Aplicación web para facilitar la corrección de prácticas de programación con integración en campus virtual

Scipio: Sistema de Corrección Inter-pares Para códIgO

TRABAJO FIN DE GRADO
Curso 2018–2019

Autor
Jose Luis Sánchez García

Director
Manuel Freire Moran

Facultad de Informática
Universidad Complutense de Madrid

Aplicación web para facilitar la corrección de prácticas de programación con integración en campus virtual

Scipio: Sistema de Corrección Inter-pares Para códIgO

Trabajo de Fin de Grado en Ingeniería Software
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Jose Luis Sánchez García

Director
Manuel Freire Moran

Convocatoria: *Septiembre 2019*
Calificación: *Nota*

Facultad de Informática
Universidad Complutense de Madrid

20 de septiembre de 2019

Autorización de difusión

El abajo firmante, matriculado en el Máster en Ingeniería en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Scipio", realizado durante el curso académico 2018-19 bajo la dirección de Manuel Freire Morán en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Jose Luis Sánchez García

20 de septiembre de 2019

Dedicatoria

Esta trabajo de fin de grado esta dedicado a todos lo que me han apoyado en este mundo apasionante que es la programación, desde mis primeros pasos en un aula del instituto cuando aun se guardaban los archivos en disquetes, hasta hoy en día.

Agradecimientos

Esto es lo que ocurre con la gente que piensa que odia los ordenadores. Lo que realmente odia es a los malditos programadores

Larry Niven – escritor de ciencia ficción

Quiero agradecer especialmente este trabajo a todos lo que me han apoyado durante su creación, a mi tutor Manuel Freire Moran por su paciencia en las tutorías cuando llegaba con la documentación sin hacer, a todos los profesores que durante la carrera me han enseñado todo lo necesario para llegar a este momento. y a mis padres por apoyarme cuando tome la decisión de querer especializarme en informática en vez de hacer el bachillerato.

Resumen

Este Trabajo de Fin de Grado se basa en la creación de una aplicación para la corrección de practicas de programación por parte de los profesores y entre los propios alumnos (ínter-pares), así como la generación de informes de dichas correcciones para el feedback.

Palabras clave

Prácticas de programación, Corrección de prácticas, Corrección ínter-pares, Aplicación web, CodeMirror, Angular, NodeJS.

Abstract

This final degree thesis project (TFG) is based on the creation of an application for the review and grading of programming exercises by teachers and among students (peer review), as well as the generation of reports of these corrections for feedback.

Keywords

Programming exercises, Code review and grading, Peer review, Web application, CodeMirror, Angular, NodeJS

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Las computadoras son como los dioses del Viejo Testamento; un montón de reglas y sin piedad."
— Joseph John Campbell (1904-1987)

1.1. Motivación

Scipio esta motivada por la necesidad de mejorar la comunicación entre el alumnado y los profesores a la hora de la corrección de las practicas de programación. Proporciona herramientas para facilitar la corrección del código por parte de los profesores, pudiendo estos indicar los fallos e incluir la corrección del mismo, y posteriormente enviar un informe a los alumnos implicados en dicha practica para que estos tengan constancia de los mismos en su contexto. Es decir, los alumnos podrán ver la correcciones (una vez finalizadas), y efectuar comentarios sobre ellas.

Con las herramientas actuales, mayoritariamente el campus virtual, hay dificultades para comunicar correcciones de practicas de programación entre profesores y alumnos: ninguno puede ver o comentar de forma sencilla los comentarios o errores en un fichero y linea concretos sin perder mucho tiempo buscándolo de forma manual. Tampoco proporciona ninguna herramienta que permita resumir en un solo documento todos los comentarios y correcciones de forma sencilla y legible, si el profesor quisiera hacer esto tendría que perder mucho tiempo seleccionando código, ademas si quisiera comentarlo en el aula a los alumnos no podría centrarse en los problemas específicos de cada alumno, teniendo que quedarse en los problemas más generales que afecten a un grupo amplio de alumnos.

Durante mi paso por la Facultad de Informática de la UCM, en la asignaturas que tienen un fuerte componente de programación tales como Fundamentos de la Programación, Tecnología de la Programación o en cursos superiores como Aplicaciones Web, Estructura de Datos y Algoritmos o Téc-

nicas Algorítmicas en Ingeniería del Software, he notado que la comunicación entre profesores y alumnos o entre los propios alumnos a la hora de ver los fallos cometidos en un código es compleja y requiere de un tiempo que, debido a lo apretado del calendario, en general no es posible destinar a esta tarea.

Por ultimo una gran motivación por mi parte fue el mejorar mis conocimientos y aprender nuevas metodologías, lenguajes y tecnologías actualmente en boga en los entornos reales de las empresas.

Con las motivaciones anteriormente mencionadas nace Scipio, con el espíritu de crear una plataforma que cubra estas necesidades y mejore ademas de agilizar esta tarea tan importante para el aprendizaje de un lenguaje de programación.

1.2. Objetivos

1.2.1. Objetivos principales

A continuación se detallan los objetivos que nos proponemos cumplir.

Corrección de entregas de prácticas de código

Los profesores son unos de principales actores de nuestra aplicación y una de sus funcionalidades es la de corregir las entregas hechas por los alumnos. Por ello mediante el uso de la aplicación los profesores podrán corregir el código de las practicas entregadas de los alumnos, pudiendo realizar comentarios y correcciones de código sobre la misma que luego mediante un informe generado el alumno podrá ver las mismas resumidas en un mismo documento. El profesor también podrá activar la función de corrección ínter-pares descrita en los objetivos avanzados.

Integración con Campus Virtual

La integración con el campus virtual permite a los profesores una mayor agilidad a la hora de poder acceder rápidamente al entorno de corrección.

Por ello el profesor podrá descargar las practicas entregadas en el campus virtual moodle (Martin Dougiamas, 2019) y cargarlas en la aplicación de forma rápida y sencilla. De forma interna se distribuyen entre los equipos (uno o varios alumnos) mediante un sistema de código en el nombre del fichero, usando el identificador del grupo al que pertenece la entrega.

Alumnos pueden ver sus correcciones

El feedback entre profesores y alumnos es parte básica de nuestra aplicación. Por ello os alumnos podrán ver los comentarios y correcciones hechas

por el profesor, ademas del informe generado. Con esto se creara un sistema de interacción entre el profesor y el alumnado, mejorando la comunicación entre ambos.

Capacidad por parte de los profesores de gestionar las asignaturas, tareas y entregas

Los profesores podrán gestionar sus asignaturas partiendo de una asignatura base común y poder administrar los equipos, tareas y entregas de la misma de forma sencilla.

Capacidad por parte de un administrador o administradores de gestionar la plataforma de manera sencilla

El/los administradores podrán realizar tareas de mantenimiento y actualización de los elementos que componen la plataforma. Sus tareas principales serán mantener la base común de asignaturas y la gestión de los perfiles de profesor.

1.2.2. Objetivos avanzados

Una vez alcanzados los objetivos anteriores, planteamos también los siguientes objetivos avanzados.

Revisión entre pares de alumnos

Los alumnos mediante un sistema aleatorio podrán realizar correcciones sobre las practicas de otros compañeros. Esto proporciona otra herramienta de aprendizaje, permitiendo a los alumnos ver el trabajo de sus compañeros y aprender de sus aciertos y errores.

Evaluación de revisiones así realizados

Los alumnos podrán emitir su valoración sobre el trabajo de otros alumnos, siendo esto visible solo por el profesor y el grupo evaluado. mediante la generación de un informe en pdf con los comentarios y posibles correcciones realizadas en el código.

1.3. Resumen objetivos

O1 Corrección de entregas de prácticas de código. Diseño y discusión en la sección 4.5.1.

O2 Integración con Campus Virtual. Diseño y discusión en la sección 4.5.2

O3 Capacidad por parte de alumnos de ver sus correcciones. Diseño y discusión en la sección 4.5.3

O4 Capacidad por parte de los profesores de gestionar las asignaturas, tareas y entregas. Sección 4.5.4

O5 Capacidad por parte de un administrador o administradores de gestionar la plataforma de manera sencilla. Sección 4.5.5

O6 "Peer review" entre alumnos. Sección 4.5.6

O7 Evaluación peer reviews así realizados. Sección 4.5.7

1.4. Organización del trabajo

El estado de la cuestión describe distintas formas de abordar el problema identificado en la motivación, y cómo se puede intentar cumplir los objetivos.

En el apartado de diseño, se describen posibles tecnologías que se pueden aplicar, así como arquitecturas, y motivo la elección de `angular2` sobre `nodejs` con `codemirror` como componente principal. Este apartado también describe el proceso por el que se eligió la metodología de trabajo aplicada, y en qué consiste ésta.

El apartado de implementación describe cómo se realizó la implementación de el sistema Scipio, y describe también los hitos fundamentales de su desarrollo. También proporciona una breve descripción de su arquitectura interna (para más detalles, ver apéndices y sobre todo, el código en su repositorio oficial[1]).

Finalmente, el apartado de resultados describe las contribuciones principales realizadas en este trabajo: el sistema entregado, así como los recursos que se proporcionan para su despliegue y gestión; y recapitula el diseño y la ingeniería que ha sido para ello utilizada.

El capítulo de conclusiones y trabajo futuro se relaciona, primero, los objetivos con los resultados; y posteriormente, se indican líneas futuras por las que se podría mejorar Scipio.

[1] https://bitbucket.org/olayer66/scipio

Chapter 2

Introduction

"Computers are like Old Testament gods; lots of rules and no mercy."
— Joseph John Campbell (1904-1987)

2.1. Motivation

Scipio is motivated by the need to improve communication between students and teachers when correcting programming exercises. It provides tools to facilitate code review by teachers, which can then not only mark errors, but also explain what is wrong and include a corrected version. Once reviews are finished, students can receive a report that allows them to see all corrections in the context of their surrounding code.

With current tools, mainly the campus-wide installation of Moodle, there are difficulties in communicating corrections of programming practices between teachers and students: neither can open or review code easily. For example, if a teacher complains that *line 10 of foo.c is wrong*, the student would have to fire up en external editor just to check what it said on that specific line. Moodle's tools also fail to provide any tool to summarize in one document all comments and corrections; if the teacher wanted to do this, much time would be wasted copying and pasting code. Instead, teachers would probably just explain general problems affecting many students, instead of the specifics of what is good or bad with each student's individual answer.

As a student at the School of Computer Science (*Facultad de Informática*) of the UCM, in subjects that have a strong programming component such as Fundamentals of Programming, Programming Technology or in higher courses such as Web Applications, Data Structures and Algorithms or Algorithmic Techniques in Software Engineering, I have noticed that communication between teachers and students, or even between students, is both

complex and time-consuming. Because of the tight schedule, in general the necessary time is never allocated.

Finally, a great motivation on my part was to improve my knowledge of new methodologies, languages and technologies currently in vogue in real business environments.

With the above mentioned motivations Scipio is born, with the spirit of creating a platform that meets these needs and improves as well as streamlines this important task for those learning to program.

2.2. Goals

2.2.1. Main goals

The objectives we intend to meet are detailed below.

Programming exercise correction

Teachers are one of the main actors of our application, and one of our goals is to allow them to review and correct code submitted by students. Therefore, through the use of the application, teachers will be able to correct the code of exercises turned in by students. This includs being able to make comments and corrections directly on the code, which, after a generated report, the student will be able access. These comments and corrections will be summarized into a single document when so requested. The teacher can also activate the peer-review correction function described in the advanced objectives.

Integration with Moodle

Integration with the virtual campus allows teachers greater agility when using Scipio to review code.

Teachers can download the exercises delivered in Moodle (Martin Dougiamas, 2019), and upload them into Scipio quickly and easily. Internally they are distributed among available teams (one or more students) through an internally-unique identifier in the file name, using the identifier of the group to which the delivery belongs.

Students can see their code-reviews

Feedback between teachers and students is a basic part of our application. Therefore, students must be able to see the comments and corrections made by the teacher, in addition to the report generated. With this system created interaction between teacher and students, improving communication between them.

Teachers can manage their classes and assignments

Teachers will be able to manage their subjects based on a common base subject and be able to manage their student teams, tasks and deliveries in a simple way.

Administrators can manage the platform

The administrator may perform maintenance and updating of the elements that make up the platform. Its main tasks will be to maintain the common base of subjects and the management of teacher profiles.

2.2.2. Advanced goals

Once the previous goals have been achieved, we also propose the following advanced objectives.

Peer review between students

Students, through a randomized system, may make corrections on the exercises of other classmates. This provides another learning tool, allowing students to see the work of their peers and learn from their successes and mistakes.

Grading of peer-reviews

Students may issue their assessment of the work of other students. When doing so, they will follow identical steps as teachers, including the generation of a summary pdf report with comments and possible corrections made in the code, visible only to the teacher and the authors of the reviewed code.

2.3. Recapitulation of goals

O1 Programming exercise correction. Design and discussion in section 4.5.1.

O2 Integration with Moodle. Design and discussion in the section 4.5.2

O3 Students can see their code-reviews. Design and discussion in section 4.5.3

O4 Teachers can manage their classes and assignments. Design and discussion in section 4.5.4

O5 Administrators can manage the platform. Design and discussion in section 4.5.5

O6 "Peer review" between students. Design and discussion in the section 4.5.6

O7 Grading of peer-reviews. Design and discussion in section 4.5.7

2.4. Layout of this paper

The state of the art describes various ways to address the problem identified in motivation, and possible approaches to meet our goals.

In the design section, possible technologies that can be applied, as well as architectures, are described; including why `angular2` was chosen over `nodejs`, with `codemirror` as the main component. This section also describes the process by which the applied work methodology was chosen, and what it consists of.

The implementation section describes how the implementation of the Scipio system was carried out, and also the fundamental milestones of its development. It also provides a brief description of its internal architecture (for more details, see appendices and above all, the code in its official repository[1]).

Finally, the results section describes the main contributions made in this work: the system delivered, as well as the resources provided for its deployment and management; and recapitulates the design and engineering that has been used for it.

The chapter on conclusions and future work relates, first, the objectives to the results; and later, future lines are indicated by which Scipio could be improved.

[1] https://bitbucket.org/olayer66/scipio

Capítulo 3

Estado de la Cuestión

3.1. Alternativas existentes

Se consideran las siguientes alternativas para facilitar la corrección de código fuente por parte de profesores y que permitan a los alumnos acceder fácilmente a estas correcciones en contexto:

- Los profesores incluyen contexto en sus correcciones tradicionales, lo suben como feedback a campus virtual, en esta caso los profesores pueden usar un editor de texto cualquiera (MS word, libre office, etc) que permita conservar el estilo del código al copiarlo, también pueden usar ficheros pdf generados a partir de `MarkDown` (Wikipedia, 2019b) o desde los editores de texto mencionados anteriormente. Posteriormente pueden enviar dicho documento a través del Campus virtual o imprimirlo y entregarlo en mano a los alumnos, esto es un proceso lento en ambos casos. Como ultimo punto este método no soporta una corrección ínter-pares, por lo que solo ayudaría a los profesores y no a los alumnos que serian los mas beneficiados.

- El uso de `GitHub` (The GitHub Team, 2019) u otro sistema que permita revisiones de código fuente, permite tener una visión del estado de los proyectos según transcurre, también permite dejar comentarios sobre el código y dispone de un sistema de `pull request`[1]. Esto puede facilitar las correcciones pero requiere de un profundo conocimiento de `GitHub`. También se requiere que todos los implicados tengan cuenta en la plataforma y el proceso de los profesores como administradores de crear los proyectos y asignar a los alumnos puede llegar a ser muy trabajoso. como ultimo punto tenemos la protección de datos, `GitHub`

[1] contribuciones que son candidatas a ser revisadas e integradas, pero que hasta que lo son quedan fuera de los fuentes oficiales, adquiriendo comentarios y revisiones

es una plataforma publica donde cualquiera puede los proyectos, esto genera ademas de un problema de posibles copias entre alumnos ademas de la protección de datos.

En conclusión, no se considera que los métodos alternativos para realizar esta tarea sean satisfactorios, y por tanto resulta necesario, para cumplir los objetivos, crear una nueva aplicación.

3.2. Elección del tipo de aplicación

Para desarrollar y poder solucionar los problemas identificados en la motivación,se ha buscado enfocar hacia una solución que permita un fácil uso por parte de los usuarios así como un mantenimiento sencillo, sin comprometer los objetivos fijados.

Se han tenido en cuenta varias opciones.

- Aplicación de escritorio
- Aplicación web
- Plugin para el campus virtual (moodle)

3.2.1. Aplicación de escritorio

El desarrollo de una aplicación de escritorio mediante Java (James Gosling y Microsystems, 2019), C++ (Bjarne Stroustrup, 2019) u otros lenguajes de similares características, tiene ventajas como la robustez o la rapidez de respuesta, pero también desventajas como la necesidad de instalación en el equipo, o de desarrollar o al menos probar versiones especificas para cada sistema operativo. En nuestro caso, debido a la naturaleza de nuestra aplicación, si bien la rapidez de respuesta sería bienvenida, la necesidad de instalación individual en cada puesto supone una desventaja difícil de salvar; y el desarrollo multiplataforma resultaría prohibitivo. Además, nuestra aplicación requiere de una persistencia de datos común a todos los usuarios, lo cual se facilita con un servicio común alojado en un servidor.

3.2.2. Aplicación web

Una aplicación web ofrece ventajas como la independencia del dispositivo y del sistema operativo; además, cualquier mejora solo requiere de una actualización de la aplicación en el servidor para estar disponible para todos sus usuarios.

En su contra tiene el requerimiento obligatorio de disponer de una conexión a internet, así como de tener que ser compatible con los navegadores

más usados[2]. Otra desventaja es que puede haber retrasos debidos al tiempo que tardan las constantes peticiones en ir y volver al servidor, algo que no sucede en una aplicación de escritorio. Esto se puede minimizar con el uso de cachés y técnicas de carga asíncrona (por ejemplo, AJAX).

3.2.3. Plugin para Moodle

El desarrollo de un plugin para Moodle (Martin Dougiamas, 2019) ofrece ventajas a la hora de la integración con el campus virtual, facilitando a los profesores y alumnos el uso de la herramienta. Por otro lado, la gestión del Campus Virtual está centralizada en la UCM, y sería administrativamente muy difícil conseguir aceptación de un plugin a esta escala. Por tanto, se desiste de realizar una integración a este nivel.

3.2.4. Conclusión

Basándonos en lo anteriormente expuesto, se ha tomado la decisión de realizar nuestro proyecto mediante una aplicación web. Esto nos ofrece muchas ventajas, incluyendo, ya que no haremos un plugin Moodle, una gran libertad para elegir cómo desarrollar la aplicación en términos de tecnologías y librerías existentes.

3.3. Peer review

La revisión ínter-pares o *Peer review* (Wind y Jensen, 2017) en el ámbito del software consiste en, según (Wikipedia contributors, 2019),

> *La revisión de software en la cual un producto de trabajo (documento, código u otro) es examinado por su autor y uno o más colegas, para evaluar su contenido técnico y calidad*

En nuestro proyecto la revisión ínter-pares va a estar representada en la corrección de las practicas de unos alumnos por parte de otros. Este es un método que fomenta el aprendizaje por medio de ver frente a un mismo problema distintas soluciones aportadas por otros que se encuentran al mismo nivel de conocimientos. El uso de este método en principio es meramente formativo por lo que los alumnos no calificaran el trabajo de sus compañeros de forma directa, aunque los profesores pueden otorgarle valor si lo consideran oportuno y de la forma que consideren mejor.

[2]Algo que, con tecnologías como HTML 5 y jquery, es relativamente sencillo y rápido

Capítulo 4

Diseño

"The Grid. A digital frontier. I tried to picture clusters of information as they moved through the computer. What did they look like? Ships, motorcycles? Were the circuits like freeways? I kept dreaming of a world I thought I'd never see. And then, one day I got in..."
— Kevin Flynn, TRON

Para el diseño del proyecto se decidió usar una aplicación web. Esto permite crear una aplicación modular que no es necesario instalar ni actualizar en cada puesto, con alta usabilidad y con un manejo de datos centralizado. Durante la fase de concepción se exploraron varias posibles tecnologías, tales como Spring MVC (The Spring Team, 2019), Ruby (Rails Contributors, 2019) frameworks tanto para el frontend como el backend. También se exploraron soluciones separadas como React (React contributors, 2019) o Angular (Angular Contributors, 2019b) para el frontend y Express (TJ Holowaychuk, 2019) o Sails (Mike McNeil, 2019) para el backend. Para la persistencia de la información se estudiaron las dos ramas principales de bases de datos, sql y nosql. Para sql se consideraron diferentes tecnologías tales como postgreSQL (Group, 2019) o MySQL (MySQL AB y Oracle Corporation, 2019). En cuanto a las bases de datos nosql se consideraron Cassandra (Lakshman y Malik, 2019) y MongoDB (MongoDB, Inc., 2019).

A continuación explicaremos mas en detalle las tecnologías estudiadas con sus ventajas y desventajas.

Tecnologías mixtas

Las tecnologías mixtas consideradas para proporcionar tanto el frontend como el backend a la vez han sido Spring ó Ruby. Usar una única tecnología para frontend y backend es una gran ventaja en simplicidad, pero a su vez esto también limita su utilidad para ciertas partes de nuestra aplicación. Por

ejemplo, el corrector de código, si va a ser realmente interactivo, tiene que ejecutarse en el frontend en forma de JavaScript.

Ruby Es una tecnología popular para la web, con similitudes respecto a Perl y Python, y dispone de un entorno para desarrollo web llamado Rails (Hansson, 2003). Esta tecnología fue descartada por el lenguaje usado dado que no se tiene conocimiento del mismo y aprender un lenguaje completamente nuevo costaría un tiempo del que no se dispone.

Spring MVC por el contrario esta basado en Java, que es uno de los lenguajes más populares del momento. Se descarto su uso aun siendo una tecnología ya conocida en la carrera, ya que se prefería usar un único lenguaje de programación para todo el proyecto (JavaScript), así como explorar nuevas tecnologías sobre este lenguaje.

Frontend

El frontend es la parte visible de la web, con la que interactúan los usuarios a través de sus navegadores. Para el frontend se pusieron sobre la mesa dos tecnologías sobre NodeJS muy populares hoy en día: React (React contributors, 2019) y Angular[1]. Ambas son muy similares y la vez muy diferentes, la principal diferencia es que mientras React es solo una librería de JavaScript, Angular es un framework completo. Ambos se basan en JavaScript, pero Angular va un paso mas allá y usa TypeScript (Microsoft, 2019) que es un superconjunto de JavaScript que esencialmente añade tipos estáticos y objetos basados en clases. Ambos son grandes tecnologías y con una popularidad similar, y ambas habrían sido válidas para nuestras necesidades. Nos hemos decantado por el uso de Angular, ya que por motivos laborales es una tecnología ya conocida y no requiere de un proceso de aprendizaje tan extenso como React.

Backend

El backend es la parte del sistema que da soporte al frontend, contiene la lógica de aplicación, y accede y gestiona los datos de la aplicación y su persistencia. Para el desarrollo del backend se propuso el uso de Sails Lozano (2014), un framework basado a su vez en el framework Express, al que añade varias capas de abstracción para hacer un desarrollo más fácil. Entre otras cosas posee un ORM[2], métodos para crear APIs RESTful[3]y soporte para manejar peticiones en tiempo real gracias a `Socket.io` (Guillermo Rauch,

[1] Ver estadísticas en lambdatest (Nikhil, 2019)

[2] Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia.Ver ORM (Wikipedia, 2019a).

[3] La transferencia de estado representacional o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos. Ver (Wikipedia, 2019c).

2019) [4]. Aunque Sails ofrece ciertas ventajas a la hora del desarrollo ,se decidió no usarlo dado que las ventajas que nos pude proporcionar no supera la necesidad que crearía de aprender a manejar una tecnología nueva. Por las razones anteriores se ha decidió usar el framework Express directamente, dado que se conoce ya su manejo por su uso durante la carrera en distintas asignaturas.

Elección del modelo y almacén de datos

En la elección de un sistema de persistencia de la información se barajó almacenar los datos tanto en una base de datos SQL como en un almacén noSQL, y dentro de cada una varias opciones de implementación de cada una.

noSQL

Los almacenes de datos noSQL difieren de las bases de datos tradicionales en diferentes aspectos importantes, como que los datos almacenados no requieren estructuras fijas o no garantizar completamente las propiedades ACID (atomicidad, consistencia, independencia y durabilidad); o que no usan SQL para consultas. Para nuestra aplicación se han estudiado dos almacenes de datos noSql, Cassandra y MongoDB. Ambas son similares sus principales diferencias son que Cassandra se basa en un modelo clave/valor y MongoDB esta orientada a documentos de esquema libre usando un formato propio de documento llamado BSON, que es una versión más eficiente (pero menos legible) de JSON.

En términos generales se rechazo este tipo de base de datos dado que su mayores ventajas que son, la agilidad, la escalabilidad y la capacidad de procesar grandes cantidades de datos, no superan las ventajas en cuanto a la consistencia y potencia del lenguaje sql que ofrecen las bases de datos sql clásicas, que en nuestro caso particular son mas necesarias por el trato a la información que necesitamos dar.

SQL

Las bases de datos relacionales que se han estudiado para implementar la persistencia son PostgreSQL y MySQL Community Server, por tener ambas una gran base instalada y ser de código abierto[5]. Para nuestro propósitos ambas son validas. Se he decidido usar MySQL dado que ya es conocida con anterioridad por asignaturas de la carrera y experiencia personal.

[4]Permite recibir peticiones de forma asíncrona del servidor, sin que éste tenga que esperar a recibir una petición antes de poder informar de nuevos eventos.

[5]Técnicamente, MySQL, tras su adquisición por Oracle, no es 100 % código abierto (aunque el Community Server sí lo es). No obstante, sería equivalente usar MariaDB, que es una versión que sí es completamente abierta y mantiene muy buena compatibilidad

Tecnologías elegidas

Tras explorar las posibles soluciones citadas anteriormente, se ha decidido usar:

Frontend Angular en su ultima versión estable (8.2.x)

Backend Express sobre NodeJS

Persistencia MySQL ó MariaDB

Diseño del frontend con Angular

Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC).

La biblioteca lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, entonces obedece a las directivas de los atributos personalizados, y une las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript (Brendan Eich, 2019).

Angular se basa en *componentes*, cuyas propiedades son usadas para hacer el una ligadura (*binding*) de datos entre los servicios expuestos por el backend y la interfaz. En dichas clases tenemos tanto propiedades (variables) como métodos (funciones a llamar).

Diseño del backend con ExpressJS

Express es una infraestructura de aplicaciones web NodeJS (Ryan Dahl, 2019) mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles. Este se puede expandir mediante componentes que permiten realizar múltiples funciones. Con esta base nos proporciona una plataforma en la que basar nuestro servicio que alimentara el frontend de Express así como recibir datos del mismo para realizar las acciones necesarias.

Persistencia de la información

La persistencia de la información se realizara mediante el uso de una base de datos con la tecnología MySQL. La integración con la aplicación se realizara mediante el uso de la librería de NodeJS llamada Knex (Knex contributtors, 2019), la cual nos permite integrar en el código la creación del

esquema de tablas y las peticiones a la BBDD, y proporcionando múltiples herramientas que facilitan el uso de la persistencia en la aplicación.

4.1. Usuarios

Los usuarios de nuestra aplicación están divididos en tres tipos.

Administradores Su principal función es la del mantenimiento de la aplicación, Pueden crear asignaturas así como dar de alta a los profesores en la aplicación.

Profesores Pueden gestionar las asignaturas, junto con sus tareas y sus entregas.

Alumnos Pueden ver las asignaturas, tareas y entregas a los que estén vinculados mediante un equipo.

Los profesores pueden tener varias asignaturas, y una misma asignatura tener varios profesores. Estos vincularán a los alumnos a los equipos de la asignatura. Idealmente, estos datos provendrían de una integración con el sistema de gestión académica donde se instale Scipio; no obstante, en el diseño inicial, hemos considerado esta integración como no-prioritaria.

4.2. Asignaturas, grupos, tareas, envíos

Junto con los usuarios de distinto tipo, las *asignaturas*, *tareas* y *envíos* constituyen el núcleo principal de la aplicación. Esto comprende la gestión de estos tres elementos, incluyendo tanto su creación y modificación como su eliminación. la implementación de las asignaturas está dividida en dos partes:

1. La asignatura en sí, que comprende su código de asignatura, su abreviatura, nombre, y otras propiedades, especificadas por los administradores del servicio. Por ejemplo, para la asignatura *Estructura de Datos y Algoritmos*, la abreviatura podría ser *EDA* y el código GEA *803210*.

2. Los *grupos* de la asignatura específicos que crea y gestiona cada profesor: el curso al que pertenece, la clase/grupo al que va a estar asignada, etcétera. Por ejemplo, un profesor puede ser responsable del grupo *A2* de la asignatura anterior.

Una vez creada una asignatura se le pueden añadir *tareas* con un nombre, una descripción y una fecha de entrega. Por defecto se crean desactivadas y el profesor puede activarlas y desactivarlas a su discreción. Cuando el profesor considere, puede iniciar la corrección de una tarea subiendo al servidor

un fichero con la estructura dada por la plataforma moodle para tareas, en su versión 3.4 (fichero *.zip* con carpetas, cada una con el nombre del alumno y equipo que la entrega; y en el interior de estas un *.zip* con la entrega). Para poder asignar la entrega a un equipo, el fichero zip de la entrega tiene que contener en el nombre el id del equipo al que pertenece. Una vez subido este fichero, se generan de forma automática los envíos y sus correcciones pertinentes; y el profesor puede ya activar la función de revisión ínter-pares. Los envíos son una colección de equipos y sus entregas. En cada uno se tendrá acceso a la información del equipo al que pertenece y la información del equipo de corrección ínter-pares, así como las correcciones asociadas. El profesor siempre tendrá una corrección, y en caso de estar activada la corrección ínter-pares también se podrán ver. los profesores podrán visualizar todas las correcciones, pero solo podrán modificar las propias, mientras que los alumnos solamente podrán ver aquellas en que estén implicados, y modificar aquellas en las que sean correctores ínter-pares, en caso de que esta opción sea activada por el profesor.

Cada corrección puede tener tres estados.

Nueva Si no se ha tocado desde su creación

En proceso Si ya se ha accedido al código

Finalizada Una vez se ha terminado de corregir y se ha generado el informe de corrección

Si el estado es *en proceso* o *finalizado* se muestra un resumen de la corrección con el total de ficheros comentados, y el número total de comentarios; y si se desea, aparecerán estas mismas cifras desglosadas por fichero. También dependiendo del estado y el usuario, se mostraran las distintas acciones posibles:

Editar corrección Ver el código de las correcciones, con la posibilidad de modificarlas

Ver corrección Ver el código y las correcciones ya realizadas

Finalizar corrección Da por finalizada la corrección, y genera el PDF con el informe de corrección

Ver PDF Solo accesible con la corrección finalizada. Devuelve un PDF con la corrección para imprimir o subir, como feedback, al Campus Virtual.

4.3. Sistema de corrección

Para la corrección del código asociado a una practica, se ha elegido como base la librería CodeMirror (CodeMirror Volunteers, 2019). Esta librería nos

permite mostrar ficheros de código de una forma en que se pueden diferenciar fácilmente las diferentes partes del mismo, ya que soporta realzado de sintaxis mediante colores y fuentes (*syntax highlighting*). También permite, entre otras cosas, la gestión de temas (por ejemplo, para elegir entre claro y oscuro); o la posibilidad de permitir modificar el contenido, o acceder en modo solo lectura. Sobre esta librería será necesario añadir funciones para gestionar comentarios, ya que no encontramos ninguna librería abierta que tuviese toda la funcionalidad necesaria.

Junto con este visor de código, se implementará un árbol de ficheros que nos permite seleccionar sobre los ficheros disponibles de una corrección aquel que queremos visualizar a la hora de generar comentarios o correcciones[6] .Para facilitar el desarrollo de la funcionalidad relativa a la gestión de comentarios sobre CodeMirror, nos decantamos por usar jQuery (jQuery Foundation, 2019).

Creación, modificación y eliminación de comentarios

Para diseñar esta parte lo primero que se hizo fue estudiar el código que genera de forma automática la librería `CodeMirror`, con esto pudimos saber que etiquetas de html nos eran útiles a la hora de insertar nuestras funciones. Para poder realizar estas funciones se ha decidió usar los números de linea que se muestran en el visor, usando un sistema de selección única o multilinea, con una selección ya realizada o un comentario ya existente y mediante el botón derecho del ratón se abrirá un menú contextual(`ContextMenu`) que nos dará las diferentes opciones disponibles. Para la creación o edición de un comentario se ha optado por el uso de una ventana modal que nos permita insertar o modificar los elementos de un comentario. También se permitirá el borrado de comentarios, mediante el menú contextual o en la ventana de edición.

4.4. Corrección ínter-pares (peer review)

La corrección ínter-pares o peer review consiste principalmente en que un alumno o grupo de alumnos (equipo) corrija la entrega de otro alumno o equipo. Este tipo de corrección favorece entre otras cosas la mejora de la comprensión del código, la capacidad de pensar, al ver diferentes soluciones a un mismo problema o la capacidad de autocrítica al comparar su código con el de otros compañeros. Este sistema de corrección es completamente opcional, el profesor puede activarlo y pararlo a su discreción. Cuando es activado el sistema genera de forma aleatoria una combinación aleatoria de

[6]Para realizar correcciones se usa la misma base que un comentario pero modificando el código seleccionado, por lo que para abreviar solo usaremos la palabra comentarios para referirnos a las dos opciones

entrega/equipo partiendo la de la base de que un equipo no puede corregir su propia practica. Esta asignación es secreta por lo que los alumnos no pueden conocer a quien están corrigiendo, ni quien les corrige a ellos, Solo el profesor ve esta información, quedando en su mano comunicársela a los alumnos o no. Los profesores no pueden cambiar dichas correcciones aunque si pueden visualizarlas, para un mayor control sobre las mismas.

4.5. Alcanzando los objetivos

El apartado anterior ha estudiado las tecnologías disponibles. Este apartado se centra en cómo lograr los objetivos planteados al comienzo del trabajo.

4.5.1. O1: Corrección de entregas de prácticas de código

> *Mediante el uso de la aplicación los profesores podrán corregir el código de las prácticas entregadas de los alumnos, pudiendo realizar comentarios y correcciones de código sobre la misma que luego mediante un informe generado el alumno podrá ver las mismas.*

Ver el código

En nuestra aplicación este objetivo se verá reflejado en el componente `Codemirror`, que incluirá el visor de código, la gestión y visualización de los comentarios realizados, y una representación del árbol de directorios y ficheros asociados a la corrección. Todo lo anterior tendrá su reflejo en el servidor del backend, con el controlador correspondiente.

Realizar comentarios y correcciones

Esta herramienta permite al profesor realizar comentarios y correcciones[7]sobre una o varias líneas de un fichero.

Podrá crear, modificar y eliminar comentarios y correcciones, siempre que no se haya enviado ya la corrección: una vez enviada quedará bloqueada la corrección de dicha practica.

En la vista podrá ver los comentarios y correcciones ya realizados así como ver de forma sencilla y rápida tanto los comentarios como las correcciones sobre el propio código.

Una vez finalizados los cambios el profesor podrá generar el informe de corrección que se enviara al/los alumno/s.

[7]Los comentarios y la correcciones van unidas, pero mientras que un comentario puede no incluir una corrección del código, las correcciones sí deben incluir obligatoriamente un comentario explicativo

Generar informe pdf

Mediante esta herramienta se generara un fichero pdf estructurado con todos los comentarios y correcciones de todos los ficheros que contenga la practica. Este a su vez sera enviado al/los alumno/s responsables de la practica. En nuestra aplicación este objetivo se ve reflejado en el componente `Corrections` en el cual se puede finalizar la corrección la cual da lugar a la generación del informe

4.5.2. O2: Integración con Campus Virtual

Para que el profesor pueda descargar las practicas entregadas en el campus virtual (moodle) y cargarlas en la aplicación de forma rápida y sencilla, permitiremos la carga directa de estos ficheros .zip en la aplicación.

EL profesor tendrá la posibilidad de cargar un conjunto de ellas o una entrega especifica si ya se ha cargado un conjunto previamente, tendiendo como referencia el nombre del archivo de la entregar que tendrá que ser igual al código distintivo del alumno o grupo que la entrega, de forma interna se distribuirán entre los alumnos y/o grupos. En nuestra aplicación este objetivo se ve reflejado en el componente `Assignments` en el cual se puede subir el fichero comprimido y genera la entregas y correcciones pertinentes.

4.5.3. O3: Alumnos pueden ver sus correcciones

Los alumnos podrán ver los comentarios y correcciones hechas por el profesor. Con esto se creara un sistema de interacción entre el profesor y el alumnado, mejorando la comunicación entre ambos. Esto sera posible una vez la fecha de entrega de la practica en cuestión termine y esta haya sido corregida por el profesor, también en el caso de tener activada la opción del revisión ínter-pares estos podrán ver la correcciones hechas por sus compañeros.

En nuestra aplicación este objetivo se ve reflejado en el componente `Submissions` en el cual los alumnos pueden ver sus correcciones y las que tengan que realizar.

4.5.4. O4: Capacidad por parte de los profesores de gestionar las asignaturas, tareas y entregas

Los profesores podrán gestionar las asignaturas, podrán crearlas y modificarlas ademas de activarlas y desactivarlas. Esta tiene como base un código de asignatura y un nombre para diferenciarlas del resto de asignaturas con un mismo código. Cada asignatura tendrá asignada una serie de equipos formados por alumnos los cuales podrán ser gestionados también por los profesores. Las tareas estarán asignadas a una asignatura y podrán ser creadas y modificadas ademas de activadas y desactivadas. También una vez subidas las

correcciones podrá activarse la corrección ínter-pares para la misma. A su vez las tareas tendrán entregas (una por equipo) que tendrán correcciones, una para el profesor y en caso de estar activada la corrección ínter-pares la generada para el alumno al que le corresponda corregir.

En nuestra aplicación este objetivo se ve reflejado en varios componentes, las asignaturas y los equipos asociados se gestionan en el componente `Subjects`, las tareas podrán ser gestionadas en el componente `Assignments`, mientras que las entregas y las correcciones podrán ser gestionadas en el componente `Submissions`.

4.5.5. O5: Capacidad por parte de un administrador o administradores de gestionar la plataforma de manera sencilla

En nuestra aplicación tendremos un usuario especial con la capacidad de gestionar ciertos aspectos.Estos son la el mantenimiento de los códigos de asignatura, el mantenimiento de los periodos lectivos y la creación de profesores. En nuestra aplicación este objetivo se ve reflejado en los componentes `adminUsers` y `adminCodes`, los cuales solo son accesibles mediante un usuario de tipo administrador. El componente `adminUsers`, permitirá la gestión de los tres tipos de usuarios (alumnos, profesores y administradores), estos se podrán crear, modificar y cambiar su contraseña, adicionalmente se les podrá activar o desactivar la cuenta, permitiendo de esta manera controlar quien tiene acceso a la aplicación. El componente `adminCodes` permitirá la gestión de los elementos relacionados con las asignaturas, estos son la gestión de los códigos de asignatura y la gestión de los cursos. Los códigos de asignatura son el núcleo que posteriormente los profesores usaran para crear sus grupos, estos se podrán crear y modificar. Los cursos se compondrán de una edición, por ejemplo, 2018/2019 y un estado, este marcara que cursos estarán activos y cuales solo se podrán visualizar.

4.5.6. O6: Revisión entre pares de alumnos

Los alumnos mediante sorteo aleatorio podrán realizar correcciones sobre las practicas de otros compañeros. Esto proporciona otra herramienta de aprendizaje, permitiendo a los alumnos ver el trabajo de sus compañeros y aprender de sus aciertos y errores. Esta funcionalidad esta sujeta a la decisión del profesor que puede activarla o desactivarla según quiera.

En nuestra aplicación este objetivo se ve reflejado en el componente `Submissions` en el cual los alumnos pueden ver las correcciones que tengan que realizar.

4.5.7. O7: Evaluación de revisiones así realizados

Los alumnos podrán emitir su valoración sobre el trabajo de otros alumnos esta parte va ligada a la revisión ínter-pares. Cuando los alumnos finalicen una corrección se les mostrara la opción de asignarle una valoración al trabajo corregido,E siendo solo visible por el profesor.

Esto se vera reflejado mediante una ventana modal para los alumnos en la que se les solicitara dar la valoración, y en el caso de los profesores se vera un campo informado con dicha nota en la `corrección` dentro de la `entrega`.

Relación de objetivos y componentes

Objetivo	Descripción	Componentes
O1	Corrección de entregas de prácticas de código	CodeMirror Submissions
O2	Integración con Campus Virtual	Assignments
O3	Capacidad por parte de alumnos de ver sus correcciones	Codemirror Submissions
O4	Capacidad por parte de los profesores de gestionar las asignaturas, tareas y entregas	Subjects Assignments Submissions
O5	Capacidad por parte de un administrador o administradores de gestionar la plataforma de manera sencilla	RootManager
A1	"Peer review" entre alumnos	Assignments Submissions

Tabla 4.1: Relación de Objetivos y componentes

4.6. Plan de implementación

Se parte de un diseño y desarrollo con una duración máxima de 12 meses, planificado para una entrega en la convocatoria de septiembre de 2019, con reuniones de coordinación bisemanales, iterando sobre prototipos desde el momento en que se seleccionen las tecnologías a usar.

Se trabaja sobre un repositorio Git para código y documentación, incluida esta memoria, con actas de reuniones mantenidas en un sistema de documentos compartidos (Google Drive), y donde se documenta el progreso y se apuntan los objetivos a conseguir de una reunión a la siguiente.

A continuación mostramos el diagrama de Gantt correspondiente al plan de implementación:

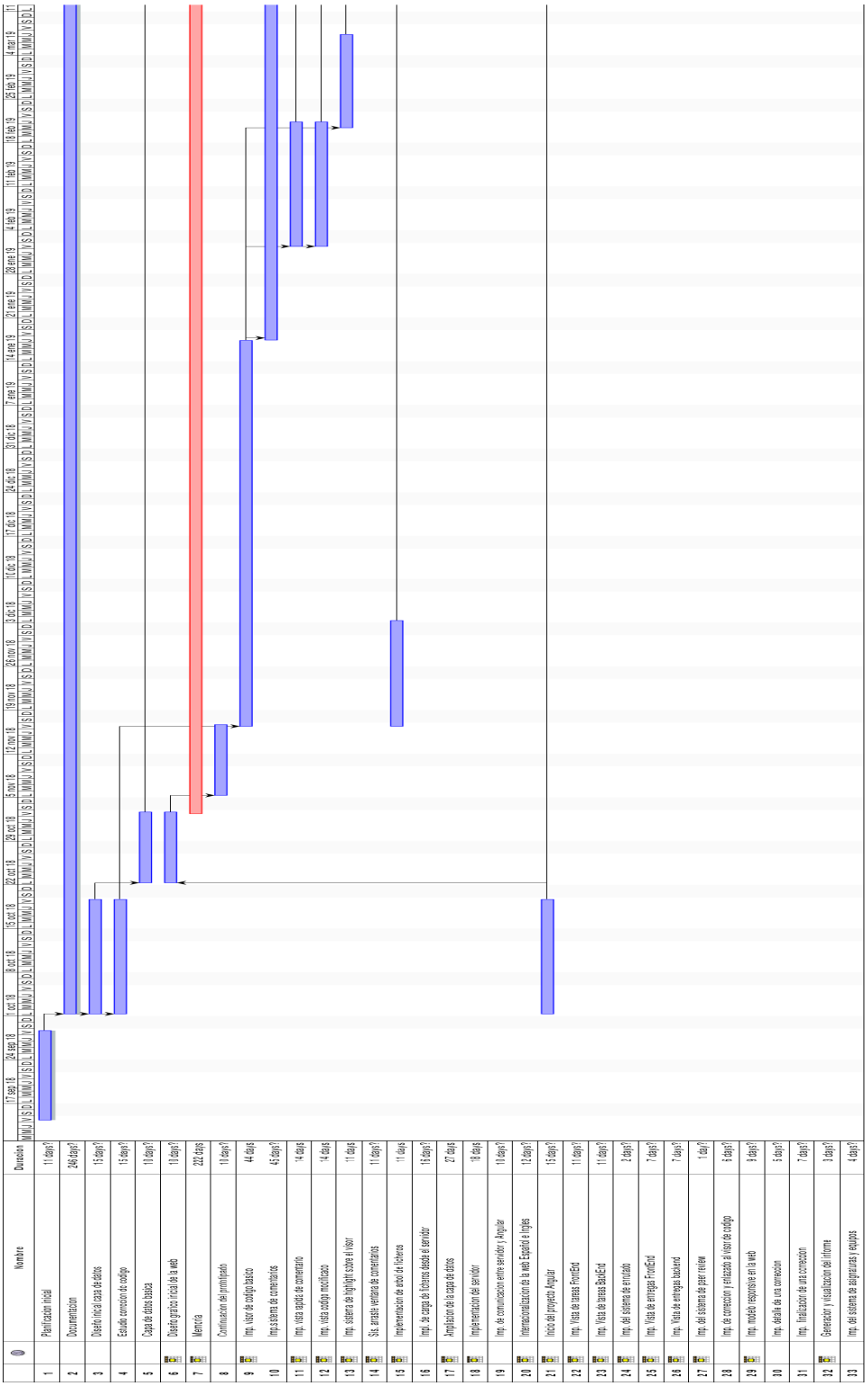


Figura 4.1: Diagrama de Gantt

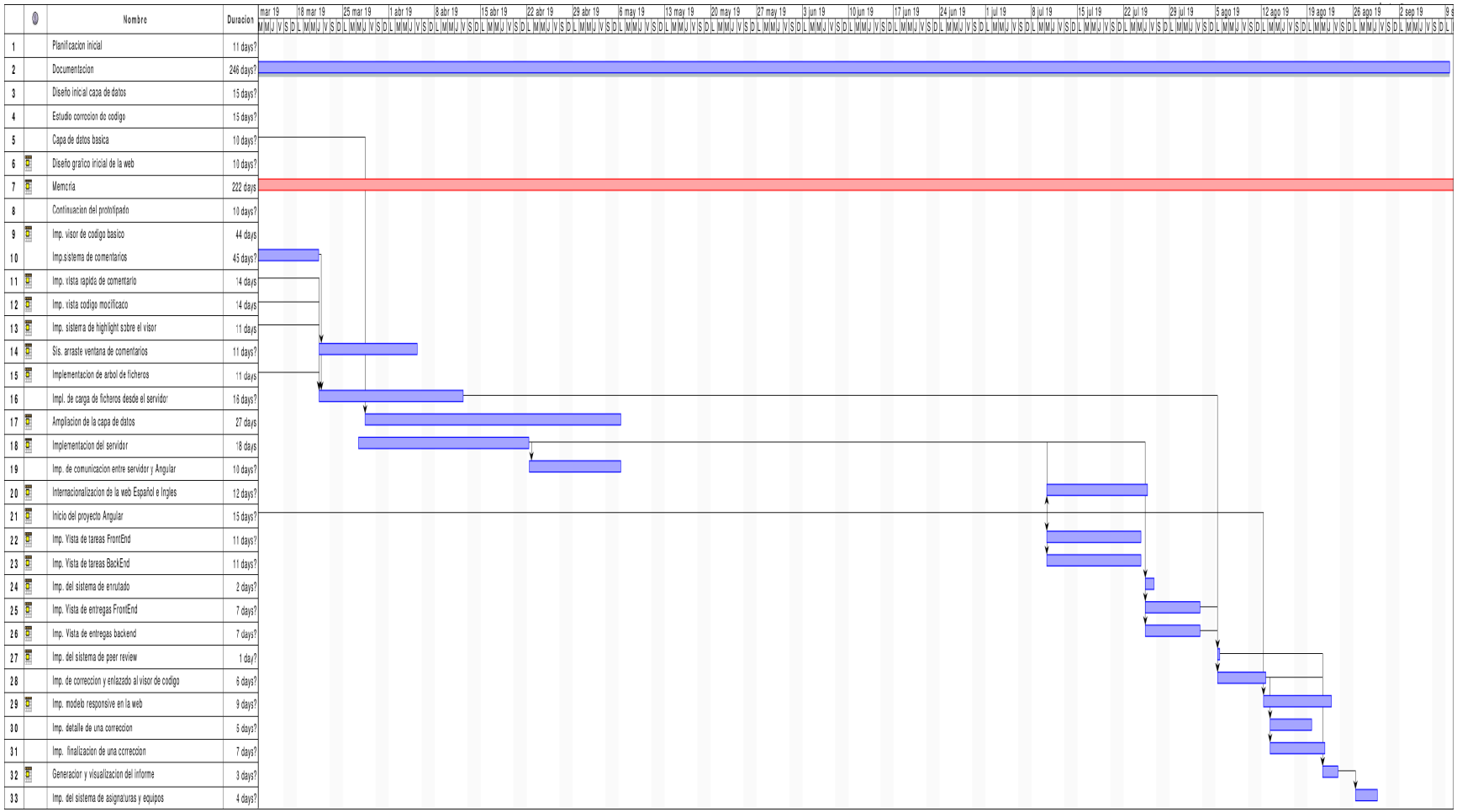


Figura 4.2: Diagrama de Gantt (continuación)

4.7. Diseño de navegación

Diagrama de relación entre los componentes de nuestra aplicación.

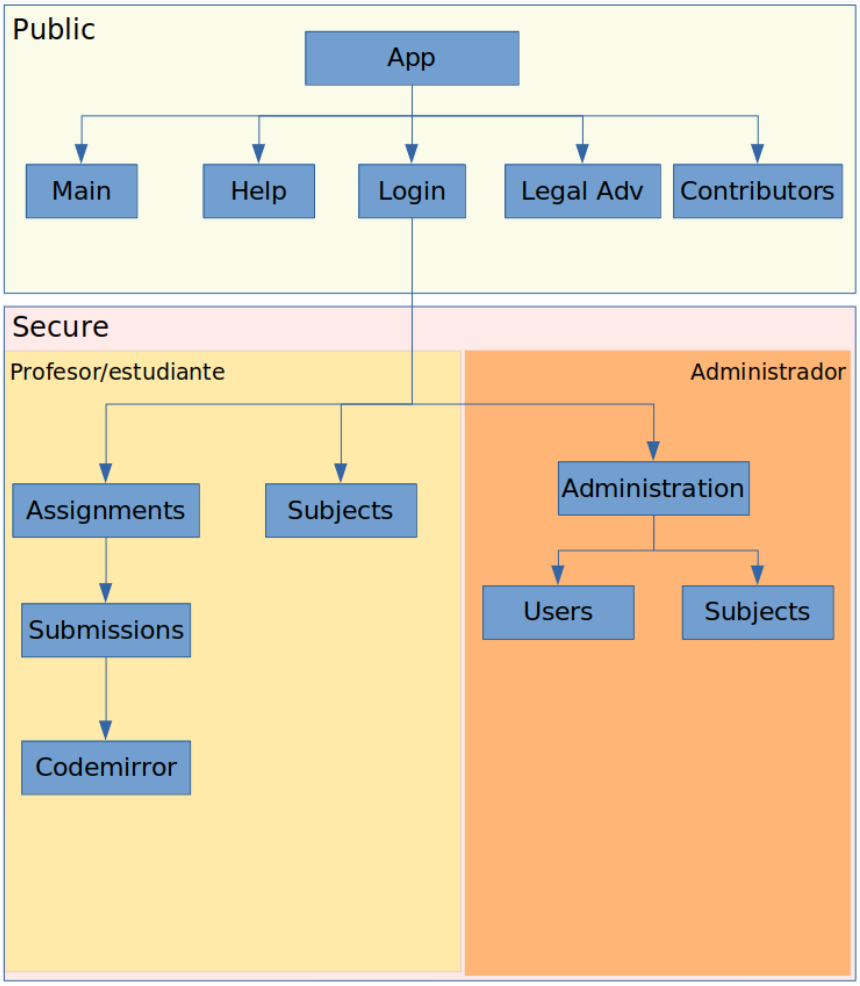


Figura 4.3: Esquema de relación entre los componentes

Capítulo 5

Implementación

5.1. Introducción

Las cuestiones básicas relacionadas con la implementación como el sistema operativo o el IDE de desarrollo se han tomado las decisiones basándose en la tecnologías que se serán usadas durante el desarrollo y posterior despliegue de la aplicación. En relación al sistema operativo se ha usado Ubuntu en su version 18.04 o superior dado que la aplicación sera desplegada en un sistema igual o similar basado en Linux con arquitectura Debian. En cuanto al IDE de desarrollo se ha usado WebStorm que esta basado en IntelliJ desarrollado por JetBrains, Este proporciona todas las herramientas para el desarrollo en Angular (typeScript) y javascript. También se han usado otras aplicaciones como MySQL Workbench para el desarrollo y mantenimiento de la BBDD. Tanto el IDE como el gestor de la BBDD se han elegido por ajustarse de la mejor manera posible a las necesidades del proyecto y conocimientos de los miembros del equipo, pudiendo usarse otros sin mayor problema.

5.2. Desarrollo del frontend con Angular 8

Para el desarrollo del frontend como se ha indicado anteriormente se ha decidido usar el framework Angular en su version mas reciente. Al tratarse de una tecnología nueva para los alumnos implicados en este proyecto ha requerido de un aprendizaje continuo que se nota en las mejoras continuas sobre el esta parte del proyecto según se profundizaba en el framework y avanzaba el proyecto. En primer lugar se creo un proyecto básico mediante el uso del cliente propio de Angular proporcionado al instalar Angular en el sistema,cuyo acceso mediante linea de comandos y AngularCLI (Angular Contributors, 2019a), proporciona ademas de la creación del proyecto las

herramientas para la creación de elementos adicionales propios del framework (componentes, servicios, etc). El proyecto inicial esta compuesto por los ficheros y directorios básicos de todo proyecto de angular, entre los que podemos encontrar.

angular.json La configuración básica del proyecto.

package.json Información de los paquetes npm instalados.

tsconfig.json Fichero donde están configurado el compilador de `typeScript`, básico para el desarrollo en angular.

src Es el directorio mas importante de todos, contiene toda nuestra web, todos los módulos, componentes servicios y demás contenidos necesarios. Ficheros importantes dentro de este directorio son por ejemplo:

- **index.html** Fichero sobre el que se cargaran todos los componentes de la web. Angular se basa en SPA(single-page application), por que todos los módulos se cargaran sobre este fichero en base a las peticiones recibidas.
- **assets** Directorio principal de recursos para nuestra web, imágenes, iconos, fichero de idioma,etc.
- **app** Contiene todos los componentes, módulos, servicios, etc, de nuestra web.

Los descrito anteriormente son los componentes indispensables para un proyecto angular básico, posteriormente se añaden mas ficheros y directorios según avanza el proyecto y su complejidad aumenta, tales como módulos para el enrutamiento, configuración del gestor del versiones, configuración del gestor de procesos (pm2) o el propio directorio donde se encuentra el servidor de backend. Todos ellos se verán a posteriori.

Angular dispone de sus propias librerías para realizar muchas de las funcionalidades (Nucleo de angular) por ejemplo Material.También hay disponibles librerías externas normalmente denominadas por las letras "ngx", por ejemplo, `ngx-translate`, añaden nuevas funcionalidades el framework o mejoran/sustituyen las del núcleo de angular. En nuestro proyecto usamos de ambos tipos dependiendo de las necesidades que se han tenido durante el desarrollo. Cabe remarcar que las librerías externas muchas veces suelen terminar formando parte del núcleo de angular en versiones posteriores del framework lo que demuestra el gran compromiso de la comunidad y de sus creadores.

En los apartados siguientes hablaremos en profundidad de la implementación de las diferentes partes de la web, Hablaremos en termino de componentes (elementos básicos de Angular) como funcionalidades, siendo así que

un componente contiene una funcionalidad de la web, por ejemplo el componente .Assignmentsçontiene las funcionales que se pueden realizar sobre la/s tarea/s de una asignatura.

Estilo grafico

Para la implementación grafica de la web se ha usado la librería grafica de angular .Angular Material"(Angular Contributors, 2019c). Esta librería nos permite mantener un estilo y diseño constante en la web ademas que proporcionar múltiples herramientas para tener un diseño responsive y adecuado.

Ademas Angular Material nos provee de elementos gráficos ya creados que nos proporcional una gran cantidad funcionalidad con una pequeña configuración, ademas me mantener una coherencia entre ellos.

A parte de la librería mencionada anteriormente tenemos una serie de hojas de estilos basadas en el lenguaje Sass (Hampton Catlin y Anne, 2019). Estas están divididas en, una hoja principal "styles.scss.en la raíz del proyecto que afecta a toda la web. Ademas cada componente tendrá su propia hoja de estilo para aquello que solo le afecte a el. En la hoja principal se encuentra todo aquello que afecta al conjunto completo de la web, entre estos elementos esta la hoja de estilo que requiere Angular Material para funcionar, y la paleta de colores de la web. En el caso de nuestro proyecto se ha optado por distribuir los estilos principales en distintas hojas para mejorar la legibilidad y la modularidad a la hora de poder modificar y mejorar en el futuro. Las diferentes hojas que se han creado están divididas por el elemento al que hacen referencia, tales como el pie de pagina o el menú principal.

Figura 5.1: Ejemplo ventana principal

Color

Los colores en nuestra web están definidos en dos sitios el primero es la hoja que define el estilo para la librería Angular Material, el otro es la paleta de colores`_palette.scss"` que nos proporciona mediante un mapa el color principal ademas de las distintas saturaciones y contrastes correspondientes y lo mismo para nuestro color secundario. Este mapa es accesible mediante el comando `map_get(mapa,valor)` que nos devuelve el código de color a usar. En relación a los colores se ha decidido usar los mismos colores que se usan en la web de la facultad para mantener un estilo común con ella.

Routing (Enrutado)

El sistema de enrutado (`routing`) de la web permite interconectar los distintos componentes entre si permitiendo entre otras cosas el paso de parámetros entre componentes. Para nuestro proyecto se ha decidido por tener un sistema múltiple con dos enrutadores. El principal que se encarga del acceso a los componentes que no requieren de acceso de usuario. Este a su vez enlaza con el enrutador de parte segura mediante un acceso por login de usuario. Al ser independientes uno del otro nos permite un mayor control sobre la seguridad del sitio. Este sistema esta implementado mediante dos módulos, el principal situado en la raíz del proyecto y el otro en directorio `Secure`. Su estructura es muy sencilla, tiene un vector de rutas que se componen de una cadena de texto que indica los parámetros de url y el componente al que ha de apuntar la ruta, este vector a su vez se usa como parámetro a la hora importar el modulo `RouterModule"` proporcionado por angular, el modulo termina con la exportación del mismo para poder ser usado. Para mas información sobre el enrutado en angular se puede consultar la documentación oficial Angular Contributors (2019d).

Internacionalización (i18n)

La Internacionalización de la web se ha realizado mediante la librería externa de angular `ngx-translate` Angular Contributors (2019e). Esta librería nos permite que mediante unos ficheros json formados por parejas de clave (Nombre de la variable) y valor con la traducción pretendida (todos han tener los mismos nombres de variable para la misma traducción) y mediante unos botones situados en la cabecera de la pagina cambiar el idioma de la web en cualquier momento sin necesidad de recargar la web.

Para su uso en las plantillas de los componentes se usa el código siguiente.

```
{{'variable'|translate}}
```

A este código se le pueden pasar variables mediante un objeto tras la llamada a `translate`.

```
{{'variable'|translate:({'var1':value})}}
```

Los ficheros han de tener el código del idioma al que pertenecen las traducciones de su interior, estos ficheros se encuentran en el directorio `assetsi18n`. El modulo se ha de configurar en el modulo principal de nuestra aplicación `app.module.ts` para así poder usado en todo el proyecto. Entre otras posibilidades nos permite seleccionar el idioma por defecto de nuestra web que en nuestro caso se ha indicado a Castellano (es).

Figura 5.2: Detalle de la selección de idioma

Esta configuración nos permite en el futuro añadir mas idiomas sin dificultad, solo hay que añadir el fichero json con las traducciones de las variables y añadir el correspondiente botón en la cabecera.

Frontend público

Como indica el titulo en esta sección trataremos la implementación de parte publica de nuestro proyecto. Todo lo incluido en esta parte de la web es de dominio publico y no requiere de acceso. Los componentes del frontend publico esta conectados mediante el enrutador principal este conectara los diferentes componentes públicos y la parte privada a través del componente `Login`.

Login

La función principal componente de `Login` Es permitir el acceso mediante usuario y contraseña a los componentes seguros de la aplicación. Otra función es crear nuevos alumnos en la aplicación, estos se darán de alta y posteriormente los profesores podrán asignarlos a equipos dentro de la asignaturas. La creación de un nuevo alumno se realizara mediante una ventana modal, los alumnos introducirán sus datos (nombre, apellidos, email, usua-

rio y contraseña). Como consideración el nombre de usuario no puede estar duplicado ni el correo estar registrado con anterioridad.

Figura 5.3: Vista del login

Legal notice

Este componente mostrará las consideraciones legales con respecto a nuestra web su visualización sera mediante una ventana modal.

Contributors

Este componente mostrará la información sobre los desarrolladores de la aplicación ademas de las tecnologías usadas su visualización sera mediante una ventana modal.

help

Este componente mostrará la ayuda de la web si el usuario no esta logeado mostrara la ayuda con respecto a la creación de un usuario y el login. En el caso de estar logeado mostrara la ayuda especifica para el tipo de usuario.

Frontend privado

Como indica el titulo en esta sección trataremos la implementación de parte privada de nuestro proyecto. Todo lo incluido en esta parte de la web requiere de acceso mediante logueo. La parte privada del frontend en primer lugar dispone del enrutador de la parte privada encargado de gestionar los distintos enlaces entre componentes, también dispone del servicio que gestiona las peticiones al servidor backend. También tiene el componente principal que acoge a los demás componentes y proporciona las utilidades básicas tales con la internacionalización o la vista del usuario. Disponemos de una serie de componentes que realizan las funcionalidades asociadas a la parte segura de la aplicación. Todas los componentes a excepción de los

dedicados al mantenimiento de la aplicación que solo son accesibles con un usuario administrador tienen dos formas definidas por el tipo de usuario.

TEACHER Profesores

STUDENT Alumnos

Subjects

Este componente contiene la visualización y gestión relativa a las asignaturas y los equipos que las forman. La vista esta dividida en primera instancia por el curso al que pertenecen las asignaturas en orden descendiente (primero la mas actual), dentro de cada pestaña se representan las asignaturas. Las asignaturas en el caso de los alumnos mostraran la información de los

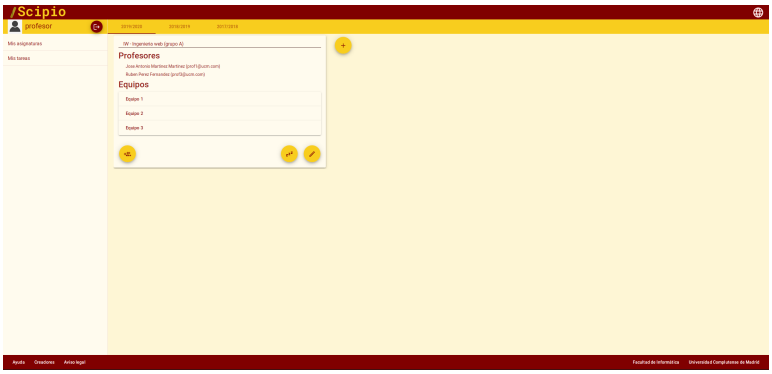


Figura 5.4: Ejemplo ventana asignaturas

profesores asociados a la asignatura y la del grupo al que pertenece.

Las asignaturas en el caso de los profesores mostrara la información de los profesores asociados y una lista desplegable con los equipos asociados a la asignatura. También dispone de las opciones gestión de los grupos y las asignaturas.

Gestión de equipos **Editar equipo** Permite editar el equipo, añadir y quitar alumnos del mismo

Eliminar equipo Elimina el equipo de la asignatura y libera a los miembros para poder ser asignados a otros equipos.

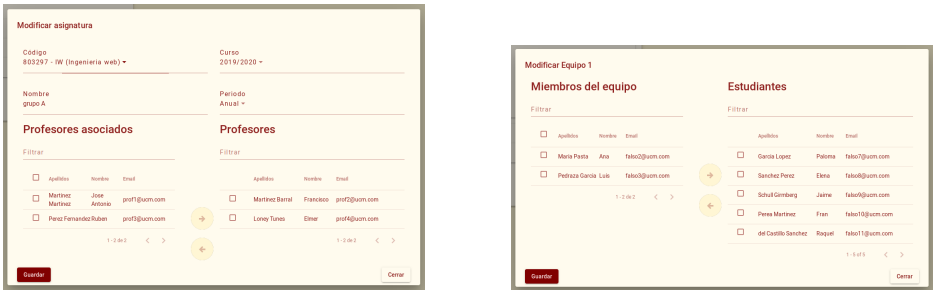
Gestión de asignaturas **Editar asignatura** Permite editar asignatura, añadir y quitar profesores, ademas de cambiar el nombre, periodo y demás datos.

Añadir equipo Permite añadir un equipo a la asignatura.

Parar asignatura Desactiva la visualización de una asignatura para los alumnos.

Activar asignatura Activa la visualización de una asignatura para los alumnos.

La gestión de las asignaturas y de los equipos se realiza mediante sendas ventanas modales. En ellas se podrán añadir o modificar las asignaturas o los equipos.



(a) Ventana de asignaturas (b) Ventana de equipos

Figura 5.5: Ventanas modales de asignaturas

Assignments

Este componente contiene la gestión de las tareas de las asignaturas, esta dividido en pestañas siendo cada pestaña una asignatura activa y en su interior contiene las tareas asignadas a ellas. Para ambos usuarios mostrara el titulo y la descripción de la practica y la fecha de entrega. Las tareas en el caso de los alumnos se activara la opción de visualizar las correcciones cuando el profesor suba los envíos.

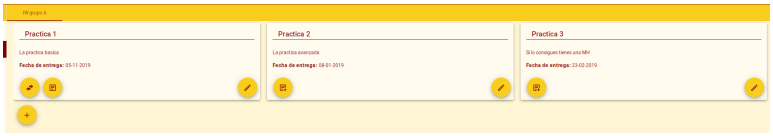


Figura 5.6: Ejemplo ventana tareas

Las tareas en el caso de los profesores dispondrá de las opciones de gestión de las tareas.

Crear tarea Permite al profesor añadir una tarea nueva a la asignatura, con un nombre, una descripción y una fecha de finalización.

Modificar tarea Permite modificar una tarea, su nombre, descripción y fecha de entrega.

Activar revisión ínter-pares Activa la revisión ínter-pares genera las correcciones cruzadas entre equipos.

Desactivar revisión ínter-pares Desactiva la revisión ínter-pares.

Ver envíos Abre la visualización de las entregas y las correcciones asociadas a estas.

La gestión de las tareas se realizan mediante una ventana modal. Esto permitirá cumplimentar o modificar los datos de una tarea.

Figura 5.7: Ejemplo gestión tareas

En el caso de los alumnos solo dispondrán del botón para visualizar las entregas cuando el profesor las suba.

Submissions

Este componente contiene la gestión de las entregas y las correcciones asociadas a ellas.Se mostraran todas las entregas en las que el usuario tenga presencia ya sea como corrector o como corregido. Las entregas en caso de los alumnos mostraran la información de su equipo la corrección del profesor sobre la entrega, ademas mostrara la corrección ínter-pares en el caso de estar activada por el profesor. También mostrara las correcciones ínter-pares a las que este asignado el alumno.

Las entregas en caso de los profesores mostraran la información del equipo al que pertenece la corrección y en el caso de estar activada la corrección ínter-pares también mostrara la información del equipo encargado de la tarea. En relación a las correcciones mostrara la que ha de realizar el profesor y todas aquellas correcciones ínter-pares asignadas a la entrega.

Las correcciones dispone de diferentes opciones según su estado.

Nueva Editar corrección Permite editar la corrección en el componente `Codemirror`.

Figura 5.8: Ejemplo gestión entregas

Figura 5.9: Ejemplo gestión entregas

En proceso Editar corrección Permite editar la corrección en el componente `Codemirror`.

Finalizar corrección Permite finalizar la corrección y generar el informe final de la misma.

Finalizada Ver corrección Permite ver la corrección en el componente `Codemirror`.

Ver PDF Abre el informe de corrección.

Codemirror

Este componente permite corregir el código de una entrega, nos muestra un árbol de directorios y ficheros donde están todos los ficheros accesibles, a su lado muestra la vista de corrección donde se mostrara el código del fichero seleccionado. Este componente nos permite crear y editar y borrar los comentarios y correcciones echas sobre el mismo. Este componente posee dos modos uno de edición solo accesible cuando se tiene permisos de edición y un segundo modo de solo visualización que permite ver los comentarios y correcciones realizados cuando se es el objetivo de la entrega o se es profesor.

Creación, modificación y eliminación de comentarios

Para que dichas funciones pudieran funcionar sobre nuestro framework `Angular` se ha usado la función `ngAfterViewInit` del componente que contiene nuestro visor de código, este nos permite mantener el código en ejecución tras la carga de la vista. Las funcionalidades introducidas se dividen en tres. La primera es la selección de las lineas del código a comentar pudiendo seleccionarse una sola linea o varias de ellas manteniendo el botón shift pulsado.

Figura 5.10: Ejemplo selección lineas

La segunda funcionalidad permite mediante el botón derecho del ratón abrir un menú contextual (`contextmenu`), el cual dependiendo de si las lineas ya están comentadas nos permitirá abrir la ventana de modificación u eliminar el comentario o si no tiene comentario crear uno nuevo.

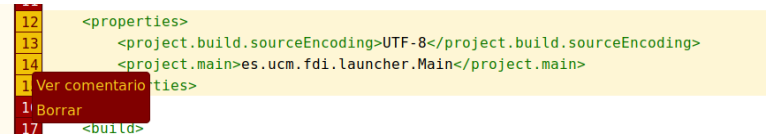


Figura 5.11: Ejemplo menú comentarios

La tercera funcionalidad es la visualización del código corregido en la parte inferior de la ultima linea de su comentario haciendo clic izquierdo con el ratón en las lineas comentadas.

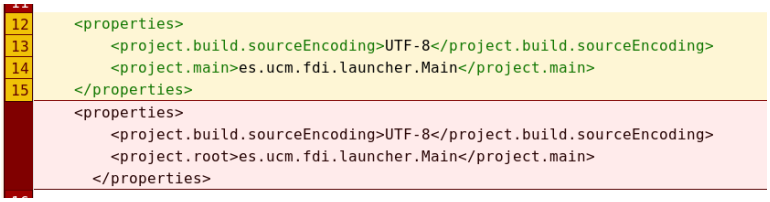


Figura 5.12: Ejemplo selección lineas

Para la creación o modificación de un comentario se ha implementado una ventana modal, que incluye todo lo necesario para realizar un comentario, así como dos ventanas de código para en caso de querer transformar

el comentario en una corrección se pueda modificar el código. En el caso de ser una modificación de un comentario ya existente también se mostrara un botón para eliminarlo.

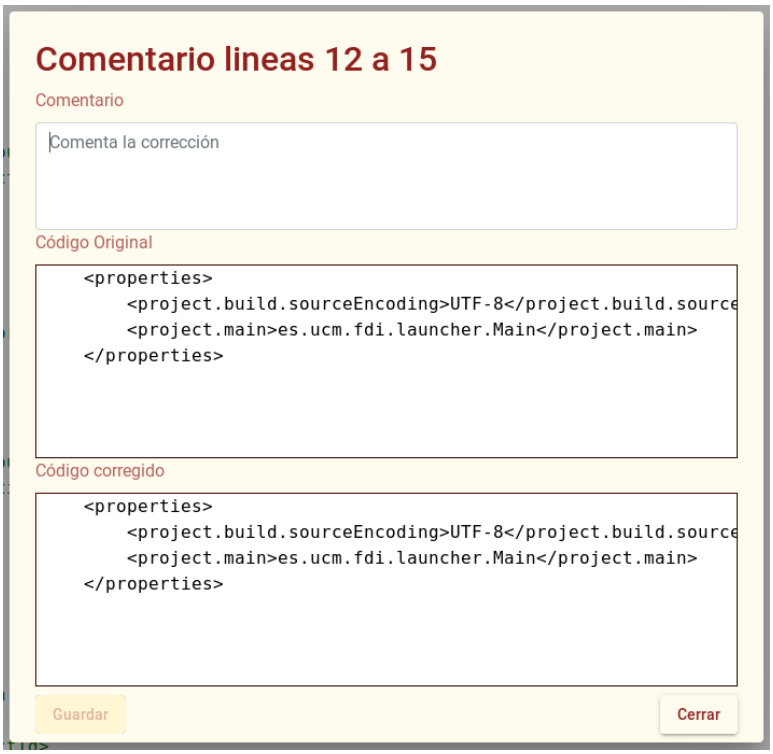


Figura 5.13: Ejemplo selección lineas

5.2.1. Administración

La implementación de la administración de la aplicación tiene dos partes diferenciadas, la administración de usuarios y la administración de asignaturas. Ambas solo son accesibles con un usuario de administración.

Administración de usuarios

Para la administración de los usuarios por parte del gestor o los gestores de la aplicación se ha implementado con una ventana que muestra mediante pestañas tres tablas con los usuarios de cada tipo (alumnos, profesores y administradores).

En cada una de estas pestañas se incluye un botón para añadir un nuevo usuario de ese tipo, ademas en ca fila de la tabla que corresponde a un usuario, se puede modificar su estado de actividad de activo a inactivo y

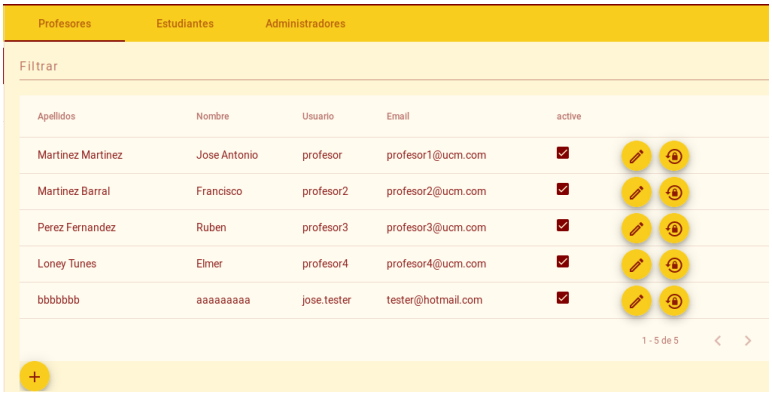


Figura 5.14: Ventana de gestión de usuarios

viceversa mediante un `checkbox`. Adicionalmente en cada fila tenemos dos opciones, modificar los datos del usuario y cambiar su contraseña.

Administración de asignaturas

Para la administración de los dos elementos que gestionan los administradores de la aplicación (códigos y cursos), se ha implementado con una ventana que muestra mediante dos pestañas, estos elementos.

La pestaña de gestión de códigos permite la creación y modificación de los códigos de asignatura, se ha implementado mediante el uso de una tabla en la que se muestran todos los códigos, en cada fila se dispone de un botón que permite la modificación del código. En la parte inferior de la tabla se muestra un botón que permite la creación de un nuevo código.

Figura 5.15: Ventana de gestión de códigos de asignatura

La pestaña de gestión de cursos permite la creación y modificación de

asignaturas, esta se ha implementado usando una tabla que muestra los cursos disponibles y su estado. El estado se puede modificar mediante un `checkbox` situado en cada fila, adicionalmente existe un botón que permite modificar el curso. En la parte inferior de la tabla existe un botón que permite añadir un nuevo curso a la aplicación que por defecto no estará activado.

Figura 5.16: Ventana de gestión de cursos

5.3. Arquitectura Interna

La aplicación tiene dos partes diferenciadas un servidor backend basado en `NodeJS` con `express` y un frontend basado en la arquitectura de `Angular` en su version mas reciente (actualmente 8).En relación a la persistencia de la información se ha optado por usar una base de datos basa en `MySQL`.

Figura 5.17: Esquema básico de la aplicación

5.3.1. Backend

La arquitectura interna del backend esta basada en un servidor nodeJS con express, esto nos permite crear la estructura básica para recibir la peticiones del frontend y responder a las mismas. Mas en profundidad tenemos varias partes diferenciadas.

Enrutadores Reciben la peticiones del del frontend y las derivan al controlador correspondiente dependiendo de la url recibida y mandan de vuelta la respuesta. Disponemos de dos, uno para la peticiones que no

requieren autenticación (publicas) y otro para aquellas que requieren seguridad (estar autenticado) para el acceso a la información.

Controladores Realizan las funciones principales del servidor, reciben los datos provenientes del enrutador, realizan los procesos necesarios y devuelven la respuesta para que sea envidada al frontend.Estos están divididos en función de las principales funciones del la aplicación, de esta forma se mejora la legibilidad del código y la escalabilidad de la aplicación.

Gestor de la BBDD Realiza las funciones respecto a la persistencia de datos. Sus principales funciones son.

- Conectar la BBDD con el servidor.
- La generación de las tablas así como de los datos básicos para el funcionamiento de la aplicación.
- Proveer a los controladores de las peticiones que necesiten para tratar con la BBDD.

En el backend también están los directorios donde se han de guardar los ficheros provenientes de los envíos asociados las las tareas, así como las imágenes de los usuarios. También disponemos de un sistema de logs basado en Winston (Charlie Robbins, 2019) que nos permite mantener un registro sobre las acciones y errores que se producen en el backend.

5.3.2. Frontend

La arquitectura del frontend esta marcada por la de Angular (Contributors, 2019), Esta esta basada en una serie de módulos, componentes y servicios, a los que se puede agregar módulos para el enrutado de la web (routing), así como otras API's externas.

Nuestra aplicación dispone de una serie de , servicios y componentes generales que proporcionan funcionalidades básicas a la aplicación en general.

notifications service Permite lanzar notificaciones mediante el uso de Toastr (Scott Cooper, 2019). Las notificaciones son traducidas al idioma actual de la aplicación.

lenguage module Modulo de configuración para la internacionalización de la web.

globals Contiene la declaración de una serie de variables básica de la aplicación.

app-routing module Enrutador principal de la aplicación

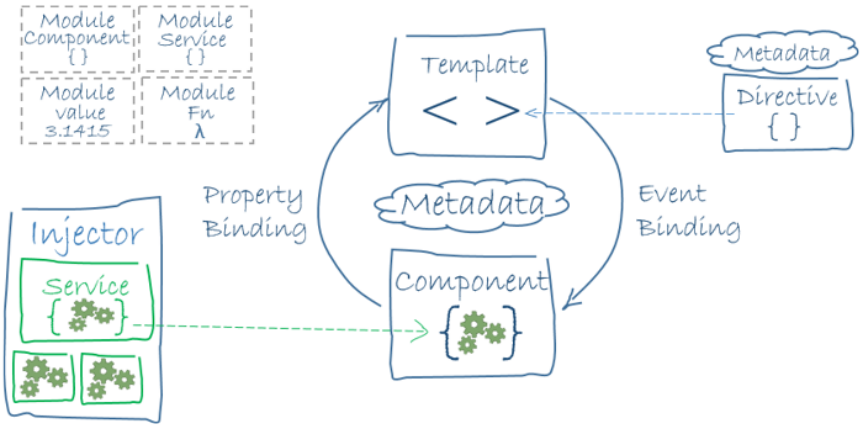


Figura 5.18: Arquitectura de Angular

app component ficheros ts, html y css encargados de la vista básica inicial de la aplicación.

angular-material module Configuración inicial del modulo material para su implementación en la web.

app module Configuración inicial de los módulos,encargado de importar los módulos, servicios y componentes, de este modulo heredan los demás módulos las importaciones de librerías básicas.

A parte de los elementos básicos descritos con anterioridad la aplicación esta divida en dos partes diferenciadas.

Public Parte publica de la aplicación que no requiere de autenticación, Pagina principal, login, ayuda,etc. A continuación listamos los módulos, componentes y servicios de los que esta compuesto.

- **Login** Componente del login de usuario
  - **Singup** Ventana modal que permite la creación de un usuario.
- **help** Ayuda de la aplicación contendrá distintos textos dependiendo de estado de la web y el usuario logeado.
- **legalNotice** Información legal de la aplicación.
- **contributors** Información de los creadores de la web y tecnologías usadas.

secure La parte privada, solo se puede acceder mediante un logeo previo, contiene el núcleo de las funcionalidades de nuestra web.

- **secure-routing module** Enrutador de la parte privada de la aplicación separado del principal por seguridad.

api service Proporciona a los componente una serie de funciones para la conexión con el servidor backend.

interfaces model Proporciona interfaces con el formato de los elementos de las tablas de persistencia.

secure module Modulo de la parte privada, encargado de importar los módulos, servicios y componentes necesarios.

secure component ficheros ts, html y css encargados de la vista principal de la parte segura, proporciona la barra de navegación y la vista indicada por defecto al entrar.

subjects component componente encargado de la gestión de las asignaturas y grupos asignados a ellas.

- **edit-team modal** Ventana modal que permite crear o modificar un equipo de una asignatura.
- **edit-subject modal** Ventana modal que permite crear o modifica una asignatura.

Assigments component **upgrade.assignment modal** Ventana modal que permite crear o modificar una tarea.

- **upload-submissions modal** Ventana modal que permite subir las entregas de una tarea.

submissions component Componente encargado de mostrar las entregas y las correcciones correspondientes a cada una

codemirror component Componente que muestra una corrección, con el árbol de ficheros de la entrega y el visor de código. permite la realización de los comentarios y correcciones sobre el código.

save-dialog modal Ventana modal encarga de preguntar al usuario en el caso de que se deje algún comentario o corrección sin guardar.

confirmation-dialog ventana modal que permite establecer un control antes de realizar una acción que no es reversible.

admin-codes Componente encargado de la gestión de las asignaturas por parte del administrador.

- **code-modal** Ventana modal que permite la creación o modificación de un código de asignatura.
- **edition-modal** Ventana moda que permite la creación o modificación de un curso.

admin-users Componente encargado de la gestión de los usuarios por parte del administrador.

- **create-user** Ventana moda que permite la creación de un usuario con el tipo especificado cuando se abre la ventana.
- **user-modal** Ventana moda que permite la modificación de un usuario.

change-password Ventana modal modal que permite el cambio de contraseña de un usuario.

profile Este componente permite la gestión por parte del usuario de su perfil, incluyendo el cambio de contraseña.

change-profile-pwd Ventana modal que permite el cambio de la contraseña por parte del usuario.

disable-control-directive Directiva que introduce la posibilidad de poder activar o desactivar un campo de un formulario de forma dinámica.

validators Este directorio contiene los validadores de los formularios customizados.

code-unique Permite validar que un `código GEA` de asignatura sea único en la base de datos.

edition-unique Permite validar que una edición de un curso sea único en la base de datos.

email-unique Permite validar que un email de usuario sea único en la base de datos.

username-unique Permite validar que un nombre de usuario sea único en la base de datos.

match-value Compara los campos de `contraseña` y `confirmar contraseña` y devuelve si son iguales o no.

5.3.3. Persistencia

La persistencia basada en una base de datos sobre un servidor MySQL, esta distribuida en varias tablas relacionadas entre si. Se ha procurado realizar una normalización de la misma manteniendo hasta la tercera forma normal [1].

[1] Ver normalización Wikipedia (2018).

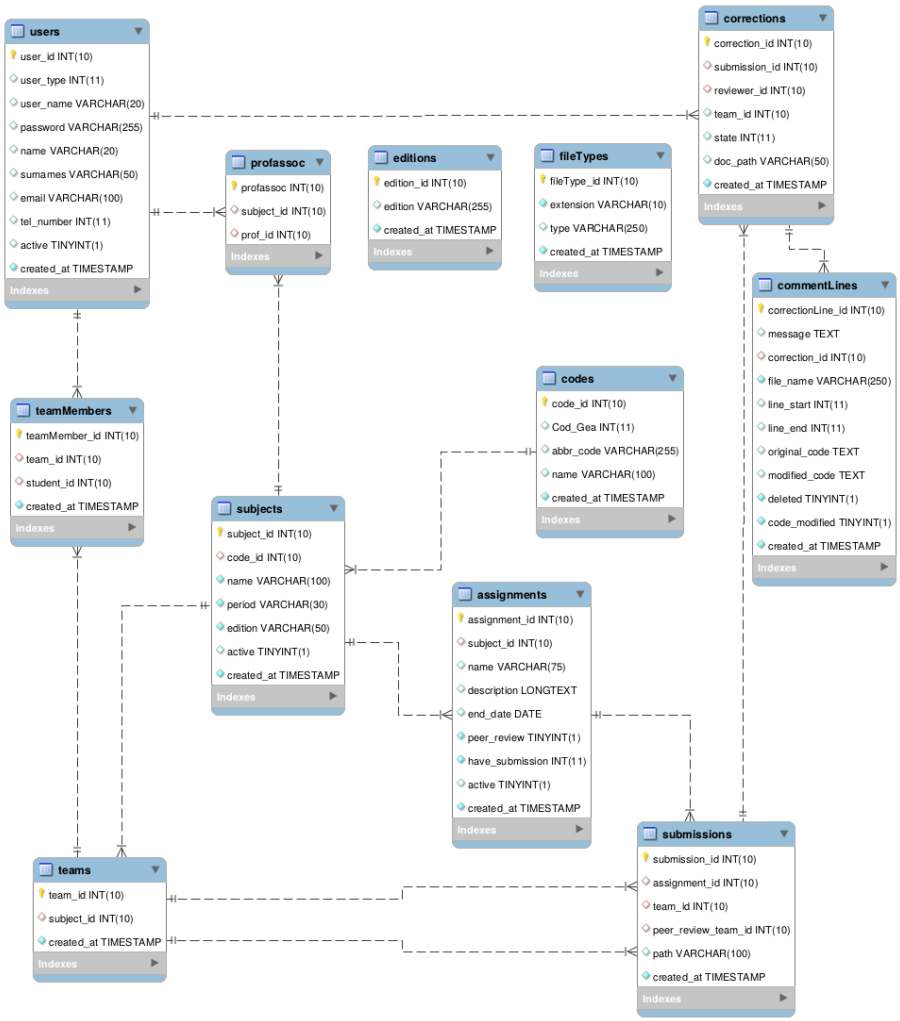


Figura 5.19: Esquema E/R de la base de datos

Capítulo 6

Resultados

"Si buscas resultados distintos, no hagas siempre lo mismo"
— Albert Einstein

Scipio es en una aplicación pensada para resolver un problema persistente en la enseñanza de la programación: La corrección de las prácticas siempre ha sido un proceso lento, y ha presentado problemas prácticos a la hora de dar buen feedback a alumnos.

En nuestra aplicación intentamos resolver ambos aspectos del problema, mediante un sistema interactivo de corrección en el que los profesores pueden corregir las practicas de código de forma sencilla, mientras los alumnos pueden ver dichas correcciones en contexto y sin tener que referenciarlas contra el código que entregaron originalmente.

Adicionalmente nuestra aplicación incorpora un sistema de corrección ínter-pares que permite la corrección de las practicas entre los propios alumnos mejorando la comprensión del código por parte de éstos, así como su capacidad crítica.

También incluye un sistema que permite gestionar todo lo relacionado con las asignaturas, tareas, entregas y correcciones que esperamos sea de uso intuitivo y fácil para los profesores.

El resultado final constituye una aplicación que, en su estado actual, ha superado las pruebas iniciales en un entorno controlado y está preparada para realizar las primeras pruebas de despliegue en un entorno real con alumnos y profesores. Este despliegue se podría realizar en la asignatura de IW (Ingeniería web) del curso 2019/20, de la que es profesor el director de este trabajo.

La aplicación web se ha desarrollado usando tecnologías punteras y mejores prácticas de programación, lo cual ha supuesto un importante beneficio desde el punto de vista de formación y aprendizaje.

Tecnologías usadas en la implementación

- Angular 8 - la última versión estable
- Express con NodeJS - para el backend de la aplicación
- Knex - para mapeo entidad-relación y acceso a la base de datos MySQL
- MySQL server - como base de datos, proporcionando persistencia a la aplicación
- CodeMirror - no solo usado, sino también extendido para permitir anotar código y mostrar anotaciones
- bash - para el script de gestión y monitorización de Scipio

Tecnologías usadas como soporte para el desarrollo

- Git - para control de versiones
- BitBucket - repositorio git en la nube
- pm2 - gestión de servidores

Tecnologías usadas en la planificación y la documentación

- LaTeX, y el paquete TeXiS para la documentación; incluyendo bibtex, listings, y otros múltiples paquetes.
- Project Libre para la planificación
- Google Docs para coordinación
- Gimp para edición de imágenes

Esfuerzo total de código, según la herramienta `cloc`[1], con la siguiente línea de comando (es decir, sin incluir documentación o librerías externas):

```
cloc conf scripts server/controllers server/i18n server/
   *.js server/DDBB server/routes README.md server.js
   src Tranlations.babel
```

La salida de este comando[2] es:

```
  217 text files.
  217 unique files.
   45 files ignored.
```

[1] Ver `cloc` (Al Danial, 2015)

[2] En la distribución se incluyen ficheros de prueba (pensados para mostrar su contenido y poder dejar comentarios sobre éste) dentro de SERVER/FILES, que no obstante no forman parte de Scipio en sí. De ahí lo detallado de la línea de comando.

```
github.com/AlDanial/cloc v 1.74  T=0.32 s (578.7 files/s, 34448.1 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                     108            399            226           4578
JavaScript                      21             49             86           2142
HTML                            26             25             66           1524
Sass                            21             63             26           1016
JSON                             7              0              0            470
Markdown                         1             73              0            167
Bourne Shell                     2              7              0            155
-------------------------------------------------------------------------------
SUM:                           186            616            404          10052
-------------------------------------------------------------------------------
```

Como se puede observar, la mayoría del código es TypeScript ó JavaScript, con algo de tecnologías web suplementarias (html, plantillas CSS), JSON para ficheros de configuración, Shell (bash) para administración del servidor y Markdown para instrucciones de instalación del proyecto.

Capítulo 7

Conclusiones y Trabajo Futuro

"No os diré no lloréis, pues no todas las despedidas son amargas"
— Gandalf el Blanco (J.R.R. Tolkien)

7.1. Conclusiones

Este trabajo buscaba desarrollar una aplicación para cubrir una importante laguna en la enseñanza de programación de la Facultad de Informática de la UCM: facilitar las correcciones de código, tanto a la hora de realizarlas por parte de los profesores como a la de comunicar sus resultados a los alumnos una vez finalizadas.

Scipio cumple estos objetivos, y además permite realizar correcciones ínter-pares, que pueden mejorar la capacidad de los alumnos para analizar y entender código ajeno, y usar esta capacidad de autocrítica para mejorar el propio, al comparar y ver comparados sus desarrollos y los de sus compañeros.

7.2. Lineas futuras

En el futuro de nuestra aplicación se pueden definir varias líneas de actuación.

- Hay multitud de pequeñas mejoras de usabilidad que pueden llevarse a cabo. Un pequeño estudio piloto con usuarios reales revelaría las más importantes. Además:
- Se podría conectar Scipio con un sistema anti-copia, que analice las diferentes entregas de una misma tarea y compruebe similitudes entre ellas para poder detectar posibles casos. Como ejemplo de esta funcionalidad tenemos la aplicación `ac2`[1].

[1] Disponible en `https://github.com/manuel-freire/ac2`

- Otra línea de actuación pasaría por realizar una integración con el Campus Virtual moodle de la UCM mediante una pasarela que permita la descarga automática de las entregas y las suba sobre Scipio, incorporando luego las revisiones al CV para que se puedan ver sin salir nunca de éste.
- Por último, otra línea de actuación pasaría por implementar un sistema de mensajería de preguntas sobre las correcciones del código entregado por parte de los alumnos y con respuesta de los profesores.

Chapter 8

Conclusions and Future Work

"I will not say: do not weep; for not all tears are an evil."
— Gandalf the White (J.R.R. Tolkien)

8.1. Conclusions

This work sought to develop an application to fill an important gap in the teaching of programming of the Faculty of Computer Science of the UCM: to facilitate the corrections of code, both when teachers perform those reviews and when students look at the results once reviews are finished.

Scipio meets these objectives, and also allows peer review corrections, which can improve the ability of students to analyze and understand foreign code, and use this capacity for self-criticism to improve their own, by being able to compare and see comparisons of their developments and those of their peers.

8.2. Future lines of work

Several future lines of work can be envisioned:

- There are many small usability improvements that can be made. A small pilot study with real users would reveal the most important ones. Additionally,
- Scipio could be connected to an anti-copy system, which analyzes the different deliveries of the same task and checks similarities between them in order to detect possible cases. As an example of this functionality we have the application `ac2`[1].

[1] Available at `https://github.com/manuel-freire/ac2`

- Another line of action would involve an integration with the Moodle Virtual Campus of the UCM, through a gateway that allows automatic downloading of deliveries and uploads on Scipio, then incorporating revisions to the CV so that they can be seen without ever leaving this.
- Finally, another line of action would be to implement a messaging system of questions about the corrections of the code delivered by the students and with the response of the teachers.

Apéndice A

Documentación de Usuario

A.1. Inicio de sesión y registro de alumnos

Cuando un usuario entra en la aplicación puede iniciar sesión o crear un nuevo alumno.

El login

Para acceder a la funcionalidad principal de la aplicación se requiere de un usuario y contraseña valida. El login se puede realizar desde la pagina principal. El usuario ha de introducir su usuario y su contraseña y pulsar el

Figura A.1: Vista del login

botón de `Acceder`, El sistema en caso de se correcto dará al usuario acceso a sección segura de la aplicación según el tipo de usuario que sea. En el caso de ser incorrecto mostrara un mensaje de error.

Crear una cuenta de alumno

En la ventana principal también se puede registrar un nuevo alumno, pulsando el botón de `Nuevo alumno` este le dará acceso a la ventana de

registro. El nuevo alumno deberá introducir, su nombre y apellidos, un email,

Figura A.2: Vista del registro de un alumno

un nick de usuario y una contraseña. Tanto el email como el nick de usuario deben de ser únicos si no lo son el sistema mostrara una alerta.

A.2. Manual de Profesor

En este manual presentaremos la funcionalidad de la aplicación desde el punto de vista del profesor, comentaremos todas las posibilidades de la que dispone y como usarlas acompañadas de ejemplos gráficos. En primer lugar para poder acceder como profesor se ha de tener una cuenta como tal esto se ha de solicitar a un administrador de la aplicación y este le proporcionara el acceso. Una vez realizado el login, se mostrara la pantalla principal del la sección segura de la web con la gestión de las asignaturas.

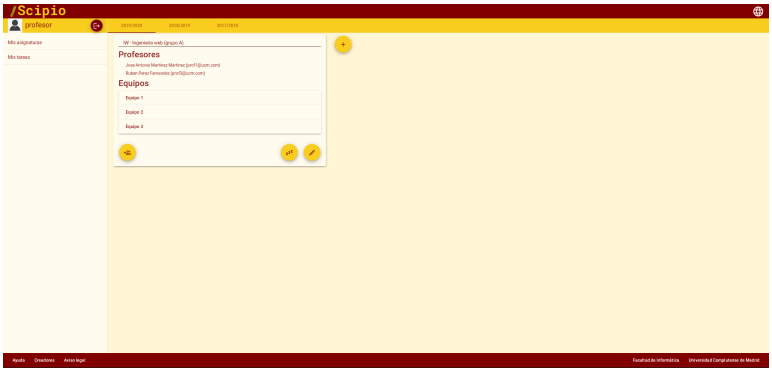


Figura A.3: Ventana principal del profesor

Gestión de las asignaturas

La gestión de las asignaturas permite a los profesores administrar todo lo relacionados con ellas. Se puede acceder a ella desde la barra de navegación lateral pulsando en `Mis asignaturas`. La vista esta dividida en primera ins-

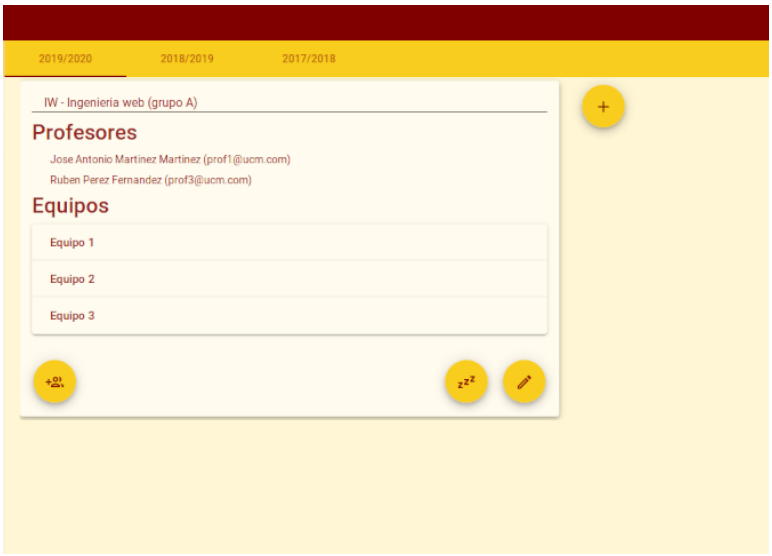


Figura A.4: Gestión de asignaturas

tancia por cursos, siendo el mas reciente el que esta en primer lugar. Dentro de cada curso se muestran las asignaturas,ademas de un botón que permite añadir una nueva asignatura siempre que el curso este activo [1]. Las asignaturas disponen de varias opciones, la primera es la activación o desactivación

[1]La activación y desactivación de un curso depende los administradores de la aplicación

de la asignatura esto afecta a la visualización de la misma por parte de los alumnos asignados a los equipos, por defecto cuando se crea una asignatura estará desactivada y no se podrá activar hasta que se introduzca al menos un equipo. Para la creación o modificación de una asignatura se podrá realizar mediante una ventana modal que se abre al solicitar esta acción. Esta ventana

Figura A.5: Ventana de creación o modificación de una asignatura

nos permite seleccionar el código de la asignatura, así como el curso y el periodo de introducir el nombre de la asignatura. En la parte inferior se podrán seleccionar los profesores asociados a la asignatura partiendo de un listado de los profesores disponibles. Si es de nueva creación por defecto aparecerá como profesor asociado el profesor que solicite la creación. Las asignaturas

tienen asignados equipos de alumnos, estos se muestran en un listado desplegable con su información y acciones disponibles. Los equipos están formados

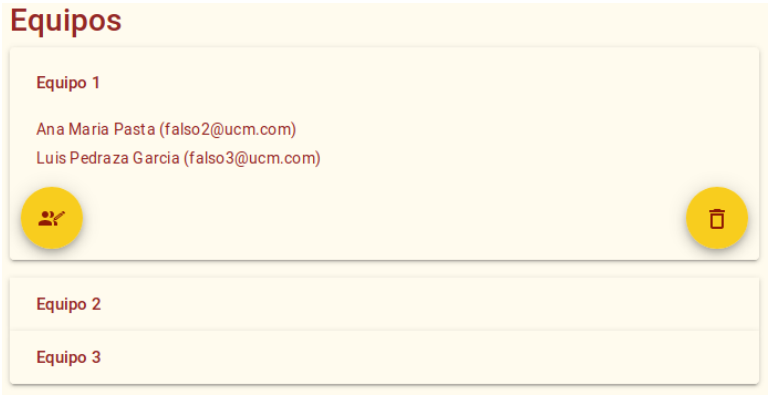


Figura A.6: Gestión de equipos

por alumnos, un mismo alumno no puede pertenecer a dos equipos dentro de una misma asignatura. La creación o modificación se realizan mediante una ventana modal que se abre al solicitar alguna de las acciones nombradas. En

Figura A.7: Ventana de creación o modificación de un equipo

la ventana podemos añadir y quitar alumnos del grupo, si el grupo ya tiene entregas y correcciones, las modificaciones en los alumnos serán efectivas a partir de las siguientes acciones que realice el equipo.

Gestión de las tareas

A la gestión de las tareas se accede desde el la barra de navegación lateral pulsando `Mis tareas`. esta ventana de la aplicación esta dividida en primera instancia en pestañas, cada una corresponde a una asignatura, estas a su vez contienen las tareas que están asignadas a esa asignatura.

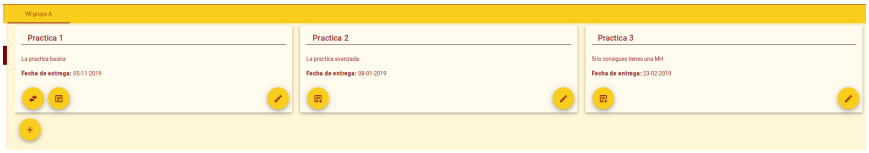


Figura A.8: Gestión de tareas

Las tareas pueden crearse y modificarse mediante una ventana modal. Las tareas están compuestas por un titulo una descripción y una fecha de entrega. Cuando se crea una tarea por defecto esta desactivada, se puede

Figura A.9: Ventana de creación o modificación de una tarea

activar cuando el profesor lo considere necesario. Una vez este activada la tarea tiene dos estados. El primero es cuando aun no se han subido las entregas y otro cuando se han subido. En ambos casos se puede modificar la tarea, cuando esta en modo sin entregar permite al profesor subir la entregas en un formato de zip de zips[2]. en este los zips de las entregas de los alumnos deberán estar identificados con el Id del equipo al que pertenece la entrega.

[2]Este es el formato de descarga de campus virtual moodle

Figura A.10: Ventana de subida de una entrega

Si ya se ha realizado la subida se activara la opción de verlas en la vista de `Entregas`. ademas el profesor podrá activar y desactivar la corrección ínter-pares.

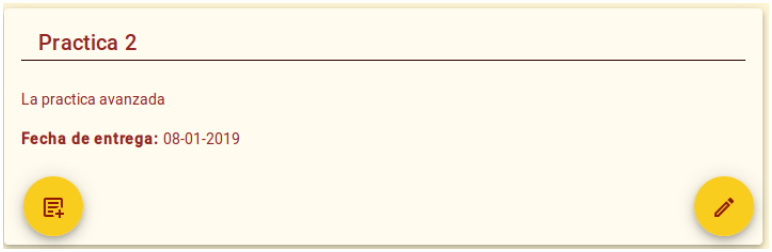


Figura A.11: Ejemplo de tarea sin entregas

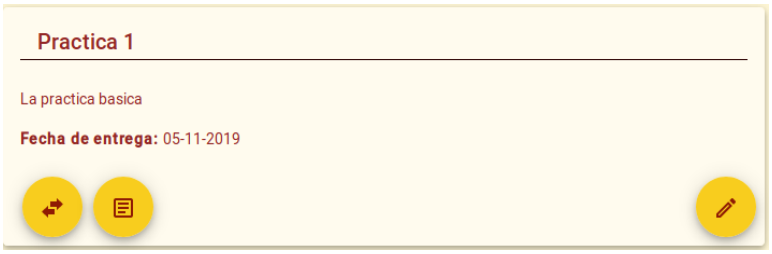


Figura A.12: Ejemplo de tarea con entregas

Cuando se activa la corrección ínter-pares se crean tantas correcciones como sean necesarias realizando una permutación aleatoria entre los equipos y generando una corrección por cada alumno del equipo.

Gestión de entregas y correcciones

La gestión de entregas y correcciones se accede desde cada tarea una vez realizada la subida de las entregas. La aplicación mostrara todas las entregas realizadas. Cada entrega mostrara la información del equipo que ha realizado

Figura A.13: Gestión de entregas y correcciones

la entrega y la corrección del profesor.

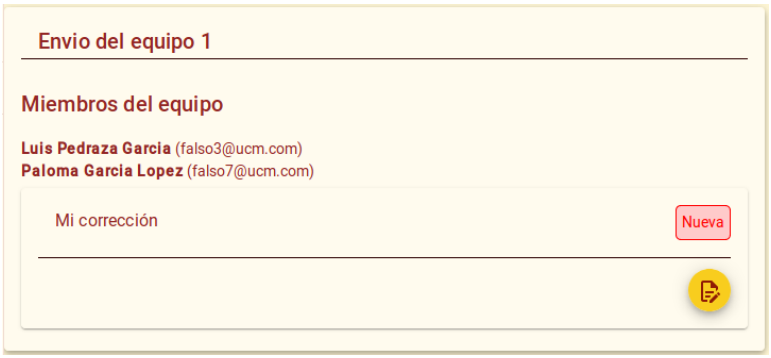


Figura A.14: Vista de una tarea

Cuando se ha activado la corrección ínter-pares en las entregas también aparecerá la información del equipo encargado de realizar la corrección sobre esa entrega, ademas se añadirán las correcciones de los miembros de equipo en modo de visualización.

Figura A.15: Vista de una tarea con ínter-pares

Las correcciones tienen tres posibles estados nueva, en proceso y terminada, cada una con diferentes acciones en cada uno. Esto se indica el la parte superior derecha de cada corrección. Las correcciones se pueden modificar cuando su estado es nueva o en proceso, ademas cuando esta en proceso adicionalmente se puede finalizar la corrección. Una vez se finaliza una corrección el sistema genera un informe pdf con los comentarios y las correcciones hechas sobre el código. Este informe estará accesible mediante un botón en la corrección.

Las correcciones si están siendo corregidas o ya se ha finalizado la corrección mostraran el numero de comentarios hechos y si se expande mostrara esta información repartida entre los diferentes fichero que contengan algún comentario o corrección.

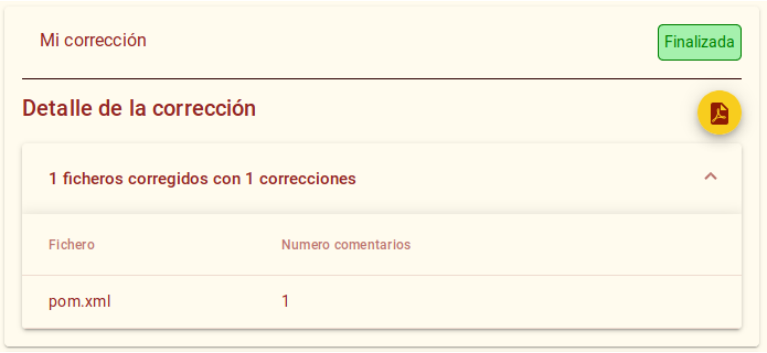


Figura A.16: Vista del detalle de una corrección

A.3. Manual de Alumno

En este manual presentaremos la funcionalidad de la aplicación desde el punto de vista del alumno, comentaremos todas las posibilidades de la que dispone y como usarlas acompañadas de ejemplos gráficos. Los alumnos cuando iniciar sesión en la vista principal verán las asignaturas en las que participan, estas estas divididas en primera instancia por cursos siendo el mas reciente el primero.

Figura A.17: Vista del detalle de una corrección

Esta cada pestaña de curso esta dividida en asignaturas, cada una le proporciona al alumno la información sobre los profesores asociados a la asignatura y los miembros del equipo al que pertenece en la asignatura.

Las Tareas el alumno las puede ver pulsando `Mis tareas`, Estas están divididas en pestañas por asignatura. Las tareas mostraran su información, titulo, descripción y fecha de entrega.

Si los profesores han subido las entregas de la tarea esta mostrara el un botón para poder acceder a la sección de `Entregas`.

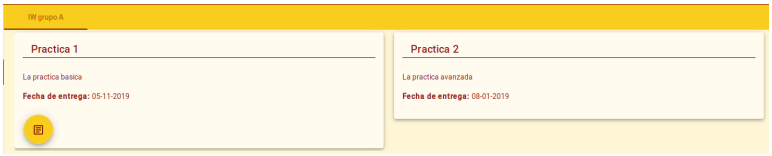


Figura A.18: Vista del detalle de una corrección

Cuando el alumno entre en las entregas de una tarea podrá ver su entrega con la corrección del profesor, ademas si esta activa la opción de ínter-pares se mostraran las correcciones que se estén haciendo sobre su practica y aquellas que tenga que hacer el alumno sobre las entregas de otros y poder ver las de sus compañeros de equipo.

Figura A.19: Vista del detalle de una entrega

A.4. Manual de corrección

La corrección de una practica es la base de nuestra aplicación en este manual explicaremos como poder realizar una. La vista principal se compone de un árbol donde se mostrara la estructura de la entrega con los ficheros disponibles para corregir y de un visor de código donde se mostrara el contenido del fichero seleccionado.

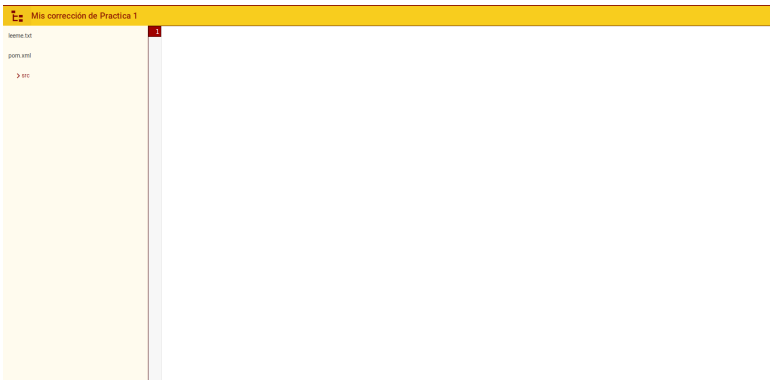


Figura A.20: Vista principal de corrección

El primer paso para corregir es seleccionar las filas que se desea comentar, se puede seleccionar una sola linea o varias seguidas, para seleccionar pulsamos sobre los números de linea y mantenemos `Shift` pulsado para seleccionar un segundo numero de linea si queremos hacer multiselección.

Figura A.21: Vista lineas seleccionadas

Una vez seleccionada pulsando botón derecho sobre la selección nos dará la opción de crear un nuevo comentario.

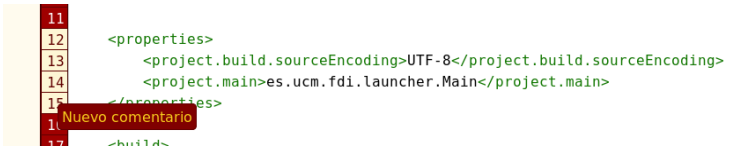


Figura A.22: Vista lineas seleccionadas

Al pulsar en `Nuevo comentario` se abrirá la ventana modal para crear un nuevo comentario.

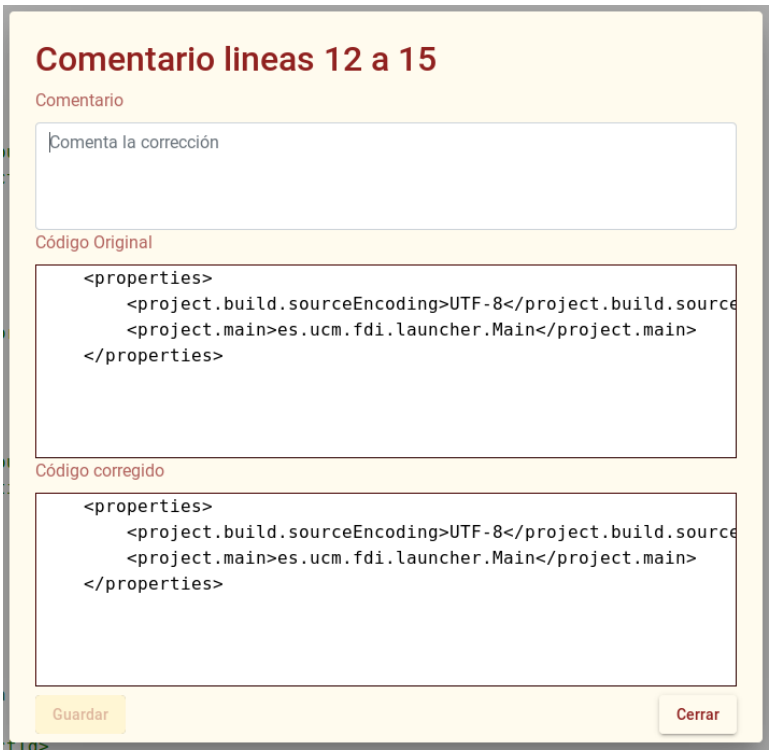


Figura A.23: Vista lineas seleccionadas

En esta ventana se muestra todo lo necesario para realizar un comentario o corrección. En el titulo nos mostrara la o las lineas que estamos corrigiendo, después podremos añadir el comentario en la caja de texto.En la parte inferior se muestra dos ventanas de código, la primera muestra el código original y la segunda nos permite modificar el código. Si se modifica el código pasa de ser un comentario a ser una corrección, esto luego se vera reflejado en el informe final. Una vez guardamos el comentario queda marcado en el código.

Figura A.24: Vista lineas comentadas

Si se ha hecho una corrección (Se ha modificado el código) si se pulsa

sobre las lineas corregidas se mostrara en la parte inferior el código corregido, y si se vuelve a pulsar se ocultara.

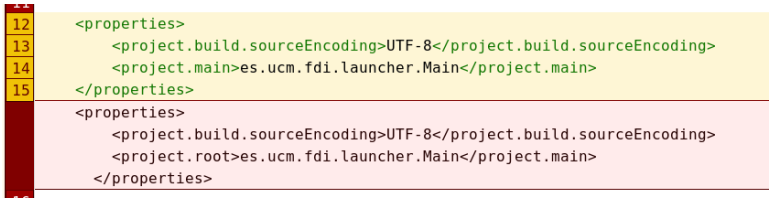


Figura A.25: Vista lineas comentadas y código corregido

Si se pasa el ratón por encima de los números de linea que estén comentados se mostrara un mensaje con el texto del comentario de esas lineas. Si pulsamos botón derecho sobre los números de linea ya comentado se nos abrirá un menú que nos mostrara dos opciones, Ver comentario y borrar.

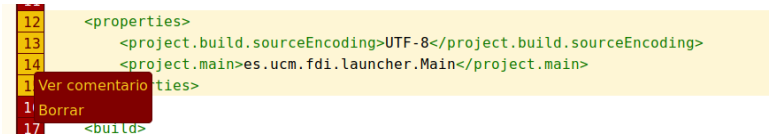


Figura A.26: Vista opciones lineas comentadas

Si pulsamos sobre la opción de `Ver comentario` Se nos abrirá la ventana de modificación del comentario, en ella se podrá modificar el comentario y la corrección del código. Si pulsamos sobre la opción de `Ver comentario` se eliminara el comentario.

A.5. Manual de Administrador

Los administradores disponen de dos funcionalidades principales, la gestión de usuarios y la gestión de asignaturas. Estas funcionalidades se han de manejar con cuidado dado que afectan al funcionamiento general de la aplicación. Una vez se realiza el login con el usuario de administración de mostrara la ventana principal.

Figura A.27: Vista principal del administrador

Gestión de usuarios

La gestión de los usuarios permite al administrador realizar la gestión y el mantenimiento general de los mismos, esto incluye crear y modificar usuarios, activar y desactivarlos, ademas de cambiar sus contraseñas. El administrador es el único que puede crear cuentas de profesor y de administrador.

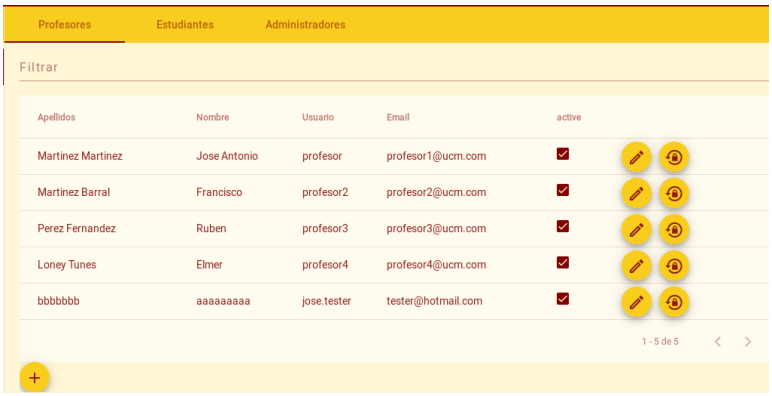


Figura A.28: Vista administración de usuarios

La ventana esta dividía en tres pestañas una por cada tipo de usuario, estas mostraran una tabla con cada tipo de usuario y un botón que permite

añadir un nuevo usuario de ese tipo.

En las tablas se mostraran los nombres y apellidos, el usuario y el correo de mismo ademas de su estado y las acciones que se pueden realizar sobre el. Mediante un `checkbox` se mostrara el estado del usuario este se puede activar o desactivar para cambiar el estado del usuario de activo a inactivo. También se pueden realizar varias acciones sobre el.

La creación de un nuevo usuario se realiza mediante una ventana modal que se abre al pulsar el botón de nuevo usuario.

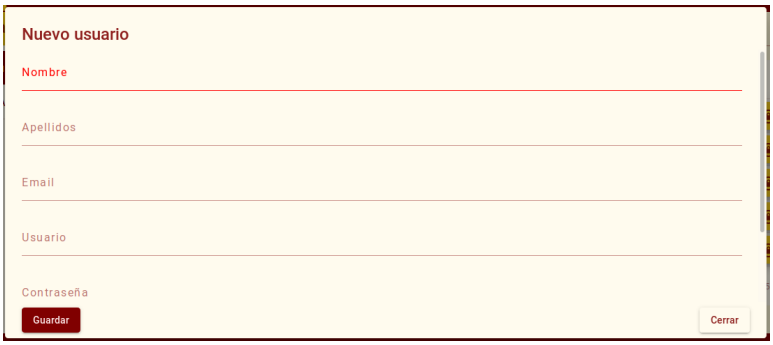


Figura A.29: Ventana nuevo usuario

En esta ventana se pueden introducir los datos del nuevo usuario, el sistema controlara de que tanto el nombre del usuario y el email sean únicos en la base de datos ademas la contraseña ha de tener al menos 8 caracteres y una letra mayúscula, una letra minúscula y un numero.

Modificar el usuario

Cuando seleccionamos esta opción se nos abrirá una ventana modal.

Figura A.30: Ventana de modificacion de un usuario

En esta ventana se puede modificar los datos principales del usuario, nombre, apellidos, nombre de usuario y el email. El sistema controlara de que tanto el nombre del usuario y el email sean únicos en la base de datos.

Cambiar la contraseña

Cuando seleccionamos esta opción se nos abrirá una ventana modal.

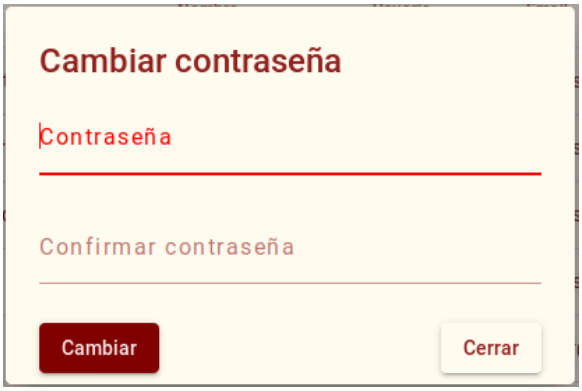


Figura A.31: Ventana de cambio de contraseña

En esta ventana se podrá cambiare la contraseña de un usuario, esta ha de tener al menos 8 caracteres y una letra mayúscula, una letra minúscula y un numero.

Gestión de asignaturas

La gestión de asignaturas consta de dos partes, la gestión de los códigos de asignatura y la gestión de los cursos. Esto se muestra en dos pestañas.

Figura A.32: Ventana de gestión de asignaturas

Gestión de códigos

La gestión de `códigos` o `asignaturas` permite la gestión de los elementos básicos de una asignatura que luego los profesores usaran para crear sus grupos. Un código esta formado por un **código GEA**, una abreviatura del nombre y el nombre de la asignatura. Estos se pueden crear pulsando el botón de añadir situado en la parte inferior de la tabla.

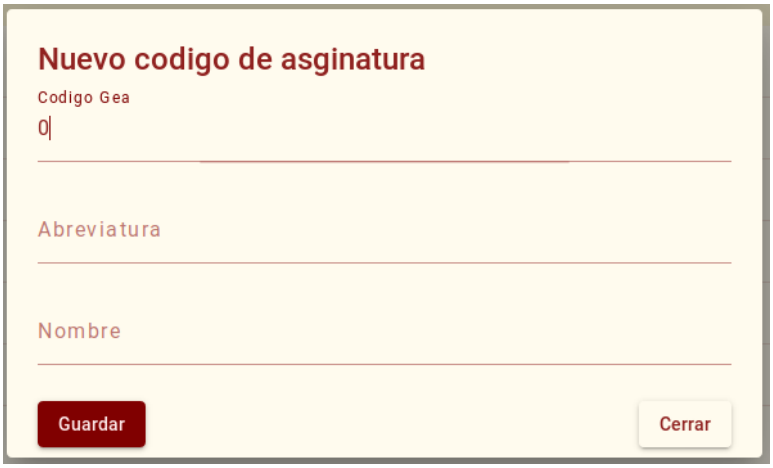


Figura A.33: Ventana de gestión de asignaturas

En tabla también se podrá modificar un código ya existente pulsando el botón de modificar en cada fila.

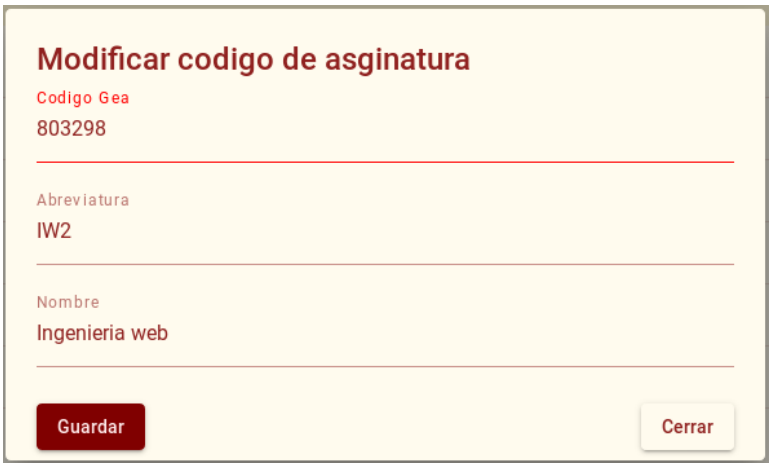


Figura A.34: Ventana de gestión de asignaturas

Los `códigos GEA` han de ser únicos estos se pueden ver en la web de la universidad complutense.

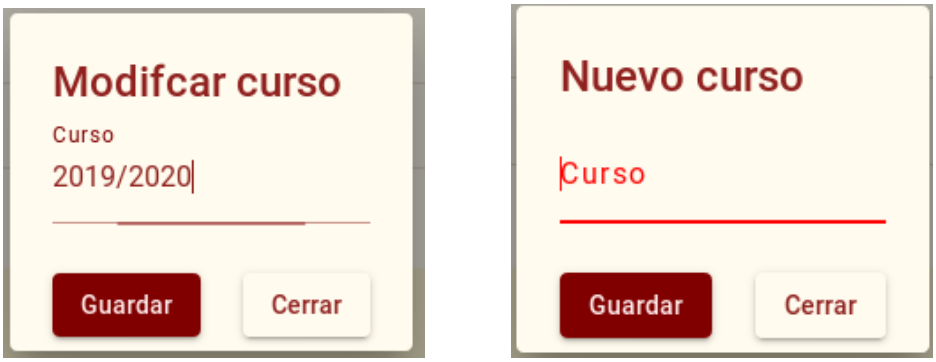
Gestión de cursos

Los cursos se componen de una edición por ejemplo 2019/2020 y un estado `actual` que marca si los grupos, equipos,etc, vinculados a ese curso pueden modificarse o son solo de visualización.

Figura A.35: Ventana de edición cursos

La creación de un curso se realiza mediante el botón situado en la parte inferior, que nos mostrara la siguiente ventana. Los cursos han de seguir el formato `yyyy/yyyy` (**yyyy** es un año con cuatro cifras, por ejemplo, 2019 o 2020) y ademas ha de ser único. cuando se crean por defecto están desactivados, se pueden activar en la tabla usando el `checkbox` de la fila correspondiente. De la misma manera se pueden desactivar. También se puede modificar

un curso ya existente siempre cumpliendo el formato citado anteriormente.



(a) Ventana de modificación de un curso (b) Ventana de creación de un curso

Figura A.36: Ventanas modales de cursos

Apéndice B

Documentación de Desarrollador

En este apéndice trataremos, algunos temas relacionados con el mantenimiento de la aplicación y actualización de algunas partes del código. En particular, hay dos extensiones que resultan relativamente fáciles de realizar sin apenas tener que tocar código:

1. Añadir nuevos lenguajes de programación al componente de visión/edición de código (CodeMirror).

2. Añadir nuevas traducciones a la interfaz, para que se pueda ver en otros idiomas.

B.1. Añadir nuevos lenguajes de programación

Añadir nuevos lenguajes al visor de código, con el tiempo puede ser algo necesario, la programación avanza a una velocidad muy alta y los lenguajes que hoy en día son los mas usados en el futuro puede que sean reemplazados por otros. Por ello a continuación explicaremos como añadirlos a nuestra aplicación.

Lo primero que debemos hacer es comprobar si nuestro visor soporta el nuevo lenguaje, para ello comprobaremos la lista de la **librería CodeMirror** donde se indica esto. Una vez comprobado tenemos que asegurarnos de que tenemos la ultima version de la librería en el sistema para ello lanzaremos el comando `npm install codemirror@latest -save` sobre la carpeta donde tengamos el código del proyecto. El siguiente paso es importar las librerías que necesitemos, para ellos abrimos el fichero `codemirror.component.ts` situado en el componente `codemirror` de la carpeta `secure` en nuestro proyecto de angular. En la parte superior están las importaciones de los modos, ahí debemos añadir nuestros nuevos lenguajes con la misma estructura, sustituyendo los asteriscos por la ruta del fichero de nuestro lenguaje:

```
import "codemirror/mode/***/***"
```

Esta misma operación hay que repetirla en el fichero `comment-line.component` situado en la carpeta `comment.modal` en el componente `codemirror`.

Por ultimo hay que añadir la correspondencia entre el lenguaje y sus extensiones esto de realiza en la tabla `filetypes` insertando una linea por cada extension nueva y su modo [1].

Con estos pasos la aplicación estará preparada para leer los ficheros del nuevo lenguaje cuando se le presenten.

B.2. Añadir nuevas traducciones a la interfaz

Para poder añadir un nueva traduccion a la interfaz de la web (por ejemplo para dar mejor soporte a grupos de inglés) lo primero que necesitamos es agregar el fichero con las mismas, para facilitar la tarea podemos coger unos de los ficheros ya existente en la ruta `src/assets/i18n` En ella encontraras una serie de ficheros json con las abreviaturas de los idiomas ya existentes en la web (`es.json`, `en.json`, etc), si duplicamos unos de ellos ya tendremos la estructura de variables a traducir. Otra opción es usar un programa dedicado a esta tarea, existen varias posibilidades, aquí os dejamos tres. La primera es el traductor de babel (GmbH, 2019), es una herramienta de pago pero con un periodo de prueba, posee una gran interfaz y es de fácil uso, como ventaja te propone traducciones de un fichero ya existente al nuevo idioma que desees. Otra opción es usar la web Translation manager (Robin, 2019) es completamente gratuita, pero su interfaz es mas sencilla y no da ayudas a la traducción. También existe una extension de `visual studio code` que realizar esta función.

Una vez traducido el fichero se ha de guardar con la abreviatura del nombre de idioma que se describen mediante una especificación IETF llamada BCP 47 (lan, 2009), por ejemplo, `ingles` es `en` o `frances` es `fr`, etc.

El segundo paso es añadir al código la opción de seleccionar ese idioma, para ello tenemos que añadir la siguiente linea a los ficheros `app.component.html` y `secure.component.html` junto a las del resto de idiomas.

```
<button mat-menu-item
  [ngClass]="{'langActive': langSelected=='[mi idioma]'}"
  (click)="changeLang('[mi idioma]')"
  title="{{'lang_[mi idioma]'|translate}}">
  <mat-icon svgIcon="[mi idioma]" class="no-padding-img">
  </mat-icon>
  <span>{{'lang_[mi idioma]'|translate}}</span>
</button>
```

[1] El nombre del modo se extrae de la lista mencionada en el primer paso

Para finalizar tenemos que añadir el icono svg del idioma, para ello lo podemos descargar de la web **flaticon**, este icono lo guardaremos en el directorio `assets/images/svg` con la abreviatura de idioma que hemos usado anteriormente y añadimos la siguiente linea a los ficheros `app.component.ts` y `secure.component.ts` junto a las del resto de idiomas.

```
iconRegistry.addSvgIcon('[mi idioma]',
  sanitizer.bypassSecurityTrustResourceUrl(
    '../assets/images/svg/[mi idioma].svg'));
```

Es recomendable realizar comprobaciones sobre la aplicación para ver que se han traducido todas las variables correctamente y quedan bien visualmente.

Con esto ya tendremos nuestro nuevo idioma en la interfaz de la web, el propio sistema se encargara de mostrarlo cuando el usuario lo solicite.

Apéndice C

README

C.1. Scipio

Scipio es una aplicación basada en la creación de una aplicación para la corrección de practicas de programación por parte de los profesores y entre los propios alumnos (inter-pares), así como la generación de informes de dichas correcciones para el feedback.

Requisitos

- Linux ubuntu 18.04 o superior
- Angular CLI en su version 8.0.1 o superior
- pm2 process manager version 3.5.0 o superior
- MySQL Community Server Ver 14.14 Distrib 5.7.27

No se puede asegurar el funcionamiento en versiones inferiores a las anteriormente listadas.

C.2. Instalación

A continuación se describe el proceso de instalación de la aplicación.

Descarga del proyecto

Para poder descargar el proyecto es necesario tener instalado Git.

```
sudo apt install git
```

Para comprobar la instalación podemos ejecutar el siguiente comando.

```
git --version
```

A continuación procedemos a descargar el proyecto

```
git clone https://bitbucket.org/olayer66/scipio.git
```

Una ver descargado procederemos a la instalación de la aplicación.

C.2.0.1. Instalación de la aplicación

Para la instalación nos vamos al directorio de la aplicación y ejecutamos

```
npm install
npm install --only="dev"
```

Una vez termine podremos a instalar los elementos externos a la aplicación necesarios (pm2 y MySQL server)

MySQL server Para instalar el servidor de MySQL se recomienda seguir el tutorial siguiente.

```
https://www.digitalocean.com/community/tutorials/
    como-instalar-mysql-en-ubuntu-18-04-es
```

Una vez instalado podremos crear la base de datos y las tablas usaremos el script de gestión de la aplicación, que se explica en la sección de Gestión. PAra gestionar la base de datos si disponemos de interfaz gráfica podemos usar mysql workbench, phpmyadmin o cualquier gestor que permita bases de datos mysql. Si no disponemos de interfaz gráfica podemos usar el comando.

```
mysql -u root -p
```

Nos permitirá entran en el modo consola de mysql.

pm2 Para instalar el gestor de procesos lanzamos el siguiente comando en la consola

```
npm install pm2@latest -g
```

angular Para instalar angular en el sistema lanzamos el siguiente comando en la consola.

```
npm install -g @angular/cli
```

nodemon Para instalar el servidor backend de desarrollo lanzamos el siguiente comando.

```
npm install nodemon -g
```

C.3. Configuración

Base de datos

Para configurar las base de datos de nuestra aplicación debemos de abrir el fichero de configuración "knex.js" que esta situado en la carpeta "conf" de la raiz del proyecto. En este fichero encontraremos dos configuraciones iguales una para desarrollo (development) y otra para producción (production). Ambas contienen la misma estructura.

```
  {
        client: 'mysql',
        connection: {
              host : '127.0.0.1',
              user : 'root',
              password : 'Tfg2019!',
              database : 'scipioDB'
        },
        useNullAsDefault: true
   }
```

En esta estructura los campos que se han de modificar son.

- host: La dirección IP del servidor MySql.
- user: El nombre del usuario de la base de datos con el que vamos a acceder, este se ha configurado en el paso "Instalación de MySql server".
- password: La contraseña asociada al usuario anterior.

Se recomienda cambiar el usuario y la contraseña un uno propio, y que sean distintos para caso, en desarrollo se puede usar el usuario de root pero en el caso de producción se recomienda el uso de un usuario con permisos mas limitados (lectura y escritura) para evitar accesos no autorizados.

Comando para el gestor

Esta parte es opcional, pero mejora la gestión a no tener que estar llamando al script de gestión del servidor de forma manual.

El primer paso es localizar el fichero '.bashrcén el directorio raíz de nuestro usuario, para ello usamos el siguiente comando

```
ls -la
```

Nos mostrara los ficheros del directorio entre ellos el que necesitamos, Ahora lo abrimos para editar para ello lo mas sencillo desde la propia consola lanzamos

```
vi .bashrc
```

Nos abrirá el editor de vi con el fichero, si no se sabe usar se puede abrir con cualquier otro editor tanto de linea de comandos como gráfico.

Ahora tenemos que insertar la siguiente linea al final del fichero.

```
alias scipio='[path]/scipio/scripts/server.sh'
```

Siendo [path] la ruta absoluta al directorio donde este la aplicación. Guardamos y cerramos, para que surta efecto debemos cerrar y abrir la consola.

En este fichero se puede insertar también la variable de entorno ENV mediante la linea

```
export ENV="prod"
```

Esta linea nos permite que en caso de reinicio del servidor ya este configurado.

Configuramos el reboot automatico con un servicio

- Nos vamos a la carpeta que contiene los servicios

  ```
      cd /lib/systemd/system
  ```

  creamos un nuevo servicio llamado vps.service

  ```
    vi scipio.service
  ```

- copiamos y pegamos el siguiente código

```
[Unit]
Description=SCIPIO server start
After=network.target
After=systemd-user-sessions.service
After=network-online.target

[Service]
User=root
Type=forking
ExecStart=[path]/scipio/scripts/server.sh
ExecStop=[path]/scipio/scripts/server.sh
TimeoutSec=30
Restart=on-failure
RestartSec=30
StartLimitInterval=350
StartLimitBurst=10

[Install]
WantedBy=multi-user.target
```

Siendo [path] la ruta absoluta al directorio donde este la aplicación.

- guardamos y cerramos la edición
- lanzamos la siguiente secuencia de comandos (si el vps ya esta arrancado saldrá un fallo al realizar el start

```
sudo systemctl start scipio.service
sudo systemctl stop scipio.service
sudo systemctl enable scipio.service
```

- Comprobamos que el servicio se ha creado bien

```
systemctl list-units --type service --all
```

Saldrán todos los servicios buscamos el nuestro y veremos algo similar esto:

```
scipio.service                              loaded
      active   running SCIPIO server start
```

- Realizamos un reboot del servidor para comprobar el funcionamiento del servicio

BONUS: https://askubuntu.com/questions/919054/how-do-i-run-a-single-command-at-startup-using-systemd

C.4. Gestión de la aplicación

Lo primero que tenemos que hacer es definir el entorno de ejecución de la aplicación para ello disponemos de la variable de sistema ENV que puede tener dos modos. - Modo desarrollo

```
export ENV="dev"
```

- Modo producción

```
export ENV="prod"
```

En caso de no iniciarla o iniciarla con otros valores la aplicación entenderá que se ejecuta en modo de desarrollo. Con esto podemos ejecutar la aplicación en sus distintos modos.

Para la gestión de Scipio disponemos de un script que permite realizar diferentes acciones sobre la aplicación. Este esta situado en la carpeta scripts situada en la raiz del proyecto.

Para ejecutarlo primero le tenemos que dar permisos de ejecución

```
chmod +x scripts/server.sh
```

Ya podemos ejecutarlo, para para lanzarlo nos situamos en el directorio donde esta la carpeta scipio.

```
./[ruta]/scipio/server.sh
```

Una vez los ejecutemos nos mostrar las siguientes opciones.

1) Arrancar servidor

Permite arrancar el servidor con el modo indicado por la variable de entorno ENV, esto arranca tanto el servidor de Angular como el servidor del backend.

2) Reiniciar el servidor

Permite reiniciar el servidor con el modo indicado por la variable de entorno ENV.

3) Parar servidor

Para los servidores.

4)Iniciar Angular

Inicia el servidor de angular en modo desarrollo, esto deja la ventana bloqueada para pararlo usar `ctrl+c` para parar la ejecución.

5) Iniciar express

Inicia el servidor de Node js con express, esto deja la ventana bloqueada para pararlo usar `ctrl+c` para parar la ejecución.

6) Actualizar repositorio

Descarga la ultima version de la aplicación del repositorio y actualiza.

7) Monitor

Abre el monitor del servidor, muestra el estado del mismo y las entradas de log.

8) info SCIPIO_BACK

Muestra la información del servidor pm2 con respecto al Backend.

9) info SCIPIO_FRONT

Muestra la información del servidor pm2 con respecto al frontend de angular.

10) Gestor DB

Abre el gestor de la base de datos, que dispone de la siguientes opciones.

1) Crear DB

2) Reiniciar tablas

3) Salir

1) Crear BD

Permite crear la base de datos y las tablas necesarias para el uso de la aplicación, además de introducir los datos básicos de funcionamiento. Para crear la base de datos requiere de usuario y clave de MySQL server con permisos para ello.

2) Reiniciar tablas

Reinicia las tablas de la base de datos al estado inicial.

3) Salir

Cierra el gestor de la base dedatos.

11) Salir

Cierra el gestor de la aplicación.

C.5. Pruebas

Para poder probar la aplicación se ha creado una serie de usuarios para facilitar esta tarea, a continuación se muestra un listado con los mismos.

- test.admin
- test.profesor
- test.profesor2
- test.alumno
- test.alumno2
- test.alumno3
- test.alumno4
- test.alumno5
- test.alumno6
- test.alumno7

Todos ellos usan la contraseña `Tfg2019!` para acceder.

Bibliografía

Una computadora puede ser llamada inteligente si logra engañar a una persona haciéndole creer que es un humano

Alan Mathison Turing

Tags for identifying languages. https://tools.ietf.org/html/bcp47, 2009. [Online; accessed 04-September-2019].

Al Danial. Cloc. http://cloc.sourceforge.net/, 2015. Accessed: 2019-09-09.

Angular Contributors. Angular cli. https://cli.angular.io/, 2019a. [Online; accessed 07-September-2019].

Angular Contributors. Angular framework. https://angular.io/, 2019b. Accessed: 2019-03-05.

Angular Contributors. Angular material. https://material.angular.io/, 2019c. Accessed: 2019-08-07.

Angular Contributors. Angular routing. https://angular.io/guide/router, 2019d. Accessed: 2019-08-23.

Angular Contributors. Internacionalización. https://github.com/ngx-translate/core, 2019e. Accessed: 2019-23-08.

Bjarne Stroustrup, L. C++. https://es.wikipedia.org/wiki/C%2B%2B, 2019. Accessed: 2019-04-29.

Brendan Eich. Javascript. https://www.javascript.com/, 2019. Accessed: 2019-04-29.

Charlie Robbins. Winston. https://github.com/winstonjs/winston, 2019. Accessed: 2019-08-05.

CodeMirror Volunteers. Codemirror. https://codemirror.net/, 2019. Accessed: 2019-04-25.

Contributors, A. Architecture overview. https://angular.io/guide/architecture, 2019. [Online; accessed 07-September-2019].

jQuery Foundation, T. jquery. https://jquery.com/, 2019. Accessed: 2019-08-19.

GmbH, C. babel editor. https://www.codeandweb.com/babeledit, 2019. [Online; accessed 04-September-2019].

Group, P. G. D. Postgresql. https://www.postgresql.org/, 2019. Accessed: 2019-08-23.

Guillermo Rauch. Socket.io. https://socket.io/, 2019. Accessed: 2019-09-09.

Hampton Catlin, C. E., Natalie Weizenbaum y Anne, J. Sass. https://sass-lang.com/, 2019. Accessed: 2019-08-12.

Hansson, D. H. Rails. http://rubyonrails.org.es/, 2003. Accessed: 2019-08-23.

James Gosling y Microsystems, S. Java. https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n), 2019. Accessed: 2019-04-29.

Knex contributtors. Knex query builder. https://knexjs.org/, 2019. Accessed: 2019-01-18.

Lakshman, A. y Malik, P. Cassandra. http://cassandra.apache.org/, 2019. Accessed: 2019-08-23.

Lozano, I. A. Sails tutorial. https://abalozz.es/introduccion-a-sails-js-un-framework-para-crear-aplicaciones-realtime/, 2014. [Online; accessed 23-August-2019].

Martin Dougiamas. Moodle. https://moodle.org/, 2019. Accessed: 2019-04-29.

Microsoft. Typescript. https://www.typescriptlang.org/, 2019. Accessed: 2019-04-29.

Mike McNeil. Express. https://sailsjs.com/, 2019. Accessed: 2019-03-05.

MongoDB, Inc. Mongodb. https://www.mongodb.com/, 2019. Accessed: 2019-08-23.

MySQL AB, S. y Oracle Corporation. Mysql. https://www.mysql.com/, 2019. Accessed: 2019-04-29.

Nikhil. Estadisticas uso forntends. https://www.lambdatest.com/blog/top-javascript-frameworks-for-2019/, 2019. Accessed: 2019-09-09.

Rails Contributors. Ruby on rails. https://rubyonrails.org/, 2019. Accessed: 2019-03-05.

React contributors. React library. https://reactjs.org/, 2019. Accessed: 2019-03-05.

Robin, N. Translation manager. https://translation-manager-86c3d.firebaseapp.com/, 2019. [Online; accessed 04-September-2019].

Ryan Dahl. Nodejs. https://nodejs.org/es/, 2019. Accessed: 2019-04-29.

Scott Cooper. Toastr. https://github.com/scttcper/ngx-toastr, 2019. Accessed: 2019-08-30.

The GitHub Team. Github. https://github.com/, 2019. Accessed: 2019-09-09.

The Spring Team. Spring framework. https://spring.io/, 2019. Accessed: 2019-03-05.

TJ Holowaychuk. Express. https://expressjs.com, 2019. Accessed: 2019-03-05.

Wikipedia. Forma normal (base de datos) — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Forma_normal_(base_de_datos)&oldid=106197642, 2018. [Online; accessed 30-August-2019].

Wikipedia. Mapeo objeto-relacional — wikipedia, la enciclopedia libre. hhttps://es.wikipedia.org/w/index.php?title=Mapeo_objeto-relacional&oldid=117375002, 2019a. [Internet; descargado 9-septiembre-2019].

Wikipedia. Markdown — wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Markdown&oldid=116367584, 2019b. Accessed: 2019-09-09.

Wikipedia. Transferencia de estado representacional — wikipedia, la enciclopedia libre. 2019c. [Internet; descargado 9-septiembre-2019].

Wikipedia contributors. Software peer review — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Software_peer_review&oldid=892996558, 2019. [Online; accessed 23-August-2019].

Wind, D. K. y Jensen, U. A. Quantifying feedback: Insights into peer assessment data. http://0-search.proquest.com.cisne.sim.ucm.es/docview/1967318296?accountid=14514, 2017. Copyright - Copyright Academic Conferences International Limited Jun 2017; Última actualización - 2019-06-28.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes