UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una plataforma web enfocada en la enseñanza de diseño de software

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: García Sastre, Jesús

Tutor/a: Pelechano Ferragud, Vicente

CURSO ACADÉMICO: 2021/2022

Resumen

En este Trabajo de Fin de Grado se ha desarrollado una plataforma web expandible para mejorar y extender el uso de patrones de diseño entre los alumnos de grado de Universidad Politécnica de Valencia.

El portal web que se ha implementado permite al usuario aprender de forma autodidacta mediante lecciones, ejercicios, documentos y proyectos de ejemplo. Se ha elegido las tecnologías web para facilitar su uso y evitar problemas de compatibilidades, de esta forma se ha podido añadir una herramienta de gestión de usuarios y contenido de patrones.

Este documento describe el recorrido que ha tenido el proceso de planificación, diseño e implementación de la aplicación. También detalla el despliegue de la aplicación para su uso de pruebas con usuarios finales.

Palabras clave: web, aplicación, ejercicio, autodidacta, patrón, usuario, alumnos.

Abstract

In this Final Degree Project, an expandable web platform has been developed to improve and extend the use of design patterns among undergraduate students at the Polytechnic University of Valencia.

The web portal that has been implemented allows the user to learn in a self-taught way through lessons, exercises, documents and example projects. Web technologies have been chosen to facilitate its use and avoid compatibility problems, so that a user management tool and pattern content could be added.

This document describes the journey through the planning, design and implementation process of the application. It also details the deployment of the application for use in end-user testing.

Keywords: web, application, exercises, self-taught, pattern, user, students.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

Cada día se necesita más y más ingenieros informáticos profesionales, que estén bien cualificados para desarrollar correctas soluciones a un gran abanico de problemas. Los alumnos del grado de Ingeniería del Software cursan la asignatura “Diseño del Software”, donde se imparten los conocimientos necesarios para diseñar buenas soluciones mediante patrones de programación.

Asimismo, los patrones de diseño son muy importantes en los desarrollos, porque ayudan a tener un código muy legible para personas ajenas al proyecto, permiten implementar funcionales muy complejas de forma muy sencilla, y ahorran una gran cantidad de tiempo de desarrollo.

La importancia de tener asimilados los patrones de diseño es muy grande para los alumnos, porque en un futuro se van a unir a un mundo laboral globalizado. Por este motivo, el caso de estudio de este Trabajo de Fin de Grado ayudará a la formación de mejores profesionales del sector, profundizando en el conocimiento de los patrones.

Se adjunta un glosario que contiene terminología específica relativa a este campo de la informática al final de la memoria.

1.1 Motivación

Durante el último tramo de mi estancia en el grado, he tenido la oportunidad de encontrar trabajo. Por esa razón, he experimentado lo importante que es tener un buen conocimiento de como programar y de qué forma diseñar buenas soluciones. Quiero ayudar a mis compañeros a tener más y mejores oportunidades, a consecuencia de formarse mejor en el campo de la implementación y del diseño de soluciones. Por falta de un trabajo de fin de grado de oferta pública que cumpla mis motivaciones, le propuse a mi tutor el actual proyecto del documento.

1.2 Objetivos

El objetivo principal es implementar una plataforma web para que alumnos puedan aprender y realizar ejercicios sobre patrones de diseño y que profesores puedan crear lecciones y ejercicios sobre patrones. Como subobjetivos se encuentran:

- Disponibilidad de ejercicios tipo test autoevaluativos para alumnos.
- Exponer una biblioteca de lecciones sobre patrones de diseño.

- Permitir la subida y descarga de proyectos de ejemplo y documentos en las lecciones.
- Tener un sistema de gestión de usuario y grupos para profesores.
- Mostrar un sistema de estadísticas de cada ejercicio por grupos.
- Ofrecer una interfaz común para crear lecciones y ejercicios en el portal web para profesores.

1.3 Impacto Esperado

Se identifican dos tipos de usuarios finales en la aplicación web: alumno y profesor. El impacto esperado para alumno es una mejora en su capacidad analítica sobre problemas y soluciones, en consecuencia, sus notas se pueden ver incrementas. Asimismo, tener una aplicación web da libertad al usuario de aprender en cualquier lugar o momento que desee.

Por otra parte, el profesor puede mantener una gran biblioteca de lecciones y ejercicios sobre patrones para los diferentes grupos que tenga asignado, y estadísticas de como esos grupos van asimilando los conocimientos. Por esta razón el impacto del profesor es un mejor control y seguimiento de sus alumnos.

1.4 Metodología

En este apartado se detalla la metodología empleada para el desarrollo del proyecto. **Scrum** ha sido el método elegido, porque ofrece un desarrollo ágil de software que se adapta muy bien a cambios durante el proceso de implementación. La Ilustración 1 expone la estructura del proceso, el cual es iterativo.

La metodología se basa en la definición de un marco de trabajo que detalla un conjunto de eventos, roles y prácticas. Los roles más característicos son el *Scrum master* que controla la gestión de cambios dentro del proyecto y el correcto uso del proceso del software; y el *Product Owner* que representa a las personas interesadas del proyecto o *stakeholders* externos e internos. En este proyecto el alumno se encargará de interpretar ambos roles.

Ilustración 1 - Esquema de SCRUM

El proceso del software empieza por una fase de análisis, donde se crean todos los requisitos del producto a desarrollar con una breve descripción. La lista final de requisitos tiene como nombre *backlog* del producto. El *backlog* se confirma con el ***product owner*** antes de pasar a la segunda fase.

La segunda fase, se basa en el concepto *sprint*, el cual es un periodo de tiempo de dos a cuatro semanas de duración para el desarrollo de una versión del producto estable. En cada *sprint* se seleccionan un número de requisitos a implementar negociando con el *product owner*.

Scrum no tiene un plan específico para los *sprints*, pero sí consejos. En el proyecto se ha definido el siguiente proceso que se realizará en cada iteración: una fase para detallar más su descripción y diseñar interfaces o prototipos, luego implementar el requisito y por último comprobar que la tarea realizada es conforme a la descripción del *product owner*.

Por concluir, se detallan las ventajas y desventajas que tiene esta metodología ágil iterativa:

+ El proceso se adapta muy bien a los cambios, porque el cliente está involucrado en el desarrollo de la aplicación, manteniendo reuniones al comienzo de los *sprints*, semanales o incluso diarias.
+ Los *sprints* tienen como finalidad ofrecer al acabar una versión del producto con las principales características que considere importantes el *producto owner*. Reduciendo así el tiempo que tarda una aplicación en llegar al usuario final.

+ Se obtiene un mejor control de la calidad del software debido a que el proceso es iterativo y su objetivo es obtener la mejor versión funcional del producto.
+ Se reducen los riesgos de que la aplicación no satisfaga a los usuarios finales por la carencia de funcionalidades importantes del producto, porque los *sprints* realizan primero las tareas categorizadas como vitales para los usuarios por el *product owner*.

– La metodología no funciona bien con grupos muy grandes, porque se invertiría mucho tiempo en la organización y control de los equipos.
– La inexperiencia del análisis del tiempo requerido para la implementación de las tareas puede provocar una mala definición del alcance de los *sprints*, y como consecuencia crear retrasos en las entregas del producto.

1.5 Estructura

A continuación, se describe el esquema que se va a seguir en esta memoria, de forma que brevemente se explique que se va a tratar en cada capítulo del documento:

- **Capítulo 2:** Estado del arte. Se expone el actual estado de la industria de herramientas digitales educativas, comparándolas con las necesidades reales de los alumnos de grado.
- **Capítulo 3:** Análisis del problema. Se describe parte de la fase de análisis, el proceso de la obtención y refinamiento de la solución a implementar.
- **Capítulo 4:** Diseño de la solución. Se expone la segunda parte de la fase de diseño, donde se ha creado una arquitectura y unos esquemas para el producto.
- **Capítulo 5:** Desarrollo de la solución. Explica la fase de desarrollo y la estructura del proyecto implementado.
- **Capítulo 6:** Implantación. Se describe el último *sprint* en el que se hace el despliegue de la aplicación.
- **Capítulo 7:** Pruebas. Se detalla el proceso que se ha llevado a cabo para poder comprobar que la aplicación está libre de errores.
- **Capítulo 8:** Conclusiones. Se exponen los resultados obtenidos por el ejercicio brevemente y una valoración personal.
- **Capítulo 9:** Trabajos futuros. Se describe una lista de características que no han podido ser implementadas por falta de tiempo o que estaban fuera del objetivo del ejercicio.

- **Capítulo 10:** Referencias. Lista de referencias bibliográficas para respaldar fuentes e información de la memoria.
- **Glosario**. Lista de conceptos técnicos definidos que aparecen en la memoria.

2 Estado del arte

Las tecnologías digitales han demostrado ser una gran herramienta para educar a las personas, sin importar su ubicación o lenguaje [1]. Es por esta razón que el mercado móvil cuenta con numerosas aplicaciones de enseñanza de diferentes campos del conocimiento: salud, lenguajes, programación, matemáticas…

Los programas enfocados en expandir el conocimiento relativo a la programación de sus usuarios nunca profundizan más allá de la sintaxis y paradigma de programación del lenguaje que se está estudiando en el curso. Las más conocidas que se encuentran tanto en móvil como con el navegador son: SoloLearn, Grasshopper, Udemy.

Ilustración 2 - Aplicaciones educativas

Por la falta de profundidad en estos servicios, quedan eliminados como opciones válidas para formar profesionales adecuados. Por esta razón, muchas instituciones están desarrollando aplicaciones propias para poder ayudar a sus alumnos a formarse mejor, como puede ser la aplicación web PoliformaT [2].

Este trabajo es una solución a esta carencia de profundidad en las actuales aplicaciones del mercado, y una herramienta para profesores y alumnos para prepararse y asimilar mejor los conocimientos relativos al diseño e implementación de soluciones. A su vez este proyecto se basa en otros trabajos finales de grado, que carecían de un sistema de gestión de grupos y una interfaz para la creación de contenido dentro de la plataforma [3].

3 Análisis del problema

En este apartado se detalla la fase de análisis del problema, la cual se realiza al principio del desarrollo del proyecto. Esta fase es muy importante porque ayuda a especificar un lenguaje común entre desarrolladores y clientes. Asimismo, sirve también para documentar problemas, características, funciones, casos usos y requisitos.

En consecuencia, se propone comprender el dominio del problema, analizar y definir los requisitos funcionales y no funcionales. A continuación, se describirá siguiendo como modelo el estándar IEEE-830 [4], el análisis realizado durante esta fase.

3.1 Glosario de términos

Actor: papel ejecutado por una entidad externa o persona que interactúa con el sistema.

Usuario: hace referencia a cualquier cuenta dada de alta en la plataforma web. Puede ser de dos tipos: alumno o profesor.

Alumno: persona que se encuentra estudiando la materia de “Diseño de Software”.

Profesor: persona que imparte la asignatura de “Diseño del Software”.

Grupo: agrupación de usuarios compuesta obligatoriamente por un profesor y al menos un alumno.

Patrón: es una descripción o plantilla de cómo resolver un problema que puede utilizarse en muchas situaciones diferentes [5].

Lección: texto explicativo sobre un patrón escrito por un profesor para los alumnos.

Ejercicio: conjunto de preguntas tipo test sobre un patrón creado por un profesor para los alumnos.

Proyecto: archivos de código para poder ser ejecutados por los alumnos.

Resultado: información sobre el número de respuestas correctas e incorrectas que el alumno ha obtenido al realizar un ejercicio.

Estadística: gráfico realizado con todos los resultados de los alumnos por grupo y ejercicio.

3.2 Diagrama de contexto

El diagrama de contexto ofrece una abstracción de alto nivel de la aplicación, donde se definen los actores o entidades externas que interactúan con el sistema, y también se describen los límites de este.

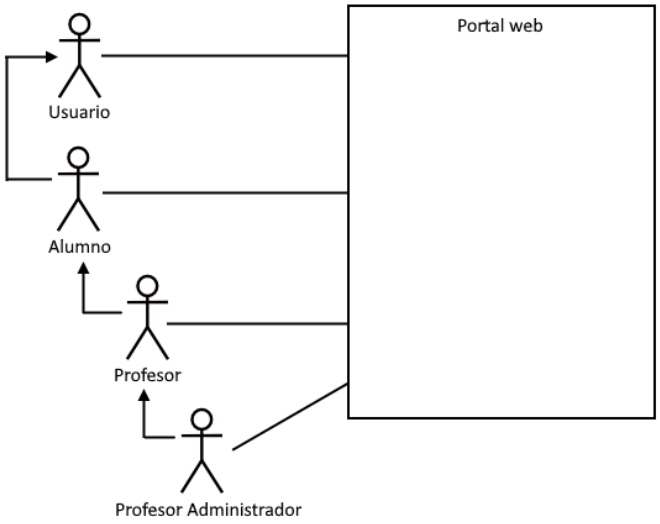

Ilustración 3 - Diagrama de contexto

En la Ilustración 3 se observa que el contexto de la aplicación es el portal web y los actores involucrados con el sistema son:

- Un usuario que es una persona con cuenta registrada en el sistema.
- Los otros tres actores son parecidos, porque los profesores administradores contienen las mismas funcionalidades que el profesor, aparte de las suyas. Y ocurre lo mismo con el profesor y el alumno respectivamente.

3.3 Modelo de dominio

El modelo de dominio tiene como finalidad representar de forma gráfica el vocabulario y conceptos clave del dominio del problema. Facilitando la comunicación entre cliente y equipo de desarrollo, y de forma indirecta, prevenir que ambas partes tengan definiciones o relaciones diferentes.

En el dominio se puede diferenciar varios conceptos importantes: individuos que interactúan con el sistema, consumiendo y creando contenido en la plataforma (usuario, alumno, profesor…), elementos que son parte del contenido del portal web (patrones, lecciones, ejercicios…) y por último las relaciones entre estos conceptos (contienen, crean, realizan…).

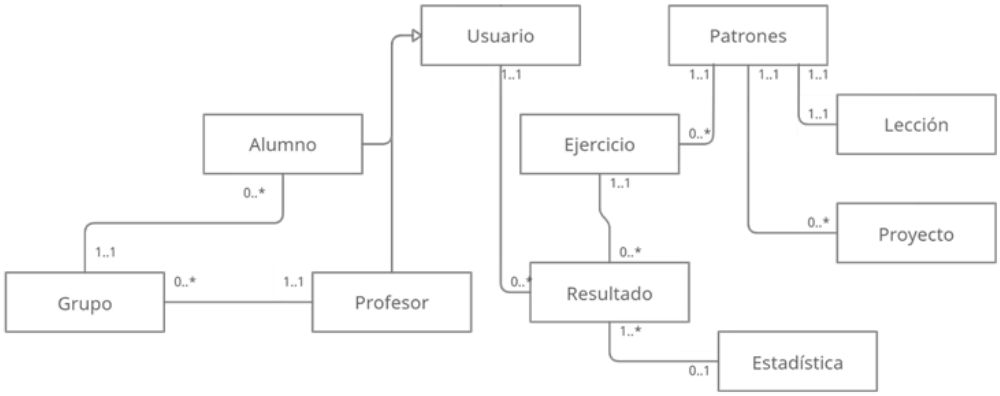

Ilustración 4 - Diagrama de dominio

Se puede observar en la Ilustración 4, el diagrama de dominio de la solución que se quiere implementar, para mejorar los conocimientos del alumno y ofrecer una herramienta versátil al profesor.

3.4 Casos de uso

En este apartado se detallan las características funcionales de la solución a implementar del problema. Cabe destacar, que se han divido las características en funcionalidades pequeñas realizadas por los distintos actores del sistema. Se adjuntan diagramas de cada actor con sus esquemas de cada caso de uso.

El esquema definido para cada caso de uso usa la plantilla recomendada por IEEE 830:

Caso de uso	Código identificativo - Nombre del caso de uso
Actor/es	Lista de entidades que realizan el caso.
Descripción	Explicación breve sobre la funcionalidad del sistema.
Desencadenante	Evento que desencadena la funcionalidad descrita. Opcional.
Precondición	Lista de condiciones que deben de cumplir antes de que la funcionalidad se realice. Opcional.
Postcondición	Breve explicación del resultado por realizar el caso de uso.
Flujo	Secuencia de eventos que suceden cuando se está realizando el caso uso. Los eventos los realiza el actor o actores y el sistema.
Include	Sección que especifica que otros casos de uso se realizan dentro de este. Solo se usará el identificador para referenciar los casos de uso.

Tabla 1 - Plantilla de casos de uso

La Ilustración 5 presenta el diagrama de casos de uso general de todos los actores con el sistema. Se omite añadir en cada caso de uso las herencias de actores, es decir, los casos de uso del usuario son todas realizables por profesores y profesores administradores.

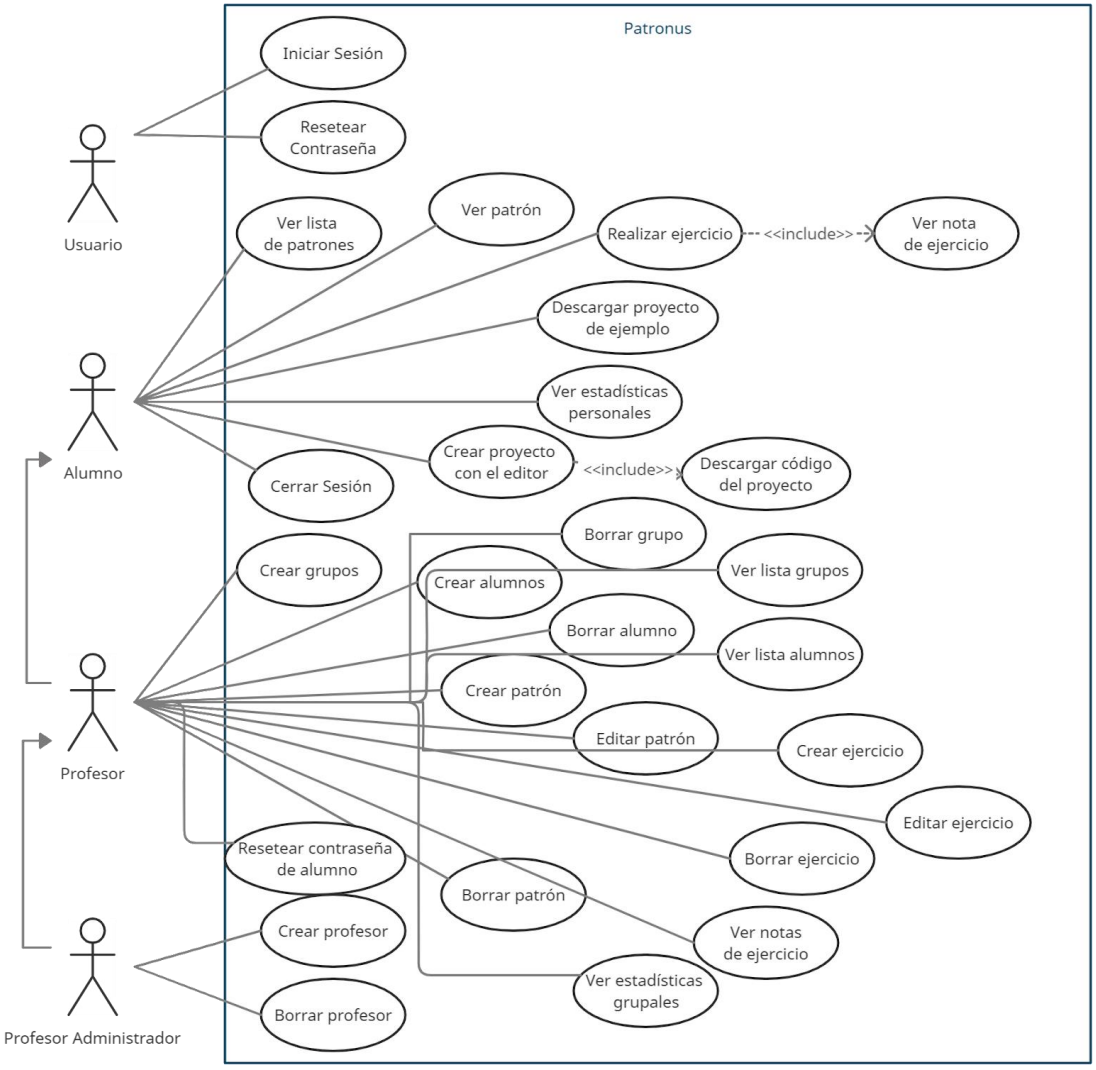

Ilustración 5 - Diagrama de casos de uso general

3.4.1 Funcionalidades del usuario

A continuación, se detallan todos los casos de uso que puede realizar el usuario con la página web. La Ilustración 6 detalla el diagrama de caso de uso relativo al usuario, para obtener una vista más simple de la general.

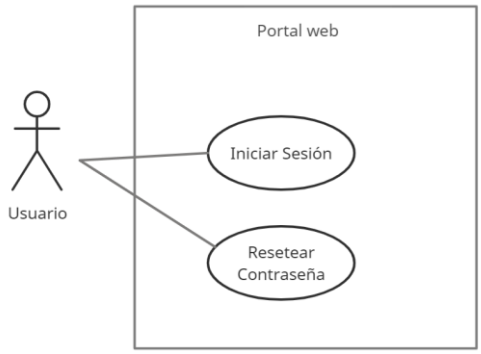

Ilustración 6 - Diagrama de casos de uso del usuario

Caso de uso	C1 - Iniciar sesión
Actor/es	Usuario.
Descripción	El usuario inicia sesión en la aplicación web y es redirigido al panel de control de su tipo de usuario.
Desencadenante	El usuario entra en la página web de inicio de sesión.
Precondición	El usuario no ha iniciado sesión aún.
Postcondición	**Sí credenciales inválidas:** Avisar al usuario de invalidez. **Sí credenciales válidas:** El usuario obtiene una sesión y es redirigido a la página web.
Flujo	1. El usuario entra en la página de inicio de sesión. 2. El usuario rellena el formulario con datos válidos. 3. El usuario da clic en iniciar sesión. 4. El sistema le redirige al panel de control de su tipo de usuario.
Include	-

Tabla 2 - Caso de uso: Iniciar sesión

Caso de uso	C2 - Reiniciar contraseña
Actor/es	Usuario.
Descripción	El usuario puede cambiar la contraseña de su cuenta por otra a su elección.
Desencadenante	El usuario entra en la opción de cambiar contraseña.
Precondición	El usuario ha iniciado sesión.

Postcondición	**Sí contraseña inválida:** Avisar al usuario de invalidez. **Sí credenciales válidas:** El usuario obtiene una confirmación del cambio y es redirigido a la página web principal.
Flujo	1. El usuario entra en la opción cambiar contraseña. 2. El sistema muestra un formulario. 3. El usuario rellena el formulario con la nueva contraseña. 4. El usuario hace clic en enviar. 5. El sistema comprueba si es válida la contraseña. 6. El sistema cambia la contraseña.
Include	-

Tabla 3 - Caso de uso: Reiniciar contraseña

3.4.2 Funcionalidades del alumno

A continuación, se detallan todos los casos de uso que puede realizar el alumno con la página web. La Ilustración 7 detalla el diagrama de caso de uso relativo al alumno, para eliminar todo el ruido que genera el diagrama general.

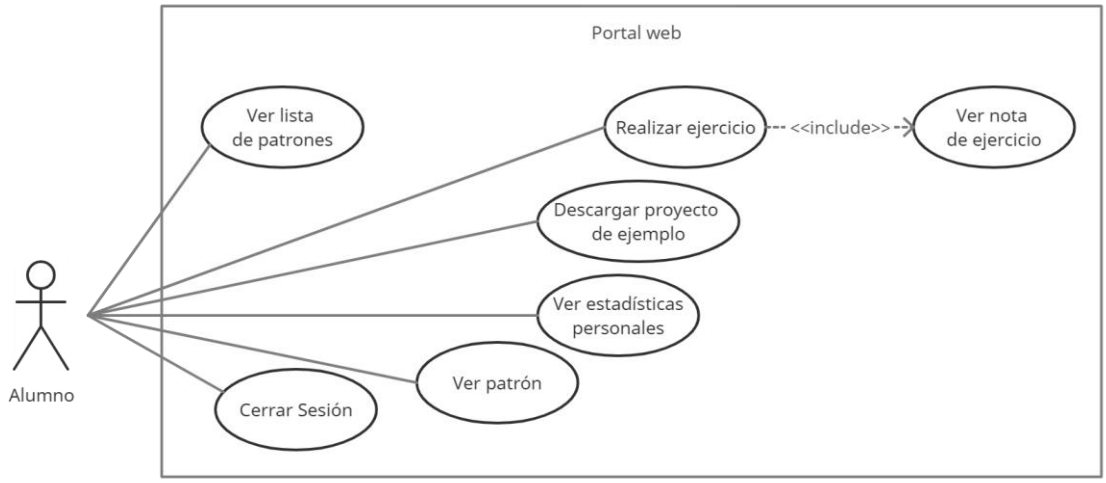

Ilustración 7 - Diagrama de casos de uso del alumno

Caso de uso	C3 - Ver lista de patrones
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario puede ver la lista de patrones disponibles en el sistema.
Desencadenante	El usuario entra en la página de lista de patrones.

Precondición	El usuario ha iniciado sesión.
Postcondición	El usuario obtiene la vista de lista de patrones.
Flujo	1. El usuario hace clic en el acceso a la vista de lista de patrones. 2. El sistema le redirige a la vista de lista de patrones.
Include	-

Tabla 4 - Caso de uso: Ver lista de patrones

Caso de uso	C4 - Ver patrón
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario puede ver la lección de un patrón.
Desencadenante	El usuario entra en la página de lista de patrones.
Precondición	El usuario ha iniciado sesión.
Postcondición	El usuario obtiene la vista del patrón seleccionado.
Flujo	1. El usuario hace clic en el patrón en la lista de patrones. 2. El sistema le redirige a la vista del patrón.
Include	-

Tabla 5 - Caso de uso: Ver patrón

Caso de uso	C5 - Realizar ejercicio
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario realiza un ejercicio dentro de la lección de un patrón.
Desencadenante	El usuario entra en la página de lista de patrones.
Precondición	El usuario ha iniciado sesión.
Postcondición	El sistema registra en el sistema el resultado.
Flujo	1. El usuario entra en el ejercicio de un patrón. 2. El sistema presenta un formulario tipo test con preguntas. 3. El usuario completa el formulario. 4. El usuario hace clic en el botón enviar. 5. El sistema redirige al usuario a la página del patrón.
Include	C6

Tabla 6 - Caso de uso: Realizar ejercicio

Caso de uso	C6 - Ver nota de ejercicio
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario ve la nota de su último resultado de un ejercicio.
Desencadenante	El usuario entra en la lección de un patrón.
Precondición	El usuario ha iniciado sesión y ha realizado el ejercicio.
Postcondición	El sistema muestra por pantalla la nota del último resultado.
Flujo	1. El usuario observa en ejercicio la nota. 2. El sistema enseña el resultado del ejercicio.
Include	-

Tabla 7 - Caso de uso: Ver nota de ejercicio

Caso de uso	C7 - Descargar proyecto de ejemplo
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario descarga un proyecto de ejemplo de un patrón.
Desencadenante	El usuario entra en la lección de un patrón.
Precondición	El usuario ha iniciado sesión.
Postcondición	El sistema envía el proyecto para que el usuario lo pueda descargar.
Flujo	1. El usuario hace clic en el enlace del proyecto de ejemplo. 2. El sistema prepara el proyecto para la descarga. 3. El sistema envía al usuario el proyecto.
Include	-

Tabla 8 - Caso de uso: Descargar proyecto de ejemplo

Caso de uso	C8 - Ver estadísticas personales
Actor/es	Alumno, Profesor, Profesor administrador.
Descripción	El usuario puede ver un gráfico con el porcentaje de ejercicios completados y su nota media.

Desencadenante	El usuario entra en el panel de control de la aplicación.
Precondición	El usuario ha iniciado sesión.
Postcondición	El sistema muestra las gráficas de sus estadísticas.
Flujo	1. El usuario entra en el panel de control del sistema. 2. El sistema muestra las gráficas.
Include	-

Tabla 9 - Caso de uso: Ver estadísticas personales

Caso de uso	C9 - Cerrar sesión
Actor/es	Alumno, Profesor, Profesor administrador
Descripción	El usuario puede cerrar sesión
Desencadenante	El usuario da clic en la opción cerrar sesión.
Precondición	El usuario cierra la sesión y el sistema le muestra la pantalla de inicio de sesión.
Postcondición	El sistema envía el código del proyecto creado en un archivo comprimido.
Flujo	1. El usuario hace clic en cerrar sesión 2. El sistema cierra la sesión con el usuario. 1. El sistema redirige al usuario a la ventana de inicio de sesión.
Include	-

Tabla 10 - Caso de uso: Cerrar sesión

3.4.3 Funcionalidades del profesor

En esta sección se detallan todos los casos de uso que puede realizar el profesor con la página web. La Ilustración 8 detalla el diagrama de caso de uso relativo al profesor, para eliminar todo el ruido que genera el diagrama general.

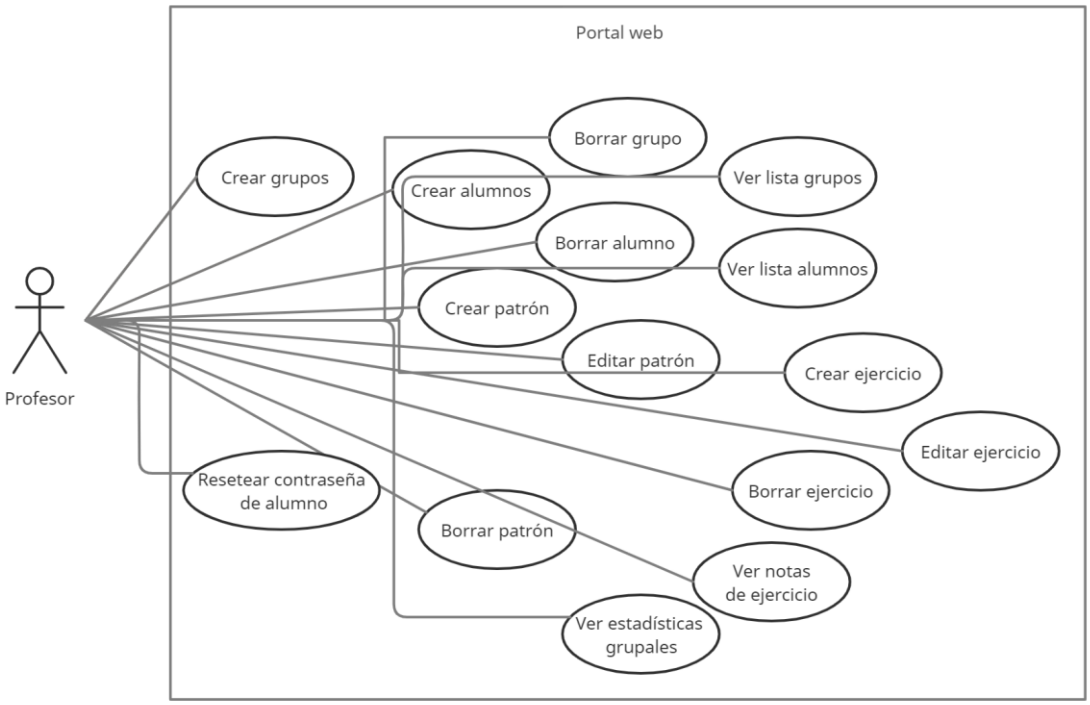

Ilustración 8 - Diagrama de casos de uso del profesor

Caso de uso	C10 - Crear grupos
Actor/es	Profesor.
Descripción	El profesor puede crear grupos de alumnos.
Desencadenante	El profesor hace clic en crear grupo.
Precondición	El profesor debe tener sesión iniciada.
Postcondición	El sistema crea un grupo nuevo vacío.
Flujo	1. El sistema muestra un formulario donde pide nombre del grupo, profesor asignado y alumnos. 2. El usuario rellena el formulario. 3. El sistema crea el grupo.
Include	-

Tabla 11 - Caso de uso: Crear grupo

Caso de uso	C11 - Borrar grupo
Actor/es	Profesor.

Descripción	El profesor puede crear grupos de alumnos.
Desencadenante	El profesor hace clic en borrar grupo.
Precondición	El profesor debe tener sesión iniciada y el grupo no tener alumnos asignados.
Postcondición	El sistema borra el grupo.
Flujo	1. El profesor entra a inspeccionar un grupo. 2. El profesor hace clic en borrar el grupo. 3. El sistema muestra un mensaje de confirmación. 4. El profesor confirma el mensaje. 5. El sistema confirma el borrado del grupo con una notificación.
Include	-

Tabla 12 - Caso de uso: Borrar grupo

Caso de uso	C12 - Crear alumno
Actor/es	Profesor.
Descripción	El profesor puede crear un alumno dentro de un grupo.
Desencadenante	El profesor hace clic en añadir alumno dentro de un grupo.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema crea el alumno dentro del grupo.
Flujo	1. El profesor hace clic en añadir usuario. 2. El sistema muestra un formulario compuesto por: nombre y correo educativo. 3. El profesor rellena el formulario con los datos del nuevo usuario y hace clic en guardar. 4. El sistema valida si la información introducida es correcta. 5. El sistema confirma la creación del usuario.
Include	-

Tabla 13 - Caso de uso: Crear Alumno

Caso de uso	C13 - Borrar alumno
Actor/es	Profesor.
Descripción	El profesor puede borrar un alumno dentro de un grupo.
Desencadenante	El profesor hace clic en borrar alumno dentro de un grupo.

Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema borra el alumno dentro del grupo.
Flujo	1. El profesor hace clic en borrar el usuario. 2. El sistema pregunta si realmente se quiere borrar el usuario. 3. El profesor confirma el borrado. 4. El sistema borra y confirma el borrado del usuario.
Include	-

Tabla 14 - Caso de uso: Borrar alumno

Caso de uso	C14 – Ver lista de grupos
Actor/es	Profesor.
Descripción	El profesor puede visualizar la lista de grupos de la aplicación.
Desencadenante	El profesor hace clic en ver grupos.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema muestra una lista ordenable y filtrable de los grupos.
Flujo	1. El profesor hace clic en ver grupos. 2. El sistema retorna la lista total de grupos.
Include	-

Tabla 15 - Caso de uso: Ver lista de grupos

Caso de uso	C15 – Ver lista de alumnos
Actor/es	Profesor.
Descripción	El profesor puede visualizar la lista de alumnos en un grupo.
Desencadenante	El profesor hace clic en ver alumnos de un grupo.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema muestra una lista ordenable y filtrable de los alumnos de un grupo.
Flujo	1. El profesor hace clic en ver alumnos de un grupo. 2. El sistema retorna la lista total de alumnos de un grupo.

Include	-

Tabla 16 - Caso de uso: Ver lista de alumnos

Caso de uso	C16 – Crear patrón
Actor/es	Profesor.
Descripción	El profesor puede crear lecciones de patrones en la aplicación.
Desencadenante	El profesor hace clic en crear nueva lección.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema crea correctamente la lección introducida por el profesor.
Flujo	1. El profesor hace clic en crear nueva lección. 2. El sistema muestra un formulario al profesor compuesto por: titulo, descripción, lección, documentos y proyectos de ejemplo. 3. El profesor rellena este formulario y hace clic en guardar. 4. El sistema valida los datos antes de guardar. 5. El sistema crea la nueva lección del patrón.
Include	-

Tabla 17 - Caso de uso: Crear patrón

Caso de uso	C17 – Editar patrón
Actor/es	Profesor.
Descripción	El profesor puede editar lecciones de patrones en la aplicación.
Desencadenante	El profesor hace clic en editar lección.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema guarda correctamente los cambios realizados por el profesor en la lección.
Flujo	1. El profesor hace clic en editar lección. 2. El sistema muestra un formulario al profesor compuesto por: titulo, descripción, lección, documentos y proyectos de ejemplo. Este formulario expone la información actual de la lección. 3. El profesor modifica este formulario y hace clic en guardar. 4. El sistema valida los datos antes de guardar. 5. El sistema guarda los cambios realizados por el profesor en la lección.

Include	-

Tabla 18 - Caso de uso: Editar patrón

Caso de uso	C18 – Borrar patrón
Actor/es	Profesor.
Descripción	El profesor puede borrar lecciones de los patrones en la aplicación.
Desencadenante	El profesor hace clic en borrar lección.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema elimina permanentemente los datos de la lección.
Flujo	1. El profesor hace clic en borrar lección. 2. El sistema avisa al usuario que la lección será eliminada completamente, incluido los ejercicios y sus resultados de los alumnos. 3. El profesor confirma la acción. 4. El sistema elimina todos los datos de la lección.
Include	-

Tabla 19 - Caso de uso: Borrar patrón

Caso de uso	C19 – Crear ejercicio
Actor/es	Profesor.
Descripción	El profesor puede crear ejercicios en las lecciones.
Desencadenante	El profesor hace clic en crear nuevo ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema crea correctamente el ejercicio introducido por el profesor.
Flujo	1. El profesor hace clic en crear nuevo ejercicio. 2. El sistema muestra un formulario al profesor compuesto por: título, intentos y preguntas. Cada pregunta tendrá la posibilidad de tener varias opciones y elegir cual es la correcta o las correctas. 3. El profesor rellena este formulario y hace clic en guardar. 4. El sistema valida los datos antes de guardar. 5. El sistema crea el nuevo ejercicio del patrón.
Include	-

Tabla 20 - Caso de uso: Crear ejercicio

Caso de uso	C20 – Editar ejercicio
Actor/es	Profesor.
Descripción	El profesor puede editar ejercicios de los patrones en la aplicación.
Desencadenante	El profesor hace clic en editar ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema guarda correctamente los cambios realizados por el profesor en el ejercicio.
Flujo	1. El profesor hace clic en editar ejercicio. 2. El sistema muestra un formulario al profesor compuesto por: título, intentos y preguntas. Este formulario expone la información actual del ejercicio. 3. El profesor modifica este formulario y hace clic en guardar. 4. El sistema valida los datos antes de guardar. 5. El sistema guarda los cambios realizados por el profesor en el ejercicio.
Include	-

Tabla 21 - Caso de uso: Editar ejercicio

Caso de uso	C21 – Borrar ejercicio
Actor/es	Profesor.
Descripción	El profesor puede borrar ejercicios de los patrones en la aplicación.
Desencadenante	El profesor hace clic en borrar ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema elimina permanentemente los datos del ejercicio.
Flujo	1. El profesor hace clic en borrar ejercicio. 2. El sistema avisa al usuario que el ejercicio será eliminado completamente, incluido los resultados de los alumnos. 3. El profesor confirma la acción. 4. El sistema elimina todos los datos del ejercicio.
Include	-

Tabla 22 - Caso de uso: Borrar ejercicio

Caso de uso	C22 – Ver notas de ejercicios
Actor/es	Profesor.
Descripción	El profesor puede visualizar las diferentes notas de los alumnos que han realizado el ejercicio. Puede filtrar estas notas por grupo.
Desencadenante	El profesor hace clic en ver notas en un ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema muestra la lista de notas del ejercicio solicitado.
Flujo	1. El profesor hace clic en ver notas en un ejercicio. 2. El sistema muestra las diferentes notas de los alumnos que han realizado el ejercicio.
Include	-

Tabla 23 - Caso de uso: Ver notas de ejercicios

Caso de uso	C23 – Ver notas de ejercicios
Actor/es	Profesor.
Descripción	El profesor puede visualizar las diferentes notas de los alumnos que han realizado el ejercicio. Puede filtrar estas notas por grupo.
Desencadenante	El profesor hace clic en ver notas en un ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema muestra la lista de notas del ejercicio solicitado.
Flujo	1. El profesor hace clic en ver notas en un ejercicio. 2. El sistema muestra las diferentes notas de los alumnos que han realizado el ejercicio.
Include	-

Tabla 24 - Caso de uso: Ver notas de ejercicio

Caso de uso	C24 – Ver estadísticas grupales
Actor/es	Profesor.

Descripción	El profesor puede visualizar todas las notas de un grupo con gráficos. Puede filtrar estas notas por ejercicios.
Desencadenante	El profesor hace clic en ver notas en un ejercicio.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema muestra un gráfico con la situación actual de las notas del grupo seleccionado.
Flujo	1. El profesor hace clic en ver estadísticas de un grupo. 2. El sistema muestra las diferentes notas de los alumnos dentro de ese grupo en forma de gráfico. El gráfico es personalizable por el profesor.
Include	-

Tabla 25 – Caso de uso: Ver estadísticas grupales

Caso de uso	C25 – Reiniciar contraseña de alumno
Actor/es	Profesor.
Descripción	El profesor puede reiniciar a la contraseña del alumno en cualquier momento.
Desencadenante	El profesor hace clic en reiniciar la contraseña del alumno.
Precondición	El profesor debe tener sesión iniciada en la aplicación.
Postcondición	El sistema reinicia la contraseña del alumno por la clave por defecto.
Flujo	1. El profesor hace clic en reiniciar la contraseña del alumno. 2. El sistema muestra una advertencia sobre la acción al profesor, explicando que el usuario perderá el acceso a la aplicación con la contraseña actual. 3. El profesor confirma la acción. 4. El sistema modifica la contraseña actual por la clave por defecto.
Include	-

Tabla 26 - Caso de uso: Reiniciar contraseña de alumno

3.4.4 Funcionalidades del profesor administrador

A continuación, se detallan todos los casos de uso que puede realizar el profesor con privilegios de administrador en la página web. La Ilustración 9 detalla el diagrama de caso de uso relativo al administrador, para eliminar todo el ruido que genera el diagrama general.

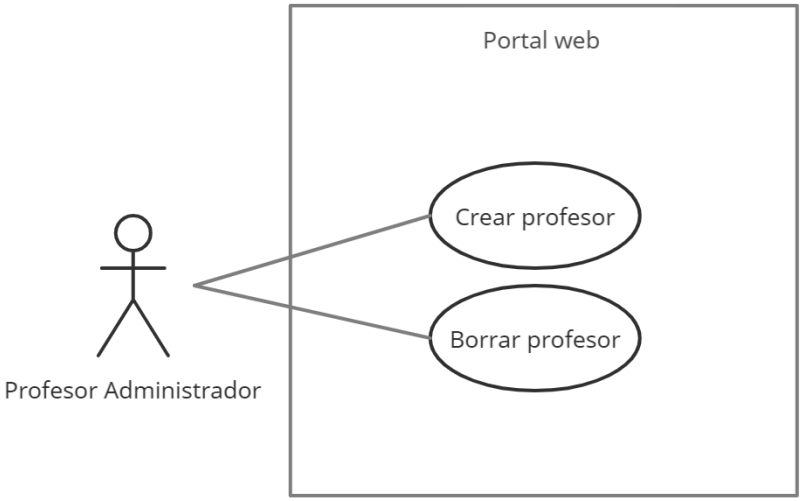

Ilustración 9 - Diagrama de casos de uso del profesor administrador

Caso de uso	C26 – Crear profesor
Actor/es	Profesor administrador.
Descripción	El administrador puede crear usuarios tipo profesor.
Desencadenante	El administrador hace clic en crear profesor.
Precondición	El administrador debe tener sesión iniciada en la aplicación.
Postcondición	El sistema crea un nuevo profesor con los datos introducidos por el administrador.
Flujo	1. El actor hace clic en crear profesor. 2. El sistema muestra un formulario que contiene: nombre, correo, grupos y contraseña. 3. El actor rellena el formulario y lo envía para su creación. 4. El sistema valida la información introducida. 5. El sistema crea el nuevo profesor.
Include	-

Tabla 27 - Caso de uso: Crear profesor

Caso de uso	C27 – Borrar profesor
Actor/es	Profesor administrador.

Descripción	El administrador puede borrar usuarios tipo profesor.
Desencadenante	El administrador hace clic en borrar profesor.
Precondición	El administrador debe tener sesión iniciada en la aplicación.
Postcondición	El sistema borra el profesor.
Flujo	1. El actor hace clic en borrar profesor. 2. El sistema muestra una advertencia al actor: "El profesor será eliminado definitivamente de la aplicación, con todos sus datos". 3. El actor confirma la acción. 4. El sistema borra el profesor completamente.
Include	-

Tabla 28 - Caso de uso: Crear profesor

3.5 Requisitos no funcionales

En este apartado se recogen todas aquellas características generales que limitan las funcionalidades o servicios del sistema, que no se pueden clasificar como requisitos funcionales:

- Todos los proyectos de ejemplo subidos a la aplicación deben de estar codificados en el lenguaje de programación Java.
- La aplicación debe ser una aplicación web disponible para navegadores web compatibles con HTML5 con acceso a internet.
- El sistema se basará en dos aplicaciones, una API implementada en Java usando el *framework* Spring Boot, y el portal web usará HTML5, Typescript, SCSS y el *framework* Angular 12.
- El sistema ha de mantener una sesión por cada usuario que caducará en veinticuatro horas.
- La base de datos donde se almacena toda la información de la aplicación ha de ser Oracle SQL.
- El despliegue se ha de realizar con un servidor basado en Wildfly, permitiendo así escalabilidad.
- Se debe de seguir un protocolo de programación para asegurar un *clean code* para su posterior mantenimiento.

- La página web tiene que ser redimensionable para los distintos tamaños de pantalla del mercado.
- La aplicación debe de permitir en un futuro la traducción a otros idiomas.
- Las contraseñas de los usuarios deben de estar cifradas en base al algoritmo MD5 [6] al almacenarlas en la base de datos.
- La aplicación debe cumplir el Reglamento Europeo de Privacidad de Datos. Para cumplir esta condición se ha de informar a todos los usuarios que los únicos datos personales almacenados serán el nombre y correo electrónico.
- Los ejercicios deben de componerse de al menos de una pregunta, y esa pregunta debe contener mínimo dos opciones. No hay restricciones de cuantas opciones son correctas.
- Los grupos no pueden existir sin un profesor o profesor administrador.

3.6 Análisis de la seguridad

El producto debe contar con un sistema de cuentas de usuario para permitir a los alumnos y profesores utilizar el portal web. Por esta razón se requiere analizar el sistema desde un punto de vista de seguridad.

A continuación, se exponen los puntos críticos que pueden exponer la integridad de la aplicación:

- Los usuarios se identifican en la aplicación mediante un nombre y una contraseña que son almacenados en la base de datos. Este es un punto crítico grave, porque cualquier persona que se conecte a la capa de persistencia del sistema tendría acceso a cualquier usuario.
- Los profesores pueden eliminar alumnos, y los administradores todo tipo de usuarios. Cualquier persona que obtenga las credenciales de un super usuario podría borrar mucha información que no esté respalda en alguna copia de seguridad.
- El producto es una página web que ofrece su servicio a los alumnos mediante internet. La aplicación tiene como limitación un determinado número de peticiones por parte de los alumnos cada segundo. Si se superase este límite, la aplicación no podría responder y colgaría el sistema. La causa de tener tantas peticiones por segundo podría deberse por un ataque de denegación de servicio (**DDoS**) [7].

3.7 Análisis de riesgos

La aplicación debe de ser un software de calidad para que el usuario esté cómodo y satisfecho usando la plataforma web. Por esta razón, se han analizado varios riesgos que impedirían tener una aplicación usable.

Por cada riesgo identificado, se ha medido su impacto en el proyecto y diseñado soluciones para abordarlos. Se utilizan los diez principios de usabilidad de Jakob Nielsen [8] como medidores de un producto software usable para los usuarios.

Nombre	Visibilidad del estado del sistema
Impacto	Moderado
Descripción	El sitio web ha de mantener informado al usuario del estado actual del sistema. Por ejemplo, si el usuario está guardando un nuevo ejercicio para sus alumnos, el sistema debe de informar al usuario del estado de este proceso.
Solución	El sistema ha de emplear componentes visuales que representen el estado del sistema como: barras de progreso, pantallas de carga, notificaciones textuales.

Tabla 29- Visibilidad del estado del sistema

Nombre	Relación entre el sistema y el mundo real
Impacto	Alto
Descripción	El sistema ha de reducir los tecnicismos al mínimo posible. Se ha de utilizar un lenguaje que el usuario entienda. También se debe de seguir un orden lógico y natural para él.
Solución	El lenguaje empleado debe ser simple y preciso. Se debe de utilizar de una forma coherente para el usuario.

Tabla 30 - Relación entre el sistema y el mundo real

Nombre	Control y libertad del usuario
Impacto	Menor
Descripción	Si el usuario se halla en un estado del sistema no deseado, él debe de disponer de herramientas u opciones para poder deshacer o salir de la situación actual.
Solución	La aplicación debe de disponer de opciones de deshacer o cancelación para situaciones crítica como el borrado de datos o edición.

Tabla 31 - Control y libertad del usuario

Nombre	Consistencia y estándares
Impacto	Moderado
Descripción	El portal web debe mantener una misma estética y un léxico único para no confundir al usuario.
Solución	La reutilización de componentes visuales mejora la asimilación del funcionamiento de las aplicaciones. Y mantener el mismo vocablo para toda la web reduce los errores cometidos por los usuarios.

Tabla 32 - Consistencia y estándares

Nombre	Prevención de errores
Impacto	Alto
Descripción	Prevenir los errores de los usuarios dentro de la aplicación, reduce el tiempo de uso de las operaciones del cliente. El tiempo empleado menor ayuda hacer que el cliente esté más satisfecho.
Solución	Limitar la entrada del usuario en los formularios, por ejemplo, en los campos que solo requieran dígitos numéricos solo permitir que el usuario escriba números.

Tabla 33 - Prevención de errores

Nombre	Reconocimiento antes que recuerdo
Impacto	Menor
Descripción	Es más fácil para el usuario recordar la interfaz, que hacer memoria de los diferentes pasos que ha seguido para obtener un resultado.
Solución	El uso de iconografía en los menús de las aplicaciones ayuda a los usuarios a recordar las distintas secciones que ofrecen.

Tabla 34 - Reconocimiento antes que recuerdo

Nombre	Flexibilidad y eficiencia de uso
Impacto	Moderado
Descripción	Una aplicación debe de poder adaptarse a todos los usuarios que la usen sin tener en cuenta el nivel de conocimiento de ellos. Al cumplir esta condición,

	el usuario estará satisfecho de poder realizar sus acciones sin confusiones.
Solución	Tanto la interfaz como las opciones ofrecidas por el sistema deben de ser lógicas para los usuarios. Por ejemplo, si el profesor quiere buscar un grupo de alumnos podrá hacerlo de diferentes formas como filtrar la lista u ordenarla.

Tabla 35 - Flexibilidad y eficiencia de uso

Nombre	Estética y diseño minimalista
Impacto	Alto
Descripción	Una estética limpia o no recargada de información ayuda a los usuarios a no saturarse. También disminuye el tiempo que se emplea en la lectura de la web, mejorando la eficiencia de la aplicación.
Solución	Usar un diseño minimalista: menús con pocas opciones y específicas, paleta de colores simples, componentes visuales como iconos deben de usarse solo para ayudar a reconocer opciones y no por estética.

Tabla 36 - Estética y diseño minimalista

Nombre	Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores
Impacto	Alto
Descripción	La aplicación se compone principalmente por formularios web, por esta razón el usuario debe de poder identificar, entender y resolver los errores que produzcan sus valores.
Solución	La implementación de mensajes emergente en el sitio web que contengan un texto autodescriptivo para ayudar al usuario son esenciales.

Tabla 37 - Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores

Nombre	Ayuda y documentación
Impacto	Moderado
Descripción	Proporcionar toda la información necesaria para entender el funcionamiento completo de la aplicación al usuario es esencial.
Solución	Añadir una sección de preguntas frecuentes (**FAQ**) con información útil y trivial de la aplicación puede ayudar al usuario a reducir sus errores.

Tabla 38 - Ayuda y documentación

3.8 Identificación y análisis de soluciones posibles

En este apartado se describe el proceso de identificación de soluciones para abordar el problema descrito en la sección anterior. Acto seguido se analizan cada una de las implementaciones exponiendo los puntos positivos y negativos.

Se ha empleado una lluvia de ideas con cuatro personas para identificar todas las soluciones posibles al problema. Cuando se obtuvieron quince ideas se pasó a descartar las soluciones imposibles de implementar por una persona en menos de seis meses. También se descartaron otras soluciones por diversas razones como, desconocimiento de la tecnología o coste de mantenimiento desmedido.

A continuación, se exponen las tres soluciones posibles que se quedaron sin descartar en el proceso de identificación:

Nombre	Aplicación de escritorio
Tecnologías	Framework para escritorio como Xamarin, Java FX, .NET
Descripción	Realizar una aplicación de escritorio para que los usuarios realicen varios tipos de ejercicios sobre los patrones de diseño. Los ejercicios serían corregidos por el sistema y de esta forma los usuarios podrían emplear la aplicación para afianzar los conocimientos.
Puntos positivos	• Tecnologías conocidas. • Desarrollo para una única plataforma. • No se necesita conexión a internet para su funcionamiento.
Puntos negativos	• Mantenimiento complicado, el usuario ha de bajarse las actualizaciones manualmente. • Desarrollo para múltiples sistemas operativos. • El usuario ha de descargarse la aplicación en su ordenador personal.

Tabla 39 - Solución: Aplicación de escritorio

Nombre	Aplicación móvil
Tecnologías	Framework para escritorio como Xamarin, Android JDK, iOS JDK
Descripción	Realizar una aplicación de dispositivos móviles para que los usuarios puedan encontrar toda la información disponible sobre los patrones de diseño. Tener todas las lecciones de un profesor a mano, puede solucionar mucho de los problemas durante los desarrollos de los alumnos.
Puntos positivos	• Implementación simple de la aplicación.

	• No se necesita conexión a internet para su funcionamiento. • Mantenimiento simple, las plataformas móviles proveen un sistema gratuito para ello.
Puntos negativos	• Desarrollo para distintos tipos de pantalla y sistema operativo. • El usuario ha de descargarse la aplicación en el teléfono móvil. • Tecnologías poco conocidas.

Tabla 40 - Solución: Aplicación móvil

Nombre	Aplicación web
Tecnologías	Framework para páginas web como Angular, React, Vue.js
Descripción	Desarrollar una página web que ofrezca a los alumnos las lecciones o apuntes de los profesores y ejercicios para afianzar la teoría. Los profesores podrían actualizar los contenidos del portal web y visualizar los resultados de los alumnos.
Puntos positivos	• Tecnologías conocidas. • Desarrollo para una única plataforma. • Mantenimiento simple, los usuarios no han de descargar nada al ser una página web. • Los usuarios no han de descargarse la aplicación.
Puntos negativos	• Se necesita conexión a internet para su funcionamiento • Desarrollo para varios tamaños de pantalla.

Tabla 41 - Solución: Aplicación web

3.9 Solución propuesta

La solución propuesta para el problema es la implementación de una aplicación web para alumnos y profesores. La portabilidad y disponibilidad de una página web permite a los usuarios aprender en cualquier sitio y momento del día. Es la razón principal porque esta solución ha sido elegida de entre las tres anteriores.

El producto ofrece a los alumnos, un sistema de lecciones de las cuales aprender todos los aspectos de los patrones de diseño. Para poner a prueba estos conceptos, el sistema ofrece varios ejercicios tipo test. Los alumnos pueden descargar documentos, proyectos de ejemplo o esquemas desde la sección de descargas.

Para los profesores, el sistema les ofrece la creación y gestión de lecciones, proyectos, ejercicios, grupos de alumnos y usuarios. El portal web proporciona una herramienta de gráficas

para visualizar los resultados de los alumnos según el grupo seleccionado. Los profesores pueden mantenerse informados del progreso de los alumnos usándola.

La implementación del producto se compondrá en cuatro fases: planificación, desarrollo, pruebas y despliegue. Para la primera fase de planificación se definen la documentación básica para afianzar el desarrollo. Después, en la segunda fase se implementa la aplicación y se crean pruebas unitarias para comprobar su fiabilidad. Por último, en las dos últimas fases se comprobará el correcto funcionamiento de la aplicación y se desplegará en una máquina virtual en Microsoft Azure.

3.10 Plan de Trabajo

En el siguiente apartado se expone el plan de trabajo previsto para la implementación de la solución detallada en la sección anterior.

Para la creación del plan, se ha tenido en cuenta que el alumno trabaja una jornada completa de cuarenta horas semanales, es decir, ocho horas diarias. Por esta razón entresemana el alumno solo dispone de tres horas para el proyecto.

El calendario dispondrá de seis meses (aproximadamente veintisiete semanas) para repartir en las distintas fases del proyecto. En primera instancia, la fase de planificación dispondrá de cuatro semanas. En las dos primeras semanas de esta fase, se analizará el problema y la solución propuesta con el profesor y se detallará las características y requisitos funcionales o no del producto. En la tercera semana se definirán los diferentes casos de uso de la aplicación y los diagramas como el glosario. Para la última semana, se diseñará la arquitectura de la aplicación y las múltiples interfaces del sistema.

A continuación, la fase de desarrollo tiene asignado veinte semanas, porque es la etapa más larga del proyecto. Durante la primera semana se instalará el entorno para poder trabajar en la implementación. En las siguientes, se creará el proyecto y se definirá la estructura a seguir para la programación.

En el apartado de metodología se ha explicado el uso de los *sprints* de dos semanas para crear versiones estables de la aplicación. A partir de la cuarta semana, se inician los *sprints* de la siguiente forma: los dos primeros días se usan para definir y diseñar mejor las tareas a realizar, los dos últimos días se utilizan para probar que las nuevas tareas no tengan errores, y por último el resto de los días son para programar.

Por concluir, las dos fases que quedan son las más cortas y por esta razón cada una tiene asignado una semana. La fase de pruebas consistirá en comprobar el correcto funcionamiento de la aplicación en general. Y el despliegue de la aplicación se compone tanto de la creación de un entorno de producción, la instalación del producto en el entorno y las pruebas con usuarios finales.

		Semanas																								
Fase	**Días**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Planificación	28																									
Desarrollo	140																									
Pruebas	7																									
Despliegue	7																									

Tabla 42 - Plan de trabajo

4 Diseño de la solución

En este apartado se describe la arquitectura interna utilizada en el desarrollo de la aplicación web y se exponen los prototipos realizados basados en los requisitos funcionales del sistema analizados en la anterior fase.

4.1 Arquitectura del Sistema

Se ha seleccionado una arquitectura de tres niveles [9] para facilitar el desacoplamiento entre los componentes del sistema. La Ilustración 10 expone el diagrama de la arquitecta a seguir. Este esquema ayuda al desarrollo independiente de cada parte, es decir, cada capa se puede desarrollar con tecnologías diferentes. Otro aspecto positivo de esta arquitectura es la capacidad de escalabilidad del sistema, porque se puede crear tantas instancias de una capa como se desee.

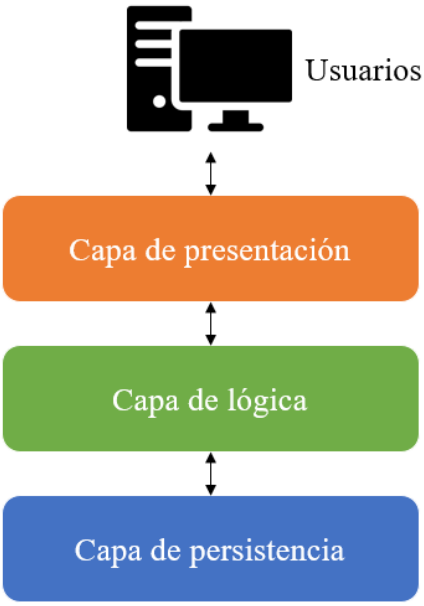

Ilustración 10 - Esquema de arquitectura de tres capas

Empezando por el final del diagrama, se encuentra la capa de persistencia, que consiste en el lugar donde se almacena la información del sistema. También provee herramientas para poder almacenar nueva información y obtener o borrar vieja. Al ser independiente, se puede utilizar el tipo de base de datos que desee, relacional o no relacional.

En este proyecto se empleará una base de datos relacional, la Ilustración 11 expone el diagrama UML de datos del sistema.

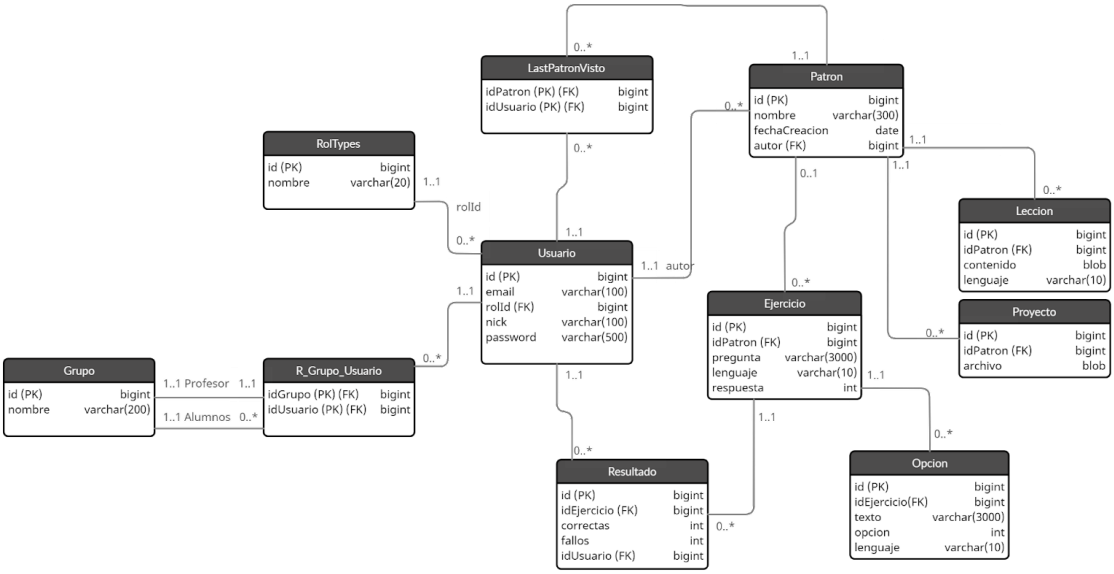

Ilustración 11 - Esquema de datos

El esquema anterior es una versión más detalla del diagrama de domino anteriormente definido. Se ha añadido relaciones cuantificadas entre los conceptos, para especificar las relaciones entre las tablas de la base de datos. Cada concepto sería una tabla, los atributos las columnas y las relaciones claves ajenas.

A continuación, se describe la funcionalidad de cada uno de los conceptos del diagrama:

- **Grupo:** almacenará la información básica de los grupos de alumnos del sistema. Obligatoriamente un grupo no puede existir sin estar relacionado a un profesor.
- **Relación grupo usuario:** contiene las relaciones entre grupo y alumnos. Un grupo puede estar relacionado a uno o más alumnos.
- **RolTypes:** tabla que contiene los tipos de roles dentro de la aplicación, que son tres: alumno, profesor y administrador.
- **Usuario:** guardará las credenciales del usuario para poder iniciar sesión y datos sensibles como nombre o correo electrónico.
- **Ejercicio:** almacenará todos los ejercicios de los patrones de la aplicación. Un ejercicio tiene que estar relacionado a un patrón, un profesor y tener mínimo dos opciones.
- **Resultado:** contiene los resultados de los alumnos que han realizado un ejercicio. Se guardará cada opción respondida por el alumno para usos de auditoría en la aplicación.

- **Patrón:** tabla que contiene la información básica de las lecciones de los patrones, como nombre, identificador del sistema y fecha de creación.
- **LastPatronVisitado:** se trata de un componente empleado para almacenar los patrones visitados por los alumnos y profesores.
- **Opción:** cada ejercicio tiene un número de opciones disponibles, estas se almacenan en esta tabla. Es obligatorio que estén relacionadas a una pregunta.
- **Proyecto:** tabla que contiene todos los archivos descargables de cada lección de los patrones.
- **Lección:** cada patrón tiene asignado un texto explicando los conceptos y características. Esta tabla alemana las lecciones y no pueden existir sin un patrón.

La segunda capa, tiene como función dar servicio a la capa de presentación y contener toda la lógica de la aplicación. Hay diferentes implementaciones para esta capa, para este proyecto se ha elegido una API Rest. La gran fortaleza de este tipo de soluciones es su escalabilidad, reutilización y que utiliza el protocolo HTTP.

La capa de lógica aparte de contener el código para crear, borrar o editar las tablas que se han mostrado en el esquema de datos, también tiene la seguridad del sistema, la validación de las entradas de los usuarios y otras funcionalidades como creación de estadísticas.

Por concluir, la capa de presentación es una página web que ofrece la posibilidad a los usuarios de utilizar el producto. Esta capa comparte los puntos positivos con la de lógica.

4.2 Prototipos de la página web

A continuación, se detalla el diseño de las diferentes interfaces de la aplicación. Se han diseñado principalmente para la pantalla de un ordenador, tomando como referencia los diez principios de usabilidad de Nielsen.

4.2.1 Inicio de sesión

En la Ilustración 12 se muestra el prototipo de la interfaz de inicio de sesión donde los usuarios podrán identificarse dentro de la aplicación. No hay ninguna forma de registrarse, porque la plataforma está ideada para que sea cerrada.

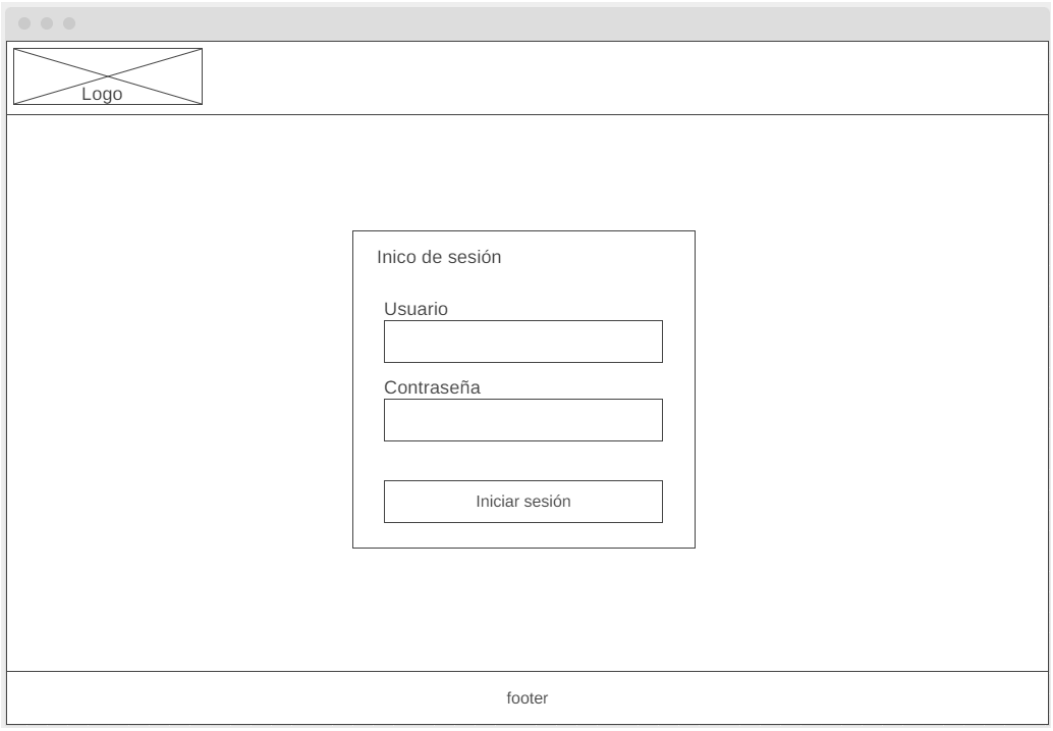

Ilustración 12 - Interfaz: Inicio sesión

4.2.2 Página principal

La aplicación está dirigida a dos grandes tipos de usuarios que se diferencian por realizar acciones diferentes en la aplicación. Se han diseñado dos tipos de interfaces para la página principal del alumno y el profesor.

Los dos tipos de interfaces comparten elementos visuales, como la barra que muestra el logo de la aplicación, el componente que muestra quien ha iniciado sesión en la esquina superior derecha y por último un menú de enlaces a otras páginas de la aplicación.

En la Ilustración 13, se expone el diseño de página principal de un alumno. El menú de opciones de la izquierda solo muestra acceso a la página de búsqueda de patrones. El contenido de la página se halla en el centro, y muestra información interesante para el alumno. Hay una gráfica que expone el progreso de las pruebas realizadas por el alumno, es decir, cuantos ejercicios ha realizado y no. También se muestran los tres últimos patrones visitados, con la posibilidad de volver a verlos; y los tres últimos resultados de los ejercicios realizados mostrando nombre del ejercicio, patrón al que pertenece, nota sobre diez y fecha de realización.

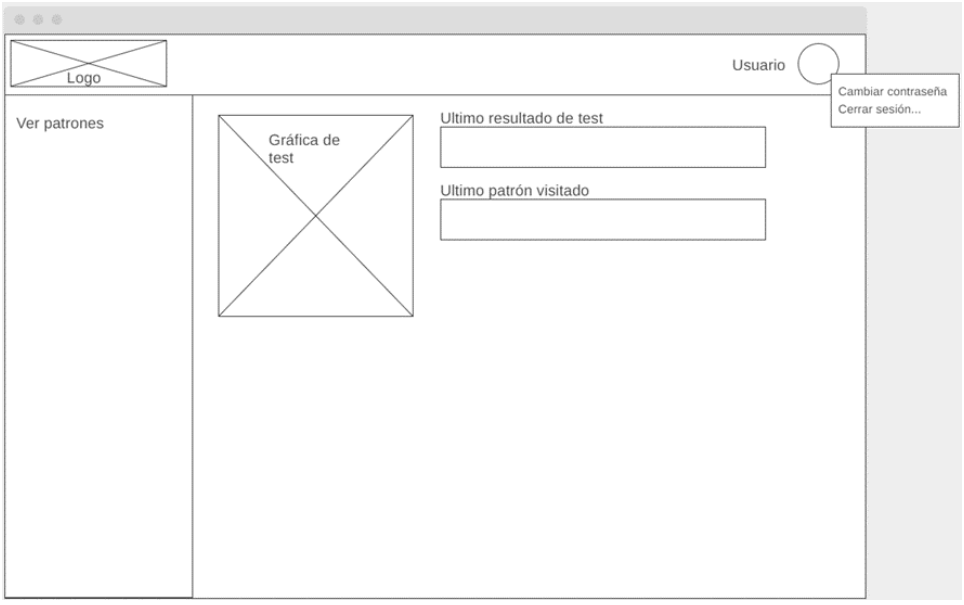

Ilustración 13 - Interfaz: Página principal del alumno

La interfaz del profesor no necesita mostrar información relativa de sus ejercicios o patrones visitados. El objetivo del profesor con la aplicación, es gestionarla. Por esta razón la página principal tiene cuatro botones que enlazan a las funcionalidades más utilizadas. El *sidebar* izquierdo ha ampliado los enlaces disponibles para la administración, como expone la Ilustración 14.

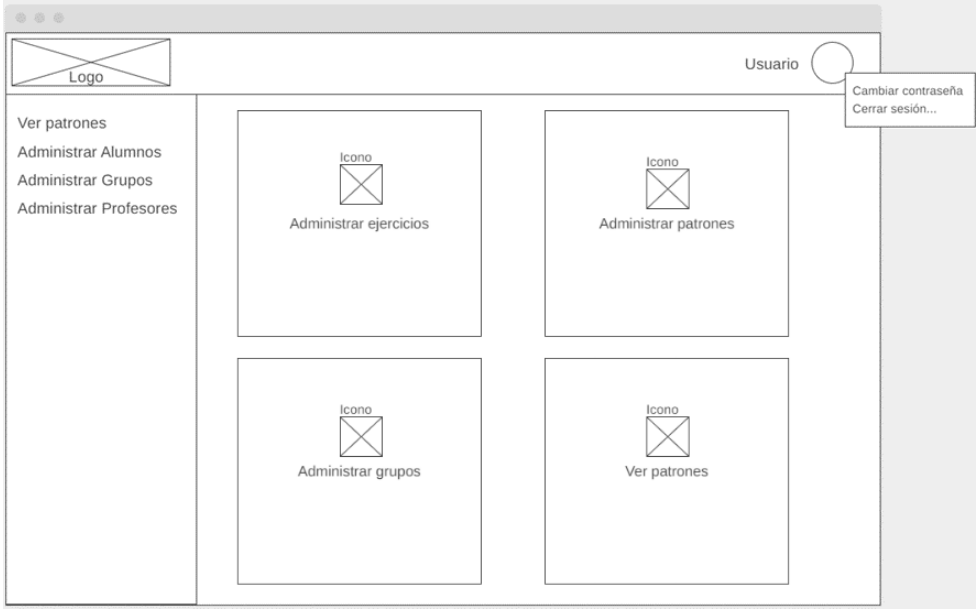

Ilustración 14 - Interfaz: Página principal del profesor

4.2.3 Lista de patrones

Los alumnos y profesores pueden buscar lecciones sobre los patrones que deseen mediante la interfaz de la Ilustración 15. La página cuenta con un filtro que permite buscar por nombre o descripción. Por cada patrón se muestra un título, descripción y el autor de la lección.

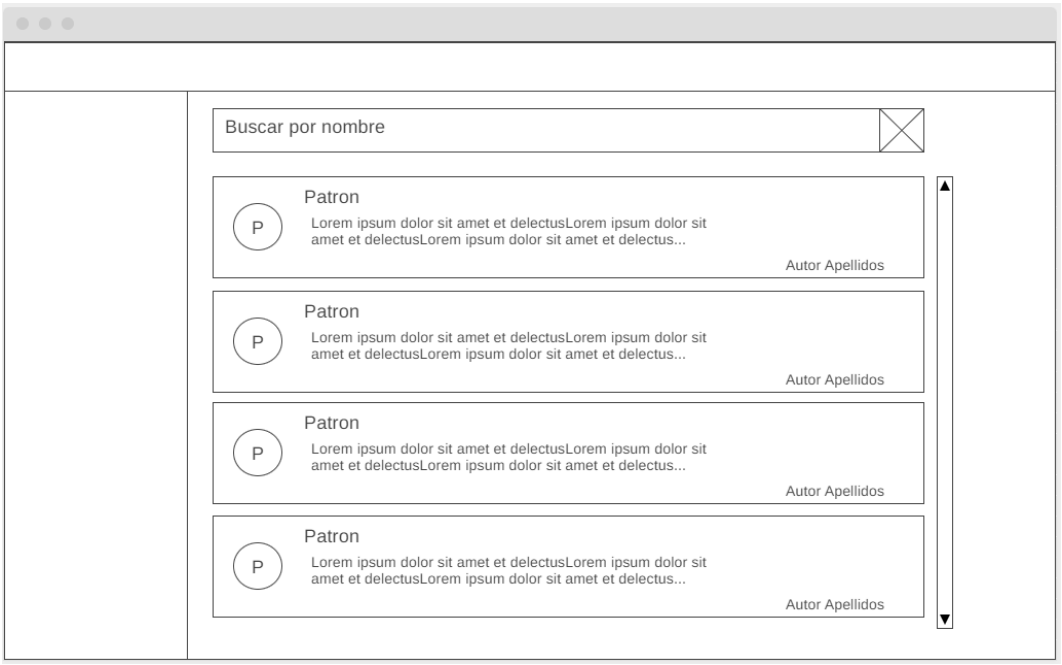

Ilustración 15 - Interfaz: Lista de patrones

4.2.4 Lección de un patrón

Las lecciones de un patrón son accesibles por alumnos y profesores. La interfaz de la Ilustración 16 muestra que una lección contiene un texto con la explicación del profesor, un conjunto de proyectos para descargar y una tabla de ejercicios.

La única diferencia entre usuarios es la interfaz de los ejercicios. La tabla de los alumnos muestra el nombre del ejercicio y los datos de la última vez que se realizó, como por ejemplo la nota y fecha de ejecución. Mientras que el profesor solo puede observar el nombre y dos enlaces: uno para visualizar las estadísticas del ejercicio y otro para editarlo.

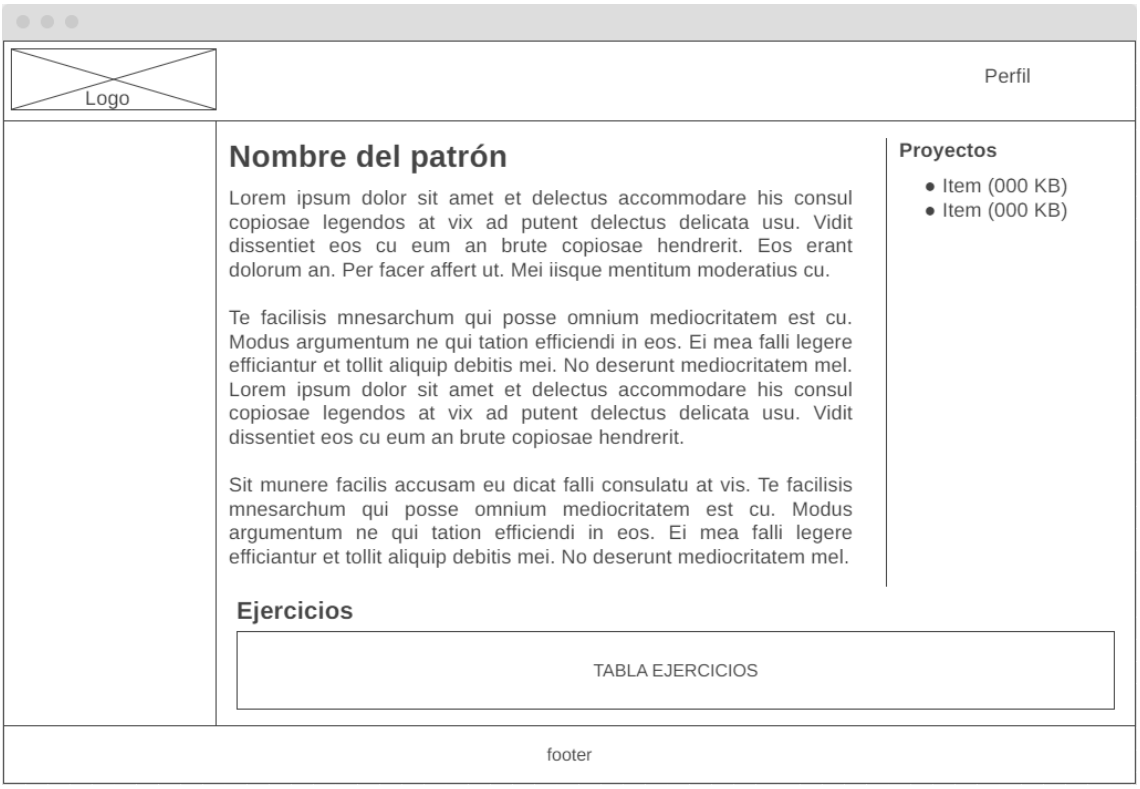

Ilustración 16 - Interfaz: Lección de un patrón

4.2.5 Realizar ejercicio

La Ilustración 17 detalla el ejercicio de un patrón en la aplicación, compuesto por preguntas de múltiple opción. La interfaz solo es accesible por alumnos.

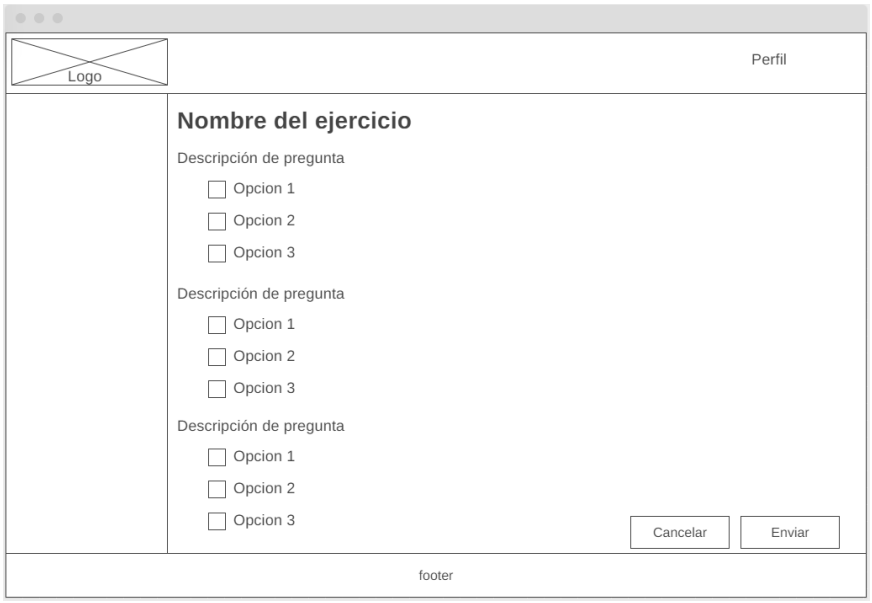

Ilustración 17 - Interfaz: Realizar ejercicio

4.2.6 Interfaces de gestión

El profesor tiene diferentes interfaces exclusivas para la gestión de grupos, usuarios, patrones y ejercicios. Para acelerar el desarrollo de la aplicación se ha diseñado una interfaz reutilizable común para estas funcionalidades. Cambiando solo los formularios de creación y edición de los objetos.

La interfaz que expone la Ilustración 18 tiene una tabla donde se encuentran todos los objetos del sistema que se requiere gestionar. Para facilitar la búsqueda en la tabla se ofrecen filtros de búsqueda personalizables. Cada elemento de la tabla tiene tres acciones posibles: ver, editar y borrar. Por último, hay un botón siempre para la creación de un nuevo artículo en la base de datos.

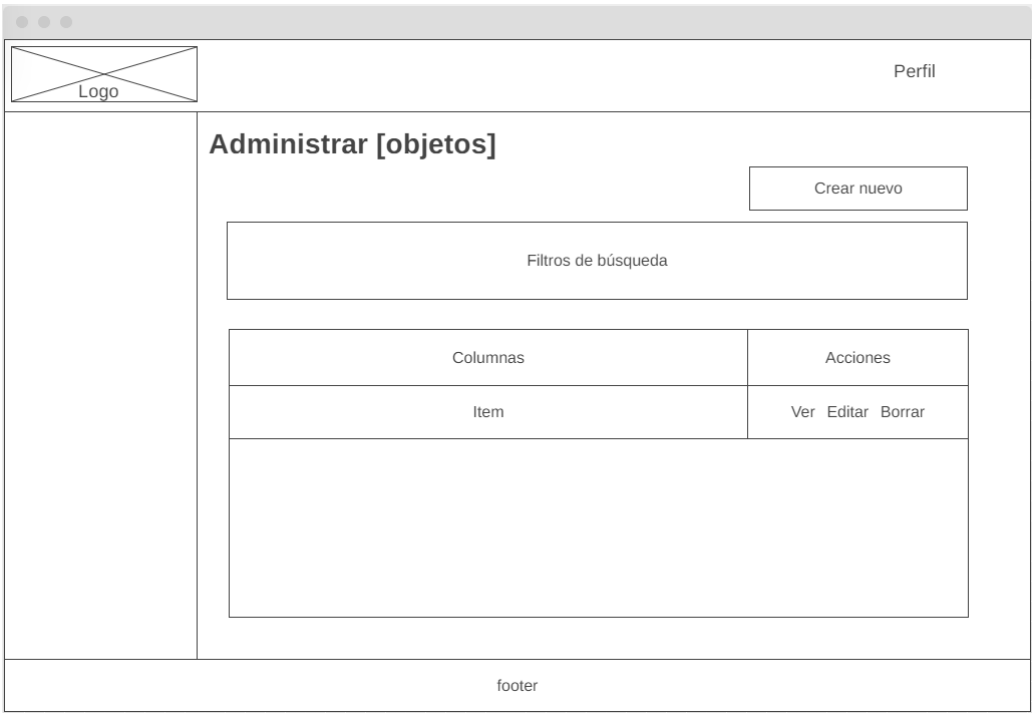

Ilustración 18 - Interfaz: Administrar genérico

4.2.7 Creación de una lección

La interfaz de creación de una lección es exclusiva para profesores. Se compone de un formulario básico para introducir nombre, descripción y el texto explicativo de los conceptos del patrón. En la Ilustración 19, se puede observar la aparición de dos tablas: la de ejercicios, donde se mostrarán y añadirán las pruebas que los alumnos pueden realizar; y la tabla de proyectos, donde se permite subir archivos para su posterior descarga en la visualización de la lección.

Cuando un profesor requiera la edición de cualquier lección del sistema, se reutilizará la misma interfaz de creación cambiando su título y rellenando todos los formularios y tablas con los datos de la lección.

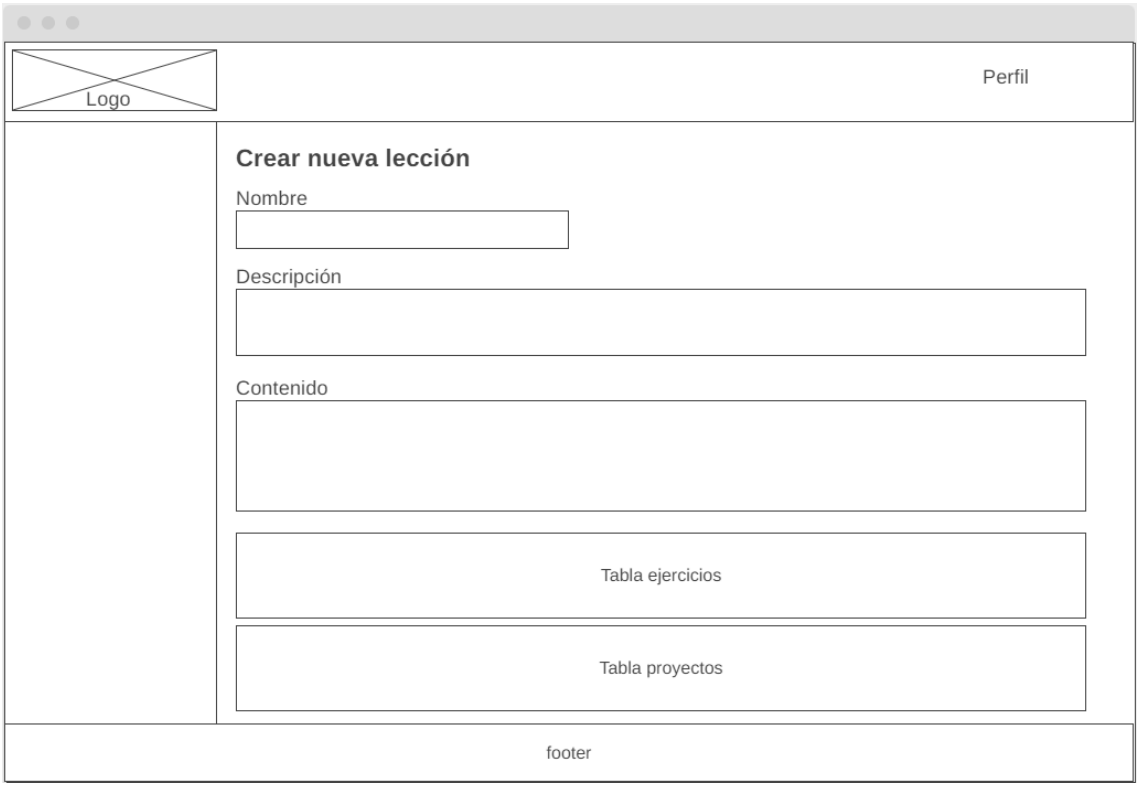

Ilustración 19 - Interfaz: Crear nueva lección

4.2.8 Creación de un ejercicio

Similar a la anterior interfaz, la creación de un ejercicio solo está disponible para profesores. La Ilustración 20 muestra un formulario dinámico, porque los profesores pueden añadir una cantidad variable de preguntas en cada ejercicio. El botón añadir opción al ser pulsado, modificaría el formulario añadiendo un campo extra de opción en la pregunta. Y por finalizar, el añadir una pregunta, crearía la estructura de la pregunta visible en la ilustración.

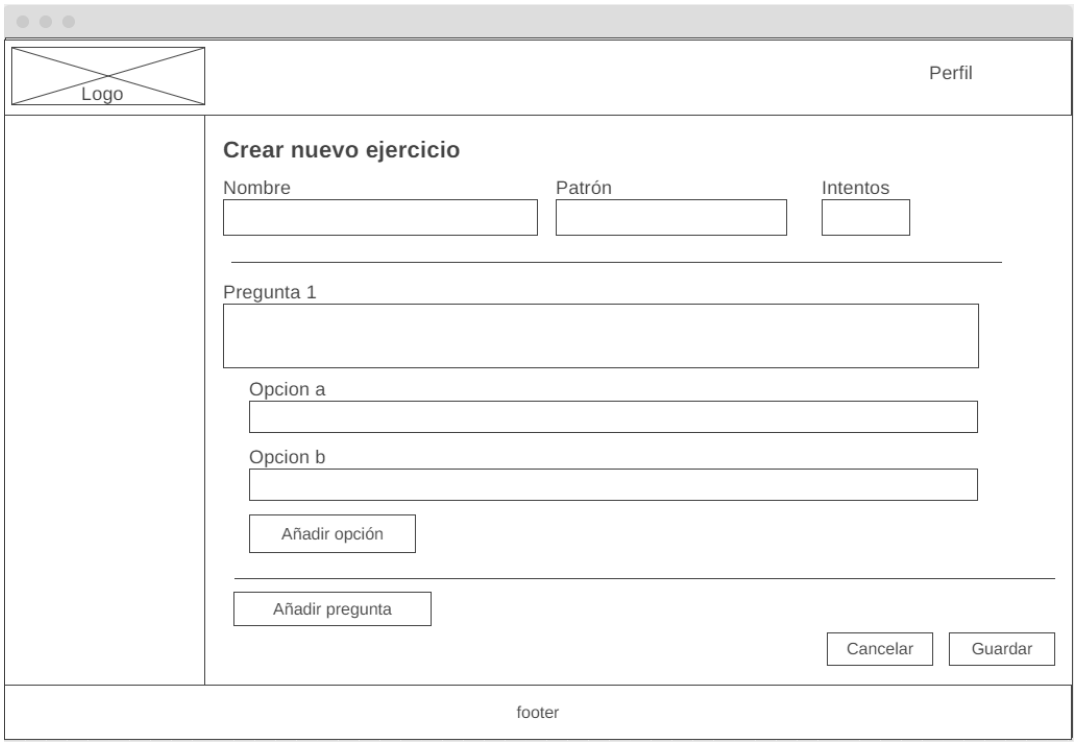

Ilustración 20 - Interfaz: Crear nuevo ejercicio

4.2.9 Creación de grupos y alumnos

Los profesores tienen la posibilidad de crear grupos de alumnos para facilitar el control de progreso de los alumnos durante la asignatura. El siguiente formulario mostrado en la Ilustración 21 se le presenta al profesor que quiera crear un grupo nuevo como ventana modal.

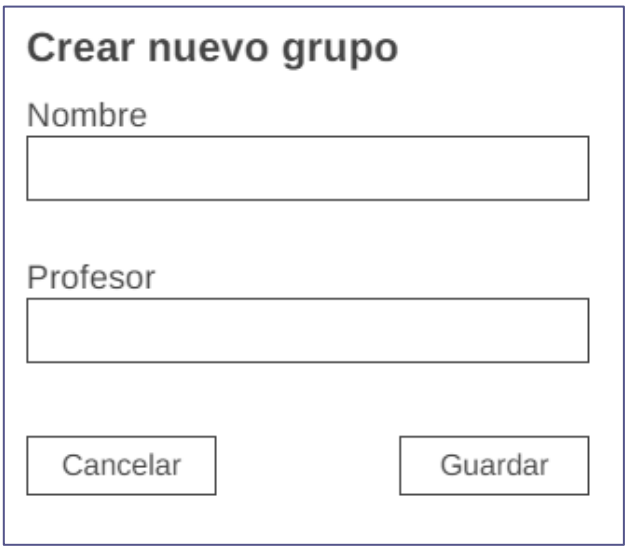

Ilustración 21 - Interfaz: Crear nuevo grupo

Una vez creado un grupo, se mostrará una página de gestión como la Ilustración 18 con el título de “Administrar grupo”. El profesor tiene la posibilidad de añadir alumnos de forma manual mediante una ventana modal con el formulario que se observa en la Ilustración 22.

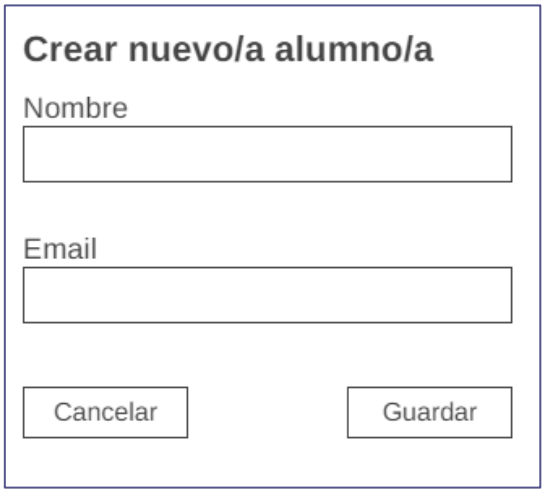

Ilustración 22 - Interfaz: Crear nuevo alumno

4.2.10 Creación de profesores

Solo los profesores con rol de administradores de la plataforma pueden crear otros profesores con o sin los mismos permisos. El formulario se muestra como una ventana modal muy similar a la creación de un alumno, pero añadiendo el campo de contraseña. La Ilustración 23 es el diseño en papel realizado.

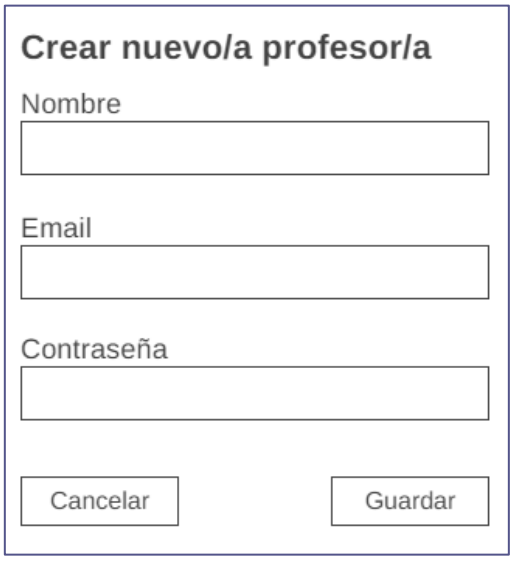

Ilustración 23 - Interfaz: Crear nuevo profesor

4.2.11 Perfil

Los usuarios de la aplicación pueden visualizar su información mediante la interfaz del perfil que se encuentra al clicar en el enlace de la esquina superior derecha. La Ilustración 24 muestra la interfaz de un usuario con rol de alumno en la aplicación.

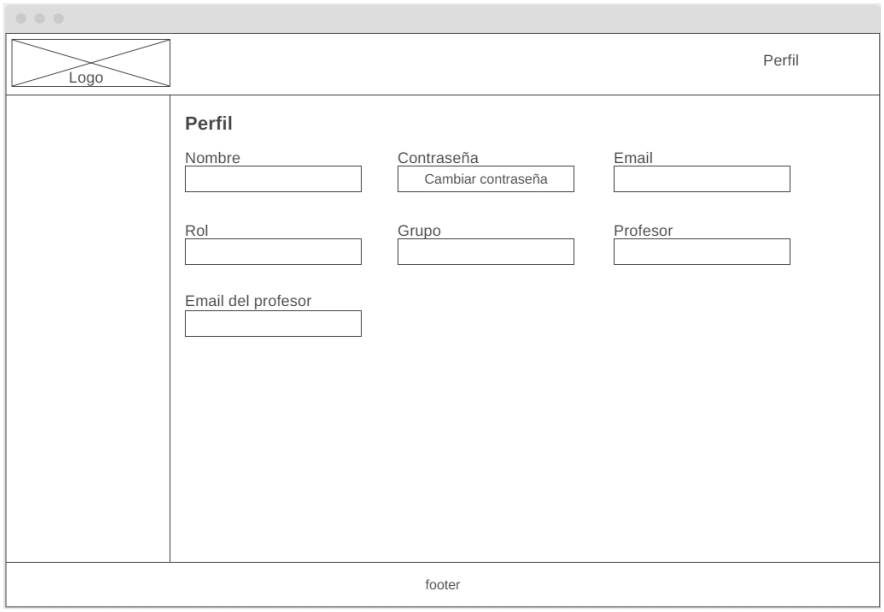

Ilustración 24 - Interfaz: Perfil de alumno

El profesorado tiene una interfaz diferente a los alumnos, porque cada profesor puede ser relacionado a más de un grupo. Por esta razón, la Ilustración 25 que es la interfaz de un profesor, tiene una lista de grupos.

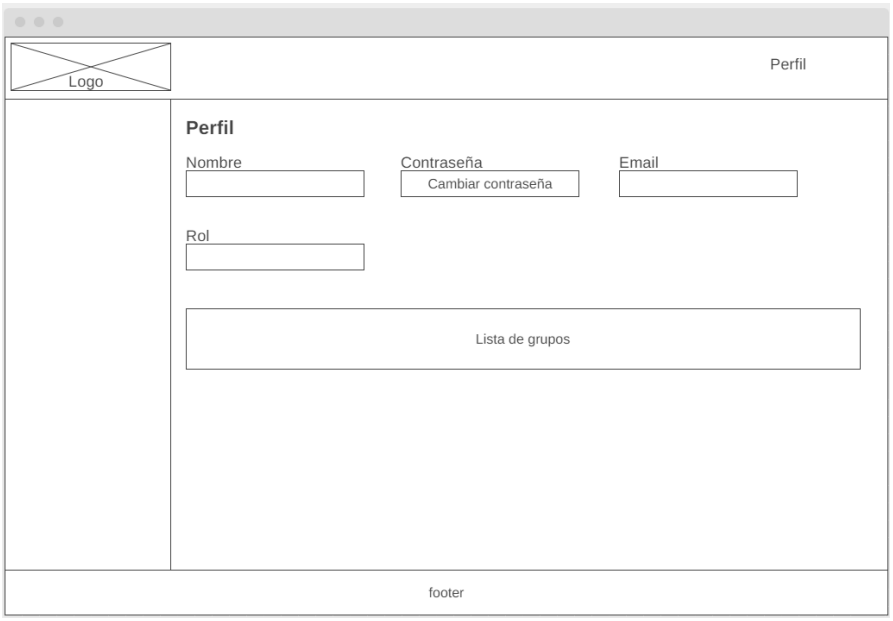

Ilustración 25 - Interfaz: Perfil de profesor

Ambos tipos de usuarios pueden modificar su contraseña mediante el botón de cambio de contraseña.

4.3 Tecnología Utilizada

En este apartado se exponen las diferentes herramientas, librerías y lenguajes empleados en el desarrollo del producto. Para cada capa de la arquitectura, se ha empleado una tecnología diferente.

- **VSCodium:** es una herramienta de edición de código para todos los lenguajes. Es de código abierto y comparte muchas similitudes con Visual Studio Code, como por ejemplo la tienda de extensiones del programa. Dentro del proyecto se ha empleado para programar la capa entera de presentación por sus herramientas de depuración.

Ilustración 26 - Logo VSCodium

- **TypeScript:** es un superconjunto de JavaScript, es decir, puedes reutilizar código escrito en JavaScript. Es un lenguaje interpretado y basado en clases. Angular obliga a utilizar este lenguaje para dotar de comportamiento las páginas web. En el proyecto se ha utilizado por esta razón.

Ilustración 27 - Logo TypeScript

- **HTML 5:** es un leguaje de marcado cuya finalidad es emplear enlaces e hipertextos para estructurar y enseñar los contenidos de una página web en internet. Se ha utilizado la versión cinco, porque intenta mejorar la lectura del código usando nuevas nomenclaturas. En el proyecto se ha usado para estructurar las interfaces de la aplicación web.

Ilustración 28 - Logo HTML5

- **SCSS:** es un metalenguaje de hojas de estilo que se traduce a CSS. Se emplea para crear estilos a los componentes HTML en la aplicación como botones, imágenes, textos, etc.

Ilustración 29 - Logo SCSS

- **Angular CLI 12:** es un *framework* para la creación de aplicaciones web de código abierto. Su arquitectura se basa en el diseño de modelo-vista-controlador, facilitando el tiempo de desarrollo y de pruebas de la aplicación. Se ha utilizado en el proyecto para crear la página web.

Ilustración 30 - Logo Angular

- **Spring Tools 4:** es un entorno de desarrollo integrado basado en Eclipse de código abierto. Cuenta con varias herramientas orientadas a la implementación y depuración de aplicaciones web que empleen Spring. Tiene disponible la tienda de extensiones de Eclipse. Dentro del proyecto se ha empleado para desarrollar la API Rest basada en Spring Boot.

Ilustración 31 - Logo Spring Tools

- **Java:** lenguaje orientado a objetos muy popular en la actualidad. El punto fuerte del lenguaje es la posibilidad de ejecutar el mismo código compilado en diferentes sistemas, gracias a su máquina virtual que interpreta el código previamente compilado. Se ha utilizado para el desarrollo de la capa de lógica.

Ilustración 32 - Logo Java

- **Spring Boot Web:** es un conjunto de librerías orientadas a la creación de aplicaciones web basadas en Java. Ofrece diversos aspectos como seguridad, persistencia, lógica para este tipo de aplicaciones. Se ha empleado parte de las librerías para la construcción de la API Rest en la capa de lógica.

Ilustración 33 - Logo Spring Boot

- **JUnit:** es una librería para la creación de pruebas unitarias automáticas en Java. Dentro del proyecto se emplea para comprobar el correcto funcionamiento de la API.

Ilustración 34 - Logo JUnit

- **DBeaver:** es un programa basado en Eclipse para la gestión de diferentes tipos de bases de datos. Ofrece diferentes herramientas para la exportación, creación de código interno o para ejecutar consultas. Se ha utilizado para crear y administrar la base de datos de la aplicación durante el desarrollo y su puesta en producción.

Ilustración 35 - Logo DBeaver

- **Oracle Database:** es un sistema de gestión de base de datos relacional propiedad de Oracle. Se ha empleado en el proyecto para crear la base de datos por su previo uso en el grado.

Ilustración 36 - Logo Oracle Database

5 Desarrollo de la solución propuesta

En este apartado se explica la estructura implementada en el proyecto, el cual esta divido en dos aplicaciones: *frontend* y *backend*. La primera parte es una página web desarrollada en Angular y la segunda una API Rest en Java y Spring Boot.

5.1 Estructura de la aplicación web

En esta sección se explica la estructura del proyecto de Angular que se ha implementado. Angular sugiere una estructura propia para trabajar, porque fomenta la reutilización de código en las aplicaciones. De esta forma la estructura se ha adaptado para reutilizar al máximo todos los componentes posibles.

Debido a la gran cantidad de código y archivos de cada uno de los paquetes del proyecto, no se detallará la funcionalidad de cada archivo. La Ilustración 37 expone la estructura raíz del proyecto Angular.

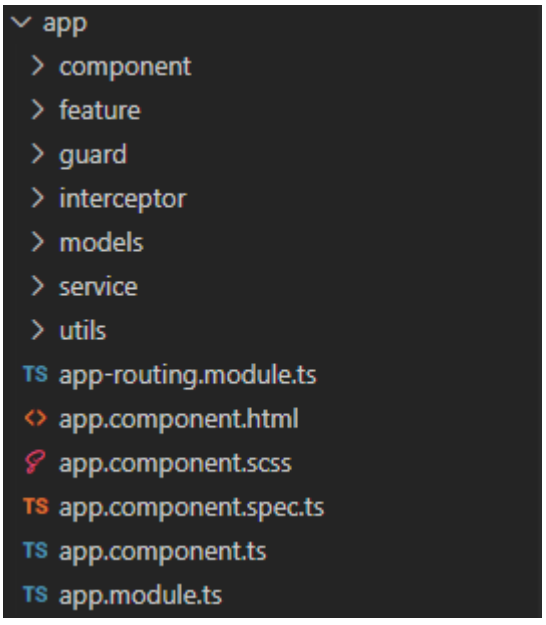

Ilustración 37 - Estructura angular

A continuación, se explica la funcionalidad de cada paquete del proyecto:

- **Component:** contiene todos los componentes reutilizables de la aplicación como campos de texto, selectores de fecha, incluso componentes visuales como tablas de datos dinámicas o efectos visuales.

- **Feature:** cada interfaz de la aplicación se considera una característica. Esta carpeta contiene todas las vistas de la página web. Los cuales reutilizan los componentes anteriormente nombrados.

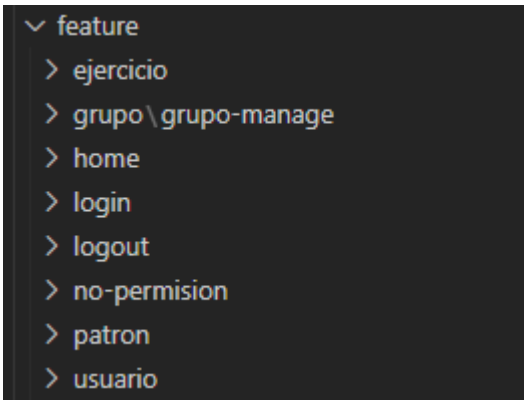

Ilustración 38 - Estructura de la carpeta Feature

- **Guard:** este tipo de objeto en Angular sirve para la seguridad del sistema. Se ejecutan cuando el cliente solicita una página de la aplicación y antes de resolverla se ejecuta un código escrito por el programado. Contiene la seguridad del sistema para no permitir a usuarios no registrados a acceder a la aplicación.
- **Interceptor:** objeto similar a un *guard*, ejecuta código antes de enviar una petición HTTP mediante el uso de la librería de HTTP de Angular. Contiene dos objetos: uno para mostrar una pantalla de carga cuando se solicitan acciones a la API, y otro para enviar el token de inicio de sesión para demostrar a la API que el usuario esta registrado.
- **Models:** carpeta que contiene todas las estructuras de datos del proyecto. Typescript tiene un sistema de clases que permite tener estructurado los diferentes tipos de datos de la aplicación.
- **Service:** contenedor que almacena todos los servicios de la aplicación. Los servicios son las clases utilizadas para hacer peticiones HTTP a servicios externos como APIs en Angular.
- **Utils:** carpeta que contiene código genérico para toda la aplicación como, por ejemplo, validadores de formularios, transformaciones de fechas u ordenación de listas.

5.2 Estructura de la API Rest

En esta sección se explica la estructura del API Rest que se ha implementado. Spring sugiere una estructura propia para trabajar, porque fomenta la reutilización de clases en las aplicaciones

mediante el patrón de inyección de dependencias. De esta forma la estructura se ha adaptado para reutilizar al máximo todas las clases posibles.

Debido a la gran cantidad de código y archivos de cada uno de los paquetes del proyecto, no se detallará la funcionalidad de cada archivo. La Ilustración 39 expone la estructura raíz del proyecto Java.

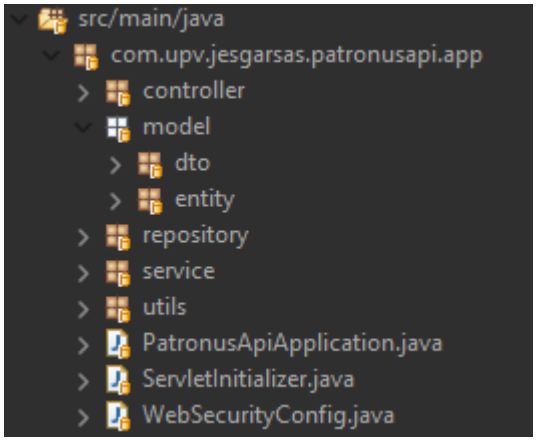

Ilustración 39 - Estructura del proyecto Java

A continuación, se explica la funcionalidad de cada paquete del API:

- **Controller:** es el paquete donde se encuentran los controladores de la API. Se usan para relacionar los servicios con una dirección web, para poder hacer una petición.
- **Model:** contender de todos los tipos y estructuras de datos de la aplicación. Se pueden diferenciar dos tipos grandes de modelos: DTO que son objetos utilizados para métodos auxiliares o envió de información de una entidad; y por último *entity* o entidad que se basa en el concepto de que una clase Java sea la representación de una tabla en la base de datos.
- **Repository:** los repositorios en Spring son las clases que se utilizan para enviar consultas a las bases de datos. Spring ofrece la posibilidad de definir consultas en SQL o utilizando la librería JPA, la cual genera automáticamente consultas traduciendo el nombre de los métodos Java de la clase.
- **Service:** al igual que los repositorios, los servicios son el grueso de la capa de lógica. Almacena todo código que extraiga, transforme, modifique o borre información de los repositorios.

- **Utils:** carpeta que contiene código genérico para toda la aplicación como, por ejemplo, validadores de formularios, transformaciones de fechas u ordenación de listas.

6 Pruebas

En este apartado se detallarán las pruebas realizadas para comprobar el correcto funcionamiento de la aplicación web y API Rest.

6.1 Pruebas unitarias API Rest

Para la comprobación de la API Rest se ha utilizado la librería JUnit para crear *suite tests*, es decir, conjuntos de pruebas unitarias. Las pruebas se han centrado en la capa de lógica y dentro de la estructura del proyecto en el paquete servicios.

Para la mayoría de los métodos de cada servicio se ha implementado de media dos tests. El primero para comprobar que el método funciona correctamente con los valores validos de entrada. Y otro para certificar que el método no admite una entrada de valores no correctos. Esta última, se puede repetir varias veces según la cantidad de casos inválidos del método.

Se adjunta la utilización de la librería Mockito para simular la recogida de datos desde los repositorios. Es una librería útil para este tipo de situaciones en las pruebas unitarias y de código abierto. La Ilustración 40 es un claro ejemplo del uso de JUnit con Mockito.

```
@Autowired
GrupoService grupoService;

@Autowired
GrupoRepository grupoRepository;

@Test
public void getGrupoByidTestSuccess() throws SQLException, ClassNotFoundException, IOException {
    Mockito.when(grupoRepository.getGrupoByid(1)).thenReturn(grupoBDMock());

    GrupoDTO grupo = grupoService.getGrupoByid(1);

    assert.equal(grupo, EXPECTED_GROUP);

}

@Test(expected = NullPointerException.class)
public void getGrupoByidTestFail() throws SQLException, ClassNotFoundException, IOException {
    GrupoDTO grupo = grupoService.getGrupoByid(null);
}
```

Ilustración 40 - Ejemplo de prueba unitaria

6.2 Pruebas funcionales

Para comprobar el correcto funcionamiento de la aplicación en su conjunto, es decir, la integración de la API Rest con la página web, se han definido pruebas funcionales o de caja negra a partir de los casos de uso.

A continuación, se detallan tres pruebas funcionales realizadas a la aplicación en el entorno de desarrollo.

6.2.1 Inicio de sesión

Los usuarios que quieran utilizar la aplicación deben de autentificarse con sus credenciales en la interfaz de inicio de sesión, como muestra la Ilustración 41.

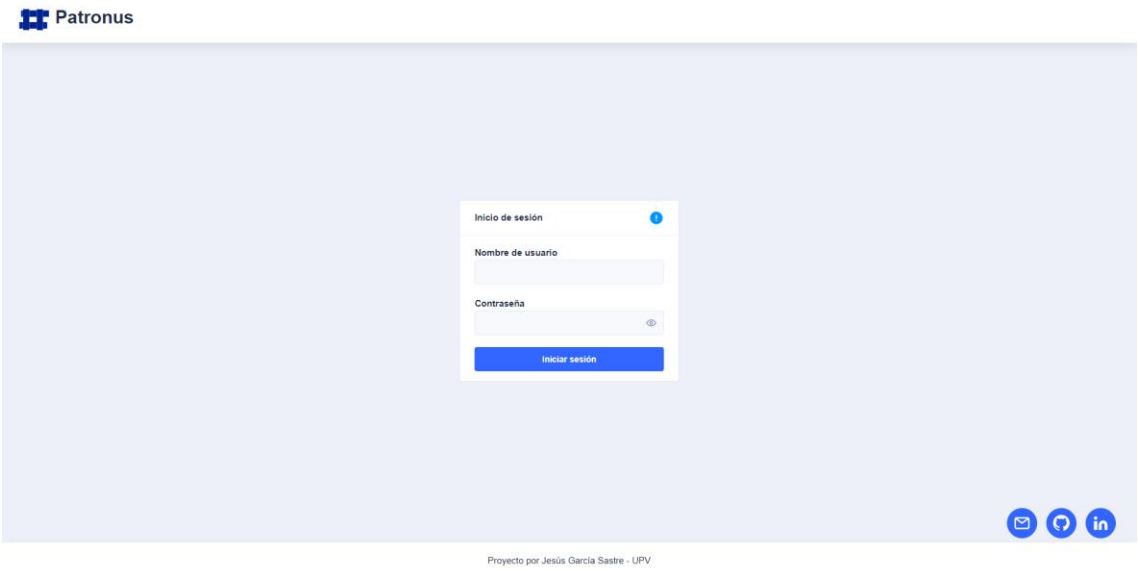

Ilustración 41 - Inicio de sesión

Si los datos introducidos no son válidos se mostrará un mensaje en la parte superior derecha que notificará al usuario de un error en las credenciales. El mensaje desaparecerá al clicarlo o después de tres segundos. Este comportamiento descrito se puede visualizar en la Ilustración 42.

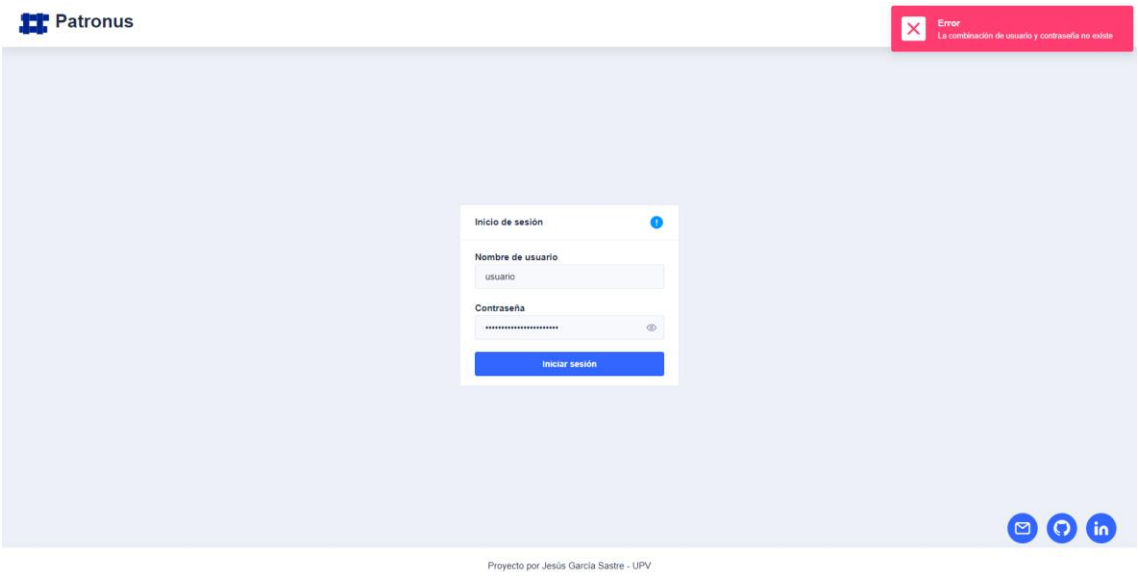

Ilustración 42 - Inicio de sesión erróneo

Pero si el usuario introduce sus credenciales de inicio de sesión y el sistema valida como correctos dichos valores, según su rol, el sistema le redirigirá a la página principal y mostrará un mensaje de confirmación del éxito de autentificación. Para los alumnos, la – expone la interfaz de página principal.

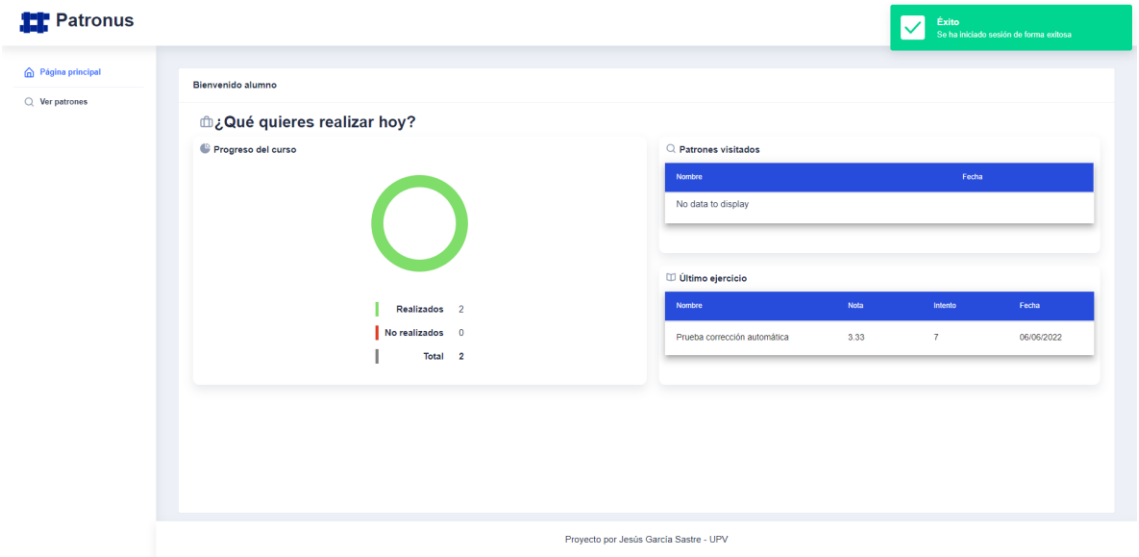

Ilustración 43 - Página principal de los alumnos

6.2.2 Realizar un ejercicio

Los únicos usuarios que pueden completar ejercicios de un patrón son los alumnos. Para poder realizar un ejercicio, primero hay que buscar la lección donde se encuentra. Se ha de entrar en la opción “Ver patrones” del menú flotante izquierdo que se puede visualizar en la Ilustración 43.

Se redirigirá al alumno a la interfaz de la Ilustración 44, con una lista de todos los patrones de la aplicación y un buscador para filtrar por nombre. Para acceder a una lección de un patrón se ha de clicar en alguna entrada de la lista de resultados.

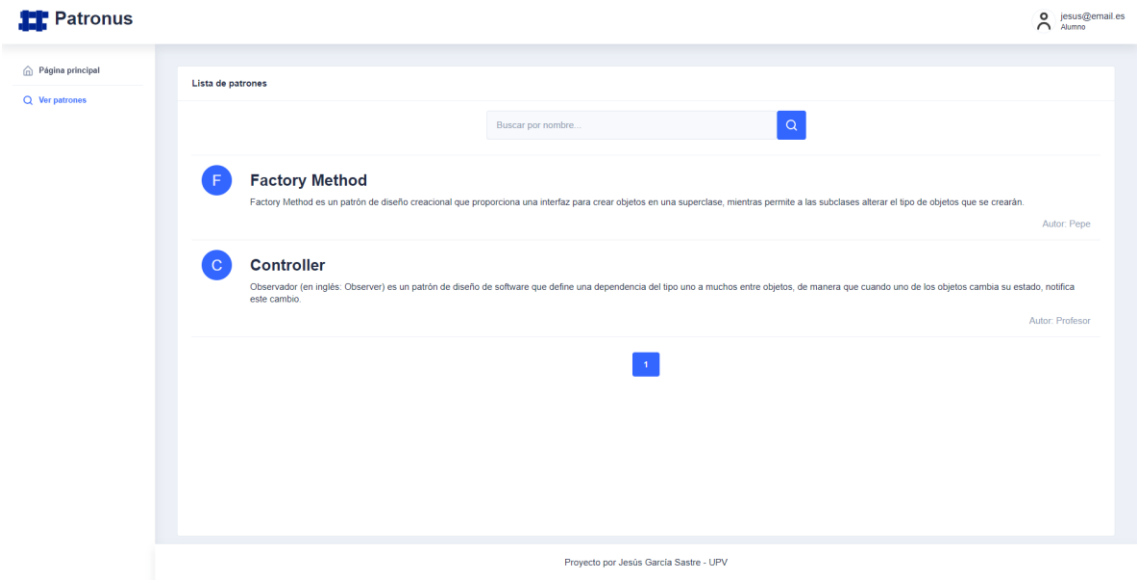

Ilustración 44 - Lista de patrones

A continuación, cuando el usuario haya elegido el patrón del que desea realizar el ejercicio, se mostrará la interfaz de lección de un patrón, que se expone en la Ilustración 45. El usuario deberá de abrir el contenedor de los ejercicios y hacer clic en el botón azul triangular del ejercicio que desea realizar dentro de la tabla.

Ilustración 45 - Lección de un patrón

La interfaz de un ejercicio es variable y depende de la cantidad de preguntas y opciones que tenga. Para esta prueba funcional se ha creado un ejercicio con el nombre “Prueba corrección automática “. A continuación, en la Ilustración 46 se muestra su página web. El usuario puede contestar todas las preguntas o no, no es obligatorio responder la totalidad. También, si el usuario decide cancelar la realización de la prueba sin enviar respuesta alguna al servidor, puede clicar el botón cancelar.

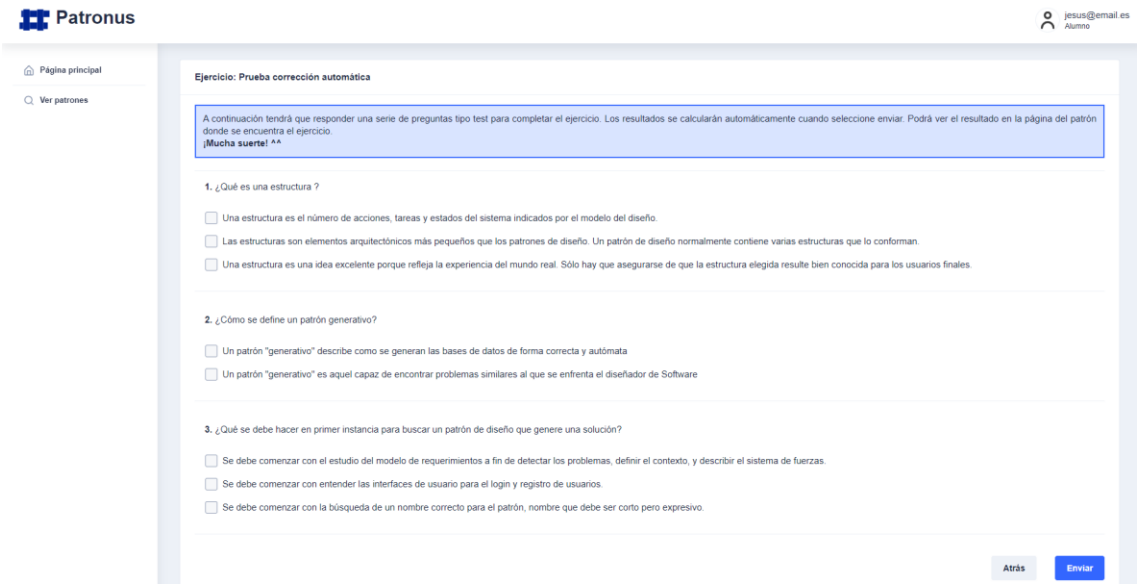

Ilustración 46 - Realizar un ejercicio

Cuando el usuario ha completado la prueba y enviado sus respuestas al servidor, se le redirigirá de nuevo a la lección del patrón. El sistema ya habrá calculado su nota automáticamente y se mostrará en la tabla de ejercicios como se muestra en la Ilustración 47.

Nombre	Nota	Nº realizado	Intentos	Fecha creación
Prueba corrección automática	6.67	2	100	02/07/2022

Ilustración 47 - Tabla de ejercicios mostrando ejercicio completado

6.2.3 Ver estadísticas de un ejercicio

La generación de gráficas para controlar el progreso de los alumnos es una funcionalidad reservada solo para profesores. Para poder realizarla, el profesor debe de entrar en la interfaz de administración de ejercicios, y hay dos rutas posibles para hacerlo. El usuario puede seleccionar del menú izquierdo el enlace “Administrar ejercicios” o del botón que se encuentra en el centro de la pantalla del mismo nombre, como muestra la Ilustración 48.

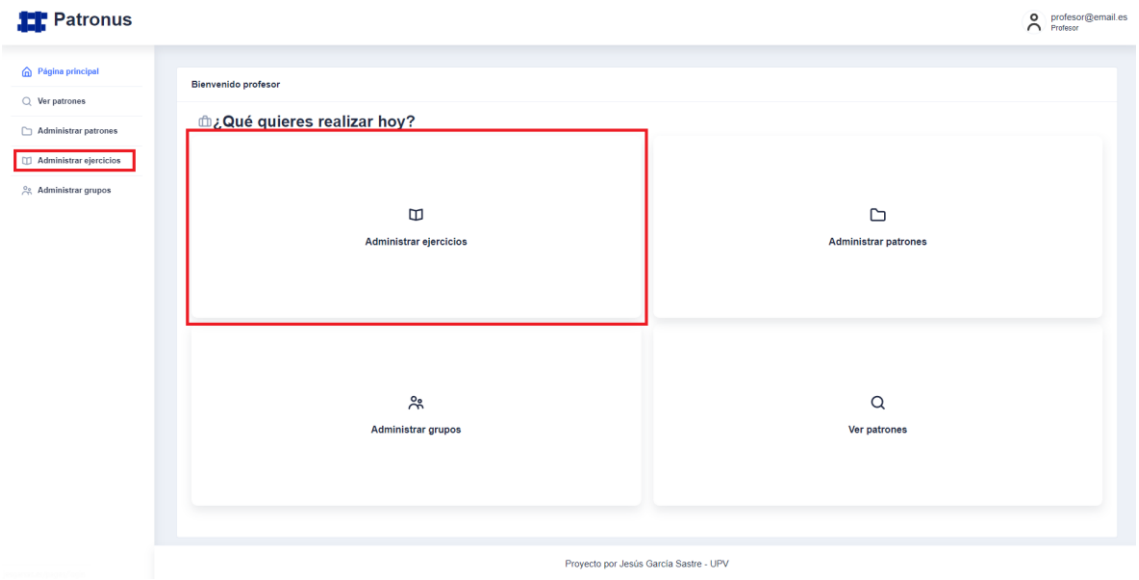

Ilustración 48 - Página principal del profesor

La página web que se muestra al clicar uno de los dos botones es la Ilustración 49, que es la administración de ejercicios. El usuario tiene filtros de búsqueda para encontrar de una forma más eficiente el ejercicio que desee. Una vez encontrado el ejercicio, para poder visualizar las estadísticas hay que clicar el botón triangular celeste.

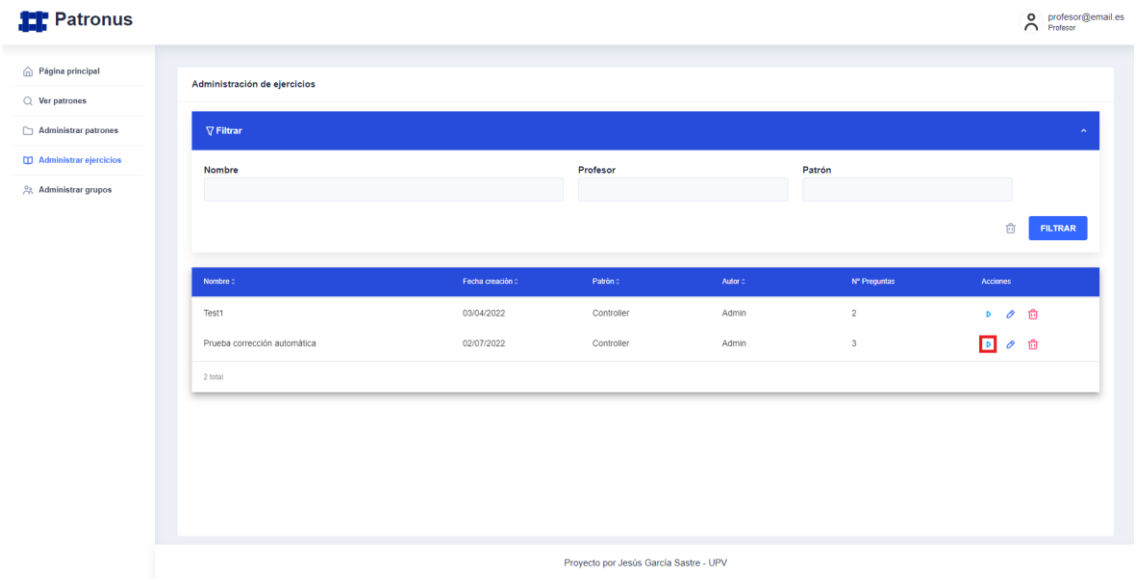

Ilustración 49 - Administración de ejercicios

El botón redirigirá al usuario a la siguiente interfaz de la Ilustración 50, mostrando los resultados de los grupos del profesor que ha iniciado sesión. Los resultados se muestran en una gráfica circular con tres series: aprobados, suspendidos y no realizados.

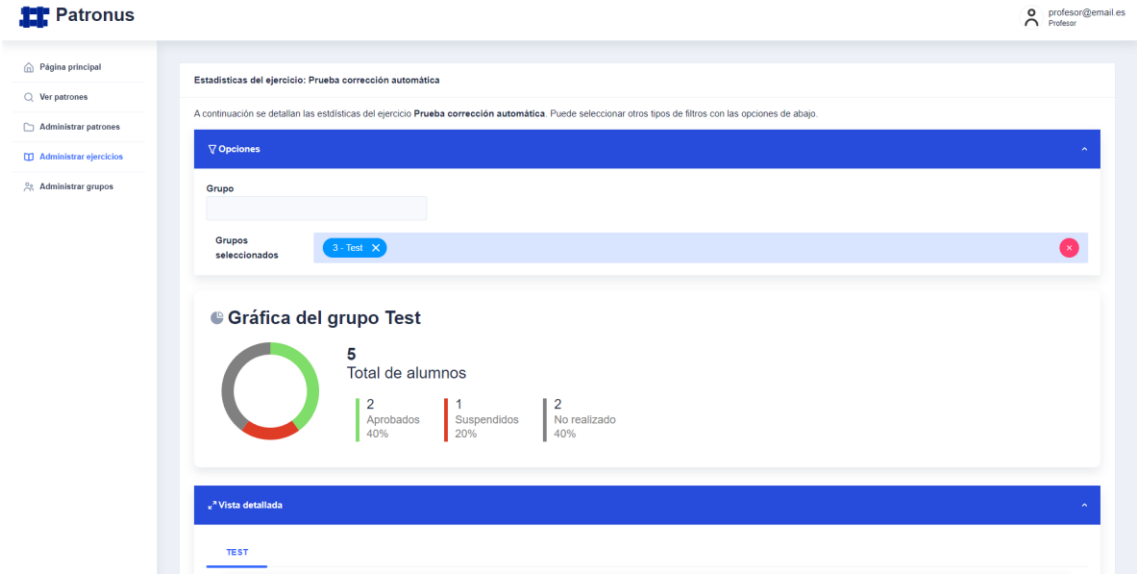

Ilustración 50 - Estadísticas de un ejercicio

La interfaz también ofrece una vista detallada de las respuestas de los alumnos según el grupo. Consiste en una tabla con todos los alumnos, sus notas, fecha de ejecución y si han

respondido o no correctamente cada pregunta del ejercicio, como se observa en la Ilustración 51.

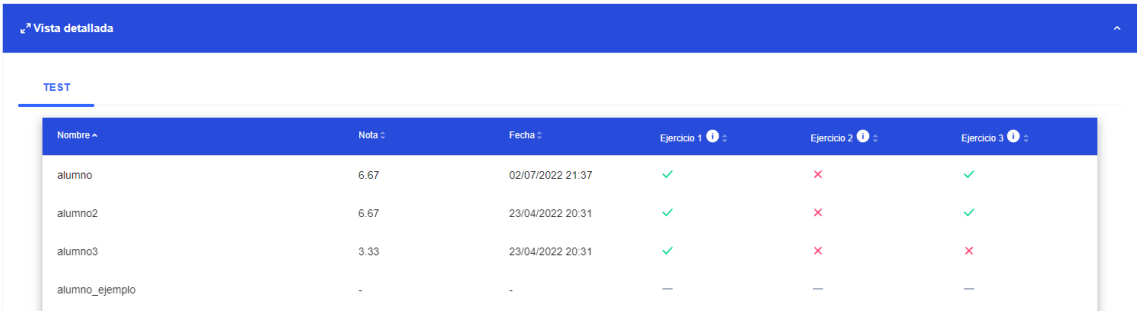

Ilustración 51 - Respuestas de los alumnos

7 Implantación

En esta sección se ha procedido a desplegar la aplicación, tanto la página web, la API Rest y la base de datos relacional en una única máquina virtual. Se han barajado varios distribuidores de máquinas virtuales como Amazon o Microsoft. Finalmente se ha optado por la oferta de Microsoft Azure, porque da un crédito para estudiantes en su plataforma.

El primer paso para la implantación es la compilación de las aplicaciones. Estas van a compilarse en archivos ejecutables *war*s, un tipo de aplicación web ejecutable por los programas basados en Apache Tomcat. Los proyectos Angular y Spring ya ofrecen herramientas para poder compilar a este tipo de archivo.

El segundo paso es elegir el programa que funcionará como servidor en nuestra máquina virtual. Como requisito no funcional, la aplicación ha de ser ejecutada en un servidor basado Tomcat. En el mercado hay diferentes opciones, la elegida por conocimiento previo en su utilización y gestión es Wildfly.

A continuación, se ha contratado una máquina virtual a Microsoft Azure de dos GB de memoria RAM con 2.44 GHz de procesador con una distribución basada en Linux, Ubuntu 18.04 LTS. El proceso de configuración es muy fácil y el sistema te guía durante ella.

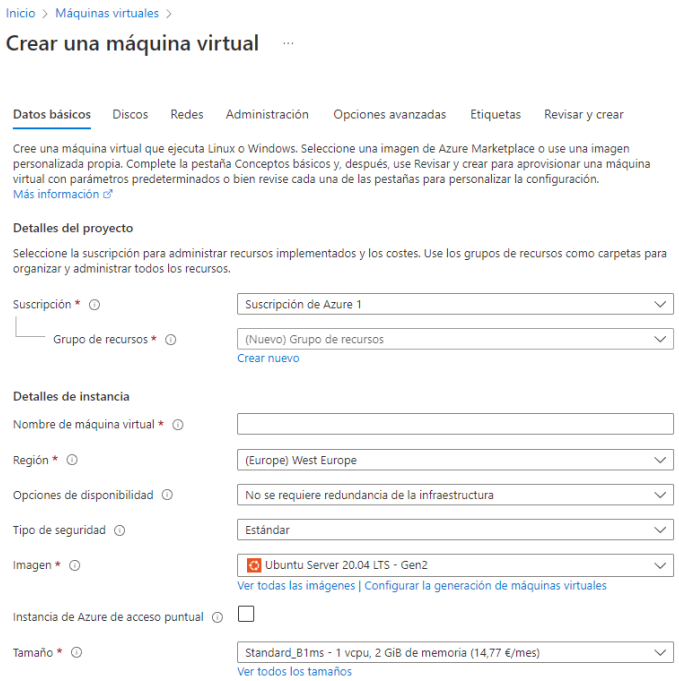

Ilustración 52 - Configuración de la máquina virtual

En la Ilustración 52 se puede observar que la configuración consta de varios pasos, no solo se selecciona el sistema operativo y las especificaciones de la máquina virtual. También se ha seleccionar conceptos como red interna, copias de seguridad, puertos, registro de DNS.

Después de la instalación del contendor, se ha pasado a instalar la base de datos de Oracle Database y migrar el esquema de datos desde el entorno local de desarrollo a la nueva base de datos. El programa DBeaver ofrece la posibilidad de copiar esquemas entre bases de datos, se ha utilizado para acelerar el despliegue en producción.

Por último, se ha instalado Wildfly en la máquina para hospedar las dos aplicaciones del sistema. El proceso de instalación ha necesitado de una pequeña configuración de los puertos en el panel de control de Microsoft Azure para dar acceso a los usuarios a través de internet.

En la Ilustración 53 se puede visualizar la consola de administrador que provee Wildfly para el despliegue de aplicaciones y configuración del servidor. También ofrece la posibilidad de ver la traza de ejecución de cada una de las aplicaciones.

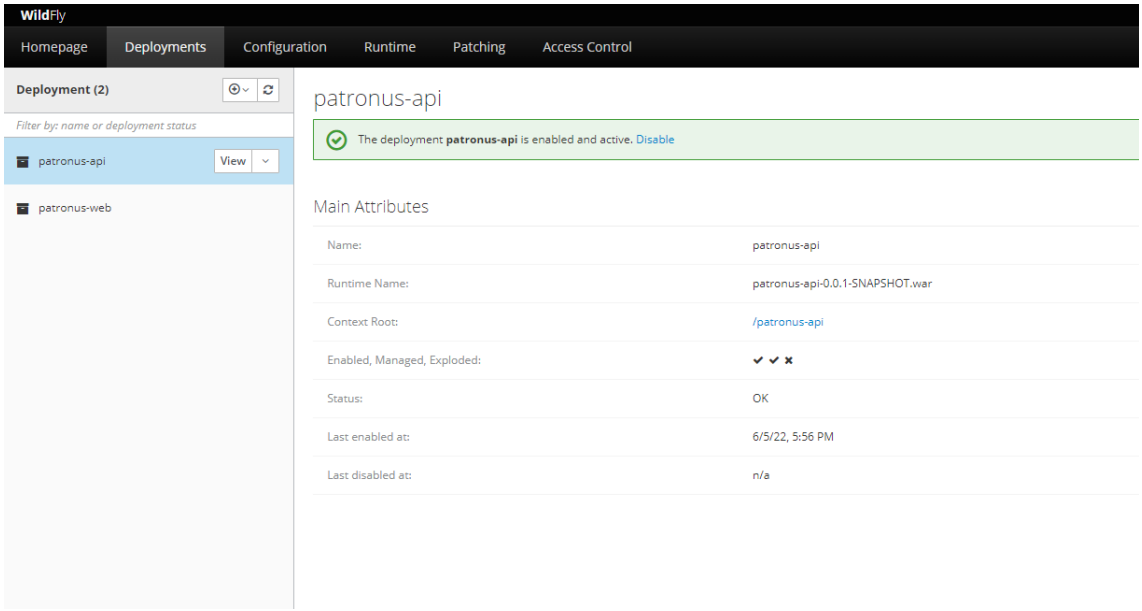

Ilustración 53 - Panel de control de Wildfly

Por concluir, se ha de comprobado que el despliegue funcione y se visualice correctamente. Para poder acceder a la aplicación se ha de acceder mediante un navegador web a la dirección:

http://jesgarsas.es. En la Ilustración 54 podemos visualizar que la aplicación funciona correctamente y el servidor se puede acceder desde fuera de la máquina virtual.

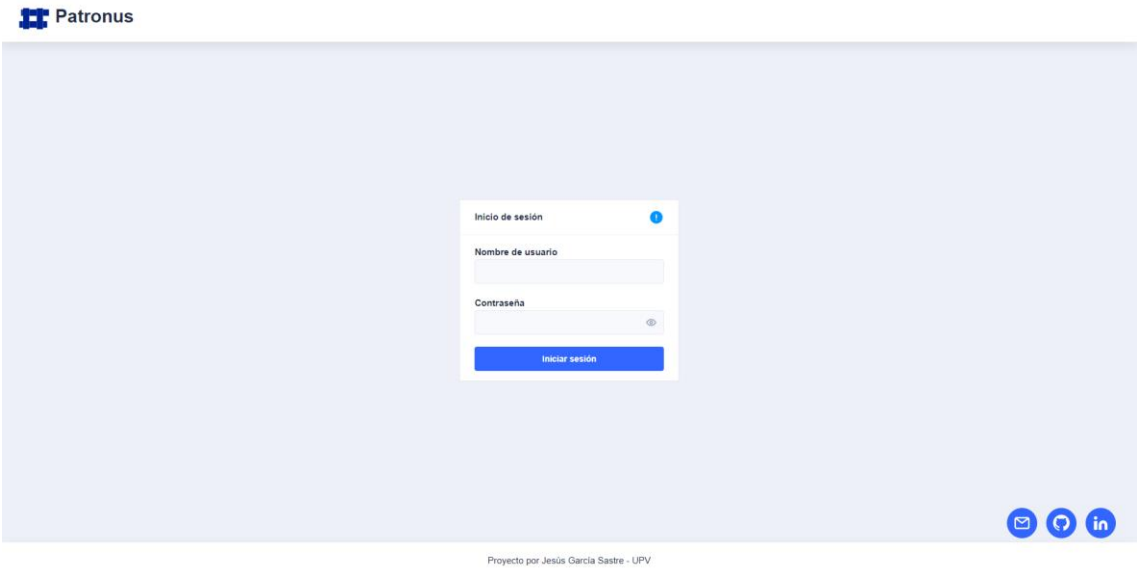

Ilustración 54 - Página de inicio de sesión

8 Conclusiones

Realizando una perspectiva hacia atrás con todos los apartados, se puede afirmar que se han conseguido todos los objetivos propuestos para este proyecto. Se ha desarrollado una aplicación web orientada a la educación de los conceptos clave de patrones de diseño para alumnos y una herramienta educativa para profesores.

8.1 Relación del trabajo desarrollado con los estudios cursados

Durante el desarrollo del proyecto se ha necesitado un conocimiento previo en diversos campos de la informática y también de gestión de proyectos. El alumno ha cursado la rama de ingeniería del software, la cual se centra en el desarrollo de aplicaciones de calidad. El conocimiento conseguido cursando el grado y la rama han sido muy beneficiosos para el proyecto.

En primer lugar, la fase de análisis o planificación ha requerido de utilizar parte de lo enseñado en las asignaturas de gestión de proyectos y proceso del software. Donde se enseño ha como utilizar diagramas de diseño y gestionar el tiempo dentro de un proyecto.

En segundo lugar, la fase de implementación es muy general y los conceptos que se han necesitado pertenecen a un gran conjunto de asignaturas. Por ejemplo, para entender el funcionamiento e implementación de las API Rest se ha utilizado las asignaturas de concurrencia y sistemas distribuidos y redes. Otro buen ejemplo es el conocimiento previo de los patrones que es el objetivo del proyecto que se relaciona directamente con la asignatura patrones de diseño.

En tercer lugar, el uso de JUnit para crear pruebas unitarias para comprobar automáticamente el correcto funcionamiento de la aplicación, se atribuye a la asignatura de lenguajes, tecnologías y paradigmas. Y no solo el uso de la librería, incluse el haber utilizado una arquitectura basa en tres capas.

Por finalizar, el despliegue de la aplicación no se puede vincular a los estudios cursados. Se pueden vincular algunos conceptos durante la instalación de Wildfly a las prácticas de sistemas operativos.

9 Trabajos futuros

Durante el desarrollo del proyecto se han dejado eliminado características que hubiesen mejorado notablemente el rendimiento de la aplicación. También surgieron ideas que se escapaban del alcance del proyecto:

- **Estilo de la página nocturno:** en la actualidad las personas dedican mucho tiempo a estar delante de las pantallas. Los horarios de uso suelen variar, pero son cada vez más las personas que usan ordenadores y teléfonos móviles de noche. Implementar una hoja de estilos para la página con colores oscuros, ayudaría a relajar la vista de estos usuarios y reducir su fatiga visual.
- **Diferentes idiomas:** sería interesante disponer de un sistema de idiomas para poder visualizar la página web y sus lecciones en otros idiomas, ya sean escritos por los profesores o traducidos automáticamente por servicios de terceros como Google Translate.
- **Mejora de los perfiles de los alumnos:** los alumnos actualmente no disponen herramientas para modificar completamente su perfil dentro de la aplicación. Solo pueden cambiar su contraseña. Mejoraría mucho la satisfacción del usuario poder cambiar su correo o usuario sin pedirlo a los profesores.

10 Bibliografía

[1] Educativa, Edutec Revista Electrónica de Tecnología, «¿Pueden las aplicaciones educativas de los dispositivos móviles ayudar al desarrollo de las inteligencias múltiples?,» [En línea]. Available: https://www.edutec.es/revista/index.php/edutec-e/article/view/63. [Último acceso: 11 01 2022].

[2] Universidad Politécnica de Valencia, «PoliformaT - Ayuda,» [En línea]. Available: https://wiki.upv.es/confluence/pages/viewpage.action?pageId=476577803. [Último acceso: 11 01 2022].

[3] RiuNet, «Aplicación de patrones de diseño para la resolución de problemas de software en el desarrollo de un aplicación móvil iOS,» [En línea]. Available: https://riunet.upv.es/handle/10251/69088. [Último acceso: 01 02 2022].

[4] IEEE Std. 830-1998, «Especificación de requisitos según el estándar,» [En línea]. Available: https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf. [Último acceso: 2022 01 14].

[5] Wikipedia.org, «Software design pattern,» [En línea]. Available: https://en.wikipedia.org/wiki/Software_design_pattern. [Último acceso: 11 03 2022].

[6] Wikipedia.org, «MD5,» [En línea]. Available: https://es.wikipedia.org/wiki/MD5. [Último acceso: 2022 01 14].

[7] CloudFlare, «¿Qué es un ataque DDoS?,» [En línea]. Available: https://www.cloudflare.com/es-es/learning/ddos/what-is-a-ddos-attack/. [Último acceso: 12 04 2022].

[8] Profile, «Los 10 principios de usabilidad de Jakob Nielsen: be user friendly,» [En línea]. [Último acceso: 12 04 2022].

[9] A. A. Cabrera, «Capas, cebollas y colmenas: arquitecturas en el backend,» [En línea]. [Último acceso: 11 05 2022].

Glosario

Patrón: es una técnica para resolver un problema de diseño de programación, que debe de cumplir ciertas características: ser reutilizable y fácil de aprender entre otras.

Sprint: periodo de tiempo definido, en el cual se propone realizar ciertas tareas dentro de un proyecto. No es necesario completar todas las tareas, porque los *sprints* son iterativos, una vez finalizado uno se empieza el siguiente con las tareas no finalizadas.

Clean code: es un convenio entre programadores, que recoge varios principios para permitir que todo código sea legible y mantenible para futuros desarrolladores.

Include: tipo de relación entre dos casos de uso, en el cual el segundo es parte esencial en la funcionalidad del primero. Sin el segundo, el primero no podría realizarse correctamente.

Disparador: acción o evento que desencadena el comienzo de una acción determinada siempre.

Super usuario: tipo de usuario que generalmente tiene todos los permisos o acciones disponibles en un sistema informático, incluido aquellas funciones que no pueden realizar usuarios básicos.

Software: conjunto de componentes lógicos que son requeridos para hacer posibles tareas específicas como por ejemplo un programa de gestión de cuentas de ahorro.

Framework: es un marco de trabajo que estandariza un conjunto de conceptos y buenas prácticas, también ofrece librerías para el desarrollo de software.

Pruebas unitarias: conjunto de acciones para comprobar el correcto funcionamiento de una acción especifica en un sistema.

Suite test: conjunto de pruebas unitarias que están relacionadas por la acción que comprueban.

API Rest: es una interfaz de programación de aplicaciones de transferencia de estado representacional.

Hoja de estilo: conjunto de reglas para dar estilo a componentes de una interfaz.

Paquete: carpeta o forma de agrupación de código.

DDoS: es un tipo de ataque malicioso a los servidores, que se basa en hacer una gran cantidad de peticiones al servidor en intervalos de tiempo muy bajos, denegando así el servicio por la saturación del servidor.

Anexo

Objetivos de desarrollo sostenible

En este anexo se hace una reflexión de los objetivos de desarrollo sostenible con el TFG.

Los líderes mundiales adoptaron un conjunto de objetivos globales el 25 de septiembre de 2015 para proteger el planeta, erradicar el hambre en el mundo, que exista la igualdad para todos como un plan de acción para un desarrollo sostenible. Cada objetivo ha de cumplirse antes de 15 años.

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**	**X**			
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**				**X**
ODS 9. **Industria, innovación e infraestructuras.**				**X**
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**			**X**	
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

El proyecto está relacionado solo con dos objetivos del plan de desarrollo sostenible y son:

- **Educación de calidad:** el proyecto está orientado a mejorar la educación de los alumnos, y en mi opinión la aplicación puede ayudar a formar a muy buenos profesionales en el futuro sin coste alguno. Toda persona dispuesta aprender, puede encontrar aquí una fuente conocimiento creada por profesores expertos en el tema. Es una relación directa.
- **Producción y consumo responsables:** el objetivo está indirectamente relacionado con el TFG, porque la aplicación intenta evitar que los alumnos fotocopien apuntes de los libros o del profesorado, no malgastando así muchas hojas de papel. Haciendo un poco de reflexión y opinión propia, si de media en una clase hay cuarenta alumnos y los apuntes suelen tener veinte hojas, por clase de media se malgastarían seis cientas hojas de papel. El mundo debería reducir el uso de hojas de papel en apuntes que ya tienen en internet.