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

Rafael Ronda García

TUTORIZADO POR

Eduardo Guzmán de los Riscos

DEPARTAMENTO

Lenguajes y Ciencias para la Computación

UNIVERSIDAD DE

MÁLAGA, SEPTIEMBRE DE 2015

Fecha defensa:

El Secretario del Tribunal

Resumen:

El proyecto consiste en el desarrollo de una aplicación web que permita la configuración propia de un centro universitario (creación de grados, menciones, cursos, grupos, asignaturas, etc) para su posterior organización. Como el tamaño del proyecto es bastante grande para que lo desarrolle una sola persona, este proyecto ha sido desarrollado por un equipo de tres personas, divididas en dos equipos.

Para dicha organización, la aplicación permite realizar la asignación entre asignaturas y franjas horarias para cada uno de los grupos registrados en la misma; teniendo en cuenta para esto las restricciones características de las asignaturas de tipo normal, *ligada*, *compartida*, *optativa común* y *optativa de mención*.

La aplicación permite también la asignación de aulas con grupos, así como con asignaturas optativas, teniendo en cuenta de nuevo restricciones que deben ser controladas a la hora de la asignación (aulas ocupadas por otros grupos o asignaturas optativas, tamaños de grupos y aulas, etc).

Por otro lado, el usuario de la aplicación dispone de varias vistas de información con distintos enfoques (vista general de horarios configurados, ocupación de aulas por asignatura, ocupación de aulas por grupos, etc), y puede también exportar en formato pdf la configuración de horarios previamente realizada. Por último, es soportado también por la aplicación la gestión de distintas configuraciones a lo largo de su uso (exportación/importación de datos desde/hacia la base de datos).

Palabras claves:

Gestión de horarios, gestión de aulas, aplicación web, SCRUM, XP (eXtreme Programming), Python, Django, Backbone, Marionette.

Abstract:

This project consists in elaborating a web application, which allows the user to manage a university center configuration (create degrees, mentions, courses, groups, subjects, etc) in order to organizing it later.

To be able to do that, the application allows the subject-scheduling shift assignment for each group registered; being aware of the normal, linked, shared, common optional and mention optional subject constraints.

The application also allows the classroom-groups/optional subject assignment, respecting again the constraints that should be aware of at the assignment (busy classrooms, groups and classrooms capacity, etc).

By other side, the user can use a set of information pages of different kind (schedules information view, subject-classroom assignment, group-classroom assignment, etc), and can exports the configured schedules to pdf as well. Finally, the application supports exporting/importing several configurations along it use (exporting/importing data from/to de database).

Keywords:

Schedule manager, classroom manager, web application, SCRUM, XP (eXtreme Programming), Python, Django, Backbone, Marionette.

Índice:

Introducción

Motivación y objetivos del proyecto:

La configuración y asignación de las aulas en un centro universitario es un proceso tedioso, ya que son muchos los factores que hay que tener en cuenta para no llegar a una configuración no posible, como por ejemplo la de tener más de un grupo asociado al mismo aula.

En vista a esta dificultad, y con el fin de automatizar todo el proceso de control de restricciones, a la hora de realizar distintas configuraciones en distintos centros universitarios, es propuesto este proyecto, en el cual se pretende hacer foco en la usabilidad, haciendo de este proceso algo sencillo, automático y despreocupado. Para esto hemos hecho uso también de interfaces dinámicas, sistemas Drag&Drop, etc.

Tecnologías:

En este apartado diferenciaremos tres grandes bloques de tecnologías usadas: herramientas y aplicaciones para el desarrollo, backend y frontend.

- Herramientas y aplicaciones para el desarrollo:
    - VirtualBox:

Gestor de máquinas virtuales utilizado para trabajar con el SO sobre el cual hemos desarrollado el proyecto.

    - Sistema operativo Ubuntu:

Distribución del sistema operativo Linux. Debido sobre todo al uso de las siguientes tecnologías a describir en este punto, Ubuntu ha sido el SO bajo el cual hemos desarrollado el proyecto, además de la familiarización con los comandos del terminal del SO.

    - MagicDraw:

Editor de diagramas UML, usado en la fase de diseño para la generación de diagramas de clases y secuencia, sobre los cuales trabajar posteriormente

- Balsamiq Mockup:

Maquetador de interfaces web y móviles, usado en la fase de diseño para la creación de mockups de las diferentes páginas web a desarrollar en el proyecto

- Pip

Gestor de paquetes Python de PyPI, usado para gestionar la descarga tanto del *framework* usado durante el desarrollo como de las diferentes librerías externas incluidas en el mismo y que están expuestas en este apartado del documento.

- VirtualEnv

Generador de entornos virtuales. Permite la creación de *sandboxes*, mediante los cuales puedes gestionar las dependencias de diferentes proyectos sin el riesgo de crear conflictos en el sistema por incompatibilidad de paquetes, etc.

- Virtualenv Wrapper

Extensión a VirtualEnv. Realiza la configuración mediante la cual se organizan todos los entornos virtuales bajo el mismo directorio y proporciona comandos de consola para poder trabajar con estos entornos de forma más cómoda y rápida.

- Pycharm

IDE sobre el cual se ha desarrollado el proyecto. Soporta compatibilidad con las tecnologías ya mencionadas al ser un IDE pensado para desarrollar en python, así como con el *framework* de Python Django, el cual está descrito posteriormente en este mismo documento.

- Asana

Herramienta de gestión de proyectos. Nos ha permitido gestionar las tareas mediante la asignación de responsables y fechas, así como generar una gráfica que refleja el flujo de trabajo a lo largo del desarrollo del proyecto.

- Git

Como gestor de repositorios. Hemos hecho uso de él a través de la plataforma web BitBuquet.

- Backend:

  - Python

Lenguaje de programación interpretado y de tipado dinámico, lo que lo hace también hace fácil trabajar con él usando un lenguaje natural y, por lo tanto, legible. Este es el mayor motivo por el cual hemos elegido Python como nuestro lenguaje de programación para el backend de nuestro proyecto, además de la potencia que ofrecen sus características funcionales, al ser un lenguaje multiparadigma.

  - Django

Framework Modelo-Vista-Plantilla basado en Python. La filosofía de Django promueve el reúso de componentes, el desarrollo rápido de entornos web, y el principio DRY (Don't Repeate Yourself). Por esto y por la cantidad de librerías ofrecidas para el entorno y de documentación acerca del correcto desarrollo mediante este framework, debido a que es de código abierto, hemos usado Django como núcleo de desarrollo para este proyecto.

  - Tastypie

Aplicación de Django que permite la fácil creación de una API Rest a partir de la definición de recursos asociados a los modelos definidos en el sistema. Tastypie permite también la definición de filtros y modificadores de contenido propios para controlar el contenido y representación de los datos obtenidos mediante las llamadas a sus recursos.

  - xhtml2pdf

Librería de Django usada para la fácil renderización en pdf del contenido de un HTML definido y renderizado previamente.

- Frontend
    - JQuery

Hemos hecho uso de JQuery por la gran cantidad de facilidades que proporciona en cuanto a la gestión de eventos, manipulación HTML, llamadas Ajax, etc, además de por haber hecho uso de tecnologías que trabajan sobre ella.

    - JQueryUI

Únicamente hemos hecho uso del módulo de Drag&Drop proporcionado por JQueryUI, con el fin de hacer un sistema más intuitivo a la hora de trabajar con él.

    - JQueryUI Touch Punch

Librería que extiende JQueryUI, Hemos hecho uso de un módulo que facilita la interpretación de eventos táctiles a eventos de ratón.

    - Underscore

Framework que trabaja sobre JQuery usado como dependencia y como renderizador de contenido dinámico en plantillas HTML encapsuladas en scripts JS.

    - Backbone

Framework que trabaja sobre Underscore. Backbone proporciona una estructura Modelo-Vista-Controlador incrustada en la Vista de la estructura de una aplicación web. Esto significa que permite la creación y encapsulado de modelos en el frontend, correspondiéndose estos con los ya definidos en la base de datos; definir colecciones de estos modelos y vistas que trabajan como controladores sobre las plantillas HTML en las cuales se integran.

    - Backbone-Tastypie

Librería JS que integra la librerías ya mencionadas Tastypie y Backbone. Backbone-Tastypie toma como entrada la respuesta de la llamada a un recurso Tastypie, interpretando el contenido y encapsulándolo en colecciones de Backbone. También establece la sincronización entre los modelos de Backbone y los propios de la base de datos. Mediante esto podemos sincronizar los datos en el servidor conforme se van actualizando en el cliente y viceversa.

- Marionette

Framework que trabaja sobre Backbone. Proporciona potentes vistas sobre las cuales trabajar para el fácil renderizado dinámico de contenido en las diferentes páginas que componen la aplicación web, control de elementos de interfaz, eventos, facilidades para el renderizado de diferentes estructuras típicas de las páginas web, etc.

- Materialize

Reciente framework CSS aun en fase de desarrollo que proporciona numerosos componentes basados en Material Design en cuanto a visualización y animación.

Estructura de la memoria:

Las secciones que siguen a partir de ahora contienen el cuerpo principal de la memoria, la cual está dividida en cinco partes: una primera donde se hace una definición general del proyecto y otras cuatro que se corresponden con las cuatro fases del proceso de desarrollo software.

Descripción general del proyecto

Ya en la introducción hemos descritos los objetivos del proyecto, que son en resumen satisfacer una necesidad presente en el centro universitario, y para ello se planteó desarrollar una aplicación web que hiciera más fácil las tareas de gestión que hasta ahora pueden llegar a ser bastante tediosas y complicadas.

El proyecto consta de 3 módulos principales (figura 2 y 3) que abarcan toda la funcionalidad necesaria para cumplir con este propósito:

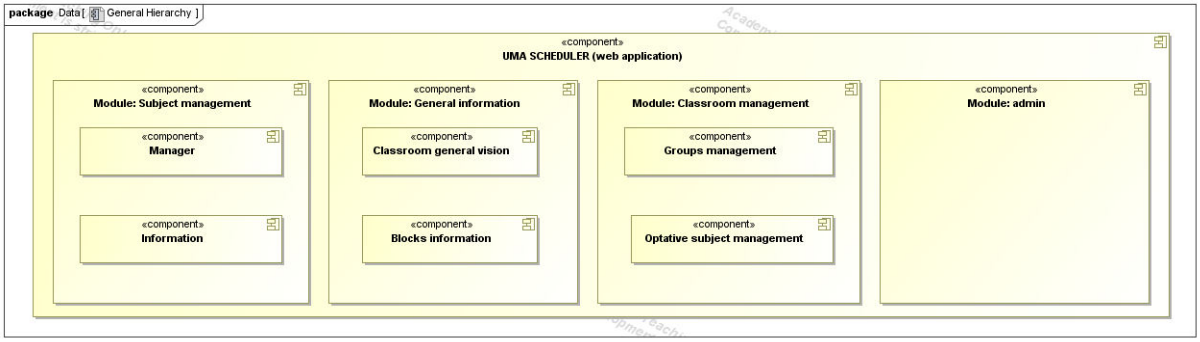

FIGURA 2 ARQUITECTURA GENERAL DE LA APLICACIÓN

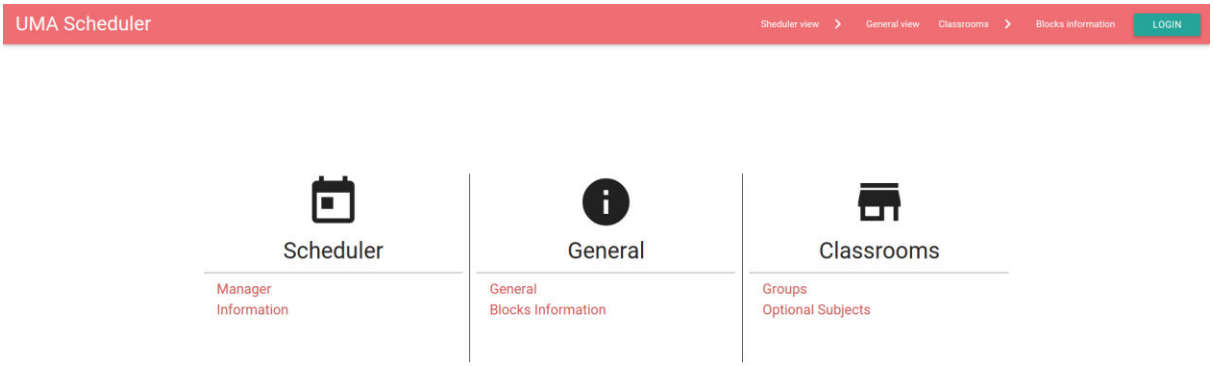

FIGURA 3 PÁGINA PRINCIPAL

Primer módulo - gestión de horarios (figura 4):

En este módulo será posible crear horarios para los grupos que existen en el sistema a través de un mecanismo Drag&Drop, con el cual podemos ir asignando las diferentes asignaturas a las franjas horarias que creamos convenientes.

Por debajo, la aplicación se encargará de realizar las comprobaciones pertinentes, descritas ya en los requisitos RF15, RF16 y RF17. Ya dependiendo del tipo de asignatura se comprobarán otros tipos de restricciones recogidas en los requisitos RF7, RF9, RF12, RF13 y RF14; ya que en el caso de tratarse de asignaturas ligadas, 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.

En este módulo también seremos capaces de intercambiar asignaturas, de forma que las restricciones se cumplan para ambas asignaturas. También podremos eliminar asignaciones realizadas previamente.

Por último, la aplicación nos permite seleccionar en este módulo 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 de esa asignatura.

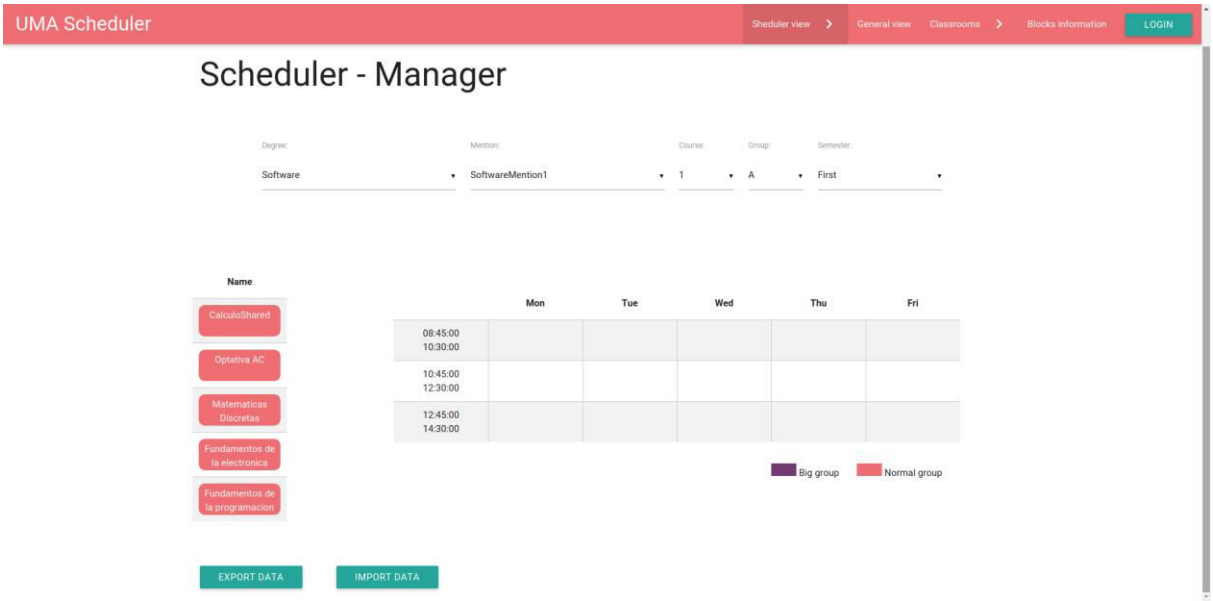

FIGURA 4 VISTA DE GESTIÓN DE HORARIOS

Segundo módulo - gestión de espacios en relación a los grupos (figura 5):

En un segundo módulo podremos tener un control sobre los espacios del centro en relación a los grupos que tienen lugar en un curso determinado. Esto quiere decir que podremos asignar los distintos grupos a las aulas existentes, teniendo en cuenta para ello tanto la capacidad del aula como su estado actual (asignada previamente a otro grupo, asignada a una asignatura optativa, vacía, etc), de esta forma evitamos conflictos, como por ejemplo que se imparta más de una asignatura a la vez en el mismo aula.

En la página correspondiente a este módulo, y referenciada en el requisito RF28, nos basamos en un código de colores para ilustrar de forma visual la información necesaria para distinguir entre los distintos tipos de estados del aula.

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

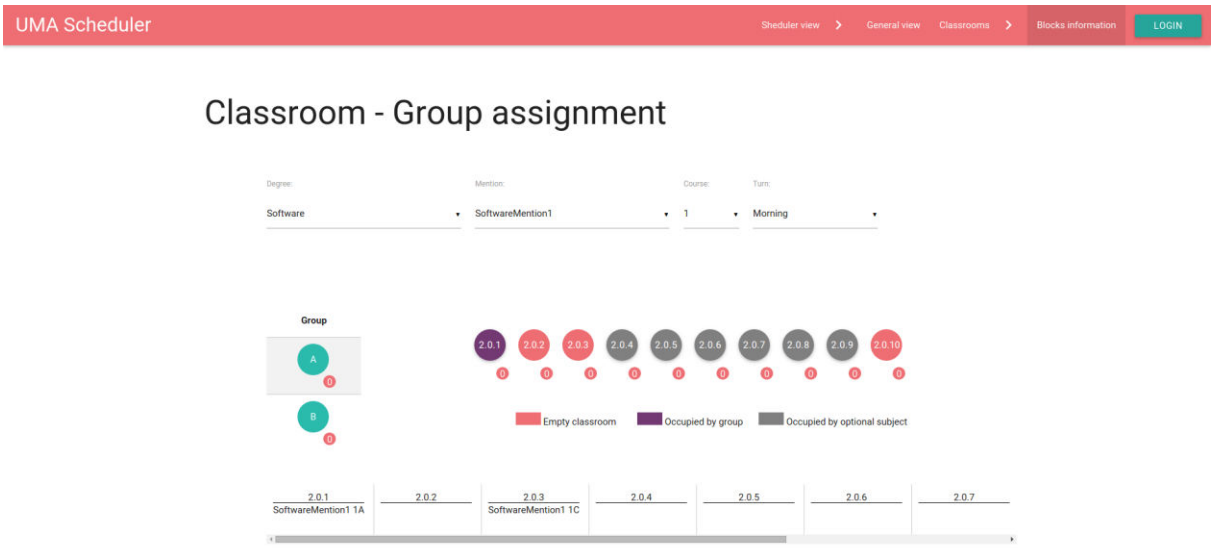

FIGURA5 5 PÁGINA DE GESTIÓN DE AULAS Y GRUPOS

Tercer módulo - gestión de espacios en relación a las asignaturas optativas (figura 6)

Al igual que en el segundo módulo, este tiene como tema principal la gestión de las aulas, pero en este caso lo que asignamos a estas son las asignaturas optativas.

Debido a esto, la lógica que funciona por detrás a la hora de realizar las asignaciones es dual con respecto al módulo anterior, es decir, las comprobaciones

a realizar están relacionadas con el estado de las aulas según los grupos asignados a ella y el estado de la configuración de los horarios de estos grupos. La vista es bastante similar, aunque con ciertos detalles distintivos necesarios para mostrar correctamente esta información, ya que en este caso asignamos las asignaturas optativas que a su vez dependen de los bloques en los que se ubican.

FIGURA 6 PÁGINA DE GESTIÓN DE AULAS Y ASIGNATURAS OPTATIVAS

Aparte de estos tres módulos funcionales, la aplicación proporciona una serie de vistas de información para complementar a estos módulos.

En relación al primer módulo, encontramos una vista de información sobre los horarios (figura 7). En esta vista se podrán consultar los horarios constituidos previamente en la aplicación, y para el uso de esta debe hacerse uso de los selectores para filtrar e ir viendo los horarios de cada grupo dividido por semestres. Por otro lado, 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.

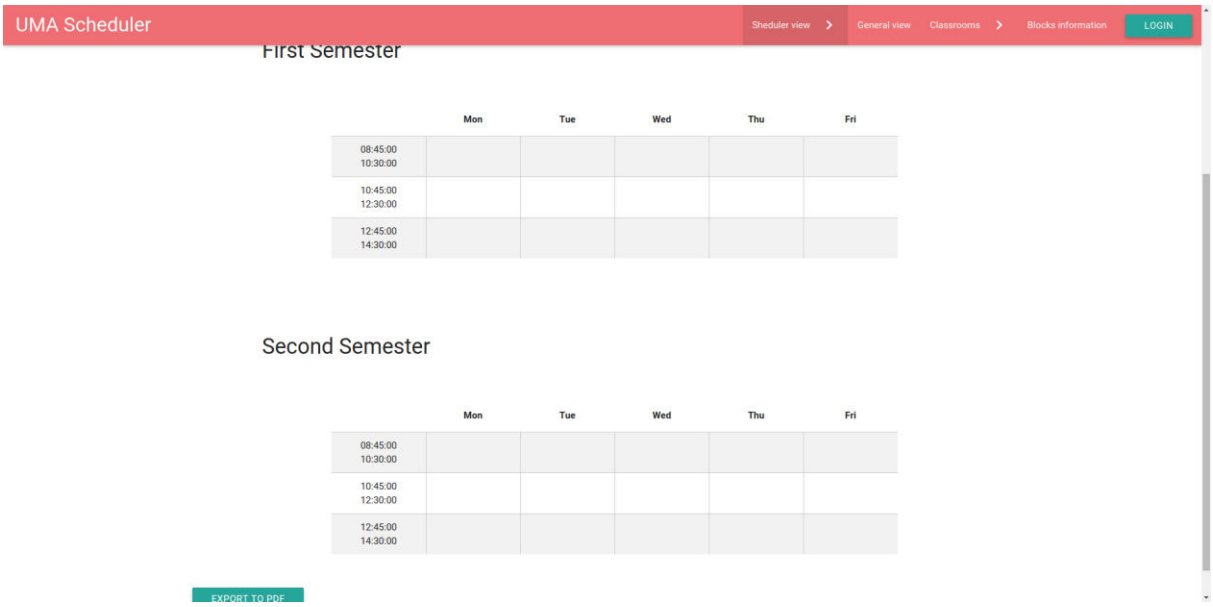

FIGURA 7 PÁGINA DE CONSULTA DE HORARIOS

También encontramos otra vista denominada "Visión General" (figura 8), en la cual se puede visualizar las asignaturas que ocupan en todo momento cada una de las aulas y en cada una de las franjas horarias. Para hacer uso de esta página se dispone de filtros de semestre, el día de la semana y el turno (mañana o tarde).

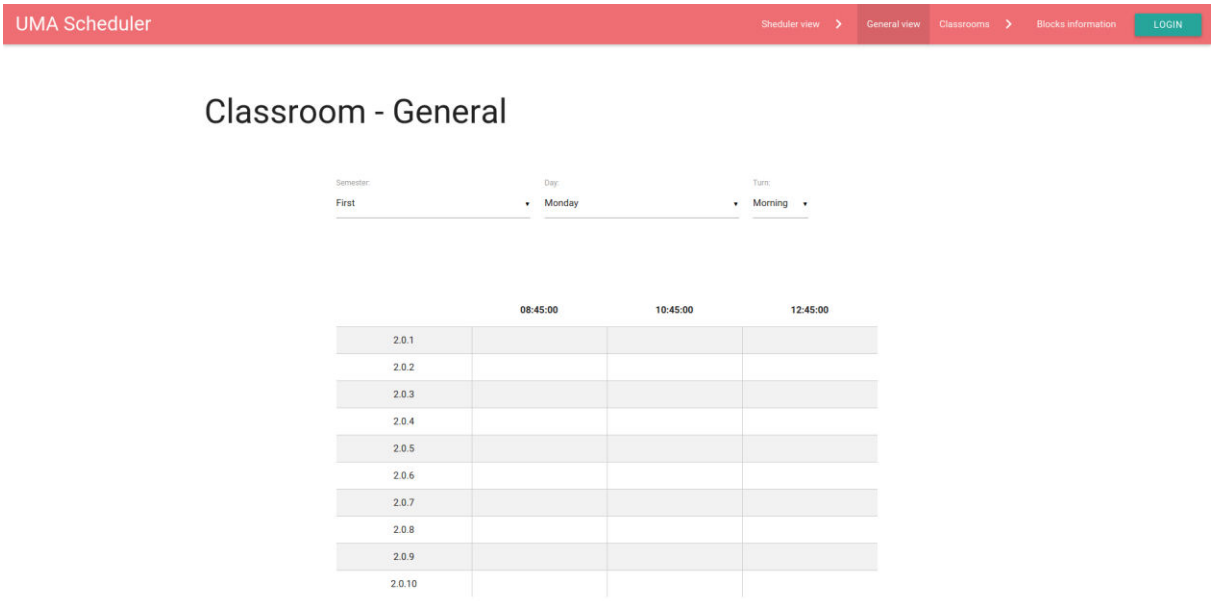

FIGURA 8 PÁGINA DE VISIÓN GENERAL

Por otro lado encontraremos también una vista en la aparecerá la información de los bloques de optativas (figura 9), de forma que se dispondrá el conjunto de asignaturas optativas que conforman cada bloque además de información correspondiente al bloque en sí mismo.

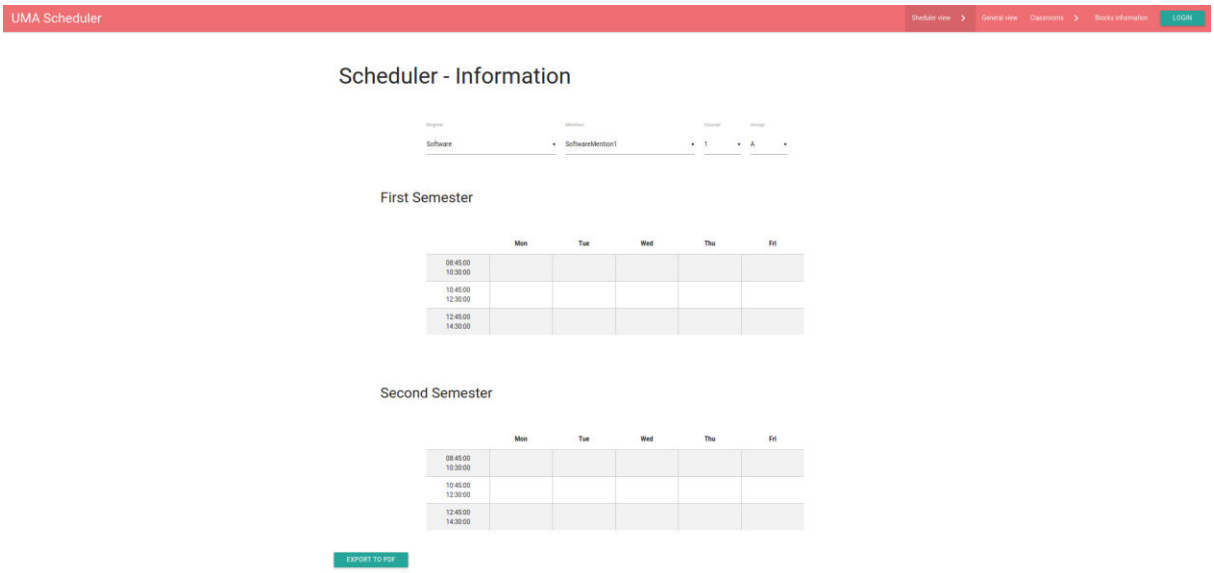

FIGURA 9 PÁGINA DE INFORMACIÓN DE BLOQUES DE OPTATIVAS

Para concluir, el usuario con el rol necesario tendrá acceso a una vista de administración, la cual permitirá crear todos los datos necesarios para dotar de la aplicación del estado básico a partir del cual darle un uso correcto.

Para ello, a la hora de generar los diferentes datos, se encargará de evitar que existan inconsistencias en ellos, ya que existen una variedad de restricciones a tener en cuenta a la hora de crear cierto tipo de objetos.

Organización y desarrollo del proyecto

Como introducción a este punto cabe mencionar que este proyecto ha sido realizado por tres personas, divididos en dos equipos por los motivos expuestos más adelante en este punto.

Al comienzo del proyecto tuvo lugar una primera fase de análisis de requisitos, en la cual discutimos las posibles dependencias tanto entre ellos como a la hora de elegir que tecnologías usaríamos.

Después de discutir entre nosotros las posibles incoherencias o dudas que podíamos tener acerca de la especificación de la herramienta, elaboramos un documento con las dudas y nos reunimos en más de una ocasión con nuestro tutor, el cual nos hizo de nexo de unión con el PO (Product Owner).

Una vez resueltas las dudas con respecto a la aplicación y formalizados los requisitos, pasamos a una segunda fase de diseño general, donde definimos un primer modelo de datos acorde a las especificaciones dadas, el cual sufrió modificaciones a lo largo de las diferentes iteraciones del proceso ágil de desarrollo llegando a ser otro totalmente distinto (ambos diagramas se encuentran más adelante en este mismo documento en la sección de diseño).

En esta fase llevamos a cabo también una toma de decisiones en dos puntos importantes: distribución general de las funcionalidades de la aplicación entre los tres miembros del equipo y metodologías de desarrollo a usar.

Debido a que un compañero no residiría a lo largo del curso en Málaga, decidimos dividirnos en dos equipos, repartiendo las funcionalidades entre los dos equipos en proporciones de uno y dos tercios y dejando las funcionalidades más cohesionadas de mano del equipo que residiría en Málaga.

Una vez realizada la división, se consensuó hacer uso de SCRUM como principal metodología de desarrollo, llevando a cabo sprints de dos semanas de duración y realizando reuniones semanales vía Skype entre los dos equipos. Además, el equipo compuesto por dos personas decidió hacer uso de XP (eXtreme Programming) en las US (user stories) que comprendían las funcionalidades más cohesionadas.

También realizamos una serie de diagramas comunes y necesarios para el inicio de la implementación del proyecto y algunos mockups para ilustrar una idea inicial de las vistas que debía proporcionar la aplicación.

A partir de ahí el proyecto se desarrolló de forma estable intercalando tareas de distintos módulos en los diferentes sprints, haciendo uso de Asana para la gestión de las diferentes tareas distribuidas en los diferentes sprints. Esta herramienta nos

proporciona una gráfica (figura 10) en la cual los sprints están definidos por dos líneas. La azul representan las tareas pendientes y la verde las tareas desarrolladas.

El sprint ideal contempla que a las dos semanas del inicio del sprint las líneas se junten en un mismo punto, cosa que, como se aprecia en la gráfica, no ocurre, ya que las tareas correspondientes al equipo que no residía en Málaga fueron demasiado complejas para poder realizar una buena estimación, con lo que gran parte de estas tareas han sido arrastradas a lo largo de todo el desarrollo, creando el margen constante que se aprecia a lo largo de toda la gráfica menos en el primer sprint, ya que el equipo residente en Málaga empezó con un sprint de margen.

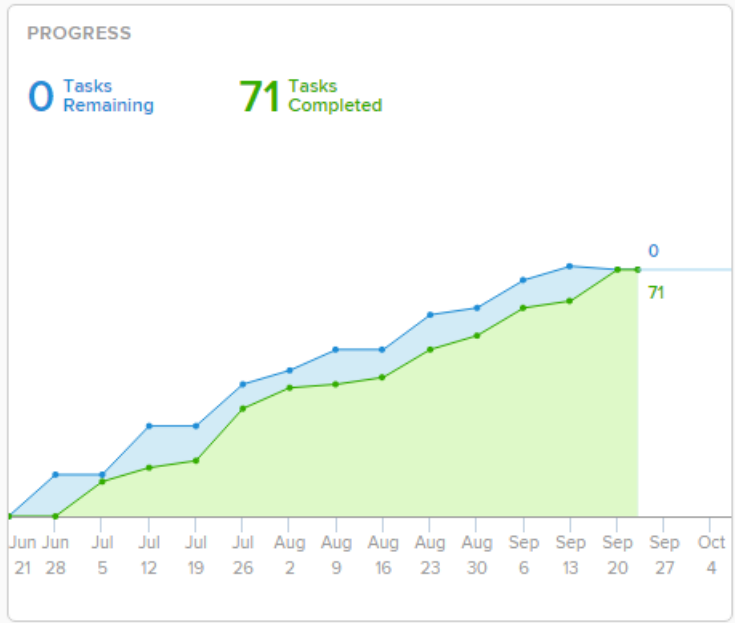

FIGURA 10 DIAGRAMA DE SPRINTS PROPORCIONADO POR ASANA

Especificación y análisis

Como primera fase de un desarrollo iterativo, como son las metodologías ágiles que tomamos como directrices, tomamos los requisitos funcionales a partir de las especificaciones del PO.

Dichos requisitos son:

- RF1: El modelo contiene varios grados.
- RF2: Los grados están relacionados con varias menciones.
- RF3: Las menciones están relacionadas con varios cursos.
- RF4: Los cursos están relacionados con uno o más grupos.
- RF5: El modelo contiene varias asignaturas por cada grupo, curso y grado.
- RF6: El modelo contiene asignaturas de tipo "ligada".
- RF7: Las asignaturas "ligadas" puede estar ligada a otras asignaturas del mismo curso en el mismo u otros grados y deben tener el mismo horario.
- RF8: El modelo contiene asignaturas de tipo "compartida".
- RF9: 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 RF1-6, RF8, RF10 y RF11

A partir de estos requisitos extrajimos los casos de uso correspondientes. Estos casos de uso son:

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 click 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 click 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 click 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 click 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 click 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 click 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 click 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 click 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. Desautenticación

Resumen: El usuario se desautentica 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 click que en el botón de "Logout"
- El sistema desautentica al usuario de la aplicación
- Redirige a la página de login.

Estos casos de uso están reflejados en los siguientes diagramas, ubicándolos en sus respectivos módulos previamente definidos en la figura 2:

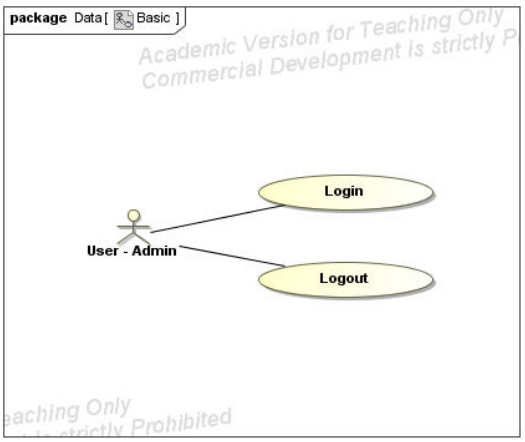

FIGURA 11 CASOS DE USO BÁSICOS

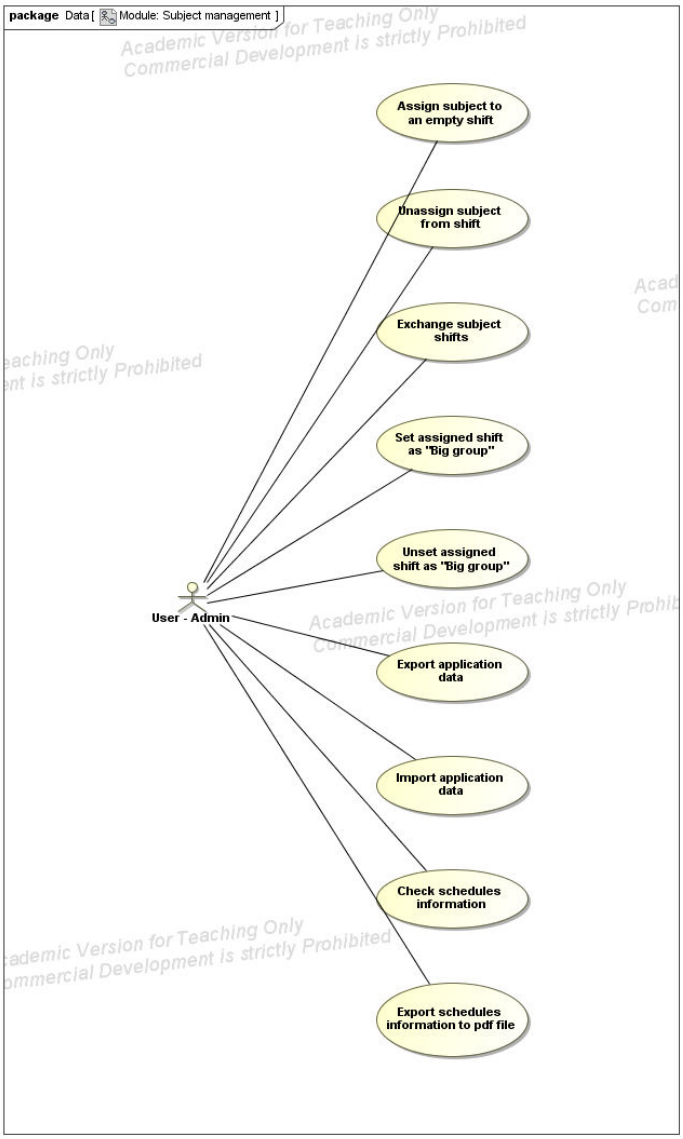

FIGURA 12 CASOS DE USO: MÓDULO DE GESTIÓN DE ASIGNATURAS

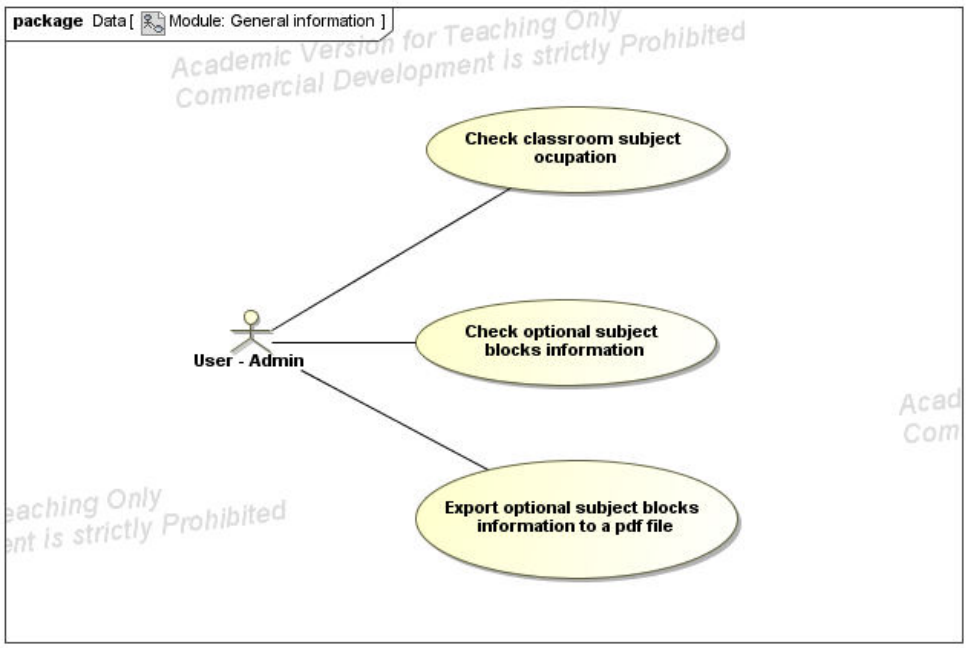

FIGURA 13 CASOS DE USO: MÓDULO DE VISIÓN GENERAL

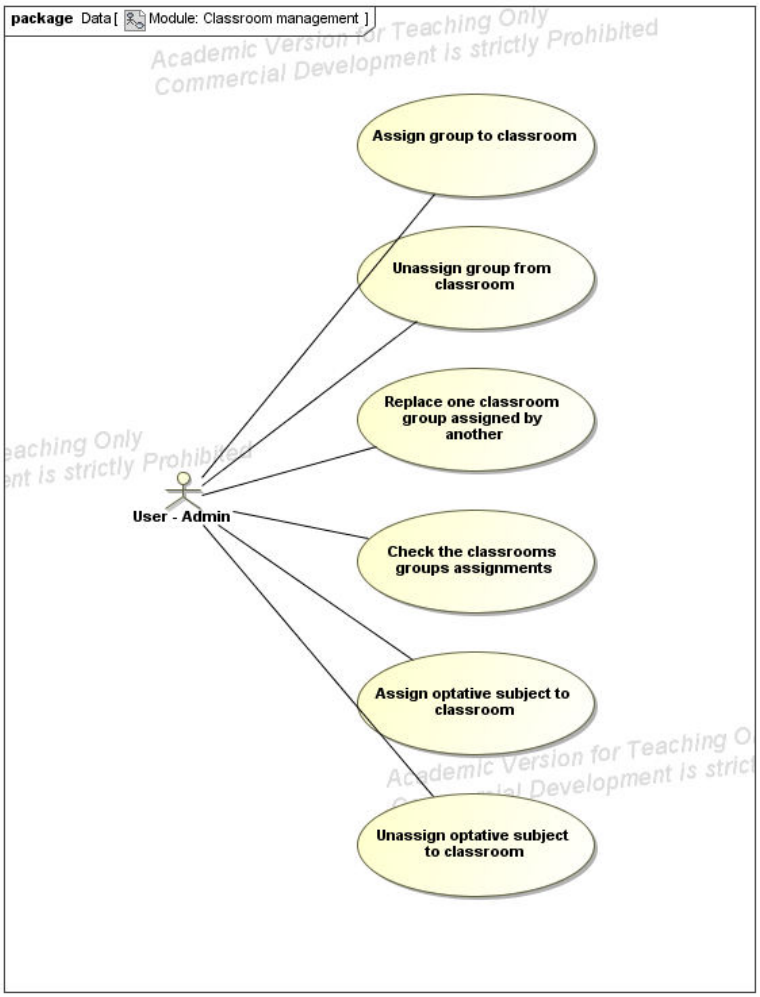

FIGURA 14 CASOS DE USO: MÓDULO DE GESTIÓN DE AULAS

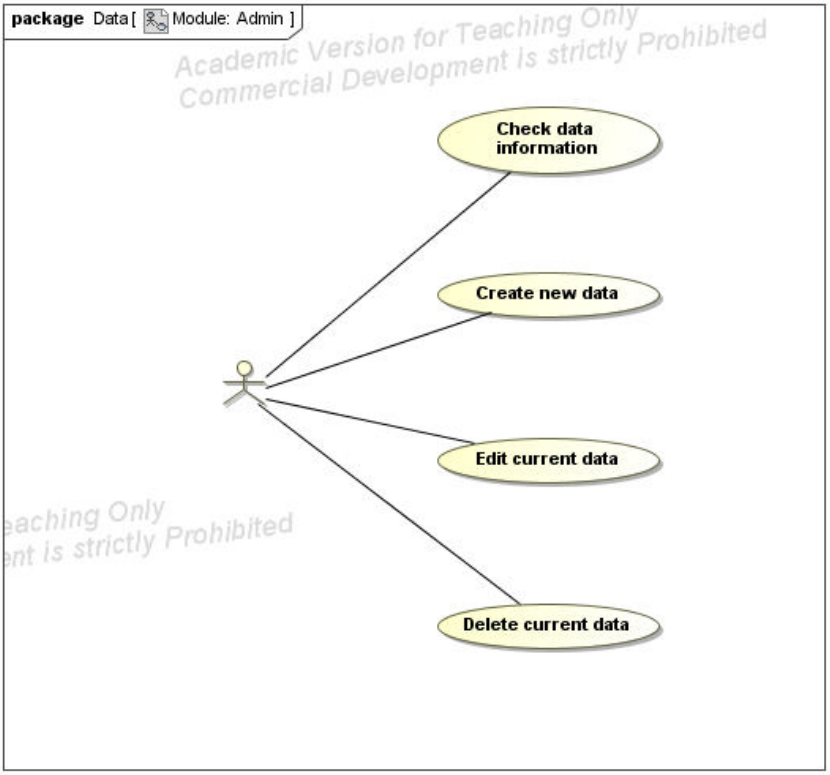

FIGURA 15 CASOS DE USO: MÓDULO DE ADMINISTRACIÓN

Diseño

Como segunda fase principal en el desarrollo realizamos un diseño inicial de las partes más fundamentales del sistema, como es en nuestro caso el diagrama de clases mediante el cual definir el modelo de la aplicación (figura 16).

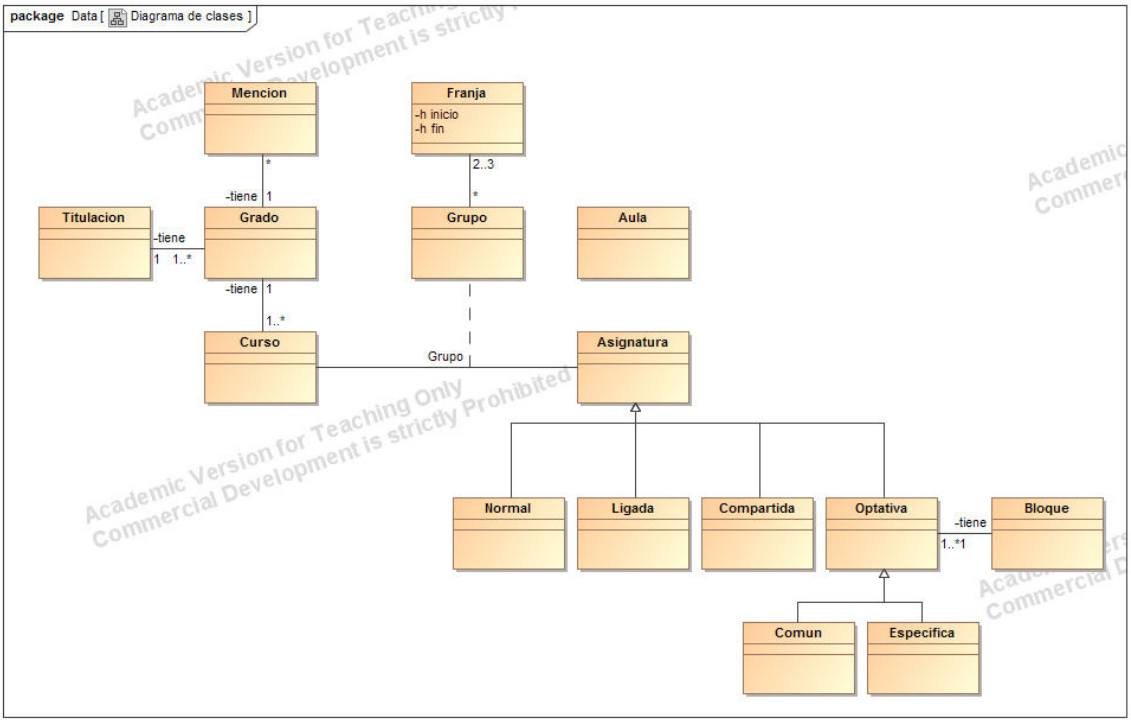

FIGURA 16 DIAGRAMA DE CLASES INICIAL

A lo largo de las diferentes iteraciones del proyecto, y al ser una metodología ágil como ya hemos dicho, tanto el modelo como otros aspectos de la aplicación sufrieron una serie de cambios, en la figura 3 se aprecia el estado final del diagrama de clases, el cual, en contraste con el inicial que es mucho más optimista, refleja los cambios necesarios introducidos a lo largo del desarrollo conforma a las nuevas implementaciones los requerían.

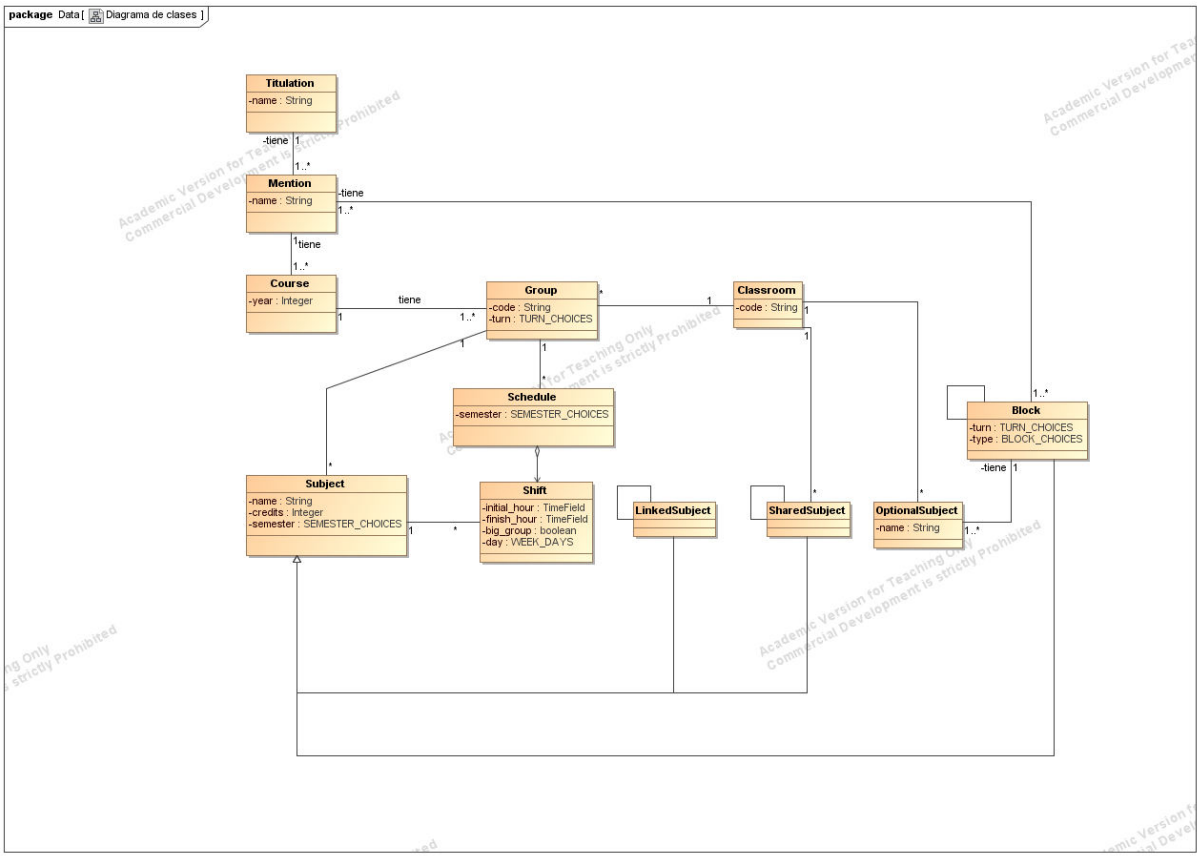

FIGURA 17 DIAGRAMA DE CLASES FINAL

Cabe mencionar también en este apartado el diseño de la arquitectura de carpetas del proyecto, la cual sigue las directrices de los colaboradores del proyecto Django en varios de sus libros de buen uso sobre Django (pe. "Two Scoops of Django: Best Practices for Django 1.8").

En la figura 4 distinguimos dos carpetas que nacen de la carpeta raíz del proyecto:

La primera es la carpeta principal del proyecto, encargada de almacenar ficheros generales y básicos del framework, como la configuración.

La segunda se corresponde con la aplicación que abarca este documento, y tiene una seria de subcarpetas que tienen diferentes propósitos.

La carpeta "api" contiene los recursos de la librería *Tastypie*, encargados de definir la API REST de la aplicación.

La carpeta "fixtures" es la encargada de almacenar las diferentes configuraciones guardadas por el usuario mediante el RF24 definido con anterioridad.

La carpeta "migrations" guarda los diferentes estados por los que atraviesa la base de datos a lo largo del desarrollo, con el fin de tener puntos de retorno a ellos.

La carpeta "static" contiene una carpeta con el mismo nombre que la de la aplicación, con el fin de identificar con mayor facilidad trazas de error en el caso de tener un proyecto Django con múltiples aplicaciones. Dentro de esta segunda carpeta encontramos las carpetas "css", "js" y "vendor", encargadas de almacenar el código CSS y JS de la aplicación respectivamente, siendo la labor de la tercera almacenar todo el código externo del que depende la aplicación (CSS, JS y fuentes).

La carpeta "templates" es la encargada de almacenar los archivos HTML de la aplicación. De nuevo encontramos una carpeta con el nombre de la propia aplicación por el mismo motivo descrito en el párrafo anterior.

Dentro de la carpeta de la aplicación encontramos también una serie de archivos:

El archivo "admin.py" contiene toda la configuración necesaria para el correcto funcionamiento del módulo "admin" de Django, así como su personalización.

El archivo "middelware.py" contiene los *middlewares* desarrollados para la aplicación.

El archivo "models.py" contiene la definición de todas las clases que comprende el modelo de la aplicación, así como los métodos que definen la lógica en cada una de ellas.

El archivo "tests.py" es el encargado de recoger los distintos tests unitarios definidos para la aplicación.

El archivo "urls.py" define las urls mediantes las cuales llamar a las diferentes vistas de Django definidas en la aplicación.

El archivo "views.py" contiene las vistas de Django definidas para la aplicación.

Por último, encontramos como hijos de la carpeta raíz del proyecto una serie de archivos:

El archivo ".gitignore" es el encargado de decirle al repositorio de Git instanciado para el proyecto que archivos no debe incluir en los diferentes *commits* realizados por el desarrollador.

El archivo "db.squlite3" es la base de datos del sistema.

El archivo "manage.py" es propio del framework de Django. Contiene multitud de utilidades a las cuales podemos acceder mediante comandos, como por ejemplo inicializar una nueva aplicación en el proyecto o crear un súper usuario.

El archivo "requirements.txt" contiene todas las librerías necesarias para el correcto funcionamiento de la aplicación.

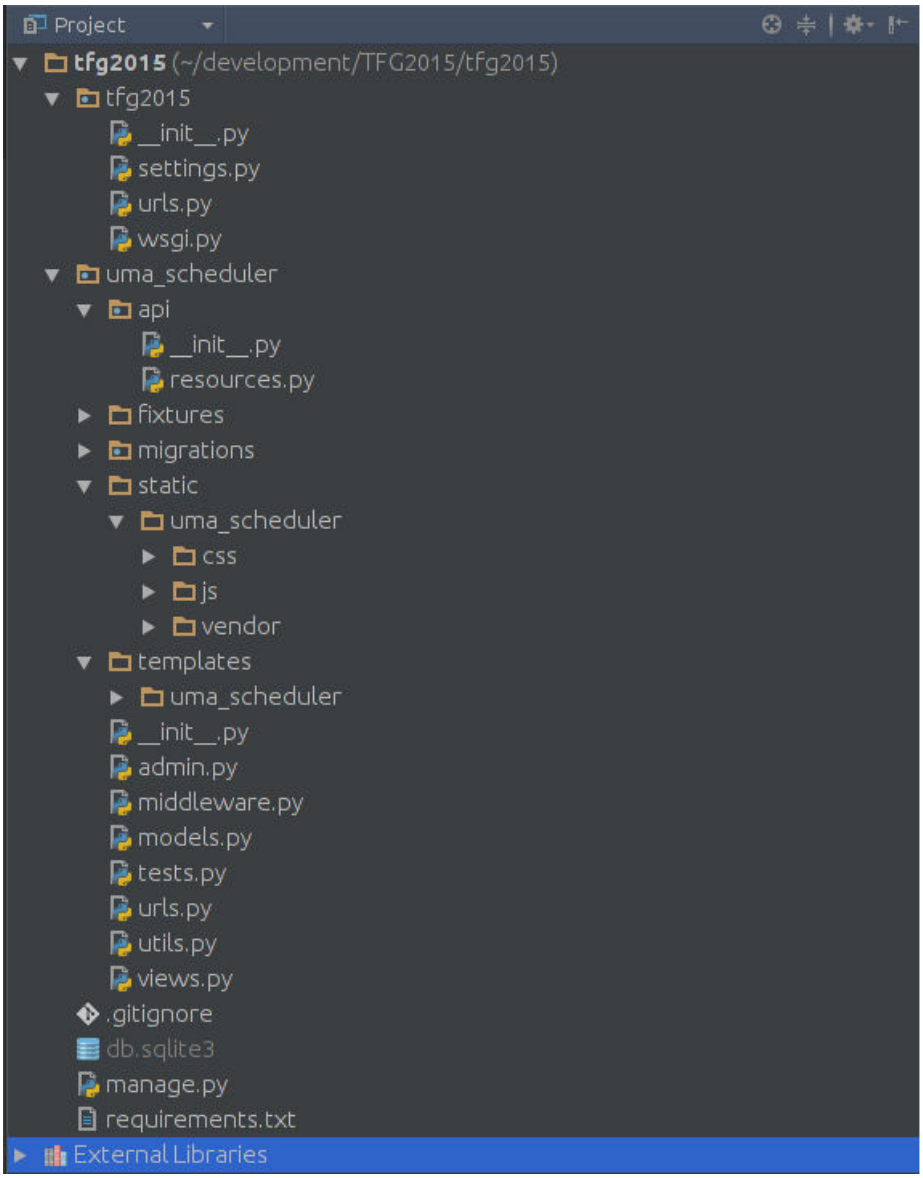

FIGURA 18 JERARQUÍA DE CARPETAS DEL PROYECTO

Por último contenido de este apartado está la arquitectura que sigue la aplicación web. Es una aplicación web típica cliente-servidor, con la peculiaridad de las aplicaciones web modernas de que la comunicación entre ambos es mediante una API REST definida en el servidor.

En la parte del cliente, y en acorde a las tecnologías definidas previamente en el documento, predominan como frameworks principales Backbone, para la gestión de modelos y colecciones, Marionette, para la gestión de las vistas (controladores) correspondientes a las regiones definidas en los archivos HTML, y JQuery, como framework base de los dos anteriores y del cual hemos hecho uso de multitud de funcionalidades.

Como segundo componente, siendo este un canal de comunicación entre el cliente y el servidor, encontramos la API, definida con Tastypie, una librería Python de Django como hemos mencionado anteriormente. Mediante ella el cliente puede acceder a la información de la base de datos ubicada en el servidor mediante llamadas Ajax a las urls definidas en los propios recursos e incluidas en los ficheros de configuración de urls de Django.

Por último, en la parte del servidor predomina el framework Django, mediante el cual se han definido tanto los modelos, con toda la lógica que contienen para realizar las diferentes funciones requeridas por los requisitos, como las vistas propias de Django, que representan el flujo principal de las funcionalidades que representan, delegando la lógica fuerte a los modelos y devolviendo respuestas HTML u JSON dependiendo del tipo de llamada y vista.

Todo esto queda reflejado en la siguiente figura.

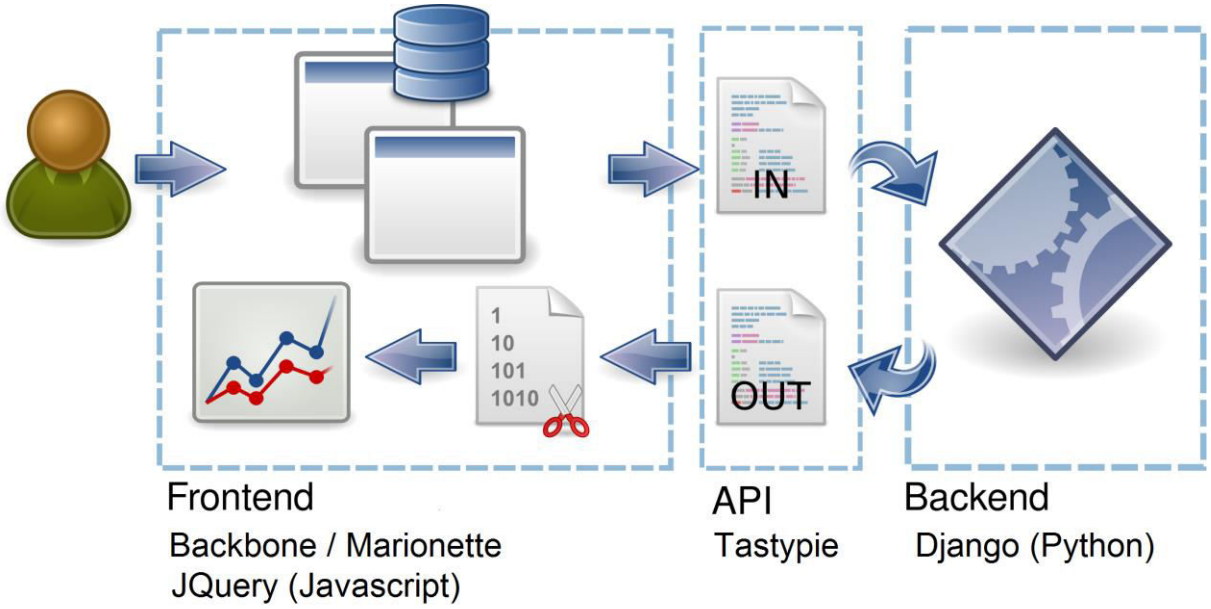

FIGURA 19 ARQUITECTURA DE COMPONENTES DE LA APLICACIÓN

Implementación y pruebas

Al comienzo del desarrollo del proyecto realizamos una serie de tareas comunes y básicas entre los tres componentes del grupo. Estas tareas fueron:

- Iniciar el repositorio Git en BitBucket.
- Crear proyecto básico Django, así como la aplicación necesaria para desarrollar el proyecto.
- Evolución continua de los requisitos y diseños en las diferentes iteraciones en las que se ha comprendido el desarrollo del proyecto.
- Generación de una API Rest mediante la definición de recursos proporcionados por la librería Tastypie, para la correcta obtención de datos desde la parte del cliente de la aplicación (frontend).

Más adelante me hice cargo de una tarea de refactorización del proyecto, tanto en cuanto a reorganización del código como de la jerarquía de carpetas del proyecto para un mejor uso de la filosofía de Django. Por otro lado, desarrolle un middleware para restringir el acceso a la aplicación en caso de no tener iniciada una sesión en el servidor.

Por último realicé la configuración básica del archivo "settings.py" para el correcto funcionamiento de la aplicación. Estos settings son:

- INSTALLED_APPS (añadir las aplicaciones externas a la tupla)
- MIDDLEWARE_CLASSES (añadir middlewares desarrollados)
- LOGIN_URL
- LOGIN_REDIRECT_URL
- TEMPLATE_DIRS
- FIXTURE_DIRS
- STATICFILES_DIRS.

El funcionamiento de estos settings puede consultarse en la página de documentación de Django, adjuntada en los enlaces bibliográficos.

Como contenido común entre las páginas, he tomado parte en el desarrollo (de forma parcial en algunos casos y total en otros) de los siguientes elementos:

- Archivo HTML base:

Definí la estructura e incluí la importación de dependencias necesarias (añadiendo todos los archivos y fuentes necesarios a nivel local para no tener dependencias externas). También definí los bloques mediante los cuales el resto de vistas que heredan de este archivo añaden su contenido.

Estos bloques son:

- extra_css: donde se añade la importación de los archivos ".css" propios de cada página

- navbar: incluye el HTML que contiene el navbar. Este bloque es sobrescrito en el login, ya que no tiene que mostrar este navbar.

- page_title: donde se incluye el título de cada página.

- page_name: donde se incluye el nombre de cada página.

- view: aquí se incluye el cuerpo de cada HTML que hereda de este archivo.

- js: donde se añade la importación de los archivos ".js" propios de cada página

- Navbar:

En el HTML de este componente, definí los enlaces de los siguientes apartados: gestor de horarios, información de horarios, vista general de aulas y gestor de aulas y grupos.

También realicé las modificaciones CSS y JS para el correcto renderizado del componente, que comprendían la correcta visualización de los iconos incluidos en la aplicación y de la sección que está activa en cada una de las páginas.

- Selectores:

A la hora de renderizar los selectores de forma dinámica descubrimos un fallo de la librería Materialize, que consistía en una mala renderización del contenido de las opciones en cada una de las opciones del selector. Mi aportación fue precisamente solucionar este bug, controlando en las vistas Marionette que encapsulaban esos selectores que al renderizar de nuevo el contenido de los selectores en función de los otros el HTML resultante contuviera el valor correcto.

- Loading:

He integrado el componente "loading" del framework Materialize en las vistas: manager de horarios, vista informativa de horarios, vista informativa general de aulas y vista de gestión de aulas y grupos. Para ello he integrado el código HTML necesario en cada una de las vistas que hacen uso de él, así como el control de que se muestre sólo mientras el contenido está siendo renderizado mediante código JS.

- Drag&Drop:

Me encargué de la integración entre el módulo Drag&Drop de JQueryUI y las vistas Marionette encargadas de renderizar las asignaturas y los grupos en sus respectivas vistas de administración.

Al ser una memoria personal pasaré a explicar el contenido desarrollado por mí a partir de ahora, delegando el resto de funcionalidades en las memorias de mis compañeros, ya que cada uno tiene un mejor conocimiento del funcionamiento de los distintos módulos y funcionalidades en las que ha trabajado.

Dichas partes desarrolladas total o parcialmente por mi son:

- Página principal

Me hice cargo de la creación del HTML y la definición de las clases CSS necesarias para el correcto renderizado de su contenido.

- Manager de los horarios:

Tablas de horarios y asignaturas:

Elaboración del HTML principal y de las plantillas HTML encapsuladas en scripts necesarias para poder renderizar correctamente las tablas de asignaturas y horarios mediante las vistas de Marionette definidas para esta página.

Por otra parte definí los modelos Backbone para encapsular las asignaturas y las franjas horarias mediante las librerías Tastypie y Backbone-Tastypie, y las vistas Marionette para renderizar dinámicamente el HTML en función de las opciones seleccionadas en los selectores de filtro de búsqueda las cuales comprendían un LayoutView para el control general de la página, dos CompositeView (uno para las asignaturas y otro para las franjas horarias) que a su vez tenían asociados como ChildView dos ItemView, que se hacían cargo de la correcta renderización de los modelos Backbone en la vista.

Se ha hecho uso también de la librería Underscore para el acceso a los atributos de los modelos Backbone durante el renderizado dinámico de la página.

Por último, definí las clases CSS que permiten que el contenido se muestre de la forma que lo hace actualmente.

Marcar asignatura como grupo grande:

Para cubrir esta funcionalidad, y en la parte del Backend, fue necesario definir una vista de Django para el control y asignación de las asignaturas como grupo grande en una franja horaria determinada, además de la inclusión de código Python en otras vistas que requerían controlar estados dependientes de esta funcionalidad.

Por el lado del Frontend en cambio requirió añadir código en el ItemView encargado de las franjas horarias, mediante el cual se hace una llamada Ajax cuyo callback recoge una respuesta JSON. En caso de éxito en la respuesta, se ejecuta una función encargada de marcar y desmarcar, si fuese necesario) las franjas horarias afectadas por la asignación de grupo grande.

Export/Import de datos:

El sistema de exportación e importación consta de dos vistas Django con sus respectivas urls. Estas vistas se encargan de generar o cargar, mediante los comandos proporcionados por el archivo "manage.py" de Django, los fixtures que encapsulen todas y cada una de las instancias de la base de datos.

Por el lado del Frontend desarrollé código HTML y JS necesario para el control de las llamadas de forma dinámica, incluyendo los elementos de la UI y los eventos necesarios en el LayoutView principal de la página.

Ya que este sistema aprovecha el sistema de fixtures proporcionado por Django, hace posible que a lo largo del uso de la aplicación sea posible la gestión de distintos centros diferentes de una forma rápida y dinámica.

Desasignar asignatura de una franja:

En la parte del Backend fue necesario el desarrollo de una vista de Django es la encargada de realizar las comprobaciones necesarias para realizar la desasignación, y desasignar la asignatura a la franja en caso de pasarlas.

Por la parte del Frontend, se modificó la plantilla de representación de las asignaturas para mostrar la imagen que hace de elemento de interfaz para que el usuario realice la desasignación.

También se añadió al ItemView encargado de encapsular una franja ocupada los elementos de la UI y los manejadores necesarios para recoger los eventos del usuario y realizar la llamada Ajax a la URL de la vista preparada para realizar la asignación.

En caso de recibir éxito en la respuesta, se ejecuta el callback encargado de hacer desaparecer la asignatura desasignada en el HTML.

Asignación e intercambio de asignaturas normales, ligadas, compartidas y optativas:

En esta funcionalidad se ha incluido el módulo de Drag&Drop de JQueryUI, para poder asignar las asignaturas a las franjas de una forma totalmente fácil e intuitiva.

En la parte del Frontend se ha llevado a cabo dicha integración, ubicando en las vistas ItemView correspondientes a las asignaturas y a las franjas horarias el código que proporcionaba la posibilidad de usar este sistema a los elementos HTML que lo necesitaban, así como la definición de los métodos llamados por la librería al droppear un elemento en otro válido.

Mediante estos métodos se llama a la vista de Django encargada de realizar todas las comprobaciones necesarias para realizar esta asignación, en la cual reside la mayor parte del peso de esta funcionalidad.

Estas comprobaciones exigían que asignar una asignatura a una franja horaria comprobase que, en función del número de créditos de la asignatura, no se superara un número determinado de ocurrencias de la asignatura en el horario, así como que no aparecieran dos asignaturas iguales en un mismo día de la semana. La complicación es combinar estas comprobaciones con el hecho de que varios de los tipos de asignaturas requeridos en los requisitos deben tener el mismo horario, por lo que a la hora de asignar una asignatura, y en el caso de estar relacionada con otras, estas otras deberían pasar también las restricciones básicas. En el caso del intercambio de asignaturas, todo esto se puede replicar también en la asignatura contraria la cual estamos moviendo, y a su vez al mover estas pueden intercambiarse con otras que también tengan algún tipo de relación.

Como este proceso puede resultar un poco saturante a la hora de tener en cuenta todos los posibles estados que puedes encontrarte desarrollamos algunos diagramas de secuencia, desgranando la funcionalidad en objetivos más pequeños y modulando el código de forma que fuera más inteligible y mantenible. A continuación comento estos diagramas, adjuntando imágenes de los mismos.

Como detonante de todo el proceso, el usuario asigna visualmente una asignatura a una franja horaria, ya sea desde fuera del horario o desde dentro del mismo. Esta asignación puede ser a una franja horaria vacía o a una ocupada por otra asignatura.

Al hacer esto una llamada Ajax ejecuta la vista de Django encargada de recoger los modelos de la asignatura asignada, así como el correspondiente shift. En caso de mover la asignatura desde dentro del horario, o de realizar un intercambio, esta vista recoge también dichos modelos.

Una vez hecho esto solo es necesario llamar al método "assign_subject_to_shift", para delegar toda la lógica del proceso en el modelo y dejar la vista como mero comunicador entre la interacción con el usuario y el procesamiento de datos, tal y como está recomendado por dos de los coladores oficiales del proyecto del framework como hemos mencionado anteriormente en este mismo documento

En este método se comprueba que la asignación entre la asignatura y la franja horaria cumple todas las restricciones y, en caso de haber un intercambio, que la otra asignatura también las cumple.

El desarrollo descrito anteriormente está representado en la figura 20.

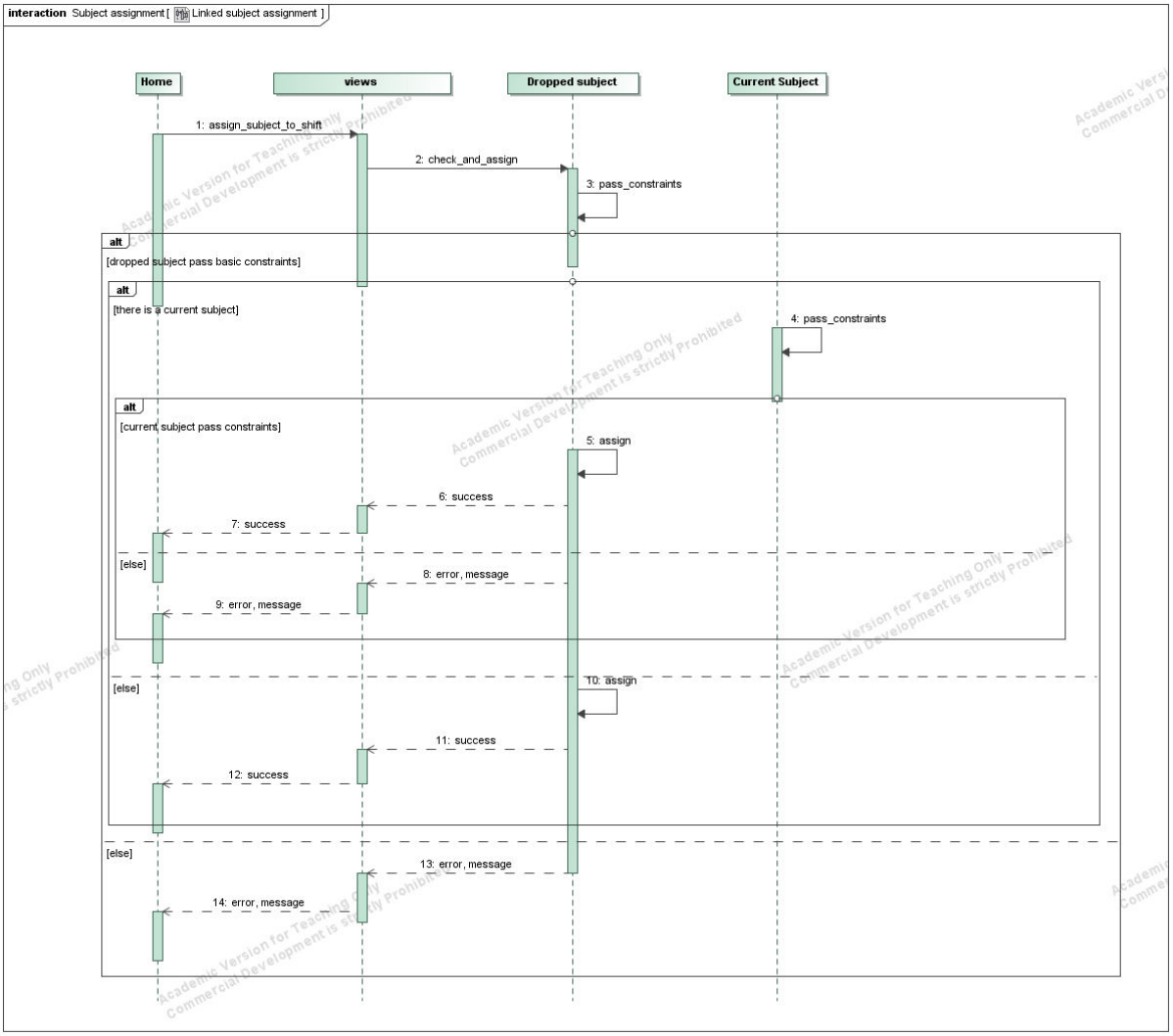

FIGURA 20 DIAGRAMA DE SECUENCIA: ASIGNACION DE ASIGNATURA A FRANJA

El método "pass_constraints" debe tener en cuenta que las asignaturas pasen las restricciones básicas requeridas en los requisitos RF15, RF16 y RF17, recogidos con anterioridad en este mismo documento. Además se debe comprobar, como ya hemos dicho anteriormente, que las asignaturas tengan relaciones de horarios con otras y, por lo tanto, modificar también las localizaciones de estas asignaturas relacionadas dentro de su propio horario.

Este último caso, el cual como hemos visto puede llevar a cierta recurrencia en caso de que asignaturas de distintos horarios se intercambien con otras asignaturas con más relaciones, está delegado a otro método que veremos más adelante.

Esta segunda fase está reflejada en el diagrama correspondiente a la figura 21.

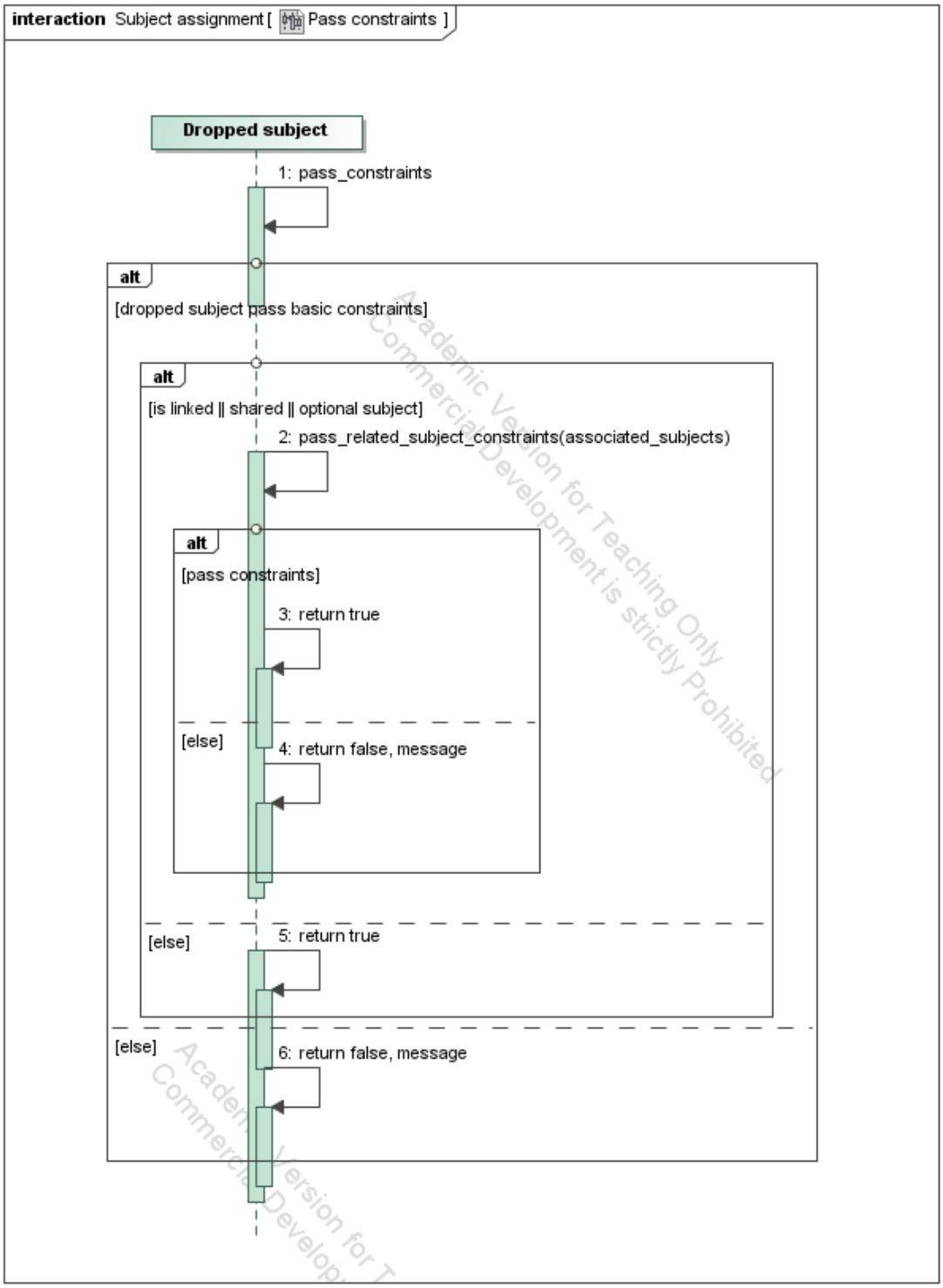

FIGURA 21 DIAGRAMA DE SECUENCIA: PASO DE RESTRICCIONES

Por último, en el caso de haber relaciones debido a que la asignatura sea del tipo "ligada", "compartida" u "optativa", iteramos sobre las asignaturas con las que está relacionada y recogemos su horario y la asignatura con la que se intercambia en el caso de que la hubiera. Por otro lado, la asignatura inicial ha podido ser intercambiada desde dentro del horario o como producto de una nueva asignación, por lo que hacemos dicha comprobación.

En caso de que exista la asignatura con la que se intercambia la asignatura relacionada (objeto iterador del bucle), que la asignatura vaya a una franja horaria (debido a que la asignatura inicial estaba en el horario) y que esta comprobación no se haya realizado antes (ya que en el diagrama de flujo de las comprobaciones existen caminos circulares que debemos eliminar), se comprueba que esa asignación también es posible llamando al método explicado previamente "pass_constraints".

En este método volverán a comprobarse las restricciones básicas y si la asignatura tenía más relaciones con otras asignaturas. Es aquí donde, debido a la recursividad de las llamadas, pueden darse los caminos circulares que hemos eliminado como acabamos de ver.

Todas estas llamadas agrupan sus respuestas en un solo valor "booleano", el cual representa si todas las asignaturas involucradas finalmente en la asignación cumplen las restricciones básicas, y, por lo tanto, si la asignación finalmente es válida.

Todo este proceso está reflejado en el diagrama etiquetado como figura 22.

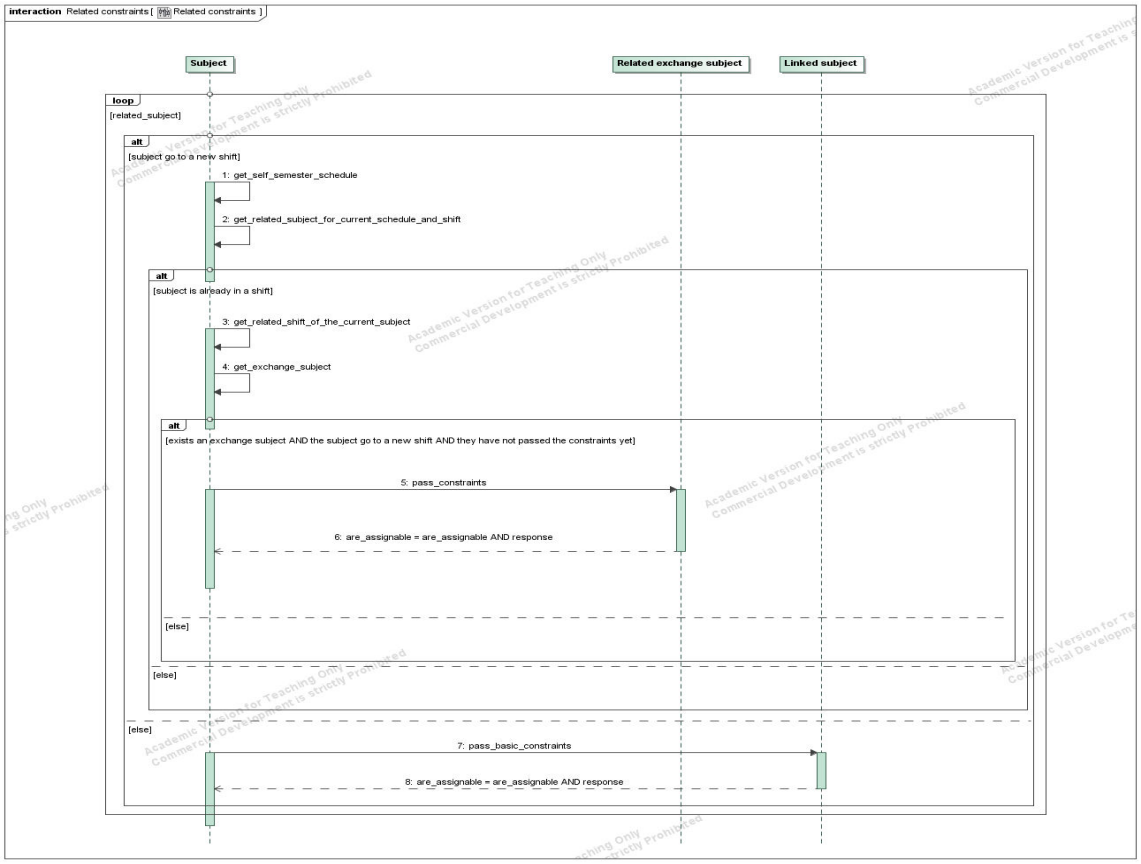

FIGURA 22 DIAGRAMA DE SECUENCIA: ASIGNATURAS RELACIONADAS - PASO RECURSIVO DE RESTRICCIONES

Página de información de los horarios:

En esta vista, correspondiente al requisito RF25, me hice cargo de la funcionalidad de exportación.

Para esto hice uso de la librería xhtm2pdf mencionada en la introducción, mediante la cual podía formatear en pdf un HTML renderizado previamente con la información de todos los horarios de los grupos registrados en la aplicación.

Página de gestión de grupos y aulas:

En esta página, correspondiente al requisito RF28, me dediqué a elaborar el código HTML, así como su representación en pantalla mediante código CSS.

También trabajé en el desarrollo del código JS que se encargaba del correcto renderizado de las plantillas HTML desarrolladas, así como la gestión de eventos y llamadas Ajax para la realización de las acciones.

Este código JS se encargaba también de gestionar las respuestas JSON del backend, de dotar de los *toast* a las clases con asignaciones hechas por grupos y de corregir errores del frontend Materialize.

En cuanto a la fase de mantenimiento y pruebas, debido al tiempo, la cohesión entre las distintas partes y la distancia entre los equipos; no supimos definir una buena dinámica de trabajo para esta fase, por lo que definimos tests manuales en función de los distintos requisitos y casos de uso definidos para comprobar el correcto funcionamiento de todos los módulos de la aplicación en la integración continua del desarrollo.

Conclusiones

Después de haber desarrollado este proyecto quisiera dividir las conclusiones en dos grupos, que abarcan por un lado mi perfil como desarrollador y, por otro, mi perfil de estudiante.

Lo primero que se me viene a la mente finalizado el desarrollo es lo enriquecedor que es conocer y aprovechar distintas tecnologías, así como entornos de trabajo distintos.

Usar Python como lenguaje de desarrollo y todas las tecnologías que envuelven el uso de Ubuntu, Django, Materialize y Marionette en nuestro proyecto me ha dado una nueva visión del desarrollo web, que no distinta de la anterior, sino más complementada, asentada y profunda. También cambia la perspectiva desde la cual partes en las fases de análisis, especificación y diseño del proyecto, ya que la propia experiencia adquirida en todo desarrollo te proporciona un *feedback*, mediante el cual puedes anticiparte mejor a situaciones que te encuentras en posteriores fases a lo largo del desarrollo.

En cuanto al proyecto en sí tengo varias conclusiones, empezando por la siguiente:

Hay un número considerable de mejoras aplicables a las funcionalidades de la aplicación, las cuales se escapaban a la definición del trabajo que debíamos realizar y, por lo tanto, al tiempo de desarrollo disponible para el mismo; aunque no suponen un problema para el correcto uso de la aplicación. Por otro lado, y después de las experiencias de desarrollo en los proyectos en los que he tomado parte desde que inicié el proceso de este trabajo, la escalabilidad del proyecto deja que desear. A finales del desarrollo llevé a cabo una refactorización que facilitara la posible mantención futura del proyecto, pero soy consciente de que varias de las funcionalidades deberían estar más desgranadas en distintas aplicaciones Django, cosa que no sucede.

También me llevo una impresión mejorable con el hecho de haber desarrollado este proyecto con un miembro a distancia y habiendo desempleado un papel distintivo en cuanto a la coordinación entre nosotros en varios aspectos, me ha hecho comprender la importancia de una buena configuración de equipo dentro de la metodología de desarrollo ágil SCRUM, refiriéndome al papel de un team leader que se encargue también de tareas de gestión dentro de cada *sprint*, algo que no hemos definido bien desde el principio.

Como aportación positiva de este proyecto me encuentro sobre todo con la sensación de capacidad de llevar a cabo un proyecto de forma independiente con un equipo y unos recursos básicos, algo que no se puede decir a lo largo de otras etapas de la carrera debido a la diferencia clara entre desarrollar tareas propuestas por la docencia con una guía bastante específica del profesor a realizar una labor de

principio a fin de desarrollo de un proyecto, con la parte de investigación propia de un trabajo como este.

Por último, hemos aprendido también muchísimo sobre frontend desarrollando esta aplicación, una parte que quizá no da tiempo a tocar de cerca a lo largo de la carrera debido a su considerable extensión. Considero que de cara a desarrollar aplicaciones web a día de hoy es una parte de suma importancia, ya que cierto número de aplicaciones que hay hoy en día carecen incluso de un backend propio, delegando en APIs la toma de datos desde el modelo.

Posibles ampliaciones

Siguiendo lo mencionado en el punto anterior, la aplicación tal y como se encuentra ahora mismo tiene una gran variedad de posibilidades de ampliación:

En primer lugar, es posible mejorar los módulos ya existentes. Debido a las relaciones que existen entre los distintos tipos de acciones y las dependencias que esto crea entre los horarios de estas asignaturas en el primer módulo, cuando se realiza una asignación o un intercambio a veces es necesario hacer comprobaciones a muchos niveles de profundidad para evitar inconsistencias en las relaciones.

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.

Es aquí donde entra una posible ampliación: modificar esta respuesta para que devolviera códigos de error descriptivos que permitieran localizar dónde se está produciendo inconsistencia, con lo que podríamos tomar una decisión más precisa sobre cómo solucionar este fallo de consistencia en nuestra configuración.

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 entre grupos y espacios del centro.

Por otro lado y en una línea totalmente diferente, es posible mejorar el funcionamiento general de la aplicación. Poniendo como ejemplo un caso práctico, aunque una asignatura tenga asignada varias franjas horarias, es posible que alguno de ellas no se use nunca o 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 contemplar esos casos para dar indicaciones al usuario, y así poder realizar una configuración mucho más eficiente y rica, usando todos los recursos proporcionados por el centro en cuestión.

También se podrían añadir módulos totalmente nuevos a la aplicación, como por ejemplo un módulo para gestionar el uso de los laboratorios para clases prácticas. De esta forma, podrían asignarse varios de las franjas horarias de una asignatura como franja de laboratorio, y asociarle un aula de distinto tipo a la que está asociada al grupo en cuestión.

Conectando con esto y por último, también existiría la posibilidad de reservar laboratorios para ocasiones especiales, asegurándose de la disponibilidad de este en el momento en el que se quiera utilizar.

Referencias

https://www.python.org/

https://www.djangoproject.com/

http://underscorejs.org/

http://backbonejs.org/

http://marionettejs.com/

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

http://blog.mathandpencil.com/using-django-tastypie-to-create-RESTful-APIs/

http://materializecss.com/

https://jquery.com/

https://jqueryui.com/

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

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