



## Proyecto de fin de ciclo 2021-2022

Ciclo de Grado Superior: Desarrollo de Aplicaciones Web

# LEARN-INF

Web de enseñanza interactiva



Autora: Natalia Cristóbal Rodríguez

Curso académico 2021-2022

Fecha de entrega: 14/06/2022

## Índice

### 1. Introducción

- ¿Qué es Learn-INF? ..... 2
- Módulos implicados en el desarrollo del proyecto ..... 2

### 2. Justificación del proyecto y objetivos.

- ¿Por qué este proyecto? ..... 3
- Referencias tomadas del mercado actual ..... 3
- Objetivos ..... 6
- Perfil del público ..... 6

### 3. Planificación ..... 7

### 4. Parte experimental

#### 4.1. Análisis. ¿Qué hará la aplicación?

- Diagrama del modelo de datos ..... 9
- Prototipado: Diseño de pantallas ..... 11

#### 4.2. Diseño. ¿Cómo se hará la aplicación?

- Estructura de la base de datos ..... 15
- Arquitectura del proyecto ..... 15

#### 4.3. Implementación y pruebas

- Desarrollo de la aplicación ..... 17
- Documentación de pruebas realizadas ..... 19

#### 4.4. Implantación y documentación

- Manual de instalación y uso ..... 22

#### 4.5. Resultados y discusión ..... 28

### 5. Conclusiones ..... 29

### 6. Bibliografía y referencias ..... 30

### 7. Anexos ..... 31

## 1. Introducción

- **¿Qué es Learn-INF?**

“Learn-INF” consiste en una web interactiva para niños con el objetivo de que aprendan los principios básicos de la programación. Con una dinámica parecida a un videojuego de preguntas y respuestas, pero con logros y un diseño dinámico para ellos. El nombre es una combinación de la palabra “**Learn**”, **aprender** en inglés, y de la abreviatura “**INF**”, la cual procede de **INFormation**. El planteamiento inicial sería:

- La web dispone de distintos “**mundos**” con temáticas relacionadas con la programación y sus lenguajes. Estos a su vez se dividen en **niveles**, donde se aumenta su dificultad según se vaya avanzando.
- Los niveles contienen **lecciones** y **tests**. Cada test se compone de **preguntas** que harán que el usuario consiga puntos para desbloquear los demás niveles y de **recompensas** extra (iconos de usuario, diplomas, trofeos, etc.). Cada nivel y mundo tendrá una **característica de desbloqueo**.
- Los **usuarios** dispondrán de un panel personal con su información y logros. Entre las distintas vistas de la web se comprobará el tipo de usuario que tenga la sesión activa, así se podrá controlar el acceso a las distintas áreas y mostrar los datos personalizados del usuario.

- **Módulos implicados en el desarrollo del proyecto**

- **Base de Datos** : Para el planteamiento, almacenamiento y uso de datos necesarios para la construcción de la web. Se realizaron los diagramas apropiados para concretar la estructura y relaciones de los elementos, también se concretaron los atributos.
- **Diseño de Interfaces Web**: La interfaz visual de la web. Primeramente, se realizó los prototipos del diseño, también se planeó el diseño responsive de la web.
- **Desarrollo Web Entorno Cliente / Lenguaje de Marcas** : Estos módulos permitieron tanto dar una mayor consistencia al diseño de interfaz como al procesado de datos recogidos desde la parte de cliente a la del servidor y viceversa.
- **Desarrollo Web Entorno Servidor**: Para controlar el Backend de la manera más eficaz y sencilla posible. La estructura a seguir para la organización del proyecto, recogiendo todo tipo de elementos tanto visuales como datos necesarios en su generación.
- **Despliegue de Aplicaciones Web**: El despliegue permitirá que la aplicación sea visible desde un servidor, la conexión de la plataforma web con la base de datos, su gestión y cambios que pueden realizarse a través de ella.

## 2. Justificación del proyecto y objetivos

- **¿Por qué este proyecto?**

Esta temática me permitió desarrollar todo lo que hemos aprendido en el curso, ya que el propósito de este proyecto ha sido abarcar tanto la parte Frontend a nivel de diseño interactivo y entendible para el público al que está dirigido, como el Backend para la gestión de niveles y seguridad.

Es una manera lógica de demostrar la programación y flujo que lleva cualquier plataforma web. A su vez, como la dinámica de uso se basa en un funcionamiento parecido al de un videojuego, la lógica a la hora de programar ha podido tomar más funcionalidades y hacerla más compleja. También ha permitido implementar una seguridad y control de acceso, ya que esta web requiere de la creación y confirmación por usuario. El paso de este usuario se controla, y eso hace delimitar su paso en ciertas zonas de la web.

- **Referencias tomadas del mercado actual**

Actualmente, disponemos de una gran cantidad de herramientas en el mercado para aprender sobre la programación y, de manera autodidacta, llegar a tener una buena base de conocimientos en este campo para todos los niveles, desde el más básico hasta para formaciones más profesionales.

Encontramos un amplio catálogo de cursos disponibles en múltiples plataformas como OpenWebinars, Domestika, Platzi, etc. Estas webs están dirigidas en su mayoría a personas adultas, aunque sea de un nivel básico, no es lo mismo enseñar a infantes o adolescentes a programar.

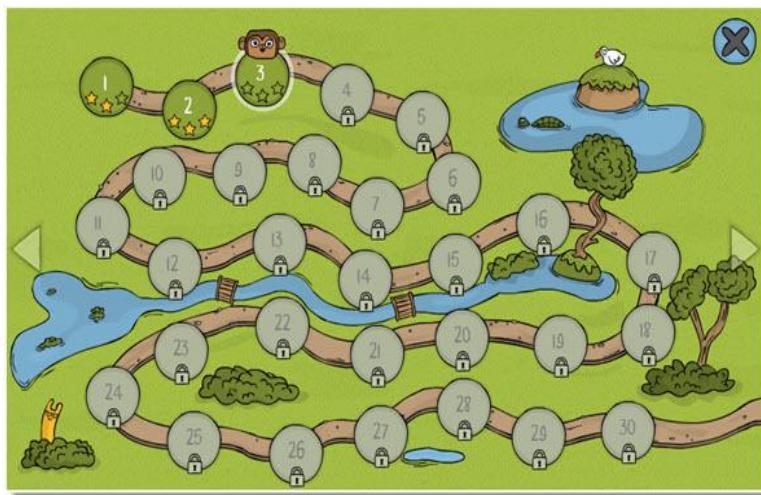
Para este aprendizaje en edades tempranas se tiene que contemplar que la edad media en la que los pensamientos abstractos no son tan complejos es a partir de los 10 a los 15 años. Esto se debe a que, aunque se puede enseñar de una manera sencilla instrucciones de uso lógicas, el cerebro no está aún lo suficientemente desarrollado para comprenderlas.

Por ello, lo más normal es que las webs de aprendizaje para infantes/adolescentes sean una combinación de ejercicios de lógicas y juegos de programación lo más sencillos posibles, con estéticas parecidas a la de los videojuegos.

Aprender a programar estimula la creatividad, enseña a trabajar en equipo y ayuda a desarrollar un pensamiento lógico. La programación es una habilidad fundamental a día de hoy y es interesante adquirirla desde una edad temprana.

Unos ejemplos de webs que utilizan estas dinámicas de enseñanza son las siguientes:

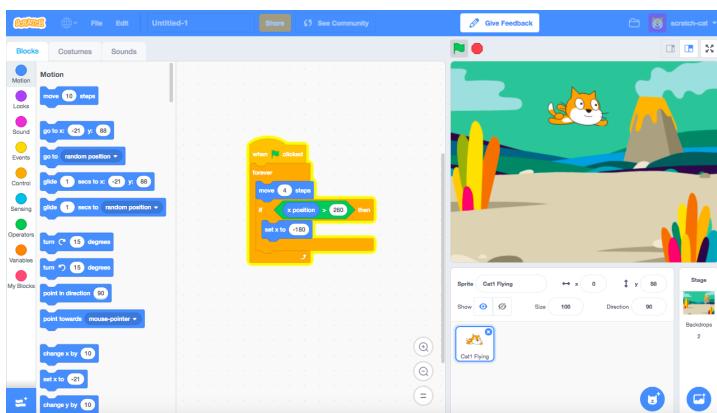
→ **Code Monkey**



Esta web nos muestra la lógica que se lleva a cabo cuando programamos de una manera sencilla y muy enfocada a la dinámica de los videojuegos. Es una buena manera de enseñar a los infantes a solucionar problemas utilizando lógica básica, pero no es programación como tal.

Su diseño también es un buen referente para este proyecto, ya que está enfocado no solo para el entendimiento, sino para la diversión para los niños.

→ **Scratch**



Una de las más conocidas dispone de un amplio catálogo de ejemplos, ideas y ejercicios de programación más enfocados a la realidad de esta. Los ejercicios se llevan a cabo mediante la lógica de bloques, la cual ayuda a comprender las interacciones y posibilidades que

tienen disponibles a la hora de crear una aplicación. Al igual que la anterior web, su diseño dirigido a los más pequeños llama la atención y facilita el aprendizaje.

## → Coding Giants



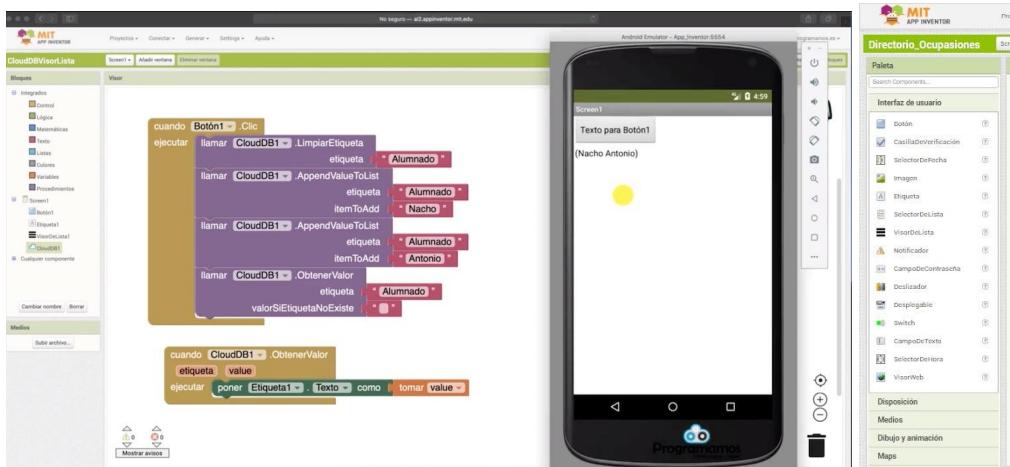
Esta web contiene un catálogo de cursos individuales para todas las edades. Muchos de ellos están basados en el diseño y desarrollo de videojuegos, hay unos más teóricos y otros puramente prácticos semejantes a la realidad, una buena combinación de métodos para las edades a las que están dirigidos.



## → MIT App Inventor



Nos permite crear aplicaciones móviles desde cero, con lógicas de programación en bloques muy similares a lo que puede ser la programación real, al igual que Scratch.



- **Objetivos**

La idea fue dar más peso al Frontend, ya que el funcionamiento de la aplicación en sí puede resultar más sencilla en su planteamiento. Al ser una web para niños y adolescentes, es lógico que el diseño fuera más llamativo, con mayor colorido, pero sin dejar atrás una buena estructura de la plataforma, que sea intuitiva y comprensible para ellos. También esto llevó a un mayor estudio y planificación, ya que es una gran parte de mercadotecnia a la hora de exponer el proyecto. En definitiva, la mayor parte del estudio estuvo enfocado en la interfaz del usuario para una experiencia más fluida, sencilla y divertida para el sector al que está enfocado.

Relacionado con esto, estudié el diseño UX/UI mediante la herramienta de Adobe XD para crear el prototipo de la aplicación y el diseño responsive.

Para el desarrollo de esta aplicación, trabajé con la herramienta de framework de Jakarta EE ya que fué útil de manejar para el código Java y mejorar el rendimiento.

Como base de datos usé MySQL. Para complementar su creación, implementé el uso de JPA, ya que es un framework que he encontrado muy útil en el curso.

Para la parte del Frontend, aparte de JavaScript y Bootstrap, utilicé unas herramientas que, a pesar de estar incluidas en el curso, no llegamos a desarrollar tanto. Estas han sido el Ajax, el compilador de Sass y JQuery.

- **Perfil del público**

El público al que está enfocada la aplicación es para una edad de 11 años en adelante. Es la mejor edad para empezar este aprendizaje debido a que se empieza a tener un pensamiento más desarrollado y un entendimiento del entorno. La estética de la web puede ser más dinámica sin perder la sencillez.

No ha sido posible que la web se convirtiera en un videojuego por completo, puesto que esto habría complicado el proyecto y no era viable para el tiempo disponible. Los test que se presentan son una combinación de preguntas teóricas y ejercicios visuales.

Al ser una aplicación dirigida a estudiantes, los profesores a cargo de estos pueden tener un historial de sus alumnos y ver su desarrollo según avancen con las lecciones en la web. Al no ser el público principal al que va dirigido, no era necesario cambiar la estética de la plataforma, simplemente es una adición a la estructura principal para la web con opciones no disponibles para los alumnos.

### 3. Planificación

Para la planificación y la distribución de las tareas he usado la plataforma de Taiga, un software de organización de proyectos. La estructura ha sido la siguiente:

→ **Épicas:** Ordenadas en fases, son la agrupación y división por temas de los distintos puntos a entregar. La división que he realizado sería:

- **Definición del proyecto y primeros esquemas** (Código de color: **rojo**)

Definición inicial del proyecto, necesaria para la creación base de este.

Realizar pruebas con tecnología no vista durante el curso: El prototipado con Adobe XD.

- **Despliegue de la aplicación** (Código de color: **amarillo**)

Despliegue de la aplicación en un servidor web. Al ser una aplicación en Jakarta, se realizará mediante Docker junto a los medios disponibles de MySQL, Wildfly y SFTP en este.

- **Arquitectura de la aplicación** (Código de color: **naranja**)

Backend de la aplicación. La estructura se programó en Jakarta EE.

- **Diseño de la aplicación** (Código de color: **verde**)

Frontend del proyecto. Siguiendo con los prototipos creados previamente se buscará el diseño estructurado de la página principal, login y registros nuevos, perfiles diferenciados de alumno y profesor, listado de mundos y niveles, lecciones, tests y recompensas.

- **Testing JUnit** (Código de color: **azul**)

Testing de las funciones de la aplicación mediante JUnit5.

- **Memoria del proyecto** (Código de color: **morado**)

Redacción de los puntos a entregar. Guías de usuario y despliegue de la aplicación.

- **Revisión final** (Código de color: **negro**)

Comprobación de funciones y despliegue final correcto. Documentación redactada por completo.

→ **Sprints:** Los tiempos estipulados para realizar determinadas partes del proyecto. Son un total de 6 intervalos, cada uno engloba unas determinadas historias de usuario:

- Planificación inicial - Prototipos - Esquemas
- Despliegue de la aplicación
- Primera versión básica de la web
- Necesidades avanzadas de la web
- Despliegue final - Revisión de funciones
- Revisión final

## → Historias de Usuario y Tareas

Los objetivos y puntos necesarios para el desarrollo del proyecto. Las historias de usuario son requisitos que pediría el supuesto cliente para la aplicación. Están relacionados a una Épica en concreto, y a su vez delimitados en un Sprint. Cada una de ellas cuenta con tareas por realizar. Hay tareas sin una historia concreta, los esquemas y documentaciones pertinentes para el proyecto estarían catalogadas aquí. Las tareas extras que tienen que ver más con bug o mejoras estarían especificadas en la pestaña de Peticiones.

The screenshot displays the Taiga.io application interface, specifically the 'User Stories' section. It shows a hierarchical structure of user stories across several epics:

- #15 Definición del proyecto y primeros esquemas**
  - #54 Prototipado inicial de la aplicación (red dot)
  - #55 Realización de diagramas de BD (red dot)
- #16 Despliegue de la aplicación**
  - #87 Despliegue de la Aplicación basico (yellow dot)
  - #126 Despliegue de la Aplicación avanzado (yellow dot)
- #17 Arquitectura de la Aplicación**
  - #65 Login + Signup (orange dot)
  - #71 Listado de niveles (orange dot)
  - #85 Test + Resultados (orange dot)
  - #108 Login + Signup (orange dot)
  - #109 Test + Resultados (orange dot)
  - #120 Opciones Profesor (orange dot)
- #18 Diseño de la aplicación**
  - #65 Login + Signup (orange dot)
  - #72 Pantalla principal (green dot)
  - #71 Listado de niveles (orange dot)
  - #85 Test + Resultados (orange dot)
  - #108 Login + Signup (orange dot)
  - #120 Opciones Profesor (orange dot)
- #19 Testing JUnit + Jasmine**
  - #110 Pruebas JUnit (blue dot)
- #20 Memoria del proyecto**
  - #91 Primera mitad Memoria (purple dot)
  - #97 Segunda mitad Memoria (purple dot)
- #66 Revisión final**
  - #67 Demo final (black dot)

Historias de Usuario desde la estructura de las Épicas, vía Taiga.io

The screenshot shows the 'Peticiones' (Requests) section of the Taiga.io interface. It lists several tasks with their corresponding epic, priority, and status:

TIPO	GRAVEDAD	PRIORIDAD	PETICIÓN
●	●	●	#116 Cambio de posiciones SignUp
●	●	●	#107 Desbloqueo de niveles por puntuacion
●	●	●	#125 Mejora vistas perfil
●	●	●	#124 Iconos botones
●	●	●	#117 Mensajes personalizados HTML5 validacion
●	●	●	#102 Cambiar formas de login
●	●	●	#90 Repaso vista movil - Test
●	●	●	#89 Cambiar BD - Lección

Peticiones realizadas en el proyecto, vía Taiga.io

## 4. Parte experimental

### 4.1. Análisis. ¿Qué hará la aplicación?

- **Diagrama del modelo de datos**

Lo principal para el planteamiento de la base de datos de este proyecto fue identificar los factores que participarán. Al ser una web en la cual van a interactuar como mínimo un tipo de usuario, partí desde este y observé las interacciones y necesidades que tendría dentro de la plataforma. Por ejemplo, si un usuario necesita una identificación exclusiva para la web, si un nivel está habilitado para este, cuántas recompensas ha desbloqueado, etc. La base de datos para este proyecto debía comprender un mínimo de entidades:

→ **Usuarios:** Se identificarán por un código de alumno. Tendrán la opción de determinar su nombre, correo, icono y contraseña en la creación de su cuenta. Cada alumno tiene una puntuación que aumentará según realice los test disponibles. Están asignados a un profesor en concreto, y pueden desbloquear una variedad de mundos distintos.

→ **Profesores:** Se identificarán por un código de profesor, correo, contraseña y su nombre. Pueden tutelar a más de un alumno, y no tienen un registro como tal.

→ **Mundos:** Los mundos se identifican por un código y una temática única. Tienen una dificultad y unos puntos de desbloqueo completos. Contienen un listado de varios niveles.

→ **Niveles:** Los niveles, al igual que los mundos, disponen de un código, la dificultad y los puntos de desbloqueo. También cuentan con el atributo de puntuación mínima para superar el nivel. Los niveles contienen una lección y un test único. Pueden desbloquearse una o varias recompensas por cada nivel.

→ **Recompensas:** Tienen un tipo definido (avatares, diplomas, medallas...), una ruta donde se encuentran y si están desbloqueadas o no.

→ **Lecciones:** Cuentan con un contenido y un código que las identifica.

→ **Test:** Tienen un código y su estado finalizado o por realizar. Los test están compuestos por varias preguntas.

→ **Preguntas:** Las preguntas tienen un código y un texto asociado. Pueden contar con una imagen para ayudar a la comprensión del alumno. Las preguntas también están asociadas a varias respuestas.

→ **Respuestas:** Las respuestas se identifican por un código y un texto. Cuentan con el atributo de si son correctas o no, que ayudará a sumar los puntos totales del test.

## Diagramas finales

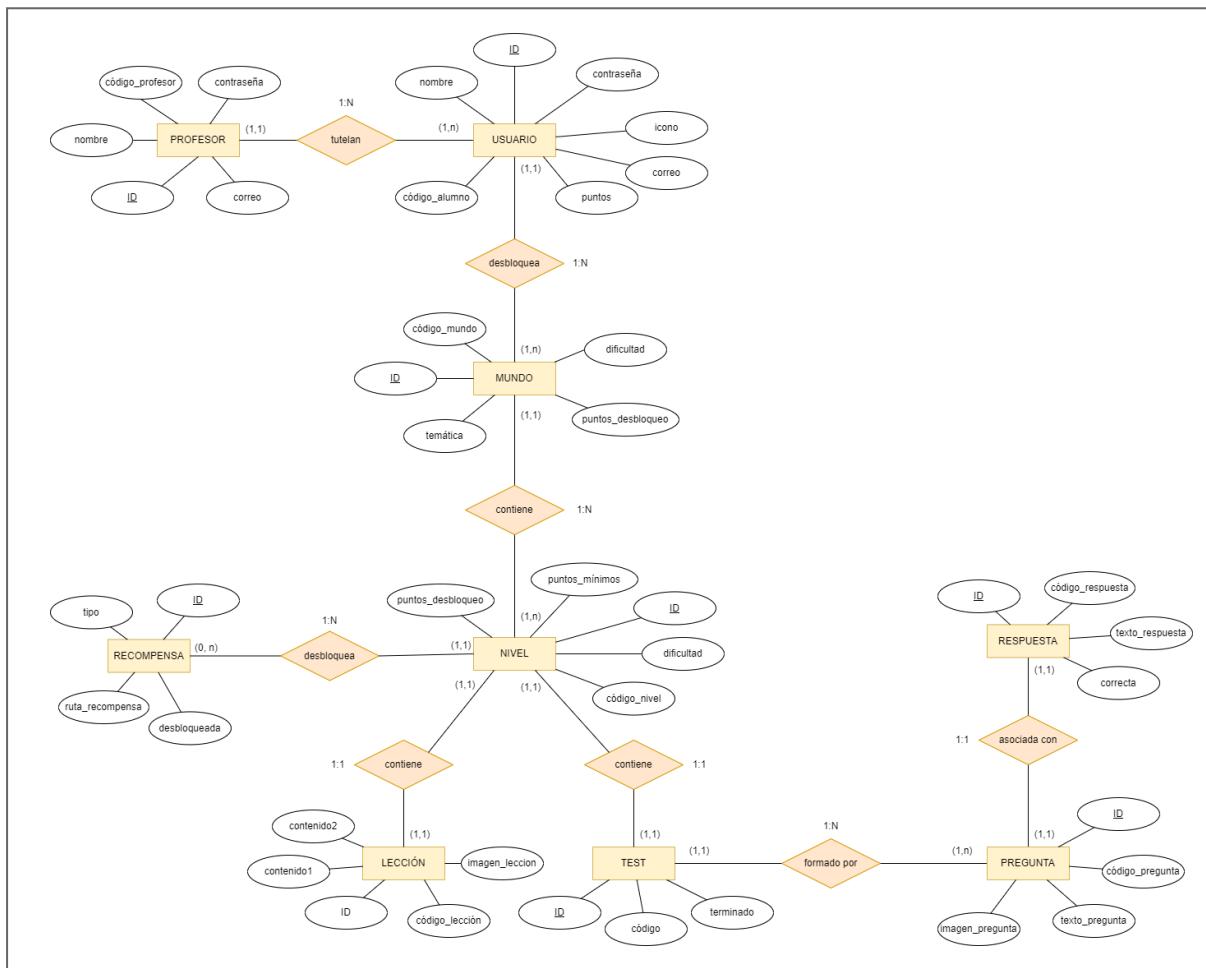


Diagrama de Entidad/Relación

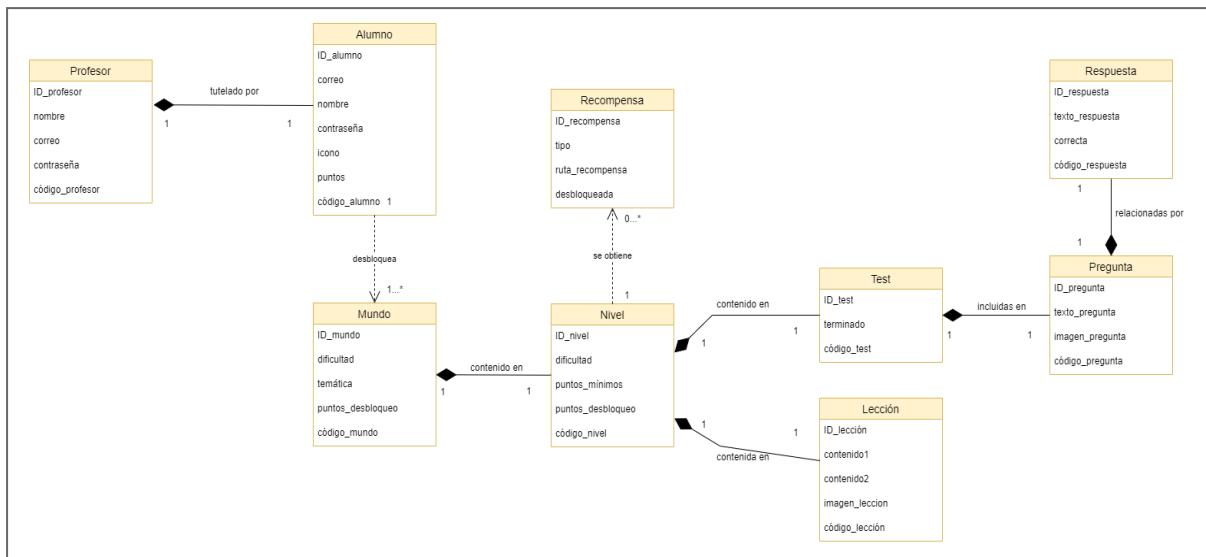


Diagrama de clases

- **Prototipado: Diseño de pantallas**



Para este punto, he decidido incorporar el uso de una tecnología que no hemos visto durante el curso. Adobe XD es un editor de gráficos vectoriales perteneciente a la suite de Adobe Inc para diseñar prototipos de páginas web y aplicaciones móviles. El uso de esta herramienta de diseño UI/UX facilita mucho la creación desde cero de la plataforma web, debido a las múltiples guías, agrupaciones de componentes y demás herramientas de personalización que posee el editor, con opción a incluir nuestras propias preferencias y elementos que queramos usar.

Es muy similar a las demás suites de Adobe como Photoshop o Illustrator, la gran diferencia con las aplicaciones de mock-up más básicas es que XD nos ofrece dar una dinámica de flujo entre pantallas, es decir, podemos crear visualizaciones de qué pasaría si hacemos una determinada acción como clickear o pasar por encima de un componente de la maqueta.

Creo que la aplicación aportó mucho a esta primera fase, no solo permitiendo una visualización básica de la plataforma creada, sino dando dinamismo a los componentes que queramos incorporar. También a la hora de diseñar los distintos formatos que podemos llegar a visualizar, hace que el diseño responsive sea más fácil de coordinar y crea una imagen más sólida de nuestra aplicación.

Para este proceso empecé diseñando la interfaz para un navegador web estándar. Las principales vistas serían una pantalla de inicio, creación de una cuenta o inicio de sesión, perfil de usuario, mundos, niveles y finalmente los tests.

En cada una de las pantallas se contará siempre con una barra de navegación, que facilitará el inicio/creación de la cuenta, la vista al perfil y el regreso a la pantalla inicial. Cada pantalla contiene la información acorde a su función.

Una pauta que he tenido en cuenta a la hora del diseño en estas ha sido el sistema de rejillas de Bootstrap. Al hacer esto, la estructura es simétrica, ordenada y facilita la lectura de la web. También hace que el diseño responsive haya sido más fácil de lograr, simplemente agregando unas clases extra a las etiquetas dentro del código HTML.

Seguidamente, para corresponder a un diseño responsive diseñé las adaptaciones de la web a la visión de una tablet y un dispositivo móvil.

## Vista en Web - 1920x1080

The wireframe displays four screens arranged in a 2x2 grid:

- Top Left (Flujo 3):** Shows the "INICIA SESIÓN" (Login) screen with fields for "Código o email" and "Contraseña" (Password), and a "INICIAR SESIÓN" (Start Session) button.
- Top Right:** Shows the "PERFIL DE USUARIO" (User Profile) screen for "María Mora López" with fields for "Nombre" (Name), "Email" (Email), "Contraseña" (Password), and a "Actualizar perfil" (Update Profile) button.
- Bottom Left:** Shows the "REGISTRO" (Registration) screen with fields for "Nombre" (Name), "Email" (Email), "Contraseña" (Password), and "Profesor" (Teacher). A "REGISTRARSE" (Register) button is at the bottom.
- Bottom Right:** Shows the same "REGISTRO" screen, but the "Profesor" field is expanded to show three options: "Rodrigo Cortés", "Aurora Palomares", and "Laura Lara".

A sidebar on the right contains a color palette titled "Colores" with various hex codes:

- FFFFFF
- #D4CECE
- #62D8BD
- #FFE572
- #FF9DA4
- #F5926E
- #E54D4D
- #8D625E

Primeras pantallas de la aplicación | Paleta de color empleada

The wireframe shows several screens:

- Top Left:** Displays a grid of levels: "Nivel 1: Utilidad 1" (HTML, JavaScript), "Nivel 2: Utilidad 2" (CSS, JavaScript), "Nivel 3: Utilidad 3" (HTML, CSS, JavaScript), "Nivel 4: Utilidad 4" (HTML, CSS, JavaScript), and "Nivel 5: Utilidad 5" (HTML, CSS, JavaScript).
- Top Right:** Shows a "JUEGOS" (Games) screen with two cards: "Nivel 1" (JavaScript) and "Nivel 2" (JavaScript).
- Middle Left:** Shows a "EXECUTAR EL JUEGO" (Run Game) screen with a large red placeholder area and a "Jugar" (Play) button.
- Middle Center:** Shows a "PREGUNTAS" (Questions) screen with three questions, each with two "Responde" (Answer) buttons.
- Middle Right:** Shows a "DETALLADO" (Detailed) screen with a "CERRAR" (Close) button.

Vista de varias pantallas de la aplicación

## Vista en Tablet - Ipad Air 820x1180

The wireframe shows three screens:

- Left Screen (Perfil):** Shows a "Empezar a jugar" (Start Playing) button and a "Flujo 2" section with a "Log in" screen.
- Middle Screen (Mundos):** Shows the "REGISTRO" (Registration) screen with fields for "Nombre" (Name), "Email" (Email), "Contraseña" (Password), and "Profesor". The "Profesor" field is expanded to show three options: "Rodrigo Cortés", "Aurora Palomares", and "Laura Lara".
- Right Screen (Niveles):** Shows the same "REGISTRO" screen, but the "Profesor" field is expanded to show three options: "Rodrigo Cortés", "Aurora Palomares", and "Laura Lara".

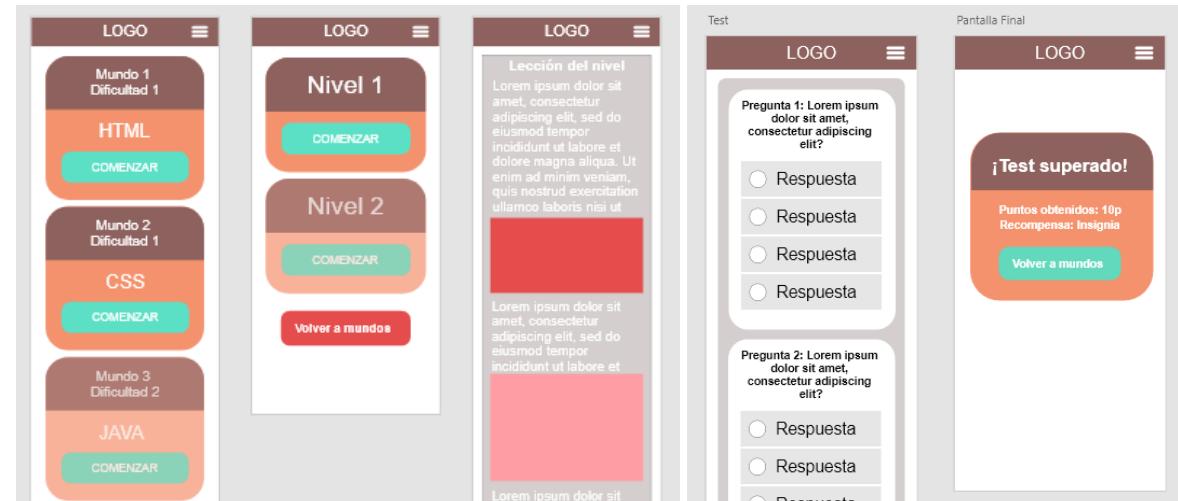
Pantallas iniciales | Registro





Pantallas de perfil | Dinámica de mundos

## Vistas en Dispositivo Móvil - Iphone 13/12 390x844



Vistas generales de móvil

A la par que se llevaba este proceso de prototipado de pantalla he querido incluir parte de los logotipos, iconos y mascota corporativa que realicé para la plataforma desde cero, ya que quería abarcar como extra la parte de diseños creativos de la web al considerarlo algo importante para una web con temática infantil.

TFG: "Learn-INF", web de aprendizaje interactivo



## 4.2. Diseño. ¿Cómo se hará la aplicación?

- **Estructura de la base de datos**

Para la estructura de la base de datos, siguiendo la lógica de los anteriores diagramas diseñados, decidí utilizar el formato de MySQL para la creación y gestionarla a través del software de DBeaver.

Las tablas principales y sus relaciones generadas por este programa serían:

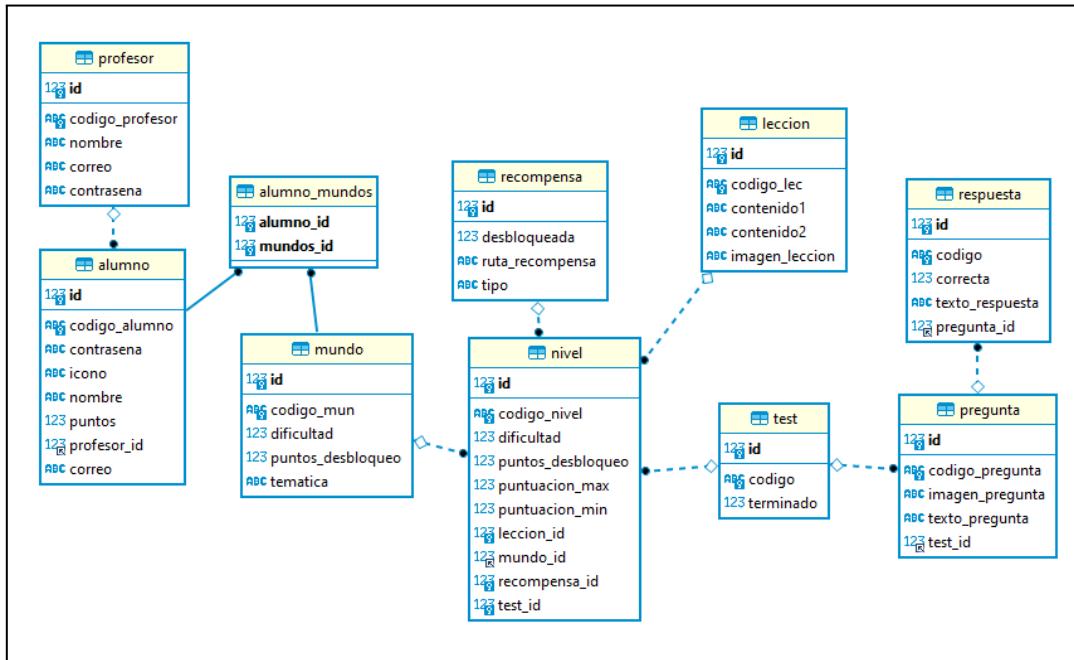


Diagrama de clases generado por DBeaver

- **Arquitectura del proyecto: Tecnologías utilizadas en cada parte del proyecto**

Las principales tecnologías que he empleado en el desarrollo del proyecto son las siguientes:

→ **Adobe XD:** Software usado para la creación de interfaces, tanto para páginas webs como para aplicaciones, perteneciente a la suite de Adobe.

Usado para el prototipado de la aplicación, el diseño de sus vistas y flujo de navegación.

→ **Visual Studio Code:** Editor de código fuente desarrollado por Microsoft.

Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalizador inteligente de código, fragmentos y refactorización de código.

Usado para el código HTML, realización de las hojas de estilo CSS y documentos JavaScript.

→ **SASS:** Lenguaje de hoja de estilos en cascada y preprocesador CSS, es una herramienta que genera automáticamente las hojas en base a variables definidas en el archivo SCSS de la aplicación.

Usado para la compilación de las hojas de estilo CSS, utilizado sobretodo para la definición de variables de la paleta de color.

→ **JavaScript:** Lenguaje de programación que entienden los navegadores, con el que se desarrolla la parte de la funcionalidad frontend en sitios web y aplicaciones web.

Usado para la validación de formularios, acciones y diseños interactivos en la web.

→ **Bootstrap:** Biblioteca multiplataforma y conjunto de herramientas de código abierto para diseño de aplicaciones web. Con clases y estilos predeterminados.

Usado para la estructura y estilos responsive de las vistas de la aplicación.

→ **IntelliJ Idea:** Entorno de desarrollo integrado para el desarrollo de programas informáticos. Es desarrollado por JetBrains y proporciona asistencia a la codificación inteligente para determinados lenguajes de desarrollo de código.

Usado para la estructura y lógicas de la parte Back de la aplicación.

→ **Jakarta:** Plataforma de Java Enterprise Edition o Java EE para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

Usado para la escritura del código fuente de la aplicación.

→ **draw.io:** Diagrams.net, fundado por draw.io, es un software de creación de diagramas libre. Usado para la realización de diagramas y esquemas para la base de datos.

→ **JPA:** Java Persistence API, más conocida por sus siglas JPA, es una API de persistencia desarrollada para la plataforma Java EE. Proporciona un mecanismo para gestionar la persistencia y la correlación relacional de objetos. La persistencia de datos es un medio mediante el cual una aplicación puede recuperar información no volátil.

Usado para el manejo, creación, modificación y gestión de las tablas en la base de datos dentro de la aplicación.

→ **DBeaver:** Aplicación de software cliente de SQL, a través de ella podemos gestionar las conexiones a las bases de datos. También podemos llevar su creación, modificación y gestión de datos a través de esta.

Usado para la conexión entre la aplicación y la base de datos. También para la gestión y la carga de datos de una manera más concreta visualmente durante el desarrollo de la aplicación.

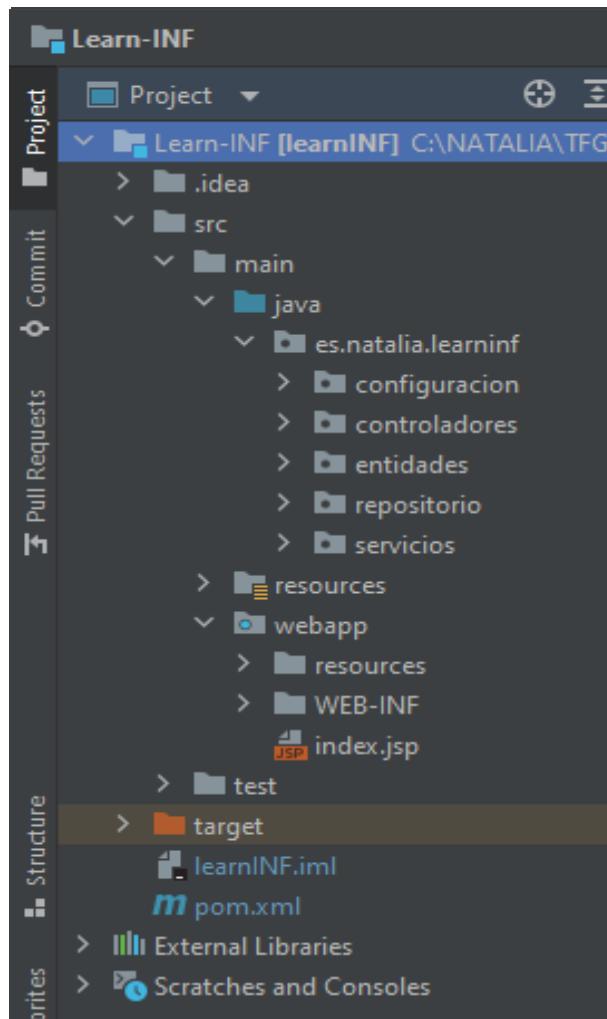
→ **Docker:** Proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software. Automatiza la virtualización de aplicaciones en múltiples sistemas operativos.

Usado para el despliegue de la aplicación y conexión con la base de datos.

### 4.3. Implementación y pruebas

- Desarrollo de la aplicación con los lenguajes y programas elegidos

En el punto anterior hemos visto las distintas aplicaciones y lenguajes elegidos y dónde han sido implementados en la estructura del proyecto. Por ello, vamos a pasar directamente a la explicación del código fuente en sí.



La estructura del proyecto completo sería la siguiente:

El código está alojado en la carpeta **src**.

Dentro de esta, en **main**, se encuentra la carpeta **java**, **resources** y **webapp** donde está toda la estructura Java y los archivos que contienen las vistas jsp, estilos y demás códigos de la aplicación. En la carpeta **test** encontraremos las pruebas unitarias realizadas con JUnit.

Dentro de la carpeta **target** generada con cada despliegue que hacemos de la aplicación encontraremos entre otros archivos el **.war** del proyecto, necesario luego para el despliegue en SFTP.

También encontraremos el archivo **pom.xml**, donde definimos los datos del artefacto, sus propiedades y dependencias necesarias para usar ciertas funciones en nuestro proyecto.

→ **Carpeta java:** Esta carpeta recolecta todos los archivos con extensión Java del proyecto. La estructura dentro de esta es recogida a su vez por un package, y dentro de este se divide en varias carpetas.

La carpeta de **configuración** simplemente especifica la ruta inicial y la extensión de los archivos de las vistas de la aplicación, definiéndolo como archivos **.jsp** haciendo que más adelante no tengamos que especificarlo.

La carpeta de **entidades** es donde se definen las tablas de la base de datos. Cada una cuenta con lo que serían los atributos de la entidad, su tipo de dato, las relaciones que guardan con otras entidades y métodos específicos de **constructores**, **getters** y **setters**. Gracias a la extensión de JPA definir estas relaciones ha sido más rápido y sencillo.

Las carpeta de **repositorios** define interfaces java para usar la funcionalidad del módulo DeltaSpike, la cual ayuda con lógicas ya predefinidas básicas para las consultas a la base de datos. Para usar esta necesitamos la clase java de Entity Manager, todas las interfaces java tendrán la relación **extend** con esta clase para usar sus funcionalidades. Son consultas simples, como búsquedas por ID.

La carpeta de **servicios** recoge los métodos que realizan consultas, eliminaciones, creaciones o modificaciones a la BD, y se pueden relacionar con repositorios. También tiene consultas predefinidas gracias a DeltaSpike, como los **findAll** o **findById**.

En la carpeta de **controladores** encontramos las funciones de enrutado, que relaciona las vistas con los servicios y proporciona estos al usuario. Por lo general suele haber un controlador por vista o conjunto de vistas relacionadas. Dependiendo la función que queramos realizar podremos obtener datos para mostrar en las vistas o recoger datos de formularios para trabajar con ellos gracias a los servicios que hayamos creado.

→ **Carpeta resources:** En esta encontraremos archivos necesarios para el uso de ciertas dependencias o servicios como DeltaSpike o las funcionalidades de hibernate.

→ **Carpeta webapp:** En esta guardaremos todas las vistas, hojas de estilo CSS, archivos JavaScript, imágenes y demás fuentes necesarias para la visualización de la aplicación. En este caso, las vistas están realizadas con extensión **.jsp** para la visualización de datos que se pasarán mediante los modelos del controlador.

→ **Carpeta test:** Aquí se guardan los test realizados mediante clases Java y el framework de testing JUnit.

El código que guardan estas carpetas está por completo comentado en cada método y aplicación de los servicios en los controladores, cómo están formadas las entidades, etc. Por ello, creo que es mucho más sencillo entenderlo mediante las explicaciones que hay en estos archivos, pero de una manera breve explicaré el flujo que puede llevar la aplicación en la situación de crear una cuenta de usuario básica y lo que se puede encontrar hasta llegar al final de un test. El ejemplo sería el siguiente:

Al ser un nuevo usuario, tendremos que crearnos una cuenta en la plataforma. Rellenando el formulario, el cual tiene validación mediante las funcionalidades de HTML5, haríamos un POST desde el controlador para la creación de la cuenta y también una sesión para guardar los datos del usuario. Una vez creada, nos llevaría a la vista de perfil, donde podremos cerrar sesión cuando queramos (destruimos la sesión del controlador) e ir a mundos donde empezaremos a navegar por los distintos menús hasta la lección.

Todos los menús están controlados. Primero comprueba que haya una sesión activa, y segundo que los puntos que tiene el usuario en su cuenta sean los necesarios para su paso a los distintos niveles y mundos. Todos los datos mostrados por pantalla serán resultado de consultas de los servicios y métodos GET para mostrar los datos a través de los modelos.

Al llegar al test se recogerán las opciones seleccionadas a través de Javascript y Ajax, para el conteo de aciertos. Dependiendo del resultado se mostrará un resultado u otro en la pantalla final, otorgando una recompensa y desbloqueando nuevos niveles.

Por último, en caso de que quisiéramos ingresar como profesor el login sería similar, solo que para acceder a estos perfiles sólo disponemos de la opción por correo. Al ser profesor, el perfil de este será distinto ya que nos mostrará, aparte de su información personal, un listado de alumnos que están asociados a este a través de su código.

El profesor tiene acceso a todos los recursos de la web. Para controlar que no haya fallos, la única opción deshabilitada es la de mandar el test para obtener los puntos. Los profesores no pueden darse de alta en la plataforma porque sería muy fácil para un alumno tener información que no les correspondiera. En caso de querer dar de alta a un profesor nuevo tendría que ser de manera interna en la base de datos.

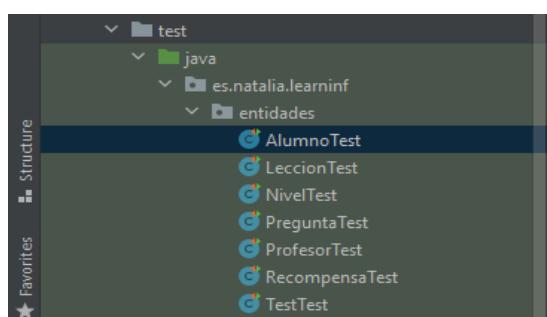
Algunas normas a destacar serían, por ejemplo, el no poder repetir un test que ha sido superado con éxito, pero sí se podría visualizar la lección. Las recompensas se pueden visualizar en otra vista de la aplicación, si pasas el ratón por encima de ellas te dirán dónde las has obtenido. También se comprueba que haya una sesión activa siempre que sea necesaria para evitar excepciones por datos nulos, y a su vez se controlará que tipo de sesión es, si es una sesión de alumno o de profesor para tener accesos y perfiles distintos.

- Documentación las pruebas realizadas

- Pruebas unitarias de las partes más complicadas del proyecto

Para realizar las pruebas de testeo de la aplicación utilicé JUnit5. Se trata de un framework para la automatización de las pruebas unitarias o de integración en proyectos.

Gracias a estas bibliotecas con sus anotaciones y métodos permite realizar pruebas de



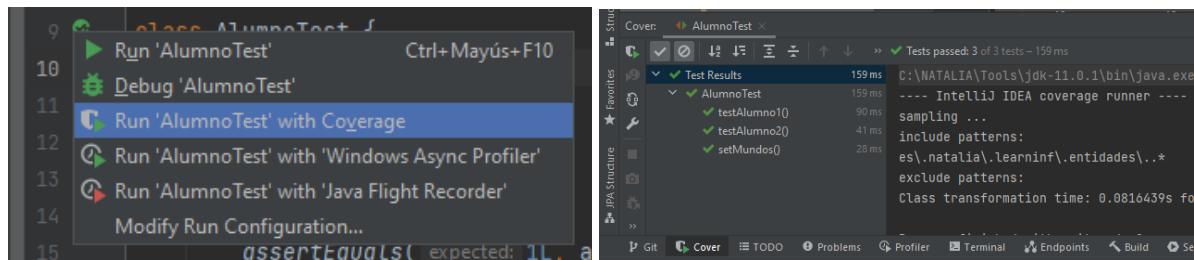
integración y asegurarnos que los métodos más importantes que usamos en el proyecto son eficaces y no dan lugar a fallos a la hora de usarlos en la aplicación desplegada.

Estas pruebas deben estar guardadas en la carpeta de **test - java** dentro del código fuente de la

aplicación, y deberá tener la misma estructura de carpetas que las clases que queramos testear. Una vez tengamos estas, es importante que las clases tengan la anotación de `@Test` en

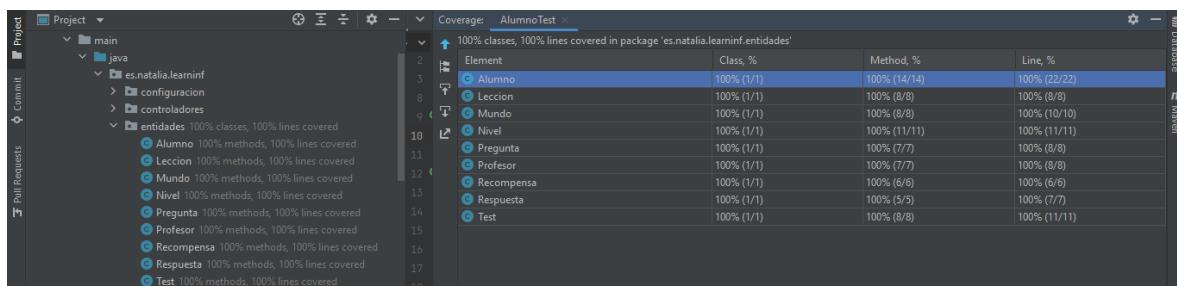
```
1 package es.natalia.learninf.entidades;
2
3 import ...
4
5
6 class AlumnoTest {
7
8     @Test
9     public void testAlumno1(){
10         Alumno a = new Alumno();
11         a.setId(1L);
12         assertEquals( expected: 1L, a.getId());
13     }
14 }
```

ellas, ya que con esto seremos capaces de arrancar las pruebas por cada método que queramos comprobar. Cuando queramos comprobar si los test realizados son correctos clickeamos en los iconos que se crearán en estas clases, y seleccionaremos Run with Coverage para comprobar qué porcentaje del código ha sido testeado con éxito.



## Muestra de funcionamiento de testeo

Para este apartado he realizado las pruebas de las clases de **entidades**, ya que al ser la raíz de la creación de los posteriores métodos era importante comprobar que los métodos de creación de estos elementos, setter y getters funcionan apropiadamente y de la manera esperada.



Vista del testing 100% completado en la carpeta entidades

- Pruebas de integración: Aceptación en las historias de usuario

Para el seguimiento de las historias de usuario, peticiones y plazos de desarrollo, como comenté con anterioridad, he utilizado la plataforma de Taiga.io incorporando todas las necesidades que han surgido a lo largo de la construcción de la aplicación.

En total han sido 6 épicas que dividían y clasificaban las historias de usuario, a grandes rasgos, entre diseño UX, FrontEnd y BackEnd. A la vez que se iban especificando estas historias y sus tareas también he incorporado peticiones, la mayoría de mejoras para la aplicación sin ser en exceso prioritarias.

The screenshot shows a list of 8 tasks under the 'Épicas' tab:

- #45 Definición del proyecto y primeros esquemas (Hecha)
- #46 Despliegue de la aplicación (Hecha)
- #47 Arquitectura de la Aplicación (Hecha)
- #48 Diseño de la aplicación (Hecha)
- #19 Testing JUnit + Jasmine (En curso)
- #20 Memoria del proyecto (Hecha)
- #66 Revisión final (Hecha)

Vista del panel de Épicas en Taiga.io

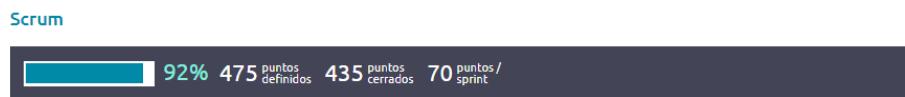
The screenshot shows a list of 8 issues under the 'Peticiones' tab:

- #116 Cambio de posiciones SignUp (Cerrada)
- #107 Desbloqueo de niveles por puntuación (Cerrada)
- #125 Mejora vistas perfil (Rechazada)
- #124 Iconos botones (Rechazada)
- #117 Mensajes personalizados HTML5 validacion (Cerrada)
- #102 Cambiar formas de login (Cerrada)
- #90 Repaso vista móvil - Test (Cerrada)
- #89 Cambiar BD - Lección (Cerrada)

Vista del panel de Peticiones en Taiga.io. Las de tipo BUG son rojas y las de tipo Mejora son celestes

En general la mayoría de las tareas han sido realizadas, algunas han tenido que traspasarse de unos sprints a otros y otras se han vuelto a incorporar en forma de petición al ser problemas menores. En total solo dos de las peticiones realizadas han sido rechazadas.

A cada tarea declarada se le asignaba determinados puntos, correspondientes a la parte de la aplicación que tocaban. En total se han definido un total de 475 puntos, logrando finalmente un 92% de los puntos definidos:



Por desgracia, los únicos puntos que no he sido capaz de cubrir por errores de procesamiento y falta de tiempo han sido parte de los testeos de código con JUnit. Aun así, considero que he completado muchas más funciones esenciales de la web de lo que esperaba y, con un poco más de tiempo, habría podido terminar los testeos ya que eran fallos de tipo null.

#### 4.4. Implantación y documentación

- **Manual de Instalación: Despliegue con Docker**

Todos los archivos necesarios para el despliegue están en la carpeta **Despliegue-LearnINF** adjunta con el resto del proyecto. El despliegue consta de tres contenedores, uno MySQL, otro de Wildfly y por último el de SFTP.

##### 1. Despliegue de contenedores Docker

En la terminal de sistema introduciremos los comandos. Descargamos la imagen de MySQL en caso de no tenerla, y creamos la imagen del Dockerfile adjunto en la carpeta.

Seguidamente, levantamos los contenedores con los siguientes comandos:

```
docker pull mysql
docker build -t wildfly:learninf .
docker compose up -d
```

```
C:\NATALIA\despliegue-learn>docker build -t wildfly:learninf .
[+] Building 7.2s (11/11) FINISHED
--> [internal] load build definition from Dockerfile
--> [internal] load .dockerignore
--> [internal] load context
--> [internal] load metadata for docker.io/jboss/base-jdk:11
--> [1/6] FROM docker.io/jboss/base-jdk:11@sha256:6b73a2cc8b8c2b2af369e9160e9e9fea22abf93f4f2894518ee87c73a14d991
--> [internal] load context
--> [internal] load metadata for docker.io/wildfly/wildfly:23.0.2.Final-wildfly-preview-2
--> [2/6] RUN curl -L -o https://download.jboss.org/wildfly/23.0.2.Final/wildfly-preview-2
--> [3/6] COPY mysql-connector-java-8.0.25.jar /opt/jboss/wildfly/modules/system/layers/base/com/mysql/main/
--> [4/6] COPY standalone.xml /opt/jboss/wildfly/standalone/configuration/
--> [5/6] COPY module.xml /opt/jboss/wildfly/modules/system/layers/base/com/mysql/main/
--> [6/6] COPY standalone.xml /opt/jboss/wildfly/standalone/configuration/
--> exporting to image
--> exporting layers
--> writing image sha256:574757aa0c76e8464c093a0d4c0cf1df3d234e4a3b9f237ddaccd20d2b652a6
--> naming to docker.io/library/wildfly:learninf
--> naming to docker.io/library/wildfly:learninf

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

C:\NATALIA\despliegue-learn>docker compose up -d
[+] Running 4/5
  Network despliegue-learn_default      Created
  Volume "despliegue-learn_deployments" Created
  Container learn-sftp                  Started
                                         0.8s
                                         0.0s
                                         1.8s v
```

Vista de la terminal con el despliegue completo de Docker

Gracias a la configuración definida tanto en el Dockerfile como en el docker-compose no tendremos que cargar la base de datos ni configurar el DataSource manualmente. Podemos comprobar accediendo a la consola habilitada de WildFly:

- **Puerto:** <http://localhost:9990/>
- **Usuario y Contraseña:** profesor / learn

Configuration		Subsystem (31)	Datasources & Drivers	Datasource
Subsystems	>	Batch jBeret	Datasources	learninfDS
Interfaces	>	Core Management	jDBC Drivers	
Socket Bindings	>	Datasources & Drivers		
Paths		Deployment Scanners		
System Properties		Discovery		
		Distributable Web		
		EE		

**Main Attributes**

- JNDI Name: java:/dbc/learninfDS
- Driver Name: mysql
- Connection URL: jdbc:mysql://learn-mysql:3306/learninfdb7au...
- Enabled: true
- Statistics Enabled: false

Si queremos comprobar los datos cargados en la base de datos podemos introducir los siguientes comandos:

- **Contraseña para usuario root: natalia**

```
docker exec -it learn-mysql mysql -p
show databases;
use learninfldb;
show tables;
show columns from alumno;
select * from alumno;
```

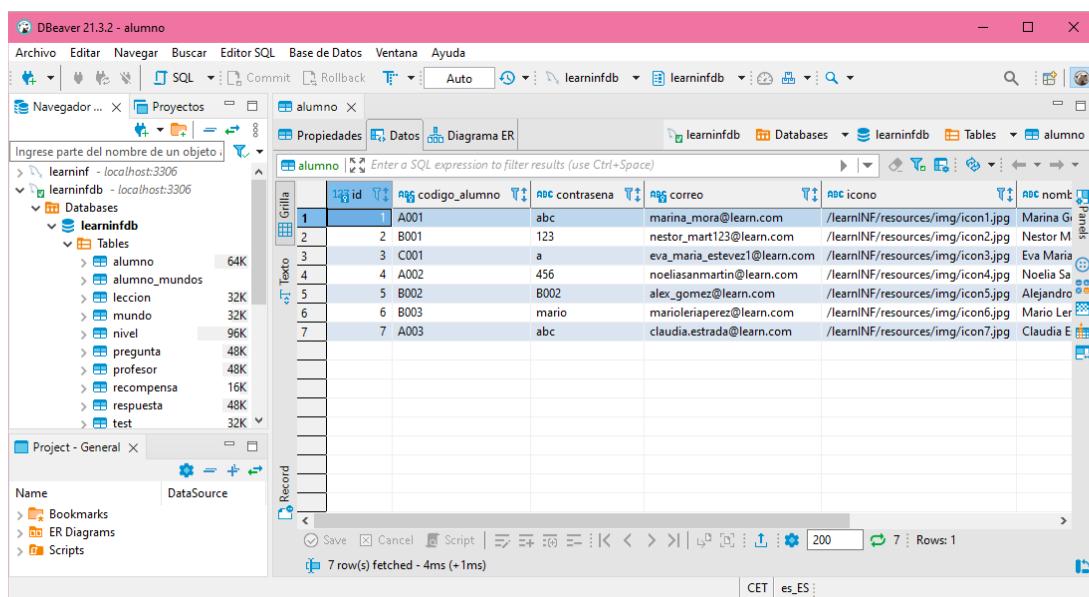
Estos nos mostrarán los datos que debería haber en la tabla de alumno. Tendría que verse algo parecido a esto:

```
mysql> show columns from alumno;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default |
+-----+-----+-----+-----+-----+
| id | bigint | NO | PRI | NULL |
| codigo_alumno | varchar(4) | NO | UNI | NULL |
| contrasena | varchar(30) | NO | UNI | NULL |
| correo | varchar(100) | NO | UNI | NULL |
| icono | varchar(300) | YES | NULL |
| nombre | varchar(100) | NO | NULL |
| puntos | int | YES | NULL |
| profesor_id | bigint | NO | MUL | NULL |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> select * from alumnos;
ERROR 1146 (42502): Table 'learninfldb.alumnos' doesn't exist
mysql> select * from alumno;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | codigo_alumno | contrasena | correo | icono | nombre | puntos | profesor_id |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | A001 | abc | marina_mora@learn.com | /learnINF/resources/img/icon1.jpg | Marina Gonzalez Mora | 0 | 1 |
| 2 | B001 | 123 | nestor_mart123@learn.com | /learnINF/resources/img/icon2.jpg | Nestor Martinez Herrero | 5 | 2 |
| 3 | C001 | a | eva_maría_estevez1@learn.com | /learnINF/resources/img/icon3.jpg | Eva Maria Estvez | 10 | 3 |
| 4 | A002 | 456 | noeliaasanmartin@learn.com | /learnINF/resources/img/icon4.jpg | Noelia Sanmartin Gil | 15 | 1 |
| 5 | B002 | B002 | alex_gomez@learn.com | /learnINF/resources/img/icon5.jpg | Alejandro Gomez Parraga | 5 | 2 |
| 6 | B003 | mario | marioleriaperez@learn.com | /learnINF/resources/img/icon6.jpg | Mario Leria Perez | 0 | 2 |
| 7 | A003 | abc | claudia.estrada@learn.com | /learnINF/resources/img/icon7.jpg | Claudia Estrada Roman | 0 | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

También podremos comprobarlo a través de un gestor de bases como es DBeaver:

- **Database: learninfldb**
- **Usuario y Contraseña: root / natalia**



## 2. Despliegue del .war con SFTP

Volvemos a la terminal, nos conectaremos al servicio SFTP definido en el docker compose con el nombre de learning.

- **Contraseña:** learn

```
sftp -P 2201 learning@localhost  
cd deployments  
put learnINF.war
```

```
C:\NATALIA\despliegue-learning>sftp -P 2201 learning@localhost  
The authenticity of host '[localhost]:2201 ([::1]:2201)' can't be established.  
ED25519 key fingerprint is SHA256:YP67akjUotvkC8t+9dkL0zz/Bkf9mono/7SH9ssk0gk.  
Are you sure you want to continue connecting (yes/no/[fingerprint])?  
Warning: Permanently added '[localhost]:2201' (ED25519) to the list of known hosts.  
learning@localhost's password:  
Permission denied, please try again.  
learning@localhost's password:  
Connected to localhost.  
sftp> ls  
deployments  
sftp> cd deployments  
sftp> ls  
README.txt  
sftp> ls  
README.txt  
sftp> ls  
learnINF.war  
Uploading learnINF.war to /deployments/learnINF.war  
learnINF.war                                              100%   21MB  57.2MB/s  00:00  
sftp> ls  
README.txt      learnINF.war  
sftp> ls  
README.txt      learnINF.war          learnINF.war.isdeploying  
sftp> ls  
README.txt      learnINF.war          learnINF.war.deployed  
sftp>
```

En el caso de que nos diera error por tener una fingerprint distinta, simplemente nos dirigimos al fichero know-hosts en la carpeta **.ssh** del equipo y borramos la que pertenezca a localhost. Entonces podremos crear una nueva introduciendo la contraseña. Una vez creada, vamos a la carpeta de deployments y hacemos un put del .war en esta. Nos aseguramos de que se haya desplegado correctamente en la terminal o en la pestaña de Deployments en la consola de Wildfly.

The screenshot shows the HAL Management Console interface. The top navigation bar includes 'Homepage', 'Deployments' (which is selected), 'Configuration', 'Runtime', 'Patching', and 'Access Control'. On the left, there's a sidebar with 'Deployment (1)' and a 'View' button. The main content area displays 'learnINF.war' with a green checkmark indicating it is enabled and active. Below this, under 'Main Attributes', are the following details:

Name:	learnINF.war
Runtime Name:	learnINF.war
Context Root:	/learnINF
Enabled, Managed, Exploded:	✓ ✓ ✘
Status:	OK
Last enabled at:	6/13/22, 7:22 PM
Last disabled at:	n/a

Ahora el proyecto está disponible en la siguiente dirección:

<http://localhost:8080/learnINF/mvc/portada>

- **Manual de Usuario: Como crear nuestra cuenta en la aplicación**

Por último vamos a poner a prueba la aplicación. Primero vamos a comprobar cómo acceder con una cuenta de profesor. Para **acceder como profesor** disponemos de este usuario:

- **Email:** anna\_c\_g@learn.com | **Contraseña:** abc

Accedemos mediante el login único, que nos llevará a nuestro perfil de profesor con la información y el listado de alumnos a su cargo. También podremos cerrar nuestra sesión.

The screenshots illustrate the user interface of the Learn-INF application. The first screenshot shows the 'Iniciar Sesión' (Start Session) screen with fields for 'Email / Código de Usuario' (anna\_c\_g@learn.com) and 'Contraseña' (abc), a 'Remember me' checkbox, and a 'Iniciar' (Start) button. The second screenshot shows the 'Perfil' (Profile) screen for a teacher named Anna Campoy Gracia, displaying her name, code A, and email anna\_c\_g@learn.com. It also shows a list of students: Marina Gonzalez Mora (code A001) and Noelia Sanmartin Gil (code A002). Buttons for 'Ir a Mundos' (Go to Worlds) and 'Cerrar sesión' (Close session) are present. The third screenshot shows a 'Listado de alumnos' (List of Students) modal window for student Marina Gonzalez Mora, showing her name, code A001, points 0, and basic level. Buttons for 'Ir a Mundos' and 'Cerrar sesión' are also here.

Vista del inicio de sesión y el perfil del profesor

Podremos acceder a todos los mundos, niveles, lecciones y test de la plataforma, pero no podremos realizar estos últimos.

The screenshots show the main menu of the Learn-INF application. It features three sections for 'Mundo 1' (Difficulty 1), 'Mundo 2' (Difficulty 2), and 'Mundo 3' (Difficulty 3), each with a 'Jugar' (Play) button. To the right, there are sections for 'Nivel 1' and 'Nivel 2', also with 'Jugar' buttons. A 'Volver a Mundos' (Return to Worlds) button is located at the bottom right. The footer contains links for social media (Facebook, Instagram, Pinterest, Twitter, YouTube) and a 'Acerca de Learn-INF' (About Learn-INF) section with a detailed description of the platform's purpose and history.



Listado de mundos y niveles, con opción a entrar a sus lecciones y tests con las respuestas correctas marcadas

Para que fuera algo más realista y segura la aplicación, los profesores deberían tener otro sistema llevado por los moderadores de la plataforma para su registro.

Para el **acceso de los alumnos**, primero crearemos una cuenta en la pestaña de signup.

Introducimos los datos, si hay algún error nos informará la aplicación al enviar la solicitud.

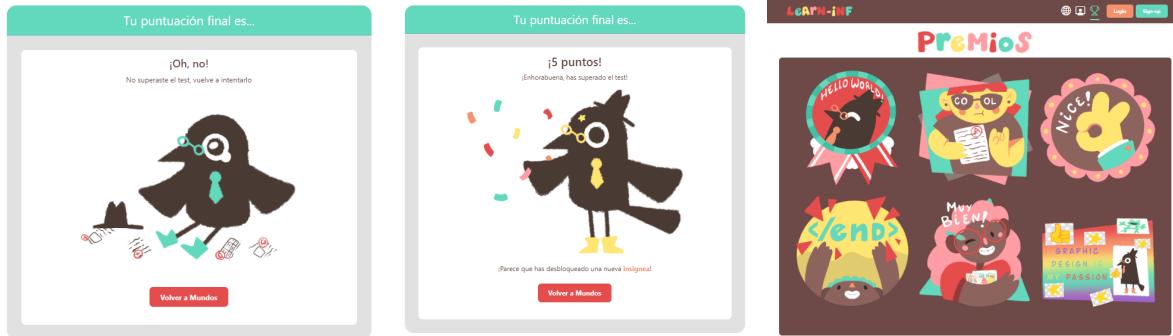
Una vez registrados, estaremos en el perfil donde veremos nuestros datos y podremos modificarlos.

Registro de un alumno en la plataforma

Navegando a mundos veremos los que tenemos disponibles. A su vez, también veremos los niveles, y dentro de estos primero tendremos la lección acorde con la temática.

Los mundos y niveles desbloqueados acorde a nuestros puntos iniciales

Estas lecciones no serán bloqueadas una vez superemos el test. Seguidamente tendremos el test que hay que completar al 100% para ganar los puntos y desbloquear los demás niveles. Dependiendo del resultado saldrán mensajes distintos. Por cada nivel desbloqueado conseguimos insignias que podemos ver en la pestaña de trofeos.



Una vez terminado el test, lo habremos superado o no. Desbloqueamos insignias con los tests. Para cerrar la sesión simplemente nos dirigimos de nuevo al perfil y clickeamos en cerrar sesión. Si queremos iniciar sesión en nuestra cuenta lo haremos mediante el login, y podremos o bien hacerlo con el código de alumno o con el correo que hayamos especificado en la creación del perfil.



Podemos acceder con el código de alumno o con nuestro correo

#### 4.5. Resultados y discusión

- **Comentarios sobre el desarrollo, detallando la temporalización real respecto a la planificada**

El desarrollo de la aplicación, a pesar de no haber completado todos los puntos que quería para ella, en lo general ha sido continuo. Ha abarcado las acciones más básicas y prioritarias de la web, como la carga de listados de mundos y niveles, hasta opciones más avanzadas, como era el login para profesores con la diferenciación del tipo de sesión.

La planificación sobre todo en la fase de inicio ha ido sobre las fechas previstas. La fase de prototipado, esquemas, inicio de estructuras y diseños iniciales, a excepciones de algunas tareas, fueron completadas con éxito en los sprints indicados. Las fases más avanzadas de diseño, estructura y despliegue, a pesar de haber sido más complicado el llevarlas al día, también se han podido ir ajustando en base al tiempo que restaba y las necesidades de mejoras o bugs que iban surgiendo con el desarrollo. Alguna vez se ha tenido que reconfigurar los sprints de estas fases pero no de manera crítica. Teniendo en cuenta que la planificación inicial también contaba con un margen de error, en conjunto se ha podido ajustar el desarrollo del código a los tiempos estipulados.

- **Dificultades más importantes encontradas**

La principal y primera de todas fue el planteamiento de la estructura que se iba a utilizar en el proyecto. En un principio pensé en hacerla con Spring MVC, ya que por una parte es lo más usado en el mundo laboral y por otra hay mucha documentación y guías donde consultar.

Pero, debido a dificultades para poder crear un proyecto con esta estructura y la limitación de tiempo, decidí realizarlo con la estructura de Jakarta.

Jakarta fue la estructura que, personalmente, trabajé durante más tiempo en el curso. Viendo sus funcionalidades y teniendo de antemano una estructura funcional similar a lo que quería para este proyecto, decidí que lo más sensato era continuar con esta estructura. También consideré que realizar este proyecto con una estructura más moderna aportaría originalidad.

- **Posibles mejoras y ampliaciones del proyecto**

Una de las mejoras que me hubiera gustado realizar era reestructurar la base de datos, ya que una vez es generada con la ayuda de JPA hay que tener muy en cuenta cómo la especificamos, y a veces no lo hace de la manera más eficiente. En cuanto a diseño, me hubiera gustado mejorar la estética de los botones para hacerlos personalizados y acordes al estilo de la web, la mejora de las vistas en el formato móvil y mejorar la estructura de la vista de los perfiles.

## 5. Conclusiones

- **Qué ha supuesto para tu formación la realización de este proyecto**

Este proyecto me ha aportado no sólo nuevos conocimientos a nivel informático, sino también a nivel de organización y disciplina. Me ha ayudado a ver cómo podría ser la gestión de un proyecto real, la importancia de establecer tiempos de desarrollo. También, el volcarme exclusivamente en un solo proyecto ha ayudado a ver cuántos factores hay que tener en cuenta a la hora de crear una web, y todo lo que se puede incluir en esta sin importar el tipo de elemento que sea.

- **En qué medida te ha servido para ampliar conocimientos**

Gran parte de lo que he aprendido realizando este proyecto ha sido en lo referente al diseño UX. Era uno de mis objetivos personales el poder explorar la experiencia de usuario, la maquetación y el uso de aplicaciones de prototipado.

A pesar de que en el curso hemos visto una de estas aplicaciones (Balsamiq Mockups), el usar una herramienta más profesional como es Adobe XD me ha permitido aprender el uso más extendido de este tipo de programas. Sobre todo ha sido interesante el saber cómo se trabaja en las primeras fases de un proyecto, y cómo depende del tipo de dinámica y estilo que se quiera implementar. También es muy útil poder definir los flujos de acción y las distintas funcionalidades desde un inicio tan temprano, ahorra mucho tiempo de planificación nivel de interfaz y definición dentro del código. Seguido a esto, otro de los formatos que he trabajado con más profundidad ha sido el uso del compilador de SASS.

Por otra parte, he podido experimentar más con el software de Jakarta. Decidí continuar el proyecto con este software debido a que a final de curso era la tecnología que más había utilizado, pero no por ello había podido explorar todas sus opciones.

- **Qué destrezas has conseguido con su realización que no se hubiesen conseguido a lo largo del ciclo**

Una de las mayores destrezas que he conseguido en este proyecto es el manejo fluido de la configuración CSS para el proyecto. En el ciclo, a pesar de haber dado también parte de diseño de interfaz es difícil centrarse exclusivamente en todo lo que nos puede permitir CSS combinado con JavaScript. Por ello considero, y también gracias a que parte de mi formación en las FCT ha sido enfocado a esto, que este proyecto me ha dado la libertad de poder explorar esta parte del desarrollo web.

También la resolución de problemas por medios propios es algo que, aunque durante el curso ya habíamos experimentado, se ha terciado más realista y ha sido parte del día a día del desarrollo en el proyecto.

## 6. Bibliografía y referencias

→ Punto 1.1:

- ◆ [Módulos DAW y su definición.](#)

→ Punto 1.2:

- ◆ [La mejor edad para programar empezar a programar.](#)
- ◆ [Cuando iniciar el aprendizaje en niños/adolescentes.](#)
- ◆ [Por qué los niños deberían aprender a programar.](#)
- ◆ [Code Monkey: Página de aprendizaje básico.](#)
- ◆ [Scratch: Página de aprendizaje básico y medio.](#)
- ◆ [MIT App Inventor: Página de aprendizaje medio y avanzado.](#)
- ◆ [Coding Giants: Página de aprendizaje medio y avanzado.](#)
- ◆ [Jakarta EE: Extensión de Java.](#)
- ◆ [Adobe XD: Prototipado para aplicaciones.](#)

→ Punto 1.4.1:

- ◆ [Manejo de la herramienta Adobe XD.](#)
- ◆ [Uso de Font Awesome con Adobe XD.](#)
- ◆ [Componentes: Dropdowns.](#)
- ◆ [Scrolling en Adobe XD.](#)

→ Punto 1.4.2:

- ◆ [JPA constraints.](#)
- ◆ [Bootstrap: Guía de elementos, funciones y clases.](#)
- ◆ [Curso de Jakarta EE.](#)
- ◆ [DeltaSpike nomenclatura.](#)
- ◆ [CSS funciones, clases y creación de estilos propios.](#)
- ◆ [HTML5 guía de funciones, validación de formularios.](#)

→ Punto 1.4.3:

- ◆ [DeltaSpike y sus funcionalidades.](#)
- ◆ [Criterios de Historia de Usuarios.](#)
- ◆ [Inject Dependencies Jakarta EE.](#)

→ Punto 1.4.4:

- ◆ [Despliegue con DataSource predeterminado.](#)
- ◆ [Despliegue con Database inicializada.](#)

## 7. Anexos

Para terminar, me gustaría añadir los archivos, videos y material gráfico que desafortunadamente no era posible incluir en los apartados posteriores, ya sea por no ser compatibles con el soporte de PDF o bien por hacer más conciso los puntos a tratar.

- Los archivos y videos de prototipado:

Son tres archivos y tienen extensión .xd ya que los realicé con la aplicación de Adobe XD de maquetación web. Los videos son una pequeña muestra de la vista previa de los tres anteriores archivos que nos facilita la aplicación de Adobe XD. En ellos podemos ver una simulación de flujo de la web. Se encuentran dentro de la carpeta de **Memoria del Proyecto** y a su vez en la carpeta de **Prototipos**.

- Los vídeos de demostración:

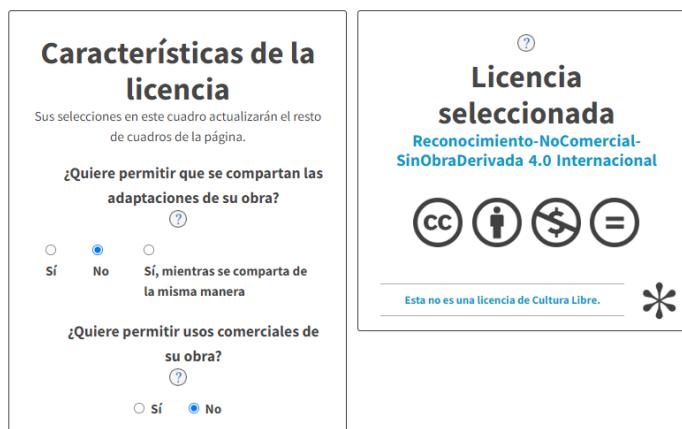
Son dos videos mostrando el despliegue y la aplicación en funcionamiento. Se encuentran dentro de la carpeta de **Memoria del Proyecto** y a su vez en la carpeta de **Videos Demo**.

- Material gráfico adicional usado en la web:

Los diseños e imágenes que se han realizado para la web, algunos los he podido incluir en este documentos pero, de todas maneras, quedan recogidos en la carpeta de **Memoria del Proyecto** y a su vez en la carpeta **Material Gráfico**.

- Licencia de Creative Commons:

El sello que acredita la autoría del proyecto. Este se muestra en el footer de la web, y estas serían sus especificaciones:



Por último, quiero añadir que todo este desarrollo ha sido subido a la plataforma de GitHub, donde se encuentra tanto la memoria completa, anexos y código del proyecto con una guía de despliegue y uso:

[https://github.com/nataliacristobal98/TFG\\_Learn-INF](https://github.com/nataliacristobal98/TFG_Learn-INF)