



ugr | Universidad
de **Granada**

TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

iOrg2.0

Herramienta de apoyo a la docencia para el Departamento
de Organización de Empresas

Autor

Rogelio Gil García

Tutor

Dr. Pedro A. Castillo Valdivieso

Cotutora

Dña. María Magdalena Jiménez Barriosuevo



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 6 de septiembre de 2017

iOrg2.0

Rogelio Gil García

Yo, **Rogelio Gil García**, alumno de la titulación **Grado en Ingeniería Informática** de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado (*iOrg2.0*) en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Este documento está desarrollado con LaTex a partir de la plantilla facilitada por la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación, y la plantilla de LaTex de Jaime Torres Benavente, publicada bajo licencia **Creative Commons Attribution-ShareAlike 4.0** (<https://creativecommons.org/licenses/by-sa/4.0/>) que se encuentra en <https://github.com/torresj/indi-android-ui/tree/master/documents>

La licencia de este trabajo es **Creative Commons Attribution-ShareAlike 4.0**(<https://creativecommons.org/licenses/by-sa/4.0/>)
Fdo: Rogelio Gil García

Granada, a 6 de septiembre de 2017

Dr. Pedro A. Castillo Valdivieso, Profesor del Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada.

Dña. María Magdalena Jiménez Barrionuevo, Doctora del Departamento de Organización de Empresas de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado *iOrg2.0*, ha sido realizado bajo su supervisión por **Rogelio Gil García**, y autoriza la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 6 de septiembre de 2017.

Los tutores:

Dr. Pedro A. Castillo Valdivieso

Dña. María Magdalena Jiménez Barrionuevo

Agradecimientos

A mi familia por darme la oportunidad de llegar hasta aquí.

A mis amigos por acompañarme y darme ánimos desde hace tantos años.

A mis tutores Dr. Pedro A. Castillo Valdivieso y Dña. María Magdalena Jiménez Barrionuevo, por su apoyo y asesoramiento en el proyecto.

Índice general

1. Resumen	1
1.1. Breve resumen y palabras clave	1
1.2. Extended abstract and key words	2
2. Introducción	3
2.1. Motivación y análisis del problema	3
2.1.1. Aplicaciones móviles híbridas	3
2.1.2. Análisis del problema	4
2.2. Estado del arte y elección de la tecnología	5
2.2.1. El lenguaje de programación	6
2.2.2. Framework web	7
2.2.3. Sistema para el control de versiones	9
2.2.4. Entorno de desarrollo	10
3. Objetivos	13
3.1. Repercusión de los objetivos	14
4. Fases en el desarrollo	17
4.1. Metodología de desarrollo	17
5. Análisis	21
5.1. Análisis de los requisitos funcionales	21
5.2. Análisis de los requisitos no funcionales	22
5.3. Casos de uso	22
5.3.1. Actores	22
5.3.2. Casos de uso	23
5.3.3. Diagrama de casos de uso.	47
5.4. Diagrama de navegación	50
6. Diseño e Implementación	51
6.1. Modelado de la base de datos	51
6.2. Diseño de las interfaces	52
6.2.1. Bocetos	53
6.2.2. Interfaz	54

6.3.	Implementación	57
6.3.1.	En el lado del servidor	57
6.3.2.	En el lado del cliente	58
6.3.3.	Otras herramientas	59
7.	Despliegue	61
7.1.	Herramientas de Heroku	61
7.2.	Entorno de nuestro código	62
7.2.1.	Entorno virtual	63
7.2.2.	Repositorio para el proyecto	63
7.2.3.	Cambios necesarios para el despliegue	64
8.	Conclusiones y vías futuras	67
Bibliografía		69
.1.	Bibliografía	69
.2.	Recursos Web	69
.2.1.	Otros recursos web	70

Índice de figuras

2.1.	App híbrida	4
2.2.	Top languages GitHub	6
2.3.	Pycharm 1/2	12
2.4.	Pycharm 2/2	12
4.1.	Modelo cascada	18
5.1.	Diagrama de casos de uso actor usuario anónimo.	48
5.2.	Diagrama de casos de uso actores alumno y profesor.	48
5.3.	Diagrama de casos de uso actor profesor.	49
5.4.	Diagrama de casos de uso actor administrador.	49
5.5.	Diagrama de navegación	50
6.1.	Entidad relación	51
6.2.	Tablas del modelo entidad relación	52
6.3.	Boceto interfaz inicio de sesión.	53
6.4.	Boceto interfaz menú lateral.	54
6.5.	Boceto interfaz conceptos.	54
6.6.	Interfaz de usuario inicio de sesión.	55
6.7.	Interfaz de usuario conceptos 1/2.	55
6.8.	Interfaz de usuario conceptos 2/2.	56
6.9.	Interfaz de usuario estadísticas de usuario.	56
7.1.	Login a Heroku.	62
7.2.	Conexión ssh a Heroku.	62

Índice de tablas

2.1. Tabla de características.	7
5.1. CU-1. Inicio de sesión.	24
5.2. Curso alterno de CU-1. Inicio de sesión.	24
5.3. CU-1. Inicio de sesión.	25
5.4. Curso alterno de CU-2. Registro en el sistema.	25
5.5. CU-3. Cerrar sesión.	26
5.6. CU-4. Consulta estadísticas propias.	27
5.7. CU-5. Consultar concepto.	28
5.8. Curso alterno de CU-5. Consultar concepto de un tema.	29
5.9. CU-6. Resolver pregunta de varias opciones y una respuesta.	30
5.10. CU-7. Resolver pregunta tipo test verdadero/falso.	31
5.11. CU-8. Resolver pregunta tipo test verdadero/falso.	33
5.12. CU-9. Resolver pregunta tipo test verdadero/falso.	34
5.13. CU-10. Registrar estadísticas de preguntas opción múltiple.	35
5.14. CU-11. Registrar estadísticas de preguntas verdadero/falso.	36
5.15. CU-12. Registrar estadísticas de examen de preguntas de opción múltiple.	37
5.16. CU-13. Registrar estadísticas de examen de preguntas verdadero/falso.	38
5.17. CU-14. Consulta estadísticas de alumno.	39
5.18. CU-15. Actualizar datos de conceptos.	40
5.19. CU-16. Actualizar preguntas con varias opciones.	41
5.20. CU-17. Actualizar preguntas verdadero/falso.	42
5.21. CU-18. Dar permisos de profesor.	44
5.22. CU-19. Eliminar permisos de profesor.	45
5.23. CU-20. Eliminar usuario.	47

Capítulo 1

Resumen

Breve resumen y palabras clave

Palabras clave: *Django, Python, Materializecss, aplicación web, desarrollo web, docencia.*

iOrg2.0 es un proyecto que nace de la necesidad en el Departamento de Organización de Empresas de la Facultad de Ciencias Económicas y Empresariales de la Universidad de Granada de desarrollar una aplicación móvil como apoyo a la docencia de sus asignaturas.

Así pues, se inició una colaboración con la Oficina de Software Libre para integrar las nuevas tecnologías en la docencia de estas asignaturas. Inicialmente se desarrolló como una aplicación móvil híbrida¹ dado que no era posible, por parte del departamento, mantener un servidor back-end y una base de datos.

iOrg2.0 pretende continuar con la idea original de explotar las virtudes de las nuevas tecnologías en el ámbito docente y a su vez solventar y mejorar situaciones de la primera versión híbrida, soluciones y decisiones que desarrollaremos en el siguiente capítulo. Sin embargo en esta versión se deja a un lado la aplicación móvil y se apuesta por el análisis y desarrollo de una solución que pasa por un servidor web.

¹Combinación de tecnologías web que no son aplicaciones móviles nativas ni tampoco están basadas en Web, porque se empaquetan como aplicaciones para distribución y tienen acceso a las APIs nativas del dispositivo.

iOrg2.0: Herramienta de apoyo a la docencia para el Departamento de Organización de Empresas

Rogelio Gil García

Extended abstract and key words

Key words: *Django, Python, Materializecss, aplicación web, desarrollo web, docencia.*

Capítulo 2

Introducción

Motivación y análisis del problema

Como ya hemos hablado en el capítulo anterior, iOrg2.0 viene de una aplicación móvil híbrida en la que se detectaron carencias que probablemente provengan de la inexperiencia y el escaso tiempo de análisis del que se disponía. Vamos a exponer brevemente las características de las aplicaciones híbridas para entender los problemas que este proyecto pretende solventar.

Aplicaciones móviles híbridas

Con la llegada de los *smartphone* se ha impulsado enormemente el consumo y producción de apps móviles. En un principio, una aplicación móvil se desarrollaba como aplicación nativa con los contras que ello supone, como una curva de aprendizaje alta de un sistema concreto, en la mayoría de los casos desarrollo monoplataforma, y también los pros como un desarrollo y control más específico del sistema, sistemas más robustos,etc.

Ni que decir tiene que una aplicación compleja necesita las ventajas de un desarrollo específico y nativo de un sistema móvil, pero ¿qué pasa con las aplicaciones simples y livianas que, cada vez más, consumimos? Es aquí donde toman importancia las aplicaciones móviles híbridas.

Las principales ventajas de estas aplicaciones son:

- Bajo coste de desarrollo.
- Multiplataforma.
- Tiempo de desarrollo corto.
- Fácil distribución.

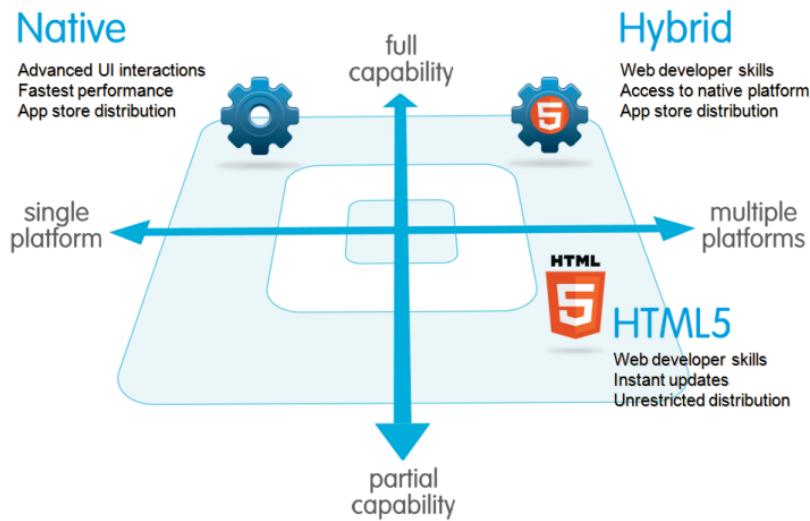


Figura 2.1: App móvil híbrida (<https://developer.salesforce.com>).

Y lo que más nos interesa para este proyecto, sus desventajas:

- Rendimiento pobre.
- Escalabilidad baja.
- Diseño sujeto al diseño web.
- Acceso a las características especiales del hardware limitado.

En un principio la aplicación que se desarrolló en el período de prácticas ICARO se planteó como una aplicación sencilla donde consultar conceptos y esquemas de las asignaturas. Los datos se consultarían desde una hoja de Google Drive común al departamento y se mostrarían en la aplicación directamente y por estas características, se .

A medida que avanzaba el desarrollo y acercándose la fase final, el “cliente” planteó nuevas funcionalidades, topándonos con el primer problema: **la escalabilidad** de estas aplicaciones.

Análisis del problema

Una vez desarrollada esta aplicación, aunque en general quedamos contentos con el resultado, acusamos los siguientes problemas:

- No disponíamos de un servidor, toda la aplicación se ejecutaba en el cliente.

- Falta de base de datos.
- No se usa la API de Google Drive
- Acceso a Google Drive en cada sección de la aplicación
- Lentitud de acceso de datos en cada sección de la aplicación
- Mala elección de framework CSS, señales de obsoleto.

Solventar estos problemas, con el fin de iniciar un proyecto mejor planificado, con miras a una mayor escalabilidad analizando los requisitos presentes y los que pudieran surgir en un futuro, es la principal motivación para este proyecto. Aprovecharemos para mejorar la experiencia de usuario, ya que la lentitud de carga entre secciones era uno de los grandes problemas que hacían tediosa la consulta del contenido de la asignatura.

El primer paso importante es olvidar la tecnología móvil híbrida, que originó muchos problemas en el proyecto anterior y dividirlo en dos, una parte de desarrollo web y una parte de desarrollo móvil capaces de escalar sin problema. Nuestro proyecto se centrará en la aplicación web.

Estado del arte y elección de la tecnología

Para abordar los problemas descritos y antes de realizar un análisis más profundo se hará un estudio de las tecnologías web que pueden sernos útiles en proyectos de este tipo. Teniendo claro que vamos a desarrollar el proyecto bajo una arquitectura web y hecho el repaso de los inconvenientes en las tecnologías híbridas podemos sintetizar los siguientes puntos que necesita nuestro nuevo proyecto:

- Servir el contenido con un servidor web.
- Consultar el contenido de la hoja de Google Drive del Departamento.
- Almacenar el contenido en una base de datos.
- Gestión de usuarios.
- Establecer roles para los usuarios.

El lenguaje de programación

Lo primero que vamos a plantearnos es el **lenguaje** a usar, y que mejor forma de abrir un abanico de posibilidades que explorar los lenguajes más usados en la comunidad de GitHub¹ desde mediados de 2016 a mediados de 2017.

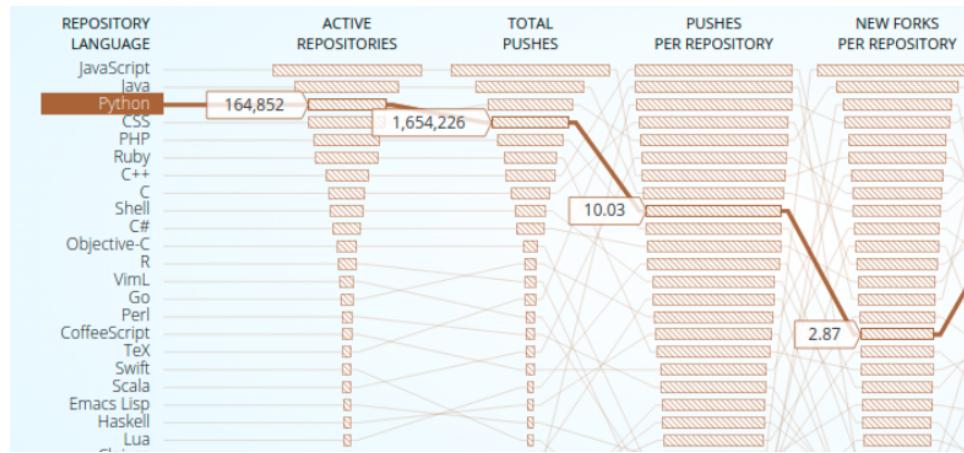


Figura 2.2: Top languages GitHub (<http://githut.info/>).

Como podemos ver en la figura anterior, entre los 5 lenguajes más usados se encuentran: JavaScript, Java, Python, PHP y Ruby (y sí, está CSS y lo hemos obviado ya que es un lenguaje usado únicamente para definir los estilos de un documento HTML o XML).

JavaScript nos ofrece la alternativa de Node.js para el lado del servidor pero no está diseñado para aplicaciones complejas. Node.js es un lenguaje interpretado y aunque trabaja de forma asíncrona, está diseñado para ejecutarse en un sólo hilo, por lo que podría presentar ciertas limitaciones a la hora de escalar en aplicaciones complejas. Además las aplicaciones con Node.js permanecen en RAM, no siendo re-interpretadas con nuevas peticiones si no que se procesa directamente desde la memoria.

JavaScript no se estudió profundamente durante el Grado en Ingeniería Informática y aunque descubrir e indagar en un lenguaje nuevo podría ser un aliciente, sus propiedades y las “limitaciones” en cuanto a la escalabilidad,

¹GitHub es una plataforma de desarrollo colaborativo donde usted puede alojar proyectos propios así como contribuir en otros proyectos. GitHub usa el sistema de control de versiones Git.

que recuerdan al desarrollo en JavaScript de las aplicaciones híbridas, nos hacen descartar este lenguaje.

Java en cambio es un lenguaje que hemos profundizado más durante los años de estudio. Si es verdad que la mayoría del desarrollo que llevamos a cabo estos años era referente a aplicaciones de escritorio, pero la idea de un lenguaje nuevo o poco profundizado nos parecía más interesante en este proyecto.

Llegamos a los tres lenguajes que nos parecieron más interesantes para el proyecto **PHP**, **Ruby** y **Python**[1]. A una primera vista estos lenguajes eran candidatos para el desarrollo, vamos a señalar las características que se observaron para decidirnos por uno:

Características		
PHP	Ruby	Python
Curva de aprendizaje corta	Curva de aprendizaje compleja	Curva de aprendizaje corta
Legibilidad y estructuración baja	Legibilidad y estructuración media	Legibilidad y estructuración alta
Popularidad alta	Popularidad baja	Popularidad media
Comunidad media	Comunidad media	Comunidad alta

Tabla 2.1: Tabla de características.

Hemos visto que PHP es el más usado en cuanto a cantidad de aplicaciones totales desarrolladas, pero a lo largo del último año, como se aprecia en la figura 2.2, Python ha crecido enormemente en el número de contribuciones y en su comunidad. Viendo la evolución de este lenguaje y teniendo en cuenta sus características que ayudan a crear aplicaciones bien estructuradas, escalables, con módulos altamente reutilizables y teniendo una comunidad, tanto anglosajona como hispana, tan grande, Python será la opción elegida como lenguaje de iOrg2.0.

Decir también que durante el grado se hizo una breve introducción a Python y se usó para pequeñas y aisladas aplicaciones, sintiendo una gran inquietud por usar este lenguaje en un proyecto más completo.

Framework web

Un framework web[5, 2] es una colección de paquetes o módulos que permiten a los desarrolladores escribir aplicaciones o servicios web sin tener

que manejar detalles de bajo nivel como protocolos, sockets o gestión de procesos.

Las aplicaciones web comúnmente utilizan y hacen uso de una serie de componentes, ya sea durante su desarrollo o en la etapa de producción, tales como un servidor HTTP, mecanismos de almacenamiento como podría ser una base de datos, un motor de plantillas, un conjunto de herramientas AJAX, etc. Estos framework agrupan y gestionan paquetes con estos componentes que ayudan en la abstracción del desarrollo de bajo nivel.

Ya que tenemos claro el lenguaje, vamos a hacer un breve estudio de los frameworks web existentes que nos permitirán un desarrollo ágil y mejor estructurado. Los tres frameworks web más importantes de Python por su volumen de proyectos, comunidad y completitud son:

- Django
- TurboGears2
- Web2Py

Entre ellos, **Django**[2] es claramente el más usado y con una comunidad más grande. Se basa en el principio DRY² lo que propicia un desarrollo más rápido, con menos código y por consecuencia más limpio y pragmático. Django se centra en automatizar tanto como sea posible. En la documentación de Django se habla de que el framework usa un enfoque MVC³ y en la misma documentación también se habla de un enfoque MVT⁴. Esto es porque ambos enfoques están estrechamente relacionados, se habla de un MVT donde el controlador lo gestiona el propio framework trabajando el desarrollador directamente con los “Templates”.

TurboGears2[7] aparece con la idea de resolver las frustraciones de otros frameworks y con la filosofía de adoptar lo mejor de otros frameworks de características similares y resolver sus puntos flacos. Framework que destaca por la agilidad en el desarrollo de aplicaciones y servicios simples y la capacidad de escalar hasta aplicaciones completas. Utiliza un concepto de modularización que permite adaptar distintos motores de plantillas y webservers. Utiliza como controlador SQLAlchemy⁵, ORM muy bien considerado y afamado.

²Don't Repeat Yourself

³Modelo Vista Controlador

⁴Modelo Vista Template

⁵Conjunto de herramientas Python SQL y ORM que ofrece al desarrollador toda la flexibilidad y funcionalidad de SQL

Por otro lado **Web2Py** es el más sencillo de los tres. No tiene archivos de configuración, no requiere instalación e incluso se puede ejecutar desde una unidad USB. Extendido y con una comunidad amplia está muy indicado para proyectos sencillos. Al igual que TurboGears2 utiliza la filosofía de agrupar lo mejor de otros frameworks. Utiliza la idea de MVC de Ruby On Rails y la filosofía de forms de Django por lo que para muchos desarrolladores será muy cómodo si conocen estos frameworks.

Para este proyecto el framework que hemos decidido utilizar es **Django**. Al ver su comunidad, volumen de proyectos y calidad de los mismos tales como Instagram, Mozilla o Pinterest [6] creemos que es el indicado para desarrollar la aplicación. Django es compatible con Python 2 y 3 y muy recomendable para proyectos con miras a una gran escalabilidad. Personalmente y antes de desarrollar iOrg2.0 me parece interesante ya no sólo la gran comunidad de desarrolladores también la calidad de su documentación en cuanto a explicaciones y ejemplos. Creemos por esto que es la mejor apuesta para el TFG.

Sistema para el control de versiones

El control de versiones se dedica a la gestión de los cambios que se realizan sobre los ficheros o configuraciones. Una revisión o versión de un fichero es el estado en el que se encuentra este en cierto momento de su desarrollo.

Un sistema de control de versiones facilita la administración de las distintas versiones del código o el producto desarrollado, así como las posibles especializaciones realizadas, ya sea para una bifurcación del desarrollo en un punto del proyecto o para mantener una versión estable en producción mientras se desarrolla un proyecto.

Durante el Grado en Ingeniería Informática hemos trabajado con Git como sistema de control de versiones satisfaciendo totalmente las necesidades de los desarrollos. No encontramos razones para dejar Git y utilizar otros sistemas. Aún así se indagó en otras opciones comunes en el desarrollo de proyectos encontrando dos opciones en la mayoría de estos: Git y SubVersion. Las principales características son:

Git

- Control de versiones distribuido.
- Se trabaja sobre copias locales del repositorio.
- Autorización para la totalidad del repositorio.

- Sólo es necesaria conexión para la sincronización.

SubVersion

- Control de versiones centralizado.
- Repositorio central donde se generan copias de trabajo.
- Autorización sobre rutas concretas del repositorio.
- Necesaria conexión a la red con cada acceso.

Ya que necesitamos un repositorio donde poder trabajar con o sin conexión a internet, excepto cuando necesitemos sincronizar nuestro trabajo local, y no necesitamos un control de permisos sobre rutas específicas del repositorio, tendremos acceso al repositorio completo, seguimos pensando que Git es la mejor opción.

Aunque hablaremos de ello más adelante, vamos a tener la precaución de dividir los repositorios del proyecto en dos. Un repositorio será destinado al código fuente y otro a la documentación. Esto nos ayudará en el momento de desplegar la aplicación en un servidor de desarrollo ya que podremos subir los ficheros fuente independientemente sin tener que manejar los ficheros de la documentación en el mismo repositorio.

Entorno de desarrollo

En un primer momento se pensó en desarrollar la aplicación sin ningún entorno de desarrollo integrado (IDE), simplemente bajo un editor de texto y las herramientas de terminal necesarias. Se planteó esta idea por dos razones principales. La primera que se buscaban herramientas livianas que no aumentaran mucho la carga de procesamiento del equipo en el que se iba a desarrollar y la segunda es que durante el grado nos hemos topado con problemas de los entornos de desarrollo al automatizar muchas tareas, si bien es cierto que esto abstrae al desarrollador de procedimientos repetitivos (por ejemplo lanzar el servidor desde la terminal cada vez que se realizan ciertos cambios en el desarrollo) también puede ser contraproducente al no controlar de forma manual todas estas tareas. Aún así esto siempre dependerá del desarrollador y del buen o mal uso que haga de estos entornos de desarrollo.

Como decimos, las primeras pruebas se desarrollaron en un editor de textos y una terminal, comprendiendo la instalación de Python y Django, sus paquetes, el entorno virtual, etc. Pronto nos topamos con algunas dificultades a la hora de desarrollar las primeras funcionalidades dentro del entorno virtual de prueba y es que al ser tecnologías nuevas para nosotros, con una sintaxis nueva y una metodología de Django un poco peculiar, invertíamos

mucho tiempo en ver si lo que programado estaba bien o no, si había errores de sintaxis, o si la forma era la correcta. Por esto decidimos investigar un poco en los IDE y probar alguno antes de dejar los entornos de prueba y empezar el desarrollo.

Durante el grado si hemos usado un IDE en algunas asignaturas este ha sido NetBeans y es el primero que exploramos pero, desgraciadamente, no es la mejor opción para Django. Si bien es cierto que encontramos artículos sobre cómo crear proyectos Django con NetBeans, también encontramos muchos artículos desaconsejándolo ya que aunque existen paquetes para poder trabajar con Django, no son muy estables, y al intentar crear un nuevo proyecto de prueba hemos encontrado problemas como :

- Variables de entorno no localizables.
- NetBeans funciona mal con el entorno virtual.
- Paquetes para Django antiguos y desactualizados.

Por esto y sin dedicar mucho a intentar solucionar con paquetes externos estos problemas pasamos al siguiente IDE ya recomendado por compañeros del Grado para Django: PyCharm. Este IDE a diferencia de NetBeans está totalmente integrado con Python y Django y las características que más agilidad van a aportar al desarrollo son:

- Entorno virtual totalmente integrado. Al arrancar el proyecto ya dispones de una consola en este.
- Consola Python con la versión actualmente utilizada.
- Arranca el servidor con el atajo de teclado Mayus+F10.
- Sincronización con el control de versiones utilizado en el entorno virtual.

En las figuras 2.3 y 2.4 podemos ver dos capturas de pantalla del entorno de desarrollo.

Además JetBrains⁶ ofrece el paquete profesional de PyCharm gratuito a estudiantes al que podemos acceder registrando nuestro correo de estudiante de la Universidad de Granada.

Por las pruebas realizadas y las ventajas que ofrece vamos a utilizar PyCharm como IDE.

⁶JetBrains: compañía desarrolladora de PyCharm (<https://www.jetbrains.com/>)

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The title bar indicates the project is named 'iOrg2' and the current file is 'settings.py'. The left sidebar displays the 'Project' view with a tree structure showing 'iOrg2' and 'External Libraries'. The main code editor window shows the contents of 'settings.py':

```
1  """
2  Django settings for iOrg2 project.
3
4  Generated by 'django-admin startproject' using
5  the 'django' template.
6
7  For more information on this file, see
8  https://docs.djangoproject.com/en/1.10/topics/
9  settings/
10 # For the full list of settings and their
11 # values, see https://docs.djangoproject.com/en/1.10/ref/
12 # settings/
13
14 import os
15
16 # Build paths inside the project like this:
#BASE_DIR = os.path.dirname(os.path.dirname(
#    os.path.abspath(__file__)))
```

Figura 2.3: Captura de pantalla de la zona del proyecto

The screenshot shows the Django Console window in PyCharm. The title bar says 'Django Console'. The console area has a dark background with white text. It shows a command prompt with '>>>'. Below the console are several small icons. At the bottom, there are tabs for 'Run', 'TODO', 'Python Console' (which is selected), 'Terminal', and 'Version Control'. The status bar at the bottom shows the path '/home/.../venv/10g/bin/python' and the date '2016-11-10 17:03:23'.

Figura 2.4: Captura de pantalla de la zona de la consola

Capítulo 3

Objetivos

iOrg2.0 persigue el objetivo general de desarrollar una aplicación web de apoyo a la docencia para las asignaturas del Departamento de Organización de Empresas de la Facultad de Ciencias Económicas y Empresariales de la Universidad de Granada. Esta aplicación deberá obtener los datos de una hoja de cálculo de Google Drive en la cual los profesores de dicho departamento tienen información docente de sus asignaturas, y trabajar con estos datos para mostrarlos al alumno. Seguiremos la filosofía de **Software Libre** heredada del anterior proyecto.

A continuación vamos a listar los objetivos principales:

- **OBJ-P-1.** Conseguir una primera versión funcional de la aplicación donde un alumno pueda consultar los conceptos y resolver algunas preguntas de la asignatura.
- **OBJ-P-2.** El proyecto debe estar estructurado de forma que sea escalable, permitiendo añadir en un futuro nuevas funcionalidades no dependientes de las actuales.
- **OBJ-P-3.** Conexión segura con Google Drive mediante su API.
- **OBJ-P-4.** Solventar los problemas de latencia al navegar entre secciones de la versión anterior.
- **OBJ-P-5.** Heredar el principio de Software Libre de la versión anterior, publicando este proyecto en GitHub y permitiendo a la comunidad participar en él.

También propondremos unos objetivos secundarios, que se intentarán alcanzar en la medida de lo posible:

- **OBJ-S-1.** Añadir estadísticas básicas sobre la actividad de los alumnos en la aplicación.

- **OBJ-S-2.** Adaptar las vistas de la aplicación web a los tamaños de resolución de diferentes dispositivos.
- **OBJ-S-3.** Tener en cuenta el uso de la aplicación desde entornos con internet de velocidad limitada, mejorando así su usabilidad.
- **OBJ-S-4.** Despliegue de la aplicación en PaaS¹.
- **OBJ-S-5.** Gestión de los usuarios y sus roles, al menos permisos de alumnos y profesores.
- **OBJ-S-6.** Consulta, por parte de los profesores, de las estadísticas generadas por los alumnos, siendo estas útiles para valorar la actividad de los alumnos en la aplicación.

Para el desarrollo de este proyecto vamos a hacer uso de los conocimientos adquiridos en las siguientes asignaturas del Grado en Ingeniería Informática:

- **Fundamentos de programación**, base para cualquier desarrollo.
- **Programación orientada a objetos**, para entender este paradigma de programación.
- **Ingeniería del software**, para hacer un correcto análisis y planificación de un proyecto.
- **Base de datos**, para comprender la gestión de las bases de datos necesarias en este proyecto.
- **Diseño de Aplicaciones para Internet**, donde vimos una breve introducción a Python, Django y el uso de varias APIs de Google.
- **Infraestructura virtual**, para comprender el uso de sistemas PaaS, entornos virtuales, despliegue de aplicaciones y Git como sistema de control de versiones.

Repercusión de los objetivos

El desarrollo de este proyecto debería resolver los objetivos principales para crear una base de proyecto en el cual se podrá seguir trabajando en un futuro y crear un apoyo docente para las asignaturas de el Departamento de Organización de Empresas. Con estos objetivos se crea una base de un desarrollo escalable, en el que se podrán incluir sin problemas nuevas funcionalidades completando cada vez más este proyecto.

¹ “Platform as a Service”. Plataforma y entorno que permite a los desarrolladores crear y ejecutar aplicaciones totalmente en la nube

Los objetivos secundarios darán más consistencia al proyecto mejorando su rendimiento, expansión y usabilidad, así como un valor añadido para las asignaturas del departamento, facilitando la labor de docentes y el aprendizaje de los alumnos. Se podrá también tener un primer acercamiento por parte de la comunidad docente al desplegarla en un PaaS y estudiar las carencias y virtudes tras su uso.

Capítulo 4

Fases en el desarrollo

Metodología de desarrollo

Como para inicio de todo proyecto deberemos atender al **modelo o metodología de desarrollo**. Es aquí donde toma más peso la ingeniería de “software” [3] estableciendo una metodología formal, con resultados predecibles por las fases de planificación que permitan desarrollar un producto de alta calidad que cumpla las expectativas del cliente.

Para esta aplicación vamos a elegir un modelo de desarrollo clásico o en cascada[9]. Las principales ventajas e inconvenientes de este modelo son:

Ventajas

- Propicio para proyectos maduros y bien especificados por parte del cliente que no necesitan de mucha investigación para garantizar el desarrollo de los objetivos.
- Modelo secuencial, en su mayoría, fácil de organizar y entender.
- Modelo muy extendido y utilizado.
- Promueve una metodología de trabajo efectiva: definir antes de diseñar, diseñar antes de codificar

Inconvenientes

- En el entorno “cliente-desarrollador” rara vez el proyecto sigue etapas secuenciales bien definidas, el proyecto va madurando a medida que se va desarrollando, lo cual lo hace ineficaz con este modelo de desarrollo.
- Tiempo de desarrollo hasta una versión funcional alto ya que no se buscan prototipos intermedios, el desarrollo finaliza una vez terminado y probado.

- Cualquier error de diseño detectado después de su etapa correspondiente conduce al rediseño y nuevo desarrollo de las etapas afectadas, lo que incremente los costos del desarrollo.
- Una etapa no se puede iniciar si no se ha terminado la anterior, lo que perjudica a equipos de desarrollo donde cada parte se encarga de una etapa.

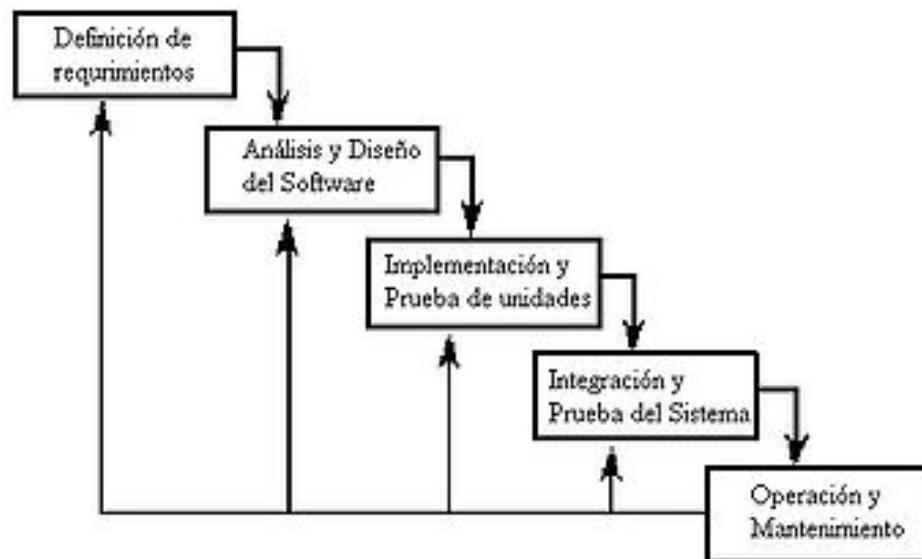


Figura 4.1: Modelo clásico o de cascada

Ya que nuestro equipo de desarrollo se compone de una única persona y los requisitos del proyecto están bien definidos vamos a adoptar este modelo de desarrollo.

El modelo de cascada se compone de las siguientes fases:

- **Análisis de requisitos:** se determinan los objetivos que debe cumplir el “software” que cubran las necesidades de los usuarios finales. Se realiza la especificación completa de las funcionalidades del sistema sin entrar en detalles internos. Por el tipo de modelo de desarrollo secuencial esta primera fase toma especial importancia. Debe quedar claro todo lo que requiere el sistema ya que no se podrán requerir nuevas funcionalidades a mitad del proceso de desarrollo del proyecto.
- **Diseño del sistema:** se descompone el problema a resolver y sintetiza en elementos que puedan elaborarse por separado. De esta forma el

equipo de trabajo puede optimizar su rendimiento y elaborar las partes no dependientes. Como resultado del diseño del sistema tendremos una descripción de la estructura global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la relación entre ellas. Distinguiremos entre diseño de alto nivel y diseño detallado.

Con el **diseño de alto nivel** de ellos se pretende definir la estructura de la solución (después de analizar el problema en la fase anterior) identificando las diferentes partes. En el segundo de ellos, el diseño detallado definiremos los algoritmos empleados para el cumplimiento de los requerimientos del cliente y la organización del código para seguir con la implementación.

- **Implementación:** en esta fase se traducen los diseños a una solución legible para el sistema. Es la fase donde se implementa el código fuente a partir de las fases de análisis y diseño.
- **Integración y prueba del sistema:** es la fase donde se ensamblan las diferentes partes implementadas y se comprueba que funcionan correctamente y que cumplen los requisitos tanto funcionales como no funcionales. Si se ha seguido un desarrollo cuidadoso de la fase de análisis y diseño, esta fase no debe de resultar costosa, las pruebas del sistema no deben encaminar a un rediseño, recordar aquí la importancia de la fase de análisis del problema.
- **Despliegue:** una vez integradas las partes del sistema y probadas pasamos a la fase de **despliegue** en el lugar donde se alojará, este es el entorno de producción. Comprobaremos que las pruebas sobre este entorno se corresponden con los resultados obtenidos en la fase anterior en el entorno de desarrollo. Una vez se compruebe la integridad de la aplicación en este entorno de producción se da por concluido el desarrollo.
- **Mantenimiento:** se entiende como fase de mantenimiento todo el proceso posterior al despliegue del entorno de producción y es una de las etapas más críticas, ya que un 75 % de los recursos se destinan a partir de esta fase. Suponiendo que el desarrollo de un proyecto medio pueda durar entre uno y dos años, la fase de mantenimiento se podría alargar entre quince y veinte años.

En esta fase el usuario final hace uso real del sistema en el entorno de producción y es cuando se determina finalmente si el producto satisface las necesidades por las cuales se implementó a lo largo del tiempo.

Capítulo 5

Análisis

Siguiendo las necesidades del cliente sintetizadas en los objetivos[3] iniciaremos el análisis de los requisitos de la aplicación. Dividiremos la fase de análisis en dos: análisis de los requisitos funcionales y no funcionales [3].

Análisis de los requisitos funcionales

Los requisitos funcionales son las características requeridas del sistema que expresan una funcionalidad de este:

- **RF-1.** Consulta de datos de una hoja de cálculo de Google Drive
- **RF-2.** Mostrar conceptos de la asignatura
- **RF-3.** Se deberá generar preguntas aleatorias de cada tema. Las preguntas deberán estar diferenciadas en dos tipos:
 - **RF-3.1.** Generación de pregunta tipo test con varias opciones y sólo una seleccionable.
 - **RF-3.2.** Generación de pregunta verdadero/falso.
- **RF-4.** Se deberá generar un examen de 20 preguntas aleatorias del tipo de preguntas seleccionado.
- **RF-5.** Se generarán estadísticas básicas de las actividades realizadas por cada alumno en la aplicación.
 - **RF-5.1.** Número de preguntas correctas e incorrectas.
 - **RF-5.2.** Número de exámenes aprobados y suspendidos.
 - **RF-5.3.** Nota media de los exámenes aprobados y suspendidos así como la nota media total.
- **RF-6.** Los profesores podrán consultar las estadísticas de los alumnos.

- **RF-7.** Los profesores podrán actualizar los datos de la aplicación, estos se leerán de la hoja de cálculo de Google Drive.
- **RF-8.** Gestión de los usuarios de la aplicación.

Análisis de los requisitos no funcionales

Los requisitos no funcionales hacen referencia a las cualidades del producto requeridas por el cliente. Estas características no se limitan a la aplicación, pueden ser referentes al sistema, al proceso de desarrollo o incluso al entorno.

- **RNF-1.** Rendimiento. Especial cuidado con la velocidad de carga entre secciones.
- **RNF-2.** Conceptos y preguntas se agruparán por temas.
- **RNF-3.** La aplicación se visualizará desde diversos dispositivos, atender a las diferentes resoluciones de pantalla.
- **RNF-4.** Eficiencia en el uso y descarga de datos. Esto es, por ejemplo, el usuario que consulta la aplicación no deberá descargar todo el contenido de la hoja, sólo deberá descargar los datos de la sección consultada.

Casos de uso

Los casos de uso son una técnica para especificar el comportamiento de un sistema. Hacen referencia a los procedimientos llevados a cabo entre el sistema y una entidad para dar una funcionalidad. Las entidades que participan en estas interacciones con el sistema se denominan actores.

Actores

- **Ac-1. Usuario anónimo.**
 - **Descripción:** Persona que visita la aplicación.
 - **Características:** Es el usuario que llega por primera vez a la aplicación.
 - **Relaciones:** Ninguna.
 - **Atributos:** Ninguno.
 - **Comentarios:** Este usuario no puede hacer uso de la aplicación más que ver qué actividades se desarrollan o alguna información sobre el proyecto.

■ Ac-2. Usuario alumno.

- **Descripción:** Persona que ha iniciado sesión en la aplicación.
- **Características:** Es el usuario que puede realizar las actividades docentes de la asignatura.
- **Relaciones:** Ninguna.
- **Atributos:** usuario, contraseña, nombre, apellidos y email.
- **Comentarios:** No tiene permisos de administración.

■ Ac-3. Usuario profesor.

- **Descripción:** Persona que visita la aplicación.
- **Características:** Usuario con permisos de administración de la información docente.
- **Relaciones:** Ninguna.
- **Atributos:** usuario, contraseña, nombre, apellidos y email.
- **Comentarios:** Podrá consultar las estadísticas generadas de los usuarios alumnos.

■ Ac-4. Usuario administrador.

- **Descripción:** Es el administrador del sistema.
- **Características:** Usuario que puede gestionar los demás usuarios, así como dar permisos al usuario profesor.
- **Relaciones:** Ninguna.
- **Atributos:** usuario, contraseña y email.
- **Comentarios:** Aunque no se restrinja, este usuario no está destinado al uso de la aplicación más allá de las labores de gestión del sistema.

Casos de uso**■ CU-1. Inicio de sesión en el sistema.**

- **Actores:** Usuario anónimo.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** No tener iniciada una sesión previamente.
- **Postcondición:** El usuario anónimo pasará a ser ususario alumno ó ususario profesor.
- **Autor:** Rogelio Gil García.

- **Versión:** 1.0.
- **Propósito:** Pasar a ser un usuario con la sesión iniciada en el sistema.
- **Resumen:** El usuario introducirá su nombre de usuario y su contraseña para iniciar sesión.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Iniciar sesión”.	
		2 El sistema muestra el formulario para inicio de sesión.
3	El usuario rellena los campos del formulario y pulsa el botón “Iniciar sesión”.	
		4a El sistema valida el usuario y la contraseña y si ambos están bien y coinciden inicia la sesión de ususario.

Tabla 5.1: CU-1. Inicio de sesión.

Curso alterno	
4b	Si algo ha ido mal en la validación del usuario y contraseña, el sistema vuelve al paso 2 de CU-1 (tabla 5.1).

Tabla 5.2: Curso alterno de CU-1. Inicio de sesión.

■ **CU-2.** Registro en el sistema.

- **Actores:** Usuario alumno.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** No tener iniciada una sesión previamente.
- **Postcondición:** El usuario tendrá una cuenta con la que iniciar sesión.
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Tener una cuenta de usuario alumno.
- **Resumen:** El usuario llenará un formulario de registro del sistema y si todo va bien, tendrá una cuenta con la que iniciar sesión.

Curso normal			
Actor		Sistema	
1	El usuario hace click en el botón “Registrarse”.		
		2	El sistema muestra el formulario para registrar un usuario. En este formulario se pedirán: nombre de usuario, contraseña, e-mail, nombre y apellidos.
3	El ususario rellena el formulario y lo envía con el botón “Registrar”.		
		4a	El sistema valida los datos y sin son correctos el sistema registra el usuario.

Tabla 5.3: CU-1. Inicio de sesión.

Curso alterno	
4b	Si algo ha ido mal en la validación del formulario, el sistema vuelve al paso 2 de CU-2 (tabla 5.3).

Tabla 5.4: Curso alterno de CU-2. Registro en el sistema.

■ **CU-3.** Cerrar sesión.

- **Actores:** Usuario alumno, usuario profesor, usuario administrador.
- **Tipo:** Primario.
- **Referencias:**
- **Precondición:** Tener una sesión iniciada.
- **Postcondición:** El usuario no tendrá la sesión iniciada.
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Cerrar la sesión de un usuario en el sistema.
- **Resumen:** El usuario pulsará el botón de cerrar sesión y el sistema cerrará la misma, volviendo a la pantalla de inicio de sesión paso 2 de CU-1 (tabla 5.1).

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Cerrar sesión”.	
		2 El sistema cierra la sesión del usuario y vuelve al paso 2 de CU-1 (tabla 5.1)

Tabla 5.5: CU-3. Cerrar sesión.

■ **CU-4.** Consulta estadísticas propias.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada una sesión de usuario.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** El usuario consulta sus estadísticas.
- **Resumen:** El usuario consultará sus estadísticas creadas a partir del uso de la aplicación.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Perfil”.	
		2 El sistema muestra el perfil del usuario con su nombre de usuario y sus estadísticas básicas: número de preguntas correctas e incorrectas, número de exámenes aprobados y suspensos, nota media de exámenes aprobados y suspensos y la nota media total de los exámenes realizados.

Tabla 5.6: CU-4. Consulta estadísticas propias.

■ **CU-5.** Consultar concepto de un tema.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener una sesión iniciada.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Consultar un concepto.
- **Resumen:** El usuario consulta el concepto deseado de un tema concreto.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Conceptos”.	
		2 El sistema muestra una lista con los temas que contiene a los conceptos.
3	El usuario pulsa sobre el tema deseado.	
		4a El sistema muestra las secciones en las que están agrupados los conceptos dentro de los temas.
5	El usuario pulsa sobre la sección en que se encuentra el concepto.	
		6 El sistema muestra los conceptos que contiene esa sección del tema.
7	El usuario pulsa sobre el concepto deseado.	
		8 El sistema muestra una pantalla con la información de ese concepto: nombre, definición ejemplo y figura(opcional).

Tabla 5.7: CU-5. Consultar concepto.

Curso alterno	
4b	Si el tema no tiene secciones el sistema nos llevará al paso 6 de CU-5 (tabla 5.7).

Tabla 5.8: Curso alterno de CU-5. Consultar concepto de un tema.

- **CU-6.** Resolver pregunta tipo test con varias opciones y una sola respuesta.
 - **Actores:** Usuario alumno, usuario profesor.
 - **Tipo:** Primario, esencial.
 - **Referencias:**
 - **Precondición:** Tener una sesión iniciada.
 - **Postcondición:** -
 - **Autor:** Rogelio Gil García.
 - **Versión:** 1.0.
 - **Propósito:** Resolver una pregunta tipo test.
 - **Resumen:** El usuario entra en la sección preguntas y resuelve una pregunta aleatoria del tipo y tema seleccionados.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Autoevaluación”.	
		2 El sistema muestra una lista los tipos de preguntas disponibles.
3	El usuario pulsa sobre el tipo “Preguntas opción múltiple”.	
		4 El sistema muestra una lista de temas en los que están agrupadas las preguntas.
5	El usuario pulsa sobre el tema del que quiere resolver la pregunta.	
		6 El sistema buscar aleatoriamente una pregunta de este tema y la muestra al usuario con un enunciado y una lista de opciones.
7	El usuario elige una de las opciones como solución a la pregunta y pulsa “Corregir”.	
		8 El sistema muestra una pantalla con la corrección de la pregunta mostrando el enunciado, si es correcta o no y una breve explicación.

Tabla 5.9: CU-6. Resolver pregunta de varias opciones y una respuesta.

- **CU-7.** Resolver pregunta tipo test verdadero/falso.
 - **Actores:** Usuario alumno, usuario profesor.
 - **Tipo:** Primario, esencial.
 - **Referencias:**
 - **Precondición:** Tener una sesión iniciada.
 - **Postcondición:** -
 - **Autor:** Rogelio Gil García.
 - **Versión:** 1.0.
 - **Propósito:** Resolver una pregunta tipo test verdadero/falso.

- **Resumen:** El usuario entra en la sección preguntas y resuelve una pregunta aleatoria del tipo verdadero/falso y tema del seleccionado.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Autoevaluación”.	
		2 El sistema muestra una lista los tipos de preguntas disponibles.
3	El usuario pulsa sobre el tipo “Preguntas verdadero/falso”.	
		4 El sistema muestra una lista de temas en los que están agrupadas las preguntas.
5	El usuario pulsa sobre el tema del que quiere resolver la pregunta.	
		6 El sistema buscar aleatoriamente una pregunta de este tema y la muestra al usuario con un enunciado y las opciones verdadero/falso.
7	El usuario elige una de las opciones como solución a la pregunta y pulsa “Corregir”.	
		8 El sistema muestra una pantalla con la corrección de la pregunta mostrando el enunciado, si es correcta o no y una breve explicación.

Tabla 5.10: CU-7. Resolver pregunta tipo test verdadero/falso.

- **CU-8.** Resolver examen tipo test de preguntas con varias opciones y una respuesta.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener una sesión iniciada.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Resolver un examen tipo test.
- **Resumen:** El usuario resuelve un examen de preguntas con opción múltiple.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Examen”.	
		2 El sistema muestra una lista con los tipos de exámenes disponibles.
3	El usuario pulsa sobre el tipo “Examen preguntas opción múltiple”.	
		4 El sistema muestra un examen de preguntas aleatorias tipo opción múltiple y una sola respuesta. El sistema selecciona 2 preguntas por tema para generar el examen.
5	El usuario resuelve el examen y pulsa el botón “Corregir”.	
		6 El sistema muestra una pantalla con la corrección del examen mostrando la puntuación total, el número de respuestas correctas, incorrectas y sin resolver. Seguidamente y en la misma pantalla se mostrará el examen corregido con la opción seleccionada por el usuario, el resultado y la opción correcta.

Tabla 5.11: CU-8. Resolver pregunta tipo test verdadero/falso.

■ **CU-9.** Resolver examen tipo test de preguntas verdadero/falso.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener una sesión iniciada.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Resolver un examen tipo test.

- **Resumen:** El usuario resuelve un examen de preguntas verdadero/falso.

Curso normal		
	Actor	Sistema
1	El usuario hace click en el botón “Examen”.	
		2 El sistema muestra una lista con los tipos de exámenes disponibles.
3	El usuario pulsa sobre el tipo “Examen preguntas verdadero/falso”.	
		4 El sistema muestra un examen de preguntas aleatorias tipo verdadero/falso. El sistema selecciona 2 preguntas por tema para generar el examen.
5	El usuario resuelve el examen y pulsa el botón “Corregir”.	
		6 El sistema muestra una pantalla con la corrección del examen mostrando la puntuación total, el número de respuestas correctas, incorrectas y sin resolver. Seguidamente y en la misma pantalla se mostrará el examen corregido con la opción seleccionada por el usuario, el resultado y la opción correcta.

Tabla 5.12: CU-9. Resolver pregunta tipo test verdadero/falso.

■ **CU-10.** Registrar estadísticas de preguntas opción múltiple.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:** CU-6 (tabla 5.9)
- **Precondición:** Realizar pregunta opción múltiple.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Registrar las estadísticas de una pregunta resuelta.
- **Resumen:** El usuario resuelve una pregunta y se le asigna el resultado a sus estadísticas.

Curso normal		
	Actor	Sistema
1	El usuario realiza una pregunta descrita en el CU-6 (tabla 5.9)	
		2 El sistema guarda el resultado (correcto o incorrecto) en las estadísticas del usuario.

Tabla 5.13: CU-10. Registrar estadísticas de preguntas opción múltiple.

- **CU-11.** Registrar estadísticas de preguntas verdadero/falso.
 - **Actores:** Usuario alumno, usuario profesor.
 - **Tipo:** Primario, esencial.
 - **Referencias:** CU-7 (tabla 5.10)
 - **Precondición:** Realizar pregunta verdadero/falso.
 - **Postcondición:** -
 - **Autor:** Rogelio Gil García.
 - **Versión:** 1.0.
 - **Propósito:** Registrar las estadísticas de una pregunta resuelta.
 - **Resumen:** El usuario resuelve una pregunta y se le asigna el resultado a sus estadísticas.

Curso normal			
	Actor	Sistema	
1	El usuario realiza una pregunta descrita en el CU-7 (tabla 5.10)		
		2	El sistema guarda el resultado (correcto o incorrecto) en las estadísticas del usuario.

Tabla 5.14: CU-11. Registrar estadísticas de preguntas verdadero/falso.

- **CU-12.** Registrar estadísticas de examen de preguntas opción múltiple.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:** CU-8 (tabla 5.11)
- **Precondición:** Realizar examen preguntas opción múltiple.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Registrar las estadísticas de un examen.
- **Resumen:** El usuario resuelve un examen y se le asigna el resultado a sus estadísticas.

Curso normal		
	Actor	Sistema
1	El usuario realiza un examen de preguntas tipo opción múltiple descrito en el CU-8 (tabla 5.11)	
		2
		El sistema guarda el resultado en las estadísticas del usuario. Se almacena la nota, y si es un examen aprobado o suspenso.

Tabla 5.15: CU-12. Registrar estadísticas de examen de preguntas de opción múltiple.

- **CU-13.** Registrar estadísticas de examen de preguntas verdadero/falso.

- **Actores:** Usuario alumno, usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:** CU-9 (tabla 5.12)
- **Precondición:** Realizar examen preguntas verdadero/falso.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Registrar las estadísticas de un examen.
- **Resumen:** El usuario resuelve un examen y se le asigna el resultado a sus estadísticas.

Curso normal		
	Actor	Sistema
1	El usuario realiza un examen de preguntas tipo verdadero/falso descrito en el CU-9 (tabla 5.12)	
		2
		El sistema guarda el resultado en las estadísticas del usuario. Se almacena la nota, y si es un examen aprobado o suspenso.

Tabla 5.16: CU-13. Registrar estadísticas de examen de preguntas verdadero/falso.

■ **CU-14.** Consulta estadísticas de un alumno.

- **Actores:** Usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada una sesión de usuario con privilegios de profesor.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** El usuario profesor consulta las estadísticas de un usuario alumno.
- **Resumen:** El usuario profesor consultará sus estadísticas, creadas a partir del uso de la aplicación, de un alumno concreto.

Curso normal			
	Actor	Sistema	
1	El usuario profesor hace click en el botón “Perfil”.		
		2	El sistema muestra la pantalla del perfil de usuario y un botón “Perfiles de alumnos”.
3	El usuario profesor hace click en el botón “Perfiles de alumnos”.		
		4	El sistema muestra la lista de alumnos.
5	El usuario profesor hace click en el alumno de el cual quiere revisar las estadísticas.		
		6	El sistema muestra las estadísticas del alumno seleccionado.

Tabla 5.17: CU-14. Consulta estadísticas de alumno.

- **CU-15.** Actualizar datos de conceptos.
 - **Actores:** Usuario profesor.
 - **Tipo:** Primario, esencial.
 - **Referencias:**
 - **Precondición:** Tener iniciada una sesión de usuario con privilegios de profesor.
 - **Postcondición:** -
 - **Autor:** Rogelio Gil García.
 - **Versión:** 1.0.
 - **Propósito:** El usuario profesor actualiza la base de datos de los conceptos.
 - **Resumen:** El usuario profesor actualiza la base de datos de los conceptos con los datos de la hoja de Google Drive.

Curso normal			
Actor		Sistema	
1	El usuario profesor hace click en el botón “Sitio administrador”.		
		2	El sistema muestra una pantalla con las diferentes opciones de actualización de la base de datos.
3	El usuario profesor hace click en el botón “Actualizar conceptos”.		
		4	El sistema accede a la hoja de Google Drive y con los datos leídos actualiza la base de datos de la aplicación.

Tabla 5.18: CU-15. Actualizar datos de conceptos.

■ **CU-16.** Actualizar preguntas con varias opciones.

- **Actores:** Usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada una sesión de usuario con privilegios de profesor.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** El usuario profesor actualiza la base de datos de las preguntas de varias opciones.
- **Resumen:** El usuario profesor actualiza la base de datos de las preguntas de varias opciones con los datos de la hoja de Google Drive.

Curso normal			
Actor		Sistema	
1	El usuario profesor hace click en el botón “Sitio administrador”.		
		2	El sistema muestra una pantalla con las diferentes opciones de actualización de la base de datos.
3	El usuario profesor hace click en el botón “Actualizar preguntas opción múltiple”.		
		4	El sistema accede a la hoja de Google Drive y con los datos leídos actualiza la base de datos de la aplicación.

Tabla 5.19: CU-16. Actualizar preguntas con varias opciones.

■ **CU-17.** Actualizar preguntas verdadero/falso.

- **Actores:** Usuario profesor.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada una sesión de usuario con privilegios de profesor.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** El usuario profesor actualiza la base de datos de las preguntas verdadero/falso.
- **Resumen:** El usuario profesor actualiza la base de datos de las preguntas verdadero/falso con los datos de la hoja de Google Drive.

Curso normal			
Actor		Sistema	
1	El usuario profesor hace click en el botón “Sitio administrador”.		
		2	El sistema muestra una pantalla con las diferentes opciones de actualización de la base de datos.
3	El usuario profesor hace click en el botón “Actualizar preguntas verdadero/falso”.		
		4	El sistema accede a la hoja de Google Drive y con los datos leídos actualiza la base de datos de la aplicación.

Tabla 5.20: CU-17. Actualizar preguntas verdadero/falso.

■ **CU-18.** Dar permisos de profesor a un usuario.

- **Actores:** Usuario administrador.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada sesión como administrador.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Dar permisos de profesor a un usuario.
- **Resumen:** El usuario administrador entra en el panel de administración y otorga permisos de profesor a un usuario.

Curso normal		
	Actor	Sistema
1	El usuario administrador entra en el panel de administración desde la ruta “ https://urlApp/admin ”.	
		2 El sistema muestra una pantalla con el formulario de inicio de sesión de administrador.
3	El administrador lo rellena y lo envía con el botón “Login”	
		4 El sistema valida el formulario, y si es correcto muestra el panel de administración.
5	El usuario administrador entra en el menú usuarios haciendo click en “Users”.	
		2 El sistema muestra una pantalla la lista de usuarios del sistema.
3	El administrador entra en el panel para administrar el usuario haciendo click sobre el nombre de usuario deseado.	
		4 El sistema muestra la pantalla de administración del usuario.
5	En la sección “Permissions” el administrador añadirá los permisos necesarios al ususario.	

Tabla 5.21: CU-18. Dar permisos de profesor.

▪ **CU-19.** Eliminar permisos de profesor a un usuario.

- **Actores:** Usuario administrador.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada sesión como administrador.
- **Postcondición:** -

- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Eliminar permisos de profesor a un usuario.
- **Resumen:** El usuario administrador entra en el panel de administración y elimina los permisos de profesor a un usuario.

Curso normal			
	Actor		Sistema
1	El usuario administrador entra en el panel de administración desde la ruta “ https://urlApp/admin ”.		
		2	El sistema muestra una pantalla con el formulario de inicio de sesión de administrador.
3	El administrador lo rellena y lo envía con el botón “Login”		
		4	El sistema valida el formulario, y si es correcto muestra el panel de administración.
5	El usuario administrador entra en el menú usuarios haciendo click en “Users”.		
		2	El sistema muestra una pantalla la lista de usuarios del sistema.
3	El administrador entra en el panel para administrar el usuario haciendo click sobre el nombre de usuario deseado.		
		4	El sistema muestra la pantalla de administración del usuario.
5	En la sección “Permissions” el administrador eliminará los permisos necesarios al ususario.		

Tabla 5.22: CU-19. Eliminar permisos de profesor.

- **CU-20.** Eliminar usuario.

- **Actores:** Usuario administrador.
- **Tipo:** Primario, esencial.
- **Referencias:**
- **Precondición:** Tener iniciada sesión como administrador.
- **Postcondición:** -
- **Autor:** Rogelio Gil García.
- **Versión:** 1.0.
- **Propósito:** Eliminar un usuario.
- **Resumen:** El usuario administrador entra en el panel de administración y elimina un usuario

Curso normal		
	Actor	Sistema
1	El usuario administrador entra en el panel de administración desde la ruta “ https://urlApp/admin ”.	
		2 El sistema muestra una pantalla con el formulario de inicio de sesión de administrador.
3	El administrador lo rellena y lo envía con el botón “Login”	
		4 El sistema valida el formulario, y si es correcto muestra el panel de administración.
5	El usuario administrador entra en el menú usuarios haciendo click en “Users”.	
		2 El sistema muestra una pantalla la lista de usuarios del sistema.
3	El administrador entra en el panel para administrar el usuario haciendo click sobre el nombre de usuario deseado.	
		4 El sistema muestra la pantalla de administración del usuario.
5	Al final de la pantalla encontramos el botón “Delete”, lo pulsamos y confirmamos	

Tabla 5.23: CU-20. Eliminar usuario.

Diagrama de casos de uso.

Para dar mas claridad al diagrama de casos de uso vamos a descomponerlo por actores. Tendremos en cuenta que todos los casos de uso pertenecen a la misma aplicación iOrg2.0.

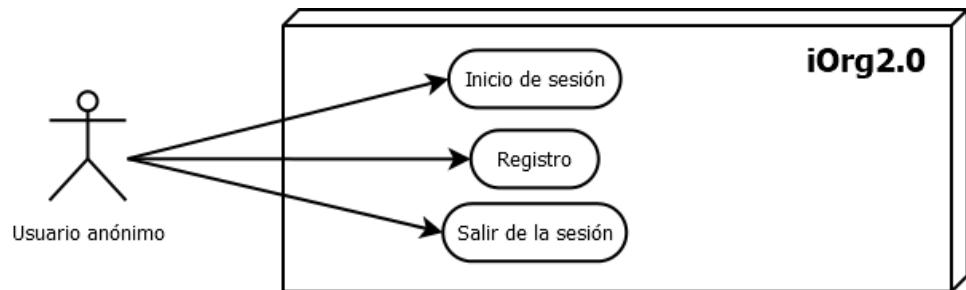


Figura 5.1: Diagrama de casos de uso actor usuario anónimo.

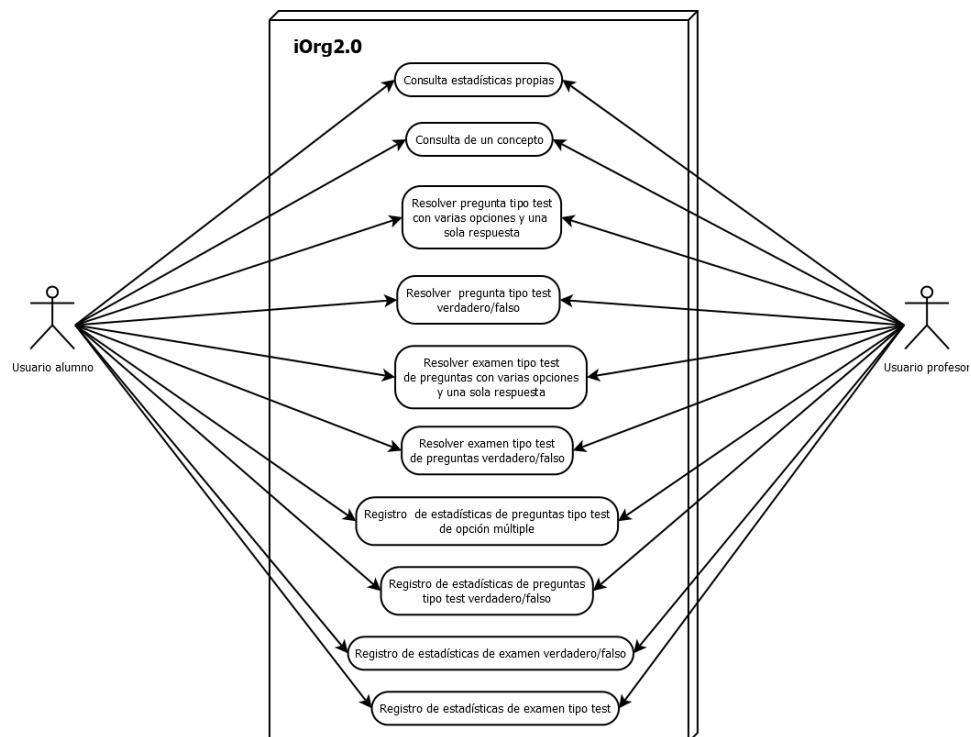


Figura 5.2: Diagrama de casos de uso actores alumno y profesor.

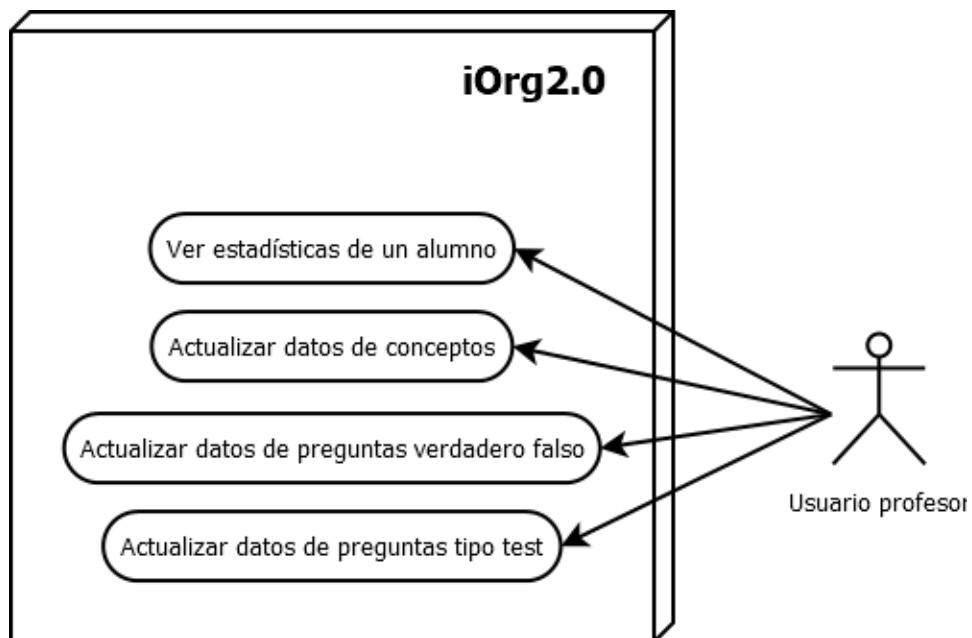


Figura 5.3: Diagrama de casos de uso actor profesor.

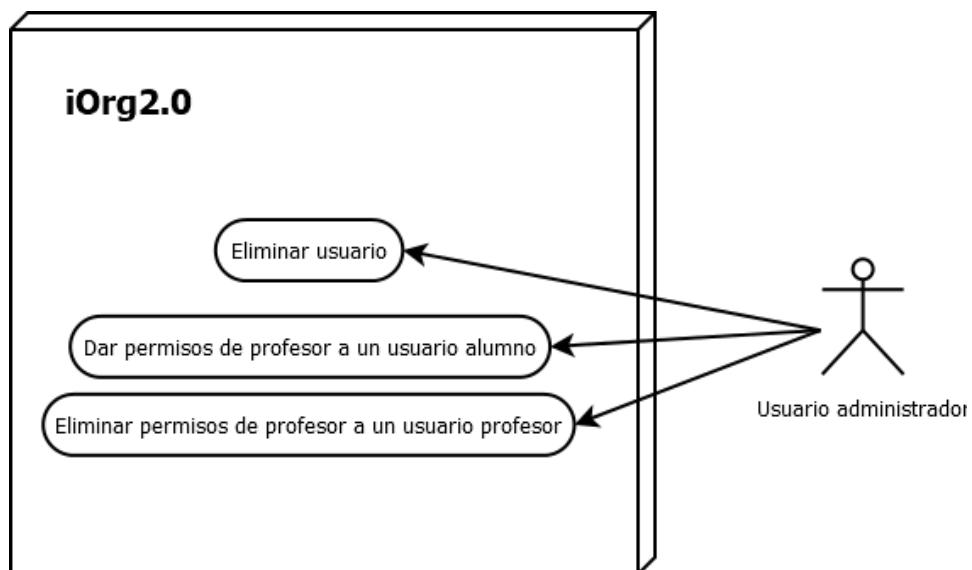


Figura 5.4: Diagrama de casos de uso actor administrador.

Diagrama de navegación

Aunque las secciones de esta aplicación no se preveen extensas en profundidad no está de más tener un diagrama donde mostrar la navegación:

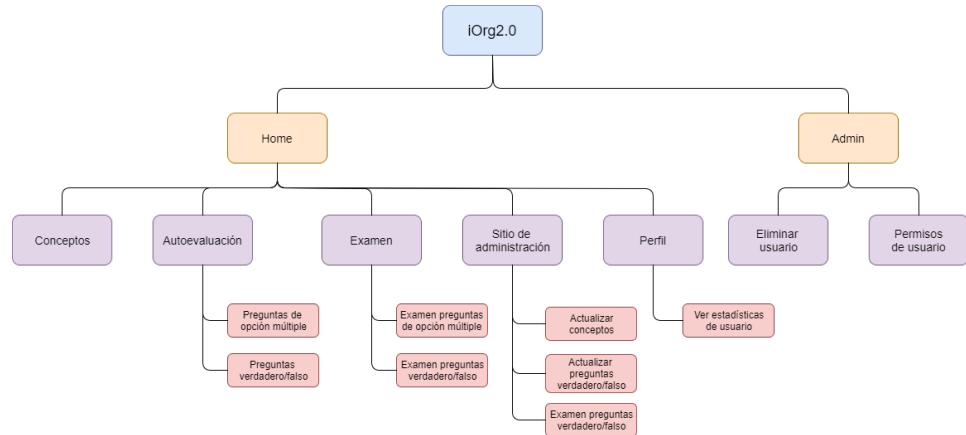


Figura 5.5: Diagrama de navegación

Capítulo 6

Diseño e Implementación

Modelado de la base de datos

En un primer diagrama para la base de datos se exponen las entidades y sus relaciones. Atendiendo a los requisitos de la aplicación se plantea un diseño sencillo y directo en cuanto a las entidades y relaciones. En un primer esquema **entidad-relación** encontramos:

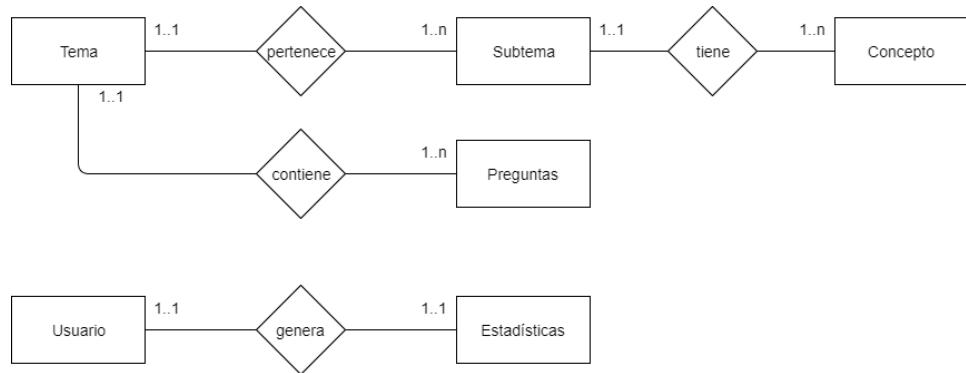


Figura 6.1: Entidad relación

Analizando el diagrama vemos que las únicas relaciones que existen son de cardinalidad 1:N y 1:1. Ambas las podemos simplificar eliminando la relación y añadiendo la clave primaria de la entidad con cardinalidad 1 en los campos de la entidad con cardinalidad N. Quedan así resumidas las tablas de la base de datos a únicamente las entidades:

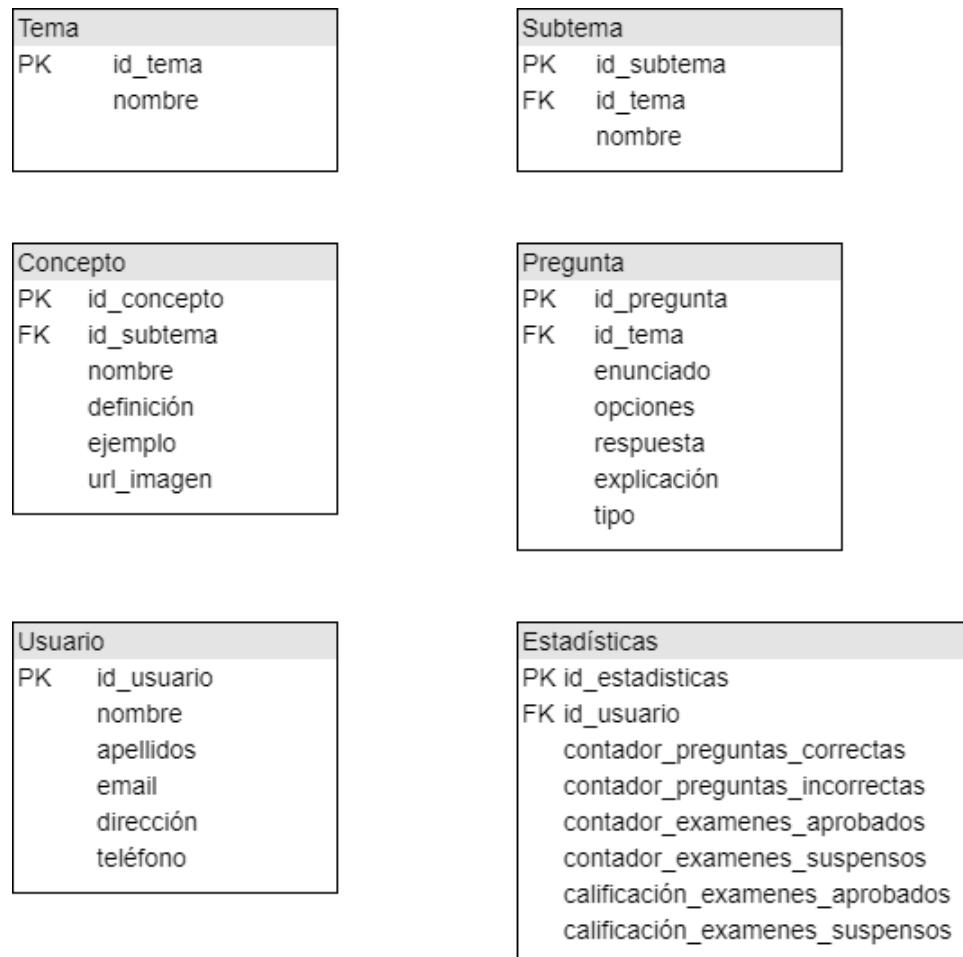


Figura 6.2: Tablas del modelo entidad relación

Diseño de las interfaces

A menudo y varias veces al día nos encontramos con cientos de interfaces[10] con las que interactuamos, muchas de estas están tan asimiladas que ni las apreciamos. Las interfaces no son un objetivo para el usuario, si no que son el medio para llegar al él, siendo estas que no apreciamos las mejores interfaces. Una interfaz[4] no debe entorpecer el camino del usuario hasta su objetivo, debe ser sencilla, rápida de asimilar y directa.

Aquí mostramos algunos bocetos sencillos y su consiguiente implementación como interfaz de usuario en la aplicación:

Bocetos

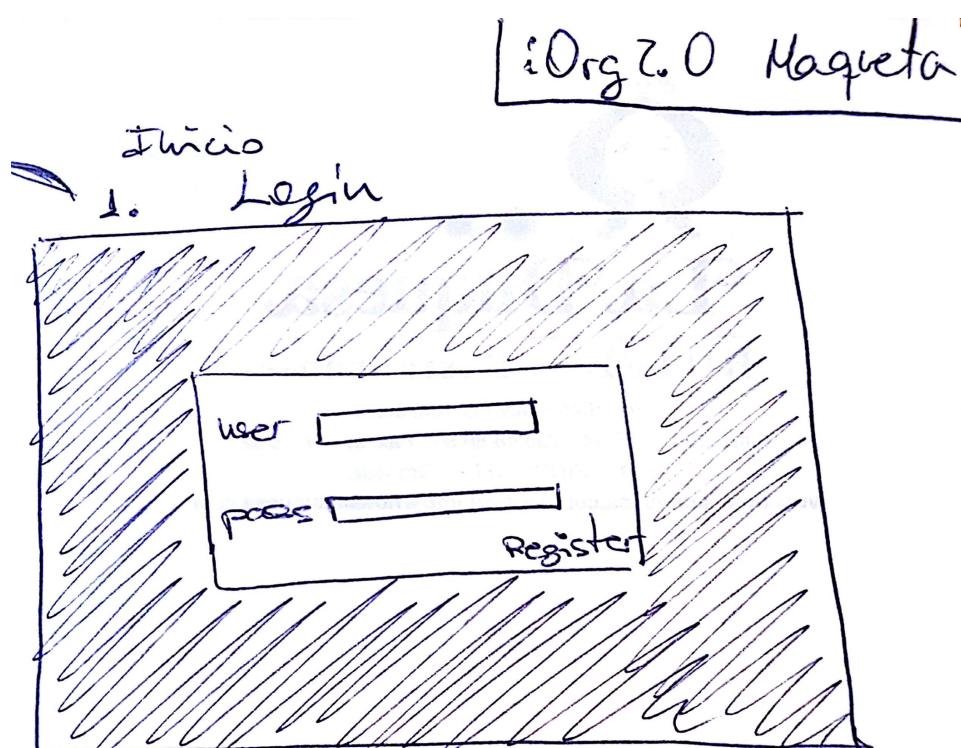


Figura 6.3: Boceto interfaz inicio de sesión.

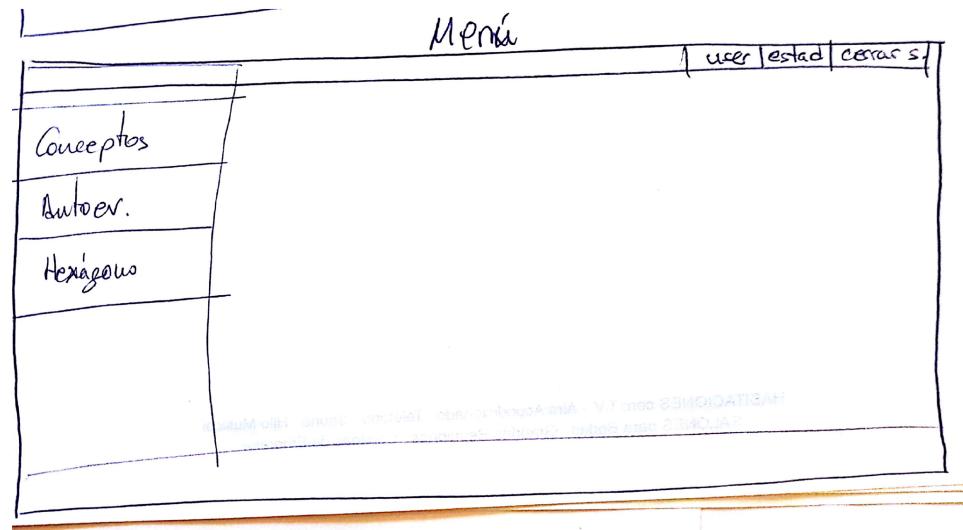


Figura 6.4: Boceto interfaz menú lateral.

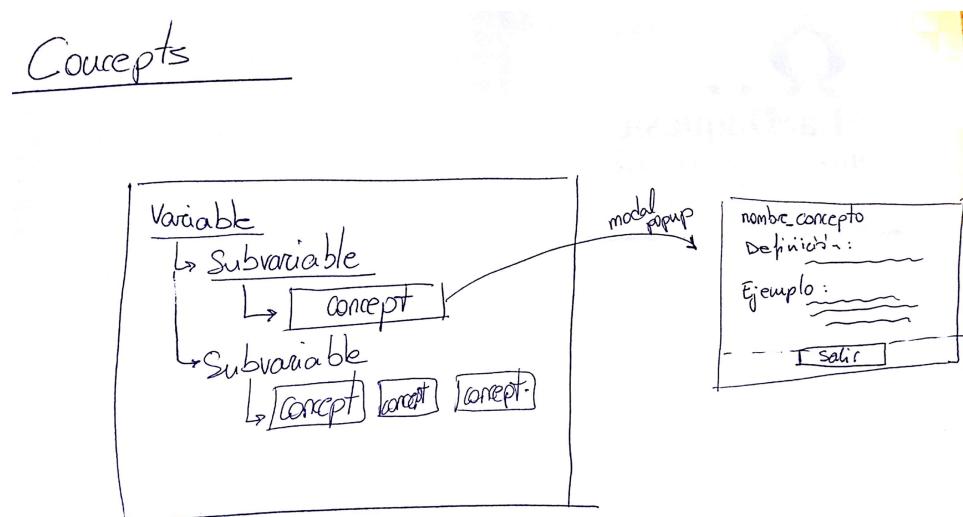


Figura 6.5: Boceto interfaz conceptos.

Interfaz

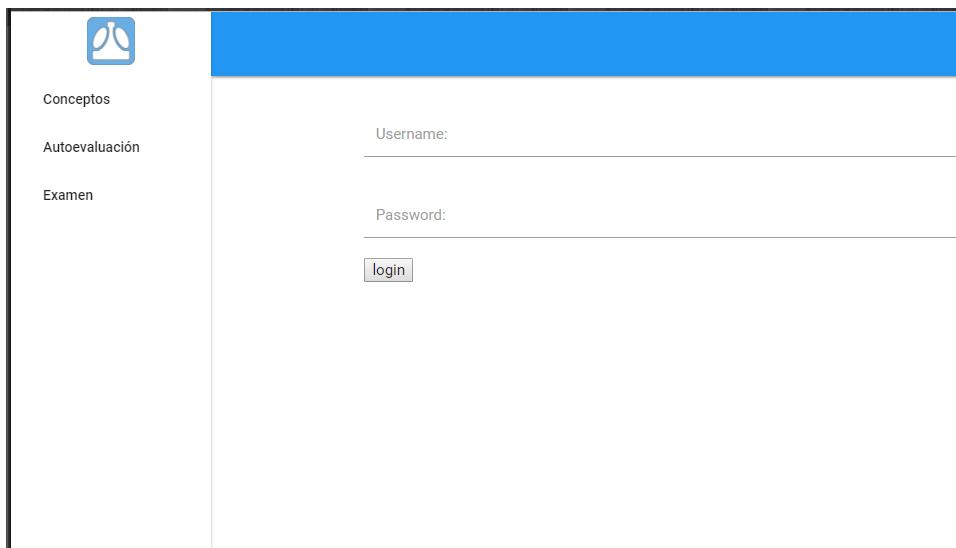


Figura 6.6: Interfaz de usuario inicio de sesión.

A screenshot of a user interface for concepts. On the left is a vertical sidebar with three items: 'Conceptos', 'Autoevaluación', and 'Examen'. The main area has a blue header bar with a logo icon and navigation links: 'Admin site', 'Profile', 'Logout', and a gear icon. Below the header is a large title 'Temas'. Underneath the title is a list of topics: 'Partes de la organización', 'Mecanismos de coordinación', 'La organización como sistema de flujos', 'Variables del diseño de puestos', and 'Variables del diseño de la estructura'. At the bottom is a section titled 'Agrupación de unidades' containing three teal-colored buttons: 'BASES DE AGRUPACIÓN', 'AGRUPACIÓN FUNCIONAL', and 'AGRUPACIÓN POR MERCADOS'.

Figura 6.7: Interfaz de usuario conceptos 1/2.

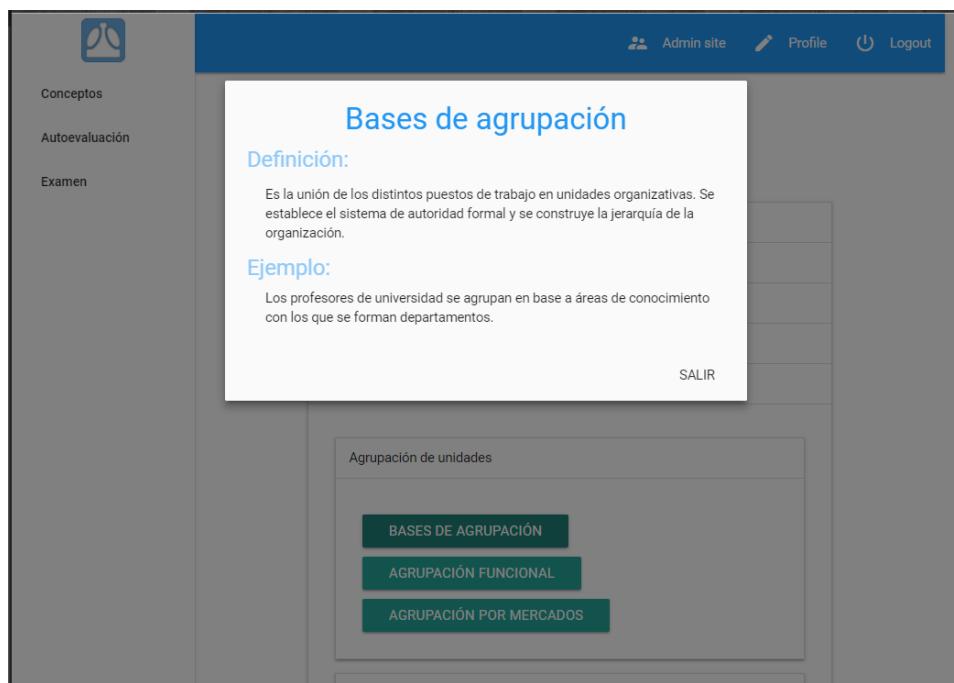


Figura 6.8: Interfaz de usuario conceptos 2/2.

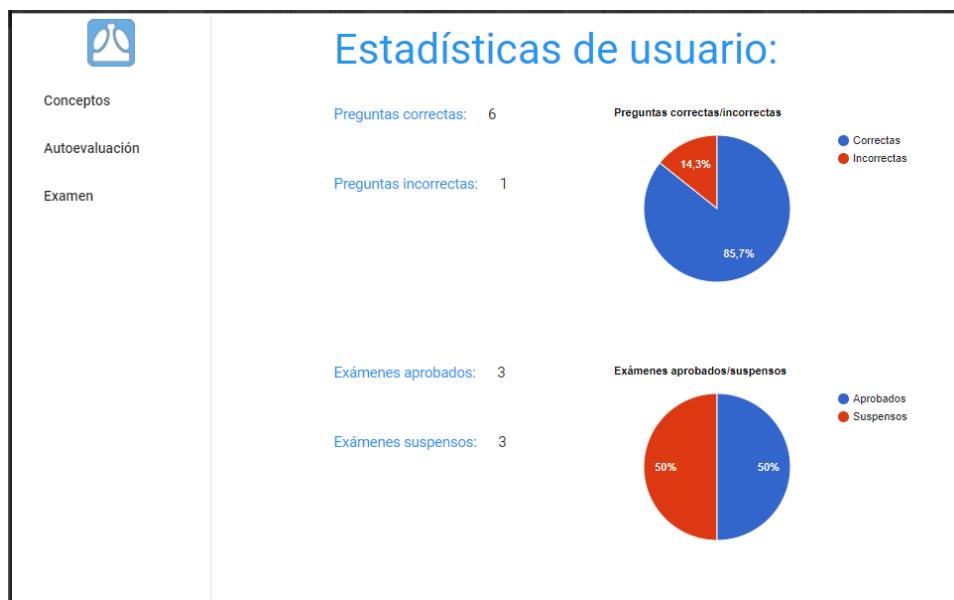


Figura 6.9: Interfaz de usuario estadísticas de usuario.

Implementación

Para una correcta y efectiva implementación destacaremos una vez más la importancia del proceso de análisis y diseño. Son en estas fases donde se asienta la implementación y si las anteriores se hicieron correcta y detalladamente esta fase se convierte en una mera interpretación o traducción de los análisis, diagramas y diseños anteriores.

Para la fase de implementación es importante conocer las herramientas de las que podemos hacer uso. Nuestra aplicación se establece en un entorno web y ya hicimos un pequeño sondeo de las tecnologías que podíamos utilizar, ahora vamos a hablar brevemente de cómo hemos usado las herramientas y lenguajes en la implementación.

Vamos a distinguir entre lenguajes y herramientas usadas en el lado del cliente y lenguajes y herramientas usados en el lado del servidor:

En el lado del servidor

En el lado del servidor tenemos Python como lenguaje y Django como framework.

- **Django:** Framework web.
- **Python:** Lenguaje en el lado del servidor.
- **SQLLite:** Sistema de gestión de base de datos.

Como ya explicamos unos capítulos atrás esta es nuestra elección para la parte del servidor. Una vez realizada la implementación podemos hablar de las experiencias con estas herramientas que ahora si, en mayor o menor medida, conocemos.

Lo primero que vamos a destacar es Django como framework web. Si bien es cierto que durante el Grado en Ingeniería Informática hemos trabajado con Django, lo hicimos dando unas ligeras pinceladas. En el desarrollo de este proyecto hemos podido comprobar la potencia de esta herramienta que abarca un amplio ámbito de componentes, desde la gestión de las rutas de nuestra aplicación pasando por la integración de los modelos y la base de datos hasta la gestión del controlador y las plantillas.

Con Django hemos conseguido abstraernos a un nivel un poco mayor de la gestión de la base de datos. Django trabaja con modelos, donde los desarrolladores definen sus “entidades” y Django se encarga de gestionarlas en la base de datos (que podemos definir en la configuración de Django) y

en nuestro caso es SQLite. Separar estos modelos nos permite, por ejemplo, cambiar la base de datos en cualquier momento y haciendo las migraciones necesarias, al hacer esto los modelos definidos en Django no sufrirán ninguna modificación.

De manera similar diferenciar entre los modelos, vistas y plantillas nos permite realizar un cambio en cualquiera de estos sistemas sin tener que cambiar sustancialmente el otro. Esto ha sido especialmente útil en este proyecto al cambiar la conexión de los modelos a las hojas de Google Drive. En una primera prueba de conexión entre Django y Google Drive, la lectura de varias celdas se hacía algo lenta. A medida que maduraba la implementación dimos con una forma de recorrer estos datos de manera más eficiente. La gestión de los modelos y el cambio que esto supuso no interfirió con las vistas ni las plantillas ya creadas lo que facilitó esta tarea.

También vamos a destacar que, a nuestro parecer, la curva de aprendizaje de Django es algo complicada. Se necesita tiempo para asimilar los diferentes conceptos que trabaja y los diferentes paquetes que podemos utilizar, cosa que creemos normal en un framework de estas dimensiones. A cambio, la potencia que ofrece este framework es más que notable.

En el lado del cliente

- **Html5:** Lenguaje de marcado para las plantillas.
- **CSS3:** Lenguaje que define los estilos de las plantillas.
- **LESS:** Precompilador CSS.
- **MaterializeCss:** Framework CSS “responsive”¹
- **jQuery:** Biblioteca multiplataforma de JavaScript.

Para mostrar la información en una aplicación web podríamos decir que lo básico que necesitamos es un lenguaje de marcado (Html5) y un lenguaje con el que definir sus estilos (CSS3). En nuestra aplicación, al igual que nos servimos de paquetes y frameworks en la parte del servidor para no rehacer funcionalidades a bajo nivel, en la parte del cliente nos basaremos en estilos ya predefinidos que nos ayudarán a dar un aspecto a la aplicación. Para esto hemos elegido MaterializeCss que nos permite agrupar los tamaños y colores de los elementos de la aplicación, así como facilitarnos funcionalidades como expandir y contraer listas, mostrar u ocultar el menú lateral en resoluciones

¹Que se adapta a varias resoluciones de pantalla

de pantalla pequeñas, mostrar ventanas modales y crear la vista de formularios. Para esta interacción y la gestión de estos eventos se hace uso de jQuery.

Aunque relacionado con la parte del cliente no se especificaron muchos requisitos podemos hacer referencia al RNF-3 (5.2) donde nos ha sido muy útil MaterializeCss ya que los elementos usados de este framework se ajustan a los diferentes tamaños de pantalla.

Otras herramientas

- **PyCharm:** En general el uso de PyCharm como entorno de programación lo podemos calificar como positivo. Destacamos la ayuda que ofrece el entorno a la sintaxis del lenguaje, que al ser nuevo nos costó un poco al inicio de la implementación, o a la gestión y uso de las bibliotecas de Django por su extensión. Señalar algunos puntos negativos que creemos que son comunes a cualquier entorno de programación y es la sensación de no saber qué está pasando en algunos momentos al incluir paquetes externos. Esto se produce por la automatización casi total de las funciones para agregar paquetes. Como caso particular al incluir el paquete de LESS el sistema no lo reconocía, tras varios intentos desinstalando e instalando con PyCharm se decidió hacerlo manualmente desde la terminal.
- **Git y Github:** Ya describimos anteriormente Git pero creemos interesante contar las conclusiones sacadas de Git y Pycharm.

Ya en el Grado en Ingeniería Informática hemos hecho uso de Git como herramienta de control de versiones, pero siempre desde la terminal. La sincronización de Git y Pycharm resultó útil en cuanto a la visualización de ficheros incluidos en el repositorio y los que aún quedaban por incluir o por subir al repositorio. Pycharm los cataloga con color verde (los actualizados en el repositorio), rojos (los que no están actualizados) y grises (los que no están incluidos). De esta forma en la barra lateral del entorno puedes ver rápidamente cómo están tus ficheros en el repositorio.

Por otro lado, la herramienta gráfica para la gestión del repositorio no resulta intuitiva y útil, quizás por ser usuarios asiduos a Git desde la terminal, pareciendo que no controlas completamente la actualización del repositorio. En este proyecto no se ha usado Git integrado con Pycharm y la gestión se ha realizado desde la terminal.

GitHub es una plataforma donde podemos conectar nuestros repositorios de Git de forma pública o privada. Los repositorios públicos forman una comunidad con código enorme de diferentes lenguajes y plataformas y con diferentes licencias de uso.

En el proyecto hemos conectado Github con nuestros dos repositorios, uno se destina al código fuente y otro para el desarrollo en LaTex de esta misma documentación. El por qué de separar estos repositorios lo desarrollaremos más adelante.

- **Código fuente:** <https://github.com/rogegg/iOrg2>
- **Documentación:** https://github.com/rogegg/iOrg2_docu

■ **Heroku:** Heroku es una plataforma como servicio de computación en

la Nube que soporta distintos lenguajes de programación. Esta plataforma nos va a permitir desplegar nuestro servidor en la nube y acercarnos al entorno final de producción de la aplicación. Llegados a este punto vemos interesante explicar cómo hemos hecho el despliegue y cómo hemos subido nuestra aplicación a la plataforma, vamos a extenderlo en el próximo capítulo.

Capítulo 7

Despliegue

Durante el Grado en Ingeniería Informática y más concretamente en la asignatura Infraestructura Virtual ya habíamos trabajado desplegando proyectos en alguna plataforma en la nube, concretamente con OpenShift. En esta plataforma echamos en falta un poco de ayuda en las herramientas de la misma para desplegar aplicaciones. Si que es verdad que para entornos sencillos quizá es una buena solución pero en este proyecto pensábamos que no era la mejor plataforma a elegir y buscamos más opciones.

Ya que conocemos nuestro entorno buscamos alguna plataforma que nos ofreciera una manera “asistida” y rápida de actualizar y desplegar nuestro proyecto. Tendremos en cuenta una serie de puntos:

- **Herramientas de la plataforma.**
- **El entorno de nuestro código.**
- **Cambios necesarios para el despliegue.**

Gracias a las herramientas de Heroku subir una aplicación Django no se hace una tarea ardua, aunque sí hemos encontrado algunas complicaciones que iremos describiendo.

Herramientas de Heroku

Para desplegar una aplicación Python en Heroku, la plataforma en su página oficial nos ofrece su CLI¹ para descargarlo e instalarlo localmente (en nuestro caso en Ubuntu). Una vez instalado podremos usar el comando “heroku” en nuestra terminal y lo usaremos para conectarnos (obviamente que necesitamos una cuenta gratuita en Heroku):

¹Command Line Interface.

```
1 $heroku login
```

```
Terminal Archivo Editar Ver Buscar Terminal Ayuda
$ heroku login
Enter your Heroku credentials:
Email: iorg...@...
Password: *****
Enter your Heroku credentials:
Email: iorg...@...
Password: *****
Logged in as iorg...
$ clear
```

Figura 7.1: Login a Heroku.

Además podemos habilitar una conexión SSH con Heroku de la siguiente manera:

```
Terminal Archivo Editar Ver Buscar Terminal Ayuda
$ ls ~/.ssh/
id_rsa id_rsa.pub known_hosts
$ heroku keys:add
Found an SSH public key at /home/roge/.ssh/id_rsa.pub
? Would you like to upload it to Heroku? Yes
Uploading /home/roge/.ssh/id_rsa.pub SSH key... done
```

Figura 7.2: Conexión ssh a Heroku.

Tras introducir nuestro usuario y contraseña, estamos conectados a Heroku desde nuestra terminal.

Entorno de nuestro código

Es aquí donde agradeceremos haber utilizado las recomendaciones sobre entornos virtuales y repositorios. Recordemos que nuestros repositorios están separados, en uno el código fuente y en otro el código de nuestra documentación.

Entorno virtual

Tener un entorno virtual bien configurado lo definiríamos como vital en este paso. Tener un entorno desorganizado complicaría mucho las tareas de sincronizar nuestro proyecto con la plataforma Heroku. Las virtudes del entorno virtual son muchas, pero destacamos las más importantes que hemos advertido durante el despliegue:

- **Especificación correcta y única de versiones:**

En nuestro sistema podremos tener instalada cualquier versión de Django o Python, pero en el entorno virtual del proyecto sólo tendremos la necesaria. Podemos probar otras versiones del lenguaje o el framework fácilmente creando otro entorno virtual con las versiones deseadas.

- **Dependencias bien definidas:**

Para desplegar la aplicación deberemos especificar las dependencias a Heroku, tener el entorno virtual con las dependencias mínimas y suficientes para la aplicación es esencial.

Como nota señalar que, durante el despliegue de Heroku, tuvimos problemas con las dependencias pues, en algún punto del proyecto que no pudimos detectar, todo indica a que instalamos un paquete de forma global desde nuestro entorno virtual. Por esto las dependencias del proyecto aumentaron, pues el entorno necesitaba algunos paquetes generales del sistema operativo. Esto lo descubrimos al subir la aplicación a Heroku y ver que era incapaz de cubrir estas dependencias y fue entonces que decidimos listarlas y descubrimos que no eran correctas.

Para solucionarlas intentamos desinstalar estas dependencias, pero el sistema parecía no funcionar establemente. Es en este punto cuando vimos la gran virtud de un entorno virtual pues no tuvimos más que crear otro entorno, instalar las dependencias antiguas correctamente y arrancar la aplicación. Sin tener un entorno virtual, especificar las dependencias únicas de nuestra aplicación a Heroku hubiese sido prácticamente imposible.

Repositorio para el proyecto

Ya estamos conectados a Heroku, nos encontramos en el directorio de nuestro repositorio y dentro de nuestro entorno virtual, ¿cómo subimos nuestro código a Heroku? Aquí es donde Heroku nos pone las cosas sencillas. Con esta plataforma podemos crear un repositorio en el mismo directorio que nuestro repositorio de Git:

```

1 $heroku create
2 $git push heroku master

```

Cambios necesarios para el despliegue

Con lo descrito anteriormente ya tenemos subido nuestra aplicación a Heroku pero tendremos nos falta un paso importante a tener en cuenta. Hemos subido un repositorio, con un código fuente, pero Heroku no ejecuta automáticamente el servidor al subirlo, para ello tendremos que añadir un archivo al directorio raíz nombrado “**Procfile**” y que va a contener un “dyno”² con la instrucción para arrancar el servidor. En nuestro caso:

```

1 web: gunicorn iOrg2.wsgi --log-file -

```

La plataforma ya sabe cómo arrancar el servidor pero no conoce las **dependencias**. Las dependencias se las especificaremos en un archivo “requirements.txt”. Para crear este fichero no tenemos más que, dentro de nuestro entorno virtual, ejecutar una orden para listar las dependencias:

```

1 $pip freeze requirements.txt

```

Nota: en nuestro caso una de las dependencias no era capaz Heroku de satisfacerla, en concreto “pkg-resources==0.0.0”. Esta dependencia no es necesaria para lanzar el servidor. Para eliminarla cada vez que actualicemos las dependencias hemos creado un pequeño script bash:

```

1 #!/bin/bash
2 pip freeze | grep -v "pkg-resources" > requirements.txt

```

Algunas configuraciones locales no funcionan en un servidor en línea por lo que habrá que retocarlas a la hora de desplegar el servidor como es el caso de los **ficheros estáticos** o “staticfiles”, estos son las hojas de estilos, imágenes cargadas en el html, iconos, etc. Ya que no estábamos familiarizados con estos ficheros al desplegarlos en un servidor, este punto ha dado un poco de trabajo hasta solucionar cómo cargarlos en Heroku. Para ello hemos utilizado el paquete “WhiteNoise”.

Sólo fue necesario añadir WhiteNoise en los paquetes de Django y habilitar el paquete en el fichero de configuración general:

²Basado en contenedores de Linux, un dyno no es más que un contenedor que ejecuta la instrucción que le indiquemos.

```
1 MIDDLEWARE_CLASSES = [
2     'whitenoise.middleware.WhiteNoiseMiddleware',
3     ...
4 ]
```

También configuraremos los “**hosts**” **permitidos**, comprobando que no esté únicamente nuestra dirección local y añadiendo la dirección del servidor:

```
1 ALLOWED_HOSTS = ['localhost', '_dirección_del_servidor']
```

Ahora sí, tenemos definidas las dependencias, configurados los ficherlos estáticos, permitido el acceso a la nueva dirección y definidas las instrucciones para arrancar el servidor, ya podemos actualizar nuestro repositorio de Heroku y lanzar el servidor indicando a Heroku que ejecute la línea del fichero “Procfile”:

```
1 $ heroku ps:scale web=1
```

Ya tenemos nuestra aplicación desplegada en Heroku.

Señalar que podemos lanzar la consola de python desde Heroku si necesitamos realizar cualquier cambio sobre la plataforma, sólo tenemos que ejecutar:

```
1 $ heroku run python manage.py shell
```

Como podemos ver, para el despliegue de Heroku hemos necesitado realizar algún cambio sobre la configuración local, pero con una correcta gestión del entorno virtual y de los repositorios conseguimos desplegar de forma fácil y eficaz el servidor de testeo.

Capítulo 8

Conclusiones y vías futuras

(Desarrollar conclusiones y vías futuras)

Bibliografía

Bibliografía

- [1] Byron Francis, 2016, “Python : The Complete Beginners Guide (The Black Book)”.
- [2] Sergio Infante Montero, 2012, “Curso Django. El framework para detaillistas con deadlines”, Maestros del Web; Edición: 1.
- [3] Ian Sommerville, 2011, “Ingeniería de software”, Pearson Educación (9^a ed.) .
- [4] Juan Manuel Carraro y Yanina Duarte, 2015, “Diseño de experiencia de usuario (UX)”, Editorial Autores de Argentina(1^a ed.).

Recursos Web

- [5] “Web frameworks”, Wiki Python, Fecha de último acceso: 12/01/2017.
<https://wiki.python.org/moin/WebFrameworks>
- [6] “Django”, Fecha de último acceso: 28/06/2017.
<https://www.djangoproject.com/>
- [7] “TurboGears”, Fecha de último acceso: 25/01/2017.
<http://turbogears.org/>
- [8] “Books and tutorials for Django”, Full Stack Python, Fecha de último acceso: 12/02/2017.
<https://www.fullstackpython.com/django.html>
- [9] “Implementación del modelo integral colaborativo”, José Luis Cendejas Valdés, Fecha de último acceso: 21/02/2017.
<http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm>

- [10] “Importancia de la interfaz de usuario”, ”Spais”, 2007, Fecha de último acceso: 11/04/2017.
<http://spaisdaniela-imdtp1.blogspot.com.es/2007/04/importancia-de-la-interfaz-de-usuario.html>

Otros recursos web

- [11] “Herramienta de capturas de pantalla para Ubuntu”, Shutter, Fecha de último acceso: 15/02/2017.
<http://shutter-project.org/>

