

Universidad de Valladolid

Escuela de Ingeniería Informática de Valladolid



DevTest: Herramienta para la evaluación de conocimientos de programación

*Trabajo de Fin de Grado
Grado en Ingeniería Informática
Mención: Ingeniería de Software*

Autor: Javier Gatón Herguedas

Tutor: Valentín Cardeñoso Payo

27 de Junio de 2021

Dedicado a mi familia por apoyarme

Agradecimientos

A mi familia, por apoyarme y convivir conmigo cuando he estado estresado.

Al profesional de HP, Rubén López Fernández, que ha actuado como tutor externo del TFG, coordinando y colaborando en la gestión del proyecto, aconsejándome acerca de qué tecnologías podía utilizar y cómo hacerlo, y que por razones burocráticas de la Universidad de Valladolid no se le ha podido reconocer oficialmente como tutor externo del TFG.

A mi tutor Valentín, por su disponibilidad y rapidez en la colaboración, por transmitirme de forma simple y concisa la información acerca de los trámites y documentos necesarios del TFG, y sus consejos acerca de la usabilidad y otros aspectos que sin duda han permitido que la calidad de la aplicación sea mayor.

A mis amigos y compañeros por ayudarme a desconectar en mi tiempo libre y por ayudarme aportándome consejos y conocimientos técnicos.

Resumen

La evaluación automática de conocimientos prácticos de programación se realiza normalmente elaborando problemas de programación, cuyas soluciones entregadas se utilizan para generar salidas correspondientes a conjuntos de entrada determinados, que se comparan con los conjuntos de salida esperada para dichas entradas.

El objetivo de este Trabajo de Fin de Grado es el desarrollo de una aplicación web que facilite la elaboración colaborativa de exámenes y preguntas de programación, tanto teóricas como prácticas, y la corrección y evaluación automática de las respuestas.

Este proyecto se enmarca en el contexto del convenio de colaboración entre la Escuela de Ingeniería Informática de la Universidad de Valladolid y la empresa HP SCDS, por el que se creó el Observatorio HP, que tiene como objetivo ofrecer a los alumnos la oportunidad de realizar Trabajos de Fin de Grado dentro de un entorno empresarial donde cuentan con la asistencia de un tutor cualificado para guiarles durante el desarrollo del proyecto.

Se ha utilizado *Angular* para la capa de presentación en el cliente web y *Golang* para para las capas de negocio y persistencia en el servidor. El software gestor de la base de datos es *MariaDB*, y se ha utilizado *Docker* para la ejecución de las respuestas de los usuarios en la evaluación automática. La *API REST* utilizada para la comunicación entre el cliente y el servidor ha sido definida utilizando *Swagger 2.0*, además de *UML*. Para el desarrollo se utilizaron procedimientos ágiles, concretamente una adaptación de *SCRUM* y *Kanban*.

Abstract

Automatic evaluation of practical programming knowledge is usually conducted creating programming problems, whose solutions are executed in order to generate outputs for specific input sets, which are then compared to sets of expected outputs for those inputs.

The objective of this Undergraduate Thesis Project is the development of a web application that allows collaborative design of exams and questions, whether they are theoretical or practical, and also allows automatic correction and evaluation of the received answers.

This project is framed between the boundaries of the collaboration agreement between the Escuela de Ingeniería Informática of the University of Valladolid and the HP SCDS company, which resulted in the creation of the HP Observatory, that aims to offer students the chance to participate in Undergraduate Thesis Projects within a business environment, where they can be mentored by a qualified tutor during the project development.

Angular has been used for the presentation layer inside the web client, and *Golang* has been used for the business and persistence layers in the web server. *MariaDB* is the chosen database management software, and *Docker* has been used for the execution of the users answers in the process of automatic evaluation. The *REST API* used for communication between server and client has been defined using *Swagger 2.0* and *UML*. *SCRUM* and *Kanban* were applied as an agile framework.

Índice general

I	Objeto, Concepto y Método	5
1.	Introducción	7
1.1.	Introducción	7
1.2.	Contexto	8
1.3.	Motivación	9
2.	Objetivos y Alcance	11
2.1.	Objetivos	11
2.2.	Alcance	11
3.	Metodología y Requisitos	13
3.1.	Uso de Scrum en el proyecto	13
3.1.1.	¿Qué es Scrum?	13
3.1.2.	Adaptación de Scrum al proyecto	15
3.2.	Planificación	15
3.2.1.	Product Backlog	15
3.2.2.	Planificación y fechas de sprints	16
3.3.	Riesgos	17
II	Marco Conceptual y Contexto	19
4.	Marco Conceptual	21
4.1.	¿Qué es la programación?	21
4.1.1.	Código máquina y lenguajes ensambladores	21
4.1.2.	Lenguajes de alto nivel	21
4.1.3.	Paradigmas de programación	22
4.1.4.	Requisitos de calidad	22
4.1.5.	Tareas asociadas a la programación	23
4.2.	La programación y su aprendizaje	24
4.3.	La programación y su evaluación	24
4.3.1.	Evaluación basada en problemas de algoritmia	24
4.3.2.	Evaluación basada en proyectos	25
4.3.3.	Evaluación basada en depuración de errores	25
5.	Soluciones Existentes	27
5.1.	Descripción breve de las soluciones existentes	27
5.1.1.	HackerRank	27
5.1.2.	LeetCode	27
5.1.3.	CodeChef	27

5.1.4. CMS	27
5.2. Comparación	28
III Desarrollo del Sistema	29
6. Análisis	31
6.1. Descripción del sistema	31
6.1.1. Roles/Tipos de Usuarios	31
6.1.2. Equipos	32
6.1.3. Preguntas	32
6.1.4. Exámenes	32
6.1.5. Etiquetas	33
6.1.6. Notificaciones	33
6.1.7. Estados de las Respuestas	33
6.1.8. Corrección automática	33
6.2. Elicitación de Requisitos	34
6.2.1. Elaboración de preguntas colaborativas	34
6.2.2. Elaboración de exámenes de forma colaborativa	35
6.2.3. Envío de respuestas y análisis por parte de profesores con acceso	35
6.2.4. Corrección automática de preguntas teóricas	36
6.2.5. Editor de código integrado	37
6.2.6. Corrección automática de preguntas prácticas de programación	38
6.2.7. Soporte a, al menos, un lenguaje de programación	38
6.3. Casos de Uso	38
6.3.1. Usuario no registrado	38
6.3.2. Usuario (Usuario registrado base)	39
6.3.3. Profesor	39
6.3.4. Administrador	42
7. Diseño	45
7.1. Diseño de Base de Datos	45
7.1.1. Tablas	45
7.1.2. Tipos enumerados	57
7.1.3. Restricciones OCL	60
7.2. Diseño por Capas	63
7.2.1. Capas de Negocio y Persistencia: BackEnd	63
7.2.2. Capa de Presentación: FrontEnd	65
7.2.3. DTO	66
7.2.4. API REST	66
7.3. Seguridad de la Aplicación	68
7.3.1. Autenticación	68
7.3.2. CSP	74
7.4. Diseño de Interfaz de Aplicación	75
7.4.1. Landing pages	75
7.4.2. Barras de navegación	76
7.4.3. Tablas	76

8. Implementación	79
8.1. Herramientas y Tecnologías de Desarrollo	79
8.1.1. Trello	79
8.1.2. Visual Studio Code	79
8.1.3. UMLet	79
8.1.4. Postman	79
8.1.5. Typescript con Angular	80
8.1.6. Bootstrap	80
8.1.7. Bootswatch	80
8.1.8. Golang	80
8.1.9. MariaDB	80
8.1.10. Docker	80
8.1.11. NGINX	81
8.1.12. Git	81
8.1.13. GitLab	81
8.1.14. Ubuntu	81
8.1.15. Swagger 2.0	81
8.1.16. Go-Swagger	81
8.1.17. Swagger-Codegen	81
8.2. Generación automática de código	82
8.2.1. Generación automática en servidor	82
8.2.2. Generación automática en cliente	82
8.3. Seguimiento del Proyecto	82
8.3.1. Sprint 1	83
8.3.2. Sprint 2	85
8.3.3. Sprint 3	86
8.3.4. Sprint 4	86
8.3.5. Sprint 5	86
8.3.6. Sprint 6	86
8.3.7. Sprint 7	87
8.3.8. Sprint 8	87
8.3.9. Sprint 9	88
8.3.10. Sprint 10	88
8.3.11. Sprint 11	89
9. Pruebas	91
9.1. Pruebas unitarias	91
9.2. Pruebas End-To-End	91
9.3. Pruebas de Usabilidad	92
10. Conclusiones	93
10.1. Aportaciones	93
10.1.1. Diseño orientado a la realización de pruebas unitarias	93
10.1.2. Diseño orientado a la reducción de trabajo del equipo de desarrollo	94
10.1.3. Separación correcta de las funcionalidades de la capa de negocio y de los DAOs	94
10.1.4. Experiencia de gestión de proyecto ágil real	94
10.1.5. Tecnologías modernas aprendidas	94
10.2. Trabajo futuro	95
10.2.1. Ampliar los lenguajes soportados	95
10.2.2. Internacionalización	95
10.2.3. Roles personalizables en capa de presentación	95

10.2.4. Súper grupo “Organización” y actor “Director”	95
10.2.5. Mejorar calidad de código	95
10.2.6. Mejorar la interfaz y la experiencia de usuario	96
Apéndices	97
A. Administración e Instalación	99
A.1. Base de Datos	99
A.2. BackEnd	100
A.2.1. Ejecución de pruebas sobre las respuestas	100
A.2.2. Servidor	100
A.3. FrontEnd	101
B. Manual de Usuario	103
B.1. Acciones de usuario sin sesión iniciada	103
B.1.1. Iniciar sesión	103
B.1.2. Registrarse	103
B.1.3. Recuperar contraseña	103
B.2. Acciones de usuario base	104
B.2.1. Ver exámenes pendientes	104
B.2.2. Ver exámenes públicos	104
B.2.3. Responder examen	104
B.3. Acciones de profesor	104
B.3.1. Crear Pregunta	104
B.3.2. Crear Examen	105
B.3.3. Publicar Examen	105
B.4. Acciones de administrador	105
B.4.1. Modificar mensajes de <i>landing page</i>	105
C. Manual de Desarrollo	107
C.1. Modificar funcionalidad sin modificar la API	107
C.2. Modificación de API	107
C.2.1. BackEnd	107
C.2.2. FrontEnd	107

Parte I

Objeto, Concepto y Método

Capítulo 1

Introducción

1.1. Introducción

La programación es un componente muy importante de la informática, y es un requisito para muchas de las tareas y los trabajos relacionados con la misma. Es comprensible que las empresas quieran asegurarse de que los candidatos para un puesto de trabajo ofertado tienen los conocimientos de programación necesarios para el mismo, sin tener que evaluar individualmente a cada uno de ellos.

Sin embargo, no hay que caer en el error de exigir un rendimiento elevado en una prueba de programación avanzada, como mecanismo de filtrado de candidatos para un puesto de trabajo para ingenieros informáticos experimentados, ya que muy probablemente no se estaría seleccionando a los candidatos más aptos para el puesto, sino a los que mejor resuelven ciertas pruebas.

Por el contrario, evaluar los conocimientos de programación para puestos de trabajo que no requieran de todos los conocimientos de un ingeniero informático, donde los posibles candidatos solamente hayan estudiado una Formación Profesional o incluso directamente hayan salido de un *Bootcamp* o curso intensivo de programación, es algo extremadamente útil pues permite filtrar a los candidatos válidos de los no válidos para el puesto, ya que estos modelos educativos son conocidos por ser más laxos pero ellos pueden seguir saliendo buenos programadores e informáticos, a los que quizá a la empresa le interesaría seguir formando por su cuenta. Este modelo puede que sea poco común en España, pero en otros países no es tan extraño.

Sea útil o no lo sea, medir los conocimientos de programación no es una tarea muy sencilla. La metodología más común consiste en plantear problemas cuya resolución requiera de la utilización de diversas estructuras de datos y de diferentes algoritmos, enfrentando las soluciones propuestas por los candidatos a unas baterías de prueba que analicen diferentes casos, y que permitan otorgar un valor numérico que represente la calidad de la solución propuesta en función de cuantas pruebas supera cada solución y cuanto tiempo tarda en la ejecución.

Esta metodología es útil para algunos aspectos de la programación, pero no permite evaluar ciertos aspectos de la calidad de software del candidato como la realización de tests unitarios, el seguimiento de buenas prácticas, la correcta utilización de diferentes tecnologías para el desarrollo de software o la integración en un equipo de desarrolladores de software.

Para evaluar conocimientos estructurales más propios de la ingeniería de software, como por ejemplo el conocimiento de patrones de software comunes y cuando utilizarlos, no se suele actuar de esta forma, sino evaluando el conocimiento teórico de dichas cuestiones, o directamente planteando un pequeño proyecto y pidiendo una demostración funcional o simplemente la descripción detallada de

como lo estructuraría. El problema de evaluar una demo de un pequeño proyecto es que requiere bastante coste por el equipo que lo evalúa, lo cual reduce la cantidad de candidatos a los que se podrá evaluar. Por eso, muchas veces se opta por lo primero, preguntando cuestiones teóricas a los candidatos, aunque este método aporte menos información sobre los aspirantes que el otro.

Ya hay herramientas disponibles para la realización de pruebas de programación automáticas, con características diferentes cada una, como se explica en el capítulo 5. La herramienta que proponemos en este TFG, DevTest, se plantea como una alternativa de software libre que permite elaborar preguntas y exámenes de forma colaborativa, para evaluar tanto los conocimientos teóricos como las habilidades de programación, para asegurarse de que se comprenden los conceptos necesarios.

1.2. Contexto

La empresa *HP Solutions Creation & Development Services*, o *HP SCDS*, del grupo *HP Inc*, ofrece mediante el Observatorio Tecnológico HP[1] realizar una serie de Trabajos de Fin de Grado (y trabajos de fin de Máster) dentro de un entorno empresarial, donde los alumnos cuentan con la asistencia de un tutor cualificado para guiarles durante el desarrollo del proyecto.

En la Escuela de Ingeniería Informática de la Universidad de Valladolid, los TFG que se desarrollen al amparo del convenio con HP se realizarán bajo la dirección conjunta de profesionales de HP y profesores de la Escuela. En este caso, el profesional de HP encargado de la dirección del TFG ha sido Rubén López Fernández, y el profesor de la escuela encargado ha sido Valentín Cardeñoso Payo.

HP SCDS es una compañía tecnológica ubicada en León especializada en el desarrollo de firmware y software para impresoras 2D y 3D de HP. Suelen publicar una cantidad relativamente alta de ofertas de puestos de trabajo, entre las cuales, en la mayoría de los casos, se les requiere unos conocimientos mínimos en algún lenguaje concreto, y, en ocasiones, a los candidatos no se les requiere el grado informático.

Una aplicación como DevTest podría ayudar mucho a que los candidatos a las ofertas pasen un filtro inicial, descargando de trabajo al equipo de selección de personal, ya que se requiere del personal técnico interno para comprobar que los candidatos seleccionados tienen los conocimientos técnicos suficientes.

La idea base inicial de la aplicación consistía en que las preguntas de cada test se eligieran aleatoriamente entre una batería de preguntas clasificadas por nivel de conocimientos y, además, podría haber exámenes ya predefinidos o basados en plantillas (perfiles de usuario) [2].

También se indicaban las siguientes ideas principales[2]:

1. La parte de programación enviaría el contenido a un servidor para compilar y evaluar una batería de tests.
2. Podría haber más de una batería de tests (públicos y privados)
3. En un principio se soportarían 2 lenguajes de programación: C++ y JavaScript
4. El servicio de compilaciones tendría que ser remoto y soportar configuraciones por Docker.

El Tutor Externo de HP, Rubén, indicó que estas ideas iniciales no eran requisitos, sino simplemente un ejemplo del enfoque que se podría tomar en el TFG, y podríamos modificarlo con bastante libertad en función de lo que decidiéramos conjuntamente. Además, eso comprendía el potencial máximo del trabajo, pero para que desde HP lo dieran por cumplido, no haría falta hacerlo todo. Los objetivos finales se exponen en la sección 2.1.

1.3. Motivación

Es importante plantearse si la realización de pruebas automatizadas de programación para algunos puestos de trabajo es una ventaja respecto a un tratamiento manual de la información.

Compañías como Twitter, DropBox, Cisco o AirBnB utilizan servicios de HackerRank[3] para la selección de personal, y otras como Nintendo o Facebook utilizan CodinGame[4], una alternativa muy similar.

Amazon utiliza un software de carácter privado que les permite realizar un tratamiento de los datos más profundo y con el que, además, dan una imagen de marca mucho más personalizada y profesional. Las pruebas que realizan se denominan *Amazon's SDE Online Assessment* y, aunque no haya una referencia oficial de Amazon, podemos encontrar en Internet una gran cantidad de experiencias personales con estas pruebas[5]. Por otra parte, el alumno autor de este trabajo certifica haber participado en una de ellas en el pasado.

El hecho de que varias de las empresas de informática más importantes del mundo utilicen herramientas de software para la evaluación automática de conocimientos de programación como ayuda en los procesos de selección de personal, no implica que esta sea la manera más certera de realizar dicha selección. Sin embargo, el coste que supondría la evaluación individual de cada uno de los candidatos sería excesivamente elevado y no podrían evaluar a todos, lo que provoca que en puestos de trabajo donde la demanda sea bastante elevada, un primer filtrado automatizado de los candidatos permita seleccionar un subconjunto de los mismos para ser evaluado en más detalle, optimizando los recursos gastados en dicha selección.

Bien es cierto que este no es el caso de la mayoría de los puestos de trabajo en informática, donde la demanda del puesto no es tan exagerada como en grandes empresas tecnológicas. Sin embargo, para puestos de trabajo de entrada al mundo laboral, la cantidad de demanda puede ser relativamente elevada, y la capacidad de evaluación de los candidatos por parte de una empresa pequeña es bastante limitada, por lo que la realización de pruebas automatizadas puede ayudar para un primer filtrado de candidatos.

También es importante recalcar que no todas las evaluaciones automáticas de conocimientos de programación son iguales, pues para diferentes puestos laborales no solo las tecnologías y los tipos de pregunta pueden cambiar, sino que la forma de evaluar las respuestas puede ser diferente. Para ello, es importante tener una capacidad de acceso a la base de datos de respuestas de los candidatos, sobre las que poder realizar ciertos análisis posteriores para obtener diferentes métricas y valoraciones.

Capítulo 2

Objetivos y Alcance

2.1. Objetivos

Finalmente, los objetivos elegidos fueron los siguientes, siendo los tres primeros los básicos para que lo dieran por válido desde HP:

1. Elaboración de preguntas de diferentes tipos (teóricas y prácticas de programación) de forma colaborativa por parte de algunos usuarios con permisos para elaborar exámenes y preguntas, denominados “profesores”.
2. Elaboración de exámenes de forma colaborativa por parte de los profesores, utilizando las preguntas creadas.
3. Envío de respuestas a los exámenes por parte de los usuarios con acceso a su realización, y análisis de las respuestas por parte de los profesores con acceso a las mismas, y de todos los administradores.
4. Corrección automática de las respuestas a las preguntas teóricas.
5. Permitir entregar código mediante un editor de código integrado en la aplicación.
6. Corrección automática de las respuestas de preguntas prácticas de programación, utilizando baterías de pruebas creadas por los profesores para la pregunta.
7. Dar soporte a un lenguaje de programación, al menos.

Estos objetivos se explicarán y desarrollarán como requisitos en la sección 6.2.

2.2. Alcance

Se pretende crear una solución que combine el uso de preguntas prácticas de programación con el de preguntas teóricas, donde estas preguntas y los exámenes formados con las mismas se elaboren de forma colaborativa entre los miembros de la organización.

Se dará un fuerte enfoque al aspecto colaborativo de la realización de exámenes y preguntas, permitiendo realizar consultas complejas sobre la base de datos de preguntas y exámenes, compartir ambos con otros usuarios, clonarlos, marcar como favorito, etcétera.

La evaluación automática de los conocimientos de programación se efectuará de la forma tradicional, enfrentando las soluciones a los casos de prueba aportados por los profesores, y almacenando las respuestas en la base de datos, permitiendo que, si alguna organización así lo requiriera, se pudieran implementar operaciones de análisis avanzado de la calidad del software utilizando como base la aplicación aquí construida.

Quedan fuera del alcance del proyecto las siguientes cuestiones, planteadas inicialmente en las discusiones con la empresa:

- Las preguntas de los exámenes no son aleatorias, sino que cada examen tiene que ser diseñado individualmente y después compartido con los usuarios.
- Las tecnologías a usar no se describen en los objetivos, sino que se irán seleccionando en función de lo que se vaya considerando en el desarrollo.
- La clasificación por nivel de conocimientos no se utilizará de forma explícita, ya que eso puede limitar las posibilidades de la organización al establecer una clasificación de los niveles de conocimientos de antemano, cuando estos pueden ser muy diferentes según el tipo de pregunta, lenguaje, etc. Esto se implementará mediante el uso de etiquetas, como se verá más adelante, en la sección 6.1.5.

Capítulo 3

Metodología y Requisitos

Nos encontramos frente a un proyecto que presenta una gran incertidumbre inicial, pues es el primer proyecto de un calibre importante que desarrolla el alumno de forma autónoma, con escasa o nula experiencia en múltiples de las tecnologías a utilizar.

También es muy difícil definir el alcance y los requisitos de forma precisa al inicio del desarrollo, teniendo solamente descritos con exactitud los requisitos de las etapas más tempranas. Por ello, y por su gran adaptación a los cambios en requisitos, se ha decidido utilizar metodologías ágiles en el desarrollo del proyecto.

Esto también conlleva la ventaja intrínseca de utilizar metodologías incrementales, que permiten aportar pequeñas mejoras graduales al proyecto, lo cual en caso de un error en la planificación de tiempo, posibilita que haya parte del proyecto finalizado y funcional listo para entregar.

3.1. Uso de Scrum en el proyecto

Se decide aplicar un marco de trabajo de tipo Scrum, o mejor dicho una adaptación del mismo, pues un Scrum “puro” es incompatible con ciertos aspectos del desarrollo de un Trabajo de Fin de Grado individual, como la limitación del equipo de desarrollo a un único integrante.

Además se va a permitir realizar pequeñas modificaciones de requisitos a lo largo del sprint, bajo juicio del equipo de desarrollo (el alumno).

Utilizar Scrum permite adaptarse dinámicamente en el desarrollo a los requerimientos del cliente para que vea qué se está haciendo, generar una buena predicción de tiempo de trabajo y mitigar y reducir riesgos de manera anticipada desarrollando primero las funcionalidades de mayor valor.

3.1.1. ¿Qué es Scrum?

Scrum[6] es un marco de trabajo ágil que fue diseñado inicialmente para reducir el *time to market* o tiempo para mercado, que es el tiempo que tarda un producto informático en llegar a estar disponible para su venta. Debido a los beneficios que aporta, explicados en el apartado anterior, es una de las metodologías ágiles más extendidas hoy en día.

De forma resumida, Scrum consiste en que un equipo de desarrollo pequeño trabaje de forma coordinada, y divida las tareas en iteraciones denominadas *sprints* que duran entre 1 y 4 semanas. A continuación, se definirán diferentes conceptos necesarios para comprender Scrum correctamente.

Artefactos

- **Product Backlog:** Lista ordenada del trabajo y requisitos del proyecto, que el equipo de desarrollo deberá llevar a cabo.
- **Sprint Backlog:** Lista de trabajo que el equipo de desarrollo deberá realizar en el siguiente sprint.
- **Incremento:** Los ítems del backlog implementados durante el sprint.

Elementos del Backlog

- **Historia de usuario (story):** Descripciones cortas de una característica que debe tener el programa, contada desde la perspectiva del usuario o cliente que la usará. Como un caso de uso, pero muy simple.
- **Historia épica (epic):** Similar a una historia de usuario, pero que requiera mucha carga de trabajo y puede que no de tiempo a hacerse en un sprint. Se dividirá en múltiples historias.
- **Puntos de historia de usuario:** Es una medición del esfuerzo de una historia, como la carga de trabajo, pero sin unidades de horas, es un valor relativo, en formato de la serie de Fibonacci.
- **Tarea:** Actividad que tiene asignada menos de 16 horas, que es una subdivisión de un requisito del sprint.

Roles

- **Product Owner:** Único responsable del *Product Backlog*, lo ordena, organiza y expresa claramente sus ítems.
- **Scrum Master:** Da soporte al equipo de desarrollo en todo el sprint, incluido en la parte del diseño del *Sprint Backlog*.
- **Equipo de Desarrollo:** Diseña el *Sprint Backlog* al inicio del sprint y lo transforma en incrementos de funcionalidad durante el sprint. El tamaño puede variar entre 3 y 9 integrantes.

Elementos del flujo de trabajo

- **Sprint:** Una iteración de entre 1 y 4 semanas para la que se establece un plan y unos objetivos a tener hechos cuando se acabe la misma.
- **Sprint planning:** Reunión al inicio del sprint que define los objetivos del mismo y diseña el *Sprint Backlog*.
- **Sprint review:** Reunión al finalizar el sprint, que revisa el trabajo completado y el trabajo planificado no completado. Después se realiza el *Sprint retrospective*.
- **Sprint retrospective:** Reunión para revisar la productividad del Sprint anterior, que intenta mejorar el proceso de desarrollo y aumentar la productividad.

- **Scrum daily:** Reunión diaria de 15 minutos del equipo de desarrollo para poner en común lo desarrollado y crear una cultura de comunicación.

Otros Artefactos

- **Definition of Done (DoD):** Criterios que deben cumplirse para que un ítem del backlog se marque como completado.
- **Spike:** Período corto de tiempo usado para investigar acerca de un concepto. Puede ser entre sprints o durante uno. No tiene por qué resultar un entregable.

3.1.2. Adaptación de Scrum al proyecto

Asignación de roles

- **Scrum Master:** El *Scrum master* será el alumno, Javier Gatón Herguedas.
- **Product Owner:** El *Product Owner* será el tutor externo, Rubén López Fernández.
- **Equipo de Desarrollo:** El equipo de desarrollo estará compuesto de un único integrante: Javier Gatón Herguedas, el alumno.

Adaptación del flujo de trabajo

Debido a que el equipo de desarrollo está formado por una única persona, no serán necesarias las reuniones del tipo Scrum Daily.

La duración de los Sprints es de dos semanas, realizándose entre cada Sprint el Sprint Review, Sprint Retrospective y Sprint Planning del siguiente mediante una reunión por videollamada.

La definición de completado (DoD) se amplía para que incluya la seguridad de las operaciones. Es decir, para que una tarea como transmitir datos por internet se marque como completada tiene que estar realizada una implementación que permita realizar ese flujo de datos de forma segura. Esto se realiza para evitar dejar la seguridad para sprints tardíos, pues se recomienda encarecidamente tener en cuenta la seguridad desde el inicio.

Al inicio, principalmente durante el primer sprint, se realizarán varios *Spikes* que consistirán en que el alumno decida qué tecnologías y qué metodologías y técnicas de seguridad usará y también para que se familiarice con las mismas.

3.2. Planificación

3.2.1. Product Backlog

A continuación se mostrará el conjunto de historias de usuario épicas descritas inicialmente, que componen la funcionalidad que se intentará abarcar. Las historias épicas 5 y 6 no serán obligatorias y formarán parte de los incrementos posteriores a acabar la funcionalidad básica, que es hasta la historia épica número 4.

Inicialmente solo tenemos definiciones con cierta precisión de las primeras historias épicas, en este caso de la primera, como se puede ver en el cuadro 3.2.

Estas historias serán divididas en tareas y serán susceptibles a cambios. En el cuadro 3.3 está descrita una descomposición inicial de las tareas de la primera historia. En la sección 8.3 se puede observar el conjunto de historias épicas e historias de usuario que hay descritas a lo largo del desarrollo.

Número	Historia	Descripción
1	Creación de preguntas	El usuario profesor podrá crear y modificar preguntas
2	Creación de exámenes	El usuario profesor podrá crear y modificar exámenes
3	Resolución de exámenes	El usuario evaluado podrá resolver exámenes.
4	Corrección automática	El profesor podrá resolver manual o automáticamente las respuestas a los exámenes.
5	Compilación y prueba en servidor	El usuario profesor podrá subir casos de prueba en las preguntas que requieran programar. El usuario evaluado podrá subir código y se compilará y probará contra los casos de prueba en el servidor.
6	Compilación en servidor y prueba de cliente	El usuario podrá compilar el código en el servidor y comprobar con casos que él suba, o con los que se den públicos en esa pregunta.

Cuadro 3.1: Historias épicas del Product Backlog Inicial

1.1	Registro como usuario	El usuario podrá registrarse como usuario
1.2	Login como profesor	El usuario profesor podrá iniciar sesión como profesor
1.3	Creación de una pregunta simple	El profesor podrá crear una pregunta simple que reciba respuestas de texto
1.4	Modificación de una pregunta	Modificación del contenido de la pregunta y los permisos de modificación
1.5	Clonado de una pregunta	El profesor podrá clonar una pregunta existente, creando una nueva igual de su propiedad.

Cuadro 3.2: Historias de usuario de la epic 1

1.1.1	API registro profesor	Creación de la API de conexión con el Backend, de manera segura y documentada
1.1.2	BD Registro profesor	Registro de profesor en la base de datos desde el BackEnd
1.1.3	Interfaz de registro	Vista del cliente que permita registrarse como profesor conectando con la API

Cuadro 3.3: Descomposición en tareas de la historia 1.1

3.2.2. Planificación y fechas de sprints

El Trabajo de Fin de Grado tiene asignados 12 créditos ECTS. Según la Universidad de Valladolid, cada crédito ECTS corresponde a 25 horas de trabajo lo que implica que el proyecto debe constar de aproximadamente 300 horas de trabajo por parte del alumno.

Teniendo 10 sprints de dos semanas, comenzando el miércoles 20 de enero de 2021, la carga de trabajo correspondería a 30 horas por sprint, 15 horas por semana. La disposición de fechas de cada sprint planeada inicialmente se muestra en el cuadro 3.4.

Finalmente, debido a diferentes causas como por ejemplo vacaciones, exámenes o entregas de prácticas, las fechas reales de los sprints no se corresponden a la perfección con lo previsto. Las fechas reales de los sprints y las acciones asociadas a dichos sprints se muestran en el cuadro 3.5. Se puede observar que se añadió un sprint final la última semana, que permitió implementar completamente la ejecución de pruebas unitarias sobre las respuestas de los usuarios.

Sprint	Fecha
Sprint 1	20/01/2021 - 03/02/2021
Sprint 2	03/02/2021 - 17/02/2021
Sprint 3	17/02/2021 - 03/03/2021
Sprint 4	03/03/2021 - 17/03/2021
Sprint 5	17/03/2021 - 31/03/2021
Sprint 6	31/03/2021 - 14/04/2021
Sprint 7	14/04/2021 - 28/04/2021
Sprint 8	28/04/2021 - 12/05/2021
Sprint 9	12/05/2021 - 26/05/2021
Sprint 10	26/05/2021 - 09/06/2021

Cuadro 3.4: Sprints y sus fechas previstas asociadas

Sprint	Fechas	Resumen
Sprint 1	20/01/2021 - 03/02/2021	Investigación de Swagger, Autorización y Autenticación
Sprint 2	03/02/2021 - 17/02/2021	Login/Signin en BackEnd
Sprint 3	17/02/2021 - 03/03/2021	Login/Logout FrontEnd y BackEnd
Sprint 4	03/03/2021 - 17/03/2021	Autorización, Equipos y Administración
Sprint 5	17/03/2021 - 06/03/2021	BackEnd y API REST Exámenes y Preguntas
Sprint 6	06/04/2021 - 19/04/2021	FrontEnd Creación de Exámenes y Preguntas
Sprint 7	19/04/2021 - 03/05/2021	Publicar Tests, compartir, alumnos responden
Sprint 8	03/05/2021 - 18/05/2021	Página de configuración, notificaciones por correo, corrección de exámenes.
Sprint 9	18/05/2021 - 02/06/2021	Ajustes de Interfaz y Experiencia de Usuario
Sprint 10	02/06/2021 - 21/06/2021	Arreglos Interfaz, Incremento 2 y empezar Incremento 3
Sprint 11	21/06/2021 - 28/06/2021	Terminar Incremento 3

Cuadro 3.5: Sprints y sus fechas reales asociadas

3.3. Riesgos

Durante el inicio de la realización del Trabajo de Fin de Grado se analizaron los posibles riesgos que conllevarían el desarrollo del mismo. Sólo se han detectado dos riesgos, explicados en los cuadros 3.6 y 3.7.

Nombre	Estimación incorrecta de tiempo de realización necesario
Probabilidad	Media
Impacto	Medio
Descripción	Debido a la escasa definición inicial de los requisitos, es posible que el desarrollo se alargue en el tiempo y no se realicen los objetivos iniciales.
Mitigación	La división del trabajo en incrementos de funcionalidad y el uso de Scrum permiten que se realicen primero los elementos de mayor valor, lo que en caso de no dar tiempo dejaría fuera solo parte del trabajo.

Cuadro 3.6: Riesgo 1

Nombre	Contracción de enfermedad grave
Probabilidad	Baja
Impacto	Alto
Descripción	Es posible que el alumno enferme con Covid19 u otra enfermedad y pierda entre unos días a varias semanas de trabajo.
Mitigación	Tomar todas las precauciones posibles para evitar el contagio.

Cuadro 3.7: Riesgo 2

Parte II

Marco Conceptual y Contexto

Capítulo 4

Marco Conceptual

4.1. ¿Qué es la programación?

La programación en informática consiste en la elaboración de programas, entendiendo los mismos como el conjunto unitario de instrucciones que permite a una computadora realizar funciones diversas, como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos, el manejo de bancos de datos, etcétera.[7]

4.1.1. Código máquina y lenguajes ensambladores

Las computadores necesitan código máquina para poder realizar sus funciones, que se trata de código compuesto de un conjunto de instrucciones particular para la máquina, las cuales permiten controlar directamente la unidad central de procesamiento (CPU) de la computadora. Cada instrucción permite realizar una tarea concreta, como la lectura o escritura en memoria, saltos o bifurcaciones del flujo de ejecución del código, o una operación de la unidad aritmética lógica (ALU) en uno o más datos almacenados en los registros de la CPU[8]. Inicialmente, salvo en los primeros programas, se utilizaban lenguajes ensambladores, que consisten en lenguajes legibles para humanos en los que, en la mayoría de casos, cada operación se traduce en una instrucción en código máquina. Al ser prácticamente código máquina legible por humanos, si dos máquinas tienen conjuntos de instrucciones diferentes, hay que utilizar un lenguaje ensamblador diferente para cada una de esas máquinas[9].

4.1.2. Lenguajes de alto nivel

En la mayoría de los casos, hoy en día se utilizan lenguajes de programación de alto nivel, que hacen de la programación algo más simple, comprensible y menos vinculado a la máquina en la que se ejecuta, permitiendo que un mismo programa pueda ser ejecutado en diferentes máquinas que tengan distintos conjuntos de instrucciones entre sí. Estos lenguajes permiten una abstracción más elevada, y tienen una riqueza sintáctica mucho mayor.

Dentro de estos lenguajes se podrían describir dos tipos, los compilados y los interpretados. Los primeros necesitan de un proceso denominado “compilación”, que consiste en transformar código de alto nivel en código máquina concreto para la máquina que lo va a ejecutar, utilizando su conjunto de instrucciones[10]. Los interpretados, traducen en tiempo de ejecución cada instrucción de alto nivel a una o varias instrucciones de código máquina, mediante el intérprete o la máquina virtual encargada de su interpretación[11].

4.1.3. Paradigmas de programación

Cada lenguaje de alto nivel está asociado a uno o más *paradigmas de programación*, entendiendo por tales los modelos de referencia establecidos en términos del modelo de cómputo de referencia y de las características sintácticas y semánticas del lenguaje de alto nivel empleado. A continuación se hará un breve repaso de los cuatro paradigmas principales[12].

Paradigma Imperativo

El paradigma imperativo se basa en que el programador instruye a la máquina acerca de qué operaciones tiene que ejecutar, cómo hacerlo y cuándo ejecutarlas. Es el paradigma más similar al funcionamiento del código máquina, lo cual explica por qué es el paradigma más antiguo, a la vez que el más utilizado.

Paradigma Funcional

El paradigma funcional es, en muchos aspectos, un paradigma más simple y claro que el imperativo. En este paradigma, el resultado deseado se declara como una serie de aplicación de funciones. El origen del paradigma funcional está en el cálculo lambda, el cual puede servir para definir qué es computable, y podría considerarse como el lenguaje de programación más pequeño del mundo[13].

Paradigma Lógico

El paradigma lógico se basa en encontrar automáticamente el resultado a una respuesta a una pregunta sobre un conjunto de reglas, relaciones y hechos del dominio del problema. Está fuertemente basado en la lógica formal, y su escritura comprende el conjunto de sentencias expresando hechos y reglas.

Orientación a Objetos

El paradigma de orientación a objetos se basa en agrupar instrucciones y datos según la sección del estado en las que operan. Utiliza el concepto de “objeto”, estructura que puede contener datos y código. Los datos en forma de campos, conocidos como atributos o propiedades, y el código en la forma de procedimientos, conocidos como métodos o funciones. Este paradigma necesita utilizarse en conjunción con otro paradigma.

4.1.4. Requisitos de calidad

Independientemente del acercamiento, metodología o paradigma tomado, el programa final debe satisfacer algunas propiedades fundamentales de calidad. Las siguientes son algunas de las principales[14][15]:

- **Fiabilidad:** Qué porcentaje de veces que se ejecuta un programa, sus resultados son correctos. Esto depende del correcto diseño de los algoritmos utilizados, la minimización de los errores de programación, la minimización de errores de gestión de recursos, y la minimización de errores lógicos.

- **Robustez:** Cómo se anticipa un programa a problemas y errores producidos por datos incorrectos, la no disponibilidad de recursos del sistema como memoria, servicios del sistema operativo o la red, errores de usuario y fallos de electricidad inesperados.
- **Usabilidad:** Facilidad con la que una persona usuaria puede utilizar y ejecutar el programa para su propósito esperado e incluso para propósitos no esperados. Esto envuelve a muchos elementos textuales, gráficos e incluso de hardware.
- **Eficiencia/Rendimiento:** Medidas de uso de recursos del sistema, como tiempo de procesador, espacio en memoria, dispositivos como discos, y la red, que consume el programa. Cuanto menos, mejor. Esto también incluye la gestión de recursos, como por ejemplo eliminar ficheros temporales y reserva de memoria no utilizada.
- **Portabilidad:** El rango de dispositivos hardware y sistemas operativos en los que el código fuente de un programa puede ser compilado o interpretado y ejecutado. Esto depende de los servicios proporcionados por diferentes plataformas, como recursos de hardware y sistemas operativos, comportamiento esperado de los mismos, y disponibilidad de compiladores y librerías para el lenguaje del código fuente.
- **Mantenibilidad:** Facilidad con la que un programa puede ser modificado por los desarrolladores presentes y futuros para hacer mejoras, personalizar funcionamiento, arreglar errores o fallos de seguridad y adaptar a nuevos entornos. El seguimiento de buenas prácticas[16] durante el desarrollo temprano es muy importante para crear código mantenible.
- **Legibilidad:** Facilidad con la que un desarrollador puede comprender el propósito, los flujos de control y las operaciones del código fuente. Afecta a la mayoría de aspectos previamente mencionados, en especial a la portabilidad, la usabilidad y principalmente la mantenibilidad.

4.1.5. Tareas asociadas a la programación

Otras tareas importantes que están asociadas a la programación y permiten cumplir los requisitos de calidad, son la realización de pruebas de software y la depuración de errores. Ambas tareas son muy importantes y el desarrollo de software tiene que estar enfocado a la facilitación de ambas, para así optimizar la forma de alcanzar una mejor calidad en el software.

Realización de pruebas de software

La realización de pruebas de software sirve para obtener información sobre la calidad del producto o servicio de software que está siendo probado[17], intentando encontrar fallos y verificando que el producto software está preparado para su utilización. Puede servir para otorgar una visión objetiva e independiente del software que permite entender los riesgos que incurren en la implementación de ese software.[18]

Las pruebas de software comprenden la ejecución de un componente o sistema software para evaluar una o más propiedades de interés. Estas propiedades son:

- Alcanzan los requisitos que guiaron su diseño y desarrollo
- Responden correctamente a todo tipo de entrada
- Cumple su función en un período temporal aceptable
- Es suficientemente usable

- Puede ser instalado y ejecutado en los entornos esperados
- Alcanza el resultado general esperado por los “stakeholders”

Depuración de errores

La depuración de errores en el desarrollo de software consiste en el proceso de búsqueda y resolución de “bugs”, defectos o problemas que impiden el funcionamiento correcto, en programas o sistemas de software. Las tácticas de depuración de errores están relacionadas con depuración interactiva, análisis del control de flujo, pruebas unitarias, pruebas de integración, análisis de ficheros de “log” o de registro, monitorizar la aplicación, volcados de memoria, etcétera. Muchos lenguajes de programación y herramientas de software ofrecen programas que ayudan en la depuración, conocidos como depuradores o “debuggers”.

4.2. La programación y su aprendizaje

El aprendizaje a programar y, por contrapartida, su enseñanza, son actividades bastante complejas que dependen parcialmente de las capacidades de cada individuo. No hay una única metodología para su enseñanza, pero para los primeros pasos de su aprendizaje si que hay cierto consenso generalizado.

La metodología más habitual consiste en la enseñanza del paradigma imperativo, comenzando por los tipos de datos y operaciones más básicos, y el uso de variables. Posteriormente se enseñan las estructuras de control de flujo, empezando por las más simples como la dupla “if-else” y terminando con estructuras más complejas, que a veces dependen del lenguaje utilizado. Finalmente, se enseña acerca de la programación procedimental, explicando las funciones, métodos, recursividad, etcétera.

Una vez se han comprendido los conceptos básicos previamente expuestos, se proceden a enseñar otros paradigmas, la orientación a objetos, estructuras de datos más avanzadas y algoritmia. También se enseñan cuestiones de ingeniería de software, análisis y diseño de sistemas software, el diseño de diferentes tipos de tests, tecnologías comunes para el desarrollo de software y nociones sobre la calidad del software.

4.3. La programación y su evaluación

La evaluación de los conocimientos de programación de una persona es una tarea complicada, pues hay una gran cantidad de variables a observar como el manejo del lenguaje y las herramientas utilizadas, como la calidad del software producido, tal y como se explica en la sección 4.1.4.

Por esto, según qué se desee evaluar exactamente, existen diferentes tipos de evaluaciones.

4.3.1. Evaluación basada en problemas de algoritmia

La evaluación basada en problemas de algoritmia permite observar el manejo y soltura que se tiene al utilizar un lenguaje de programación concreto, al mismo tiempo que los conocimientos acerca de algoritmia y la capacidad de implementar los algoritmos y funcionalidades necesarias en lenguajes

concretos.

4.3.2. Evaluación basada en proyectos

La evaluación basada en proyectos o demos de proyectos consiste en pedir un pequeño proyecto al evaluado, y que éste lo realice en un breve período de tiempo, o simplemente indique la forma en la que lo estructuraría.

Este tipo de evaluación permite observar la capacidad de abstracción, análisis, diseño y, en ocasiones, implementación, que se tiene respecto a ciertos tipos de problemas y sistemas. Comprende la utilización adecuada de los patrones de software necesarios, la capacidad de estructurar correctamente un sistema para optimizar las diferentes características que favorecen su calidad, y la capacidad de orientar el sistema a la realización de pruebas de software y a la futura depuración de posibles errores en el mismo.

4.3.3. Evaluación basada en depuración de errores

La evaluación basada en depuración de errores permite analizar de forma más detallada los conocimientos que tiene una persona sobre un lenguaje de programación específico, además de permitir ver sus habilidades de depuración de errores.

Consiste en otorgar una serie de fragmentos de software, o sistemas completos, que contengan uno o más errores y pedir al evaluado que lo arregle. Estos errores pueden ser de muchos tipos, desde errores de compilación o errores sintácticos, hasta errores de rendimiento de un sistema, pasando por errores en fiabilidad, donde una función o sección realiza cálculos o acciones que pretendían, según un enunciado, obtener resultados diferentes.

Capítulo 5

Soluciones Existentes

Observando las diferentes alternativas existentes para la creación de pruebas de programación en línea, se estudiarán las cuatro principales y más utilizadas: *HackerRank*, *LeetCode*, *CodeChef* y *CMS*.

5.1. Descripción breve de las soluciones existentes

5.1.1. HackerRank

HackerRank[19] es una plataforma de programación competitiva, que permite participar en concursos y exámenes, organizar tus propios concursos y tus propios problemas. Posee un servicio de pago para empresas, con el que el equipo de *HackerRank* ayuda a dichas empresas en la labor de selección de personal.

5.1.2. LeetCode

LeetCode[20] es una plataforma de concursos de programación competitiva. Se caracteriza porque algunos de sus problemas son muy comunes en entrevistas de trabajo, y se utiliza para entrenamientos para las mismas. Posee un servicio para empresas mediante el que se puede patrocinar algunos concursos e incluso diseñar los mismos.

5.1.3. CodeChef

CodeChef[21] es otra plataforma de programación competitiva, en la que están publicados diversos concursos, y se celebran concursos concretos periódicamente. Ofrece la posibilidad de alojar concursos para centros educativos de forma gratuita, y lo mismo para empresas pero pagando[22].

5.1.4. CMS

CMS (Contest Management System)[23] es un proyecto de código abierto diseñado para la Olimpiada Internacional de Informática de 2012. También puede utilizarse para otro tipo de concursos, aunque quizá necesitando ciertas modificaciones.

5.2. Comparación

Como se puede ver en el cuadro 5.1, DevTest destaca de la mayoría de alternativas al ser gratuito y de software libre, aunque no se ofrece un servicio de alojamiento externo. Esto último puede ser un requisito importante para algunos clientes, aunque otros podrían preferir una herramienta que les permita alojar el concurso de forma independiente usando recursos propios de almacenamiento.

Herramienta	Preguntas Teóricas	Alojamiento Externo	Alojamiento Interno	Gratuito	Software Libre
HackerRank	No	Sí	No	Sí	No
LeetCode	Sí	Sí	No	No	No
CodeChef	No	Sí	No	Depende	No
CMS	No	No	Sí	Sí	Sí
DevTest	Sí	No	Sí	Sí	Sí

Cuadro 5.1: Comparación de las diferentes soluciones con DevTest

Aún así, se puede ver que DevTest comparte muchas características con CMS, lo cual lo convertiría en su competidor o alternativa principal. Lo que más diferenciaría a DevTest de CMS y del resto de alternativas, además de las tecnologías utilizadas, es que DevTest permite realizar preguntas teóricas, permite organizar y separar a los usuarios en equipos, permite explorar los exámenes y preguntas creadas según etiquetas y otros parámetros y, finalmente, que DevTest permite ser personalizado en gran medida y en diferentes niveles.

Respecto a la exploración de exámenes y preguntas según etiquetas y otros parámetros, HackerRank tiene funcionalidades similares. De hecho, el proyecto intenta asemejarse bastante a HackerRank en algunos aspectos, pero mejorando la experiencia e interfaz de usuario y permitiendo que se aloje de forma interna, siendo software libre.

Respecto a la capacidad de personalización y modificación, sin llegar a tener que editar el código, se pueden modificar los mensajes de la pantalla de inicio de cada tipo de usuario, ya que estos pueden ser configurados por los administradores utilizando markdown, lo que permite escribir texto con diferentes opciones e incluso insertar imágenes.

También es importante mencionar que el estilo gráfico se basa en el uso de Bootstrap, tal y como se indica en la sección 8.1, que es uno de los principales frameworks de CSS. Si se creara una hoja de estilos basada en Bootstrap y se enlazara en el FrontEnd, se podría cambiar de forma global. El estilo de la aplicación se basa en una hoja de estilos predefinida de Bootswatch, concretamente el estilo Lumen, excepto para el color del fondo de pantalla de la web, que está introducido manualmente.

Las dos formas de modificación explicadas previamente, permiten modificar la imagen de la aplicación web, lo cual es útil en la situación de querer personalizar la imagen de marca expuesta para los usuarios. Sin embargo, también es relativamente sencillo modificar la funcionalidad, pues la documentación de DevTest es muy completa teniendo diversos diagramas de UML, que explican el funcionamiento de las diferentes capas y partes de la aplicación, como la API, el BackEnd y el FrontEnd.

En conclusión, DevTest es una opción suficientemente competitiva y original respecto al resto de soluciones existentes y puede ser la más adecuada en función de los requisitos que tenga el cliente.

Parte III

Desarrollo del Sistema

Capítulo 6

Análisis

6.1. Descripción del sistema

El proyecto consiste en crear una plataforma de evaluación de conocimientos teóricos y prácticos relacionados con el software, permitiendo la elaboración de preguntas y exámenes de forma colaborativa. Todo esto se hará manteniendo una cierta similitud con una red social, donde los diferentes usuarios podrán ver los contenidos a los que tengan permiso de acceso, y podrán compartir, copiar, valorar o intentar realizar los diferentes exámenes y preguntas que haya en el sistema.

6.1.1. Roles/Tipos de Usuarios

Las funcionalidades disponibles y casos de uso necesarios para el usuario variarán en función del tipo del mismo. Podríamos distinguir cuatro roles, incluyendo a los usuarios visitantes no registrados.

Usuario no registrado

Se trata del usuario que entra en la página web sin estar registrado en el sistema o sin haber iniciado sesión. Debe tener disponibles las operaciones necesarias para iniciar sesión o registrarse, y además debe poder ver los exámenes que hay publicados en el sistema, en función de si así lo deciden los administradores.

Usuario base/Estudiante

El usuario que va a ser evaluado, ya sea un estudiante o un candidato, podrá resolver exámenes y deberá poder ver directamente qué exámenes tiene pendientes de realizar. También debe poder explorar los exámenes públicos, sus respuestas pasadas, los equipos a los que pertenece, etcétera.

Profesor

Los profesores son un tipo de usuario privilegiado que tiene las mismas capacidades que un usuario normal y además tiene otras, las cuales son sus acciones principales, que son la creación y modificación de preguntas y exámenes.

Administrador

Los administradores tienen disponible las mismas funcionalidades que los profesores, pero pueden modificar los datos de cualquier pregunta o examen independientemente de que el autor les haya dado o no acceso a su edición, podrán modificar los roles de los usuarios, y modificar datos y ajustes generales del sistema.

6.1.2. Equipos

Un equipo es una entidad que representa un conjunto de usuarios a los que se puede otorgar determinados permisos o a los que se puede asociar un perfil común con mecanismos de gestión conjuntos. Hemos optado por separar los equipos en dos tipos, en función de las características de sus miembros: los equipos que administran preguntas o exámenes solo pueden contener profesores o administradores. Si contienen usuarios básicos, servirán únicamente para invitarles a la participación en un examen. Los usuarios base no pueden crear ni administrar equipos.

6.1.3. Preguntas

Las preguntas son solicitudes de información que se harán en un examen, y servirán para comprobar los conocimientos que tiene el evaluado acerca del asunto concreto de la pregunta. Se pueden distinguir tres tipos: Preguntas de respuesta de texto (ya sea la respuesta un número, una frase o un texto largo), preguntas de tipo test o respuesta múltiple, que a su vez pueden ser de elección única o de elección múltiple, y preguntas prácticas de programación, cuya respuesta será un fragmento de código, el cual se compilará y ejecutará en el servidor, probándolo con pruebas unitarias creadas por los profesores.

Las preguntas pueden ser modificadas por los administradores del sistema, el autor de la pregunta, o los profesores miembros de los equipos a los que se ha compartido la pregunta.

6.1.4. Exámenes

Los exámenes son conjuntos de preguntas que se podrán enviar y asignar a diferentes usuarios para su resolución y comprobar sus conocimientos acerca de la temática de las preguntas del examen. Desde un examen se podrá acceder a las respuestas al mismo. Pueden tener límites de duración, donde si un usuario no entrega antes de cierto tiempo, ya no podrá entregar.

La manera de modificar un examen es igual que en las preguntas, donde existe un autor, un conjunto de equipos que podrán modificarla, y donde todos los administradores del sistema también tienen derecho de edición.

Un examen puede estar en uno de dos estados: borrador y publicado. Cuando un profesor quiera enviar un examen a un usuario para que lo responda, el profesor no podrá enviar el borrador de examen, sino que deberá publicar previamente el examen. La versión publicada de examen no puede ser modificada, y esa versión podrá ser enviada a uno o múltiples usuarios. Esto se plantea así para evitar que un profesor pueda modificar un examen después de enviárselo a un alumno, ya sea por error o con alguna intención maliciosa.

6.1.5. Etiquetas

Tanto las preguntas como los exámenes podrán tener etiquetas asociadas, que servirán para clasificarlos según diferentes criterios como la temática o la dificultad, para poder explorar de forma más cómoda las preguntas y los exámenes que están relacionados con un conjunto de temas, y para poder hacer una búsqueda más detallada.

6.1.6. Notificaciones

La plataforma tendrá un sistema de notificaciones por correo electrónico que enviará información importante a los usuarios, como la invitación a la realización de un examen, o el enlace para la recuperación de contraseña. Utilizará la dirección de correo electrónico dada por los administradores, permitiendo el uso de un servidor de correo privado o el uso de uno gratuito.

6.1.7. Estados de las Respuestas

Una vez el usuario responda al examen, éste no podrá ver sus respuestas ni su puntuación hasta que lo indique el profesor o cualquier otro administrador del examen. Los administradores del examen podrán configurarlo para que las respuestas se vuelvan visibles manualmente, al marcar como corregido el examen o directamente nada más entregarlo, y para que la respuesta se marque como corregida en el momento de ser entregada, o que lo tenga que marcar el profesor automáticamente. También podrán decidir si una pregunta es o no es de corrección automática. Un examen puede tener preguntas de corrección automática y de corrección manual al mismo tiempo.

6.1.8. Corrección automática

La corrección automática variará en función del tipo de pregunta. En las preguntas teóricas el profesor podrá añadir un porcentaje de penalización, donde si envían una respuesta incorrecta, tendrán puntuación negativa en esa pregunta.

En las preguntas con respuesta de texto, la corrección automática consiste en que el texto enviado por el usuario coincida con la respuesta esperada que ha introducido el profesor. Esto está pensado para preguntas cortas o problemas matemáticos, donde se pueda corregir fácilmente.

En las preguntas de respuesta múltiple, el profesor añadirá opciones e indicará cuales son correctas y cuales son incorrectas. Si es una pregunta de elección única, se marcará como correcta en caso de la opción seleccionada sea correcta. Si, por el contrario, la pregunta es de elección múltiple, el usuario deberá seleccionar todas las opciones correctas para que se marque la respuesta como correcta.

En las preguntas prácticas de programación, el profesor configurará diferentes casos de prueba, y se ejecutará el programa contra esos casos de prueba. La puntuación obtenida en esa respuesta se calculará en función de la cantidad de casos que supere, y el valor asociado a cada caso/prueba.

6.2. Elicitación de Requisitos

Los requisitos nacen de los objetivos explicados en la sección 2.1. A continuación se expondrán los requisitos funcionales y los no funcionales más importantes que permiten cumplir cada objetivo. Esto no incluye toda la funcionalidad implementada o funcionalidad a implementar, ya que ciertos elementos se tienen que hacer simplemente para mejorar la calidad del producto software final, o porque son características pequeñas que surgen en un sprint avanzado del desarrollo.

Tal y como se indica en los objetivos, los tres primeros son necesarios para que el producto final se considere terminado por parte de HP. Los otros cuatro no son obligatorios, y se dispondrán en tres incrementos de funcionalidad, que para que estén completos tienen que implementarse los requisitos correspondientes. El primer incremento consiste en los requisitos de la sección 6.2.4, el segundo en los requisitos de la sección 6.2.5, y el tercero en los requisitos de las secciones 6.2.6 y 6.2.7.

6.2.1. Elaboración de preguntas colaborativas

Requisitos Funcionales

- **RF101:** El sistema debe permitir a los usuarios iniciar sesión.
- **RF102:** El sistema debe permitir a los profesores crear una nueva pregunta.
- **RF103:** El sistema debe permitir a los profesores modificar el tipo de pregunta de una pregunta, entre pregunta de respuesta de texto, de respuesta múltiple de elección única y respuesta múltiple de elección múltiple.
- **RF104:** El sistema debe permitir a los profesores modificar los datos de preguntas existentes.
- **RF105:** El sistema debe permitir a los profesores compartir los derechos de administración de las preguntas con otros equipos de profesores.
- **RF106:** El sistema debe permitir a los profesores clonar las preguntas.
- **RF107:** El sistema debe permitir a los profesores marcar y desmarcar las preguntas como favoritas.
- **RF108:** El sistema debe permitir a los profesores buscar preguntas, según diversos parámetros y filtros.
- **RF109:** El sistema debe permitir a los profesores crear equipos.
- **RF110:** El sistema debe permitir a los profesores añadir a usuarios a equipos.
- **RF111:** El sistema debe permitir a los usuarios abandonar equipos.

Requisitos No Funcionales

- **RNF101:** Los usuarios tendrán que usar una contraseña y un nombre de usuario o correo para iniciar sesión.
- **RNF102:** Las preguntas sólo las pueden modificar los profesores que estén en algún equipo que administre la pregunta concreta, el profesor autor, o todos los administradores. Con modificar también se entiende añadir equipos para compartir derechos de administración.

- **RNF103:** Las preguntas las podrán ver todos los profesores salvo que algún usuario con derechos de edición la marque como privada, en ese caso solo la verán los el conjunto de usuarios descrito en el requisito no funcional anterior.
- **RNF104:** Los usuarios base no pueden ser añadidos a equipos de profesores.
- **RNF105:** Los usuarios base no pueden administrar un equipo.

6.2.2. Elaboración de exámenes de forma colaborativa

Requisitos Funcionales

- **RF201:** El sistema debe permitir a los profesores crear un nuevo examen borrador.
- **RF202:** El sistema debe permitir a los profesores modificar los datos del examen borrador.
- **RF203:** El sistema debe permitir a los profesores añadir y quitar preguntas existentes al examen borrador.
- **RF204:** El sistema debe permitir a los profesores compartir los derechos de administración de los exámenes con otros equipos de profesores.
- **RF205:** El sistema debe permitir a los profesores clonar exámenes.
- **RF206:** El sistema debe permitir a los profesores marcar y desmarcar exámenes borradores como favoritos.
- **RF207:** El sistema debe permitir a los profesores buscar exámenes borradores, según diversos parámetros y filtros.

Requisitos No Funcionales

- **RNF201:** Los exámenes sólo los pueden modificar los profesores que estén en algún equipo que administre el examen concreto, el profesor autor, o todos los administradores. Con modificar también se entiende añadir equipos para compartir derechos de administración.
- **RNF202:** Los exámenes borradores los podrán ver todos los profesores salvo que algún usuario con derechos de edición la marque como privada, en ese caso solo la verán los el conjunto de usuarios descrito en el requisito no funcional anterior, RNF201.

6.2.3. Envío de respuestas y análisis por parte de profesores con acceso

Requisitos Funcionales

- **RF301:** El sistema debe permitir a los profesores publicar exámenes a partir de los exámenes borradores. Se creará una copia no modificable de los datos del examen y de sus preguntas.
- **RF302:** El sistema debe permitir a los usuarios buscar y explorar exámenes publicados en función de diversos parámetros y filtros.
- **RF303:** El sistema debe permitir a los usuarios iniciar la respuesta a un examen publicado.
- **RF304:** El sistema debe permitir a los usuarios enviar una respuesta a una pregunta.
- **RF305:** El sistema debe permitir a los usuarios borrar su respuesta a una pregunta.

- **RF306:** El sistema debe permitir a los usuarios marcar una respuesta iniciada a un examen como finalizada.
- **RF307:** El sistema debe permitir a los profesores ver las respuestas enviadas a los exámenes.
- **RF308:** El sistema debe permitir a los profesores corregir las respuestas a las preguntas de los exámenes, indicando la puntuación obtenida.
- **RF309:** El sistema debe permitir a los profesores marcar y desmarcar como corregida una respuesta a un examen.
- **RF310:** El sistema debe permitir a los profesores marcar y desmarcar como visible para el autor una respuesta a un examen.
- **RF311:** El sistema debe permitir a los profesores automatizar cuando se marcará como visible y/o corregida una respuesta a un examen.
- **RF312:** El sistema debe permitir a los usuarios ver sus respuestas entregadas.

Requisitos No Funcionales

- **RNF301:** Las preguntas sólo se pueden modificar si son la pregunta original y no la copia publicada que se genera al publicar un examen.
- **RNF302:** Los datos de los exámenes y sus preguntas asociadas sólo se pueden modificar si son el examen borrador original y no la copia publicada que se genera al publicar un examen.
- **RNF303:** Cualquier usuario puede ver y realizar un examen publicado salvo que algún usuario con derechos de administración del examen indique que se necesita invitación para realizar dicho examen.
- **RNF304:** Los usuarios sólo podrán enviar y borrar respuestas a una pregunta si tienen una respuesta al examen iniciada y no finalizada.
- **RNF305:** Los profesores solo podrán marcar (o desmarcar) como corregida y/o visible las respuestas a exámenes que administren, siguiendo los criterios que se explican en el RNF201.
- **RNF306:** Los usuarios solo podrán ver los detalles de sus respuestas entregadas si éstas están marcadas como visibles para el autor.

6.2.4. Corrección automática de preguntas teóricas

Requisitos Funcionales

- **RF401:** El sistema debe permitir a los profesores indicar una respuesta esperada para las preguntas teóricas.
- **RF402:** El sistema debe permitir a los profesores indicar un porcentaje de penalización en caso de error.
- **RF403:** El sistema debe permitir a los profesores indicar si una pregunta es de corrección automática.

Requisitos No Funcionales

- **RNF401:** Para las preguntas de respuestas de texto la respuesta esperada deberá ser una cadena de texto.
- **RNF402:** Para las preguntas de tipo test o de respuesta múltiple, sean de elección única o elección múltiple, la respuesta esperada será indicar qué opciones son las correctas y cuáles son las incorrectas.
- **RNF403:** En las preguntas con corrección automática de respuestas de texto la respuesta se marcará como correcta si corresponde con la respuesta esperada.
- **RNF404:** En las preguntas con corrección automática de tipo test o respuesta múltiple que sean de elección única, las respuestas se marcarán como correctas si la opción elegida está indicada por el profesor como correcta.
- **RNF405:** En las preguntas con corrección automática de tipo test o respuesta múltiple que sean de elección múltiple, las respuestas se marcarán como correctas si las opciones elegidas son todas las opciones que el profesor a marcado como correctas.
- **RNF406:** Para poder indicar si la pregunta es de corrección automática, indicar el porcentaje de penalización o indicar la respuesta esperada de una pregunta, el profesor debe pertenecer al conjunto de usuarios especificado en el requisito RNF102.
- **RNF407:** Para poder indicar si la pregunta es de corrección automática, indicar el porcentaje de penalización o indicar la respuesta esperada de una pregunta, la pregunta debe cumplir lo especificado en el requisito RNF301.

6.2.5. Editor de código integrado

En caso de implementar estos requisitos, los requisitos RNF502 y RNF503 no son estrictamente necesarios para considerar el incremento como implementado, pero deberían ser ignorados exclusivamente si el equipo de desarrollo encuentra muy difícil su implementación.

Requisitos Funcionales

- **RF501:** El sistema debe permitir a los usuarios introducir directamente sus fragmentos o programas de código en las preguntas prácticas de programación.

Requisitos No Funcionales

- **RNF501:** El editor de código deberá mostrar correctamente los espacios y las tabulaciones.
- **RNF502:** El editor de código deberá generar cerrar automáticamente los paréntesis, llaves y corchetes.
- **RNF503:** El editor de código deberá soportar el lenguaje de programación tal y como se especifica en los requisitos de la sección 6.2.7, indicando con colores cierta información acerca de la gramática y sintaxis del lenguaje.

6.2.6. Corrección automática de preguntas prácticas de programación

Requisitos Funcionales

- **RF601:** El sistema debe permitir a los profesores añadir casos de prueba que la respuesta debería cumplir.
- **RF602:** El sistema debe permitir a los profesores modificar los casos de prueba de las preguntas.
- **RF603:** El sistema debe permitir a los profesores modificar el valor de los casos de prueba respecto a la puntuación final de la respuesta a la pregunta.
- **RF604:** El sistema debe permitir a los profesores modificar la visibilidad del caso de prueba.

Requisitos No Funcionales

- **RNF601:** Para poder añadir casos de prueba a una pregunta o modificarlos, el profesor debe pertenecer al conjunto de usuarios especificado en el requisito RNF102.
- **RNF602:** Para poder añadir casos de prueba a una pregunta o modificarlos, la pregunta debe cumplir lo especificado en el requisito RNF301.

6.2.7. Soporte a, al menos, un lenguaje de programación

Requisitos Funcionales

- **RF701:** El sistema debe permitir utilizar lenguajes de programación, y compilar y ejecutar los fragmentos de código contra las pruebas oportunas.
- **RF702:** El sistema debe indicar al usuario qué lenguaje de programación está siendo utilizado.

Requisitos No Funcionales

- **RNF701:** Se deberá soportar al menos un lenguaje de programación.
- **RNF702:** El lenguaje de programación a soportar será C++14.

6.3. Casos de Uso

Hay cuatro actores en el sistema, que corresponden con los roles/tipos de usuarios explicados en la sección 6.1.1. A continuación se expondrán los casos de uso principales de cada uno.

6.3.1. Usuario no registrado

El usuario no registrado solamente puede realizar un caso de uso, y así está representado en la figura 6.1. Evidentemente puede realizar más acciones, como iniciar sesión o registrarse, pero esas operaciones no tienen valor de negocio[24], por lo que no se incluyen como casos de uso.

- **Buscar Examen:** Búsqueda y filtrado de los exámenes disponibles para el actor, buscando por el título y filtrando por las etiquetas. También incluye la ordenación de los exámenes según parámetros como la fecha de publicación.

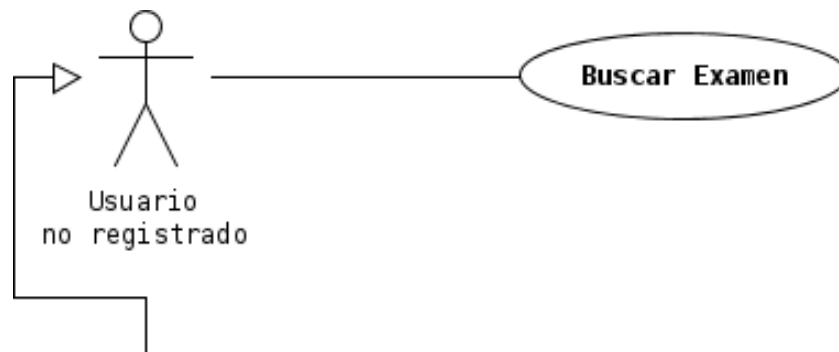


Figura 6.1: Casos de uso del Actor: Usuario no registrado

6.3.2. Usuario (Usuario registrado base)

Los casos de uso del actor Usuario se exponen en la figura 6.2.

- **Ver datos personales:** El actor podrá ver los datos personales relacionados con él mismo que están almacenados en el sistema.
- **Modificar datos personales:** El actor podrá modificar sus datos personales almacenados en el sistema.
- **Ver sus equipos:** El actor podrá ver los equipos a los que pertenece.
- **Salir de un equipo:** El actor podrá abandonar un equipo al que pertenezca.
- **Realizar Examen:** El actor podrá iniciar la realización de un examen y responder a sus diferentes tipos de preguntas.
- **Responder Pregunta:** El actor envía la respuesta a una pregunta concreta, dependiendo del tipo de pregunta. Solo se puede acceder a este caso de uso a través de la realización del caso de uso "Realizar Examen".
- **Ver respuestas enviadas:** El actor podrá ver respuestas enviadas por él mismo anteriormente, y podrá ver sus detalles si así se lo permite el profesor responsable.

6.3.3. Profesor

El actor Profesor puede realizar una gran cantidad de casos de uso, por lo que se ha tenido que dividir en dos figuras, las figuras 6.3 y 6.4.

- **Buscar usuario:** El actor podrá buscar a otro usuario a través del nombre de usuario.
- **Crear Equipo:** El actor podrá crear un equipo nuevo, decidiendo si admitirá cualquier tipo de usuarios o solo profesores (y administradores).
- **Modificar Equipo:** El actor podrá modificar un equipo que administre.
- **Añadir Usuario a Equipo:** El actor podrá añadir a un usuario a un equipo que el actor administre, siempre que el otro usuario cumpla los requisitos necesarios.

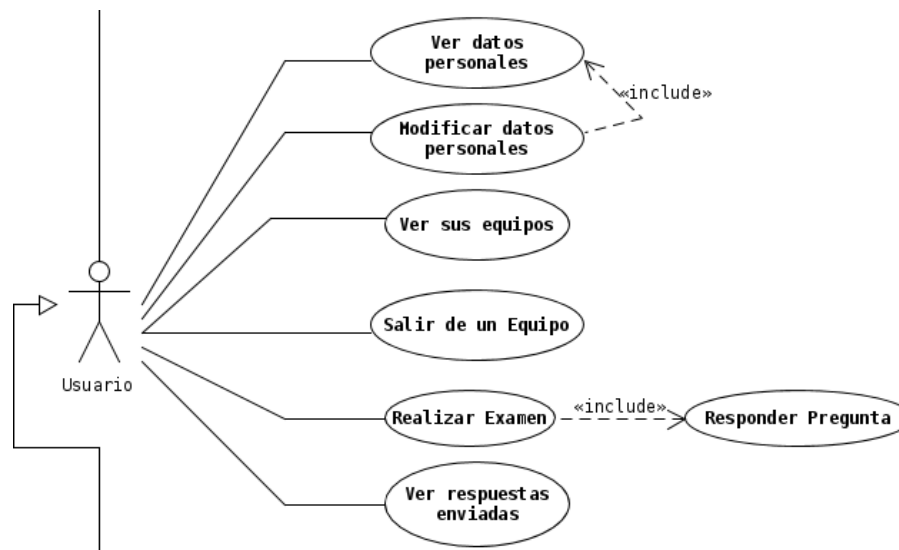


Figura 6.2: Casos de uso del Actor: Usuario

- **Modificar Rol de un profesor en Equipo:** El actor podrá modificar el rol de otro usuario en un equipo que el actor administre, cumpliendo las restricciones.
- **Eliminar Usuario de Equipo:** El actor podrá eliminar la membresía de otro usuario de un equipo que el actor administre.
- **Crear Pregunta:** El actor podrá crear una pregunta nueva.
- **Crear Examen:** El actor podrá crear un examen borrador nuevo.
- **Buscar Pregunta:** El actor podrá buscar una pregunta existente utilizando filtros de búsqueda oportunos.
- **Asignar Pregunta a Examen:** El actor podrá añadir una pregunta a la que tenga acceso a un examen borrador que administre. Incluye la búsqueda de pregunta.
- **Buscar Examen Borrador:** El actor podrá buscar un examen borrador existente utilizando los filtros oportunos.
- **Modificar Examen Borrador:** El actor podrá modificar un examen borrador que administre.
- **Modificar Pregunta:** El actor podrá modificar una pregunta editable que administre.
- **Publicar Examen:** El actor podrá publicar un examen borrador que administre, creando una copia no editable de las preguntas y los datos del examen borrador.
- **Clonar Examen:** El actor podrá crear una copia de un examen y sus preguntas como un examen borrador de su propiedad.
- **Clonar Pregunta:** El actor podrá crear una copia de una pregunta como otra pregunta igual pero de su propiedad.
- **Marcar Pregunta como Favorita:** El actor podrá marcar y desmarcar una pregunta como favorita.

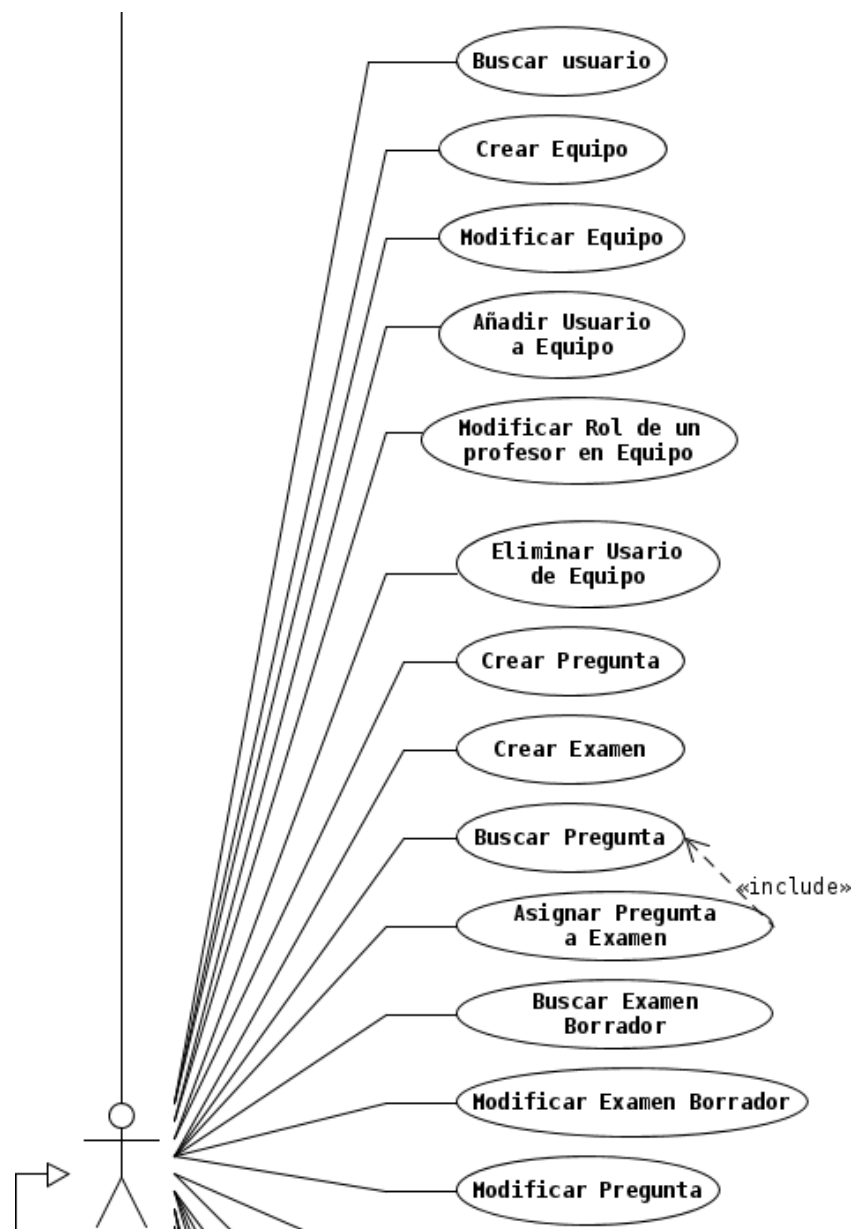


Figura 6.3: Casos de uso del Actor: Profesor. Parte 1

- **Marcar Examen como Favorito:** El actor podrá marcar y desmarcar un examen borrador como favorito.
- **Asignar Examen a Equipo de Profesores:** El actor podrá añadir a la administración de un examen que administre a otro equipo de profesores.
- **Enviar invitación de Examen Publicado a Equipo:** El actor podrá invitar a la realización de un examen publicado a un equipo.
- **Enviar invitación de Examen Publicado a Usuario:** El actor podrá invitar a la realización de un examen publicado a un usuario.
- **Ver Respuesta de un Usuario:** El actor podrá ver la respuesta de un usuario a un examen que administre.

- **Ver lista de Respuestas a Examen Publicado:** El actor podrá ver la lista de respuestas a un examen publicado que administra
- **Corregir Respuesta a Pregunta:** El actor podrá corregir y puntuar una respuesta a una pregunta de un examen publicado que administra.
- **Modificar estados de Corregido y/o Visible de una Respuesta a Examen:** El actor podrá marcar o desmarcar como corregida una respuesta a un examen que administra, y también marcarlo o desmarcarlo como visible para el usuario autor de dicha respuesta.

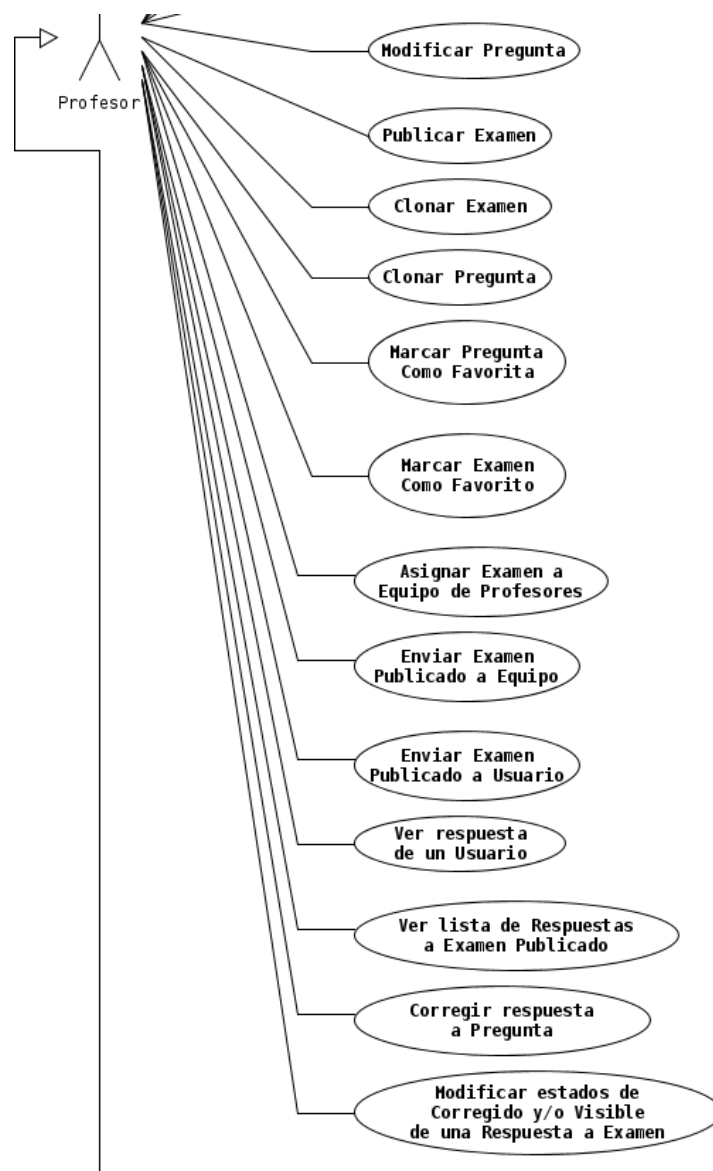


Figura 6.4: Casos de uso del Actor: Profesor. Parte 2

6.3.4. Administrador

Finalmente, nos encontramos con los casos de uso del actor Administrador, que como se puede ver en la figura 6.5, son bastantes menos que los del actor Profesor.

- **Ver panel de gestión de usuarios:** El actor puede ver el panel de gestión de usuarios con las opciones permitidas correspondientes.
- **Crear usuario con correo electrónico:** El actor puede crear un nuevo usuario mediante correo electrónico, lo que le hará recibir un correo electrónico al nuevo usuario.
- **Cambiar el rol de otro usuario:** El actor podrá modificar el rol que tiene otro usuario.
- **Modificar datos de correo electrónico del sistema:** El actor podrá modificar los datos del correo electrónico que el sistema usará para enviar notificaciones.
- **Modificar los permisos de cada rol:** El actor podrá modificar algunos permisos y capacidades que tiene cada diferente rol existente del sistema.
- **Personalizar mensajes de bienvenida:** El actor podrá modificar los mensajes de bienvenida que ven los usuarios en la página de inicio según su rol.

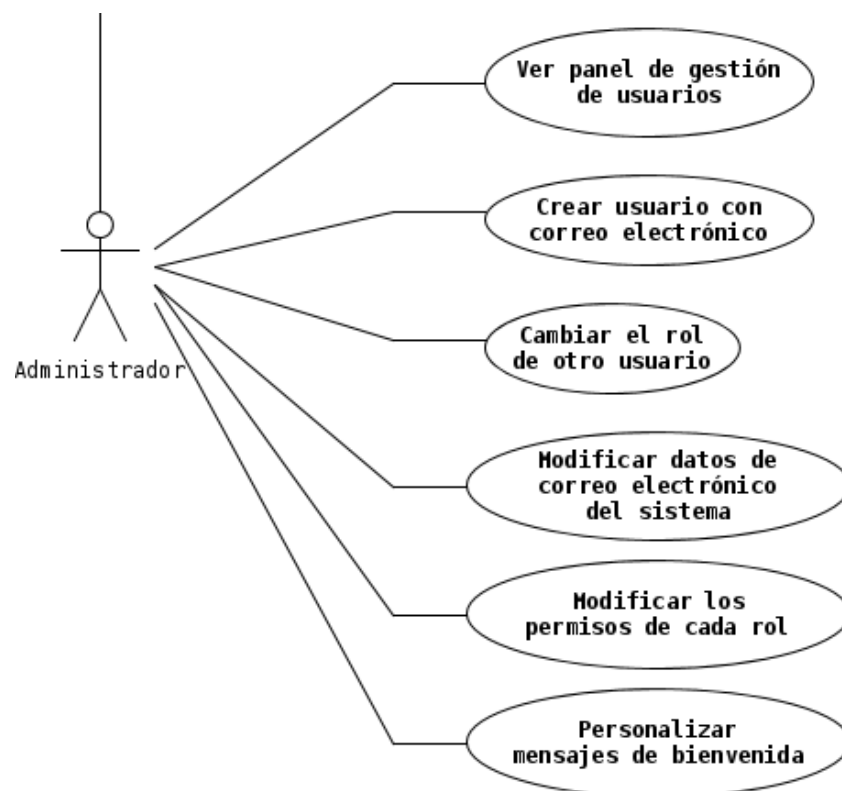


Figura 6.5: Casos de uso del Actor: Administrador

VistaPersonalizada

La tabla VistaPersonalizada almacena información que se mostrará en la capa de presentación del usuario, como un mensaje personalizado. Se muestra en la figura 7.2 y tiene los siguientes atributos:

- **rolBase**: Tipo de usuario al que se le aplicarán las características de la vista. Los tipos de RolUsuarioE se explican en el apartado correspondiente de la sección 7.1.2.
- **mensajeInicio**: Mensaje que se mostrará en las pantalla de inicio de la aplicación.

VistaPersonalizada
- rolBase: RolUsuarioE {id}
- mensajeInicio: string

Figura 7.2: Detalle de Tabla VistaPersonalizada

TipoRol

La tabla TipoRol almacena los tipos de usuarios existentes. Los tipos básicos de usuarios están descritos en RolUsuarioE, pero esta tabla se diseñó con la idea de que los administradores del sistema pudieran crear diferentes tipos de usuario según sus necesidades, y darles diferentes permisos de acceso y edición a cada uno. Sin embargo, estas funcionalidades no se llegaron a implementar en la capa de aplicación, existiendo solamente tantas instancias de TipoRol como valores existen en RolUsuarioE. Se muestra en la figura 7.3 y sus atributos son los siguientes:

- **id**: Clave primaria para identificar cada instancia.
- **rolBase**: Rol de usuario básico que tendrá el tipo de usuario definido. Servirá para saber cuales son los casos de uso importantes para el mismo. Los tipos de RolUsuarioE se explican en el apartado correspondiente de la sección 7.1.2.
- **nombre**: Nombre, que no se puede repetir, que tendrá la instancia. Servirá para que los administradores y usuarios lo identifiquen.
- **prioridad**: Importancia que tiene un TipoRol respecto a los demás. El valor más prioritario es el 0. Indica si el usuario con dicho TipoRol asociado puede cambiar el TipoRol asociado a otro usuario. Para poder cambiarlo, el TipoRol del usuario que cambia tiene que tener `changeRoles==true` y tener una prioridad con valor menor o igual (ser más prioritario) que los TipoRoles origen y destino del otro usuario.
- **verPTests**: Capacidad de ver exámenes publicados.
- **verETests**: Capacidad de ver exámenes borradores.
- **verEQuestions**: Capacidad de ver preguntas borradores.
- **verPQuestions**: Capacidad de ver preguntas de exámenes publicados.
- **verAnswers**: Capacidad de ver respuestas de otros usuarios a exámenes.
- **changeRoles**: Capacidad de cambiar los roles de otros usuarios, en caso de que las prioridades lo permitan.

- **tenerTeams**: Capacidad de crear y tener equipos de usuarios.
- **tenerEQuestions**: Capacidad de crear y tener preguntas.
- **tenerETests**: Capacidad de crear y tener exámenes borradores.
- **tenerPTests**: Capacidad de tener exámenes publicados.
- **adminPTests**: Capacidad de modificar cualquier examen publicado.
- **adminETests**: Capacidad de modificar cualquier examen borrador.
- **adminEQuestions**: Capacidad de modificar cualquier pregunta borrador.
- **adminAnswers**: Capacidad de modificar cualquier respuesta.
- **adminUsers**: Capacidad de modificar los datos de cualquier usuario.
- **adminTeams**: Capacidad de modificar cualquier equipo.
- **adminConfiguration**: Capacidad de modificar configuración de la aplicación como la información de la dirección de correo electrónico utilizada.
- **adminPermissions**: Capacidad de modificar permisos de los TipoRoles y crear o eliminar instancias de los mismos.
- **tipolnicial**: Indica si a un nuevo usuario se le asociará este TipoRol. En caso de haber múltiples, por defecto se debería escoger el TipoRol menos importante, es decir el que tenga un valor de prioridad mayor. Se permite la existencia de múltiples TipoRoles marcados como tipolnicial, pues es posible que el sistema se modifique para que se realicen ciertos cambios de roles de forma automática, y este valor podría servir.

Usuario

La tabla Usuario representará cada cuenta de usuario registrada en el sistema. Se muestra en la figura 7.4. Sus atributos son los siguientes:

- **id**: Clave primaria que identificará a cada usuario.
- **username**: Nombre de usuario, único para cada usuario.
- **email**: Correo electrónico, único para cada usuario.
- **pwhash**: Hash de la contraseña, que se comprobará al iniciar sesión. Así se evita almacenar la contraseña, por si se filtra la información.
- **fullname**: Nombre completo del usuario.

TokenCorreo

La tabla TokenCorreo almacena los tokens enviados por correo a usuarios, que sirven para diferentes acciones como recuperar la contraseña, recibiendo una nueva. Una vez utilizado se debería eliminar. Se muestra en la figura 7.5. Sus atributos son:

- **token**: Cadena que representa el valor del token enviado al usuario.
- **caducidad**: Fecha tras la cual el token no podrá ser utilizado.

TipoRol
<ul style="list-style-type: none">- id: int {id}- rolBase: RolUsuarioE- nombre: string {ak}- prioridad: int- verPTests: bool- verETests: bool- verEQuestions: bool- verPQuestions: bool- verAnswers: bool- changeRoles: bool- tenerTeams: bool- tenerEQuestions: bool- tenerETests: bool- tenerPTests: bool- adminPTests: bool- adminETests: bool- adminEQuestions: bool- adminAnswers: bool- adminUsers: bool- adminTeams: bool- adminConfiguration: bool- adminPermissions: bool- tipoInicial: bool

Figura 7.3: Detalle de Tabla TipoRol

Usuario
<ul style="list-style-type: none">- id: int {id}- username: String {ak}- email: String {ak}- pwhash: String- fullname: String

Figura 7.4: Detalle de Tabla Usuario

TokenCorreo
<ul style="list-style-type: none">- token: string {id}- caducidad: DateTime

Figura 7.5: Detalle de Tabla TokenCorreo

Equipo

La tabla Equipo representa los diferentes equipos de usuarios que hay en el sistema. Se muestra en la figura 7.6 y sus atributos son los siguientes:

- **id**: Identificador numérico que funcionará como clave primaria de las instancias.
- **teamname**: Nombre del equipo, el cual será la manera que tendrán los usuarios de referirse al equipo. Será único entre todos los equipos.
- **description**: Descripción del equipo.
- **soloProfesores**: Valor que indica si el equipo solo puede tener usuarios que sean profesores (y administradores), o si puede tener usuarios base también.

Equipo
- id: int {id}
- teamname: String {ak}
- description: String
- soloProfesores: bool

Figura 7.6: Detalle de Tabla Equipo

EquipoUsuario

La tabla EquipoUsuario representa entidades débiles que permiten relacionar usuarios y equipos respetando las multiplicidades. También almacena el rol que tiene un usuario en un equipo. Se muestra en la figura 7.7 y sus atributos son los siguientes:

- **rol**: Rol que tiene el usuario en el equipo. Los valores posibles de RolEquipoUsuarioE se explican en la sección 7.1.2.

EquipoUsuario
- rol: RolEquipoUsuarioE

Figura 7.7: Detalle de Tabla EquipoUsuario

Test

La tabla Test representa los exámenes almacenados en el sistema. Tienen asociados un conjunto de Preguntas, tal y como se explica en la figura 7.1, y un conjunto de atributos de gestión y control de entregas y respuestas. Se muestra en la figura 7.8 y sus atributos son los siguientes:

- **id**: Identificador numérico del examen que actuará como clave primaria.
- **title**: Título del examen.
- **description**: Descripción del examen.
- **maxMinutes**: Cantidad de minutos máximas que tiene un usuario para resolver el examen.

- **accesoPublico**: Indica, en caso de que sea un examen publicado, si cualquier usuario con acceso a exámenes publicados puede ver y realizar el examen. En el caso contrario, se necesitará una invitación para poder realizar el examen.
- **editable**: Indica si todos los datos del examen se pueden modificar por un usuario con permisos. Si sí se pueden modificar es un examen borrador, los cuales no se pueden realizar, y si no, es un examen publicado.
- **accesoPublicoNoPublicado**: En caso de ser un examen borrador, indica si cualquier usuario con acceso a los exámenes borradores puede acceder a este examen.
- **horaCreacion**: Fecha y hora de creación del examen.
- **autoCorrect**: Valor que indica si las respuestas al examen se marcarán como corregidas una vez entregadas, o si necesitarán confirmación de un usuario con capacidad de administración del examen para confirmar la corrección de las respuestas.
- **visibilidad**:
- **cantidadFavoritos**: Indica la cantidad de usuarios que han marcado como favorito el examen. Este valor se podría calcular observando otras tablas, pero mantener el dato aquí es mucho más eficiente para las consultas que se van a realizar.
- **tiempoEstricto**: Valor que indica cuando es negativo si, en caso de superar la cantidad de tiempo indicada en maxMinutes, el usuario podrá seguir entregando el examen o si, por el contrario, ya no podrá.
- **maxIntentos**: Cantidad máxima de intentos que tiene un usuario para realizar el test. Si es cero, no hay límite de intentos.

Test
<ul style="list-style-type: none"> - id: int {id} - title: String - description: String - maxMinutes: int - accesoPublico: bool - editable: bool - accesoPublicoNoPublicado: bool - horaCreacion: DateTime - autoCorrect: bool - visibilidad: VisibilidadTestE - cantidadFavoritos: int - tiempoEstricto: bool - maxIntentos: int

Figura 7.8: Detalle de Tabla Test

GestionTestEquipo

La tabla GestionTestEquipo es la entidad débil que representa la asociación de los exámenes y los equipos que tienen derechos de administración sobre los mismos. Se puede ver en la figura 7.9.

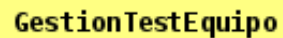
Un rectángulo amarillo con un borde negro que contiene el texto "GestionTestEquipo" en negrita.

Figura 7.9: Detalle de Tabla GestionTestEquipo

InvitacionTestEquipo

La tabla `InvitacionTestEquipo` es la entidad débil que representa la asociación de los exámenes y los equipos que tienen una invitación a la realización de dicho examen. Se puede ver en la figura 7.10.

Un rectángulo amarillo con un borde negro que contiene el texto "InvitacionTestEquipo" en negrita.

Figura 7.10: Detalle de Tabla InvitacionTestEquipo

InvitacionTestUsuario

La tabla `InvitacionTestUsuario` es la entidad débil que representa la asociación de los exámenes y los usuarios que tienen una invitación a la realización de dicho examen. Se puede ver en la figura 7.11.


Un rectángulo amarillo con un borde negro que contiene el texto "InvitacionTestUsuario" en negrita.

Figura 7.11: Detalle de Tabla InvitacionTestUsuario

TestFavorito

La tabla `TestFavorito` es la entidad débil que representa la asociación de los exámenes y los usuarios que tienen marcado ese examen como favorito. Se puede ver en la figura 7.12.

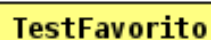
Un rectángulo amarillo con un borde negro que contiene el texto "TestFavorito" en negrita.

Figura 7.12: Detalle de Tabla TestFavorito

Pregunta

La tabla Pregunta almacena las diferentes preguntas que han creado los usuarios. El tipo de la pregunta depende del valor del campo “solucion”, donde, si no es nulo, será de respuesta de texto. Si es nulo y “eleccionUnica” no es nulo, será de respuesta múltiple o tipo test, y si ambos son nulos será una pregunta práctica de programación. Se puede ver la tabla en la figura 7.13 y a continuación se explicarán sus atributos:

- **id**: Identificador numérico del examen que actuará como clave primaria.
- **title**: Título de la pregunta.
- **question**: Cuerpo de la pregunta.
- **estimatedTime**: Tiempo, en minutos, que se estima necesario para completar la respuesta a la pregunta.
- **autoCorrect**: Indica si las respuestas a la pregunta se corregirán automáticamente una vez se entregue la respuesta al examen.
- **penalizacion**: Porcentaje del valor final de la pregunta en el examen que se descontará de la nota de la respuesta al examen en caso de que se entregue una respuesta errónea para esta pregunta.
- **editable**: Indica si la pregunta puede modificarse completamente por un usuario con permisos de administración de la misma. En ese caso se considera una pregunta borrador. Si es negativo, la pregunta formará parte de un examen publicado.
- **eleccionUnica**: Valor que indica, en caso de ser una pregunta de tipo test o respuesta múltiple, si se puede seleccionar exclusivamente una opción o si se pueden seleccionar varias.
- **solucion**: Respuesta esperada para las preguntas de texto.
- **accesoPublicoNoPublicada**: En caso de ser una pregunta borrador, es decir, una pregunta que no forma parte de un examen publicado, indica si cualquier usuario con acceso a las preguntas borradores puede acceder a esta pregunta.
- **cantidadFavoritos**: Indica la cantidad de usuarios que han marcado como favorita la pregunta. Este valor se podría calcular observando otras tablas, pero mantener el dato aquí es mucho más eficiente para las consultas que se van a realizar.

PreguntaFavorita

La tabla PreguntaFavorita es la entidad débil que representa la asociación de las preguntas y los usuarios que tienen marcada esa pregunta como favorita. Se puede ver en la figura 7.14.

TestPregunta

La tabla TestPregunta es la entidad débil que representa la asociación de las preguntas y los exámenes a los que pertenecen. Se puede ver en la figura 7.15, y a continuación se explicarán sus atributos:

- **valorFinal**: Valor que tiene la pregunta en la puntuación total del examen al que está asociada.

Pregunta
<ul style="list-style-type: none"> - id: int {id} - title: String - question: String - estimatedTime: int - autoCorrect: bool - penalizacion: int - editable: bool - eleccionUnica: bool - solucion: String - accesoPublicoNoPublicada: bool - cantidadFavoritos: int

Figura 7.13: Detalle de Tabla Pregunta

PreguntaFavorita

Figura 7.14: Detalle de Tabla PreguntaFavorita

- **posicion:** Valor que permite obtener la posición de la pregunta en el examen respecto a las demás. Cuanto menor sea el valor, antes va. En caso de que dos preguntas tengan el mismo valor de "posicion", se considera primera la que tenga un menor valor en el campo "id".

TestPregunta
<ul style="list-style-type: none"> - valorFinal: int - posicion: int

Figura 7.15: Detalle de Tabla TestPregunta

PreguntaEquipo

La tabla PreguntaEquipo es la entidad débil que representa la asociación de las preguntas y los equipos que tienen derechos de administración sobre las mismas. Se puede ver en la figura 7.16.

PreguntaEquipo

Figura 7.16: Detalle de Tabla PreguntaEquipo

Opcion

La tabla Opcion almacena las diferentes opciones existentes de las preguntas de respuesta múltiple o tipo test. Se puede ver en la figura 7.17 y sus atributos son:

- **indice**: Identificador de la opción, que actúa como parte de la clave primaria.
- **texto**: Cuerpo de la opción.
- **correcta**: Indica si la opción es una opción correcta y debería seleccionarla el usuario, o si, por el contrario, es una opción errónea.

Opcion
- indice: int {id}
- texto: String
- correcta: bool

Figura 7.17: Detalle de Tabla Opcion

Etiqueta

La tabla Etiqueta almacena las etiquetas de exámenes y preguntas que se utilizan o han sido utilizadas. Se puede ver en la figura 7.18 y sus atributos son:

- **nombre**: Valor de la etiqueta en sí, que además actúa como clave primaria.

Etiqueta
nombre: String {id}

Figura 7.18: Detalle de Tabla Etiqueta

PreguntaEtiqueta

La tabla PreguntaEtiqueta es la entidad débil que representa la asociación de las preguntas y las etiquetas que tienen asociadas. Se puede ver en la figura 7.19.

PreguntaEtiqueta

Figura 7.19: Detalle de Tabla PreguntaEtiqueta

TestEtiqueta

La tabla TestEtiqueta es la entidad débil que representa la asociación de los exámenes y las etiquetas que tienen asociadas. Se puede ver en la figura 7.20.

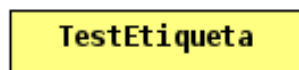


Figura 7.20: Detalle de Tabla TestEtiqueta

RespuestaExamen

La tabla RespuestaExamen almacena las respuestas de los usuarios a los exámenes. Es la composición de un conjunto de RespuestaPreguntas, que coinciden con las Preguntas del Test asociado a la RespuestaExamen. Se muestra en la figura 7.21 y sus atributos son los siguientes:

- **id**: Identificador numérico de la respuesta al examen que actúa como clave primaria.
- **startTime**: Fecha y hora a las que se inició la respuesta.
- **finishTime**: Fecha y hora a las que se finalizó la respuesta.
- **entregado**: Indica si la respuesta fue entregada manualmente.
- **visibleParaUsuario**: Indica si el usuario autor de la respuesta tiene acceso a la lectura de los detalles de la respuesta y las RespuestaPreguntas asociadas.

RespuestaExamen
- id: int {id}
- startTime: DateTime
- finishTime: DateTime
- entregado: bool
- visibleParaUsuario: bool

Figura 7.21: Detalle de Tabla RespuestaExamen

RespuestaPregunta

La tabla RespuestaPregunta almacena las respuestas a las preguntas de exámenes publicados. Se muestra en la figura 7.22 y sus atributos son los siguientes:

- **puntuacion**: Calificación obtenida en la RespuestaPregunta. Es un porcentaje, su valor final será ese porcentaje respecto al "valorFinal" de la Pregunta en el Test.
- **corregida**: Indica si la RespuestaPregunta está corregida.

- **respuesta:** Respuesta asociada a la pregunta en caso de ser una pregunta de respuesta de texto.
- **estado:** Estado de la ejecución y compilación de la respuesta. Servirá para saber si la ejecución se ha ejecutado con éxito, si ha habido problemas en la compilación, o si todavía se está ejecutando. Los tipos de EstadoRespCodigoE se explican en el apartado correspondiente de la sección 7.1.2.
- **errorCompilacion:** En caso de ser una respuesta a una pregunta práctica de programación, almacena la salida emitida por el compilador que contiene el error/los errores de compilación en caso de haberse producido alguno.

RespuestaPregunta
<ul style="list-style-type: none"> - puntuacion: int - corregida: bool - respuesta: String - estado: EstadoRespCodigoE - errorCompilacion: String

Figura 7.22: Detalle de Tabla RespuestaPregunta

OpcionRespuesta

La tabla OpcionRespuesta es la entidad débil que representa la asociación de las RespuestaPreguntas y las Opciones que tienen asociadas, que son las elegidas como respuesta por el usuario. Se puede ver en la figura 7.23.

OpcionRespuesta

Figura 7.23: Detalle de Tabla OpcionRespuesta

Prueba

La tabla Prueba almacena la información relativa a cada una de las pruebas unitarias de ejecución a las que se someterán las diferentes respuestas de los usuario para una pregunta práctica de programación determinada. Se muestra en la figura 7.24 y sus atributos son los siguientes:

- **id:** Identificador de cada instancia
- **entrada:** La entrada estándar que se le dará en la ejecución
- **salida:** La salida que se espera obtener por salida estándar para que se considere superada la prueba

- **visible**: Indica si el usuario que está respondiendo al examen puede ver la entrada y la salida de la prueba, o si esos datos solo son accesibles.
- **postEntrega**: Indica si la prueba se ejecutará únicamente después de entregar el examen, o si también se ejecutará cuando el usuario pruebe su respuesta durante la realización de la misma.
- **valor**: Valor que tiene la Prueba respecto a la puntuación total de la pregunta. Su valor final en la pregunta consiste en el valor de la prueba dividido entre la suma de los valores de todas las pruebas de la pregunta.

Prueba
- id: int {id}
- entrada: String
- salida: String
- visible: bool
- postEntrega: bool
- valor: int

Figura 7.24: Detalle de Tabla Prueba

Ejecución

La tabla Estado almacena la información relativa a la ejecución de una respuesta de una pregunta práctica de programación frente a una prueba concreta. Se muestra en la figura 7.25 y sus atributos son los siguientes:

- **estado**: Estado de la ejecución. Servirá para saber si la ejecución se ha ejecutado con éxito, y en caso contrario cual ha sido el error. Los tipos de EstadoEjecucionE se explican en el apartado correspondiente de la sección 7.1.2.
- **salidaReal**: Salida estándar obtenida tras la ejecución de la prueba. Si coincide con la salida esperada de la prueba, la ejecución será correcta.

Ejecución
- estado: EstadoEjecucionE
- salidaReal: String

Figura 7.25: Detalle de Tabla Ejecución

7.1.2. Tipos enumerados

En las diferentes tablas se usan algunos tipos enumerados, que sirven para mantener una variable que tenga una cantidad de valores posibles restringida. A continuación se explican los existentes en esta base de datos.

VisibilidadTestE

VisibilidadTestE representa cuándo se puede marcar como visible una respuesta entregada a un examen. Se puede ver en la figura 7.26, y los valores posibles son los siguientes:

- **AlEntregar**: La RespuestaExamen se marcará como visible cuando el usuario entregue la RespuestaExamen. (Siempre será visible).
- **AlCorregir**: La RespuestaExamen se marcará como visible cuando la RespuestaExamen se marque como corregida.
- **Manual**: La RespuestaExamen tendrá que ser marcada como visible manualmente por un profesor responsable.

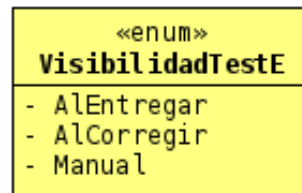


Figura 7.26: Detalle de Tipo Enumerado VisibilidadTestE

RolEquipoUsuarioE

RolEquipoUsuarioE representa los tipos de roles que puede tomar un usuario en un equipo. Se puede ver en la figura 7.27, y los valores posibles son los siguientes:

- **Miembro**: El usuario es simplemente un miembro del equipo.
- **Admin**: El usuario es administrador del equipo y puede gestionarlo.

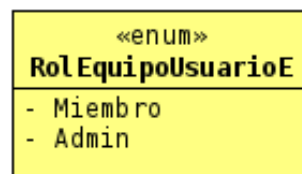


Figura 7.27: Detalle de Tipo Enumerado RolEquipoUsuarioE

RolUsuarioE

RolUsuarioE representa los roles básicos que puede tener un tipo de rol de los usuarios. Se basan en los actores de la sección 6.3, pues servirán para facilitar al usuario, según qué actor sea, los casos de uso más importantes para él. Se puede ver en la figura 7.28, y sus valores posibles son los siguientes:

- **NoRegistrado**: Representa al actor Usuario no registrado.
- **Estudiante**: Representa al actor Usuario (Usuario base).
- **Profesor**: Representa al actor Profesor.
- **Administrador**: Representa al actor Administrador.

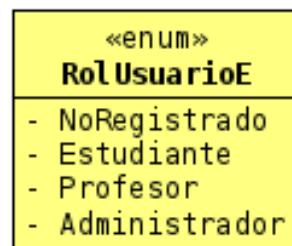


Figura 7.28: Detalle de Tipo Enumerado RolUsuarioE

EstadoEjecucionE

EstadoEjecucionE representa los estados en los que puede estar una ejecución de una prueba. Se pueden ver en la figura 7.29, y sus valores son los siguientes:

- **Correcto**: La ejecución de la prueba ha finalizado y se ha superado con éxito
- **TiempoExcedido**: La ejecución de la prueba se ha interrumpido porque la ejecución ha durado más del tiempo de ejecución máximo.
- **ErrorRuntime**: La ejecución de la prueba se ha interrumpido porque se ha producido un error en tiempo de ejecución.
- **SalidaIncorrecta**: La ejecución de la prueba ha finalizado pero la salida resultante no corresponde con la salida esperada.

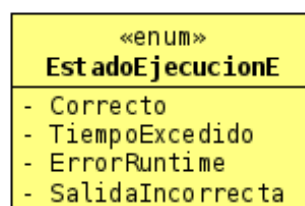


Figura 7.29: Detalle de Tipo Enumerado EstadoEjecucionE

EstadoRespCodigoE

EstadoRespCodigoE representa los estados en los que puede estar una respuesta a una pregunta práctica de programación. Se pueden ver en la figura 7.30, y sus valores son los siguientes:

- **NoProbado**: La respuesta todavía no se ha intentado probar contra las pruebas.
- **ErrorCompilacion**: La respuesta se ha intentado probar y se ha producido un error en la compilación.
- **Ejecutando**: La respuesta se está probando en estos momentos.
- **Probado**: La respuesta se ha ejecutado contra las pruebas y no ha habido ningún error de compilación.

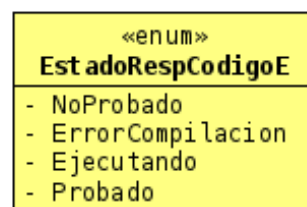


Figura 7.30: Detalle de Tipo Enumerado EstadoRespCodigoE

7.1.3. Restricciones OCL

Para describir formalmente ciertas restricciones sobre los modelos entidad relación previamente descritos utilizando UML, se ha utilizado OCL 2.4[25]. Únicamente se han descrito formalmente las restricciones más importantes del sistema.

VistaPersonalizada

```

context VistaPersonalizada inv:
VistaPersonalizada.allInstances() -> exists(v| v.rolBase = 'NoRegistrado')
VistaPersonalizada.allInstances() -> exists(v| v.rolBase = 'Estudiante')
VistaPersonalizada.allInstances() -> exists(v| v.rolBase = 'Profesor')
VistaPersonalizada.allInstances() -> exists(v| v.rolBase = 'Administrador')
  
```

TipoRol

```

context TipoRol inv:
self.rolBase = 'NoRegistrado' implies self.usuarios -> isEmpty()
self.prioridad >= 0
TipoRol.allInstances() -> exists(t| t.tipoInicial = true)
TipoRol.allInstances() -> exists(t| t.adminPermissions = true)

self.rolBase <> 'Administrador' implies
  self.adminPTests = false and self.adminETests = false and
  
```

```

self.adminEQuestions = false and self.adminAnswers = false and
self.adminUsers = false and self.adminTeams = false and
self.adminConfiguration = false and self.adminPermissions = false

```

```

TipoRol.allInstances() -> exists(t| t.rolBase = 'NoRegistrado')
TipoRol.allInstances() -> exists(t| t.rolBase = 'Estudiante')
TipoRol.allInstances() -> exists(t| t.rolBase = 'Profesor')
TipoRol.allInstances() -> exists(t| t.rolBase = 'Administrador')

```

Equipo

```

context Equipo inv:
self.equipoUsuarios -> exists(e| e.rol='Admin')
self.soloProfesores=true implies self.usuarios.tipoRol.rolBase<>'Estudiante'

```

EquipoUsuario

```

context EquipoUsuario inv:
self.rol=='Admin' implies self.usuario.tipoRol.rolBase<>'Estudiante'

```

Test

```

context Test inv:
self.usuario.tipoRol.rolBase<>'Estudiante'
self.cantidadFavoritos = self.testFavoritos -> size()

self.editable=true implies
    self.preguntas -> forAll(p| p.editable=true) and self.test -> notEmpty()

self.editable=false implies
    self.preguntas -> forAll(p| p.editable=false) and self.test -> isEmpty()

self.tests -> forAll(t| t.editable=false)

```

InvitacionTestUsuario

```

context InvitacionTestUsuario inv:
self.test.editable=false

```

InvitacionTestEquipo

```

context InvitacionTestEquipo inv:
self.test.editable=false

```

GestionTestEquipo

```

context GestionTestEquipo inv:
self.equipo.soloProfesores=true

```

Pregunta

```
context Pregunta inv:
self.cantidadFavoritos = self.preguntaFavoritas -> size()
self.usuario.tipoRol.rolBase<>'Estudiante'
self.penalizacion >=0
self.penalizacion <=100

self.opciones -> notEmpty() implies self.eleccionUnica -> notEmpty()

self.eleccionUnica -> notEmpty() implies
  self.opciones -> notEmpty() and self.solucion -> isEmpty()

self.solucion -> notEmpty() implies
  self.opciones -> isEmpty() and self.eleccionUnica -> isEmpty()

self.editable=false implies
  self.tests -> size() = 1 and self.tests -> forAll(t| t.editable=false)

self.editable=true implies self.tests -> forAll(t| t.editable=true)
```

PreguntaEquipo

```
context PreguntaEquipo inv:
self.equipo.soloProfesores=true
```

RespuestaExamen

```
context RespuestaExamen inv:
self.test.editable=false
```

RespuestaPregunta

```
context RespuestaPregunta inv:
self.pregunta.editable=false

self.opciones -> notEmpty() implies
  self.respuesta -> isEmpty() and self.pregunta.opciones -> notEmpty()

self.opciones -> size() > 1 implies self.pregunta.eleccionUnica=true

self.respuesta -> notEmpty() implies
  self.opciones -> isEmpty() and self.pregunta.solucion -> notEmpty()

self.puntuacion<=100
self.puntuacion>=-100
```

Prueba

```
context Prueba inv:
self.postEntrega=true implies
```

```
self.visible=false
```

```
self.valor >= 0
```

7.2. Diseño por Capas

La aplicación utiliza un diseño basado en capas, donde el usuario ejecuta en su máquina la Capa de Presentación o *FrontEnd*, explicada en la sección 7.2.2, la cual se comunica con la capa de negocio mediante una API REST, explicada en la sección 7.2.4, utilizando el patrón DTO y los modelos definidos en la sección 7.2.3. La capa de negocio se comunica con la capa de persistencia, la cual usa el patrón DAO, y esta última capa se comunica directamente con la base de datos. Estas dos capas forman el *BackEnd*, y se explican en la sección 7.2.1.

7.2.1. Capas de Negocio y Persistencia: BackEnd

En la figura 7.31 se muestra, de forma simplificada, el diseño seguido en el BackEnd.

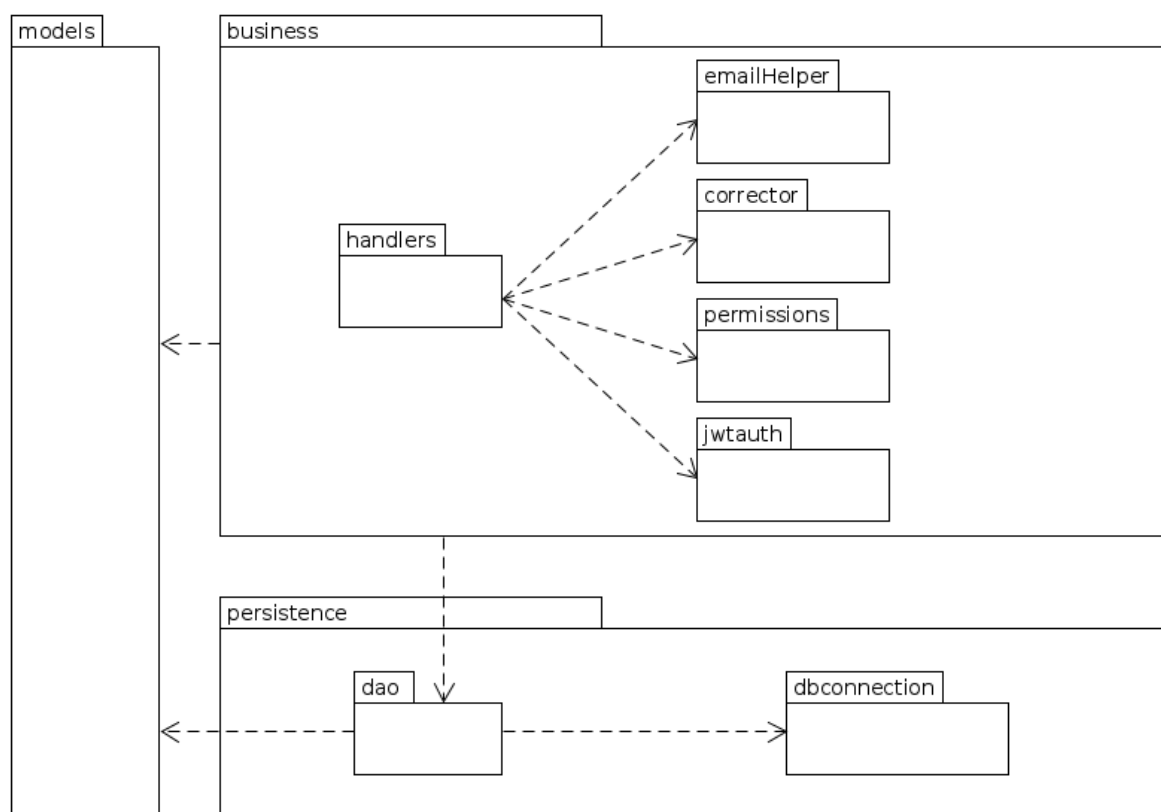


Figura 7.31: Diagrama de Capas de Negocio y Persistencia simplificado.

El BackEnd se escribió utilizando Go (Golang), tal y como se explica en la sección 8.1, lo cual tiene implicaciones en el diseño como, por ejemplo, que no es posible la existencia de métodos estáticos en las clases, sino que tienen que ser de alcance de paquete, o que el patrón *Singleton* ha de ser

implementado mediante el uso de semáforos binarios o *mutex*.

Capa de Negocio

El paquete “handlers” es el paquete que contiene las operaciones que proporcionarán las funcionalidades de la API. Siguen la estructura necesaria pedida por el framework *Go-Swagger*, el cual se explica en la sección 8.1.

El paquete “emailHelper” contiene las operaciones necesarias para acceder a los datos del correo electrónico del sistema y modificarlos, y para enviar correos electrónicos a los usuarios.

El paquete “corrector” contiene las operaciones encargadas de la funcionalidad de corrección de preguntas, ya sean teóricas o prácticas de programación.

El paquete “permissions” contiene las operaciones necesarias para saber si un usuario tiene los permisos y autorizaciones que necesita para realizar una operación.

El paquete “jwtauth” implementa operaciones y estructuras que permiten la generación y validación de los JWT, en los cuales se basa la autenticación del sistema, tal y como se explica en la sección 7.3.1.

Una vez finalizado el diseño y la implementación, debido a la dificultad de crear tests unitarios para el paquete “handlers”, se ha considerado que habría sido buena idea crear un paquete separado con las operaciones de negocio, ya que las operaciones del paquete “handlers” requieren de parámetros complejos propios del framework, difíciles de imitar.

Capa de Persistencia

El paquete “dbconnection” provee de conexiones con la base de datos mysql/mariadb. Utiliza el patrón *Singleton* para evitar tener que crear una conexión con la base de datos cada vez que se quiera hacer una operación con la misma.

El paquete “dao” utiliza el patrón *DAO (Data Access Object)* y contiene todos los modelos y operaciones que permiten transformar una instancia de alguna clase del paquete “models” a una instancia de una clase correspondiente del paquete “dao”, la cual se puede introducir directamente en la base de datos. También permite hacer lo contrario, obtener una o varias columnas de la base de datos y transformarlas en las instancias de las clases correspondientes del paquete “models”.

Se utiliza el patrón *Singleton* en el paquete “dao” para algunas tablas de la base de datos que vayan a ser rara vez modificadas, pero leídas con mucha frecuencia. Concretamente para las tablas TipoRol y VistaPersonalizada, ya que la primera se necesita acceder a la tabla TipoRol cada vez que se quieran comprobar los permisos de un usuario en la capa “permissions”, lo cual se hace al menos una vez por cada operación de la API, y, en la segunda, se necesita acceder a la tabla VistaPersonalizada cada vez que un usuario abra la *landing page* de la aplicación, lo cual será relativamente frecuente, sin embargo se modificará la información con poca frecuencia.

Capa auxiliar: Models

El paquete “models” corresponde con el DTO y es el definido en la sección 7.2.3, generado también gracias al framework *Go-Swagger*.

7.2.2. Capa de Presentación: FrontEnd

La capa de presentación se diseñará pensando en la utilización de Typescript con el framework Angular. Se sigue el patrón modelo-vista-controlador, donde el modelo será una clase que implemente la interfaz aportada por el DTO y utilice los objetos generados por el DTO, pero en algunos casos, como en las listas de elementos, se usará directamente los modelos generados por el DTO.

La cantidad de componentes creados es demasiada como para incluir una explicación de cada uno en esta memoria, por lo que se hará una explicación general de componentes más importantes, y superclases comunes.

LoggedInController

Todos los controladores de vistas que no sean exclusivas de usuarios no registrados, como la vista de inicio de sesión, heredan de la superclase abstracta “LoggedInController”, la cual implementa muchas operaciones como la comprobación de datos de sesión importantes (que no de autenticación, como ya se explica en la sección 7.3.1) en el almacenamiento local del navegador, funciones que permiten el control de errores, etc. Hay una subclase abstracta que hereda de “LoggedInController” llamada “LoggedInTeacherController”, la cual será la superclase de un subconjunto de vistas que son exclusivas para roles de profesores y administradores.

Listas de elementos

Algunos componentes son listas de elementos, las cuales no se implantan directamente como vistas, sino que se utilizan como partes de otras vistas. Esto se hace para modularizar las vistas de listas de elementos y sus posibles filtros, y para reutilizar esas mismas vistas según la función concreta de la API que se quiera utilizar.

DataService

DataService provee de la capacidad de almacenar mensajes que se mostrarán en todas las vistas, que se mantienen entre elementos de navegación. Para ello se tiene un implementa un Mensaje con diferentes operaciones, implementando un patrón Observador utilizando BehaviourSubject[26].

SessionService

SessionService aporta información global sobre la sesión actual, indicando si el usuario tiene la sesión iniciada y, en ese caso, cual es el nombre de usuario del usuario actual. Almacena los datos en el almacenamiento local del navegador, para que la lógica de la información de sesión se mantenga aunque el usuario cierre y vuelva a abrir la pestaña del navegador. Aquí también se implementa un patrón Observador con BehaviourSubject.

Algunos elementos, como los permisos de cada TipoRol, necesitarían actualizarse a cada petición que haga el usuario, pues los permisos podrían cambiar en cualquier momento. Como es algo poco frecuente, se implementa un almacenamiento en memoria de los TipoRoles con un tiempo de expiración que, por defecto, es de cinco minutos. Cuando se vaya a utilizar esa información y haya pasado el tiempo, se tendrá que hacer otra petición al BackEnd.

Una función importante es “handleErrRelog”, la cual permite realizar una segunda llamada a la función, en caso de que el fallo de la misma sea producido por un error 401, significando que el token de autenticación ha caducado y necesita actualizar los tokens mediante la función “relog” de la API, tal y como se explica en la sección 7.3.1.

```
handleErrRelog<T>(err: any, action: string, primera: boolean,  
  callbackFn: (this: T, prim: boolean) => void, that: T): void
```

InterceptorService

InterceptorService contiene la clase “CustomHttpInterceptorService”, la cual intercepta todas las peticiones http y actúa en consecuencia.

En caso de que el método sea PUT, POST, o DELETE, cambia el mensaje mostrado al usuario para informarle de que se están enviando datos, y, cuando detecta una petición GET, borra el mensaje, salvo que sea un mensaje de error.

También se encarga de rellenar la cabecera “NotLoggedIn” con el valor correspondiente, tal y como se explica en la sección 7.3.1, asignando el valor “true” si el usuario no tiene una sesión iniciada, y “false” si, por el contrario, si que hay una sesión iniciada en el cliente.

7.2.3. DTO

Para transmitir los datos entre ambas capas se ha utilizado el patrón DTO[27], definiendo los esquemas en las especificaciones escritas en Swagger 2.0. Los modelos definidos se pueden ver en la figura 7.32.

No todos los atributos de los DTO diseñados son necesarios, pues para algunas operaciones solo se utilizan ciertos atributos. La información de qué atributos son necesarios y cuando se utilizan está descrita en las especificaciones de la API, descritas usando Swagger 2.0, las cuales se pueden leer en la siguiente dirección web: <https://app.swaggerhub.com/apis/JavgatDevTest/uva-devtest/1.0.0>.

7.2.4. API REST

En esta memoria se utilizará el diagrama creado en UML para exponer la API implementada. Al ser de gran tamaño se ha dividido en cuatro partes, y se necesitaría girar el documento noventa grados en sentido antihorario, o girar la cabeza en noventa grados en sentido horario. Está descrito en las figuras 7.33, 7.34, 7.35 y 7.36.

La descripción completa de todas las operaciones que implementadas en la API con sus comentarios asociados, está en la documentación escrita utilizando Swagger, también conocido como OpenApi2.0. Esto ha permitido la generación de documentación interactiva automática, la cual se

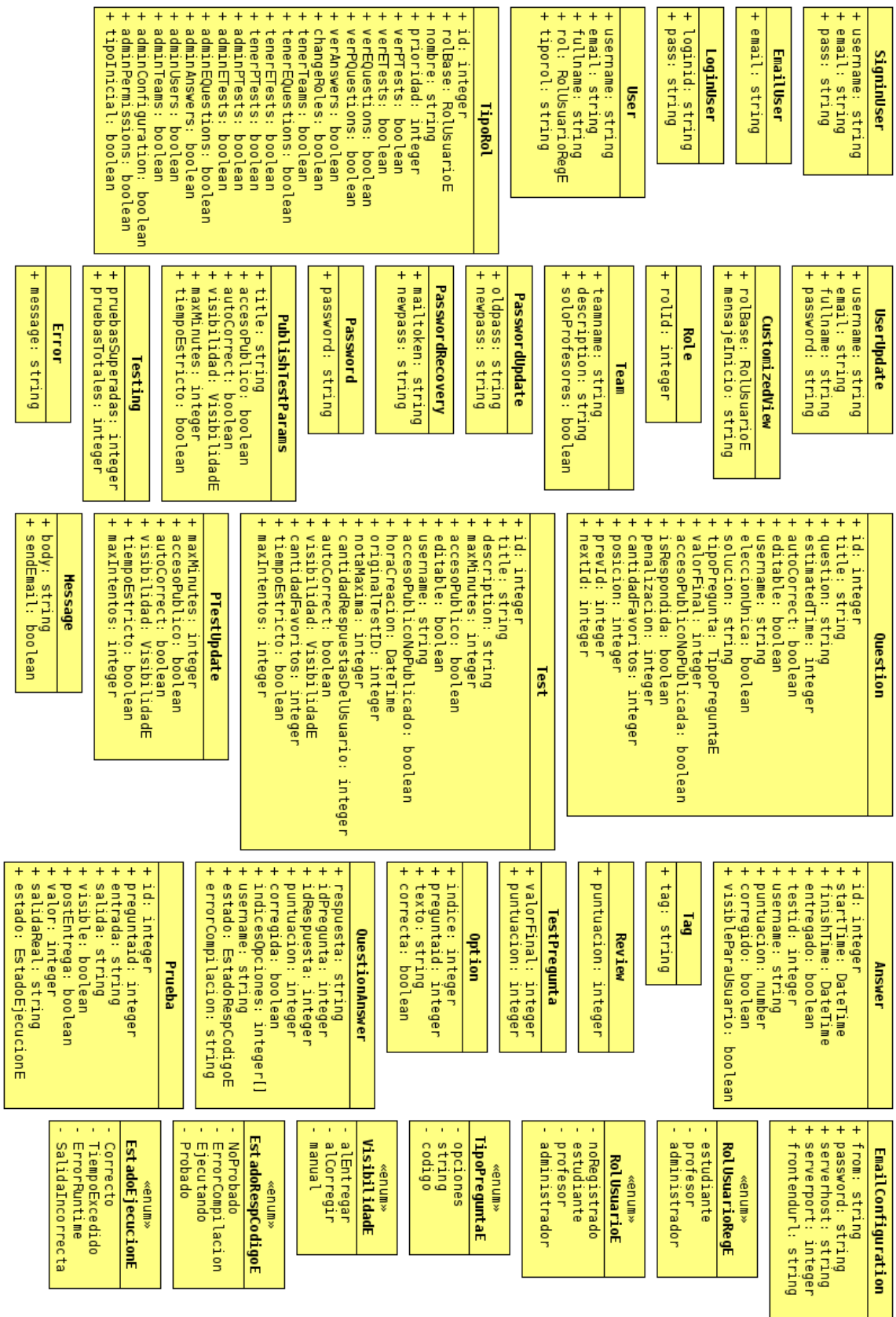


Figura 7.32: Diagrama del DTO

puede leer en la siguiente dirección web: <https://app.swaggerhub.com/apis/JavgatDevTest/uva-devtest/1.0.0> .

7.3. Seguridad de la Aplicación

7.3.1. Autenticación

Tokens JWT

Para almacenar la información de autenticación en el cliente, y hacer ver al servidor que el cliente es quien dice ser, es decir, autenticarse en cada operación, se utilizarán tokens con el formato JWT[28]. En el cuerpo, entre otros datos, irá el correo electrónico del usuario que usa el token. El secreto de la firma será el campo “pwhash” que hay almacenado en la base de datos para el usuario con ese correo electrónico.

Se barajó la posibilidad de utilizar el protocolo OAuth 2.0[29], el cual es muy útil cuando hay muchos tipos de clientes, para evitar que el cliente guarde credenciales del usuario sin que éste lo sepa. En este caso el único cliente va a estar bajo el control del equipo desarrollo, por lo que no es algo necesario, lo que lo convierte en un coste innecesario.

Ataques XSS

Los ataques XSS o ataques de *Cross Site Scripting* son ataques de inyección de código javascript, en las cuales el atacante modifica el cliente web de un usuario, ya sea mediante una extensión de navegador maliciosa o de cualquier otra forma, provocando que el usuario ejecute código malicioso en un sitio web benigno.

Esto supone un peligro, sobre todo si el cliente web tiene acceso a tokens de autenticación, por lo que el atacante podría enviarse ese token para suplantar su identidad. Por esta razón, la fundación OWASP[30] recomienda no almacenar los datos de sesión en el almacenamiento del navegador[31], y recomienda el uso de Cookies con la bandera httpOnly para transmitir los tokens de sesión mitigando los riesgos de ataques XSS.

La bandera httpOnly provoca que el navegador no transmita la cookie al cliente web que se está ejecutando, evitando así que los ataques XSS puedan permitir al atacante obtener el token de sesión del usuario.

Otra técnica seguida para evitar los ataques XSS y similares se explica en la sección 7.3.2.

Ataques man-in-the-middle

Un ataque man-in-the-middle[32] consiste en que el atacante de alguna manera ha vulnerado la seguridad de la red, permitiéndole leer todos los mensajes que se envían a través de la misma.

La manera más sencilla de solucionarlos es asegurarse de utilizar el protocolo HTTPS, que protege la intimidad de los contenidos intercambiados mediante HTTP. Además, la cookie utilizará la bandera

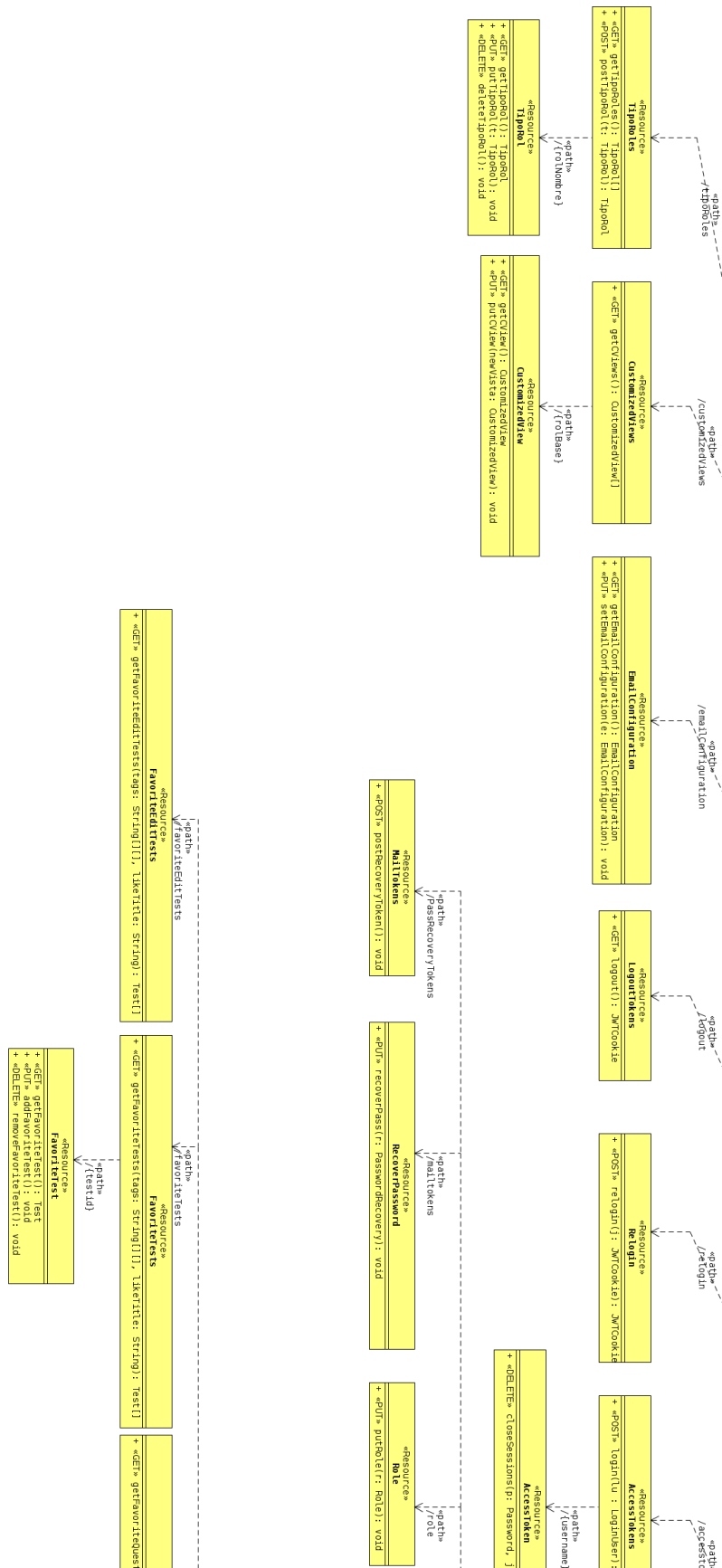


Figura 7.33: Diagrama del diseño de la API REST. Parte 1

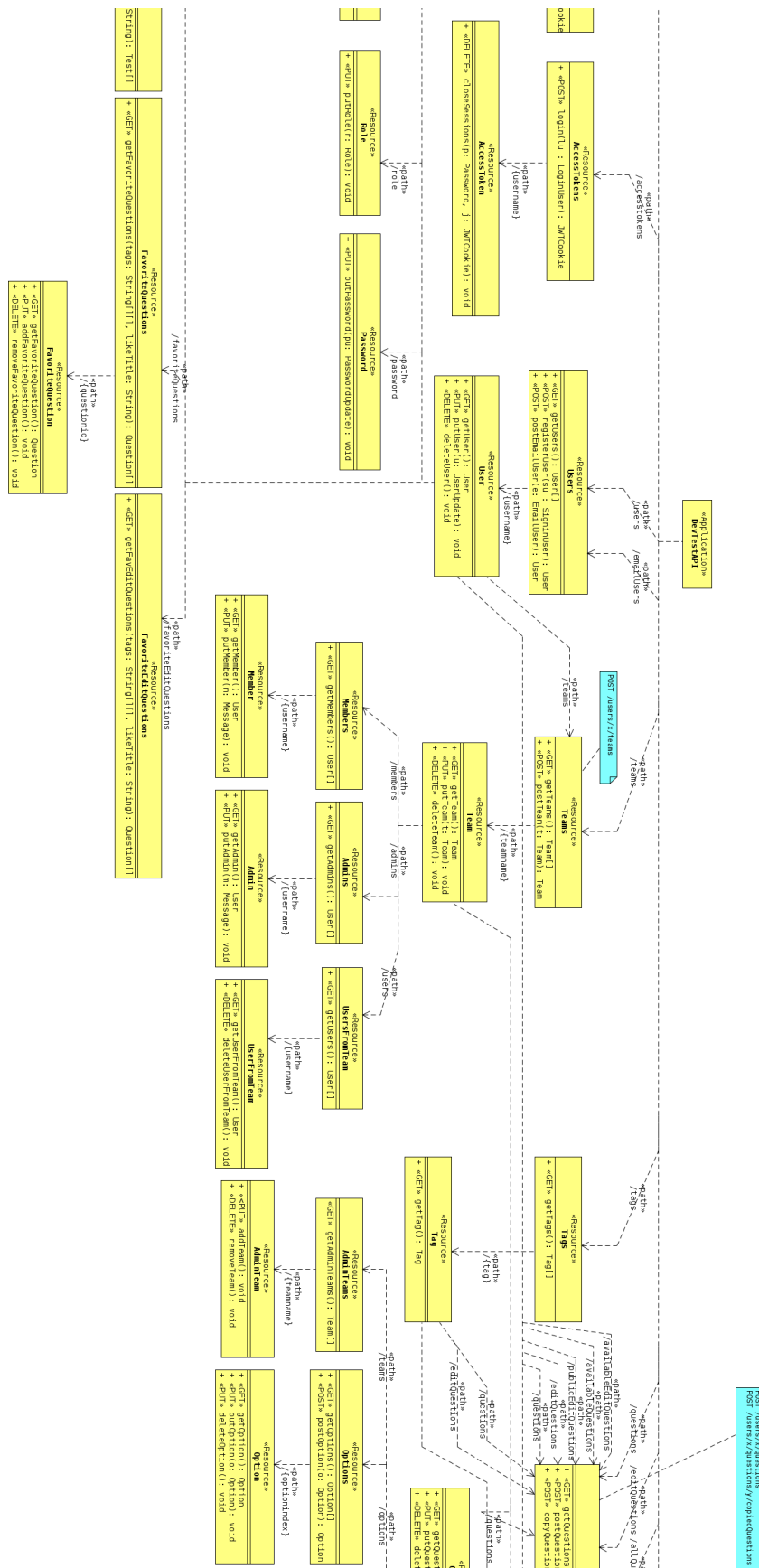


Figura 7.34: Diagrama del diseño de la API REST. Parte 2

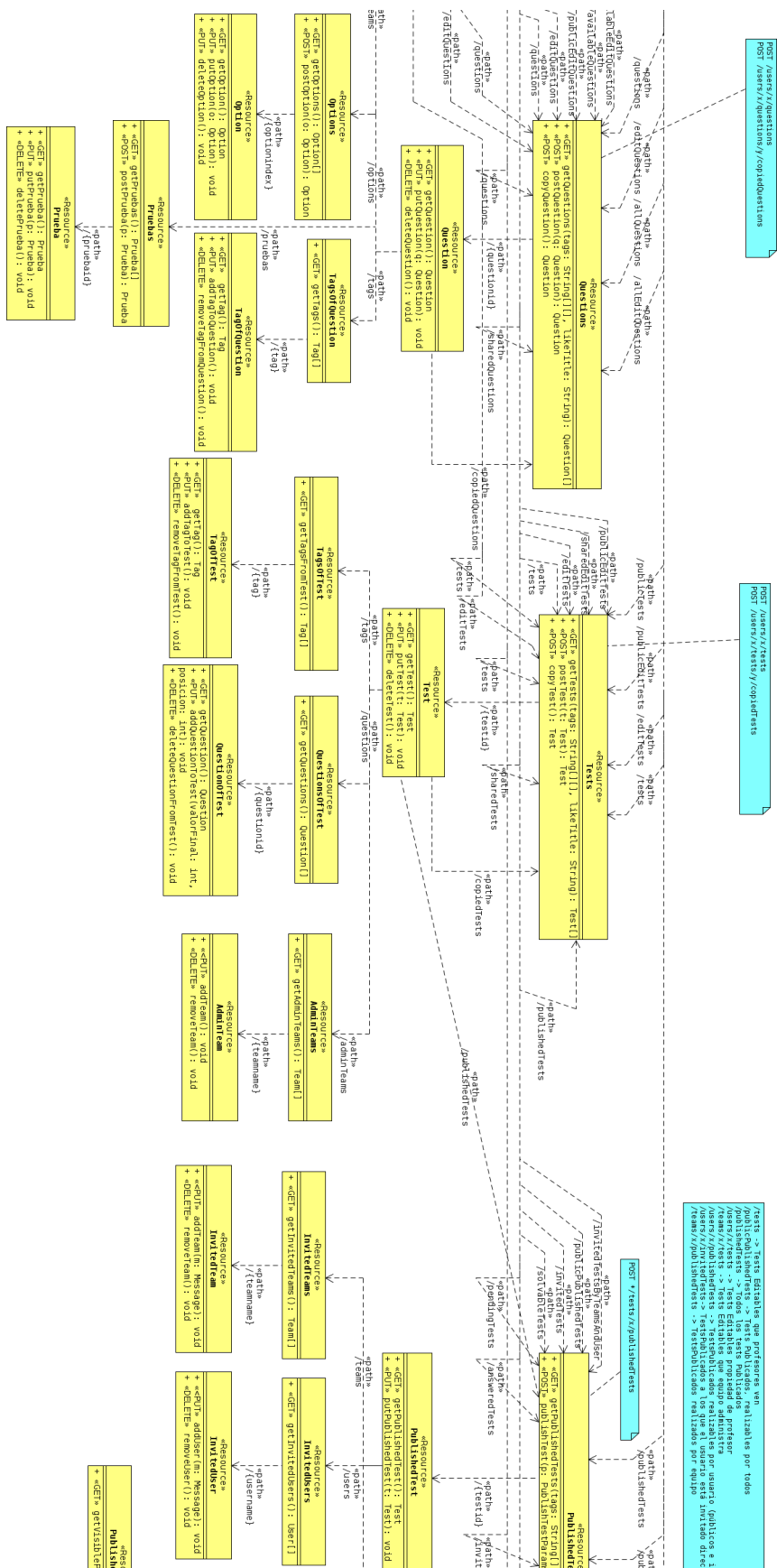


Figura 7.35: Diagrama del diseño de la API REST. Parte 3

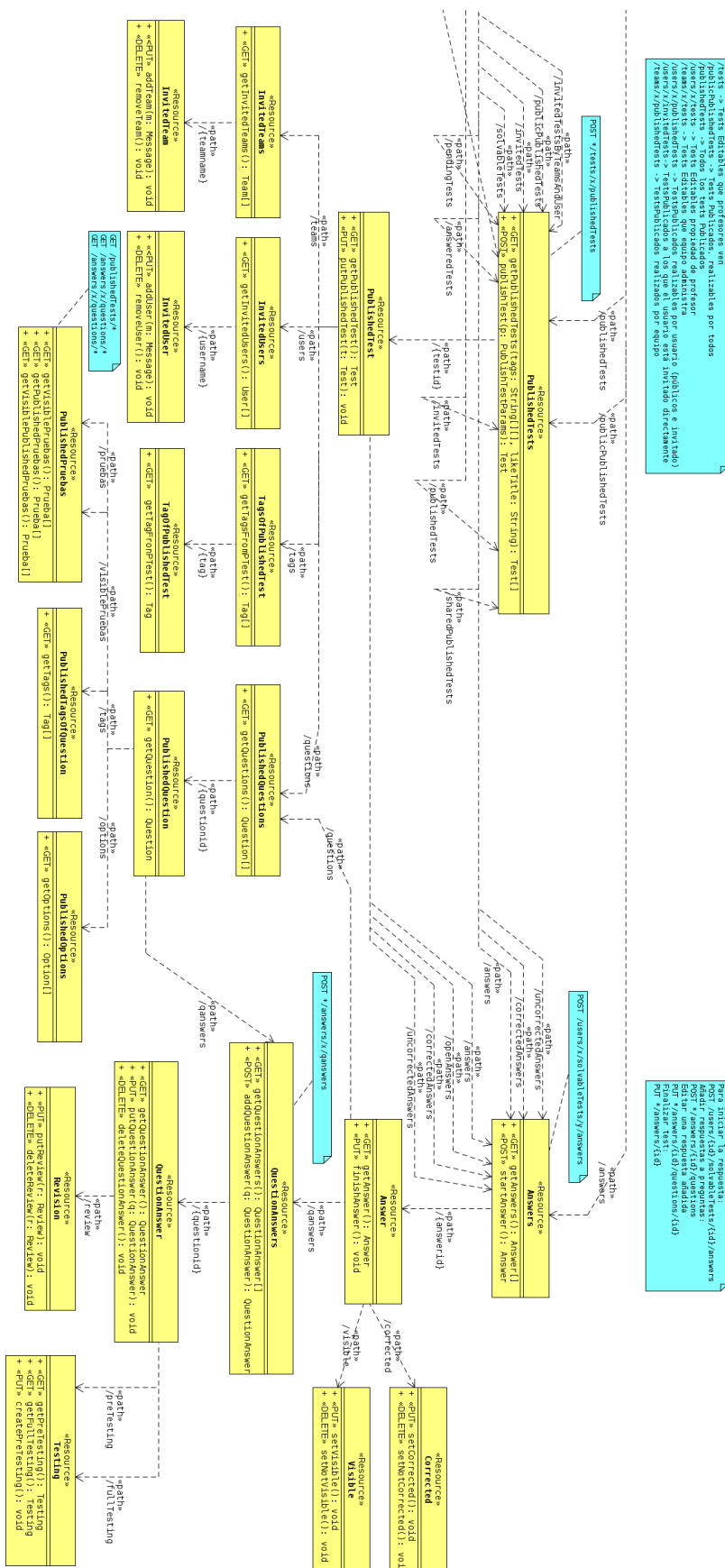


Figura 7.36: Diagrama del diseño de la API REST. Parte 4

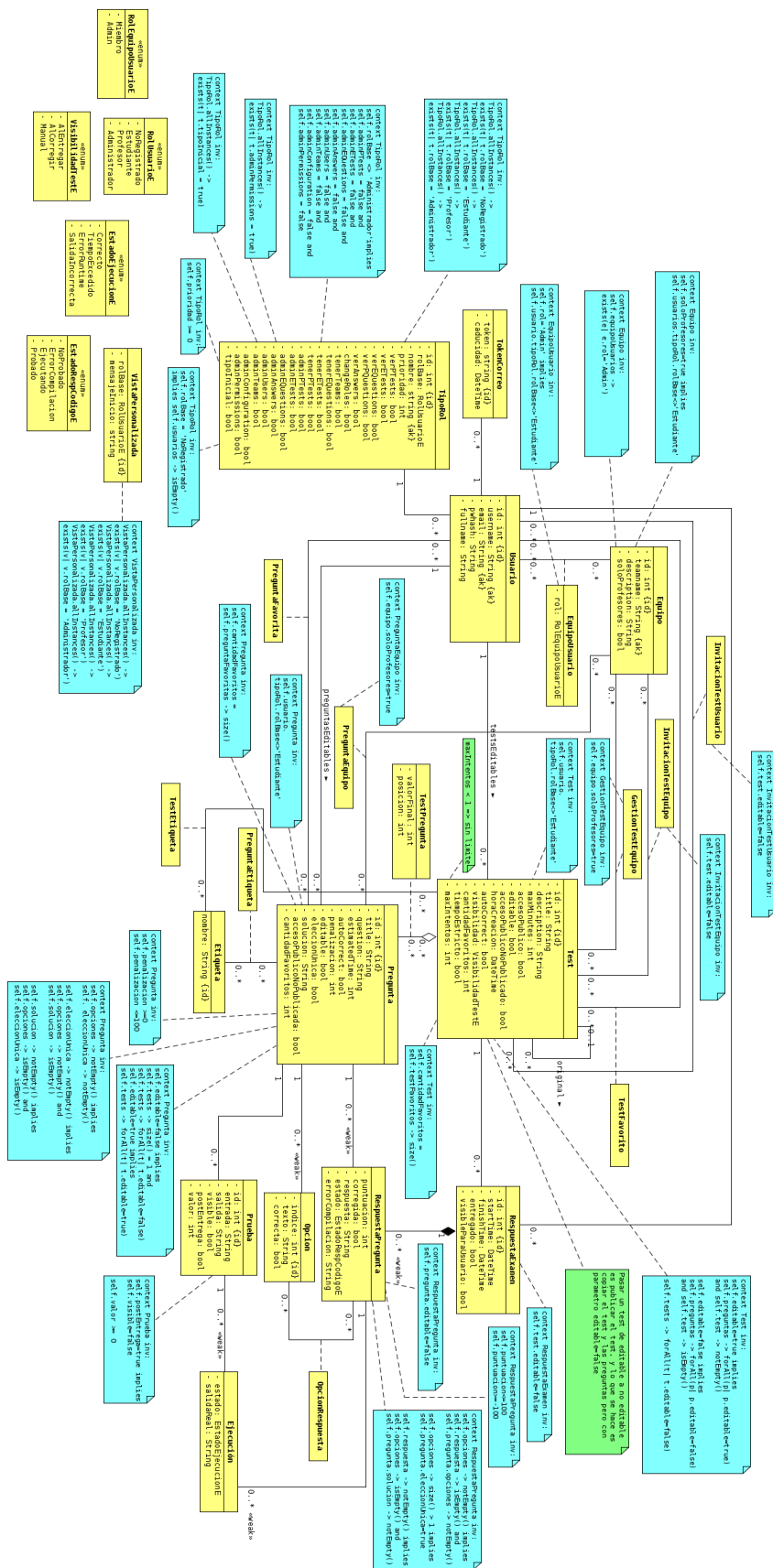


Figura 7.37: Diagrama Entidad Relación Detallado

“secure”, la cual indica que solamente puede ser transmitida por protocolos seguros.

Ataques CSRF/XSRF

Al utilizar cookies para mantener la información de sesión, se genera la posibilidad de ser vulnerables a ataques de *Cross-site request forgery*[33] también llamados ataques CSRF o XSRF.

Estos ataques consisten en que, desde otra página web, se hace una petición (ya sea automáticamente o tras la acción del usuario) hacia el servidor al que haría normalmente el cliente web benigno las peticiones. Como se utiliza una cookie para mantener la sesión, esta cookie se mandaría también en esa petición, por lo que se estaría mandando una petición sin que el usuario lo supiera, suplantándole la identidad.

Para evitar estos ataques se puede utilizar el atributo “SameSite”[34] señalando que estás en modo estricto, lo que indica que esa cookie solo podrá ser enviada a un sitio web directamente o cuando la dirección del cliente web que la está enviando sea la misma que la del servidor web.

Extensión de sesión

La sesión se debe poder mantener durante el tiempo que el usuario esté utilizando el sistema, sin embargo mantener las cookies de sesión eternamente es un error que vulnera la seguridad de la aplicación.

Para solucionar esto, se utilizan dos cookies de sesión. La primera es la que se utilizaba inicialmente, pero con un tiempo de vida máximo de corta duración, por defecto quince minutos, tras el cual la cookie ya no servirá. La segunda cookie, o cookie de reautenticación, contiene la misma información de sesión que la primera, pero tiene un tiempo de vida máximo de larga duración, por defecto dos días. Esta cookie únicamente servirá para una operación, denominada “relog”, que modificará las cookies del cliente actualizándolas para que tenga una cookie de autenticación renovada y una cookie de reautenticación nueva.

Esto significa que habrá situaciones en las que un usuario mandará una petición al BackEnd teniendo la cookie de autenticación caducada. Éste le devolverá el código de error 401, el cual el cliente web interpretará como que se ha caducado la cookie, intentará una reautenticación y, si la realiza con éxito, vuelve a intentar una única vez más la operación inicial.

Algunas operaciones de la API se pueden hacer aunque no estés autenticado pero requieren saber si estás o no autenticado. En ese caso, habrá que especificar una cabecera llamada “NotLoggedIn”, que tendrá el valor “true” en caso de que se esté haciendo la petición de forma no autenticada. Si esa cabecera no se especifica, el servicio de la API asumirá que estás autenticado pero que se ha caducado la cookie, entonces devolverá el código de error 401.

7.3.2. CSP

Como se ha explicado previamente, los ataques XSS son ataques de inyección de código javascript, en las cuales el atacante modifica el cliente web de un usuario, provocando que el usuario ejecute

código malicioso en el cliente web original.

Para prevenir estos ataques y otros ataques de inyección de datos existe la Política de Seguridad del Contenido o *Content Security Policy* (CSP)[35] en inglés, la cual es una capa de seguridad adicional que ayuda a prevenir y mitigar esos tipos de ataques.

Lo que hace, de forma simplificada, es definir de dónde pueden venir los diferentes tipos de recursos de un documento html como los scripts o imágenes, incluyendo como posible origen a permitir o no permitir la inserción de código javascript directamente, lo cual permite evitar ciertos ataques XSS si esto se prohíbe.

A continuación se expondrá la CSP utilizada en el despliegue del proyecto durante el desarrollo. La dirección “https://virtual.lab.inf.uva.es:20173” es la dirección de la web donde está alojada la aplicación, la cual deberá ser sustituida cuando se modifique la dirección en la que está desplegada la aplicación.

```
<meta http-equiv="Content-Security-Policy" content="
  default-src 'self' https://virtual.lab.inf.uva.es:20173;
  img-src * data:;
  script-src 'self' https://cdn.jsdelivr.net https://bootswatch.com;
  font-src 'self' https://fonts.gstatic.com;
  style-src 'self' https://cdn.jsdelivr.net https://bootswatch.com
  https://fonts.googleapis.com 'unsafe-inline';" />
```

7.4. Diseño de Interfaz de Aplicación

En esta sección se tratarán elementos generales del diseño de la aplicación. Para observar los elementos de diseño en su totalidad y poder interactuar con ellos, se puede acceder a la aplicación desplegada, cuya ubicación se explica en la sección 9.3.

7.4.1. Landing pages

Las *landing page*, o páginas de aterrizaje, son las páginas que ven los usuarios cuando entran en la aplicación.

La más importante es la *landing page* que se ve sin haber iniciado sesión, que se puede observar en la figura 7.38, pero también existe una *landing page* para cada tipo de usuario con sesión iniciada.

Todas las de la aplicación tienen una estructura similar, con la barra de navegación arriba, un mensaje de bienvenida personalizable para cada tipo de usuario, que puede incluir imágenes, pues está hecho utilizando *markdown*, y un elemento de interés principal, que en caso de usuarios no registrados son los exámenes públicos (si se les permite ver esos elementos), para usuarios normales son sus exámenes pendientes, y para profesores y administradores son diferentes botones a diferentes páginas importantes del sistema.

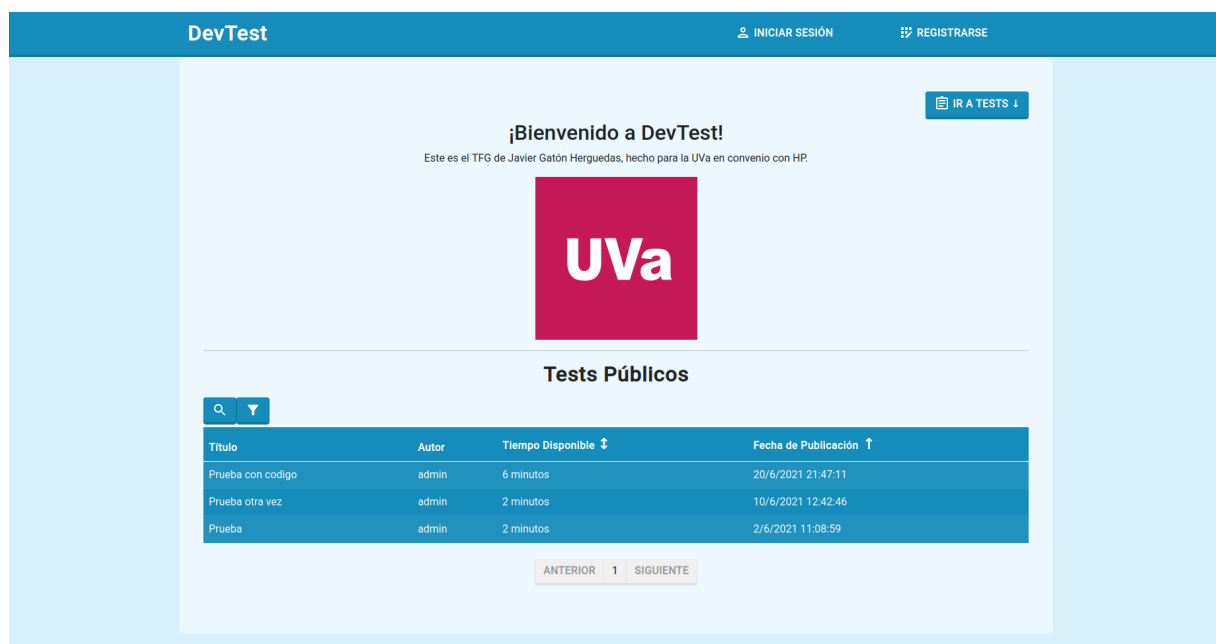


Figura 7.38: Landing page sesión no iniciada

7.4.2. Barras de navegación

Las barras de navegación permiten a los usuarios saber qué recursos y acciones tienen disponibles, lo cual significa que tienen que ser diseñadas de forma clara y concisa. Para ello se han utilizado diversidad de iconos, y se ha mantenido un diseño simple.

En la figura 7.38, previamente mencionada, se puede ver la barra de navegación de un usuario no registrado, la cual permite iniciar sesión y registrarse.

En la figura 7.39 se puede ver la barra de navegación de un usuario base, la cual permite ver los tests públicos, sus tests pendientes, sus respuestas y ver su perfil.

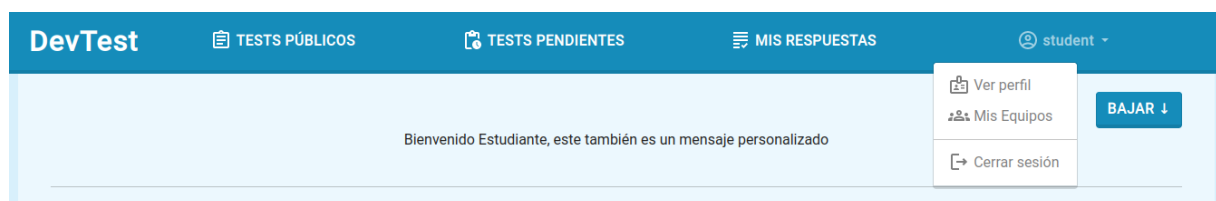


Figura 7.39: Barra de navegación de usuario base

En la figura 7.40 se puede ver la barra de navegación de un administrador, que coincide con la de un profesor, pero esta además tiene la opción de acceder al panel de administración.

7.4.3. Tablas

Uno de los elementos no triviales más comunes de la aplicación son las tablas. Sus componentes varían mucho según el tipo de recurso expuesto, pero casi todas tienen componentes comunes, como la opción de ordenar según cierto atributo, la búsqueda, y el filtrado.



Figura 7.40: Barra de navegación de administrador

En la figura 7.41 se puede ver una tabla de tests en estado contraído, que permite ordenar según el Tiempo Disponible o la Fecha de Publicación. En ese instante está ordenada según la Fecha de Publicación, poniendo primero las más recientes.

Título	Autor	Tiempo Disponible ↓	Fecha de Publicación ↑
Prueba con codigo	admin	6 minutos	20/6/2021 21:47:11
Prueba otra vez	admin	2 minutos	10/6/2021 12:42:46
Prueba	admin	2 minutos	2/6/2021 11:08:59

ANTERIOR 1 SIGUIENTE

Figura 7.41: Tabla contraída

Si en esa tabla se pulsa sobre el icono de la lupa, se expondrá la opción de búsqueda por título, y si se pulsa sobre el icono del filtro se expondrá la opción de filtro por etiquetas. La misma tabla pero con los cuadros de búsqueda y filtrado expandidos se muestra en la figura 7.42. No todas las tablas tienen filtrado por etiquetas, y algunas tienen más de una opción de búsqueda. Este diseño variará levemente según esas diferencias entre tablas.

TÍTULO

Título

AÑADIR

Etiqueta

Título	Autor	Tiempo Disponible	Fecha de Publicación
Prueba con codigo	admin	6 minutos	20/6/2021 21:47:11
Prueba otra vez	admin	2 minutos	10/6/2021 12:42:46
Prueba	admin	2 minutos	2/6/2021 11:08:59

ANTERIOR

1

SIGUIENTE

Figura 7.42: Tabla con opciones de búsqueda y filtrado expandidas

Capítulo 8

Implementación

8.1. Herramientas y Tecnologías de Desarrollo

8.1.1. Trello

Trello[36] es una herramienta en línea que sirve para gestionar proyectos y tareas de forma personal o en equipo. Cuenta con una interfaz de usuario sencilla, que permite manejar y gestionar las tareas organizándolas en diferentes tableros, creando subtareas, asignando fechas límite, etcétera.

8.1.2. Visual Studio Code

Visual Studio Code[37] es un editor de código hecho por Microsoft que tiene herramientas para acciones como la depuración de código o la refactorización, y permite a los usuarios personalizar atajos de teclados y estilos. Tiene una gran cantidad de plugins disponibles, lo cual aumenta su funcionalidad.

8.1.3. UMLet

UMLet[38] es una herramienta de software libre diseñada para la creación de diagramas UML. Destaca frente a sus alternativas privativas en que, además de ser gratuito y libre, es simple y rápido, cualidades útiles cuando la documentación y los diagramas UML son pensados como una ayuda y no como una tarea a realizar.

Existe un plugin para Visual Studio Code, herramienta explicada en la sección 8.1.2, que permite el funcionamiento de UMLet integrado en el editor de código.

8.1.4. Postman

Postman[39] es una herramienta gratuita para la ejecución de peticiones HTTP que suele utilizarse para pruebas end-to-end de una API, pues permite realizar peticiones HTTP personalizadas a cualquier dirección, modificando el método, cabeceras y cookies, además de, evidentemente, el cuerpo.

8.1.5. Typescript con Angular

Typescript[40] es un superset de *Javascript*, que tiene soporte para el tipado seguro y otras herramientas. No puede ser ejecutado directamente por los navegadores, sino que debe ser “transpilado” a *Javascript* utilizando el compilador *tsc*.

Angular[41] es un framework creado por Google orientado para el diseño de aplicaciones web que permite crear aplicaciones de una sola página eficientes y sofisticadas. Su lenguaje principal es *Typescript*. Se ha utilizado la versión 12.0.1.

8.1.6. Bootstrap

Bootstrap[42] es un conjunto de herramientas para el desarrollo de Front-End que provee de elementos HTML, CSS y Javascript simples y flexibles que permiten crear componentes e interacciones de usuario populares para el diseño de interfaces de usuario. Se ha utilizado la versión 5.0.1 para este proyecto.

8.1.7. Bootswatch

Bootswatch[43] es un conjunto de temas personalizables para Bootstrap, que permiten modificar la apariencia de un sitio web que usa bootstrap de forma muy sencilla. Se utiliza el tema “Lumen”.

8.1.8. Golang

Go, también conocido como Golang[44], es un lenguaje de programación diseñado por Google que se ha popularizado[45] debido a la facilidad de su lectura, alta escalabilidad, eficiencia similar a C/C++, y manejo de paralelismos al nivel de Java gracias a las *Goroutines*. Se ha utilizado la versión 1.15.8 en este proyecto.

8.1.9. MariaDB

MariaDB[46] es un software de gestión de base de datos, y es una bifurcación/fork de MySQL[47] que pretende mantenerse como software libre. Se ha usado la versión 15.1 para este trabajo.

8.1.10. Docker

Docker[48] es un proyecto de código abierto que sirve para automatizar el despliegue de aplicaciones dentro de contenedores de software, otorgando una capa adicional de abstracción y automatización de virtualización de aplicaciones en diversos sistemas operativos. Se ha usado la versión 20.10.7 para este trabajo.

8.1.11. NGINX

NGINX[49] es un software open source diseñado para servir aplicaciones web, funcionar como proxy inverso, gestión de caché, gestión de carga, streaming de contenido multimedia, etcétera. Además de sus capacidades como servidor HTTP, NGINX puede funcionar como servidor proxy para email (IMAP, POP3, and SMTP) y como proxy inverso y gestor de carga para servidores HTTP, TCP, y UDP.

8.1.12. Git

Git[50] es un software distribuido de gestión y control de versiones, totalmente gratuito y de código abierto. Está diseñado para ser utilizado en proyectos de cualquier tamaño de forma eficiente.

8.1.13. GitLab

GitLab[51] es una aplicación web que provee de un servicio de alojamiento de repositorios utilizando git, y además aporta una interfaz gráfica que simplifica la gestión de las diferentes ramas del repositorio.

8.1.14. Ubuntu

Ubuntu[52] es un sistema operativo de software libre. Es una distribución de Linux basada en Debian. Se ha utilizado tanto para el entorno de desarrollo, es decir, es el sistema operativo del autor, como para el entorno de pruebas de producción, ya que es el sistema operativo de la máquina virtual prestada por la escuela. En ambos casos se ha usado la versión 20.04 “Focal Fossa”.

8.1.15. Swagger 2.0

Swagger[53] es un lenguaje de descripción de interfaces diseñado para la descripción de APIs REST. Se escribe utilizando YAML. Además tiene asociado un conjunto de herramientas de software de libre, que sirven para diseñar, construir, documentar, y utilizar servicios web RESTful. Actualmente, a este estándar también se le conoce como OpenApi, donde la versión más reciente es la 3.0. Sin embargo, se ha decidido utilizar la versión 2.0, pues no se han encontrado herramientas fiables para la generación de código de servidor en Golang utilizando Swagger.

8.1.16. Go-Swagger

Go-Swagger[54] es una herramienta de software libre que provee de la implementación de Swagger 2.0 (u OpenApi 2.0) para Golang. Permite la generación de tanto servidores como clientes web en Golang basándose en unas especificaciones en Swagger 2.0. La generación automática de código se explica más detalladamente en la sección 8.2.

8.1.17. Swagger-Codegen

Swagger-Codegen[55] es un proyecto de software libre que permite la generación de plantillas de código en distintos lenguajes, basándose en unas especificaciones en OpenApi. Se ha utilizado para

la generación de parte del código de las operaciones del cliente angular. Se explica en más detalle en la sección 8.2. Concretamente se ha utilizado la versión 2.4.20. Las versiones 2.X son para OpenApi 2.0, y las 3.X para OpenApi 3.0.

8.2. Generación automática de código

La utilización de Swagger 2.0 para la descripción de la API y de los DTOs permite la generación automática de código tanto para el cliente como para el servidor, gracias a diversas herramientas.

8.2.1. Generación automática en servidor

La generación automática de código del servidor en Golang se realiza gracias a Go-Swagger, el cual se explica en la sección 8.1.16, permite generar los DTOs en el paquete “models”, tal y como se explica en la sección 7.2.1.

También genera todas las funciones que se encargan de servir y conectar las diferentes peticiones recibidas con las operaciones adecuadas de la API, permitiendo al desarrollador implementar directamente las operaciones siguiendo el esquema dado en la documentación de Swagger 2.0.

8.2.2. Generación automática en cliente

La generación automática de código para el cliente Angular, utilizando Swagger-Codegen, herramienta ya mencionada en la sección 8.1.17, genera los modelos de los DTOs y las operaciones usando la API, las cuales obtienen como respuesta y envían como parámetros, dichos DTOs.

El código se ha generado en un componente individual, que se ha publicado en npmjs bajo la dirección de <https://www.npmjs.com/package/@javgat/devtest-api>. Este componente se instala en el cliente angular, y permite utilizar las funciones y los modelos generados previamente.

Hay algunos errores en la generación del código, pues el fichero package.json generado no es correcto, ya que utiliza paquetes inexistentes para algunas versiones de angular, como el http, que a partir de la versión 5 se encuentra dentro de otros paquetes. Para solucionar esto, se creó un fichero denominado package_old.json que contenía la documentación de dependencias adecuada, y cada vez que se generaba una nueva versión se sustituían los contenidos del archivo generado por los del package_old.json, salvo el identificador de la nueva versión.

8.3. Seguimiento del Proyecto

El seguimiento del proyecto se ha realizado en Trello, creando dos tableros, uno para el FrontEnd y otro para el BackEnd, y seis listas en cada uno. Las listas son “Freezer”, “Back Burner”, “Front Burner”, “Doing”, “Peer Review” y “Done”.

- **Freezer:** Tareas que surgen durante un sprint y no se van a realizar en el mismo, o tareas que podrían realizarse pero probablemente no se hagan ya que tienen mucho coste y muy poco valor. Se trata de dejar tareas para un futuro, el cual para algunas nunca llegará.

- **Back Burner:** Parte trasera del Sprint Backlog. Lo que se va a realizar en el sprint pero con sólo una leve descripción y sin distribuir en las tareas necesarias.
- **Front Burner:** Parte delantera del Sprint Backlog. Lo que se va a realizar dentro de poco y ya está subdividido en tareas y descrito correctamente.
- **Doing:** Lo que se está haciendo.
- **Peer Review:** Lo que se ha acabado en el Sprint y se espera a una reunión para examinarse por una comprobación por pares, para asegurarse de que cumple los requisitos.
- **Done:** Lo que se ha realizado en algún Sprint y pasco la comprobación por pares.

A partir del Sprint 2, no se incluirán las subtarefas de cada historia de usuario, pues en algunos casos quedaría muy extenso, ni tampoco tareas de corrección de bugs, modificación leve de funcionalidad, o cambio no importante en la interfaz de usuario. Para ver los detalles precisos de cada tarea creada, se pueden ver los tableros con las tareas realizadas (en la lista “done”) de cada sprint. Hay dos tableros, uno para el FrontEnd, <https://trello.com/b/rThasZAy/frontend-tfg-javgat-devtest>, y otro para el BackEnd, <https://trello.com/b/pXM5LAN2/backend-tfg-javgat-devtest>.

8.3.1. Sprint 1

Sprint planning

Este sprint está caracterizado por una gran carga de tareas de aprendizaje. Las tareas de aprendizaje se denominan *Spikes*, y se añaden al *Sprint Backlog* pero no al *Product Backlog*.

Debido a que es el primer sprint, y coincidía con algún examen y con la colaboración por parte del alumno en la Olimpiada Informática de Castilla y León, la carga de trabajo de este Sprint se reduce a 20 horas.

El Sprint Backlog del Sprint 1 se encuentra en el cuadro 8.1.

ID	Nombre	Descripción	Peso
S.1.1	Escoger lenguaje de FE. Angular/React	Escoger lenguaje en el que se desarrollará el Front-end. Investigar acerca de las dos opciones principales. Angular y React. El equipo de desarrollo ya conoce Angular.	1h
S.1.2	Escoger lenguaje de BE. Node.js/Go	Escoger lenguaje en el que se desarrollará el Back-end. Investigar acerca de las dos opciones principales. Node.js y Go (Golang).	3h
S.2.1	Investigar OpenApi	Investigar y entender qué es OpenApi, leer la documentación oficial.	3h
S.2.2	Investigar Swagger con YAML	Investigar acerca de implementar la documentación Swagger/OpenApi mediante YAML	7h
S.3	Investigación de seguridad. JWT y OAuth2	Investigar acerca de la seguridad en comunicación web y por API, principalmente acerca de JWT y OAuth2. Decidir si usar alguno u otra cosa, ver problemas.	6h

Cuadro 8.1: Sprint Backlog del Sprint 1

Sprint review

S.1.1 Lenguaje de FrontEnd

Respecto al lenguaje de FrontEnd, se escoge Angular. Investigados ambos por encima se ve que tienen un potencial similar, y se escoge Angular principalmente porque el alumno tenía experiencia con él.

S.1.2 Lenguaje de BackEnd

En cuanto al lenguaje de BackEnd, se hizo una comparación descrita a continuación y se acabó escogiendo Go.

- Go es apoyado por Google: Buena documentación
- Node.js es lo más usado, lo que hará que haya más documentación en Internet.
- Node.js puede usarse con Typescript, al usar Angular en Frontend esto me permitiría reutilizar código para elementos del cliente y servidor.
- Go es más eficiente pero tiene peor gestión de errores
- Go es orientado a microservicios, promete a futuro, directamente compilable.

S.2.1 Investigar OpenApi

Investigado, documentación leída. Se trata de un proyecto colaborativo de código abierto, que trata de una especificación para interfaces API que permita ser leído y entendido por máquinas y humanos.

S.2.2 Investigar Swagger con YAML

Inicialmente se elabora una lista de Proyectos y herramientas open-source interesantes que traten con Swagger usando YAML

Proyectos/Herramientas de Swagger con YAML:

- **swagger-editor**: Editor online de especificaciones Swagger con YAML (editor.swagger.io)
- **swagger-template**: Proyecto plantilla para convertir archivos YAML en documentación Swagger
- **swagger2markup**: Convierte JSON/YAML de Swagger en documentación AsciiDoc o Markdown, que se puede completar para hacer PDF, etc. Tiene una versión para Gradle
- **Swagger Parser**: Parsea, valida y dereferencia especificaciones Swagger JSON/YAML en Node y navegadores
- **openapi-mock**: CLI para iniciar servidor mock basado en especificación Swagger JSON/YAML
- **spec-synthase**: Herramienta que une YAML pequeñitos en una especificación Swagger grande

Posteriormente se realiza una demo de Swagger usando YAML en SwaggerHub, ubicada en: https://app.swaggerhub.com/apis/JavgatDevTest/Prueba_Empleados/1.0.0-oas3

S.3 Investigación de seguridad. JWT y OAuth2

OAuth 2.0 es un protocolo. Tiene sentido para cuando hay muchos tipos de clientes, o clientes que no controlas, evitando que ese cliente se guarde las credenciales del usuario, pero si el cliente va a estar bajo el desarrollo del mismo equipo de desarrollo, no es algo necesario.

JWT es un formato de token. Tiene diferentes partes, firma, etc.

Información de seguridad de tokens:

- Usar siempre HTTPS, evitando ataques Man-in-the-Middle
- Guardar token en Cookies. Es peligroso guardarlas en almacenamiento web, ataques XSS.
- Evitar exploits CSRF/XSRF en las cookies. Técnicas existentes.

JWT es algo útil y se usará como formato de token, y tengo 2 posibilidades reales de protocolos: OAuth2 y Bearer. Bearer es suficiente para lo que se quiere hacer. Requiere menos carga de trabajo y menos despliegue final.

ID	Nombre	Resultado	Done?
S.1.1	Escoger lenguaje de FE. Angular/React	Se escoge Angular.	Y
S.1.2	Escoger lenguaje de BE. Node.js/Go	Se escoge Go	Y
S.2.1	Investigar OpenApi	Investigado	Y
S.2.2	Investigar Swagger con YAML	Demo realizada	Y
S.3	Investigación de seguridad. JWT y OAuth2	Se usará JWT	Y

Cuadro 8.2: Sprint Review del Sprint 1

8.3.2. Sprint 2

Sprint planning

A partir de esta sección se omitirá el apartado Peso, pues era el peso estimado de las historias, el cual no se solía corresponder con la realidad y se ignoró en el seguimiento.

El Sprint Backlog del Sprint 2 se encuentra en el cuadro 8.3.

ID	Nombre	Descripción	
2.1	Servicio Go Registro usuario	Servicio de Go que permita registrar un nuevo usuario mediante la llamada correspondiente a la API.	
2.2	Servicio Go Login usuario	Servicio de Go que permita recibir las credenciales y cookies de un usuario mediante la llamada correspondiente a la API, con los datos correspondientes. Diseño de ciberseguridad de la aplicación.	

Cuadro 8.3: Sprint Backlog del Sprint 2

Sprint review

A diferencia del Sprint 1, las tareas ya no son Spikes, por lo que no se expondrá una conclusión individual de cada tarea. En caso de que todo se haya realizado con éxito, se omitirá el Sprint Review del sprint. En este caso se ha realizado todo con éxito.

8.3.3. Sprint 3

Sprint planning

El Sprint Backlog del Sprint 3 se encuentra en el cuadro 8.4.

ID	Nombre	Descripción	
3.1	Signin en FrontEnd	Botones y vista de registro de usuario	
3.2	Login en FrontEnd	Botones y vista de inicio de sesión, y confirmación de inicio.	

Cuadro 8.4: Sprint Backlog del Sprint 3

8.3.4. Sprint 4

Sprint planning

El Sprint Backlog del Sprint 4 se encuentra en el cuadro 8.5.

ID	Nombre	Descripción	
4.1	Conservar sesión iniciada	Cuando se cierre la aplicación y se vuelva a abrir, o se actualice la aplicación, que la sesión del usuario se mantenga iniciada.	
4.2	Logout	Usuario puede cerrar sesión.	
4.3	Redirección si no está la sesión iniciada	Si intenta acceder a algún elemento que requiera sesión iniciada sin haberla iniciado, se redirige a la página inicial.	
4.4	Perfil de usuario	Perfil de usuario e información asociada al mismo.	
4.5	Página admin de gestión usuarios	Página para el usuario donde verá a los usuarios y podrá entrar para modificar datos y el rol.	

Cuadro 8.5: Sprint Backlog del Sprint 4

8.3.5. Sprint 5

Sprint planning

Este sprint se caracteriza por el diseño e implementación de las operaciones de la API. El Sprint Backlog del Sprint 5 se encuentra en el cuadro 8.6.

8.3.6. Sprint 6

Sprint planning

El Sprint Backlog del Sprint 6 se encuentra en el cuadro 8.7.

ID	Nombre	Descripción	
5.1	Operaciones de relogin	Creación de las operaciones que permitan recibir las cookies correctas y alargar la sesión de usuario.	
5.2	Gestión de equipos	Profesor puede crear equipos, añadir usuarios, modificar roles en equipo.	
5.3	Creación de pregunta por profesor	Profesor puede crear una pregunta.	
5.4	Búsquedas avanzadas de preguntas	Búsqueda de preguntas con filtros de etiquetas y por nombre.	
5.5	Creación de test borrador por profesor	Profesor puede crear un examen borrador.	

Cuadro 8.6: Sprint Backlog del Sprint 5

ID	Nombre	Descripción	
6.1	Re-Login automático	Cuando se acaba la sesión corta, el FrontEnd se encarga de renovar las cookies de sesión.	
6.2	Gestión de equipos FrontEnd	Profesor puede crear equipos, añadir usuarios, modificar roles en equipo.	
6.3	Creación de pregunta por profesor FrontEnd	Profesor puede crear una pregunta.	
6.4	Búsquedas avanzadas de preguntas FrontEnd	Búsqueda de preguntas con filtros de etiquetas y por nombre.	
6.5	Creación de test borrador por profesor FrontEnd	Profesor puede crear un examen borrador.	
6.6	Seguridad CSP	Configuración inicial del CSP.	

Cuadro 8.7: Sprint Backlog del Sprint 6

8.3.7. Sprint 7

Sprint planning

El Sprint Backlog del Sprint 7 se encuentra en el cuadro 8.8.

8.3.8. Sprint 8

Sprint planning

A partir de este sprint hay bastantes elementos que no se incluyen en esta memoria porque son modificaciones pequeñas de la funcionalidad, correcciones de errores o cambios leves en la interfaz. Se recomienda ver los enlaces a los tableros de trello explicados en la sección 8.3.2.

El Sprint Backlog del Sprint 8 se encuentra en el cuadro 8.9.

ID	Nombre	Descripción	
7.1	Usuario: Iniciar respuesta	Usuario puede iniciar la respuesta a un examen.	
7.2	Usuario: Visualizar y Enviar respuesta	Usuario puede visualizar la respuesta actual y enviarla como finalizada.	
7.3	Usuario: Ver Test Publicado	Usuario puede visualizar test publicado.	
7.4	Profesor: Invitar a Test Publicado	Profesor puede invitar a usuarios a la realización de un test, por equipo o individual. Recibirán correo.	
7.5	Favoritos	Añadir preguntas y exámenes a favoritos. Listas de favoritos.	
7.6	Recuperar contraseña por correo electrónico	Botón de recuperación de contraseña por correo. Se envía link con token de tiempo limitado que permitirá recuperar la contraseña.	
7.7	Búsqueda avanzada de Tests	Búsqueda avanzada de tests con filtros de etiquetas.	
7.8	Clonar Test	Clonar un test con sus preguntas.	
7.9	Clonar Pregunta	Clonar una pregunta, crea una idéntica pero de otro usuario.	

Cuadro 8.8: Sprint Backlog del Sprint 7

ID	Nombre	Descripción	
8.1	Editar ajustes de correo del sistema	Los administradores podrán modificar los ajustes de correo del sistema.	
8.2	Visualización a respuestas corregidas de test		
8.3	Corrección automática de preguntas teóricas	Las preguntas teóricas se podrán corregir automáticamente cuando el usuario entregue la respuesta	
8.4	Mensajes personalizados en notificaciones	Algunas notificaciones deben permitir un mensaje personalizado del emisor	

Cuadro 8.9: Sprint Backlog del Sprint 8

8.3.9. Sprint 9

Sprint planning

Las tareas del sprint 9 no eran historias de usuario, sino correcciones de errores, mejora del rendimiento, y, principalmente, modificaciones de la interfaz de usuario. Se pueden ver completamente en el tablero de trello.

8.3.10. Sprint 10

Sprint planning

El sprint 10 se dedico en gran parte a la modificación de interfaz de usuario, documentación y escritura de la memoria del TFG. También se dedicó en realizar el incremento 2 e iniciar el incremento 3, cuyos elementos en algunos casos sí historias de usuario.

El Sprint Backlog del Sprint 10 se encuentra en el cuadro 8.10.

ID	Nombre	Descripción	
10.1	Crear Preguntas de Código	Los profesores podrán crear preguntas de tipo código	
10.2	Responder Preguntas de Código	Los usuarios podrán responder preguntas de código con un editor de código integrado en la aplicación	
10.3	Creación de Pruebas de Ejecución	Los profesores podrán editar y crear pruebas de ejecución para preguntas de código	
10.4	Comprobación de Ejecución de Pruebas	Los usuarios y profesores podrán comprobar los resultados del código contra las pruebas de código. (De momento resultado simulado, nunca compila)	

Cuadro 8.10: Sprint Backlog del Sprint 10

8.3.11. Sprint 11

Sprint planning

El Sprint Backlog del Sprint 11 se encuentra en el cuadro 8.11.

ID	Nombre	Descripción	
11.1	Compilación y ejecución en BE	La ejecución en BE será real y no un stub	
11.2	Usuario ve resultados completos	El usuario verá con detalles lo que ha sucedido con las diferentes pruebas ejecutadas sobre su respuesta	

Cuadro 8.11: Sprint Backlog del Sprint 11

Capítulo 9

Pruebas

9.1. Pruebas unitarias

Las pruebas unitarias[56] son un tipo de pruebas de código que consisten en dividir y modularizar fragmentos y funciones lo máximo posible, y realizar pruebas de cada módulo para comprobar que funcionan correctamente. Son las pruebas que mejor permiten detectar errores tempranos, y que cuando lo encuentran, son las que provocan que la solución sea de menor coste.

Se pueden hacer tanto en BackEnd como en FrontEnd, pero se han decidido hacer únicamente en BackEnd, ya que es donde hay que asegurarse que el flujo de ejecución es el correcto, y además que la lógica del FrontEnd es muy simple.

Para realizar pruebas en Go[57], hay que asignar un nombre a cada función de prueba de tal modo que todas empiecen por “Test”. Posteriormente cuando se ejecutar dichas funciones, se podrá hacer llamando al comando “go test”.

Se han realizado tests unitarios a diferentes funciones de los paquetes explicados en la sección 7.2.1, como en jwtauth, algunas funciones del paquete dao y al paquete dbconnection. Sin embargo, no se han realizado funciones directamente sobre el paquete “handlers”. Esto habría sido buena idea, pero habría que haber separado mejor el código, creando un paquete propio para la lógica de la aplicación, y no hacerlo directamente desde “handlers”, pues las operaciones tienen parámetros y respuestas de tipos y clases concretas para satisfacer al framework de Go-Swagger.

En un futuro habría que refactorizar el diseño de la forma explicada en el párrafo anterior, mejorando así la calidad del software, orientándolo a la creación simple de tests unitarios para todos los paquetes.

9.2. Pruebas End-To-End

Las pruebas *End-To-End* permiten validar un sistema de software completamente, de principio a fin con la integración de interfaces externas. El propósito es comprobar la integración y comunicación correcta y el funcionamiento de la base de datos.

Se han distinguido dos tipos de pruebas End-To-End en este sistema. El primer tipo, consiste en pruebas contra la API REST, para las cuales se ha utilizado la herramienta Postman, de la que se habla en la sección 8.1. Se realizaban un conjunto de operaciones contra la API, y se comprobaba

que posteriormente el estado de la base de datos fuera el correcto, leyendo los datos directamente y leyéndolos mediante las operaciones de lectura de la API.

El segundo tipo se trata de pruebas End-To-End completas, para las que se ha utilizado la interfaz generada en el FrontEnd, ejecutado diversas operaciones, y comprobado que el estado se cambiaba correctamente. La diferencia con el tipo anterior es que el anterior tipo se aseguraba de que la API funcionara correctamente, y este tipo se centra en comprobar que el FrontEnd hace las llamadas correctas a la API.

9.3. Pruebas de Usabilidad

La usabilidad es definida por la norma ISO 9241-11 como “el grado en que un producto puede ser utilizado por usuarios específicos para lograr los objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso”[58].

Se han analizado los comportamientos y actuaciones de varios usuarios de prueba p *testers* frente a las acciones expuestas en el apéndice B, y analizados los diferentes errores que han tenido y el grado de completitud alcanzada en las distintas acciones.

Para ello se ha desplegado la aplicación en un servidor y una dirección web otorgada por la Escuela de Ingeniería Informática. La dirección es <https://virtual.lab.inf.uva.es:20173/>, pero debido a problemas de acceso en algunos navegadores debido a que el certificado SSL es autofirmado, se ha desplegado también en <http://virtual.lab.inf.uva.es:20172/>, aunque en esta versión no se implementan todas las funcionalidades de seguridad explicadas en la sección 7.3.1.

Las pruebas de usabilidad se han realizado sobre todo para los casos de uso de los usuarios base o estudiantes y los usuarios no registrados, pues los usuarios de tipo profesor y administrador tienen acciones más complejas, además de que las van a realizar muchas veces, lo cual permite un pequeño margen de aprendizaje. Sin embargo, los usuarios base en muchos casos usarían el sistema únicamente una o dos veces, lo cual no da ningún margen para el aprendizaje, por lo que hay que exigir una mejor usabilidad.

No se han probado todas las funcionalidades con igual frecuencia e intensidad, pues las últimas funcionalidades implementadas, relativas a las preguntas prácticas de programación, no han tenido pruebas de usabilidad debido a la falta de tiempo.

Para la evaluación subjetiva de la usabilidad se ha utilizado la escala SUS[59], mediante la cual los diferentes usuarios o *testers* han dado su opinión acerca del sistema. Por las razones explicadas previamente, las funcionalidades más evaluadas han sido las relativas a los usuarios base o estudiantes, y la mejora de la usabilidad en estas funcionalidades es bastante notable, pues inicialmente llegó a haber algún valor SUS de las funcionalidades de estudiante inferior al 60, y las últimas pruebas sitúan el valor SUS cercano al 90.

Capítulo 10

Conclusiones

Tras la finalización de los objetivos iniciales, explicados en la sección 2.1, también se consiguió realizar el objetivo 4, habiendo realizado finalmente una red social que permite la creación, de forma colaborativa, de preguntas y exámenes teóricos, y su corrección automática.

Sin embargo, tres de los cuatro objetivos no obligatorios no se lograron en tiempo. Aún así, se puede decir que el trabajo se finalizó con éxito, pues los tres objetivos necesarios y un cuarto voluntario se realizaron completamente.

La causa principal de no haber terminado todos los objetivos es que prácticamente todas las tecnologías utilizadas, salvo SQL, HTML y CSS, eran completa o prácticamente desconocidas para el equipo de desarrollo, es decir, el autor. Estas dificultades no fueron asumidas del todo desde el inicio, lo que hizo que se pretendiera tomar una mayor carga de trabajo de la que era posible. Aún así, el trabajo realizado es bastante y cumple con creces los requisitos iniciales.

10.1. Aportaciones

Es la primera vez que el autor desarrolla una aplicación completa, desde la base de datos hasta la interfaz. Esto ha permitido observar diversas dificultades y problemas que surgen durante el desarrollo, a las que se les ha dado poca importancia, pero luego han influido en gran manera sobre la calidad del código y la cantidad de funcionalidades implementadas. Estos temas se tratan en las secciones 10.1.1 y 10.1.2.

10.1.1. Diseño orientado a la realización de pruebas unitarias

Las pruebas unitarias de las funcionalidades de la capa de negocio no se pudieron realizar completamente, pues al estar parte de la funcionalidad de negocio integrada en el paquete “handlers”, en operaciones que se encargan de responder a las diferentes peticiones de la API, la realización de estas pruebas estaba dificultada, pues crear los parámetros necesarios para probar completamente todas las operaciones era muy costoso.

Si la capa de negocio se hubiera aislado completamente de la “capa de transporte” (por ponerle un nombre), las pruebas unitarias se habrían realizado de forma mucho más sencilla, lo cual habría aumentado la calidad del código, pues habría aumentado la cantidad de pruebas unitarias realizadas.

10.1.2. Diseño orientado a la reducción de trabajo del equipo de desarrollo

Durante el diseño del sistema, las decisiones se han tomado prácticamente ignorando el coste de la implementación del sistema. Es posible que para ciertas operaciones o decisiones de diseño, hubiera una alternativa que diera prácticamente las mismas funcionalidades pero que fuera menos costosa de implementar.

Un ejemplo de esto serían las operaciones de la API, las cuales son muchas, pero en algunos casos no se utilizan en el FrontEnd y simplemente están diseñadas e implementadas para que el diseño esté completo. Es bueno que el diseño esté completo, pero cuando eso sacrifica el “Time-To-Market”, y no sirve para implementar funcionalidad para el cliente, es una mala decisión de gestión de proyectos.

10.1.3. Separación correcta de las funcionalidades de la capa de negocio y de los DAOs

La mayoría de las operaciones relacionadas con la lógica de negocio se realizan en la capa de negocio, sin embargo, algunas de ellas, más simples, se realizan directamente en los objetos DAO, en la capa de persistencia. Esto se debe probablemente a la ausencia de una capa de negocio robusta, tal y como se indica en la sección 10.1.1, y permitiría entender el sistema de forma más simple y rápida de cara a futuros desarrolladores. Esta decisión es una decisión importante de diseño que no se puede dejar correr permitiendo que el resultado final no sea uniforme, es decir, que haya operaciones similares en capas diferentes.

10.1.4. Experiencia de gestión de proyecto ágil real

La realización de un proyecto de no pequeña envergadura, utilizando metodologías ágiles, permite experimentar de primera mano problemas que se producen durante el desarrollo y la gestión de proyectos ágiles. Principalmente, el no ser suficientemente ágil, pues se realizan diseños e implementaciones de características no muy valiosas, y, por falta de tiempo, se ignoran otras que tienen más valor.

Para solucionar esto, habría que haber realizado una mejor gestión ágil, ignorando los elementos menos valiosos y planificando para una realización previa los incrementos más valiosos. Esto podría haberse hecho, por ejemplo, planificando primero la compilación y corrección automática de código, y más tarde, ciertas características como búsqueda de preguntas mediante filtros, ordenación, equipos, etcétera.

10.1.5. Tecnologías modernas aprendidas

El autor ha aprendido muchas tecnologías modernas relacionadas con el desarrollo web, como Angular, Typescript, Golang, Docker, Nginx, etcétera. El nivel de manejo con ellas es bastante bajo, porque, evidentemente, con unos meses es imposible volverse un experto, pero ahora tiene un mayor control sobre dichas herramientas y lenguajes, y, en un futuro, el software generado con las mismas o similares será de mayor calidad, y lo generará más rápido.

10.2. Trabajo futuro

10.2.1. Ampliar los lenguajes soportados

Una vez se ha implementado para un único lenguaje, C++, no resultaría demasiado complicado ampliar la funcionalidad para permitir al usuario responder en diferentes lenguajes de programación, y permitir a los profesores elegir los lenguajes de programación que se podrán usar en sus preguntas.

10.2.2. Internacionalización

Los elementos de texto que no provienen de la base de datos, están directamente escritos en los archivos html. Para poder modificar de forma coherente los diferentes textos de la capa de persistencia, se deberían almacenar en archivos independientes, lo cual también permitiría traducir los elementos de la aplicación de forma sencilla. Este proceso se llama internacionalización o “i18n”.

10.2.3. Roles personalizables en capa de presentación

Tanto en la base de datos como en la API, se permite crear TipoRoles y modificar los permisos de los diferentes TipoRoles existentes, los cuales tienen un rol base que coincide con los actores analizados en la sección 6.3. Esto permitiría, por ejemplo, crear el TipoRol “moderador”, que fuera como un administrador, pero con menos permisos.

Sin embargo, estas características no se han implementado en la capa de presentación, pues habría que modificar muchas vistas y elementos de la lógica de la misma capa, para mostrar a los usuarios los diferentes elementos disponibles según su tipo de rol, y no según su rol base, como está ahora mismo.

10.2.4. Súper grupo “Organización” y actor “Director”

Se puede realizar una modificación del modelo de negocio para dar más valor al trabajo, en la que se permitiría crear súper equipos llamados “organizaciones”. Los equipos pasarían a ser internos de las organizaciones, los usuarios podrían tener un rol distinto en cada organización en la que estén, y habría un nuevo actor y nuevo rol denominado “director”, que serían como los actuales administradores pero a nivel de organización. Esto permitiría que el producto se alojara en un único servidor y que diera servicio a múltiples organizaciones sin tener que instalar ni alojar el sistema.

10.2.5. Mejorar calidad de código

Tal y como se explica en las secciones 10.1.1 y 10.1.3, hay ciertos elementos del proyecto que se podrían haber realizado de otras formas para generar software de mejor calidad. Estas modificaciones del código consistirían en modificar los paquetes “handlers” y “dao” y crear un paquete independiente en la capa de negocio que contenga toda la funcionalidad que haya en esas otras capas que no esté relacionada con el transporte de datos en la primera ni con la persistencia en la segunda.

Estas refactorizaciones permitirían que sobre este nuevo paquete se podrían realizar pruebas unitarias con menor coste, y permitir las modificaciones y ser mantenido de forma más sencilla, lo cual

aumentaría la calidad del software.

10.2.6. Mejorar la interfaz y la experiencia de usuario

Aunque la interfaz de usuario se haya diseñado teniendo en cuenta opiniones de posibles usuarios, de otros desarrolladores y de expertos, hay bastantes elementos que se podrían haber realizado mejor. Debido a los conocimientos limitados del autor al respecto de las herramientas utilizadas para la creación de la interfaz de usuario, y debido también a la limitación temporal que no le ha permitido obtener los conocimientos necesarios sobre las mismas, no se han podido implementar una interfaz y experiencia de usuario óptimas.

Apéndices

Apéndice A

Administración e Instalación

La instalación de la aplicación se explicará para un entorno Linux, poniendo de ejemplo Ubuntu 20.04.

Lo primero de todo es clonar el repositorio de git, que se encuentra en <https://gitlab.com/HP-SCDS/Observatorio/2020-2021/uva-devtest>.

```
git clone https://gitlab.com/HP-SCDS/Observatorio/2020-2021/uva-devtest .
```

Una vez descargado, se puede dividir la instalación en tres partes: Base de datos, BackEnd y FrontEnd.

A.1. Base de Datos

El software de gestión de bases de datos utilizado ha sido MariaDB, en la versión 15.1. Se recomienda utilizar esa versión o una posterior para el funcionamiento correcto de la aplicación.

Una vez instalada, se deberá ejecutar el fichero de inicialización de la base de datos, en **/Backend/init.sql**.

Finalmente, habrá que crear un directorio llamado *config* dentro de */Backend*, que contendrá un fichero *dbinfo.json* con la información necesaria para conectarse a la base de datos.

```
cd Backend
mkdir config
cd config
vi dbinfo.json
```

Dentro de ese fichero habrá un json con la información de la base de datos, siguiendo la siguiente estructura, poniendo de ejemplo una base de datos alojada en localhost con el puerto 3306, y con el nombre de usuario “admin” y contraseña “admin”:

```
{
  "Username" : "admin",
  "Pass" : "admin",
  "Host" : "localhost",
  "Port" : "3306",
  "Name" : "uva_devtest"
}
```

A.2. BackEnd

A.2.1. Ejecución de pruebas sobre las respuestas

Para la ejecución de pruebas sobre las respuestas enviadas por los usuarios se ha utilizado Docker, concretamente la versión 20.10.7.

Una vez se finalice la instalación de Docker en el sistema, se recomienda añadir al usuario que ejecute los procesos del servidor al grupo de usuarios de Docker[60], para así evitar tener que ejecutar los procesos con permisos de superusuario.

A continuación hay que construir la imagen de Docker encargada de ejecutar los procesos, para eso habrá que entrar en el directorio **/Backend/docker/** y ejecutar el script **build_container.sh**.

Una vez hecho esto ya se podrá ejecutar el servidor. Desde el servidor se llamará a los diferentes scripts que hay en el directorio principalmente mencionado, y los directorios que se compartirán con el contenedor de Docker estarán en el directorio **/tmp**, dentro de un subdirectorio llamado **pruebas** que se creará automáticamente, por lo que hay que asegurarse de que el usuario tiene los permisos necesarios sobre el directorio **/tmp**.

A.2.2. Servidor

El lenguaje de programación utilizado en el servidor del BackEnd es Go, o Golang, y la versión utilizada es la 1.15.8. Se recomienda utilizar esa versión o una posterior compatible para compilar el software del BackEnd.

Una vez instalado go, se necesitará compilar el código fuente. Para ello basta con ejecutar el script **/Backend/compile.sh**

```
cd Backend
mkdir bin
./compile.sh
```

Habrà que añadir unos archivos extra al directorio **/Backend/config**. Dos relacionados a la conexión HTTPS, el certificado TLS, que se denominará "cert.pem" y la clave (key) "key.pem". Un tercer archivo estará relacionado con la configuración del correo electrónico del sistema, que contendrá la dirección de correo, la contraseña, el servidor de correo, el puerto utilizado, y la dirección de la página web que se enlazarà en los correos.

```
{
"from": "UVaDevTest@gmail.com",
"password": "password",
"serverhost": "smtp.gmail.com",
"serverport": "587",
"frontendurl": "https://localhost:4200"
}
```

Finalmente, para activar el servidor habrá que ejecutar el fichero **/Backend/serve.sh**, el cual se puede configurar para modificar los puertos donde se ejecutarán el servicio HTTP y HTTPS. Para eliminar el servicio HTTP y quedarse solo con HTTPS, habrá que modificar la línea correspondiente a HTTP

en el fichero **/swagger.yml**, ejecutar el fichero **/BackEnd/generate.sh**, y modificar el fichero **/BackEnd/serve.sh**, eliminando la opción “**–port:**”.

A.3. FrontEnd

Para ejecutar el FrontEnd se necesita la aplicación compilada, la cual sirve la misma en cualquier sistema operativo, por lo que la compilación no es necesario que se haga en la misma maquina en la que se despliega el sistema. Para compilar se necesita angular-cli (ng) versión 12.0.1.

Hay dos tipos de compilación posible, modo producción o modo inseguro, el cual es la aplicación utilizando HTTP en vez de HTTPS, pues durante el desarrollo del proyecto no había un certificado SSL real, por lo que se usaba uno autofirmado, el cual daba problemas en algunos navegadores.

Antes de compilarlo, hay que modificar los archivos de entorno y el index. En los de entorno existe un campo denominado **API_BASE_PATH** que contiene la dirección de la API que se va a llamar. Se trata de los archivos en el directorio **/FrontEnd/devtest-client/src/environments/**, concretamente **environment.prod.ts** y **environment.unsecure.ts**.

Los archivos de “index” se encuentran en el directorio **/FrontEnd/devtest-client/src/**, concretamente **index.prod.html** y **index.unsecure.html**. En ambos hay que modificar la etiqueta relativa al Content-Security-Policy, cambiando la dirección de despliegue del BackEnd por defecto a la que se ha escrito en los archivos de entorno.

Una vez hecho esto, se ha de compilar, siguiendo la configuración que se necesite, o ambas, con el comando **ng build**. Por ejemplo se pone el modo producción:

```
ng build --configuration production
```

Se generarán los archivos necesarios para el despliegue en un directorio dentro de **/FrontEnd/devtest-client/dist/**.

Es importante mencionar que, tal y como se explica en la sección 7.3.1, el FrontEnd y el BackEnd tienen que estar desplegados bajo el mismo nombre de dominio.

Apéndice B

Manual de Usuario

A continuación se explicarán las maneras de proceder de las principales acciones disponibles para los usuarios. Todas las acciones se iniciarán asumiendo que el usuario está en la *landing page* correspondiente.

B.1. Acciones de usuario sin sesión iniciada

B.1.1. Iniciar sesión

1. Hacer click en **Barra de Navegación >Iniciar Sesión**
2. Rellenar datos de nombre de usuario y contraseña
3. Hacer click en el botón **Acceder**

B.1.2. Registrarse

1. Hacer click en **Barra de Navegación >Registrarse**
2. Introducir datos de nombre de usuario, email, contraseña y confirmar contraseña
3. Hacer click en el botón **Registrarse**

B.1.3. Recuperar contraseña

1. Hacer click en **Barra de Navegación >Iniciar Sesión**
2. Hacer click en el enlace **He olvidado mi contraseña**
3. Introducir nombre de usuario o correo electrónico
4. Hacer click en el botón **Enviar correo de recuperación**
5. Buscar el correo recibido¹
6. Hacer click en el enlace de recuperación
7. Introducir nueva contraseña
8. Hacer click en botón de **Actualizar contraseña**

¹Los envíos de correos electrónicos no funcionan en el servidor de producción de prueba de virtual.uva debido a las limitaciones con los puertos y limitaciones de conocimientos del autor para configurarlo correctamente.

B.2. Acciones de usuario base

B.2.1. Ver exámenes pendientes

1. Hacer click en **Barra de Navegación >Tests Pendientes**

B.2.2. Ver exámenes públicos

1. Hacer click en **Barra de Navegación >Tests Públicos**

B.2.3. Responder examen

1. Hacer el procedimiento para ver exámenes pendientes o exámenes públicos
2. Hacer click en un examen
3. Hacer click en el botón **Responder**
4. Hacer click en el botón **Iniciar Respuesta** del modal que aparece
5. Hacer click en cualquier pregunta de las que aparecen
6. Introducir la respuesta de texto si es teórica, seleccionar la correcta si es de elección múltiple o escribir un programa si es pregunta práctica de programación.
7. Pulsar el botón **Guardar Respuestas**
8. Pulsar el botón **Volver a la lista de respuestas, Pregunta Anterior, Pregunta Siguiente o Revisar y Entregar**, según la acción que se quiera tomar de entre las disponibles
9. Cuando se terminen de responder preguntas y se vuelva a la lista de preguntas, pulsar el botón **Enviar todo y terminar**
10. Pulsar el botón **Enviar** del modal que aparece

B.3. Acciones de profesor

B.3.1. Crear Pregunta

1. Hacer click en **Barra de Navegación >Preguntas >Crear Pregunta**
2. Introducir los datos básicos de la pregunta
3. Hacer click en el botón **Crear**
4. En caso de tener marcada la pregunta con corrección automática, o de ser pregunta de elección múltiple, hacer click en **Editar Respuesta Esperada**
5. Introducir la respuesta esperada si es de texto, las pruebas a ejecutar si es práctica de programación, o añadir las opciones si es de elección múltiple, marcando las correctas.

B.3.2. Crear Examen

1. Hacer click en **Barra de Navegación >Tests Borradores >Crear Test Borrador**
2. Introducir los datos básicos del Test
3. Pulsar el botón **Crear**
4. Pulsar el botón **Añadir Pregunta**
5. Hacer click en la pregunta que se quiera elegir. (En cualquier parte de la pregunta excepto en el botón .Abrir")

B.3.3. Publicar Examen

El procedimiento de publicar un examen se explicará desde un Test Borrador sobre el que se tenga derechos de administración

1. Hacer click en el botón **Publicar**
2. Introducir los parámetros de publicación necesarios en el modal
3. Hacer click en el botón **Publicar** del modal

B.4. Acciones de administrador

B.4.1. Modificar mensajes de *landing page*

1. Hacer click en **Barra de Navegación >Administración**
2. Hacer click en el botón **Vistas**
3. Pulsar en los mensajes que se quieran modificar y modificarlos como cuadros de texto
4. Hacer click en el botón **Guardar**

Apéndice C

Manual de Desarrollo

Para modificar la aplicación es necesario leer los capítulos 6 y 7, para así poder entender el funcionamiento de la misma.

C.1. Modificar funcionalidad sin modificar la API

Para modificar los efectos de una llamada a la API, simplemente hay que editar las funciones necesarias del BackEnd y volverlo a compilar, tal y como se indica en el apéndice A.

En caso de que sea una modificación de interfaz o experiencia de usuario, añadir una vista o modificar el flujo de la aplicación, simplemente hay que modificar los componentes de Angular necesarios y volverlo a compilar tal y como se explica en el apéndice previamente mencionado.

C.2. Modificación de API

Sin embargo, si se quiere añadir una llamada a la API, o añadir parámetros a una existente, habrá que modificar las especificaciones de Swagger 2.0. Una vez modificado, habrá que implementar la modificación en BackEnd y adaptar el componente “devtest-api” para el FrontEnd.

C.2.1. BackEnd

En BackEnd hay ejecutar el archivo **/Backend/generate.sh**.

```
cd Backend
./generate.sh
```

Una vez hecho esto, hay que añadir la llamada a la función en el archivo **/Backend/restapi/configure_dev.go**, declarándola e implementándola en el paquete “handlers”.

C.2.2. FrontEnd

Después, en FrontEnd hay que ejecutar primero el archivo **/FrontEnd/generate_api.sh**, lo cual actualiza el directorio **/FrontEnd/devtest-api**.

```
cd FrontEnd
./generate_api.sh
```

En ese mismo directorio, habrá un fichero **package.json** y otro **package_old.json**. Hay que sustituir la versión del segundo por la del primero, y después sustituir todo el contenido del primero por el contenido del segundo. Esto es porque la generación automática no funciona correctamente en lo relativo a las dependencias. También, en caso de que no se tenga permiso para modificar el repositorio original de npmjs, se puede cambiar el nombre del paquete para poder publicarlo.

Una vez modificado, se podrá publicar el componente ejecutando el archivo **publish_api.sh** del directorio **/FrontEnd/**.

```
./generate_api.sh
```

Cuando la publicación se realice con éxito, habrá que modificar el archivo **package.json** del directorio **/FrontEnd/devtest-client/**, actualizando la versión utilizada del componente de la api, y posteriormente ejecutar “npm install”.

```
cd devtest-client
npm install
```

Bibliografía

- [1] HP SCDS. *Observatorio HP*. Consultado el 24 de Junio de 2021. 2020. url: <https://hpscds.com/observatorio-hp/>.
- [2] HP SCDS. *Observatorio HP: DevTest*. Consultado el 24 de Junio de 2021. 2020. url: <https://hpscds.com/observatorio-hp/#DevTest>.
- [3] HackerRank. *HackerRank | Developer Skills Platform*. Consultado el 24 de Junio de 2021. url: <https://www.hackerrank.com/products/developer-skills-platform/>.
- [4] CodinGame. *CodinGame | Screening*. Consultado el 24 de Junio de 2021. url: <https://www.codingame.com/work/offers/screening/>.
- [5] Shahid Nehal. *Quora | What can I expect in Amazon's SDE online assessment?* Consultado el 24 de Junio de 2021. url: <https://www.quora.com/What-can-I-expect-in-Amazons-SDE-online-assessment-Can-someone-who-has-taken-the-assessment-please-give-me-examples-of-the-types-of-questions-asked>.
- [6] Scrum.org. *What is scrum?* Consultado el 24 de Junio de 2021. url: <https://www.scrum.org/resources/what-is-scrum/>.
- [7] RAE. *RAE | Definición de programa*. Consultado el 24 de Junio de 2021. url: <https://dle.rae.es/programa>.
- [8] Wikipedia. *Wikipedia | Machine code*. Consultado el 24 de Junio de 2021. url: https://en.wikipedia.org/wiki/Machine_code.
- [9] Tom Anderson. *stackoverflow | How do assembly languages depend on operating systems?* Consultado el 24 de Junio de 2021. url: <https://stackoverflow.com/questions/6859348/how-do-assembly-languages-depend-on-operating-systems>.
- [10] Shubham Singh. *GeeksforGeeks | Difference between Compiled and Interpreted Language*. Consultado el 24 de Junio de 2021. url: <https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/>.
- [11] Jason C. McDonald. *What Is An "Interpreted" Language?* Consultado el 24 de Junio de 2021. url: <https://dev.to/codemouse92/what-is-an-interpreted-language-3fef>.
- [12] Aalborg University Kurt Nørmark. *Overview of the four main programming paradigms*. Consultado el 24 de Junio de 2021. url: https://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html.
- [13] Vishal Verma. *GeeksforGeeks | Functional Programming Paradigm*. Consultado el 24 de Junio de 2021. url: <https://www.geeksforgeeks.org/functional-programming-paradigm/>.
- [14] J. Nicholas Hoover. *InformationWeek | NIST To Develop Cloud Roadmap*. Consultado el 24 de Junio de 2021. url: <https://www.informationweek.com/cloud/nist-to-develop-cloud-roadmap/d/d-id/1093958?>.
- [15] Michael Marcotty James L. Elshoff. «Improving computer program readability to aid modification». En: *Communications of the ACM* (1982). Consultado el 24 de Junio de 2021. doi: 10.1145/358589.358596.

- [16] Wisdom Geek. *Wisdom Geek | Programming 101: Tips to become a good programmer*. Consultado el 24 de Junio de 2021. url: <https://www.wisdomgeek.com/development/programming/tips-become-good-programmer/>.
- [17] Cem Kaner. *Exploratory testing*. Consultado el 24 de Junio de 2021. url: <http://www.kaner.com/pdfs/ETatQAI.pdf>.
- [18] IEEE. «IEEE Standard Glossary of Software Engineering Terminology». En: (1990). Consultado el 24 de Junio de 2021. doi: 10.1109/IEEESTD.1990.101064.
- [19] HackerRank. *HackerRank | About us*. Consultado el 24 de Junio de 2021. url: <https://www.hackerrank.com/about-us/>.
- [20] LeetCode. *LeetCode*. Consultado el 24 de Junio de 2021. url: <https://leetcode.com/>.
- [21] CodeChef. *CodeChef | About us*. Consultado el 24 de Junio de 2021. url: https://www.codechef.com/aboutus/?itm_medium=navmenu&itm_campaign=aboutus.
- [22] CodeChef. *CodeChef | Host your contest*. Consultado el 24 de Junio de 2021. url: <https://www.codechef.com/hostyourcontest>.
- [23] CMS. *CMS | Contest Management System*. Consultado el 24 de Junio de 2021. url: <https://cms-dev.github.io>.
- [24] Gusztáv Jánvári. *Imprestige | Is Login a Use Case? Use Cases vs Functional Decomposition*. Consultado el 24 de Junio de 2021. url: <https://imprestige.biz/is-login-a-use-case-use-cases-vs-functional-decomposition/>.
- [25] OMG. *OMG OML 2.4 Specification*. Consultado el 24 de Junio de 2021. 2014. url: <https://www.omg.org/spec/OCL/2.4/PDF>.
- [26] Brian Troncone. *BehaviorSubject*. Consultado el 24 de Junio de 2021. url: <https://www.learnrxjs.io/learn-rxjs/subjects/behaviorsubject>.
- [27] Oscar Blancarte. *Data Transfer Object (DTO) – Patrón de diseño*. Consultado el 24 de Junio de 2021. url: <https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/>.
- [28] Auth0. *Get Started with JSON Web Tokens*. Consultado el 24 de Junio de 2021. url: <https://auth0.com/learn/json-web-tokens/>.
- [29] Aaron Parecki. *OAuth 2.0*. Consultado el 24 de Junio de 2021. url: <https://oauth.net/2/>.
- [30] OWASP. *Who is the OWASP Foundation?* Consultado el 24 de Junio de 2021. url: <https://owasp.org/>.
- [31] OWASP. *HTML5 Security*. Consultado el 24 de Junio de 2021. url: https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#local-storage.
- [32] Luke Valenta Gabbi Fisher. *Monsters in the Middleboxes: Introducing Two New Tools for Detecting HTTPS Interception*. Consultado el 24 de Junio de 2021. url: <https://blog.cloudflare.com/monsters-in-the-middleboxes/>.
- [33] Chris Shiflett. *Cross-Site Request Forgeries*. Consultado el 24 de Junio de 2021. 2004. url: <https://shiflett.org/articles/cross-site-request-forgeries>.
- [34] Mozilla y contribuyentes individuales. *SameSite cookies*. Consultado el 24 de Junio de 2021. url: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie/SameSite>.
- [35] Mozilla y contribuyentes individuales. *Content Security Policy*. Consultado el 24 de Junio de 2021. url: <https://developer.mozilla.org/es/docs/Web/HTTP/CSP>.
- [36] Trello. *Acerca de Trello*. Consultado el 24 de Junio de 2021. url: <https://trello.com/es/about>.

- [37] Microsoft. *Visual Studio Code*. Consultado el 24 de Junio de 2021. url: <https://code.visualstudio.com/>.
- [38] UMLet. *UMLet*. Consultado el 24 de Junio de 2021. url: <https://www.umlet.com/>.
- [39] Postman Inc. *Postman*. Consultado el 24 de Junio de 2021. url: <https://www.postman.com/>.
- [40] Google. *TypeScript configuration*. Consultado el 24 de Junio de 2021. url: <https://angular.io/guide/typescript-configuration>.
- [41] Google. *Introduction to the Angular Docs*. Consultado el 24 de Junio de 2021. url: <https://angular.io/docs>.
- [42] Bootstrap. *Bootstrap*. Consultado el 24 de Junio de 2021. url: <https://getbootstrap.com/>.
- [43] Thomas Park. *Bootswatch*. Consultado el 24 de Junio de 2021. url: <https://bootswatch.com/>.
- [44] Golang. *Golang*. Consultado el 24 de Junio de 2021. url: <https://golang.org/>.
- [45] Pavan Belagatti. *Why Golang is so Popular Among Developers?* Consultado el 24 de Junio de 2021. url: <https://www.geeksforgeeks.org/why-golang-is-so-popular-among-developers/>.
- [46] MariaDB Foundation. *About MariaDB*. Consultado el 24 de Junio de 2021. url: <https://mariadb.org/about/>.
- [47] Wikipedia. *MariaDB*. Consultado el 24 de Junio de 2021. url: <https://en.wikipedia.org/wiki/MariaDB>.
- [48] Docker. *Docker*. Consultado el 24 de Junio de 2021. url: <https://www.docker.com/>.
- [49] F5 Inc. *What is NGINX?* Consultado el 24 de Junio de 2021. url: <https://www.nginx.com/resources/glossary/nginx/>.
- [50] Git. *Git*. Consultado el 24 de Junio de 2021. url: <https://git-scm.com/>.
- [51] GitLab. *What is GitLab?* Consultado el 24 de Junio de 2021. url: <https://about.gitlab.com/what-is-gitlab/>.
- [52] Canonical. *What is Ubuntu?* Consultado el 24 de Junio de 2021. url: <https://help.ubuntu.com/lts/installation-guide/s390x/ch01s01.html>.
- [53] Wikipedia. *Swagger*. Consultado el 24 de Junio de 2021. url: [https://en.wikipedia.org/wiki/Swagger_\(software\)](https://en.wikipedia.org/wiki/Swagger_(software)).
- [54] Go Swagger. *Go Swagger*. Consultado el 24 de Junio de 2021. url: <https://goswagger.io/>.
- [55] swagger-api. *swagger-codegen*. Consultado el 24 de Junio de 2021. url: <https://github.com/swagger-api/swagger-codegen>.
- [56] Wikipedia. *Unit testing*. Consultado el 24 de Junio de 2021. url: https://en.wikipedia.org/wiki/Unit_testing.
- [57] Mark McGranaghan. *Go by Example: Testing*. Consultado el 24 de Junio de 2021. url: <https://gobyexample.com/testing>.
- [58] ISO. *ISO 9241-11:2018 Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. Consultado el 24 de Junio de 2021. url: <https://www.iso.org/standard/63500.html>.
- [59] Wikipedia. *Wikipedia | System usability scale*. Consultado el 24 de Junio de 2021. url: https://en.wikipedia.org/wiki/System_usability_scale.
- [60] ConfigServerFirewall. *Add User To Docker Group In Ubuntu Linux*. Consultado el 24 de Junio de 2021. url: <https://www.configserverfirewall.com/ubuntu-linux/add-user-to-docker-group-ubuntu/>.

