo drag&drop.

Casos de Uso

Diagramas Generales

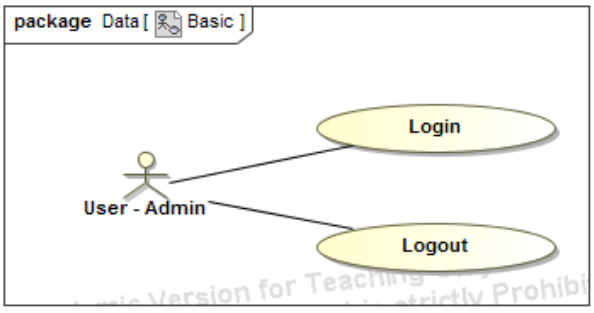

Ilustración 14. CASO DE USO BÁSICO

Este diagrama cumple los casos de uso CU01 y CU52, expuestos posteriormente.

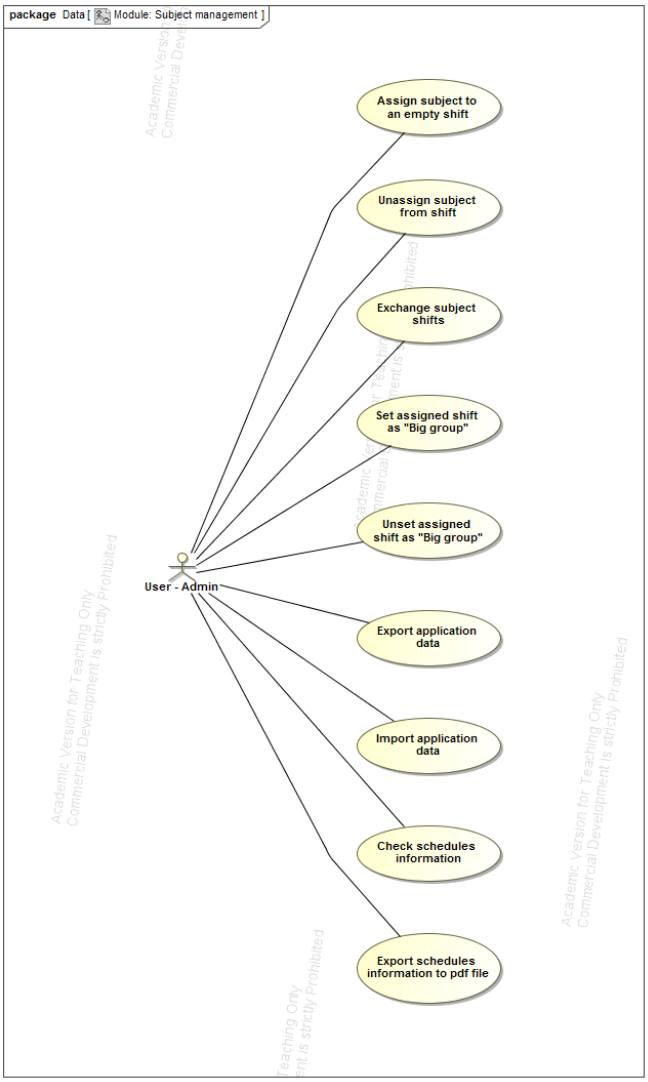

Ilustración 15. CASO DE USO: MÓDULO DE GESTIÓN DE ASIGNATURAS

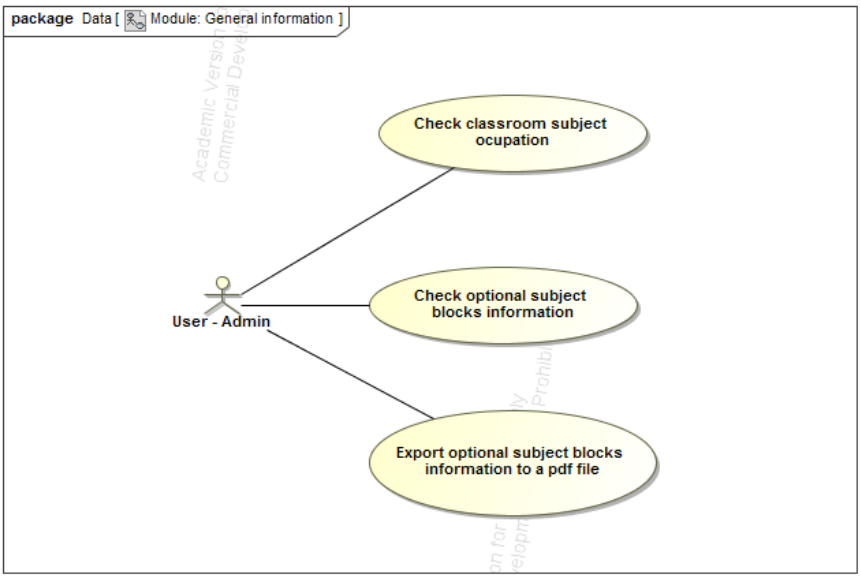

Ilustración 16. CASO DE USO: MÓDULO DE VISIÓN GENERAL

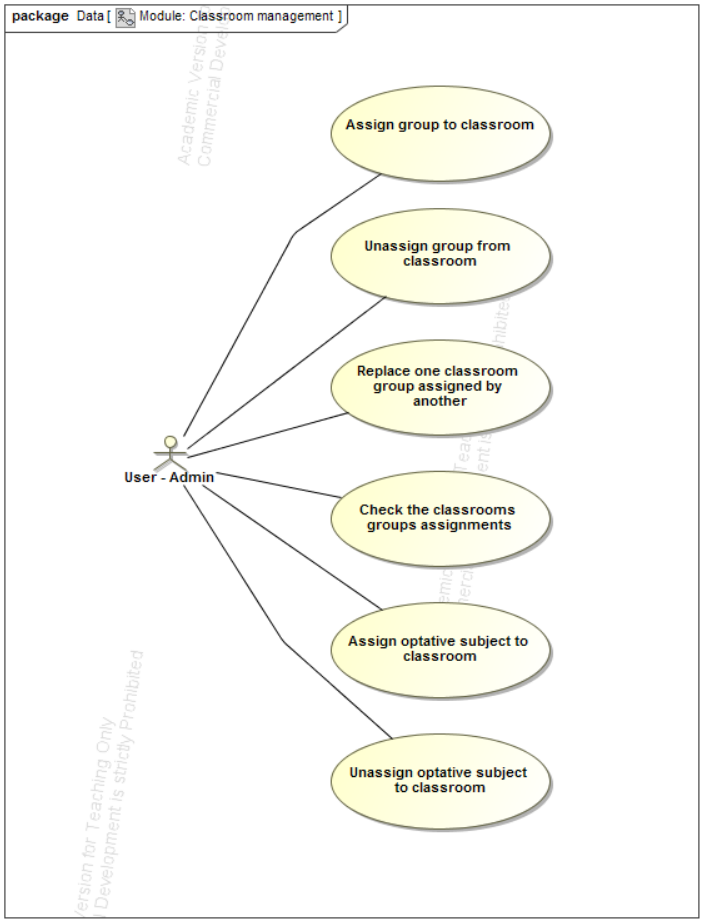

Ilustración 17. CASO DE USO: MÓDULO DE GESTIÓN DE AULAS

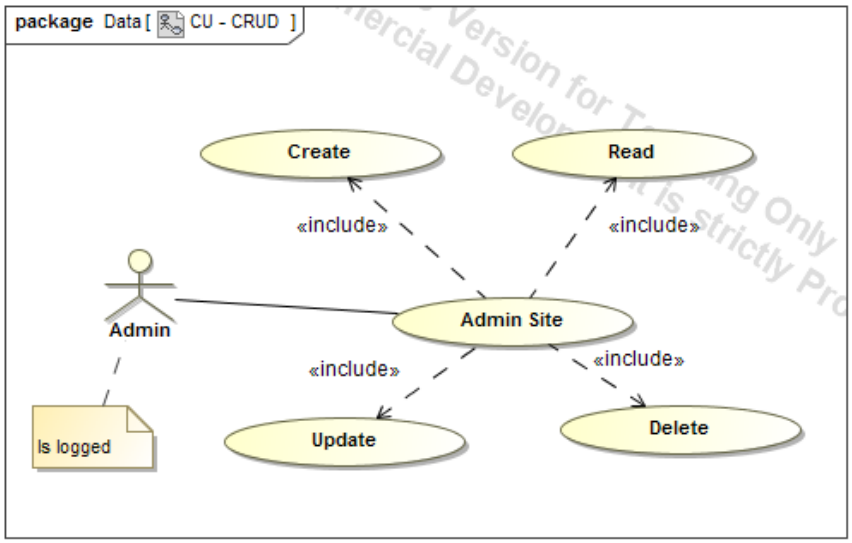

Ilustración 18. CASO DE USO: MÓDULO DE ADMINISTRACIÓN

Este diagrama cumple los casos de uso: desde CU02 hasta CU31, expuestos posteriormente.

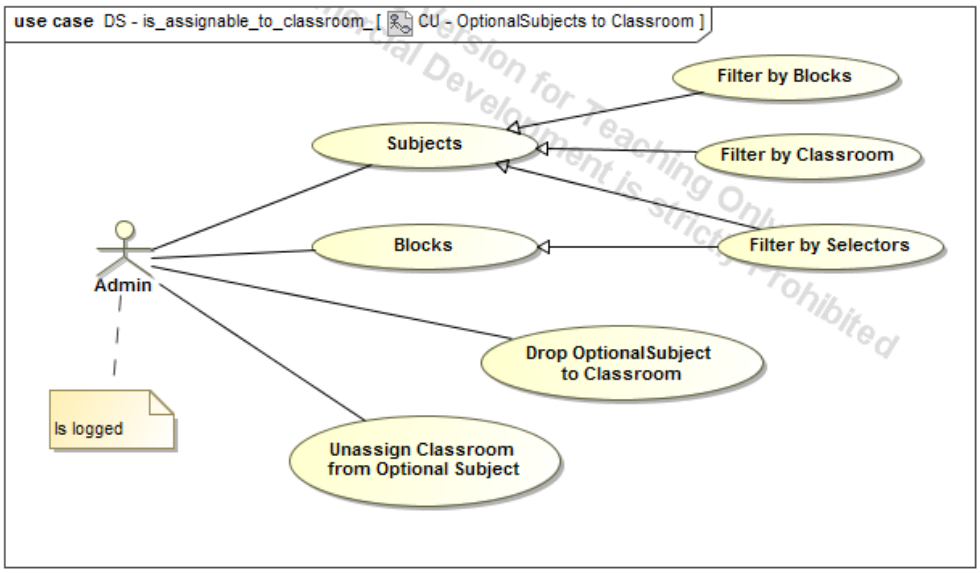

Ilustración 19. CASO DE USO: ASIGNACIÓN DE ASIGNATURAS OPTATIVAS A AULAS

Este diagrama cumple los casos de uso CU48 y CU49, documentados posteriormente.

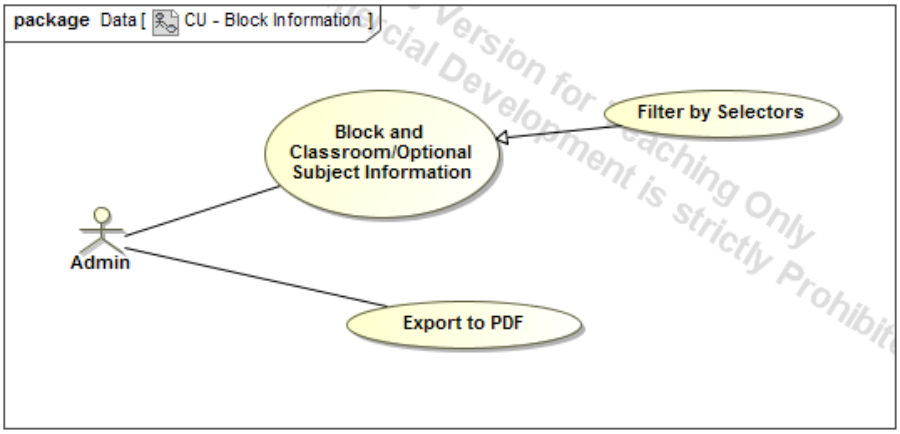

Ilustración 20. CASO DE USO: INFORMACIÓN DE BLOQUES DE OPTATIVIDAD

Este diagrama cumple los casos de uso CU50 y CU51, documentados posteriormente.

Especificación

CU01. Login

Resumen: El usuario se autentica en la aplicación.

Actor: Usuario.

Precondición: El usuario debe tener una cuenta registrada en la aplicación.

Escenario:

- El usuario accede a la página de login.
- Introduce su nombre de usuario.
- Introduce su contraseña.
- Hace clic en el botón “Login".
- El sistema identifica al usuario.
- El usuario es redireccionado a la página principal.

CU02. Registrar un nuevo Grado

Resumen: El usuario registra un nuevo grado en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir un grado
- Introduce la información necesaria para crear un grado
- Pulsa el botón crear un grado
- El sistema crea el grado correctamente y aparece en la aplicación

CU03. Editar un Grado

Resumen: El usuario edita la información de un grado de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de un grado
- Introduce la información a cambiar en el grado
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU04. Borrar un Grado

Resumen: El usuario borra un grado registrado en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página un grado
- Pulsa el botón borrar
- El sistema borra el grado de la base de datos.

CU05. Registrar una nueva Mención

Resumen: El usuario registra una nueva Mención en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una mención
- Introduce la información necesaria para crear una mención
- Pulsa el botón crear una mención
- El sistema crea la mención correctamente y aparece en la aplicación

CU06. Editar una Mención

Resumen: El usuario edita la información de una mención de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una mención
- Introduce la información a cambiar en la mención
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU07. Borrar una Mención

Resumen: El usuario borra una mención registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una mención
- Pulsa el botón borrar
- El sistema borra la mención de la base de datos.

CU08. Registrar un nuevo Curso

Resumen: El usuario registra un nuevo curso en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir un curso
- Introduce la información necesaria para crear un curso
- Pulsa el botón crear un curso
- El sistema crea el curso correctamente y aparece en la aplicación

CU09. Editar un Curso

Resumen: El usuario edita la información de un curso de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de un curso
- Introduce la información a cambiar en el curso
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU10. Borrar un Curso

Resumen: El usuario borra un curso registrado en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página un curso
- Pulsa el botón borrar
- El sistema borra el curso de la base de datos.

CU11. Registrar un nuevo Grupo

Resumen: El usuario registra un nuevo grupo en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir un grupo
- Introduce la información necesaria para crear un grupo
- Pulsa el botón crear un grupo
- El sistema crea el grupo correctamente y aparece en la aplicación

CU12. Editar un Grupo

Resumen: El usuario edita la información de un grupo de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de un grupo
- Introduce la información a cambiar en el grupo
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU13. Borrar un Grupo

Resumen: El usuario borra un curso registrado en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página un grupo
- Pulsa el botón borrar
- El sistema borra el grupo de la base de datos.

CU14. Registrar una nueva Asignatura

Resumen: El usuario registra una nueva asignatura en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una asignatura
- Introduce la información necesaria para crear una asignatura
- Pulsa el botón crear una asignatura
- El sistema crea la asignatura correctamente y aparece en la aplicación

CU15. Editar una Asignatura

Resumen: El usuario edita la información de una asignatura de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una asignatura
- Introduce la información a cambiar en la asignatura
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU16. Borrar una Asignatura

Resumen: El usuario borra una asignatura registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una asignatura
- Pulsa el botón borrar
- El sistema borra la asignatura de la base de datos.

CU17. Registrar una nueva Asignatura Ligada

Resumen: El usuario registra una nueva asignatura ligada en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una asignatura ligada
- Introduce la información necesaria para crear una asignatura ligada
- Pulsa el botón crear una asignatura ligada
- El sistema crea la asignatura ligada correctamente y aparece en la aplicación

CU18. Editar una Asignatura Ligada

Resumen: El usuario edita la información de una asignatura ligada de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una asignatura ligada
- Introduce la información a cambiar en la asignatura ligada
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU19. Borrar una Asignatura Ligada

Resumen: El usuario borra una asignatura ligada registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una asignatura ligada
- Pulsa el botón borrar
- El sistema borra la asignatura ligada de la base de datos.

CU20. Registrar una nueva Asignatura Compartida

Resumen: El usuario registra una nueva asignatura compartida en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una asignatura compartida
- Introduce la información necesaria para crear una asignatura compartida
- Pulsa el botón crear una asignatura compartida
- El sistema crea la asignatura compartida correctamente y aparece en la aplicación

CU21. Editar una Asignatura Compartida

Resumen: El usuario edita la información de una asignatura compartida de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una asignatura compartida
- Introduce la información a cambiar en la asignatura compartida
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU22. Borrar una Asignatura Compartida

Resumen: El usuario borra una asignatura compartida registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una asignatura compartida
- Pulsa el botón borrar
- El sistema borra la asignatura compartida de la base de datos.

CU23. Registrar una nueva Asignatura Optativa

Resumen: El usuario registra una nueva asignatura optativa en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una asignatura optativa
- Introduce la información necesaria para crear una asignatura optativa
- Pulsa el botón crear una asignatura optativa
- El sistema crea la asignatura optativa correctamente y aparece en la aplicación

CU24. Editar una Asignatura Optativa

Resumen: El usuario edita la información de una asignatura optativa de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una asignatura optativa
- Introduce la información a cambiar en la asignatura optativa
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU25. Borrar una Asignatura Optativa

Resumen: El usuario borra una asignatura optativa registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una asignatura optativa
- Pulsa el botón borrar
- El sistema borra la asignatura optativa de la base de datos.

CU26. Registrar un nuevo Bloque de Optativas

Resumen: El usuario registra un nuevo bloque de optativas en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir un bloque de optativas
- Introduce la información necesaria para crear un bloque de optativas
- Pulsa el botón crear un bloque de optativas
- El sistema crea el bloque de optativas correctamente y aparece en la aplicación

CU27. Editar un Bloque de Optativas

Resumen: El usuario edita la información de un bloque de optativas de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de un bloque de optativas
- Introduce la información a cambiar en el bloque de optativas
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU28. Borrar un Bloque de Optativas

Resumen: El usuario borra un bloque de optativas registrado en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página un bloque de optativas
- Pulsa el botón borrar
- El sistema borra el bloque de optativas de la base de datos.

CU29. Registrar una nueva Clase

Resumen: El usuario registra una nueva clase en la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de añadir una clase
- Introduce la información necesaria para crear una clase
- Pulsa el botón crear una clase
- El sistema crea la clase correctamente y aparece en la aplicación

CU30. Editar una Clase

Resumen: El usuario edita la información de una clase de la aplicación.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página de editar de una clase
- Introduce la información a cambiar en la clase
- Pulsa el botón de guardar la información
- El sistema guarda la información correctamente.

CU31. Borrar una Clase

Resumen: El usuario borra una clase registrada en el sistema.

Actor: Administrador.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de administrador
- Accede a la página una clase
- Pulsa el botón borrar
- El sistema borra la clase de la base de datos.

CU32. Asignar una asignatura a una franja horaria vacía

Resumen: El usuario asigna una asignatura a una franja horaria vacía de un horario perteneciente a un grupo.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- El usuario arrastra una asignatura a una franja del horario vacía
- El sistema asigna la asignatura a esa franja horaria
- La asignatura aparece en la franja en la cual se ha asignado.

CU33. Desasignar una asignatura a una franja horaria

Resumen: El usuario desasigna una asignatura de una franja horaria de un horario perteneciente a un grupo.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- El usuario hace clic en la cruz para desasignar una asignatura
- El sistema desasigna la asignatura de esa franja horaria
- La asignatura ya no aparece en la franja de la cual se ha desasignado.

CU34. Asignar una asignatura a una franja horaria con otra asignatura

Resumen: El usuario reemplaza una asignatura de una franja horaria con otra asignatura.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- El usuario arrastra una asignatura a una franja del horario con otra asignatura
- El sistema asigna la asignatura a esa franja horaria
- El sistema desasigna la asignatura reemplazada de esa franja horaria
- La asignatura aparece en la franja en la cual se ha asignado
- La asignatura reemplazada ya no aparece en la franja en la cual se encontraba.

CU35. Intercambiar las franjas horarias entre dos asignaturas

Resumen: El usuario intercambia la posición de dos asignaturas ya asignadas en el horario.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- El usuario arrastra una asignatura que ya está en el horario a una franja del horario con otra asignatura
- El sistema asigna la asignatura a esa franja horaria
- El sistema asigna la asignatura reemplazada en la franja horaria correspondiente a la asignatura que la ha reemplazado
- La asignatura aparece en la franja en la cual se ha asignado
- La asignatura reemplazada aparece en la franja en la cual se encontraba la asignatura que la ha reemplazado.

CU36. Establecer la franja horaria de una asignatura como “franja de grupo grande”

Resumen: El usuario establece la franja horaria de una asignatura como “franja de grupo grande”.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- Marca el check de una asignatura que se encuentre asignada en el horario
- El sistema marca y muestra la franja en la que se encuentra la asignatura como “franja de grupo grande”

CU37. Desestablecer la franja horaria de una asignatura como “franja de grupo grande”

Resumen: El usuario desmarca la franja horaria de una asignatura como “franja de grupo grande”.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- Desmarca el check de una asignatura que se encuentre asignada en el horario y cuya franja esté marcada como “franja de grupo grande”
- El sistema desmarca la franja en la que se encuentra la asignatura como “franja de grupo grande”.

CU38. Guardar una configuración

Resumen: El usuario guarda todos los datos creados en la base de datos en un archivo para guardar una configuración del sistema.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de gestión de horarios
- Pulsa el botón de exportar la configuración
- Introduce el nombre de la configuración
- Hace clic en guardar la configuración
- El sistema guarda los datos de la configuración de la aplicación.

CU40. Cargar una configuración

Resumen: El usuario carga datos creados previamente desde un archivo para restablecer una configuración del sistema.

Actor: Usuario.

Precondición: El usuario debe estar autenticado. Debe existir al menos una configuración creada.

Escenario:

- El usuario accede a la página de gestión de horarios
- Pulsa el botón de importar la configuración
- Escoge que configuración quiere cargar en el sistema
- Hace clic en cargar la configuración
- El sistema carga todos los datos de la configuración de la aplicación.

CU41. Consultar los horarios

Resumen: El usuario consulta los horarios que existen en el sistema divididos por grupos.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de información de horarios
- El sistema muestra los horarios pertenecientes a un grupo.

CU42. Exportar horarios a PDF

Resumen: El usuario exporta a un fichero "pdf" la información de los horarios del sistema.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de información de horarios
- Hace clic en el botón de exportar a pdf
- El sistema genera un archivo pdf con toda la información de los horarios del sistema.

CU43. Consultar ocupación de aulas por asignaturas.

Resumen: El usuario consulta la ocupación por asignaturas de cada aula por cada franja horaria.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de visión general
- El sistema muestra la ocupación por asignaturas de las clases dividido por horas de un determinado día.

CU44. Asignar un grupo a un aula sin grupo.

Resumen: El usuario asigna un grupo a un aula que no tenga ningún grupo asignado.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de grupos a aulas
- Arrastra un grupo a un aula sin grupo asignado
- El sistema realiza la asignación
- Muestra el grupo asignado al aula.

CU45. Desasignar un grupo de un aula.

Resumen: El usuario desasigna un grupo de un aula.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de grupos a aulas
- Hace clic en desasignar la asignatura que esté asignada en un grupo
- El sistema realiza la desasignación
- Ya no muestra el grupo asignado al aula.

CU46. Asignar un grupo a un aula con un grupo ya asignado.

Resumen: El usuario asigna un grupo a un aula que tiene un grupo asignado.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de grupos a aulas
- Arrastra un grupo a un aula con un grupo ya asignado

- El sistema realiza la asignación del grupo arrastrado en la franja
- Desasigna el grupo que estaba asignado en la franja
- Muestra el nuevo grupo asignado al aula.

CU47. Consultar la ocupación por grupos de las aulas

Resumen: El usuario consulta las asignaciones entre grupos y aulas.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de grupos a aulas
- El sistema muestra una tabla con la ocupación de las aulas por los grupos asignados.

CU48. Asignar una asignatura optativa a un aula.

Resumen: El usuario asigna una asignatura optativa a un aula.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de asignaturas optativas a
- Escoge un bloque de optativas
- Arrastra una asignatura optativa a un aula.
- El sistema realiza la asignación
- Muestra la asignatura optativa asignada al aula.

CU49. Desasignar una asignatura optativa de un aula.

Resumen: El usuario desasigna un grupo de un aula.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página asignación de grupos a aulas
- Hace clic en desasignar la asignatura optativa que esté asignada en un grupo
- El sistema realiza la desasignación
- Ya no muestra la asignatura optativa asignada al aula.

CU50. Consultar bloques de optativas

Resumen: El usuario consulta la información de los bloques de asignaturas optativas.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página información de bloques de optativas
- El sistema muestra los bloques de optativas con su información y sus asignaturas.

CU51. Exportar bloques de optativas a PDF

Resumen: El usuario exporta a un fichero "pdf" la información de los bloques de optativas del sistema.

Actor: Usuario.

Precondición: El usuario debe estar autenticado.

Escenario:

- El usuario accede a la página de información de bloques
- Hace clic en el botón de exportar a pdf
- El sistema genera un archivo pdf con toda la información de los bloques de optativas del sistema.

CU52. Cerrar la sesión

Resumen: El usuario cierra la sesión en la aplicación.

Actor: Usuario.

Precondición: El usuario debe tener una cuenta registrada en la aplicación y estar autenticado en ella.

Escenario:

- El usuario hace clic que en el botón de cerrar sesión
- El sistema desautentica al usuario de la aplicación
- Redirige a la página de login.

5. Diseño del Sistema

Una vez definidos los requisitos, con sus correspondientes casos de uso y sus diagramas, el siguiente paso lógico consistía en definir el modelo de datos que utilizaríamos posteriormente en el desarrollo, para ello utilizamos un diagrama de clases. Esta no fue una tarea fácil, ya que como ya se ha comentado con anterioridad la definición de algunos conceptos en la especificación eran un tanto ambiguos y/o incompletos. Por ello durante las etapas iniciales del desarrollo este modelo cambió en más de una ocasión. En la siguiente imagen se puede ver cuál fue el primer modelo que se creó.

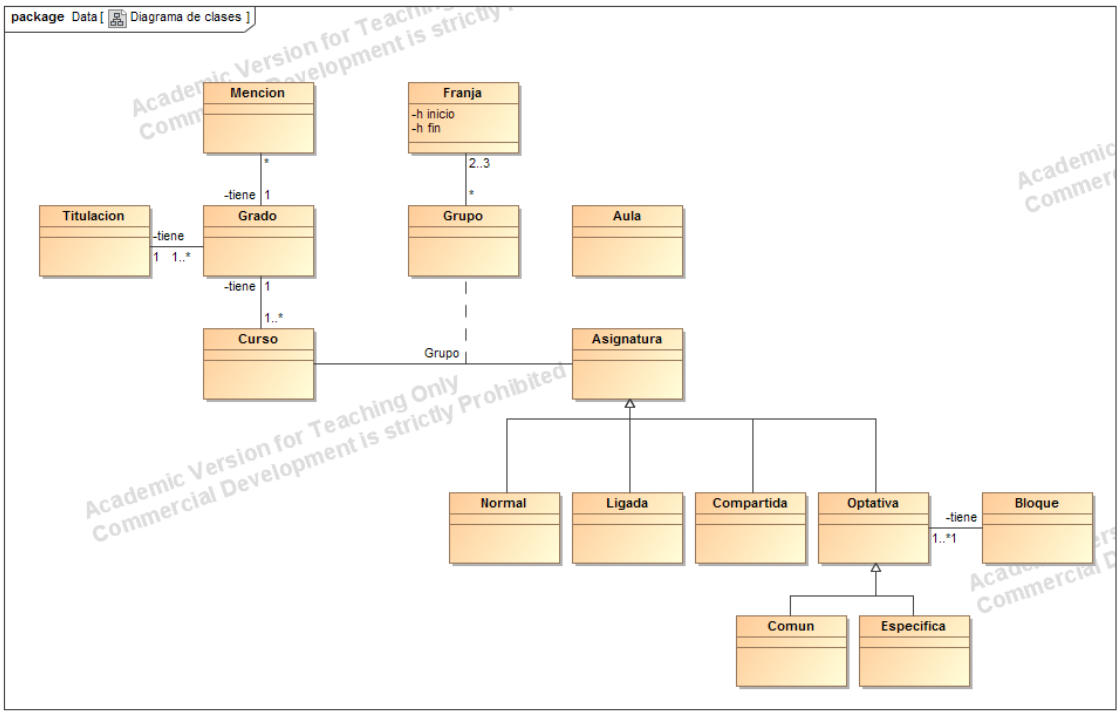

ILUSTRACIÓN 14. PRIMERA VERSIÓN DEL DIAGRAMA DE CLASES

Esta primera versión sufrió infinidad de cambios en las etapas iniciales del proyecto. Incluso en etapas más avanzadas sufrió pequeñas modificaciones para adaptarse a ciertas situaciones que no habíamos definido consistentemente al inicio o incluso para añadir algunas nuevas pequeñas funcionalidades que surgieron a la largo de todo el proceso.

Finalmente tras muchos cambios y reuniones para exponer las diferentes opiniones de los integrantes respecto a la definición del modelo se llegó a la versión definitiva que actualmente forma parte del proyecto.

Conceptualmente podrían existir alternativas más óptimas del modelo de datos, pero tras considerar otras cuestiones como la funcionalidad final de la aplicación optamos por esta versión.

Por ejemplo, una de las cosas que más complejidad y coste supuso, en este nivel, fue la definición de un bloque de optativas. En nuestro diagrama, este bloque de optativas está definido como una herencia desde la clase Asignatura, lo cual a nivel conceptual no tiene demasiado sentido. Pero a nivel de la aplicación no hay que olvidar el comportamiento que estos bloques tienen, ya que a la hora de crear los horarios, aunque un bloque esté formado por un conjunto de asignaturas optativas, este se comporta como una única asignatura a la hora de ser asignado a una franja horaria. Por ello, tanto las asignaturas compartidas como ligadas y los bloques de optativas poseen un comportamiento similar en nuestro modelo y todos ellos son objetos que heredan de la clase Asignatura.

En la siguiente imagen podemos observar la representación de la versión definitiva del modelo de datos mediante un Diagrama de Clases.

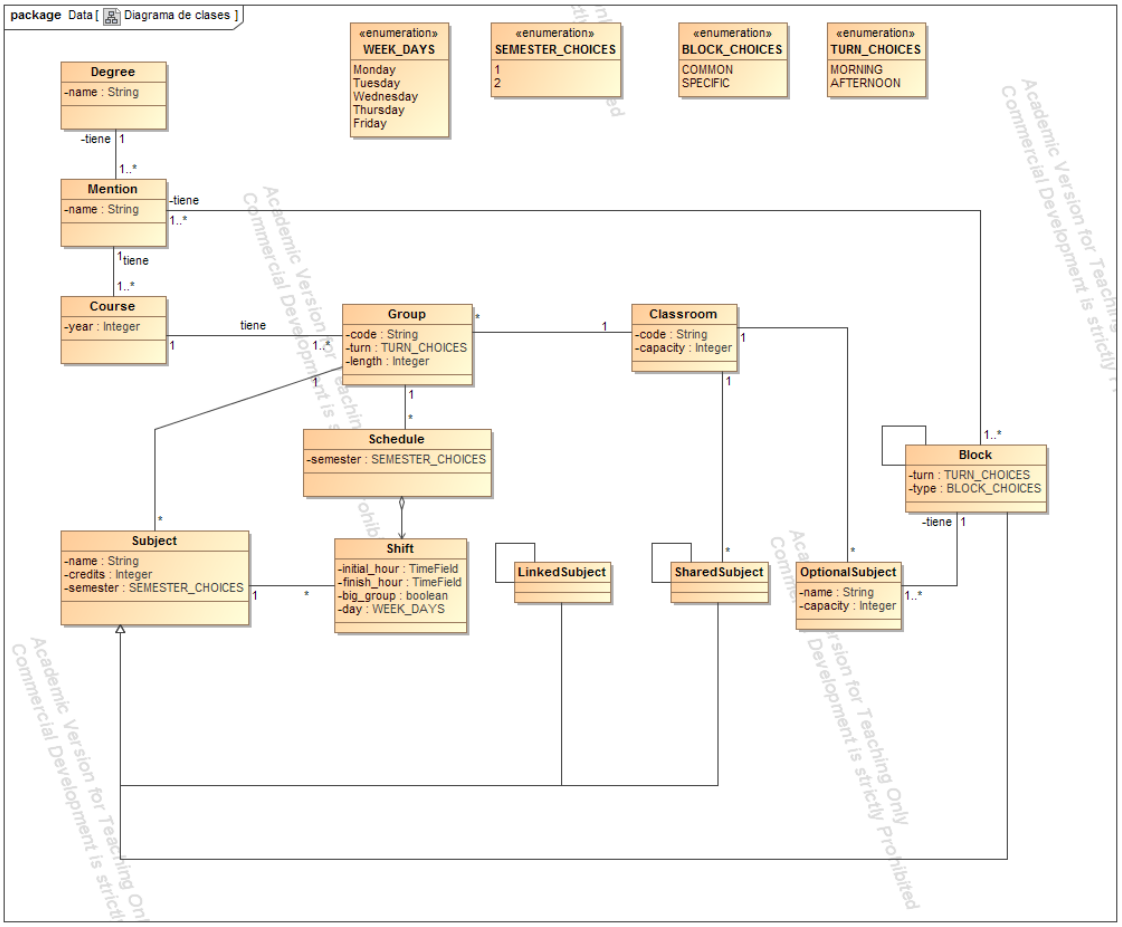

Ilustración 21. ÚLTIMA VERSIÓN DEL DIAGRAMA DE CLASES

Tras esto ya se iba acercando el momento de empezar a desarrollar. Pero antes de eso era necesario hacer una primera división del trabajo. Para ello haciendo una estimación lo más aproximada posible de las tareas que podían surgir del proyecto, dividimos el trabajo en tres posibles partes y las repartimos, de forma que el equipo de desarrollo que se encontraba aquí tuviera dos partes y el equipo que se encontraba fuera tuviera

una tercera parte, intentando que no estuviera demasiado cohesionada con el resto para evitar posibles conflictos. Ya que debido a las dificultades existentes entre la comunicación entre ambos equipos debido a la distancia, era preferible no encontrarnos con este tipo de conflictos.

También para facilitar el trabajo de ambos equipos se aprovechó de las herramientas que teníamos a disposición y utilizando el sistema de control de versiones se crearon dos ramas para que cada equipo trabajara en una y evitar conflictos. Ya una vez que el equipo estuvo reunido en las etapas finales se procedió a hacer un "Merge" o mezcla de ambas ramas resultando en el proyecto definitivo.

Una vez definidas todas estas cosas solo faltaban unos pasos más antes de que cada uno pudiera dedicarse a su parte.

Creamos el proyecto básico de Django sobre el que íbamos a trabajar y le añadimos las aplicaciones necesarias que eran necesarias para utilizar las tecnologías que habíamos decidido usar.

A la hora de definir el proyecto, diseñamos un diagrama de componentes en el cual se puede observar la estructura general del proyecto.

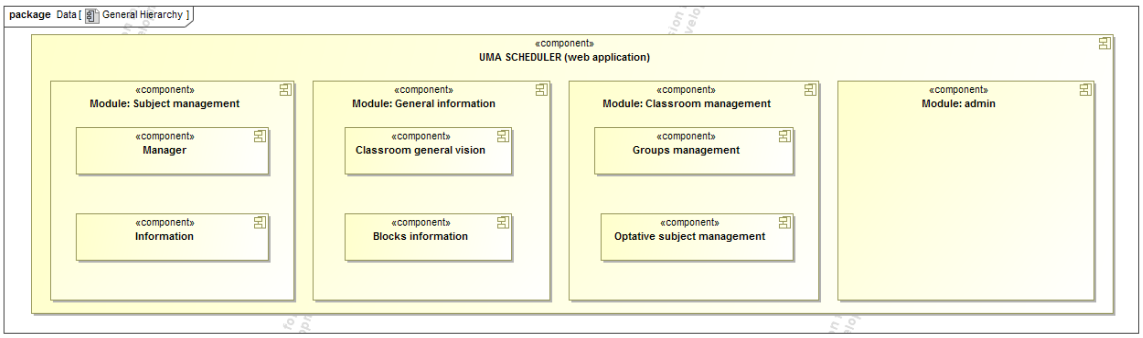

Ilustración 22. DIAGRAMA DE COMPONENTES

De esta forma la estructura de paquetes del proyecto quedó como se muestra en la siguiente ilustración.

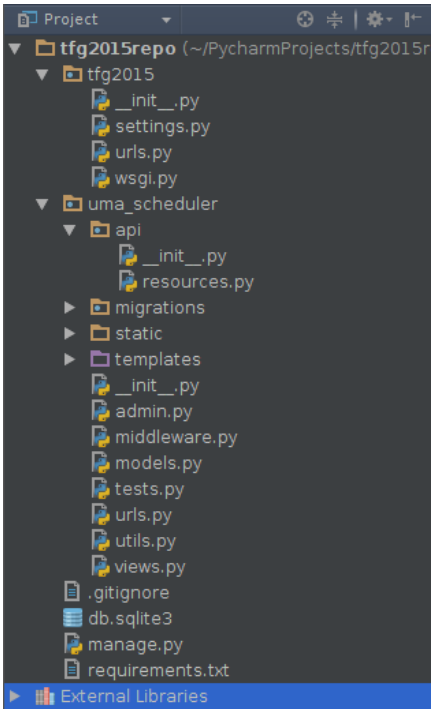

Ilustración 23. ESTRUCTURA DE PAQUETES DE LA APLICACIÓN

Una vez el proyecto estaba creado se inició el repositorio de Git en BitBucket, se subió el proyecto a este y se crearon las ramas sobre las que se iba a trabajar.

Tras toda la selección de las tecnologías y diseño del proyecto quedó con la siguiente arquitectura que se puede observar en la imagen.

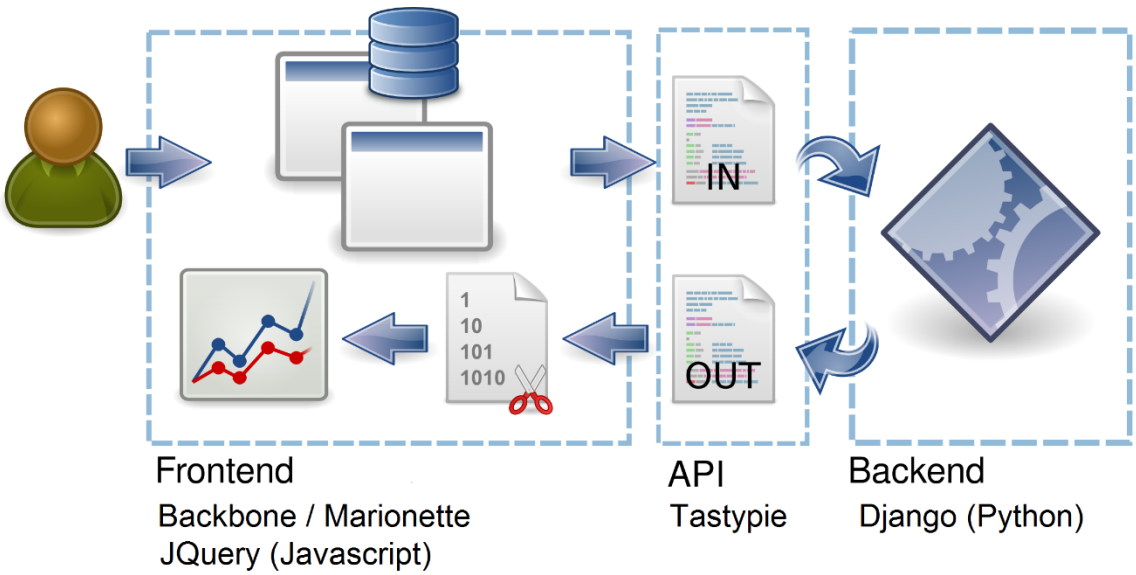

Ilustración 24. DIAGRAMA DE ARQUITECTURA DE LA APLICACIÓN

El siguiente paso en común fue la definición de los recursos de Tastypie para poder acceder a los datos a través de una API RESTful durante todo el desarrollo. Como esto

era algo que necesitaríamos los tres integrantes del grupo, se realizó al principio del desarrollo para que todos lo tuviéramos a nuestra disposición. Aunque luego a lo largo del desarrollo cada uno introdujo pequeñas modificaciones para que se adaptara a sus necesidades.

Modelado del Sistema

Diagrama de Clases

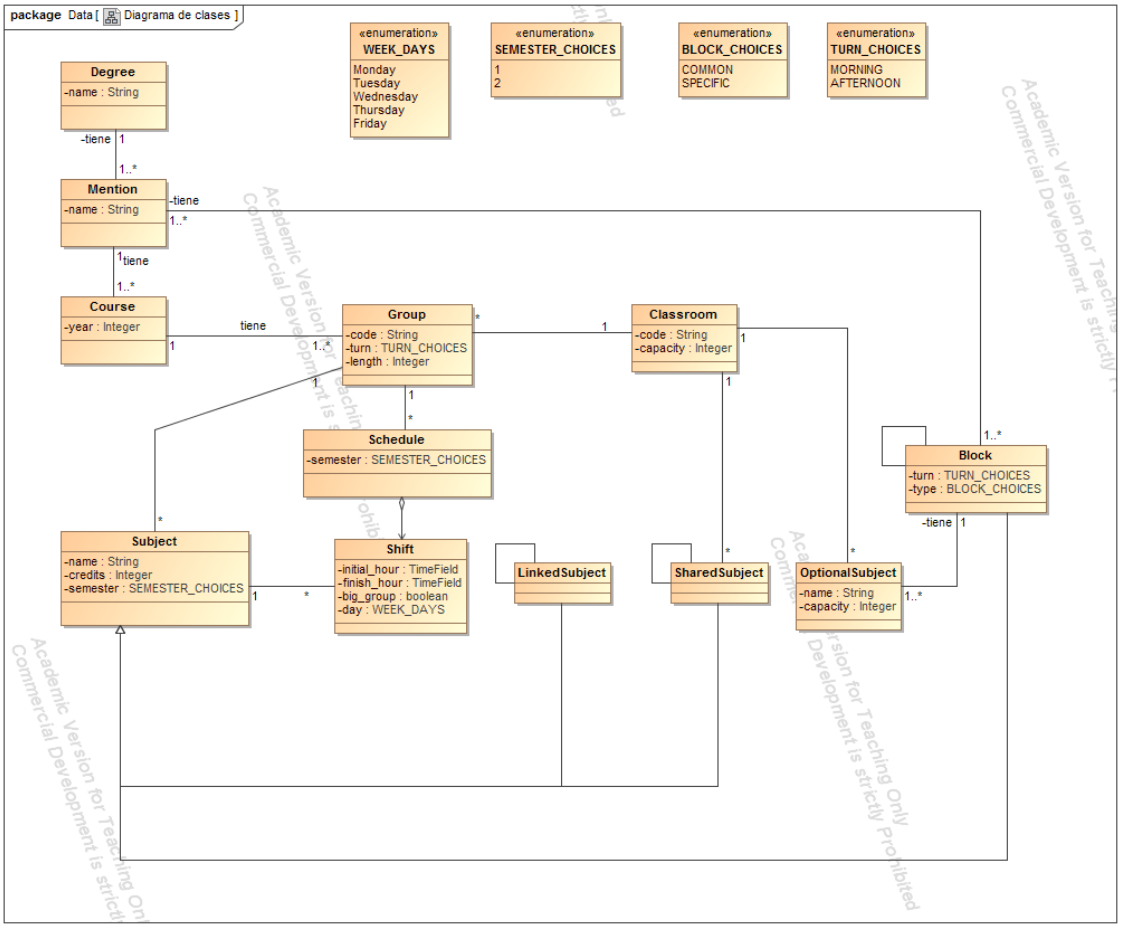

Ilustración 25. VERSIÓN DEFINITIVA DEL DIAGRAMA DE CLASES

El diagrama de clases es una representación gráfica de nuestro modelo de datos, permitiendo visualmente distinguir las diferentes clases de las que dispone nuestro sistema y las listas estáticas o tipos enumerados que se usan como valores en algunos de los atributos de nuestro sistema.

Ahora pasaremos a explicar individualmente las diferentes clases que conforman nuestro modelo de datos:

Clase *Degree*

La clase *Degree* modela o representa los diferentes Grados de los que consta nuestra Escuela, tales como Ingeniería del Software o Ingeniería Informática, por ejemplo. Conforma la clase inicial o base sobre la que se construye nuestro sistema.

Relaciones

Degree/Mention de 1 a 1..*

(aclarar que el asterisco " * " significa muchos)

Métodos

Dispone de los métodos propios de guardar y borrar pre-definidos por Django.

(Almacena una instancia con sus datos o borra la susodicha instancia)

Clase *Mention*

Relaciones

Mention/Degree de 1..* a 1

Mention/Block de 1..* a 1..*

Mention/Course de 1 a 1..*

Métodos

Dispone de los métodos propios de guardar y borrar pre-definidos por Django.

(Almacena una instancia con sus datos o borra la susodicha instancia de la clase)

Clase Course

Relaciones

Course/Mention de 1..* a 1

Course/Group de 1 a 1..*

Restricciones

Tiene una restricción única conjunta (*unique_together*) del atributo "year" y la clave foránea "mention".

Métodos

Dispone de los métodos propios de guardar y borrar pre-definidos por Django.

(Almacena una instancia con sus datos o borra la susodicha instancia de la clase)

Clase *Group*

Relaciones

Group/Course de 1..* a 1

Group/Classroom de * a 1

Group/Schedule de 1 a *

Restricciones

Tiene una restricción en la relación Group/Classroom que no permite que el atributo "length" de Group sea superior al atributo "capacity" de Classroom.

Tiene una restricción única conjunta (*unique_together*) del atributo "code" y la clave foránea "course".

Métodos

+ save_admin(): void

Al crear una nueva Instancia:

- crea una instancia de la clase "Group" con los datos introducidos por el usuario.
- crea dos instancias de "Scheduler" asociadas a la instancia "Group" creada previamente.

- Se filtran todas las instancias "Subject" por los campos "course" y "turn"; se excluyen todas las subclases y las instancias con el atributo "name" repetido; las resultantes son duplicadas y asociadas al grupo creado previamente, exceptuando la clave foránea "shifts" que se deja vacía (no se duplican las relaciones con "Shift").

- Las "LinkedShared" se duplican del mismo método que las instancias "Subject", explicadas previamente.

- Se duplican las "SharedSubject" relacionadas con los mismos campos "turn" y "year", excluyendo las que tengan el mismo campo "name", y se guardan asociadas con la nueva instancia de "Group". Y se actualiza el atributo "shifts" para enlazarlo con los relacionados a su "Schedule" del mismo turno o atributo "turn".

- Se filtran todas las instancias "Block" que tengan el atributo "type" igual a "COMMON" y todas las que "type" sea igual a "SPECIFIC" y relacionada con la misma instancia "Mention" que la instancia "Group" creada al comienzo del método; y se excluyen las que tengan el atributo "name" repetido. Las resultantes, son duplicadas y se asocian a la instancia "Group" creada. Se actualiza, en cada instancia, el atributo "shifts" para enlazarlo con los relacionados a su "Schedule" del mismo atributo "turn" que la instancia "Group" creada. Además, las instancias "OptionalSubject" relacionadas a cada "Block" son duplicadas y asociadas a esta nueva instancia de "Block".

Al actualizar la Instancia

Se obtienen todos las instancias de grupo con el mismo código y curso, que teóricamente sería solamente uno y se llama al método guardar, .save(), que trae por defecto, para actualizar los datos.

El método borrar, *delete*():void, está pre-definido por Django y borra la susodicha instancia de la clase.

Clase *Subject*

Relaciones

Subject/Group de * a 1

Subject/Shift de 1 a *

Restricciones

Tiene una restricción única conjunta (*unique_together*) del atributo “name” y la clave foránea “group”.

Métodos

+delete_admin():void

Con respecto a la instancias “Subject” que van a ser eliminadas, se obtienen todas las instancias “Subject” con el mismo atributo “name” y la misma relación con la Instancia “Course”, obtenida a través de la clave foránea “group”; previamente se eliminan todas las relaciones con la instancia “Shift” y con las instancias “SharedSubjet” y “LinkedSubject”; y posteriormente se eliminan todas las instancias “Subjects” que fueron filtradas.

+save():void

Al crear la instancia:

Filtramos las instancias “Group” por el campo “course” y por cada una de ellas creamos una nueva instancia “Subject” asociada a esa instancia “Group”.

Al actualizar la instancia:

Obtenemos, mediante una llama a la base de datos, todas las instancias “Subject” filtradas por la clave foránea “course” y el atributo “name”, previo a la modificación. Y actualizamos los campos “name” y “credits” de todas ellas.

+assign(current_shift:Shift, new_shift:Shift, shifts_assigned:*Shifts): void

Este método, que ha sido desarrollado por otro compañero del proyecto, se ha reutilizado parcialmente para asignar a una instancia “Subject” o subclase de la misma una franja horaria o “Shift”.

+unassign(current_shift:Shift): void

Este método, que ha sido desarrollado por otro compañero del proyecto, se ha reutilizado parcialmente para desasignar de una instancia “Subject” o subclase de franja horaria o “Shift”.

Clase *Block*

Relaciones

Block/Mention de 1..* a 1..*

Block/Block de * a *

Block/OptionalSubject de 1 a 1..*

Restricciones

Tiene una restricción única conjunta (*unique_together*) del atributo “name” y la clave foránea “group”.

Tiene una restricción a nivel de relación Block/Group que comprueba desde el “controlador del administrador” (admin.py) que existe alguna instancia “Group”, en la instancia “Course” o la instancia “Mention” y atributo “year” indicados por el usuario.

Métodos

+ save_admin(): void

Al crear una nueva Instancia:

Filtramos las instancias “Group” dependiendo del atributo “type”: si el atributo “type” es igual a “COMMON” filtramos por el atributo “year”, obtenido a través de la clave foránea ”group” de la instancia “course”; en caso contrario, filtramos por el atributo “year” y la relación con la instancia “mention” obtenida a través de la clave foránea “group”. Para cada una de dichas instancias obtenidas creamos una nueva instancia “Block” asociada. Y llamamos a los métodos de la clase Block: create_field_blocks() y create_field_mentions().

Al actualizar la instancia:

Obtenemos, mediante una llama a la base de datos, todas las instancias “Block” filtradas por los atributos “turn” y “name”, previo a la modificación. Y actualizamos los campos “name” y “credits” de todas ellas.

+delete_admin():void

Se filtran todas las instancias “Block” con el mismo atributo “name” y se eliminan previamente todas las relaciones con “Shift”, para evitar el borrado de dichas instancias, y posteriormente se elimina dicha instancia “Block”.

+create_filed_blocks():void

Añade al campo “blocks” todas las referencias a las instancias “Block” con el mismo atributo “name”.

+create_filed_mentions():void

Sí el atributo “type” es igual a “COMMON”, se añaden todas las claves foráneas de “Mention” distintas.

+update_field_blocks():void

Se filtran las instancias “Block” por el “name”, excluyéndose la instancia que llamó al método, y se añaden todas al atributo “blocks. Posteriormente, se recorren dichas instancias añadiéndole al campo “blocks” de cada uno la instancia “Block” que llamó al método.

+update_field_mention(mention:Mention):void

Si el atributo dado “mention” no está contenido en el campo “mentions” de la instancia llamante y “type” es de tipo “COMMON” se añade la instancia “mention” al campo “mentions” de la instancia que llamó al método.

+assign_shifts_to_block(shifts:*Shift, schedule:Schedule):void

Recorremos cada instancia “Shift” referenciada en la lista “shifts”, y a cada una de ellas le asignamos como clave primaria su correspondiente de la instancia “Shift” con el mismo “day” y el mismo “initial_hour” pero del “schedule” pasado por referencia. Además, para cada una de las instancias “Shift”, actualizamos el campo “schedule” con el pasado por referencia y el campo “subject” con la instancia “Block” que realizó la llamada.

+def duplicate_optional_subjects(optional_subjects:*OptionalSubject):void

Por cada instancia de la lista “optional_subjects” se duplica la instancia y se modifica el atributo “block” asignándolo al que realizó la llamada al método. Y se guarda.

Clase Linked*Subject*

Relaciones

LinkedSubject/LinkedSubject de * a *

Tiene una restricción única conjunta (*unique_together*) del atributo “name” y la clave foránea “group”.

Métodos

Hereda el método borrar de la superclase “Subject”.

+save_admin():void

Al crear la instancia:

Filtramos las instancias “Group” por el campo “course” y por cada una de ellas creamos una nueva instancia “LinkedSubject” asociada a ese grupo.

Al actualizar la instancia:

- Obtenemos la instancia actual y la referencia de todas sus “LinkedSubjects” asociadas. Si hemos eliminado relaciones asociadas lo que haremos será eliminar todas las asociaciones con las otras “LinkedSubjects” y actualizarlo con la nueva lista. Posteriormente, desasignamos todas las instancias “Shift” de su campo “shifts” y añadimos esta instancia “linkedSubject” a cada “linkedSubject” seleccionadas en linked_subjects.

- Obtenemos, mediante una llamada a la base de datos, todas las instancias “LinkedSubject” filtradas por la clave foránea “course” y el atributo “name”, previo a la modificación. Y actualizamos los campos “name” y “credits” de todas ellas, incluida la instancia que realizó la llamada al método.

Clase *Classroom*

Relaciones

Classroom/Group de 1 a *

Classroom/SharedSubject de 1 a *

Classroom/OptionalSubject de 1 a *

Restricciones

Tiene una restricción a nivel de relación Classroom/Group que comprueba desde el “controlador del administrador” (admin.py) que el atributo “capacity” no es inferior al atributo “length” de ninguna de las instancias “Group” relacionadas.

Tiene una restricción a nivel de relación Classroom/SharedSubject que comprueba desde el “controlador del administrador” (admin.py) que el atributo “capacity” no es inferior al atributo “capacity” de ninguna de las instancias “SharedSubject” relacionadas.

Tiene una restricción a nivel de relación Classroom/OptionalSubject que comprueba desde el “controlador del administrador” (admin.py) que el atributo “capacity” no es inferior al atributo “capacity” de ninguna de las instancias “OptionalSubject” relacionadas.

Métodos

+delete():void

Limpiamos las relaciones de los campos “groups”, “sharedsubject_set” y “optional_subjects” y posteriormente eliminamos la instancia.

Clase Shared*Subject*

Métodos

+save_admin():void

Al crear la instancia:

Filtramos las instancias “Group” por las claves foráneas “course” y el atributo “year”, obtenido a través del campo “group”; y por cada una de ellas creamos una nueva instancia “SharedSubject” asociada a ese grupo. Y llamamos al método interno update_fields_shared_subjects() que actualiza el campo “shared_subjects”,el cual es una lista con las “SharedSubject” relacionadas.

Al actualizar la instancia:

Filtramos las instancias “SharedSubject” por el atributo “name” y los campos “turn” y “year”, obtenido a través de la clave foránea “group”. A cada una de estas instancias, se le actualizan los atributos “name” y “credits”.

+update_fields_shared_subjects():void

Filtramos las instancias “SharedSubject” por el atributo “name” y los campos “turn” y “year”, obtenido a través de la clave foránea “group”. A cada una de estas instancias, se le actualizan el campo “shared_subjects” con todas estas instancias, excepto ella misma.

Clase Optional*Subject*

Relaciones

OptionalSubject/Classroom de * a 1

OptionalSsubject/Block de 1..* a 1

Restricciones

Tiene una restricción a nivel de relación OptionalSubject/Classroom que comprueba desde el “controlador del administrador” (admin.py) que el atributo “capacity” de la instancia “OptionalSubject” no es superior al atributo “capacity” de ninguna de las instancias “Classroom” relacionadas.

Tiene una restricción única conjunta (*unique_together*) del atributo “name” y la clave foránea “block”.

Métodos

+delete_admin():void

Se eliminan todas las “OptionalSubject” con el mismo “name”, y “name” de la instancia “Block”, obtenido a través de la clave foránea “block”.

+save_admin():void

Al crear la instancia:

Filtramos las instancias “Block” por el campo “name”, obtenido a través de la clave foránea “block” y por cada una de ellas creamos una nueva instancia “OptionalSubject” asociada a esa instancia “Block”.

Al actualizar la instancia:

Se filtran todas las “OptionalSubject” con el mismo “name”, y “name” de la instancia “Block”, obtenido a través de la clave foránea “block”; y se actualizan los atributos “name” y “capacity” y la clave foránea “classroom”.

+check_and_assign(new_classroom:Classroom): String, String

Comprueba que la capacidad de la clase es suficiente, es decir, si el atributo “capacity” de la instancia “OptionalSubject”, que llamó al método, es menor que el atributo “capacity” de “new_classroom”. Si es así, comprueba que “new_classroom” es diferente a la actual referencia de “classroom” y; por último, se llama al método

“is_assignable_to_classroom(new_classroom)” que comprueba que la instancia que referencia “classroom” no está ya ocupada por otra instancia “OptionalSubject” del mismo bloque y por último llama al método save() de “OptionalSubject”, que actualiza el atributo “classroom” de todas las “OptionalSubjects” con el mismo “name” y “name” de la instancia “Block” relacionada. En este caso, los mensajes que se devuelven son satisfactorios.

Si no se ejecuta por este orden el método los mensajes son de error.

En cualquiera de los casos se devuelve los mensajes “code” y “message”.

+is_assignable_to_classroom(new_classroom:Classroom): Boolean

Se obtiene todas las instancias relacionadas de “OptionalSubject” a través de la clave foránea “block”, excluyéndose a sí misma, y filtradas por “new_classroom”.

Si la lista obtenida no está vacía significa que no puede ser asignada dado que ya hay otra “OptionalSubject” del mismo “Block” en esa instancia “Classroom” y, por tanto, devolvemos *False*. En caso contrario, devolvemos *True.*

+unlink_classroom(): String, String

Limpia/Elimina la relación de la clave foránea “classroom” y llama al método save() de “OptionalSubject”, que actualiza el atributo “classroom” de todas las “OptionalSubjects” con el mismo “name” y “name” de la instancia “Block” relacionada. En este caso, los mensajes que se devuelven son satisfactorios. En caso contrario son mensajes de error.

En cualquiera de los casos se devuelve los mensajes “code” y “message”.

Clase *Schedule*

Relaciones

Schedule/Group de * a 1

Existe una relación de composición entre Schedule y Shift, de forma que un Schedule está compuesto de Shifts

Restricciones

Tiene una restricción única conjunta (*unique_together*) del atributo “semester” y la clave foránea “group”.

Métodos

+save_admin():void

Al crear la instancia:

Guardamos la instancia “Schedule”.

Creamos 15 instancias “Shift” (3 franjas horarias x 5 dias), en sus respectivos días y franjas horarias, estás últimas van en función del valor del atributo “turn”, “MORNING” o “AFTERNOON”, de la instancia “Group” relacionada, a través de la clave foránea “group”.

Clase *Shift*

Relaciones

Shift/Subject de * a 1

Métodos

Dispone de los métodos propios de guardar y borrar pre-definidos por Django.

(Almacena una instancia con sus datos o borra la susodicha instancia)

Modelo Entidad-Relación

Diagrama

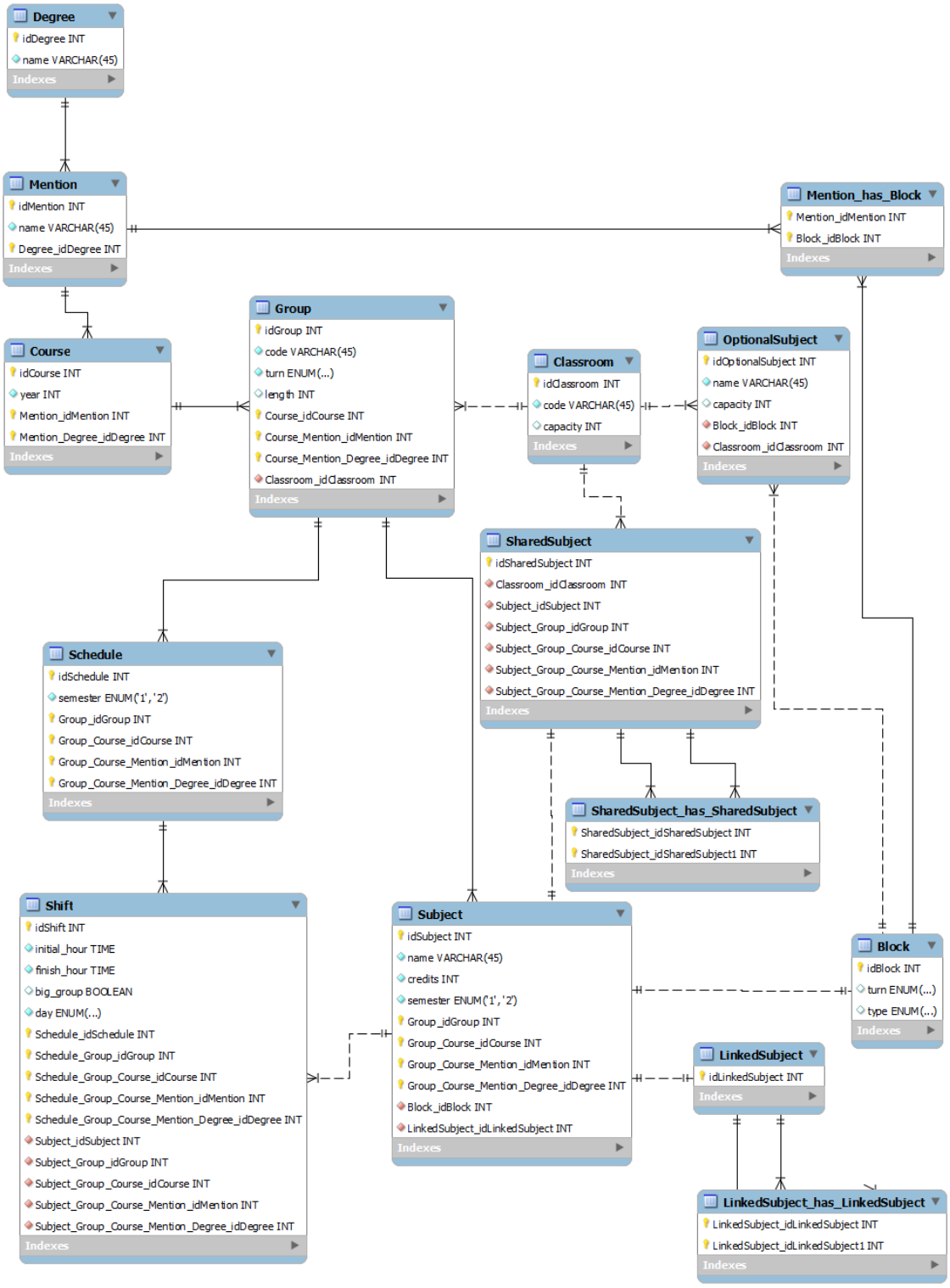

Ilustración 26. DIAGRAMA ENTIDAD-RELACIÓN

En el diagrama de Entidad-Relación podemos observar, de una forma visual, las distintas relaciones que componen nuestro modelo así como las claves primarias y/o foráneas que los componen y si dichas relaciones son “fuertes” o “débiles”. También se pueden

observar las nuevas tablas creadas cuando se tratan de relaciones muchos a muchos como en el caso de Bloque con Mención o las asignaturas ligadas consigo mismas o lo mismo en el caso de las asignaturas compartidas. Dichas relaciones consigo mismas se deben a que comparten ciertos atributos entre ellas como es el horario, por ejemplo.

Las diferentes relaciones que conforman nuestro modelo quedan recogidas en el sub-apartado previo.

Diagramas de Actividad

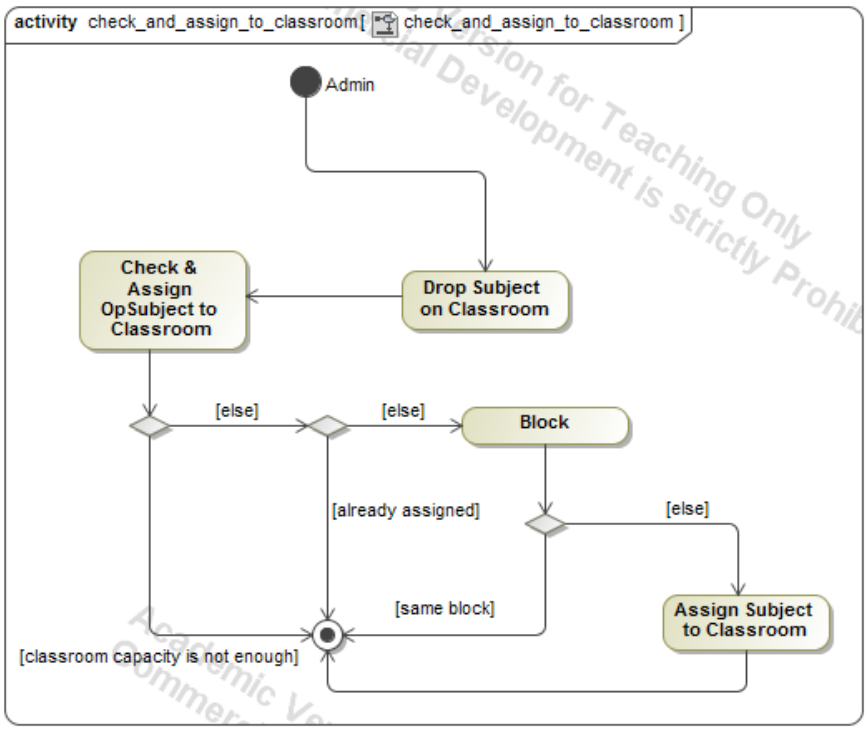

Ilustración 27. DIAGRAMA DE ACTIVIDAD: ASIGNACIÓN DE ASIGNATURA OPTATIVA A UN AULA

Mediante un sencillo diagrama de actividad podemos observar el camino de ejecución que sigue el sistema cuando el usuario arrastra una asignatura optativa a una clase. Tras arrastrar la asignatura al aula, el sistema realiza tres comprobaciones: comprueba que la capacidad del aula es suficiente para albergar los posibles alumnos de dicha asignatura, comprueba que la asignatura optativa no está actualmente asignada a un aula y que el aula no alberga una asignatura del mismo bloque, por motivos de compatibilidad de horario (posible solapamiento). Si todas las comprobaciones son satisfactorias se procede a la asignación de la asignatura optativa en la susodicha aula. En caso contrario, se termina el proceso, es decir, se cancela la asignación y se le muestra un mensaje por pantalla al usuario indicando el motivo de dicha cancelación.

Su desarrollo técnico queda explicado en el apartado: Clases del Sistema, en el método “check_and_assign(new_classroom:Classroom): String, String” de la clase “Optional Subject”.

Diagramas de Secuencia

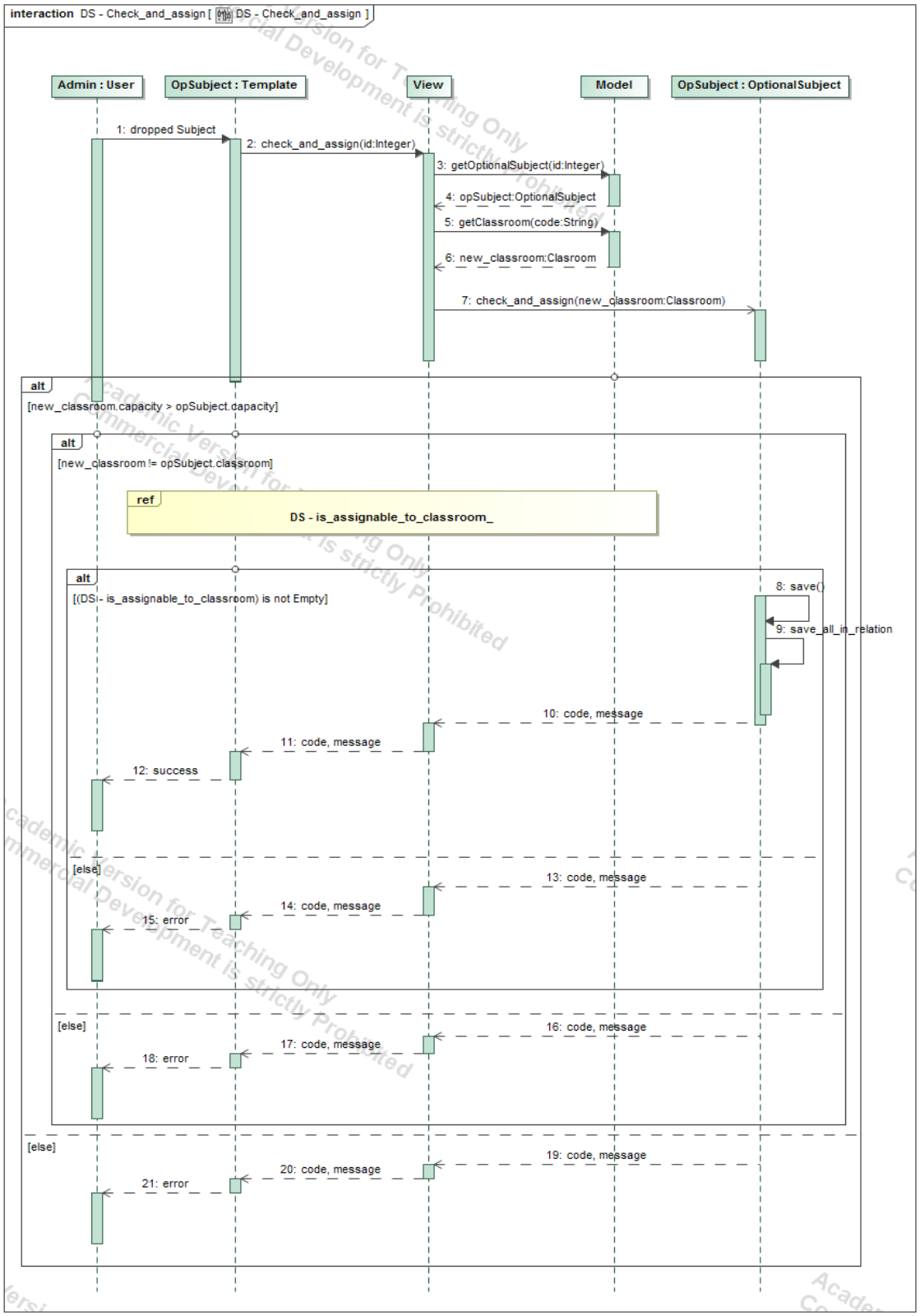

Ilustración 28. DIAGRAMA DE SECUENCIA: ASIGNACIÓN DE UNA ASIGNATURA OPTATIVA A UN AULA

En el diagrama de secuencia podemos observar de una forma más detallada, desde un punto de vista del sistema, las sucesivas llamadas internas que se van lanzando.

El usuario desde la interfaz, arrastra una instancia de tipo asignatura optativa a un aula, al hacerlo, de forma dinámica, mediante Ajax, se le pasa un mensaje al controlador con los datos, el cual le recoge las instancias de aula y asignatura desde el modelo, con los datos que le fueron enviados. Posteriormente se llama al método “check_and_assign” de la clase “OptionalSubject” y el sistema realiza tres comprobaciones: comprueba que la capacidad del aula es suficiente para albergar los posibles alumnos de dicha asignatura, comprueba que la asignatura optativa no está actualmente asignada a un aula y que el aula no alberga una asignatura del mismo bloque, por motivos de compatibilidad de horario (posible solapamiento). Si todas las comprobaciones son satisfactorias se procede a la asignación de todas las asignaturas optativas relacionadas (con el mismo nombre) en la susodicha aula. En caso contrario, se termina el proceso, es decir, se cancela la asignación y se le muestra un mensaje por pantalla al usuario indicando el motivo de dicha cancelación.

Su desarrollo técnico también queda explicado en el apartado: Clases del Sistema, en el método “check_and_assign(new_classroom:Classroom): String, String” de la clase “Optional Subject”.

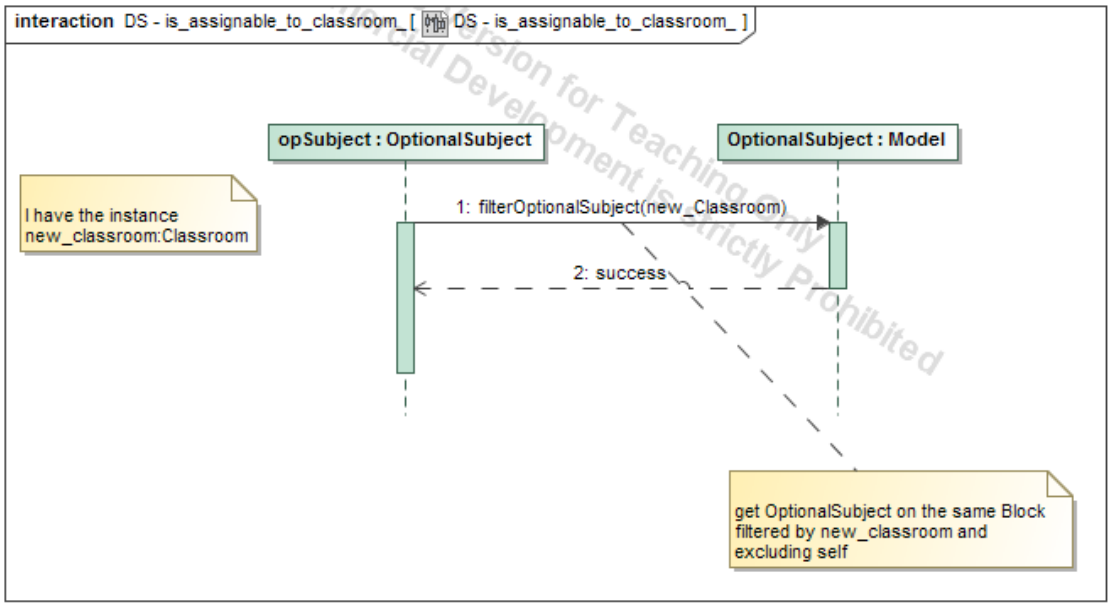

Ilustración 29. DIAGRAMA DE SECUENCIA: COMPROBACIÓN OPTATIVA ASIGNABLE A UN AULA

Este diagrama de secuencia es bastante simple y forma parte de un diagrama superior. En él se observa que se está realizando una consulta mediante el método “filterOptionalSubject(new_Classroom)” el cual comprueba si existe una asignatura optativa del mismo bloque en el aula dada como argumento. Devolviendo si es verdadero o falso.

Su desarrollo queda explicado en el apartado previo: Clases del Sistema, situado en el método "is_assignable_to_classroom(new_classroom:Classroom): Boolean", el cual puede encontrarse en la clase "OptionalSubject".

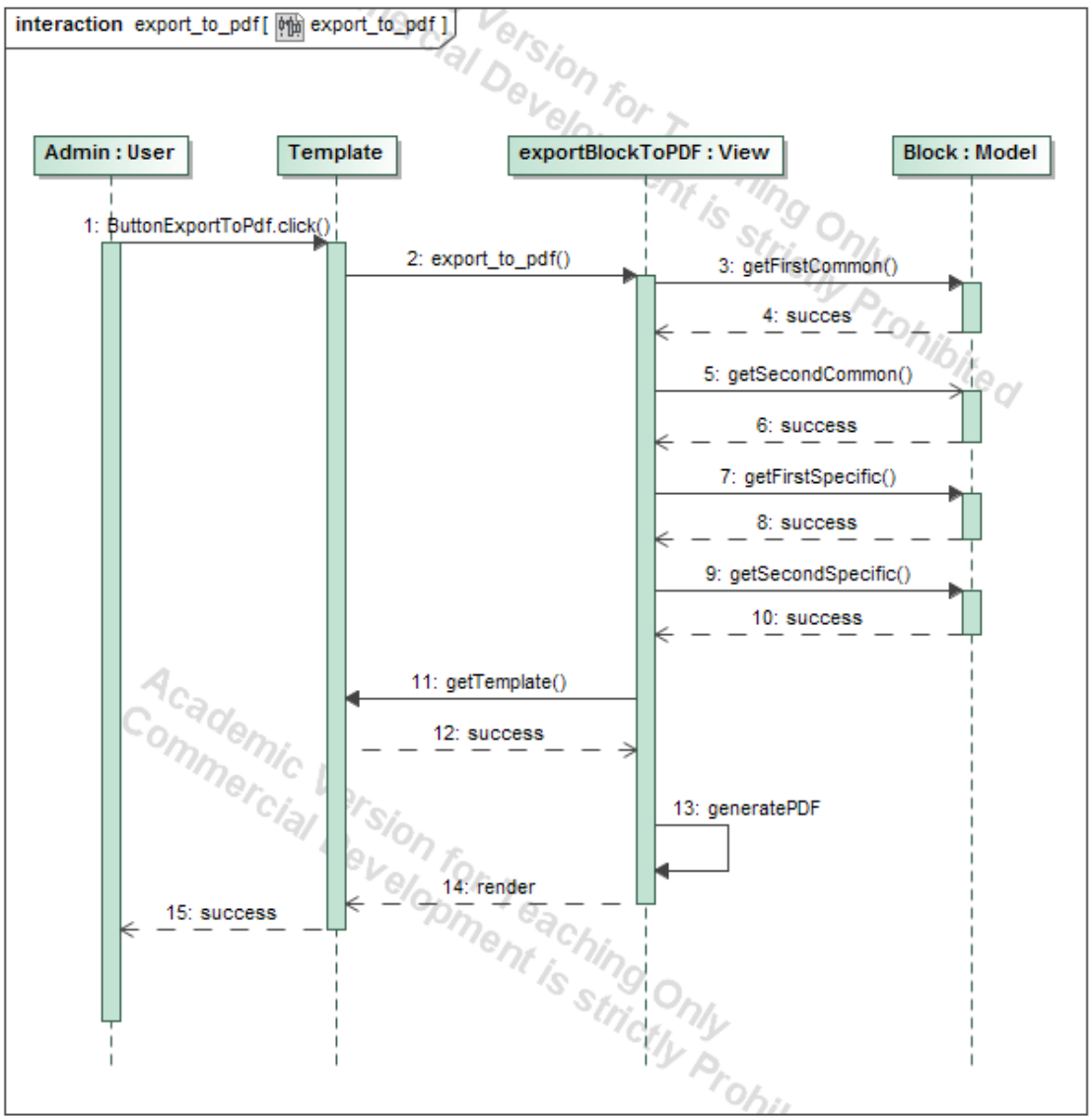

Ilustración 30. DIAGRAMA DE SECUENCIA: EXPORTAR INFORMACIÓN DE BLOQUES A PDF

Realiza cuatro llamadas al modelo obteniendo así los bloques de asignaturas divididos por semestres y por tipo (común y específica), luego se asignan dinámicamente (*renderizan)* los datos en la plantilla dada y se genera un archivo, formato PDF. Finalmente se le vía la información a la interfaz para que el usuario puedo visualizarlo desde el navegador y descargarlo si así lo quisiese.

6. Implementación y Pruebas

En este apartado se pretende dar una visión más personal sobre las aportaciones individuales que realizó el alumno al proyecto. Se pretende no ser redundante por lo que se obviaran algunos de los detalles referentes a la parte técnica o que ya han sido desarrollados y explicados anteriormente.

Implementación de la Funcionalidad

Expresándolo de una forma un tanto más coloquial, a continuación, se explicara el desarrollo y aportación que tuve personalmente en las distintas partes del proyecto dividiéndolo por funcionalidades o páginas:

Archivo HTML base

En este archivo colaboré con la importación de algunas de las librerías necesarias para el proyecto y la distribución por bloques necesaria como esqueleto del proyecto. Una de las últimas librerías importadas se trataba de JQuery UI Touch Punch, la cual permite utilizar los eventos de toque de JQuery UI en dispositivos móviles y tabletas (*tablets*).

Selectores

En varias de las vistas de la aplicación es necesario filtrar la información que aparece en función de ciertos datos, como pueden ser la Titulación, la Mención, el Curso, el Grupo, el Semestre, o incluso el día y el turno. Para ello me encargué un sistema de selectores que permitieran decidir qué información es la que se utiliza para filtrar los datos que vemos en cada momento.

La principal complicación que tenía esto es que algunos de los selectores dependían de otros, en el sentido de que por ejemplo las opciones que aparecen en el selector correspondiente a las menciones dependen de la opción que haya seleccionada en el de las titulaciones.

Por ello utilizando las herramientas proporcionadas por Marionette construí un sistema de eventos en el cual cada vez que se seleccionaba un dato en un selector, se disparará un evento el cual hacía que la información de los selectores que dependían de él se actualizara y la información que aparece en la página se actualizara también de forma dinámica. Así la información que uno ve en la página es siempre consistente y no da lugar a confusión.

Drag&Drop

En este apartado uno de mis compañeros se encargó de construir un prototipo funcional de un sistema Drag&Drop utilizando JQuery UI, que permitiera insertar y eliminar objetos de una tabla similar a la que se encuentra en la aplicación, de forma externa, con un HTML y un CSS generado por cuenta propia. Posteriormente, me encargué de adaptar este prototipo para la interfaz de asignación de asignaturas optativas a aulas de forma que se integrase perfectamente con Marionette y con la parte visual de nuestra aplicación.

Barra de Navegación (NavBar)

Para esta sección que es heredada en la mayoría de páginas, me encargué de añadir el botón que permitiera iniciar y cerrar la sesión; y el botón para acceder al módulo de administrador. Por otro lado los enlaces que permitieran navegar a mis páginas asignadas. También me encargué del diseño CSS de los botones.

Página Principal

En esta vista me encargué de añadir los enlaces a las vistas que yo había creado, para una identificación más rápida y visual de estas.

Página de Información de Bloques de Asignaturas Optativas

Construí íntegramente está página, para ello construí un esqueleto HTML con sus respectivas plantillas para rellenarlas a partir de las vistas de Marionette. La información se divide por tipos de bloques (comunes y específicos) y esto a su vez por semestres, generándose un total de cuatro tablas que contienen los bloques con el turno, horario y días en los que se imparte y a su vez se detallan las asignaturas que albergan los bloques y en que aula se imparte cada una de ellas.

La página dispone de selectores para filtrar la información de los bloques de tipo específico, así que tuve que construir la estructura para que cuando cambiaran estos selectores se actualizara la información pertinente a dicha región.

Por otro lado, la página consta de un botón que permite generar un PDF sobre una plantilla que se ha construido manualmente para que siga unos estilos definidos.

Para esto hice uso de la librería xhtm2pdf mencionada en la introducción, mediante la cual podía formatear, en formato PDF, un HTML, *renderizado* previamente con la información pertinente.

Página de Gestión de Asignaturas Optativas y Aulas

Esta parte del proyecto, podemos dividirla en dos partes, por un lado está el lado del *frontend* y por otro el *backend*. En la parte del *frontend*, como ya se ha indicado anteriormente, me encargué de construir el sistema Drag&Drop utilizando funciones de JQuery y JQuery UI, que permitían coger elementos de una tabla y arrastrarlos a otra en la cual se depositaban.

Posteriormente me encargué de integrar este prototipo con las vistas de Marionette, de forma que los ItemView que forman cada tabla, se comportaran igual que los elementos del prototipo creado por mi compañero pero cambiando detalles de visualización.

Cuando se asigna alguna asignatura optativa a las aulas que forman parte de la tabla, antes de realizar la asignación definitiva, es necesario realizar una serie de

comprobaciones asegurando que esta cumple todas las restricciones de la funcionalidad.

Por ello, para realizar estas comprobaciones, que se hacen en el *backend*, debido a que no son comprobaciones simples, se realiza una llamada AJAX a través de las funciones que JQuery proporciona para ello. Esta llamada AJAX invoca a la vista correspondiente de Django que se encarga de gestionar todo el proceso tanto de comprobación de restricciones y de asignación de la asignatura optativa al aula en caso de cumplirse estas. Seguidamente se muestra un mensaje temporal indicando lo sucedido durante la asignación, teniendo constancia si se produjo la asignación o se produjo una situación de restricción que impidió la asignación de la misma.

A la hora de asignar una asignatura optativa a un aula, hay que tener en cuenta unas restricciones básicas. Dichas restricciones comprueban por un lado que el aula tiene capacidad suficiente para albergar o todos los alumnos de la asignatura y otra restricción comprueba que un aula no contiene dos o más asignaturas del mismo bloque, puesto que ello supondría que los horarios tendrían un solapamiento.

En el *frontend,* todos sigue un estilo de colores que permite al usuario tener una idea principal del estado de cada asignatura, generándose dinámicamente, al arrastrar el puntero sobre dicha asignatura en el navegador, la información de la capacidad de la asignatura y el aula a la que se encuentra asignada, si es que procede, distinguiéndose por colores si dichas asignaturas han sido asignadas o no. Al pulsar sobre las asignaturas, mediante estilos con CSS se transforma la asignatura seleccionada en una bola de color con sus iniciales y de esa forma pueden asignarse a las aulas de una forma más sencilla y a su vez se muestra por colores las aulas que pueden ser asignadas, son demasiado pequeñas o contienen una asignatura del mismo bloque,

Módulo de Administración

Este se trata de uno de los módulos más importantes de la aplicación puesto que se encarga de la inserción, consulta, eliminado y modificación de los datos que contiene nuestra aplicación. Podríamos dedicarle un buen tiempo a explicar dicho módulo pero puesto que nos imaginamos que se sobreentiende su funcionalidad vamos a resumirlo de una forma breve y concisa.

Tenemos varias vistas que nos permite consultar los datos, internamente se nos permite eliminar los datos. Podemos seleccionar la creación de un nuevo objeto, lo cual nos llevaría su correspondiente vista con el formulario de creación, con sus restricciones pertinentes cuando procedemos a guardar los datos. También podemos acceder a la edición de los mismos para cambiar determinados valores.

Las páginas, aunque siguen el estilo de administración del *framework* Django, con el cual estamos trabajando, están "*customizadas"* para tener un control personalizado de las mismas.

Por un lado la parte visual del módulo de administrador, tras valorar el tiempo de desarrollo, consideramos que era una parte secundaria, centrándonos en el *backend* o parte lógica para el correcto funcionamiento de mismo. Dicha generación y/o edición tienen una lógica compleja y por ello, en el momento de guardar, los métodos han sido sobre-escritos.

Pruebas

Las pruebas realizadas podemos dividirlas en dos tipos:

- ✓ Pruebas de Usuario (manuales) realizadas a lo largo del desarrollo del proyecto.
- ✓ Pruebas Manuales sobre un servidor real en la nube (Amazon Web Services)

La aplicación ha sido desplegada sobre un servidor real en la nube, concretamente en Amazon Web Services, probando su funcionamiento satisfactoriamente. También fue comprobado su funcionamiento sobre un iPad (dispositivo móvil), dando unos resultados satisfactorios y siendo necesario corregir únicamente algún detalle en la interfaz de usuario.

7. Conclusiones y Líneas de Trabajo Futuras

Objetivos Cumplidos

Tras una evaluación general del proyecto se puede afirmar que los objetivos del proyecto, planteados inicialmente, han sido cubiertos.

Tras un arduo trabajo se ha conseguido desarrollar un proyecto que integra nuevas tecnologías y el desarrollo de una aplicación web para la administración y organización interna de aulas, horarios, grupos, asignaturas y bloques, adaptado a las necesidades de la Universidad de Málaga.

Se han cumplido todos los objetivos individuales, generalmente:

- Sección de acceso e identificación.
- Módulo de administrador para crear, borrar, editar y consultar los datos.
- Sección para asignar/desasignar asignaturas optativas a aulas.
- Sección para consultar la información referente a los bloques (aula, asignaturas y horarios) y exportar dicha información a PDF.

Es cierto que por el camino se han redefinido algunos detalles planteados inicialmente. Por ejemplo, el hecho de hacer que la aplicación fuera totalmente adaptativa "*responsive*". Aunque en un principio era esta la idea, a lo largo del desarrollo nos dimos cuenta de que no tenía sentido utilizar la aplicación en dispositivos de pequeño tamaño como puede ser un móvil, porque debido a la forma en la que está estructurada la funcionalidad no resultaría práctica ni fácil de usar. Por ello la aplicación no está preparada para un correcto visualizado en estos dispositivos. Si bien es cierto que la aplicación sí está preparada para que funcionalmente no presente ningún tipo de problemas en dispositivos móviles, por lo que la aplicación se puede utilizar perfectamente en una tablet o dispositivo de mayor tamaño a un móvil convencional.

Dificultades Encontradas

En el Desarrollo…

Componentes tales como el *framework* de estilos "*Materialize*" se encuentran en fase de desarrollo con una API un tanto escueta y algunas dificultades con la integración conjunta debido a la fase de madurez en la que se halla.

SQLite, en este caso versión 3, soporta únicamente la distinción de objetos por clave primaria o foránea, lo que imposibilitó de una forma sencilla obtener los objetos como las asignaturas sin repeticiones y teniendo que filtrarlas manualmente aunque ello es más óptimo fue tedioso controlarlo para todos los casos, incluida la API.

En la Organización…

Por otro lado, hablando sobre la metodología de desarrollo que se ha usado en general.

Este proyecto era de una envergadura superior en términos de límites de tiempo para que fuera desarrollado por una única persona, por ello nos embarcamos tres personas en el desarrollo del mismo.

El hecho de contar con tres integrantes ha tenido sus puntos positivos y sus puntos negativos. Ha enriquecido la calidad del proyecto al poder complementar los conocimientos que los integrantes poseían. Por otro, lado la toma de decisiones y, sobre todo, el desarrollo inicial del proyecto se veía afectado, aún más, por el hecho de que personalmente me encontraba cursando un programa de intercambio de Estudios (ERASMUS +) en el extranjero y dificultaba las tareas de organización y toma de decisiones y desarrollo, prioritariamente en su etapa inicial.

Al formar parte de un grupo otro problema que se ha detectado en las fases iniciales del proyecto es que la sensación de responsabilidad está más dispersa, de forma que cuesta más implicarse durante esta etapa. Aunque este último problema se va resolviendo a medida que cada uno empieza a tener su trabajo de forma más individual y diferenciada.

La conclusión es que a pesar de sus dificultades ha sido una experiencia positiva y enriquecedora pues ha sido una experiencia real de lo que nos espera en el mundo laboral, dado que hoy en día es necesario tener la capacidad de trabajar a distancia y organizarse en grupos, pues no hay horizontes en nuestro trabajo.

Líneas Futuras

Posibles Ampliaciones

En la etapa actual donde se encuentra la aplicación dispone de una gran variedad de posibilidades de ampliación. Por una parte, es posible mejorar los módulos ya existentes. En el primer módulo de creación de horarios, tal y como está implementado ahora, cuando detecta que se va a producir una inconsistencia, no se realiza la asignación o intercambio y se muestra un error poco descriptivo sobre cuál ha sido ese problema. Una posible ampliación sería modificar este código para que devolviera códigos de error descriptivos que permitieran localizar dónde se está produciendo la inconsistencia que no nos permite realizar la acción que queremos.

Siguiendo en la misma línea de los códigos de error, en el segundo módulo se encuentra una situación similar, en la que se podría refinar los códigos de error para poder identificar más fácilmente el motivo que nos impide la asignación.

Con respecto al tercer módulo podría implementarse un control del número de asignaturas que están asignadas a cada aula para balancear la carga de las aulas.

Y con respecto al cuarto módulo de administración podría diseñarse una interfaz que continuara la filosofía y estética general de la aplicación, por ejemplo. También intentar unificar los tipos de asignaturas para el formulario de creación de forma que se generase dinámicamente el formulario en función del tipo de asignatura para agilizar su creación.

Continuando por otra línea totalmente diferente, es posible mejorar el funcionamiento de la aplicación. Pensando en un caso práctico, aunque una asignatura tenga asignada

varios huecos horarios, es posible que nunca se utilice el aula que tiene asignada ese grupo para recibir las clases. Por lo tanto sería una buena idea tener la posibilidad de tener un "*feedback*" o retroalimentación real del uso de las aulas para poder reasignar de una forma más óptima los recursos de nuestra facultad.

También se podrían añadir módulos totalmente nuevos a la aplicación. Un ejemplo sería un módulo para gestionar el uso de los laboratorios para clases prácticas, de forma que podría asignarse varios de los bloques horarios de una asignatura para impartirse en un lugar distinto al que tiene asignado el grupo en cuestión. Y conectando con esto último, también la posibilidad de reservar laboratorios para ocasiones especiales asegurándose de la disponibilidad de este en el momento en el que se quiera utilizar.

La app ha sido ideada para una posible implementación futura como módulo extra de la web Universitaria y, con ello, facilitar la organización en tiempo y recursos, dicha implementación sería una posible ampliación.

La aplicación web está pre-adaptada a dispositivos móviles iPad, pudiéndose adaptar completamente a posteriori.

Se podría añadir una sección que permita asignar/desasignar "asignaturas compartidas" a aulas.

En definitiva, se pueden pensar una variedad de opciones de ampliación para la aplicación si centramos nuestra línea de pensamiento en los usos que esta puede tener en un caso real del centro.

Referencias

PYTHON: https://www.python.org/

DJANGO: https://www.djangoproject.com/

UNDERSCORE: http://underscorejs.org/

BACKBONE: http://backbonejs.org/

MARIONETTE: http://marionettejs.com/

DJANGO TASTYPIE: https://django-tastypie.readthedocs.org/en/latest/

API RESTFUL USING TASTYPIE: http://blog.mathandpencil.com/using-django-tastypie-to-create-RESTful-APIs/

MATERIALIZE: http://materializecss.com/

JQUERY: https://jquery.com/

JQUERYUI: https://jqueryui.com/

MAGICDRAW: http://www.nomagic.com/products/magicdraw.html/

BALSAMIQ MOCKUPS: https://balsamiq.com/products/mockups/