HackerKid

Nombre Estudiante
María Gallego García

Nombre Consultor
Oriol Martí Girona
Data Entrega

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Hackerkid*
Nombre del autor:	*María Gallego García*
Nombre del consultor/a:	*Oriol Martí Girona*
Nombre del PRA:	*Santi Caballé Llobet*
Fecha de entrega (mm/aaaa):	06/2019
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Ingeniería del software*
Idioma del trabajo:	***Español***
Palabras clave	*Angular, Scrum, Web, Java*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.*

El objetivo del proyecto es acercar la informática a los más pequeños, para ello se realiza el desarrollo de una aplicación orientada a la docencia que permita la gestión de los contenidos y a su vez el seguimiento de cursos y la ejecución de los mismos.

La temática de la aplicación es la informática, proveerá unos cursos iniciales, al ser configurable posteriormente se podría emplear en más materias.

La metodología a seguir a lo largo del proyecto ha sido Scrum, con la idea de agilizar el proceso de desarrollo.

El resultado es una aplicación web desarrollada en Angular, con un diseño intuitivo y responsivo, que proporcionará al usuario una buena experiencia en el uso y además podrá usarse en distintos dispositivos.

Abstract (in English, 250 words or less):

The goal of the project is to bring the computer science to the little ones, for this the development of an application oriented to the teaching is made that allows the management of the contents and in turn the follow-up of courses and the execution of the same ones.

The subject of the application is computer science, it will provide some initial courses, being configurable later it could be used in more subjects.

The methodology to follow throughout the project has been Scrum, with the idea of streamlining the development process.

The result is a web application developed in Angular, with an intuitive and responsive design, which will provide the user with a good experience in the use and can also be used in different devices.

Índice

Índice de figuras

1 Descripción del sistema

1.1 Descripción del proyecto

El proyecto que se desea desarrollar tiene como objetivo la realización de una aplicación educativa destinada al apoyo a la docencia, para ello dispondrá de diversas opciones de aprendizaje mediante distintos módulos temáticos.

La aplicación constará de una interfaz de administración que permitirá gestionar tanto el contenido como los usuarios que podrán acceder y una interfaz de estudiante que permitirá el acceso al contenido, realización de ejercicios, edición del perfil y mostrará el progreso y los puntos conseguidos.

Incluirá aspectos de accesibilidad y usabilidad, dado que el objetivo es alcanzar al mayor número de usuarios posibles.

1.2 Definición del alcance

1. Objetivos del proyecto
Qué se pretende con la implantación del proyecto
• Acercar la informática a los niños mediante una aplicación amigable con información básica acerca de conceptos informáticos y salidas profesionales. • Proporcionar a los padres, madres o tutores legales acceso a actividades para realizar • Proporcionar a las administraciones una herramienta de apoyo para la docencia.
Cómo se consigue
• Desarrollando una aplicación que contenga unidades didácticas y/o juegos educativos.
Factores críticos de éxito del proyecto
• Terminar el proyecto antes del próximo inicio de curso • Robustez de procesamiento y consistencia.
Razón y oportunidad del proyecto
• Escasez de proyectos de este tipo en el sistema educativo

2. Requisitos y características del proyecto	
Código	Descripción
R001	Facilitar un diseño usable e intuitivo
R002	Diseñar de cara a la accesibilidad
R003	Gestionar los contenidos desde la aplicación

R004	Mostrar distintas opciones en función del usuario
R005	Mostrar el progreso del usuario
R006	Tener capacidad para añadir contenido nuevo
R007	Diseño responsivo

3. Criterios de aceptación	
Código	Descripción
C001	No hay errores críticos en el sistema
C002	Diseño intuitivo
C003	Navegabilidad correcta en la web
C004	Se cumplen los requisitos en un 90%

1.3 Registro de interesados

Los principales *stakeholders* de aplicación son:

- **Niños en edad escolar**: son los principales usuarios de la aplicación, tendrán acceso a los contenidos.
- **Padres, madres o tutores legales**: tienen acceso a la zona de configuración de la aplicación y pueden gestionar el contenido
- **Profesorado**: podrán utilizar la aplicación como complemento a unidades didácticas, podrán editar y gestionar el contenido.
- **Administración**: podrán distribuir la aplicación en función de las necesidades educativas

2 Metodología

El proyecto seguirá la metodología Scrum, se dispondrá de un backlog actualizado que incluirá los requisitos en las historias de usuario. Estas historias se estimarán antes de cada Sprint y también se refinarán los requisitos.

Dado que lo ideal es que las historias de usuario sean pequeñas se dividirán, cuando aplique, en análisis, configuración, implementación en el Front-End, implementación del Back-end, integración, test del Front-End y test del Back-End, otras. La prioridad irá en ese orden, que será en el cual serán cogidas.

Se dispondrá de un tablero Kanban para seguir el estado de las historias de usuario, con al menos tres estados, ‘To Do’, ‘In Progress’ y ‘Done’, siendo el primero dónde se encuentran las historias por realizar, el segundo las que están en curso, preferiblemente una sola y en el último las que ya se han realizado.

2.1 Backlog inicial

Dispondrá de las historias iniciales y se irá incrementando en función de las necesidades que vayan surgiendo, problemas o requerimientos durante el sprint.

Sprint 1

Goals:

- Elección de tecnologías, entorno, configuración: cubierta en historias US0001, US0002, US0003
- Página de login: cubierta con la historias US0004

Historias de usuario iniciales			
Código	Título	Descripción	Prioridad
US0001	Análisis del entorno	Como usuario quiero una documentación inicial del sistema y diseño inicial de la configuración de entorno para iniciar la configuración	1
US0002	Configuración inicial	Como cliente quiero disponer de una configuración inicial del entorno para iniciar los desarrollos	2
US0003	Login - Análisis	Como usuario quiero disponer de seguridad en el inicio de sesión	3
US0004	Login – Implementación en FE	Como usuario quiero iniciar sesión para acceder al contenido	4

3 Planificación

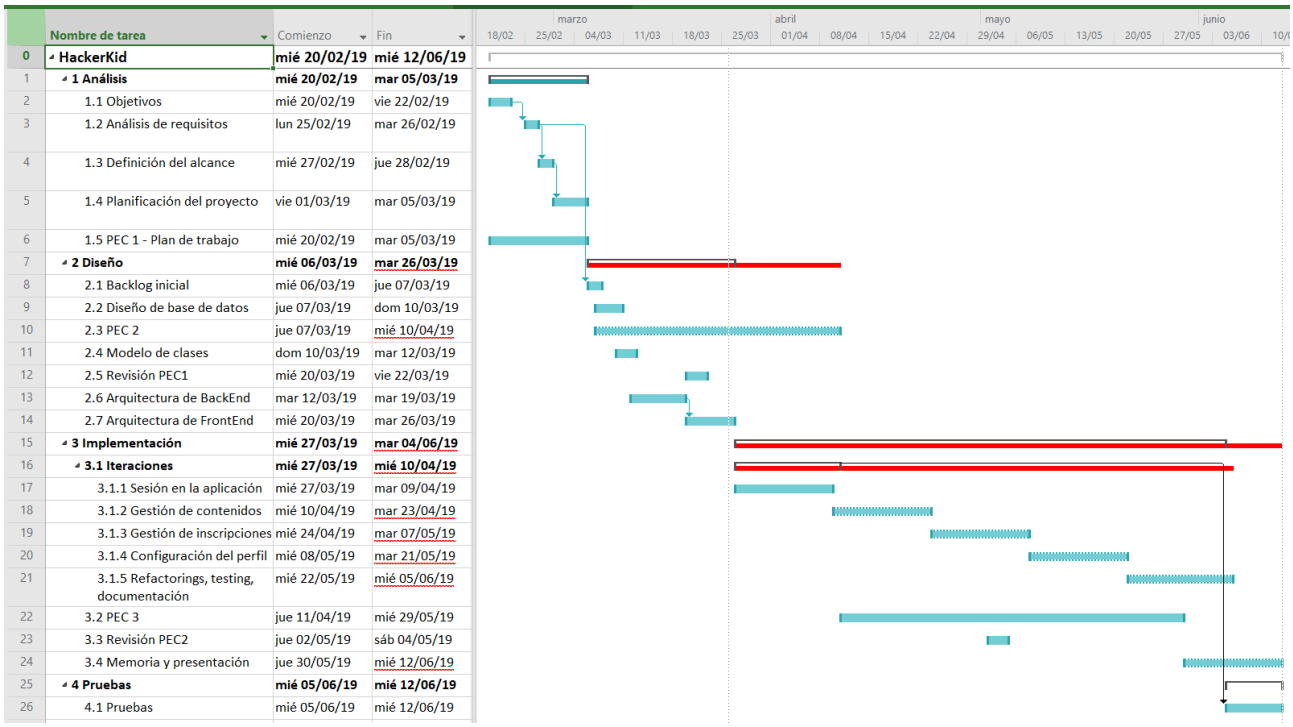
El proyecto estará dividido en varias etapas, Análisis, Diseño, Implementación y Pruebas, cada una de ellas subdividida en sus distintos apartados.

La fase de análisis constará de la definición de los objetivos, el análisis de requisitos, la definición del alcance, la planificación del proyecto y la entrega del plan de trabajo. Esta fase está estimada en dos semanas de duración.

En la fase de diseño se incorporará el backlog inicial, con las primeras historias de usuario estimadas, el diseño de la base de datos, la entrega de la PEC2, el modelo de clases, se reservará asimismo tiempo para la revisión del feedback de la entrega de la PEC1 y posteriormente el diseño de la arquitectura tanto de BackEnd como de FrontEnd.

La fase de implementación constará de cinco iteraciones, de dos semanas cada una divididas en sus tareas de elección de backlog, especificación de las tareas, diseño e implementación:

- Sesión en la aplicación: En esta iteración se decidirá e implementará la gestión de los usuarios del sistema, así como su interfaz y forma de acceso.
- Gestión de contenidos: Constará del diseño e implementación de los contenidos de la aplicación; gestión de las materias, temas y ejercicios.
- Gestión de inscripciones: Se incorporará a la aplicación la opción de inscribir a un usuario a una materia, esta aparecerá en el dashboard del usuario
- Configuración del perfil: Se definirá que elementos serán editables según el usuario y se proporcionará una interfaz que muestra los datos y opciones del usuario con respecto al perfil.
- Refactorings, testing y documentación: Fase reservada a mejoras sobre lo ya implementado, el valor añadido de esta fase estará orientado a la robustez del sistema y a la presentación del proyecto.



	Nombre de tarea	Comienzo	Fin
0	HackerKid	mié 20/02/19	mié 12/06/19
1	1 Análisis	mié 20/02/19	mar 05/03/19
2	1.1 Objetivos	mié 20/02/19	vie 22/02/19
3	1.2 Análisis de requisitos	lun 25/02/19	mar 26/02/19
4	1.3 Definición del alcance	mié 27/02/19	jue 28/02/19
5	1.4 Planificación del proyecto	vie 01/03/19	mar 05/03/19
6	1.5 PEC 1 - Plan de trabajo	mié 20/02/19	mar 05/03/19
7	2 Diseño	mié 06/03/19	mar 26/03/19
8	2.1 Backlog inicial	mié 06/03/19	jue 07/03/19
9	2.2 Diseño de base de datos	jue 07/03/19	dom 10/03/19
10	2.3 PEC 2	jue 07/03/19	mié 10/04/19
11	2.4 Modelo de clases	dom 10/03/19	mar 12/03/19
12	2.5 Revisión PEC1	mié 20/03/19	vie 22/03/19
13	2.6 Arquitectura de BackEnd	mar 12/03/19	mar 19/03/19
14	2.7 Arquitectura de FrontEnd	mié 20/03/19	mar 26/03/19
15	3 Implementación	mié 27/03/19	mar 04/06/19
16	3.1 Iteraciones	mié 27/03/19	mié 10/04/19
17	3.1.1 Sesión en la aplicación	mié 27/03/19	mar 09/04/19
18	3.1.2 Gestión de contenidos	mié 10/04/19	mar 23/04/19
19	3.1.3 Gestión de inscripciones	mié 24/04/19	mar 07/05/19
20	3.1.4 Configuración del perfil	mié 08/05/19	mar 21/05/19
21	3.1.5 Refactorings, testing, documentación	mié 22/05/19	mié 05/06/19
22	3.2 PEC 3	jue 11/04/19	mié 29/05/19
23	3.3 Revisión PEC2	jue 02/05/19	sáb 04/05/19
24	3.4 Memoria y presentación	jue 30/05/19	mié 12/06/19
25	4 Pruebas	mié 05/06/19	mié 12/06/19
26	4.1 Pruebas	mié 05/06/19	mié 12/06/19

Figura 1: Planificación del proyecto

4 Implementación

La aplicación se desarrollará en Angular 7+ con Material design en el lado cliente, que proporcionará buen rendimiento y escalabilidad.

El back-end se implementará en Java 1.8+, dispondrá de recursos REST desde los que se comunicará el cliente, el uso de servicios REST hace que sea independiente de la plataforma o lenguaje.

La base de datos se creará con el motor SQLite, por ser liviano y de rápido acceso.

La aplicación constará del mínimo de librerías externas.

La aplicación constará de un repositorio Git, para ello se dispondrá de una cuenta de GitHub asociada al repositorio local.

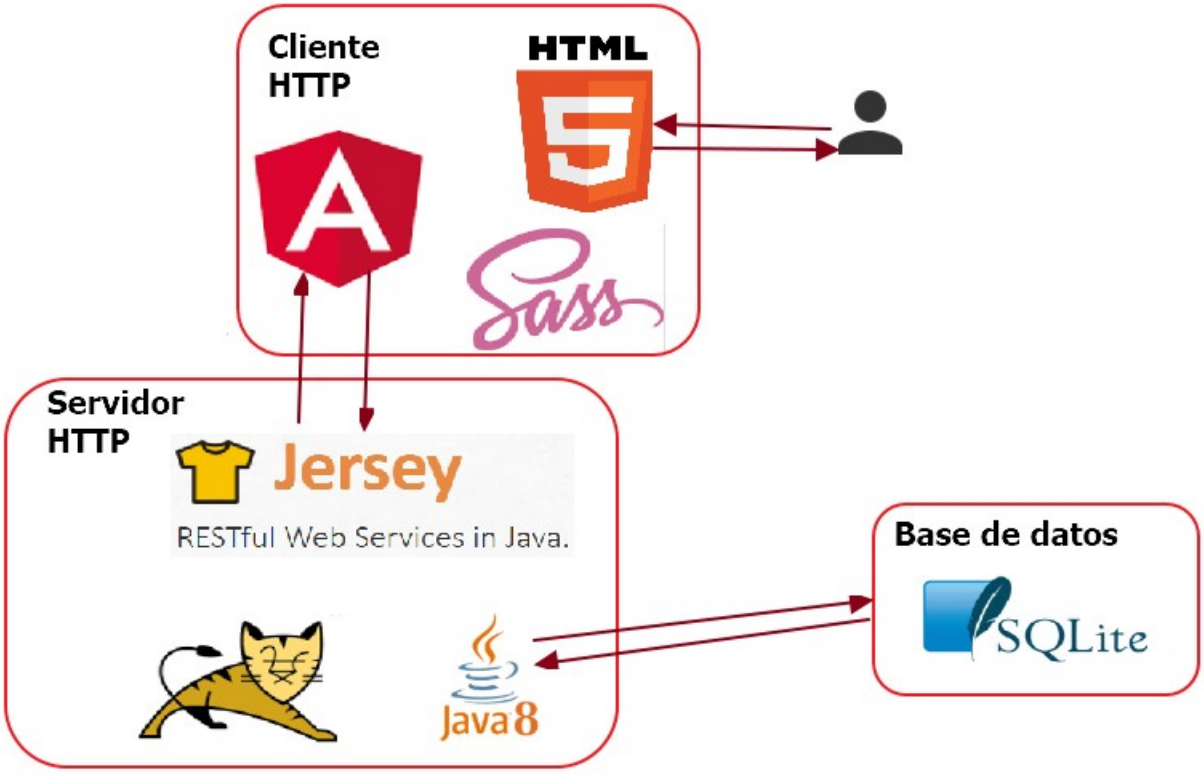


Figura 2: Arquitectura de la aplicación

5 Especificación del sistema

5.1 División en subsistemas

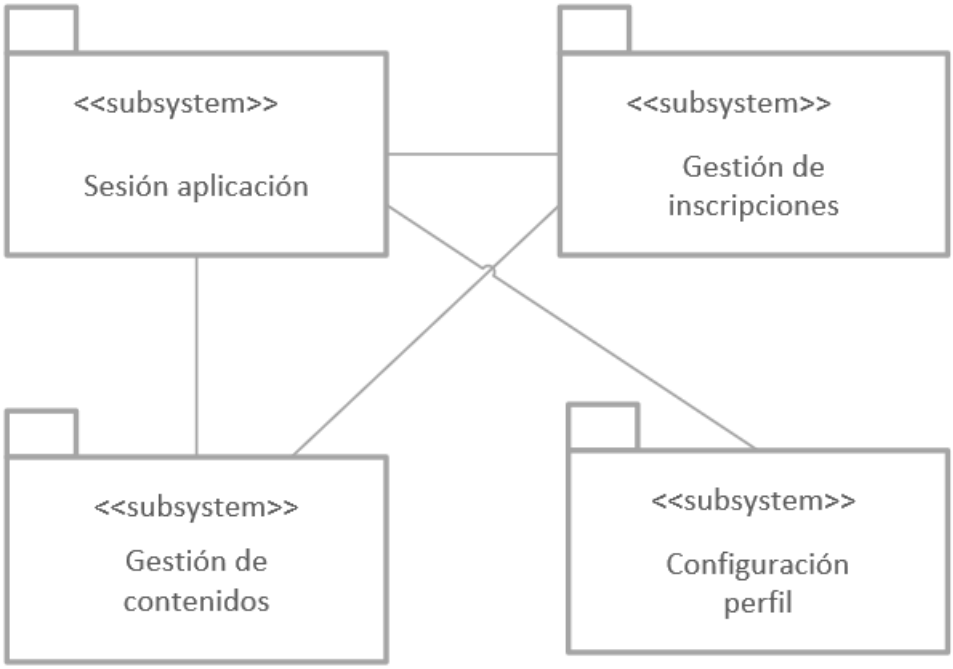


Figura 3: Subsistemas

Subsistema	Función
Sesión en la aplicación	Subsistema que permitirá el acceso a la aplicación y la gestión de los usuarios del sistema.
Gestión de contenidos	Subsistema que se ocupa de la administración del contenido de la aplicación, materias, teoría, preguntas y respuestas.
Gestión de inscripciones	Subsistema encargado de la inscripción de los estudiantes en las materias.
Configuración del perfil	Subsistema que permitirá la configuración del perfil privado del estudiante. Inicialmente el caso básico, ampliable a futuro.

6 Especificación de subsistemas

6.1 Subsistema “Sesión en la aplicación”

6.1.2 Modelo de casos de uso

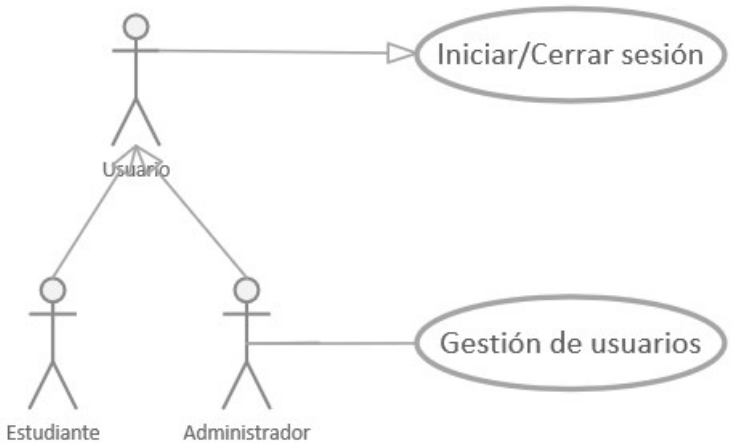


Ilustración 1: CU: Sesión en la aplicación

6.1.2 Descripción de casos de uso

6.1.2.1 Objetivo

La función de la sesión en la aplicación es permitir que los usuarios puedan acceder y salir de la aplicación. Además, el usuario administrador podrá crear, borrar, modificar y consultar los usuarios del sistema.

6.1.2.2 Especificación de casos de uso

Caso de uso “Iniciar/ cerrar sesión”
ID: CU 1.1
Objetivo: El objetivo de este escenario es el inicio y cierre de sesión
Quién lo comienza: Cualquier usuario de la aplicación.
Quién lo finaliza: Cualquier usuario de la aplicación.
Descripción: 1. **El usuario comienza la acción accediendo a la web de la aplicación.** 2. **Deberá rellenar los campos: Usuario y Contraseña, con los datos que disponga para el inicio de sesión.** 3. **En caso de faltar algún dato, o no estar en el sistema, se mostrará un mensaje de error.**

4. **Si no ha habido ningún problema al pulsar el botón de inicio de sesión redireccionará a la página de Dashboard.**

5. **El usuario finalizará la sesión al pulsar el botón de cierre de sesión.**

Caso de uso "Gestión de usuarios"
ID: CU 1.2
Objetivo: El objetivo de este escenario es la gestión de usuarios (CRUD)
Quién lo comienza: El administrador.
Quién lo finaliza: El administrador.
Descripción: 1. **El administrador comienza accediendo a la aplicación.** 2. **En el Dashboard deberá seleccionar la opción de administrar usuarios.** 3. **Se mostrará la pantalla del listado de usuarios con las opciones de crear, borrar, editar y modificar usuario. Sólo serán editables los campos Usuario y Contraseña.** 4. **El administrador selecciona la acción que quiere realizar.** 5. **El sistema guarda los cambios**

6.1.3 Interfaces de usuario

6.1.3.1 Prototipo de la interfaz

- Pantalla de "Inicio de sesión": Al iniciar la aplicación se abrirá la pantalla de login, esta pantalla pedirá los datos de acceso, usuario y contraseña y tras comprobarlos abrirá la pantalla del Dashboard.

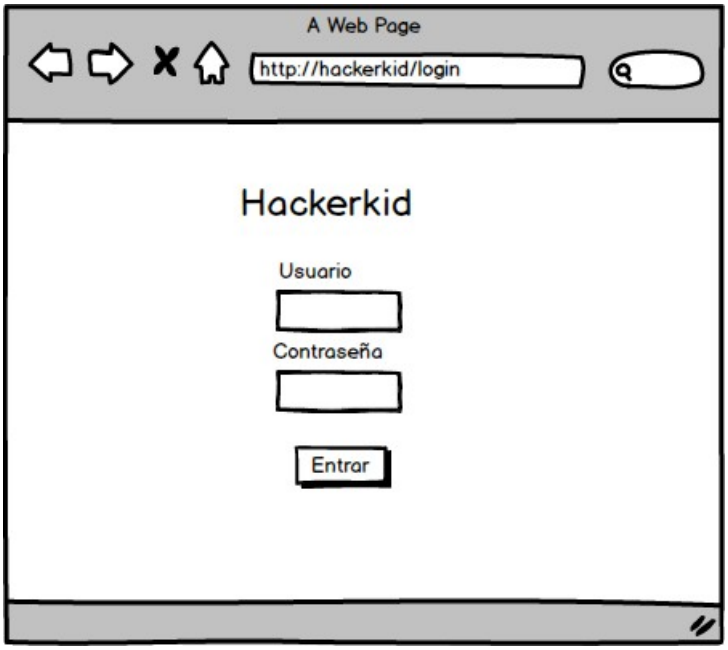


Ilustración 2: Pantalla Inicio de sesión

- Pantalla “Listado de usuarios”: Mostrará la lista de usuarios con los datos que disponemos, nombre de usuario, nombre y apellidos en caso de que los proporcione y puntos si ha iniciado alguna de las actividades que los proporcionan. El listado será editable, permitirá añadir, modificar y borrar usuarios, el administrador no tendrá puntos asignados, los demás por defecto 0.

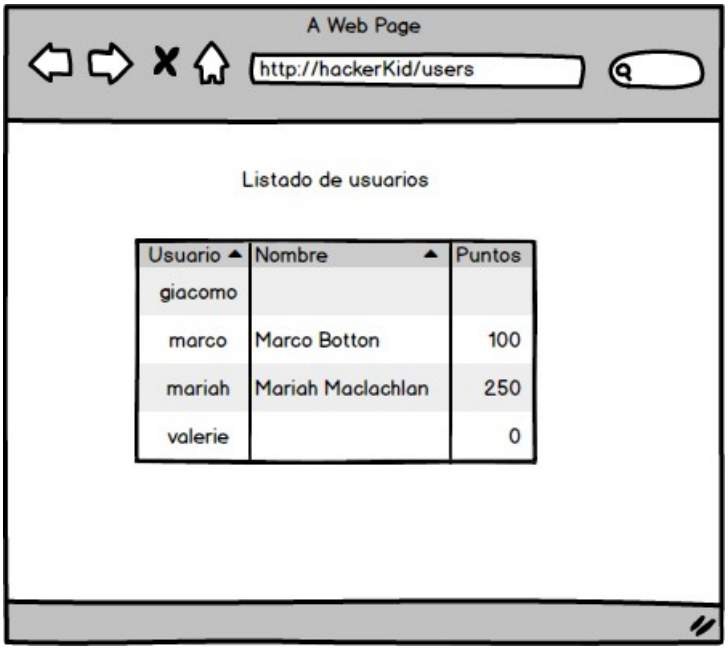


Ilustración 3: Pantalla Listado de usuarios

6.2 Subsistema “Gestión de contenidos”

6.2.1 Modelo de casos de uso

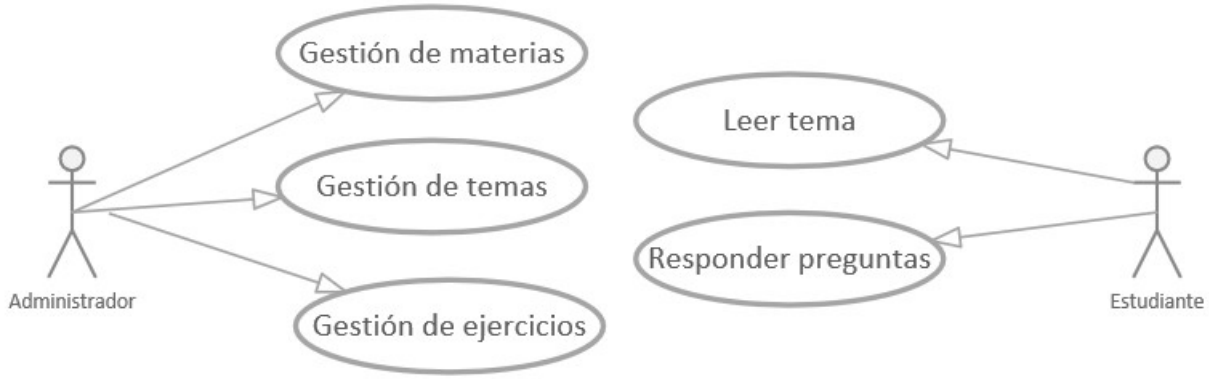


Ilustración 4: CU: Gestión de contenidos

6.2.1.1 Objetivos

Este subsistema tiene como objetivo permitir al administrador añadir el contenido al que tendrán acceso los usuarios.

Otro objetivo será posibilitar que el usuario pueda acceder al contenido al que previamente esté inscrito y realizar las operaciones de lectura de temas, resolución de ejercicios y que tenga disponible la información sobre el progreso.

6.2.1.2 Especificación de casos de uso

Caso de uso "Nueva materia"
ID: CU 2.1
Objetivo: El objetivo de este escenario es añadir una nueva materia.
Quién lo comienza: El administrador inicia la acción.
Quién lo finaliza: El administrador
Descripción: 1. **El administrador comienza la acción accediendo a la web de la aplicación.** 2. **El administrador accede a la configuración del contenido** 3. **Pulsa el botón *Añadir material*** 4. **Se abre un diálogo con la opción de escribir el nombre de la material** 5. **El administrador introduce el nombre y guarda.** 6. **El sistema guarda la nueva materia y muestra en pantalla proporcionando la opción de crear nuevo nivel y de asignar estudiantes.**

Caso de uso "Borrado de una materia"
ID: CU 2.2
Objetivo: El objetivo de este escenario es borrar una materia
Quién lo comienza: El administrador inicia la acción.
Quién lo finaliza: El administrador
Descripción: 1. **El administrador comienza la acción accediendo a la web de la aplicación.** 2. **El administrador accede a la configuración del contenido** 3. **Desde el listado de materias accede en la derecha al menú de la materia a borrar, pincha en el icono de los tres puntos que abre el menú y pulsa el botón borrar.** 4. **El sistema borra la materia y todos sus datos asociados.**

Caso de uso "Gestión de temas"

ID: CU 2.3

Objetivo: El objetivo de este escenario son las operaciones CRUD sobre los temas.

Quién lo comienza: El administrador inicia la acción.

Quién lo finaliza: El administrador

Descripción:

1. **El administrador comienza la acción accediendo a la web de la aplicación.**
2. **El administrador accede a la configuración del contenido**
3. **Selecciona la materia sobre la que se va a añadir el tema**
4. **El administrador pulsa el botón añadir nivel**
5. **El sistema muestra el botón de añadir tema y el de añadir ejercicios**

Caso de uso "Gestión de ejercicios"

ID: CU 2.4

Objetivo: El objetivo de este escenario son las operaciones CRUD sobre las respuestas.

Quién lo comienza: El administrador inicia la acción.

Quién lo finaliza: El administrador

Descripción:

1. **El administrador comienza la acción accediendo a la web de la aplicación.**
2. **El administrador accede a la configuración del contenido.**
3. **Selecciona la materia sobre la cual se va a añadir el ejercicio.**
4. **Pulsa el botón añadir ejercicios, del nivel deseado.**
5. **El sistema muestra la pantalla de los ejercicios correspondiente al administrador.**
6. **El administrador inserta el enunciado de la pregunta.**
7. **Pulsando el botón de añadir respuesta, repetidas veces, añade las respuestas a la pregunta, la verdadera y varias falsas.**
8. **El administrador marca una de las respuestas como la verdadera.**
9. **La acción final al pulsar el botón de guardar.**

Caso de uso "Leer tema"
ID: CU 2.5
Objetivo: El objetivo de este escenario es proporcionar al estudiante la forma de leer el contenido de las materias
Quién lo comienza: El estudiante inicia la acción.
Quién lo finaliza: El estudiante
Descripción: 1. **El estudiante comienza la acción accediendo a la web de la aplicación.** 2. **El sistema mostrará en el Dashboard el progreso de cada materia** 3. **El estudiante selecciona la materia sobre la que quiere leer el tema** 4. **El estudiante completa la lectura** 5. **El sistema guarda los puntos de la lectura del tema** 6. **El estudiante pincha en el botón de volver**

Caso de uso "Responder preguntas"
ID: CU 2.6
Objetivo: El objetivo de este escenario es que el estudiante pueda conseguir más puntos respondiendo a preguntas sobre un tema
Quién lo comienza: El estudiante inicia la acción.
Quién lo finaliza: El estudiante
Descripción: 1. **El estudiante comienza la acción accediendo a la web de la aplicación.** 2. **El sistema mostrará en el Dashboard el progreso de cada materia** 3. **El estudiante selecciona la materia sobre la que quiere realizar los ejercicios** 4. **El estudiante selecciona la opción de los ejercicios** 5. **Contesta correctamente a los ejercicios del tema** 6. **El sistema guarda los puntos de cada pregunta** 7. **El estudiante pulsa el botón volver**

6.2.3 Interfaces de usuario

6.2.3.1 Prototipo de la interfaz

- Pantalla del *dashboard:* dispondrá de un interfaz distinto en función del tipo de usuario. Para el usuario administrador permitirá el acceso a la configuración y a la gestión de los usuarios, mientras que al resto de usuarios les mostrará los módulos a los que pueden acceder, con código de colores para los niveles y texto.

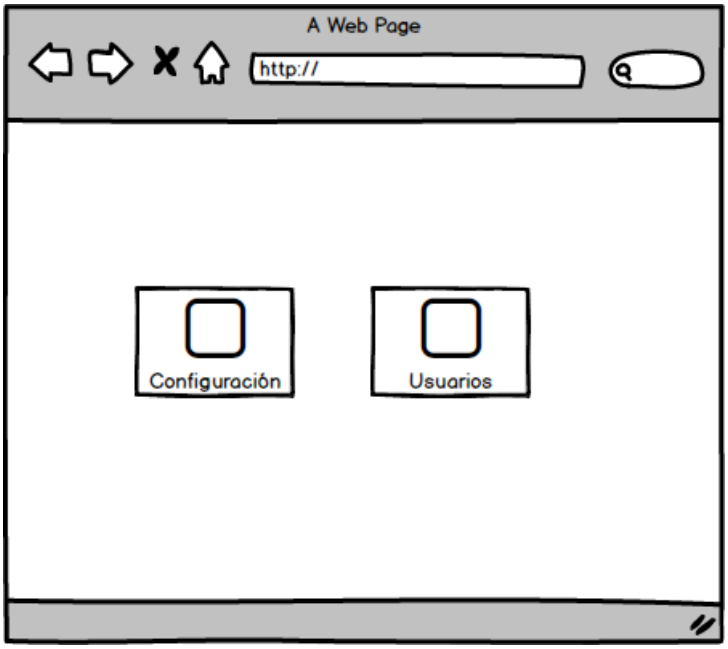


Ilustración 5: Dashboard usuario administrador

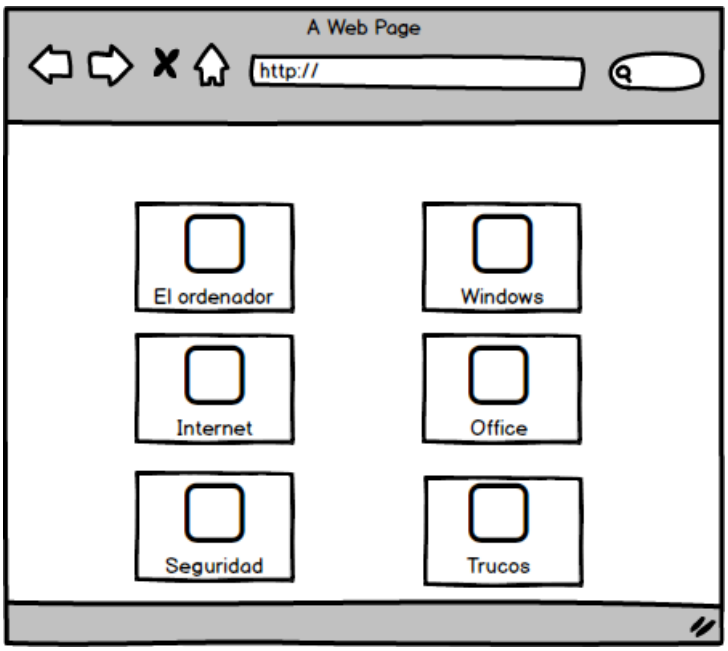


Ilustración 6: Pantalla Dashboard del usuario

- Pantalla Materias: Esta pantalla será desde la que el administrador realizará las operaciones de creación, consulta, modificación y borrado de materias. Cada materia tendrá niveles asociados y cada nivel temas y preguntas.

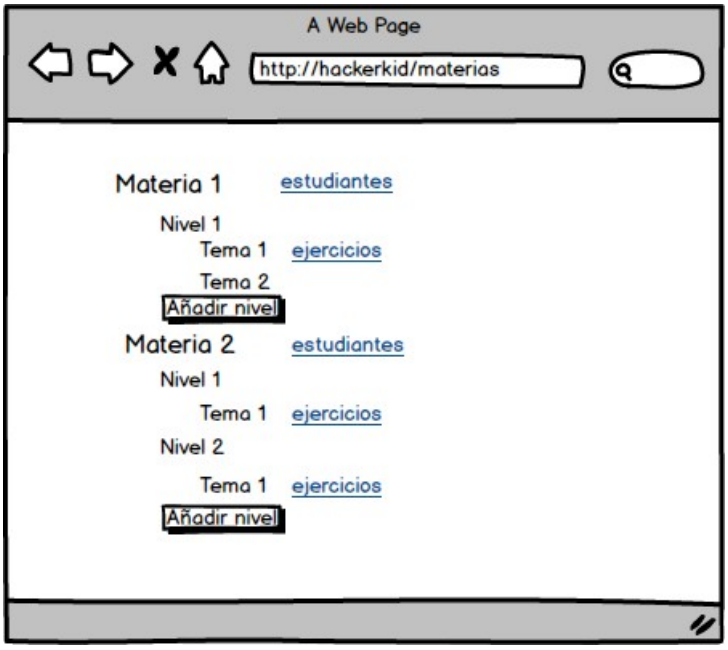


Ilustración 7: Pantalla Materias

6.3 Subsistema “Gestión de inscripciones”

6.3.1 Modelo de casos de uso

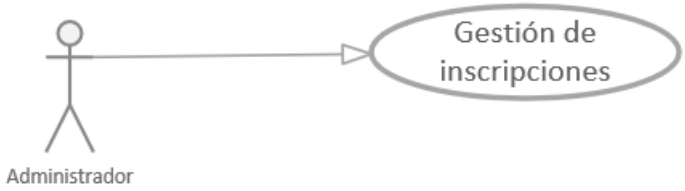


Ilustración 8: CU: Gestión de inscripciones

6.3.1.1 Objetivos

El objetivo de la gestión de inscripciones es poder proporcionar al estudiante acceso a los contenidos y llevar un seguimiento.

6.3.1.2 Especificación de casos de uso

Caso de uso “Inscribir estudiante”
ID: CU 3.1
Objetivo: El objetivo de este escenario es la inscripción de un estudiante a una materia.
Quién lo comienza: El administrador.

Quién lo finaliza: El administrador.

Descripción:

1. **El administrador comienza la acción accediendo a la web de la aplicación.**
2. **Desde el Dashboard de administración selecciona la opción de gestión de contenido.**
3. **El administrador pincha en la materia a la que se quiere inscribir al estudiante.**
4. **En el componente de búsqueda escribirá el nombre del estudiante a asignar.**
5. **El sistema mostrará la lista de estudiantes que no están asignados todavía a esa matería.**
6. **El administrador selecciona el estudiante que quiere añadir**
7. **El sistema guarda la inscripción automáticamente.**

Caso de uso "Inscribir estudiante"
ID: CU 3.2
Objetivo: El objetivo de este escenario es desasignar a un estudiante de una inscripción.
Quién lo comienza: El administrador.
Quién lo finaliza: El administrador.

Descripción:

8. **El administrador comienza la acción accediendo a la web de la aplicación.**
9. **Desde el Dashboard de administración selecciona la opción de gestión de contenido.**
10. **El administrador accede a la materia sobre la que se va a realizar la acción de borrado.**
11. **Desde el apartado *Estudiantes* seleccionará, de entre los estudiantes registrados en el sistema, el estudiante que se va a eliminar pulsando en la equis.**
12. **El sistema guarda los cambios automáticamente.**

6.4 Subsistema “Configuración del perfil”

6.4.1 Modelo de casos de uso

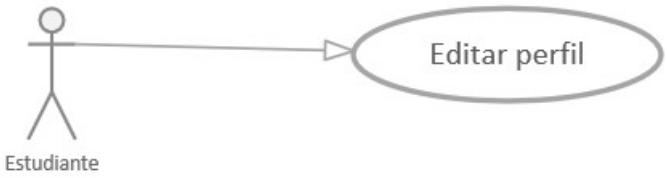


Ilustración 9: CU: Editar perfil

6.4.2 Descripción de casos de uso

6.4.2.1 Objetivo

El objetivo del subsistema de la configuración del perfil es facilitar al usuario estudiante la opción de editar el perfil, incluir el nombre, los apellidos y cambiar la contraseña. Estas acciones sumarán puntos al total de puntos.

6.4.2.2 Especificación de casos de uso

Caso de uso “Configuración inicial del perfil”
ID: CU 4.1
Objetivo: El objetivo de este es la creación del perfil del estudiante
Quién lo comienza: El usuario estudiante.
Quién lo finaliza: El usuario estudiante.
Descripción: 1. **El estudiante comienza la acción al introducir el usuario y la contraseña en la pantalla de login.** 2. **La aplicación muestra un *stepper* con los pasos a seguir para la configuración inicial de la cuenta.** 3. **El usuario introduce el nombre y los apellidos** 4. **El usuario pulsa el botón *Siguiente*** 5. **La aplicación pasa al paso dos de la configuración inicial** 6. **El usuario escribe una nueva contraseña** 7. **El estudiante confirma los cambios pulsando el botón *Finalizar*.**

8. **El sistema registra los cambios, muestra un mensaje de que se ha configurado bien la cuenta y asigna los primeros puntos al estudiante.**

7 Modelo de datos

La base de datos se implementará con SQLite y seguirá el modelo relacional, constará de las distintas tablas de la aplicación, que guardarán el progreso de aprendizaje del estudiante así como la configuración de módulos con sus temas, preguntas y respuestas asociadas.

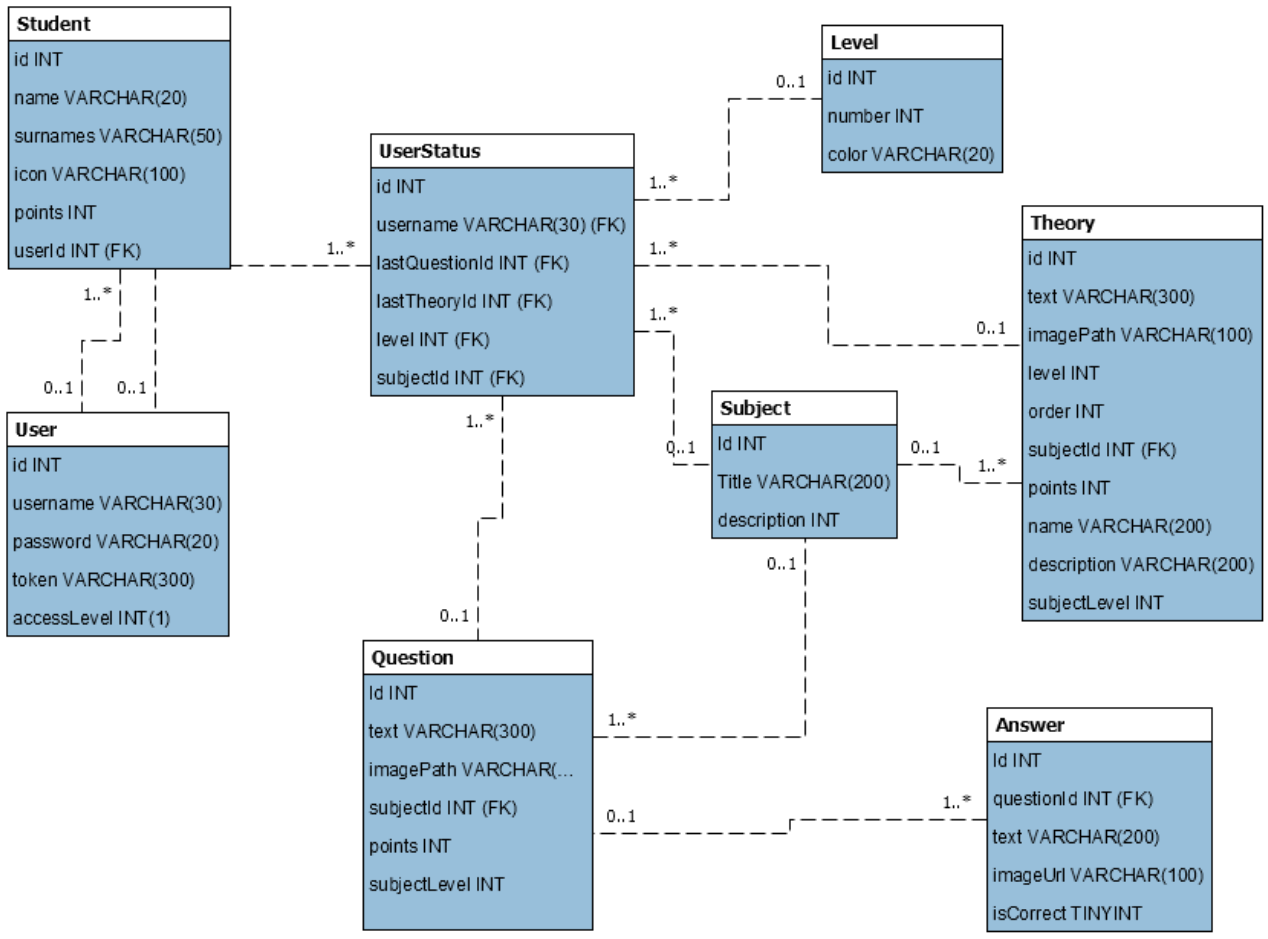


Figura 4: Modelo de datos

En este diagrama el modelo de datos se representa con notación clásica para los objetos y UML para las relaciones.

7.1 Tablas

Cada tabla dispondrá de un identificador autonumérico de tipo entero, *id*, como clave primaria.

La notación que se aplicará será pascal case para los nombres de tablas y camel case para los campos.

7.1.1 User

Registrará los usuarios de la aplicación, tanto administradores como estudiantes, se distinguirán con el campo *AccessLevel*, siendo el 0 para el usuario administrador y el 1 para el estudiante. Esta tabla será la encargada de guardar los datos de autenticación, nombre de usuario, contraseña y token.

Table name: User ☐ WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							*NULL*
2	Username	VARCHAR (30)							*NULL*
3	Password	VARCHAR (20)							*NULL*
4	Token	VARCHAR (300)							*NULL*
5	AccessLevel	INTEGER (1)							*NULL*

Ilustración 10: Tabla User

7.1.2 Student

Esta tabla guarda los datos del perfil del estudiante, nombre, apellidos, icono elegido, puntos actuales y el id de usuario de la aplicación.

Table name: Student ☐ WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							*NULL*
2	name	VARCHAR (20)							*NULL*
3	surnames	VARCHAR (50)							*NULL*
4	icon	VARCHAR (100)							*NULL*
5	points	INTEGER							*NULL*
6	userId	INTEGER							*NULL*

Ilustración 11: Tabla Student

7.1.3 UserStatus

Esta tabla contiene los datos del avance del usuario, registra cada materia a la que está asignado, la pregunta por la que va, la última entrada de teoría que ha leído y el nivel.

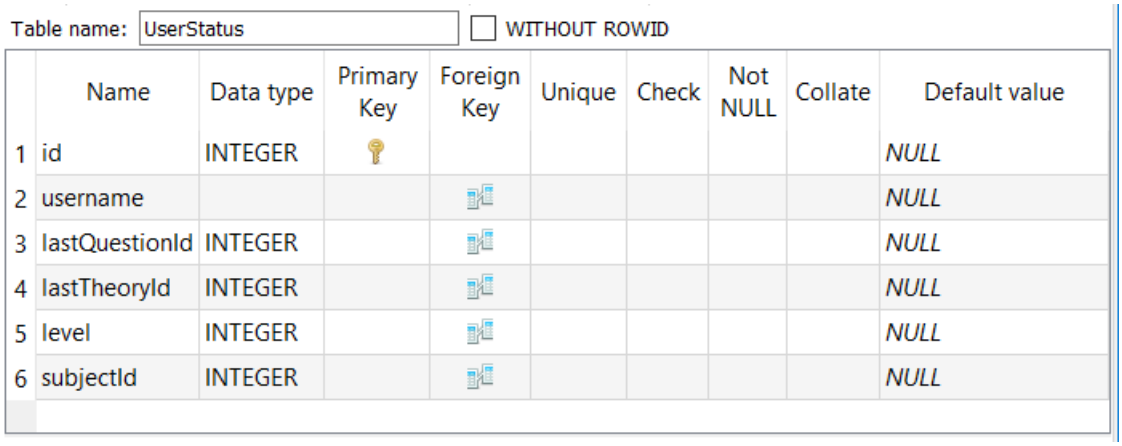


Table name: UserStatus WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							NULL
2	username								NULL
3	lastQuestionId	INTEGER							NULL
4	lastTheoryId	INTEGER							NULL
5	level	INTEGER							NULL
6	subjectId	INTEGER							NULL

Ilustración 12: Tabla UserStatus

7.1.4 Theory

Tabla que contendrá la teoría de cada materia, los contenidos podrán ser tanto texto como imágenes (ruta al archivo), tendrá asociado un nivel y una cantidad de puntos.

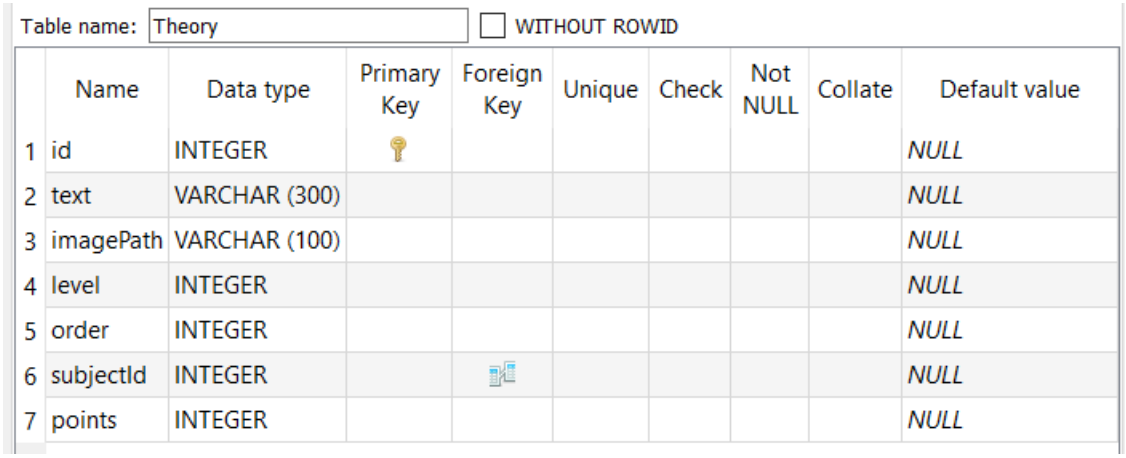


Table name: Theory WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							NULL
2	text	VARCHAR (300)							NULL
3	imagePath	VARCHAR (100)							NULL
4	level	INTEGER							NULL
5	order	INTEGER							NULL
6	subjectId	INTEGER							NULL
7	points	INTEGER							NULL

Ilustración 13: Tabla Theory

7.1.5 Subject

Esta tabla registra las materias/módulos que se incluyen en la aplicación, inicialmente solo contiene el título

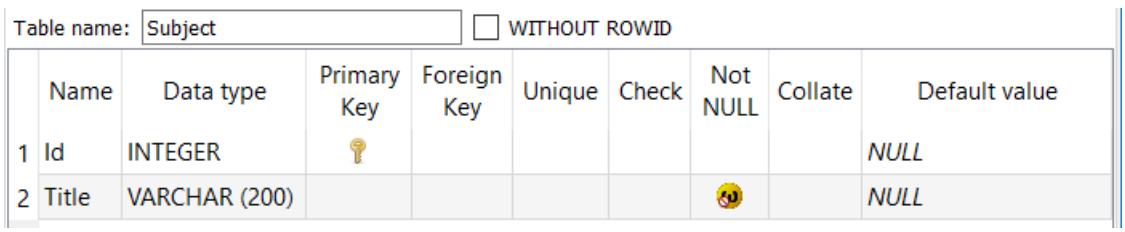


Table name: Subject WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	Id	INTEGER							NULL
2	Title	VARCHAR (200)							NULL

Ilustración 14: Tabla Subject

7.1.6 Question

Tabla que guarda los datos de las preguntas, texto, ruta de la imagen en caso de que tenga, materia a la que pertenece y puntos.

Table name: Question WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	Id	INTEGER							*NULL*
2	text	VARCHAR (300)							*NULL*
3	imagePath	VARCHAR							*NULL*
4	subjectId	INTEGER							*NULL*
5	points	INTEGER							*NULL*

Ilustración 15: Tabla Question

7.1.7 Level

Tiene los datos del nivel, número y color, está pensada para el caso en el que se permita al administrado cambiar el color del nivel

Table name: Level WITHOUT ROWID

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Default value
1	id	INTEGER							*NULL*
2	number	INTEGER							*NULL*
3	color	VARCHAR							*NULL*

Ilustración 16: Tabla Level

7.1.8 Answer

En esta tabla se guardarán las respuestas de cada pregunta, se distinguirá entre válidas y no válidas mediante el campo *isCorrect*, texto en caso de necesitarlo e imagen si es requerida.

8 Diagrama de clases

Diagrama de clases UML que describe las clases del sistema con sus operaciones, atributos y relaciones entre objetos.

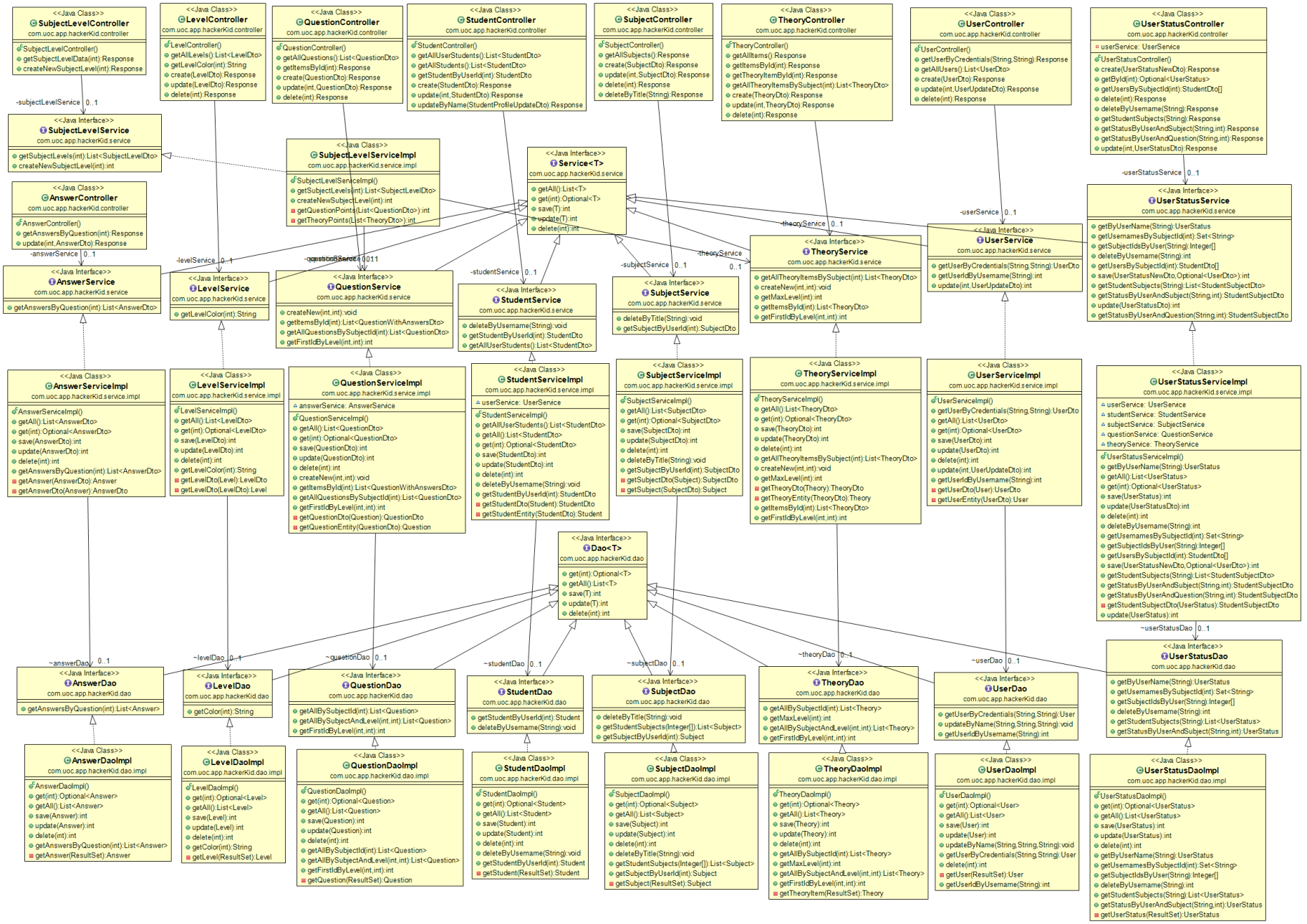


Figura 5: Diagrama de clases

9 Iteraciones

9.1 Sprint 1

9.1.1 Análisis

Análisis funcional

- Login

Todos los usuarios de la aplicación se registran en la tabla de usuarios, la diferencia entre un usuario administrador y un estudiante es el nivel de acceso, 0 para los administradores y 1 para los usuarios.

Se utilizará una sola tabla para evitar diferentes consultas, manejo de mayor número de datos y mayor número de operaciones que penalicen el rendimiento y la respuesta de cara al usuario.

No se creará registro del estudiante hasta que él mismo entre en la aplicación y configure su propia cuenta. La pantalla de configuración se creará mas adelante.

- Gestión de usuarios

Los usuarios se creen inicialmente con la contraseña igual al nombre, en el caso de los estudiantes se pedirá en el primer inicio de sesión que la cambien. Los administradores podrán cambiar el nombre de usuario y la contraseña de cualquier usuario, no podrán editar mas campos.

Administradores: Los usuarios administradores solo tienen acceso a la configuración y no tienen registros en el sistema aparte de su propia cuenta de usuario, con lo cual la única información que se puede manejar es el nombre de usuario y la contraseña, estos se podrán editar.

La pantalla de gestión de usuario debe mostrar el listado de administradores y el de estudiantes con las opciones de crear, borrar y modificar nombre de usuario y contraseña.

El listado de estudiantes mostrará la información del estudiante y los datos introducidos de nombre y apellidos, así como los puntos.

- Gestión de materias

Las materias se mostrarán como un listado configurable en el que se podrán realizar las operación de creación, borrado y actualización.

- Listado de materias

Los usuarios podrán ver las materias a las que están inscritos.

- Dashboard de administración

El administrador tiene dos cometidos principales en la aplicación, la gestión de usuarios y la de contenidos, al iniciar sesión se mostrarán dos opciones al administrador para que pueda dirigirse a las páginas encargadas de ello.

9.1.2 Diseño

9.1.2.1 Infraestructura

HackerKid – Proyecto Angular (Front-end)

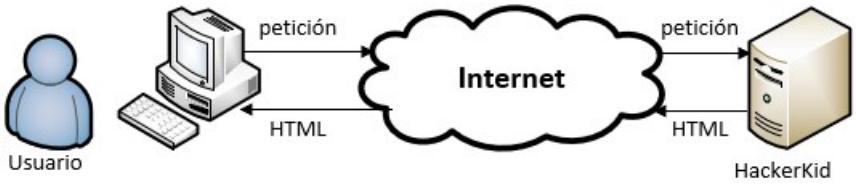


Figura 6: Comunicación del usuario con la aplicación

Se debe desarrollar un proyecto Angular con la última versión disponible para el desarrollo de las pantallas de la aplicación, 1.7+.

El código estará organizado en módulos y estos en componentes, cada componente tendrá asociados al menos un fichero typescript y un template html, los estilos irán en archivos sass y los test en archivos spec de la siguiente forma:

```
/login
login.component.ts // código
login.component.html // template
login.component.spec.ts  // test
login.component.scss  // estilos
login.module.ts // modulo
```

El proyecto se basará en material design para mostrar el contenido, usando cuando se pueda sus componentes. El tema será *deeppurple-amber*, se deberá importar en la aplicación, esto establecerá una paleta de colores por defecto con base de color violeta, que meterá estilos en algunos componentes y se podrán usar desde toda la aplicación.

Librerías iniciales a incluir en el fichero de configuración packages-json, de haber una posterior se instalará la posterior:

Visual Studio Code versión 1.19.0
Angular 7.0.0

HackerKidApis – Proyecto Java (Back-end)

El Back-end proveerá una API REST que será la que se encargue que proporcionar los recursos necesarios para el cliente, la información necesaria para la aplicación.

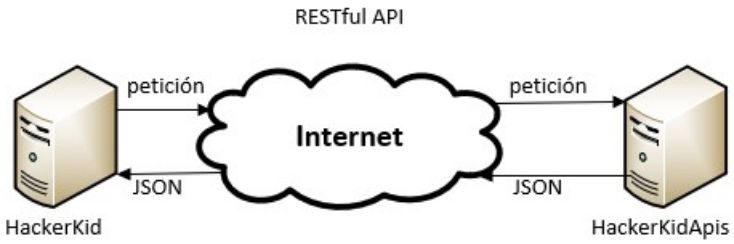


Figura 7: Integración entre el Frontend y el Backend

Configuración del workspace

Se creará un Dynamic web project, con Maven, para el despliegue en local se usará Apache Tomcat como servidor de aplicaciones y para la REST API se usará JAX-RS con el framework Jersey, la decisión de esta configuración es debido a lo liviano que es, es una configuración sencilla con pocas dependencias de librerías externas.

Para comunicarse con la base de datos la capa de datos necesitará el driver JDBC para SQLite.

IDE: Eclipse Oxigen
Proyecto: Dynamic 3.1
Servidor: TomCat 9
REST API: Jax-rs 1.1 con framework: jersey 2.27

El descriptor de despliegue derá contener las siguientes librerías:

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.xerial</groupId>
        <artifactId>sqlite-jdbc</artifactId>
        <version>3.25.2</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.core</groupId>
        <artifactId>jersey-server</artifactId>
        <version>2.27</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.containers</groupId>
        <artifactId>jersey-container-servlet</artifactId>
        <version>2.27</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.inject</groupId>
        <artifactId>jersey-hk2</artifactId>
        <version>2.27</version>
    </dependency>
    <dependency>
        <groupId>org.glassfish.jersey.media</groupId>
        <artifactId>jersey-media-json-jackson</artifactId>
        <version>2.27</version>
    </dependency>
</dependencies>
```

Figura 8: Descriptor de despliegue

9.1.2.2 Diseño técnico

- Back-end

Seguirá el patrón MVC, estará dividido en capa de datos, capa de negocio y capa de presentación.

Para la capa de datos se implementará el patrón DAO para desacoplar la implementación del acceso a base de datos de la propia base de datos, será independiente.

En la capa de presentación se empleará el patrón DTO, para la transferencia de datos entre Front-end y Back-end,

Se separará la lógica de conexión a la base de datos, se creará un paquete db en el que irá la conexión con el driver JDBC. Inicialmente tendremos aquí la cadena de conexión.

```java
public class SQLiteJDBCDriverConnection {
    public static final String URL = "jdbc:sqlite:D:/HackerKidApis/hackekidApi/hacker.db";

    /**
     * Connects to the database
     */
    public static Connection getConnection() {
        Connection conn = null;
        try {
            // create a connection to the database
            Class.forName("org.sqlite.JDBC");
            conn = DriverManager.getConnection(URL);

        } catch (SQLException e) {
            System.out.println(e.getMessage());
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }

         return conn;
    }
}
```

Figura 9: SQLiteJDBCDriverConnection - ejemplo conexión a la base de datos

Ejemplo de implementación de un servicio en Back-end: Caso del login

Capa de datos

El login necesita acceder a la tabla de los usuarios, así que partiendo de la capa de datos se creará una clase *UserDaoImpl* que realizará la consulta de los credenciales en la base de datos, se abstraerá esta implementación mediante una interfaz que será la que se comunique con la capa de negocio, *UserDao*.

Se usará un *prepared statement* para evitar inyección SQL al pasar los datos como contenido de los parámetros y como parte de la SQL.

Se usará el *try-with-resources* de Java 7 para que la conexión se cierre siempre tanto en caso de error como si no lo hay.

Se devolverá el usuario, el cual será necesario para la sesión en la aplicación.

```java
@Override
public User getUserByCredentials(String name, String password) {
    try (Connection connection = SQLiteJDBCDriverConnection.getConnection()) {
        PreparedStatement ps = connection.prepareStatement("SELECT * FROM user WHERE username=? and password=?");
        ps.setString(1, name);
        ps.setString(2, password);

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            User user = getUser(rs);
            return user;
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return null;
}
```

Figura 10: UserDaoImpl - getUserByCredentials

La interfaz *UserDao* tendrá la signatura del método *getUserByCredentials..*

```java
package com.uoc.app.hackerKid.dao;

import com.uoc.app.hackerKid.model.User;

public interface UserDao {

    User getUserByCredentials(String name, String password);

}
```

Figura 11: Interfaz UserDao

Modelo: Se creará la clase *User* para la entidad usuario con los datos del usuario, los atributos serán privados y dispondrá de métodos de acceso públicos, set y get, para establecer o recuperar los valores.

Capa de negocio

En esta capa irá la lógica, a través de la interfaz *UserDao* creará un objecto de la clase *UserDaoImpl,* tendrá el método *getUserByCredentials* que llamará al método del mismo nombre en la capa de datos, este método será público para que sea accesible desde la capa de presentación.

Aplicando el patrón DTO se convierte la salida del método a un objecto serializable DTO, que en este caso tendrá los mismos atributos y métodos que la entidad.

La interfaz *UserService* contendrá la signatura del método.

```java
public class UserServiceImpl implements UserService {
    UserDao userDao = new UserDaoImpl();

    @Override
    public UserDto getUserByCredentials(String name, String password) {
        User user = userDao.getUserByCredentials(name, password);

        UserDto dto = getUserDto(user);
        return dto;
    }
```

Figura 12: UserServiceImpl - Método getUserByCredentials

Capa de presentación

Será la que defina los servicios REST, para ello tendrá clases que hacen de controlador, en este caso el *UserController*, que llamará al *UserService* de la capa de negocio y en función de lo que reciba devolverá una Reponse con status 200 y los datos del usuario o una respuesta con status 204 indicando que la petición ha ido bien pero no se han recuperado los datos.

Siguiendo la nomenclatura REST el servicio tendrá la siguiente ruta: '*/users/name/{name}/password/{password}*', donde entre llaves irán los parámetros de entrada del servicio, que nos llegan del Front-end, el método será e tipo GET y devolverá un objeto de tipo *UserDto* que se enviará en formato JSON.

```java
@Path("/users")
public class UserController {
    private UserService userService = new UserServiceImpl();

    @GET
    @Path("name/{name}/password/{password}")
    @Consumes(MediaType.APPLICATION_JSON)
    @Produces(MediaType.APPLICATION_JSON)
    public Response getUserByCredentials(@PathParam("name") String name, @PathParam("password") String password) {
        if (null == name || name.isEmpty()) {
            return Response.status(Response.Status.NO_CONTENT).build();
        }

        UserDto user = userService.getUserByCredentials(name, password);
        if (user == null) {
            return Response.status(Response.Status.NO_CONTENT).build();
        } else {
            return Response.status(Response.Status.OK).entity(user).build();
        }

    }
```

Figura 13: UserController - getUserByCredentials

A continuación se muestra el diagrama de secuencia donde se pueden apreciar las llamadas que se realizan de una capa a otra a través de los interfaces desde el controlador hasta llegar al driver de la base de datos.

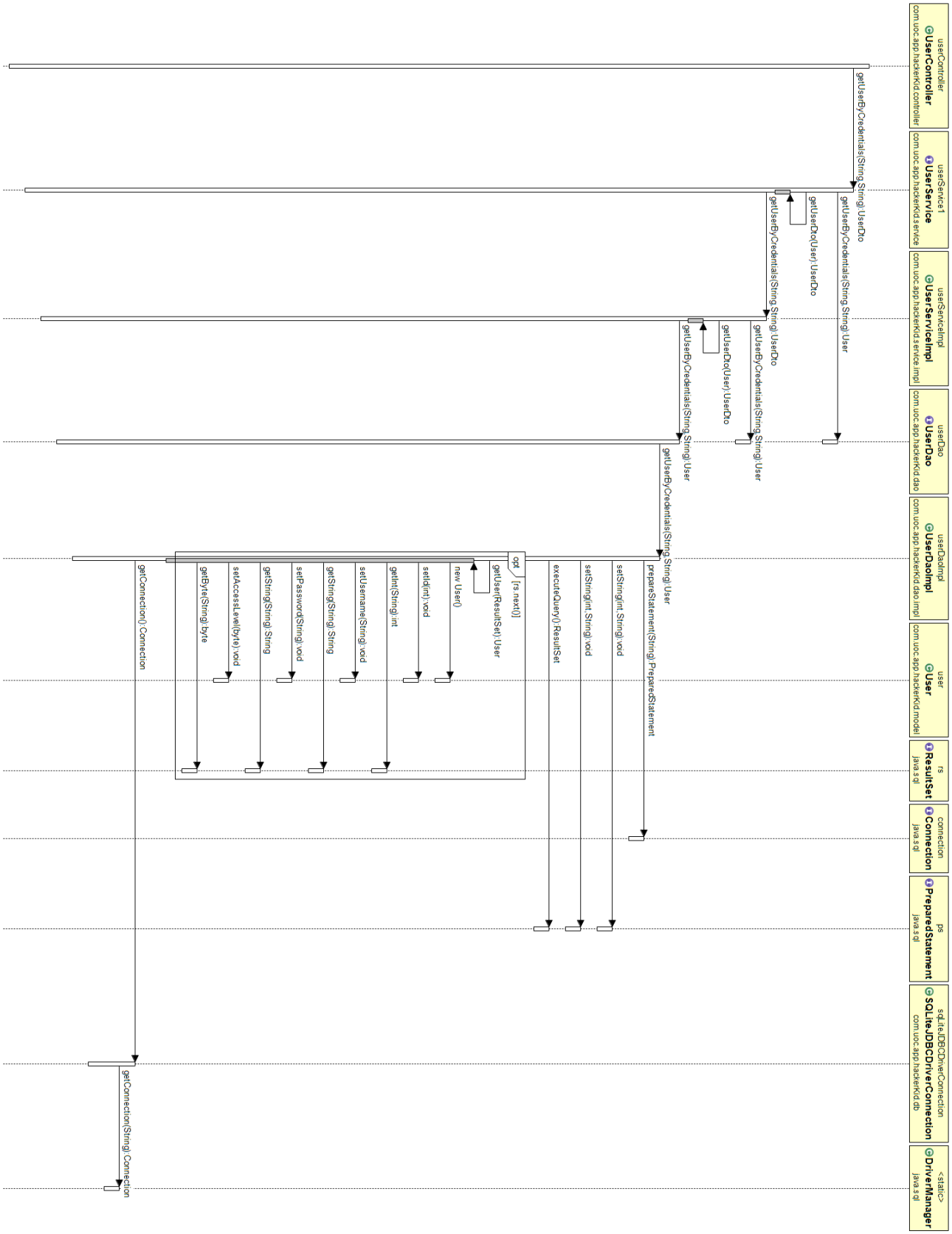


Figura 14: Diagrama de secuencia – petición getUserCredentials

- Front-End

Estructura

Módulo app: Módulo inicial o raíz, inicia la aplicación, se crea al crear el proyecto.

El resto de la aplicación se organizará en módulos de manera funcional. Por ejemplo: el módulo del login, que contendrá lo necesario para efectuar el login en la aplicación.

Login

El módulo del *Login* será un componente que cuelga del raíz, se invocará desde el módulo app de la aplicación y se cargará directamente al arrancar la aplicación.

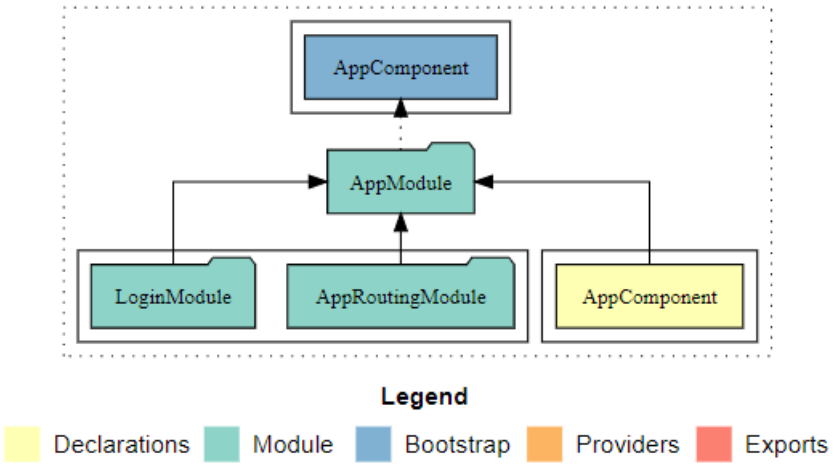


Figura 15: Módulo app da la aplicación

Este módulo contendrá el componente, que tendrá la *template*, el fichero de estilos y el código, así mismo constará de una carpeta en la que tendrá el servicio usado por el login y otra con el modelo.

```
@NgModule({
    declarations: [LoginComponent],
    imports: [
        MatFormFieldModule,
        MatInputModule,
        MatCardModule,
        MatButtonModule,
        MatSnackBarModule,
        ReactiveFormsModule,
        CommonModule,
        LoginRoutingModule
    ],
    providers: [LoginService],
})
```

Figura 16: Ejemplo de declaración de módulo - caso del login

Decorador *@Component*: registra el componente, en selector irá el nombre del componente, en templateUrl la ruta al código HTML y en styleUrl, la ruta o rutas a los ficheros de estilos.

```
@Component({
  selector: 'app-login',
  templateUrl: 'login.component.html',
  styleUrls: ['./login.component.scss']
})
```

Figura 17: Decorador del componente

- Servicios

Se deberá crear un servicio para comunicar el componente del login con el servicio REST expuesto por el Back-end. Este servicio estará en una carpeta services dentro de la carpeta del componente de login.

Para indicar que es un servicio se le añade el decorador *@Inyectable.*

En el contructor se incluirá la clase *HttpClient* de angular, que nos permitirá hacer peticiones *GET, POST, PUT, PATCH y DELETE.*

El el módulo se debe añadir el servicio en el listado de *providers* y en el componente se inyectará a través del constructor, luego se podrán invocar sus métodos desde el resto de la clase.

```
/**
 * Gets the user credentials
 */
getUserCredentials(username: string, password: string): Observable<User> {
  return this.http.get<User>(
    `http://localhost:8080/hackerKid/users/name/${username}/password/${password}`
  );
}
```

Figura 18: Ejemplo de llamada a servicio REST desde el servicio

9.1.3 PBR

Al ser la versión inicial se dará más prioridad a la configuración del entorno y pruebas de integración de los proyectos.

La estimación se realiza con tallas de camiseta, con los valores XS, S, M, L, XL y ? cuando no está claro.

8.1.1.1 Product backlog

Historias de usuario

Código	Título	Descripción	Prioridad	Talla
US0001	Implementación de la base de datos	Como usuario quiero disponer de una base de datos que se contenga los datos de la aplicación así como la configuración de contenido y usuarios	1	S
US0002	Pantalla de login	Como usuario quiero disponer de una pantalla que me permita el acceso a la aplicación con mi cuenta de usuario	1	XL
US0003	Listado de administradores	Como usuario administrador quiero disponer de las opciones de gestión de cuentas de usuario administrador.	2	M
US0004	Listado de estudiantes	Como usuario administrador quiero disponer de las opciones de gestión de cuentas de usuario estudiante.	2	M
US0005	Pantalla de configuración de materias	Como administrador quiero poder realizar las labores de gestión de materias	3	S
US0006	Pantalla de listado de materias	Como estudiante quiero poder ver a que materias estoy inscrito	4	S
US0007	Pantalla de dashboard de administrador	Como administrador quiero tener una pantalla que me permita tras iniciar sesión elegir en ir a la configuración de usuarios o a la de la aplicación	4	XS

9.1.4 Planning

Las metas de este sprint son la pantalla de login y la pantalla de listado de usuarios. En este sprint se deberán crear los proyectos y la base de datos, tras lo cual se deberán realizar las pantallas de login y listado de los usuarios.

Goal 1 – Pantalla de login	Work items
Infraestructura - Implementación de la base de datos	US0001
Proyecto inicial Angular	US0002
Proyecto iniciat Java	US0002
Integración de los proyectos	US0002
Pantalla de login	US0002
Goal 2 - Pantalla de listado de usuarios	**Work items**
Listado de administradores	US0003
Listado de estudiantes	US0004

Figura 19: Goals Sprint 1

9.1.4.1 Sprint backlog

Historias de usuario				
Código	Título	Descripción	Prioridad	Talla
US0001	Implementación de la base de datos	Como usuario quiero disponer de una base de datos que se contenga los datos de la aplicación así como la configuración de contenido y usuarios	1	S
US0002	Pantalla de login	Como usuario quiero disponer de una pantalla que me permita el acceso a la aplicación con mi cuenta de	1	XL

		usuario		
US0003	Pantalla de listado de usuarios	Como usuario administrador quiero disponer de las opciones de gestión de cuentas de usuario.	2	M

9.1.5 Implementación

Frontend

El proyecto se crea con el comando *ng new hackedKid* y el componente inicial del login con el comando *ng generate component login*

- Login

Se crea la web de login que pide el usuario y la contraseña, ambos campos son obligatorios así que mientras esté uno vacío el botón de *Entrar* estará deshabilitado.

Se mostrará un mensaje si no se puede iniciar sesión con esos datos y otro si ocurre algún problema distinto con un *Snackbar*.

- Listado de estudiantes

Usando una tabla de Angular material se muestran los datos de los estudiantes. Se ofrecen las opciones de editar y borrar fila mediante botones, dejando a futuro la decisión de si cambiarlo por un menú.

- Listado de administradores

Será similar al de estudiantes, salvo que contiene menos datos ya que no se guardan registro de los datos personales.

Backend

- Login

Desde el back-end se publicará un servicio para comprobar los credenciales del usuario en el sistema.

Integración

Servicios REST a crear en este sprint:

Servicio para comprobar los credenciales: a partir de un nombre de usuario y una contraseña devolverá los datos del usuario.

CRUD de usuarios para los listados de administradores y estudiantes

Se siguen las REST convections para nombrar recursos REST, se eliminan los errores que aparecen al probar los servicios desde el swagger.

Los servicios devuelven código 200 si todo ha ido bien, 204 si la llamada ha ido bien pero no han habido cambios y 500 si no ha ido bien la petición.

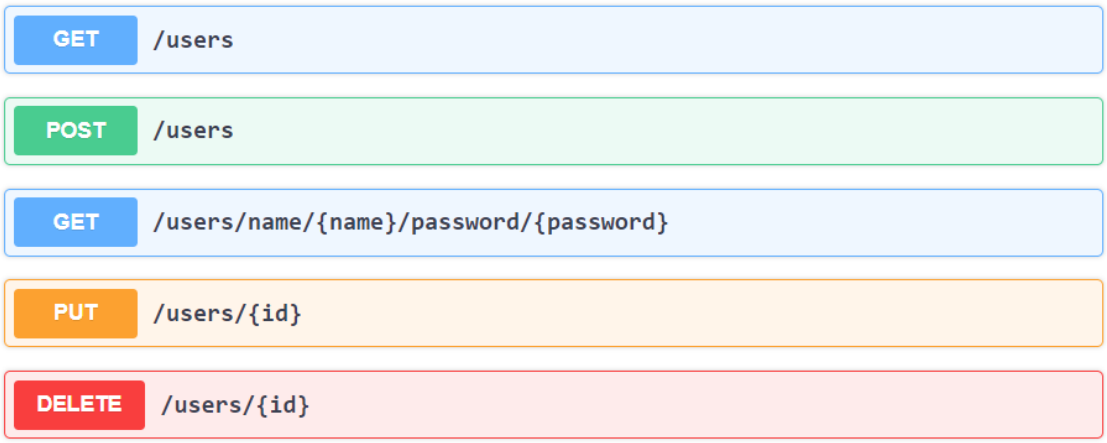


Figura 20: Servicios REST - Gestión de usuarios

Ejemplo: Delete

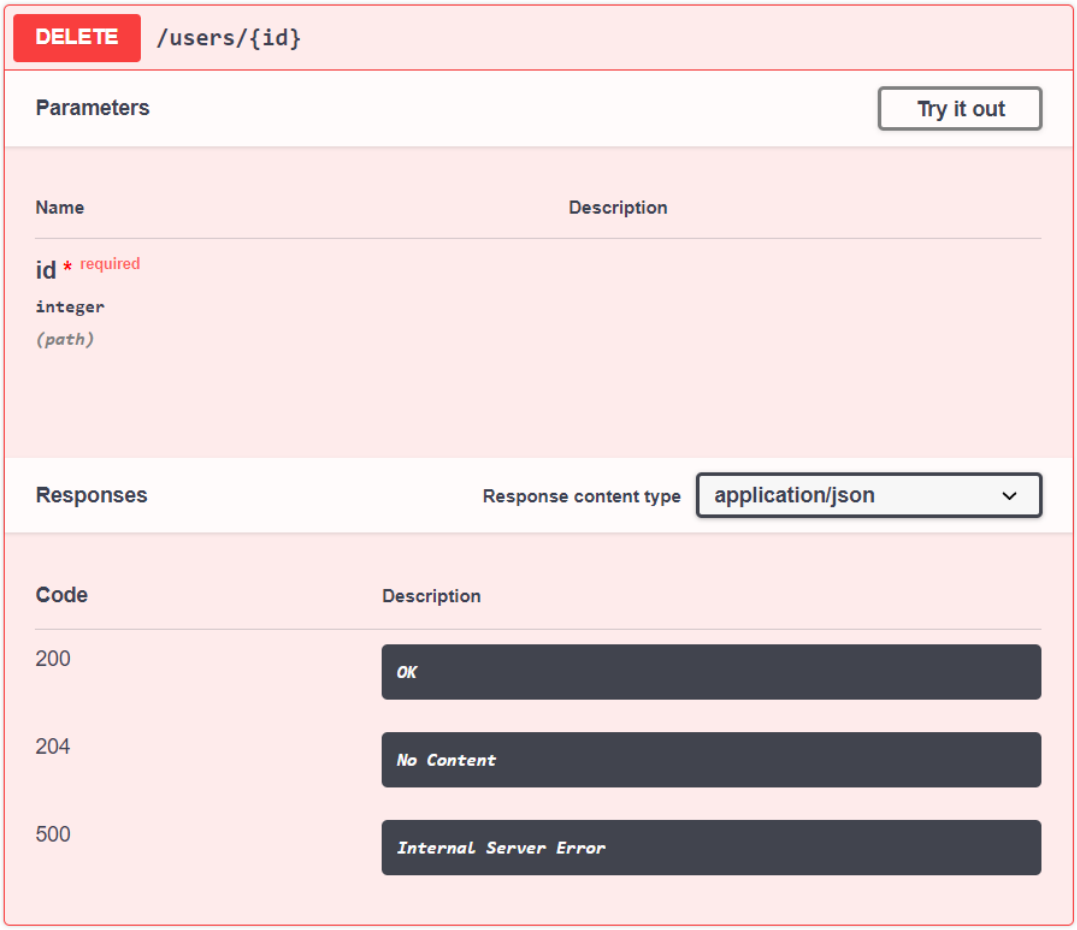


Figura 21: Delete swagger view

9.1.6 Review

9.1.6.1 Goal 1 – Pantalla de login

- US0001 – Implementación de la base de datos

Descripción: Esta historia tiene como objetivo crear la base de datos SQLite a partir del diseño inicial. Inicialmente irá incluida en el directorio raíz de la aplicación.

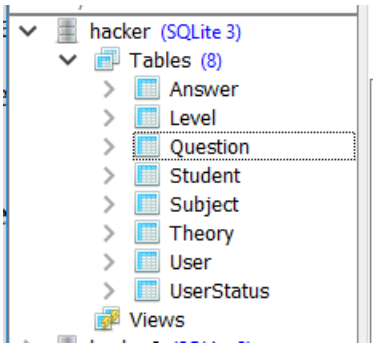


Figura 22: Base de datos en SQLiteStudio

Resultado: Se crea la base de datos correctamente. Se prueban distintas consultas CRUD de las tablas.

- US0002 – Pantalla de login

Descripción: El objetivo principal de esta historia de usuario es la creación de la pantalla de inicio de sesión de los usuarios.

Resultado: Se crea un componente de inicio de sesión que permite acceder a la aplicación. Los campos son requeridos y en caso de no rellenarse se indica en color rojo.

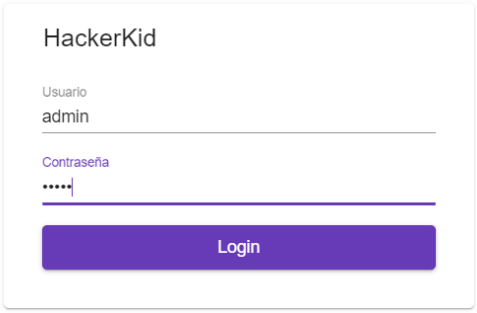


Figura 23: Pantalla de login

9.1.6.2 Goal 2 – Pantalla del listado de usuarios

- US0003 – Listado de administradores

  Descripción: Esta historia de usuario tiene como objetivo el crear una interfaz para que los administradores puedan gestionar las cuentas de los administradores.

- US0004 – Listado de estudiantes

  Descripción: Esta historia de usuario tiene como objetivo el crear una interfaz para que los administradores puedan gestionar las cuentas de los estudiantes.

Resultado: Se crea una pantalla en la que se muestran los listados tanto de administradores como de estudiantes, como las opciones de creación, edición y borrado.

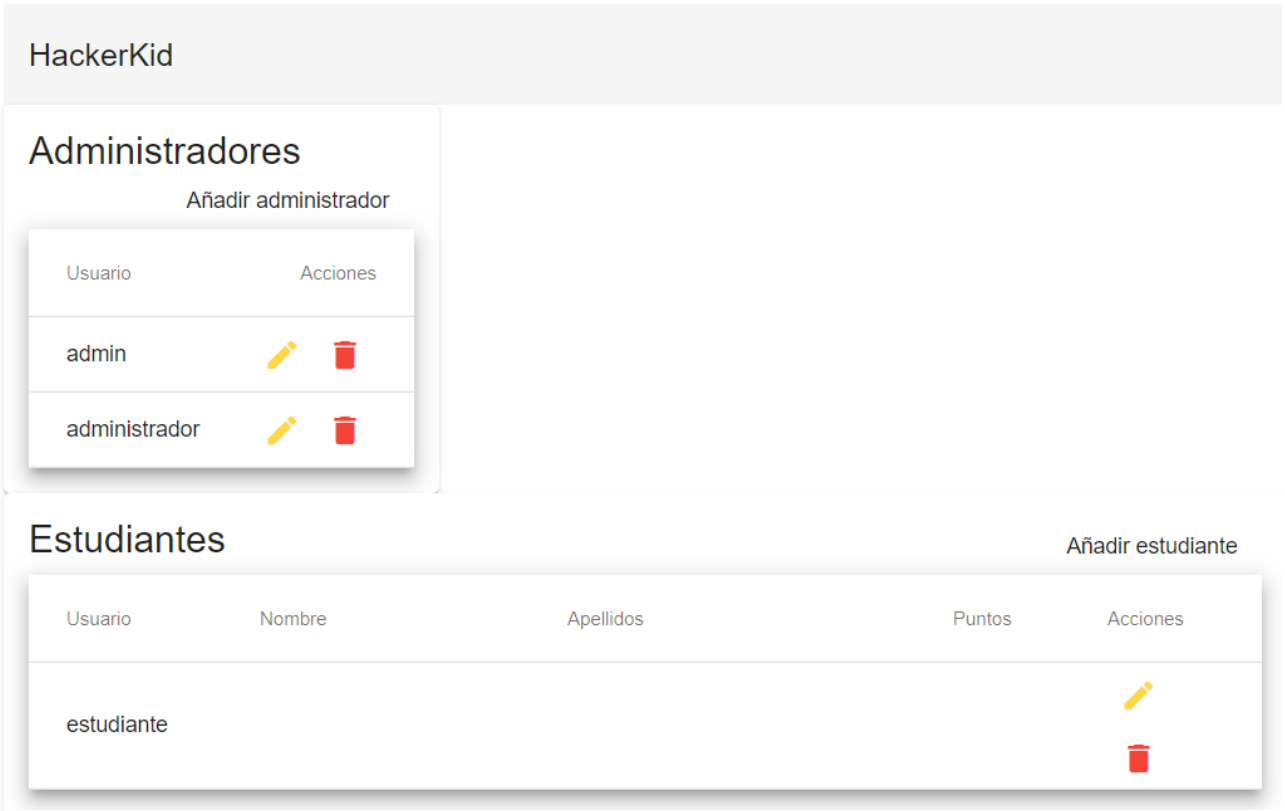


Figura 24: Pantalla "Gestión de usuarios"

Diálogo de crear o editar usuario. Permitirá la creación de nuevos usuarios

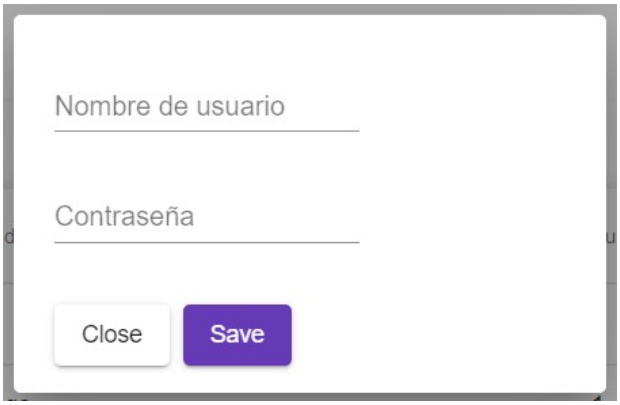


Figura 25: Diálogo de edición de usuario

9.1.7 Retrospective

Resultados de la retrospective

A pesar de que configurar el proyecto de Back-End ha llevado más tiempo de la cuenta, debido a diversos problemas con versiones de librerías y de configuración del entorno, el sprint ha ido bien y se han cumplido las metas.

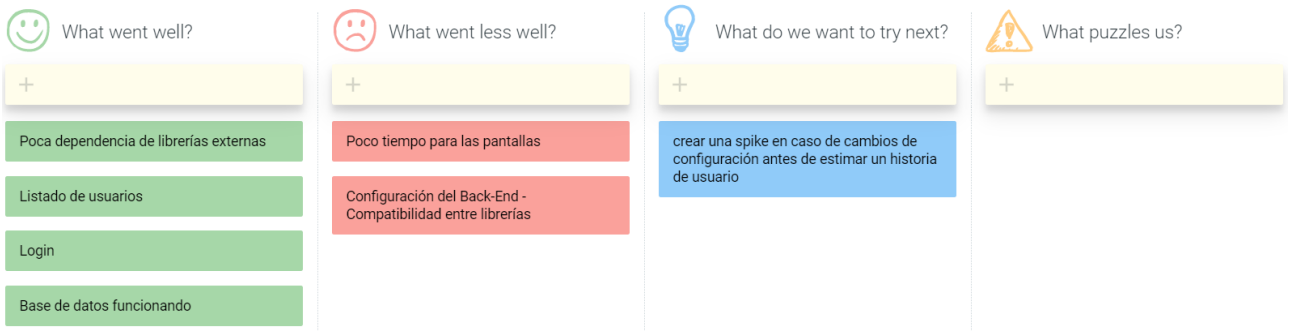


Figura 26: Retrospective del sprint 1

9.2 Sprint 2

9.2.1 Análisis

Análisis funcional

- Dashboard de estudiantes

Los estudiantes tendrán materias asignadas, para poder verlas se debe disponer de una pantalla que las muestre. Los datos relevantes para el estudiante serán el nombre de la materia, el nivel y los puntos.

- Dashboard de administración

El administrador tiene la responsabilidad de la gestión de usuarios y la gestión de contenidos que verán los estudiantes. Se debe dar al administrador la opción de elegir que acción quiere realizar.

- Configuración de materias

Las materias disponen de título, descripción e imagen, el administrador podrá realizar CRUD sobre ellas.

9.2.2 Diseño

- Dashboard de estudiantes

Back-end

Debido a que más adelante harán falta para otras entidades, se decide crear las interfaces *Dao* y *Service* con los métodos CRUD. Las clases del modelo implementarán la interfaz *Dao* y los servicios implementarán la interfaz *Service.* Esto homogeneizará los métodos y avanzará trabajo de cara a futuras implementaciones.

Se creará un servicio REST que permita recuperar las materias por usuario activo.

Front-end

Se creará una página para el usuario que contenga las materias a las que está inscrito. Se creará un componente que contenga los datos de la materia y se mostrará recorriendo la lista proporcionado por el servicio

- Dashboard de administración

Será una pantalla que muestre al usuario las dos opciones de navegación, se cargará mediante *lazy loading* y a su vez también el dashboard de estudiante, que tendrá mayor carga.

- Configuración de materias

Back-end

Se implementará el CRUD de materias siguiendo la estructura descrita en el ejemplo del login. Se crearán los servicios REST para exponer estas funcionalidades. De la misma manera se crearán también para los temas, preguntas y respuestas.

Front-end

Se proporcionará al administrador una página con el listado de materias y las opciones de añadir, editar y borrar.

Dentro de la página del listado de materias se incluirá la opción de añadir nivel, dentro estarán los temas, las preguntas y las respuesta, de las que se debe implementar también el CRUD.

9.2.3 Implementación

Back-end

Se crea el método *getSubjectByUserId* en la capa de datos que recupera una materia por id de usuario.

Front-end

Se crea el módulo *studentDashboard,* para mostrar las materias a las que está subscrito el estudiante se usan *mat-card* de Angular Material, tras probar los diseños que propone la página se elige uno con la imagen en la parte superior y los datos del nivel, nombre de la materia y puntos en la parte de abajo.

El color de fondo corresponde al nivel, para los colores se eligen diez que permitan ver bien el texto, de tono cercano al blanco, *aliceblue*. Los colores se guardan en un enumerado en la carpeta *common*, por si hacen falta para otro componente.

En la carpeta common irá lo que sea común o necesario tener accesible para toda la aplicación.

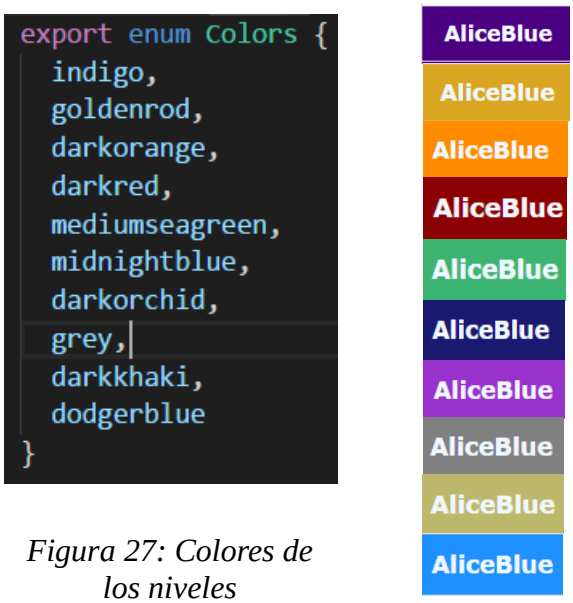


Figura 27: Colores de los niveles

Figura 28: Ejemplo de materia

- Gestión de temas

Intentando mantener un diseño sencillo y evitar que el administrador tenga que navegar por muchas páginas para la edición de contenido se crea el diálogo de edita tema para la edición de temas, el mismo diálogo sirve para la creación. En el se permitirá cambiar o añadir el nombre, la descripción (opcional), el texto, los puntos y la imagen.

Las imágenes se seleccionan de la carpeta assets del proyecto de front-end, la implementación se lleva a cabo teniendo en cuenta que a futuro las imágenes deberían guardarse en un servidor, con lo cual en lugar de guardan en un blob el archivo, se guarda la ruta.

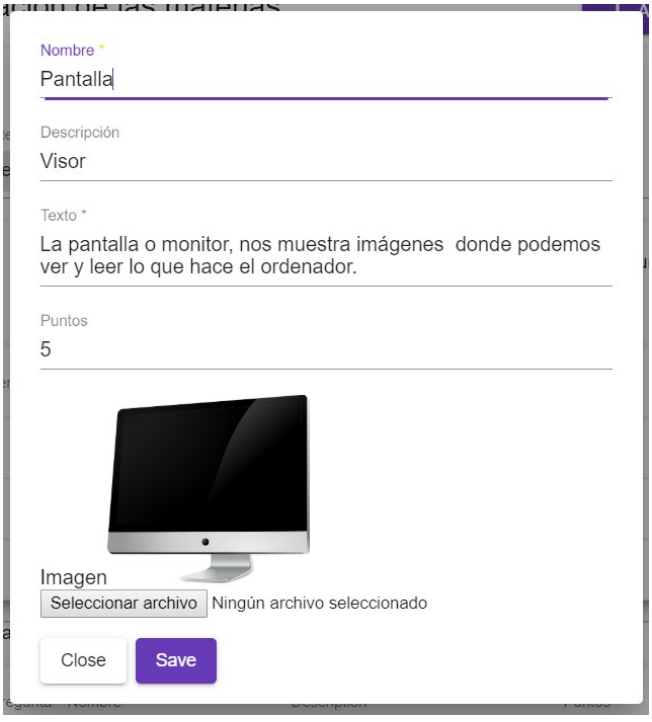


Figura 29: Edición de un tema

9.2.4 PBR

Se cambia la prioridad del desarrollo del dashboard a más alta por ser muy pequeña y permitir el acceso desde la aplicación a la gestión de contenidos.

9.2.4.1 Product backlog

Historias de usuario				
Código	Título	Descripción	Prioridad	Talla
US0007	Pantalla de dashboard de administrador	Como administrador quiero tener una pantalla que me permita tras iniciar sesión elegir en ir a la configuración de usuarios o a la de la aplicación	3	XS
US0005	Pantalla de configuración de materias	Como administrador quiero poder realizar las labores de gestión de materias	3	S
US0006	Pantalla de listado de materias	Como estudiante quiero poder ver a que materias estoy inscrito	4	S
US0009	Añadir nuevo nivel	Como administrador quiero poder añadir nuevos niveles a las materias	4	S
US0010	Pantalla de creación de temas	Como administrador quiero poder crear, modificar y borrar temas dentro de los niveles	4	M
US0011	Pantalla de lectura de temas	Como estudiante quiero poder leer los temas de las materias a las que estoy inscrito	4	M
US0012	Pantalla de creación de preguntas	Como administrador quiero poder crear, modificar y borrar preguntas asociadas a un nivel	4	M
US0013	Configuración inicial del usuario	Como estudiante quiero poder configurar mis datos la primera vez que inicie sesión	4	M
US0008	Mover los botones de editar y borrar a un menú	Como usuario quiere poder acceder a las opciones de editar y borrar desde un menú en lugar de botones	5	M

9.2.5 Planning

En este sprint las prioridades son los dashboard tanto del administrador como del cliente y la pantalla de configuración de materias.

Goal 1 – Dashboards	Work items
Dashboard de adminstración	US0007
Dashboard de estudiante	US0006
Goal 2 – Configuración de materias	**Work items**
Pantalla de configuración de materias	US0005

Figura 30: Goals Sprint 2

9.2.2.1 Sprint backlog

Las historias para este sprint son las relacionadas con la gestión de contenido

Historias de usuario				
Código	Título	Descripción	Prioridad	Talla
US0007	Pantalla de dashboard de administrador	Como administrador quiero tener una pantalla que me permita tras iniciar sesión elegir en ir a la configuración de usuarios o a la de la aplicación	3	XS
US0005	Pantalla de configuración de materias	Como administrador quiero poder realizar las labores de gestión de materias	3	S
US0006	Pantalla de listado de materias	Como estudiante quiero poder ver a que materias estoy inscrito	4	S
US0009	Añadir nuevo nivel	Como administrador quiero poder añadir nuevos niveles a las materias	4	S
US0010	Pantalla de creación de temas	Como administrador quiero poder crear, modificar y borrar temas dentro de los niveles	4	M
US0011	Pantalla de lectura de temas	Como estudiante quiero poder leer los temas de las materias a las que estoy inscrito	4	M

9.2.6 Review

9.2.6.1 Goal 1 - Dashboards

- US0006 – Pantalla de listado de materias (dashboard del usuario)

  Descripción: El objetivo de esta historia es la creación del dashboard del usuario, que mostrará las materias a las que está inscrito.

  Resultado: Se crea la pantalla, muestra las materias en recuadros, con el nivel y los puntos. El diseño es responsivo, se adapta a la pantalla. El color de fondo corresponde con el color del nivel.

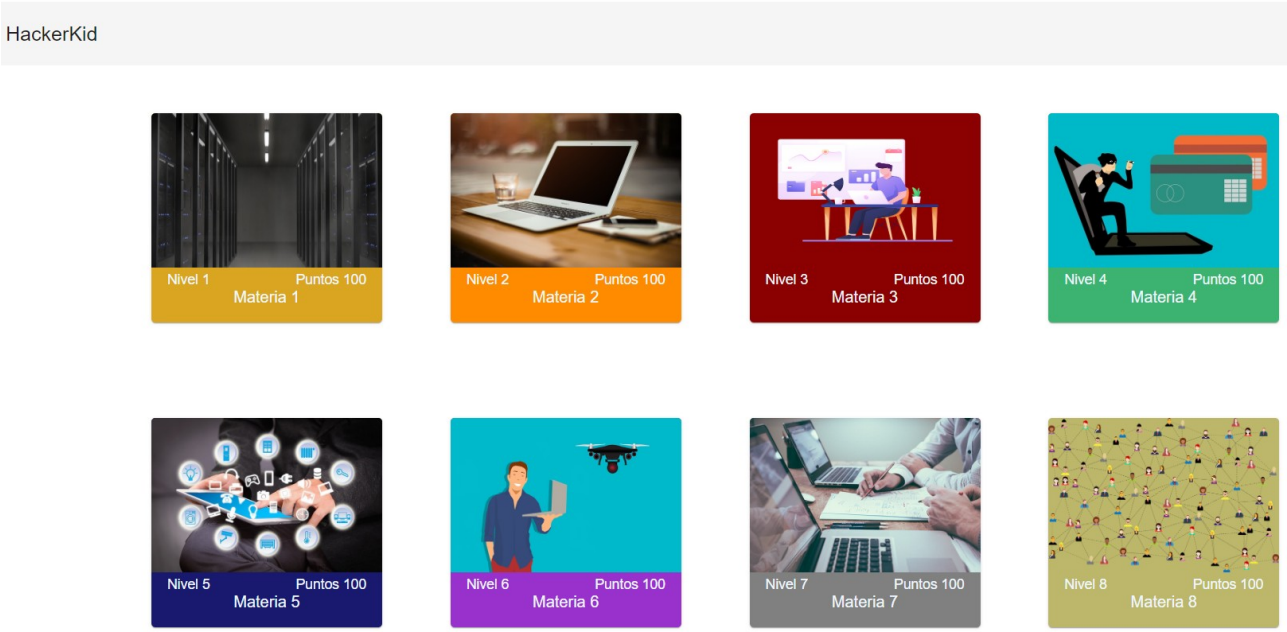


Figura 31: Dashboard de usuario con datos de prueba

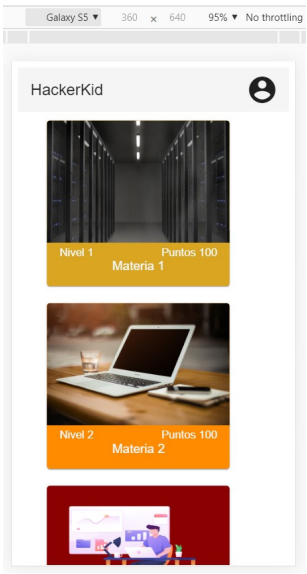


Figura 33: Resolución de móvil

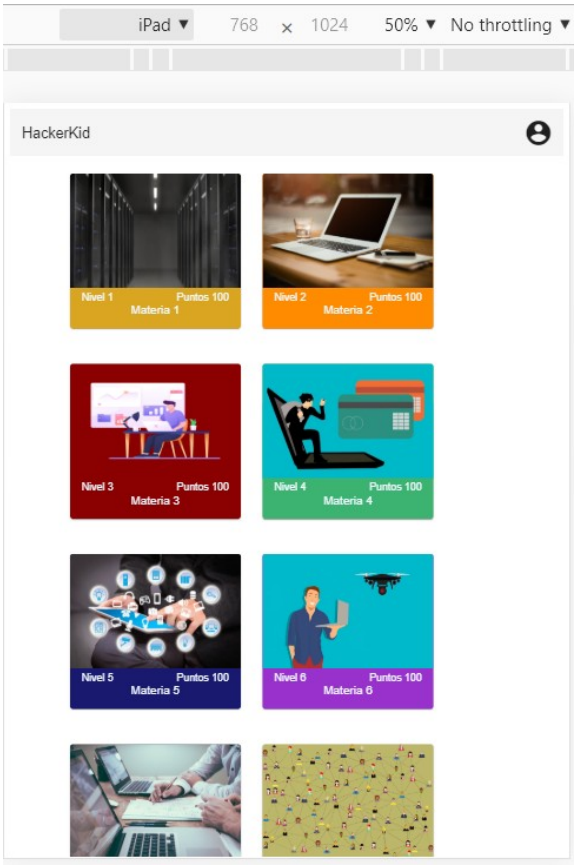


Figura 32: Resolución de tablet

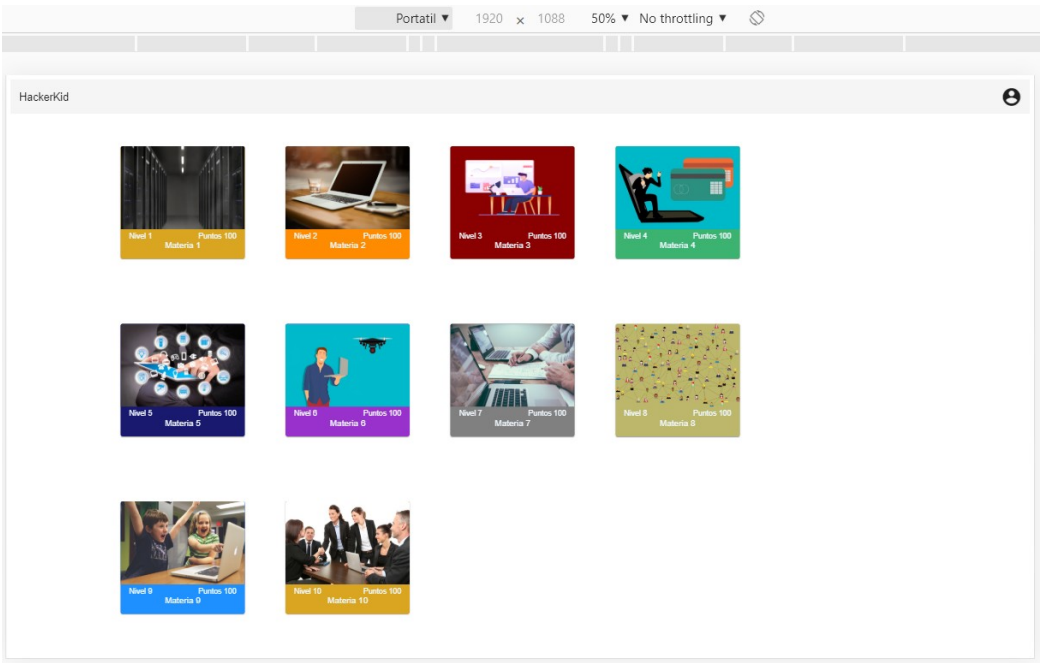


Figura 34: Resolución de portátil

Las anteriores imágenes son ejemplos de como se muestra el contenido de la página con distintas resoluciones, probado con las *DevTools* de Chrome.

- US0007 – Pantalla de dashboard de administradores

  Descripción: Esta historia de usuario tiene como objetivo que el usuario puede elegir entra ir a la gestión de usuarios o a la de contenidos al iniciar sesión.

  Resultado: Pantalla con dos opciones de navegación, *Usuarios* y *Configuración*

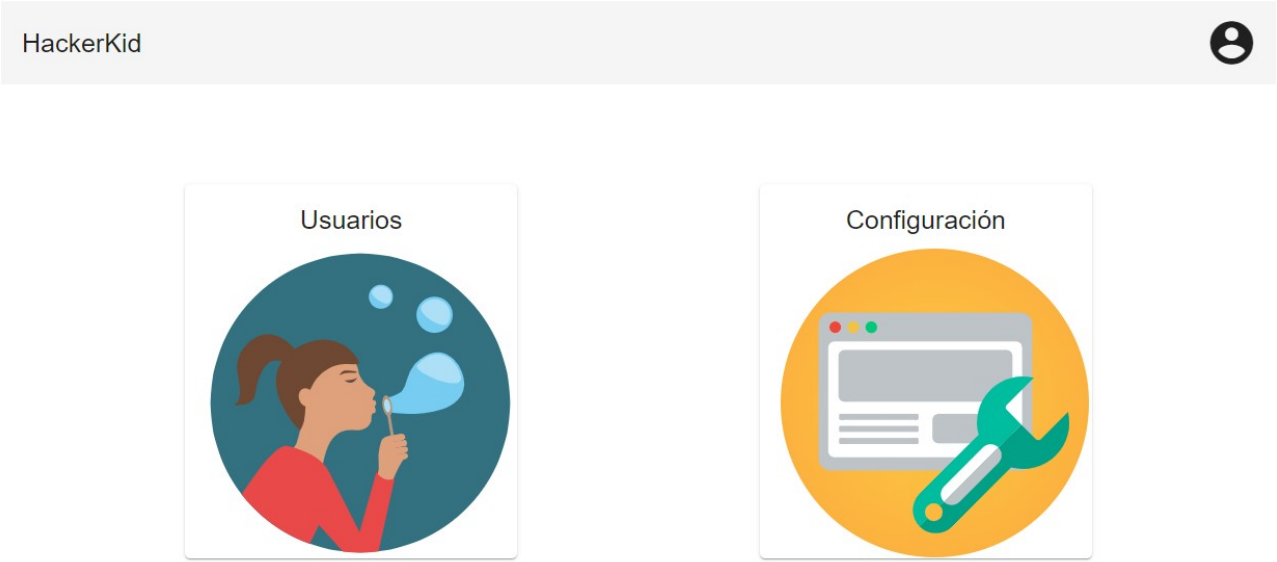


Figura 35: Dashboard de administrador

9.2.6.2 Goal 2 – Configuración de materias

- US0005 – Pantalla de configuración de materias

  Descripción: Esta historia tiene como objetivo el crear una pantalla para la configuración del contenido, inicialmente la gestión de materias

  Resultado: Pantalla con el listado de materias, inicialmente con el borrado y la creación.

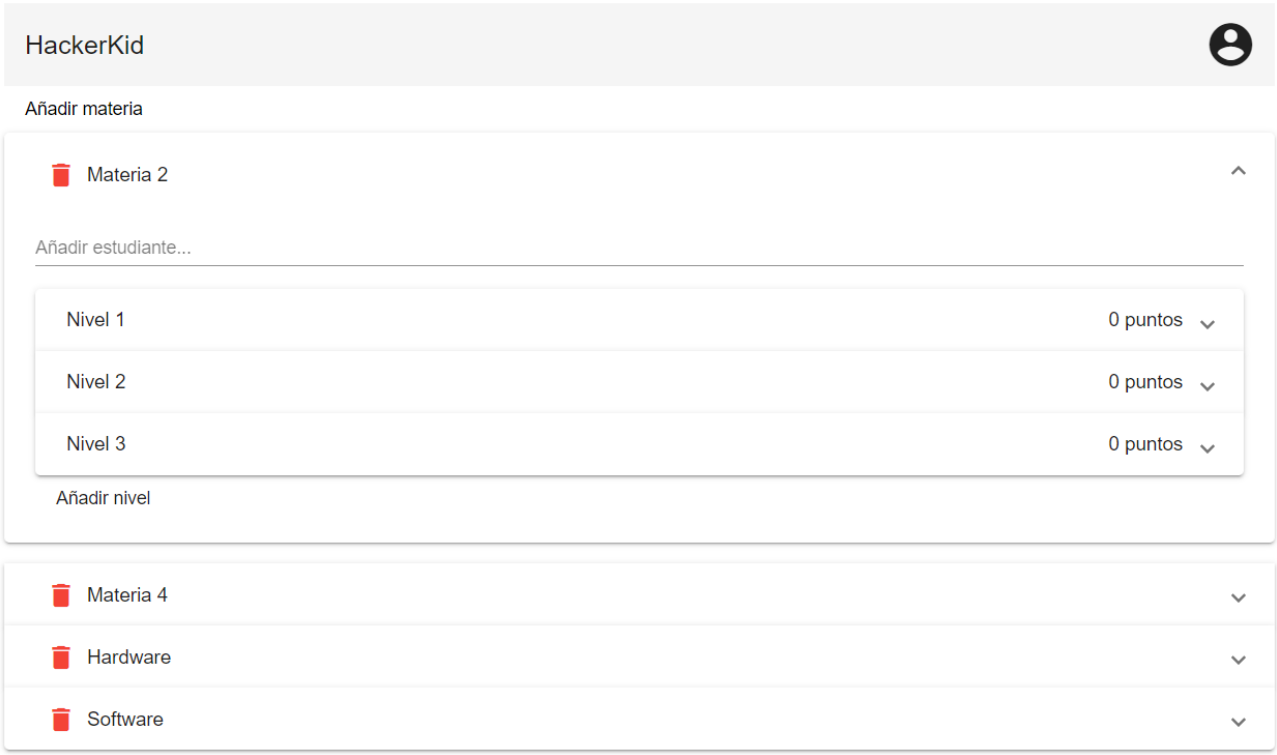


Figura 36: Pantalla de configuración

- US0009 – Añadir nivel

  Descripción: La materias está constituidas por nivel, el objetivo de esta historia es permitir las operaciones con ellos.

  Resultado: Por mantener un diseño sencillo se emplea la misma página para la gestión de materias, niveles y creación de nuevos temas y preguntas. Al crear un nuevo nivel se crearán un tema y una pregunta asociados.

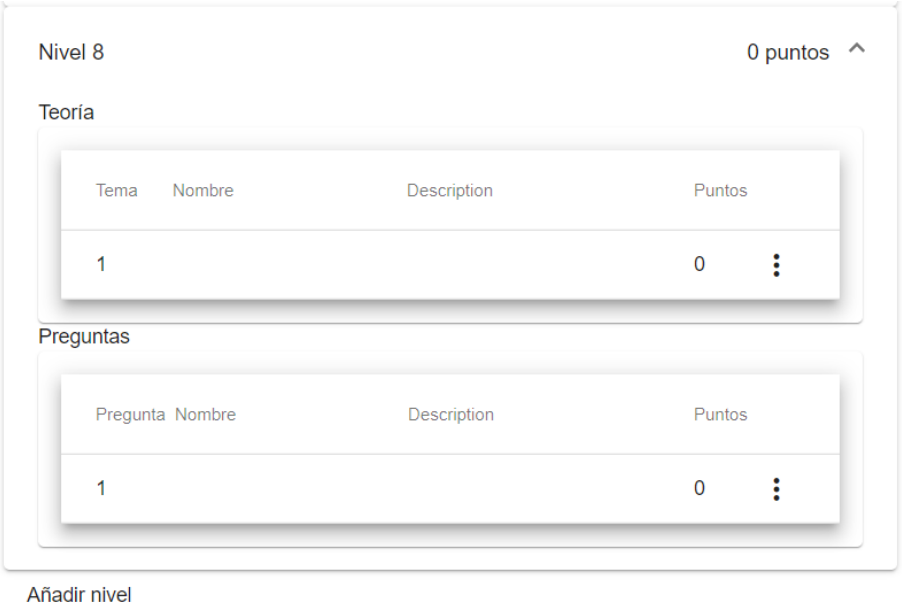


Figura 37: Nuevo nivel

- US0010 – Pantalla de creación de temas

Descripción: El objetivo de esta historia de usuario es la creación y edición de temas. Se incluirá el borrado también

Resultado: Se crea un cuadro de diálogo con las opciones a rellenar, de la imagen se muestra una vista previa

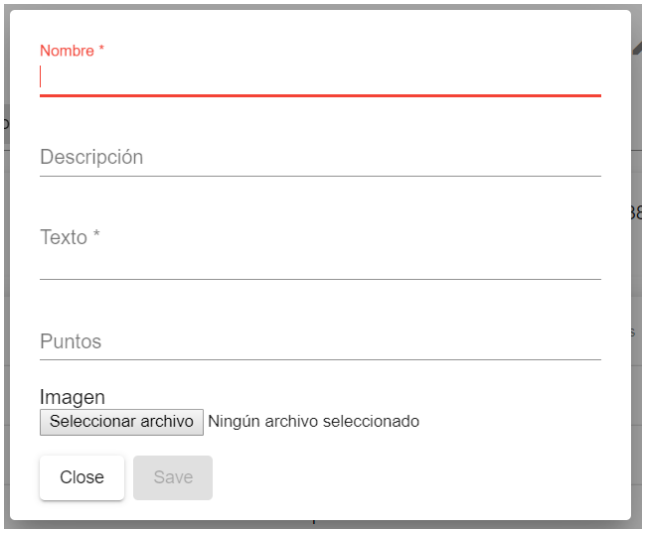


Figura 38: Creación de tema

9.2.7 Retrospective

Resultados de la retrospective

Figura 39: Retrospective sprint 2

9.3 Sprint 3

9.3.1 Análisis

- Inscripción de un estudiante a una materia

Los administradores de la aplicación son los encargados de inscribir a los usuarios a las materias, deberán disponer de la opción de seleccionar usuario y asignarlo a una materia.

- Temas y preguntas

Las materias tienen distintos niveles, dentro de cada uno habrá temas y preguntas. El administrador tiene que poder ver los listados tanto de temas como preguntas y editar su contenido.

- Lectura de temas

Para el usuario debe ser intuitivo navegar entre los distintos temas (unidades didácticas), se hará de manera secuencial y podrá aportar puntos.

9.3.2 Diseño

- Inscripción de un estudiante a una materia

Se debe proporcionar la lista de usuarios de la aplicación y permitir la selección, esto irá incluido como parte de la materia en el listado dentro de la configuración.

- Temas y preguntas

Front-end

En la configuración se deben mostrar en formato tabla, se mostrarán los valores de nombre, descripción y puntos.

Se deben implementar las opciones de editar, borrar y crear nuevo para ambos, tanto temas como preguntas, pudiendo ser mediante botones o mediante un menú, en la propia línea de la tabla.

- Lectura de temas

Back-end

Dado un id de tema debe devolver el listado de todos los temas que del nivel en que se encuentre el del id aportado.

Front-end

Se mostrarán los temas por nivel en el momento que el usuario pinche en una materia. Se deberá cargar todos los temas del nivel en el que se encuentre el estudiante, se recuperarán llamando al servicio creado por el backend.

Se deben mostrar los puntos que tiene el estudiante para esa materia, si los temas tienen puntos se sumarán al acceder el estudiante al tema y se actualizarán en pantalla de manera que el estudiante sepa que ha ganado puntos.

Se llevará registro del id de lo último leído para saber donde se encuentra el estudiante y evitar a su vez sumar dos veces los puntos de un tema.

El último tema debe mostrar un botón para ir a las preguntas.

9.3.3 PBR

Sube la prioridad de las historias relacionadas con la gestión de los temas y las preguntas, pero se opta por dar más prioridad a las previstas para este sprint.

9.3.1.1 Product backlog

Historias de usuario				
Código	Título	Descripción	Prioridad	Talla
US0013	Configuración inicial del usuario	Como estudiante quiero poder configurar mis datos la primera vez que inicie sesión	1	M
US0010	Pantalla de creación de temas	Como administrador quiero poder crear, modificar y borrar temas dentro de los niveles	2	M
US0011	Pantalla de lectura de temas	Como estudiante quiero poder leer los temas de las materias a las que estoy inscrito	2	M
US0012	Pantalla de creación de preguntas	Como administrador quiero poder crear, modificar y borrar preguntas asociadas a un nivel	2	M
US0008	Mover los botones de editar y borrar a un menú	Como usuario quiere poder acceder a las opciones de editar y borrar desde un menú en lugar de botones	5	M

9.3.4 Planning

En este sprint se desarrollará la inscripción de estudiantes a materias, se continuará con las historias de gestión de temas y ejercicios y la pantalla de lectura de temas.

Goal 1 – Inscripción de estudiante	Work items
Inscripción de un estudiante a una materia	US00013
Goal 2 – Configuración de materias	**Work items**
Pantalla de creación de temas	US0010
Pantalla de creación de preguntas	US0012
Goal 3 – Consulta de materias	
Pantalla para lectura de temas	US0011

Figura 40: Goals del sprint 3

9.3.5 Implementación

Front-end

- Inscripción de un estudiante

Al desplegar la materia, en la configuración, se añade como lo primero un listado de chips que tendrá un desplegable con los usuarios de tipo estudiante, de este listado se podrá seleccionar qué estudiantes estarán inscriptos en qué asignatura.

- Pantalla de creación de temas

Se incluye una tabla que mostrará los valores requeridos y un menú en el lateral para proporcionar las opciones de crear, editar y borrar.

La creación y edición se implementa como un cuadro de diálogo que muestra las opciones disponibles.

- Pantalla de creación de preguntas

Al igual que para los temas en el caso de las preguntas también se incluye una tabla que mostrará los valores requeridos y un menú en el lateral para proporcionar las opciones de crear, editar y borrar. También se muestra otra opción para mostrar el diálogo de las respuestas.

La creación y edición se implementa como un cuadro de diálogo que muestra las opciones disponibles.

- Pantalla de lectura de temas

Se implementa como un *stepper* forzando al estudiante a seguir un orden secuencial, no se podrá pasar a la siguiente pregunta sin contestar bien primero la actual.

Back-end

- Pantallas de temas y preguntas

Se aprovechan los método CRUD implementan anteriormente y se exponen los servicios web para permitir las operaciones.

- Inscripción de un estudiante

Se crea un nuevo servicio que guardará los datos del estudiante creando un nuevo resgitro en la base de datos.

9.3.6 Review

9.3.6.1 Goal 1 - Inscripción de usuarios

- US0013 - Inscripción de un estudiante a una materia

  Descripción: El objetivo de esta historia es permitir las inscripción de estudiantes

  las materias.

  Resultado: Se incluye un listado de usuarios para seleccionar en cada materia, se podrán tanto añadir como borrar. En caso de no estar configurada aún la cuenta de estudiante se mostrará el nombre de usuario, en caso de estarlo se muestran nombre y apellidos.

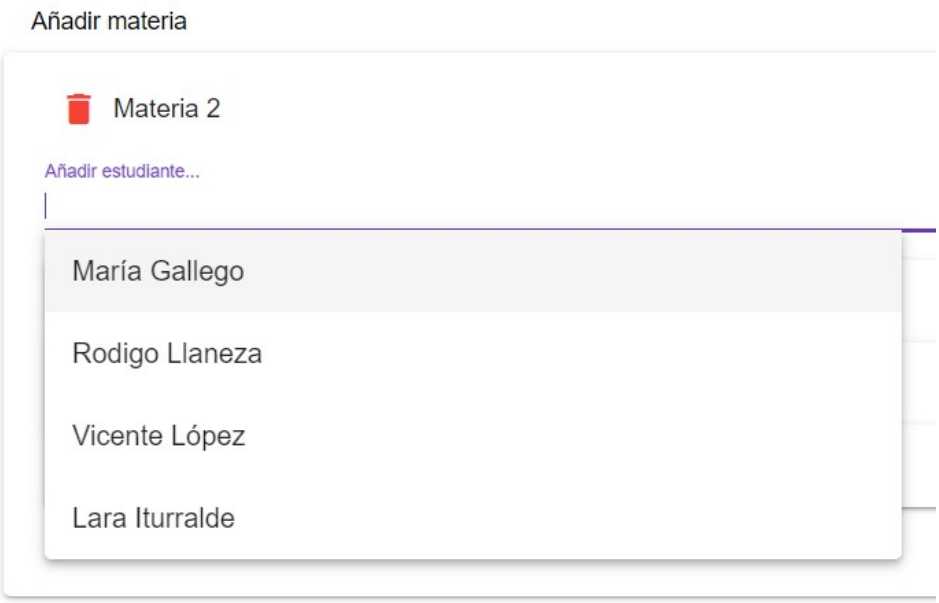


Figura 41: Inscribir estudiante - listado de estudiantes

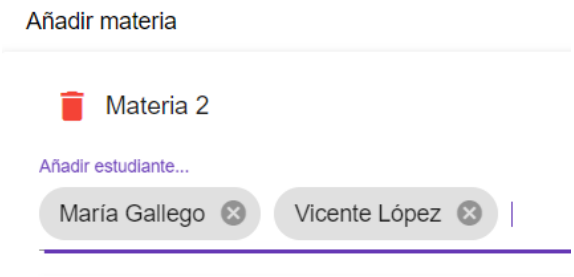


Figura 42: Estudiantes asignados a una materia

9.3.6.2 Goal 2 – Configuración de materias

- US0010 – Pantalla de creación de temas

  Descripción: Historia de usuario creada para incorporar la gestión de los temas en la configuración.

  Resultado: Se añade un listado de temas dentro de cada nivel. Se incluye un menú con las opciones de añadir nuevo, editar y borrar.

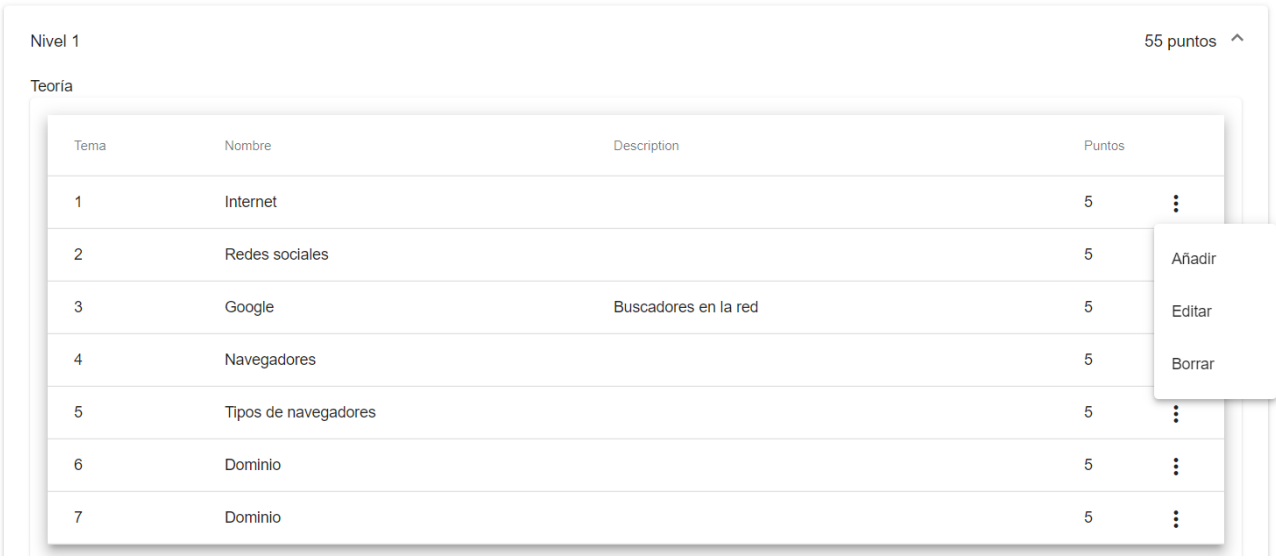


Figura 43: Listado de temas

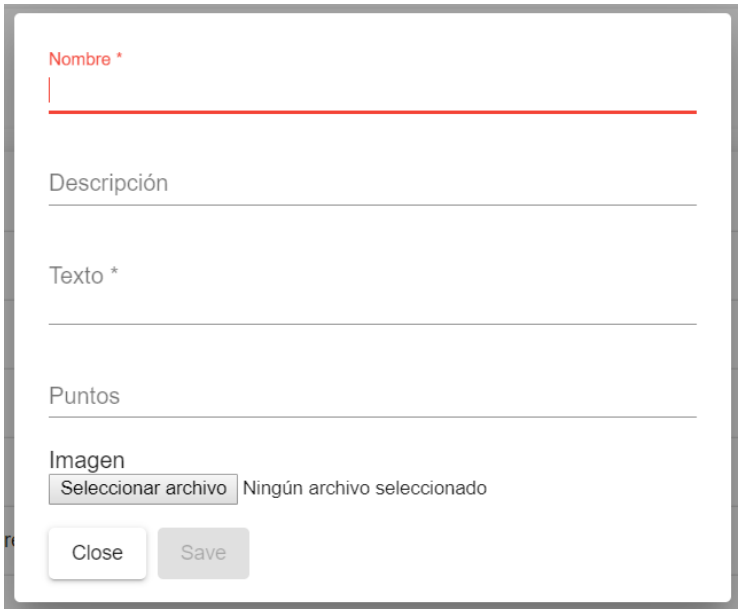


Figura 44: Diálogo de edición de temas - Creación

- US0012 – Pantalla de creación de preguntas

  Descripción: Historia de usuario creada para incorporar la gestión de las preguntas en la configuración.

  Resultado: Se añade un listado de preguntas dentro de cada nivel, justo después de los temas. Se incluye un menú con las opciones de añadir nuevo, editar, borrar y respuestas.

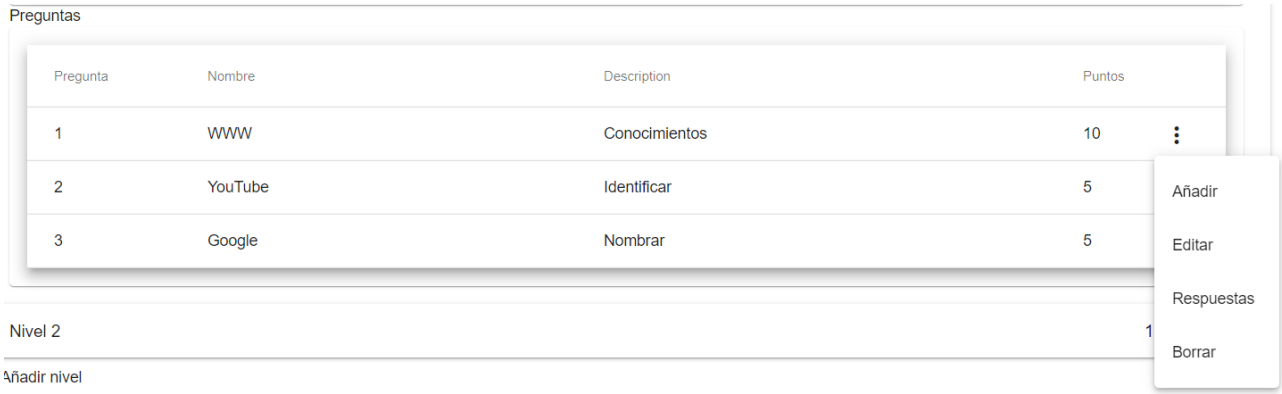


Figura 45: Listado de preguntas

El diálogo de editar pregunta muestra las mismas opciones que el de editar tema, sin embargo la opción de añadir imagen a pesar de que aparece todavía no está disponible, es un “nice to have”.

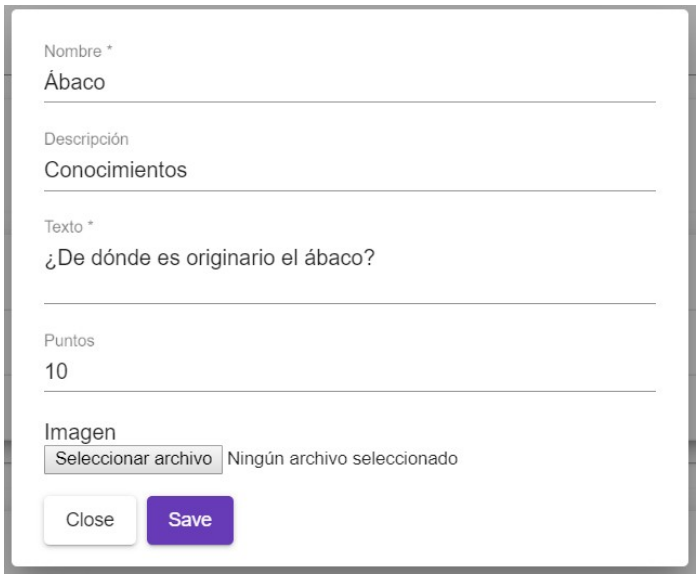


Figura 46: Diálogo de editar pregunta - Edición

9.3.6.3 Goal 3 – Consulta de materias

- US0011 – Pantalla de lectura de temas

  Descripción: El objetivo de esta historia es crear una pantalla para que el estudiante pueda consultar un tema

  Resultado: Se crea una pantalla que permitirá moverse de forma fácil entre los distintos temas. Muestra los puntos actuales del usuario y se ve como se van actualizando.

  Al final se añade un botón para acceder a las preguntas.

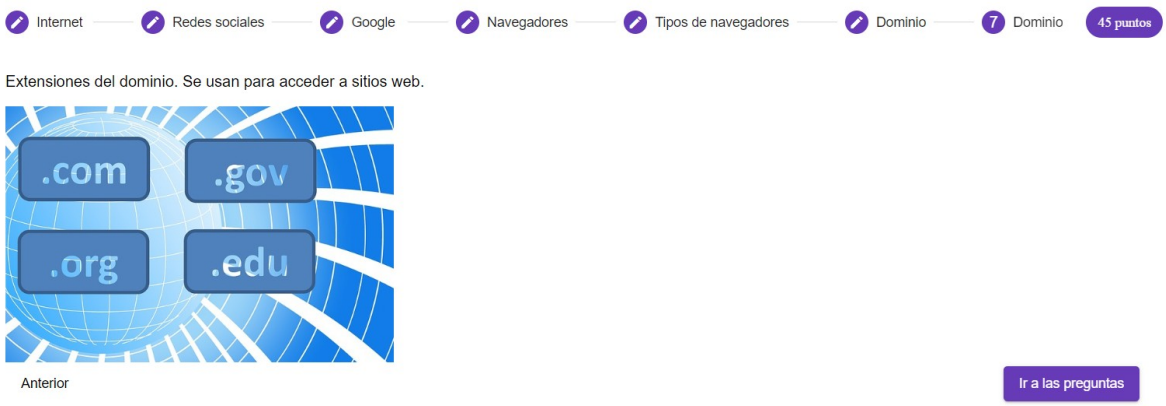
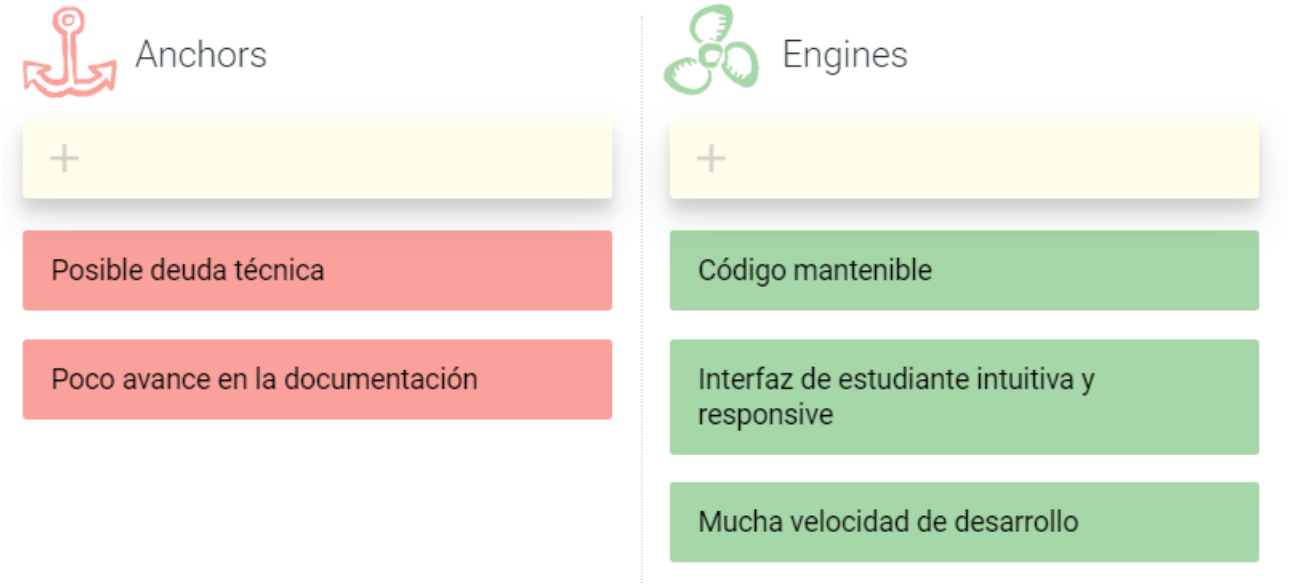


Figura 47: Pantalla de lectura de temas

9.3.7 Restrospective

Resultados de la retrospective



9.4 Sprint 4

En este sprint se implementará la lógica y la pantalla para permitir al usuario elegir respuesta a una pregunta.

No habrá PBR puesto que se trata del último sprint, se decide eliminar el sprint 5 debido a que no aporta valor añadido, las refactorizaciones y mejoras que se habían planificado para ese sprint se llevarán a cabo tras finalizar el sprint 4.

9.4.1 Análisis

Análisis funcional

- Configuración inicial del estudiante

Al iniciar sesión por primera vez el estudiante debe configurar la cuenta, para ello la aplicación mostrará un cuadro de diálogo con varios pasos a seguir, siendo necesario completar uno para pasar al siguiente.

El usuario deberá aportar sus datos para crearse el perfil de usuario, en esta pantalla solo se pedirá nombre y apellidos y se dará la opción de cambiar la contraseña, que inicialmente habíamos guardado con el valor del nombre de usuario, al final se mostrará que todo ha ido bien y los puntos recibidos.

- Edición de respuestas

Cada pregunta tendrá al menos dos respuestas, al menos una de ellas debe ser válida. Se deben poder incluir hasta cuatro y proporcionar la opción de marcar como válida.

La respuesta estará compuesta por un texto, que será la respuesta y un indicador de si es correcta o no.

9.4.2 Diseño

- Configuración inicial del estudiante

Back-end

Recibirá los datos aportados por el usuario en la configuración inicial, para guardarlos se creará un nuevo registro en la tabla de estudiantes. Siguiendo la estructura del ejemplo del login. Se deberá crear un nuevo servicio de actualización.

Front-end

Se deberá pedir al estudiante que ingrese los datos nombre, apellidos y además se le pedirá que cambie la contraseña.

Se deberá mostrar un mensaje al terminar que incluya los puntos obtenidos.

- Edición de respuestas

Back-end

Se deberán crear dos servicios para la creación y edición de respuestas.

Los campos a guardar serán, el id de la pregunta, el texto de la respuesta y un valor booleano que indicará si es correcta.

Front-end

Se debe mostrar un cuadro de dialogo con los campos de las cuatro posibles respuestas, marcando dos de ellas como requeridas y dando la opción de marcar como correcta.

9.4.3 Planning

Se empezará con la configuración inicial de los datos del usuario posteriomente en este sprint se implementará la lógica y la pantalla para permitir al usuario elegir

respuesta a una pregunta y terminará con las correcciones que se encuentren y mejoras de estilos.

Goal 1 – Configuración del perfil de usuario	Work items
Edición de los datos del estudiante	US00014
Goal 2 – Gestión de las repuestas	**Work items**
Elección de respuestas	US0015
Goal 3 – Correcciones y mejoras de estilos	
Correcciones y mejoras de estilos de la aplicación	US0016

Figura 48: Goals de sprint 4

9.4.4 Implementación

Front-end

- Configuración inicial del usuario

Se crea un cuadro de diálogo de Angular material, con un componente Stepper también de Angular material, este componente pide los datos siguiendo unos pasos, primero pide nombre y apellidos, luego el cambio de contraseña y después mostrará el mensaje de verificación requerido, es un diseño simple e intuitivo que deja claras las opciones sin falta de mensajes explicando el funcionamiento ni comprobaciones adicionales.

- Elección de respuestas

Se incluyen las opciones de respuesta en las preguntas, se muestra en rojo y con mensaje si es incorrecta y en verde el mensaje de si es correcta.

Se deshabilita el botón siguiente hasta que la respuesta es correcta.El botón siguiente o de subir nivel estará deshabilitado si la opción es incorrecta.

- Test

Se incluye el framework Mockito al back-end para realizar test unitarios, versión 1.9.5.

Se cambian los test del fron-end de jasmine a jest

- Refactorings

Se extrae la lógica de los temas a su propio módulo, así como la de las preguntas.

9.4.5 Review

9.4.5.1 Goal 1 - Configuración inicial del usuario

- US0014 Edición de los datos del estudiante.

  Descripción: Como estudiante quiero configurar mi cuenta añadiendo mis datos y también cambiar la contraseña inicial.

  Resultado: Se muestra el diálogo de configuración de la cuenta la primera vez que el estudiante inicia sesión. Se muestra como varios pasos a seguir, obligatorios para entrar en la aplicación.

Figura 49: Configuración inicial de usuario

9.4.5.2 Goal 2 - Gestión de las respuestas

US0015 Elección de respuestas

Descripción: Esta historia de usuario tiene como objetivo el proporcionar al estudiante la manera de contestar a las preguntas.

Resultado: Se añaden las respuestas a las preguntas en la página de preguntas. Se muestran los puntos actuales y se actualizan con los obtenidos al contestar preguntas.

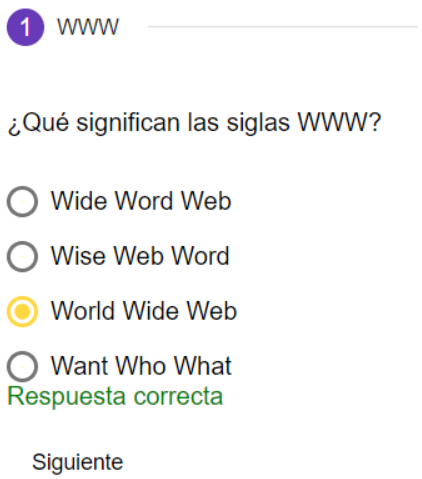


Figura 50: Pregunta con respuestas

9.4.5.3 Goal – Correcciones y mejoras de estilos

- Cambios de estilos
  - Se cambian de sitio los botones de editar y borrar materias, además se cambia el color a uno menos llamativo, ya que no son elementos que necesiten resalte.
  - Se aplica el color principal del tema, que se había añadido al principio, a la barra de herramientas.

Figura 51: Nuevo diseño de la configuración de materiales

- Se incluyen los estilos del tema principal en los cuadros de diálogo

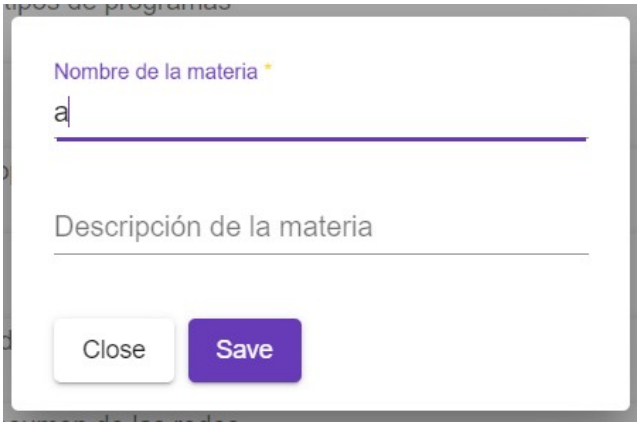


Figura 52: Diálogo de editar materia con los estilos del tema

10 Anexos

- Mapa de navegación

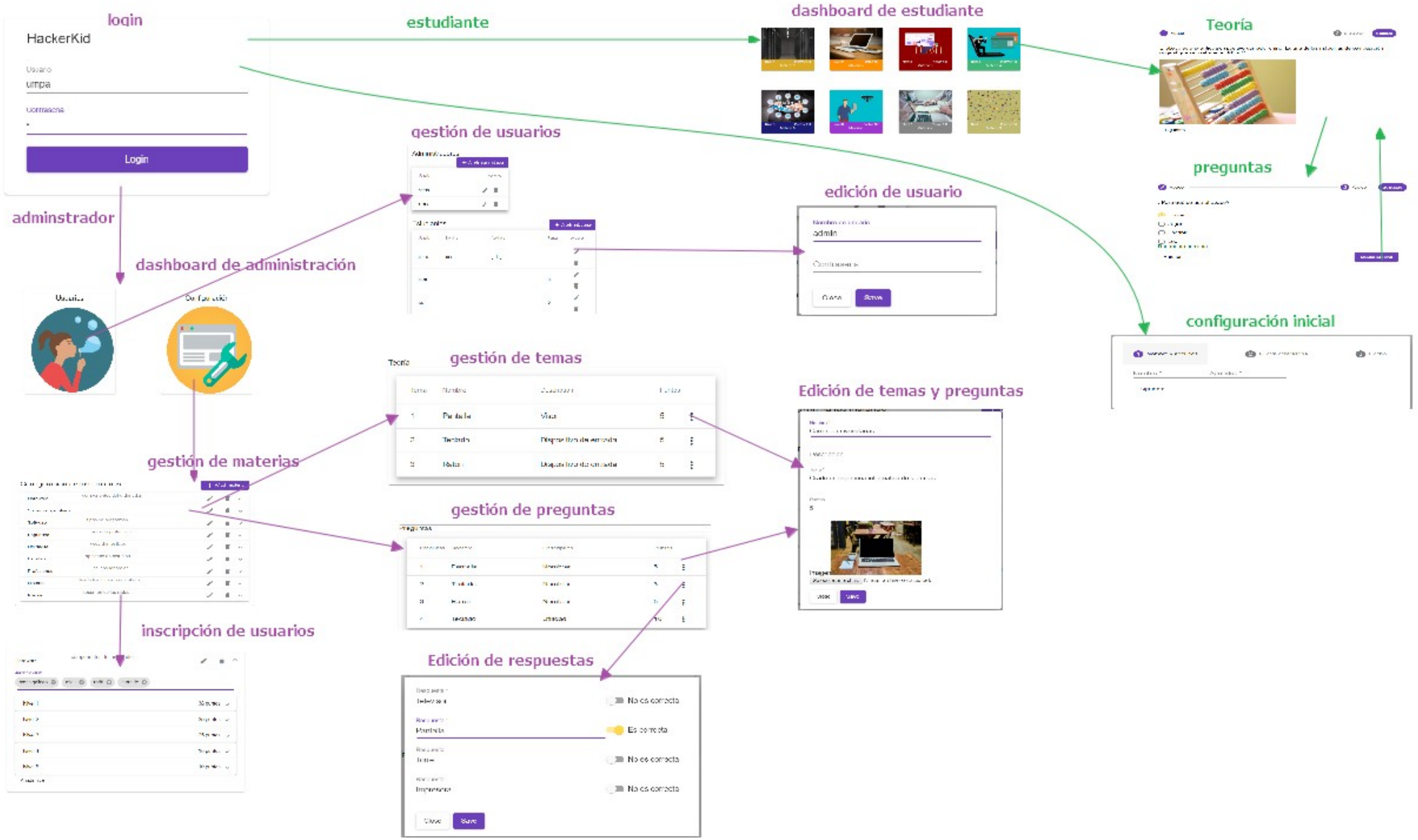
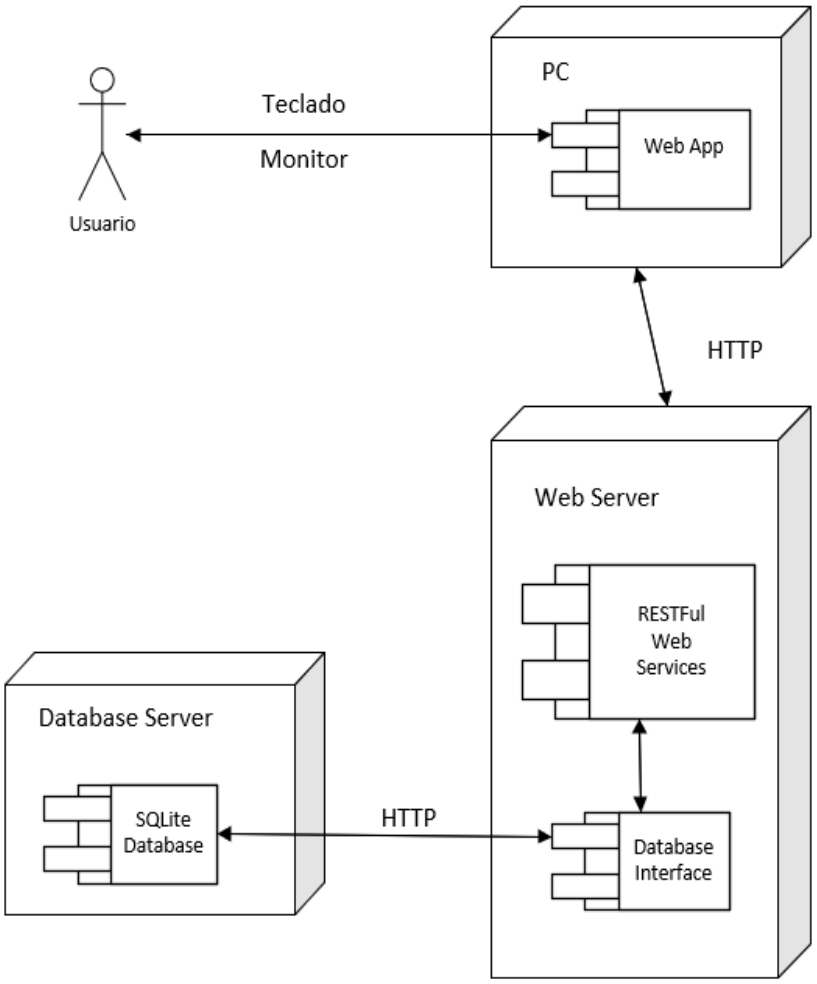


Figura 53: Mapa de navegación

Se decide realizar un mapa de navegación para mostrar una visión global de la navegación por las pantallas y las distintas opciones de los usuarios, se marcan en verde las opciones del estudiante y en morado las del administrador.

- Diagrama de despliegue



11 Posibles mejoras

- Incluir branding: Una idea sería un logo con un superhéroe
- Internacionalización: Utilizar ficheros de recursos para permitir el cambio de idioma.
- Gamificación: Incluir minijuegos educativos, para reforzar el aprendizaje.

12 Glosario

API: Interfaz de programación de aplicaciones

Back-end: Parte que procesa la entrada desde el Front-end

CRUD: Create, read, update y delete, se usa cuando se hacen operaciones de creación, lectura, actualización y borrado de una entidad.

DevTools: Herramienta del navegador para desarrolladores

Diseño adaptativo o responsive: tipo de diseño web que busca la correcta visualización en distintos dispositivos.

Front-end: Parte del software que interactúa con los usuarios

Mockito: Framework para test unitarios en Java

REST/RESTful: Representational State Transfer

SQL: Lenguaje de consulta estructurada, usado para administrar y recuperar datos de la base de datos.

13 Bibliografía

Wikipedia: https://es.wikipedia.org

REST naming convections: https://restfulapi.net/resource-naming/

Imágenes (libres): https://pixabay.com

Imágenes del dashboard de administrador: https://www.flaticon.com/packs/seo-and-web

Angular: https://angular.io

Mockito: https://site.mockito.org/

CRUD con SQLite: https://www.codexpedia.com/java/crud-create-read-update-and-delete-in-sqlite-with-java/

Retrospective online: https://www.teamretro.com/

GitHub: https://github.com/