Proyecto de fin de ciclo 2021-2022

LeARN-iNF

Web de enseñanza interactiva para niños

Natalia Cristóbal Rodríguez 2º DAW

Curso académico 2021-2022

Fecha de entrega:

IES Francisco de Quevedo
Instituto bilingüe de la Comunidad de Madrid

UNIÓN EUROPEA
FONDO SOCIAL EUROPEO
INVIERTE EN TU FUTURO

Índice

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**", cuyo significado es **aprender** en inglés, y de la abreviatura "**INF**", la cual procede de **INFormation**.

Este proyecto podría cubrir dos dinámicas de uso: La primera sería la creación de la interfaz para los niños, y en la segunda una interfaz para profesores para la gestión de información y seguimiento del aprendizaje.

El planteamiento de inicio sería el siguiente:

- ➔ 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 hará que el usuario consiga puntos para el desbloqueo de siguientes niveles y de **recompensas** extra (iconos de usuario, diplomas, trofeos…).
- ➔ Cada nivel y mundo tendrá una **característica de desbloqueo**. No se podrá acceder a estos a no ser que dicha característica sea cumplida por el usuario.
- ➔ Los **usuarios** dispondrán de un panel personal con su información y logros.

Entre las distintas vistas de la web se comprobará el usuario que tenga la sesión activa, así se podrá controlar el acceso a las distintas áreas y mostrar los datos personalizados del usuario activo.

- **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. Primero se dibujarán los diagramas apropiados para concretar la estructura y relaciones entre los distintos elementos de la BD. También se concretarán los atributos de cada elemento y sus características, y se podría empezar a plantear consultas básicas.

  - **Diseño de Interfaces Web**

De cara a la planificación de la interfaz visual de la web. Primeramente, se realizará prototipos del diseño, también se puede plantear el hacerlo responsive dependiendo de la pantalla que se esté usando (formato web, tablet y dispositivo móvil).

- **Desarrollo Web Entorno Cliente / Lenguaje de Marcas y Sistemas de Gestión de Información**

Estos módulos permitirán tanto dar una mayor consistencia al diseño de interfaz como a el procesado de datos recogidos desde la parte de cliente a la del servidor y viceversa. El manejo de formularios, mostrar datos y recogerlos, permite la diversidad de datos independientemente de los usuarios conectados a través de plantillas predeterminadas con la BD y efectos para una visualización más fluida.

- **Desarrollo Web Entorno Servidor**

Para controlar el Backend de la manera más eficaz y sencilla posible. Definir la estructura a seguir para la organización de información, recogiendo todo tipo de elementos tanto visuales como datos necesarios en la generación de la web. La conexión a la BD, a la vez que su misma creación.

- **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 permitirá desarrollar todo lo que hemos aprendido en el curso, ya que el propósito de este proyecto sería abarcar tanto la parte Frontend a nivel de diseño interactivo y entendible para el público al que está dirigido, como la Backend para la gestión de niveles y seguridad.

De manera lógica, es una buena manera 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 puede tomar más funcionalidades y hacerla más compleja.

También permite implementar una seguridad y control de acceso, ya que esta web requerirá la creación y confirmación por usuario. El paso de este usuario se deberá controlar y delimitar 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 para 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 ya que, aunque se puede enseñar de una manera sencilla instrucciones de uso lógicas, el cerebro no está aún lo suficientemente desarrollado para comprenderlas.

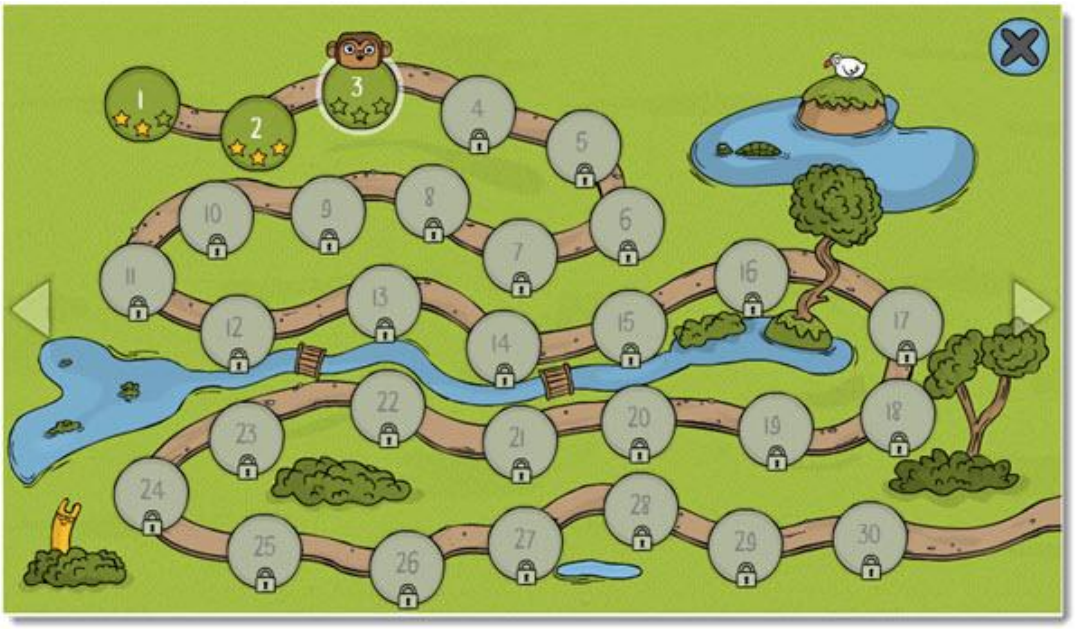
Por ello, lo más normal es que las webs de aprendizaje para infantes/adolescentes sean una combinación de lógicas de programación lo más sencillas posibles combinando estos ejercicios con juegos y estéticas parecidas a la de los videojuegos.

Aprender a programar estimula la creatividad, enseña a trabajar en equipo y el desarrollo de un pensamiento lógico. La programación es una habilidad fundamental a día de hoy, es muy interesante que desde pequeños sea adquirida.

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

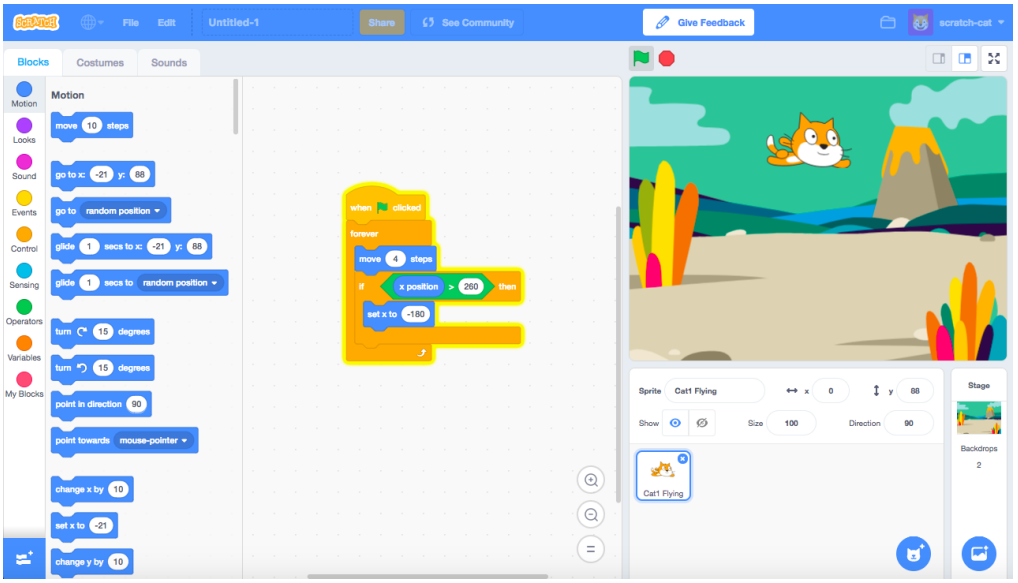
➔ **Code Monkey**

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 con lógica básica, pero no es una programación como tal. Su diseño también es un buen referente para este proyecto, ya que está enfocado para no solo el entendimiento, si no 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, estos ayudan 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 a estos y facilita el aprendizaje.

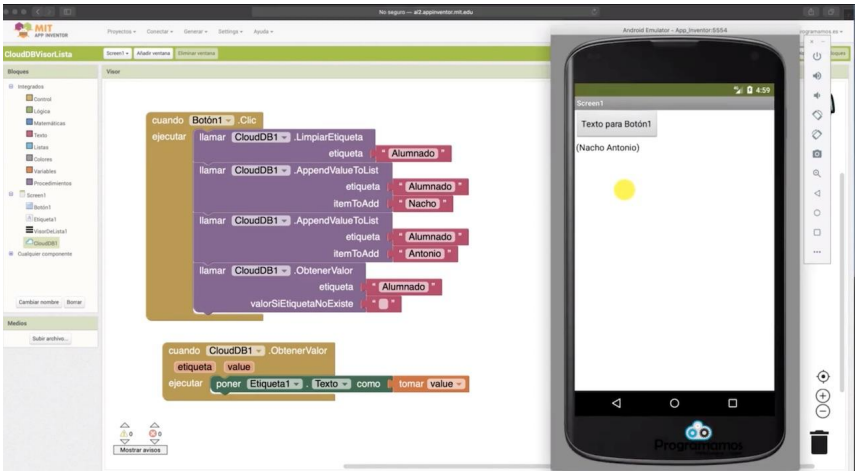


➔ **Coding Giants**

Esta web contiene un catálogo de cursos individuales para todas las edades. Muchos de estos están basados en el diseño y desarrollo de videojuegos, hay otros más teóricos y puramente prácticos a lo que es 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 bloque muy similares a lo que puede ser la programación real al igual que Scratch.



- **Objetivos**

A ser posible, la idea sería 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 sea más llamativo, con mayor colorido pero sin dejar atrás una buena estructura de la plataforma, que sea intuitiva y comprensible para ellos. También esto puede llevar mayor estudio y planeación, ya que es una gran parte de mercadotecnia a la hora de exponer el proyecto. En definitiva, la mayor parte del estudio estaría enfocado en la interfaz del usuario para una experiencia más fluida, sencilla y divertida para el sector al que está enfocado esta idea.

Para el desarrollo de esta aplicación, me gustaría trabajar con la herramienta de framework de Jakarta ya que será útil de manejar para el código Java y mejorará el rendimiento de la aplicación. También me gustaría implementar el uso de JPA, ya que es un framework que he encontrado muy útil en el curso y ayuda a la generación de la BD. Como base de datos usaré MySQL.

Por la parte del Frontend, aparte obviamente del uso de JavaScript y Bootstrap, es donde me gustaría implementar una herramienta no estudiada en el curso, algunas opciones que tengo sería trabajar algo más en profundidad el compilador de Sass y JQuery.
Si fuera posible, también me gustaría estudiar el diseño UX/UI mediante la herramienta de Adobe XD para crear el prototipo de la aplicación y el diseño responsive en ella.

- **Perfil del público**

El público al que estará enfocado será para una edad de 11 años en adelante, ya que después de haber leído sobre cuál es la mejor edad para empezar este aprendizaje, creo que podría ser un buen punto de partida. Ya que a esta edad 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.

Como la web tiene que acercarse lo más posible a la realidad, los ejercicios y dinámicas de uso pueden ser más variadas al contrario de las que tendrían que ser para niños más pequeños. Tampoco puede ser que la web se convierta en un videojuego por completo, ya que eso complicaría el proyecto y no sería viable para el tiempo que hay disponible. Los test que se presentan pueden ser entonces una combinación de preguntas teóricas y ejercicios visuales.

También sería posible que, al ser una aplicación dirigida a estudiantes, los profesores a cargo de estos puedan 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 habría una necesidad de cambiar la estética de la plataforma, simplemente es una adición a la estructura principal, que es la que está completamente dirigida a los alumnos y añadiría un login distinto entre alumnos y profesores.

3. Planificación

Para la planificación y la distribución de las tareas se usará la plataforma de Taiga, un software de organización de proyectos.

La estructura será la siguiente:

➔ **Épicas**

Ordenadas en fases, son la agrupación y división por temas de los distintos puntos a entregar.

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

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

Se dará una descripción, un estudio de mercado previo, se debe definir los objetivos iniciales que pueden variar según avance el proyecto.

Creación de los diagramas necesarios para la estructura de la base de datos, definir sus elementos y relaciones.

Diseño de las pantallas web: Prototipado inicial y responsive.

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 Wildfly y Docker.

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

Backend de la aplicación. La estructura se realizará 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 los siguientes puntos:

- Página principal.
- Login y Signup para los alumnos.
- Perfil del alumno.
- Listados de Mundos y Niveles.
- Test a realizar

También se buscará un Login diferenciado para los profesores, con el perfil también personalizado para estos. Avisos dinámicos de logros conseguidos, pruebas superadas, etc.

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

Testing de las funciones de la aplicación, tanto para el Backend como para el Frontend.

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

Redacción de los puntos a entregar + Guías de usuario y despliegue.

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

Comprobación de funciones y despliegue correcto final. 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, sobre todo los esquemas y documentaciones pertinentes para el proyecto estarían catalogadas aquí.

- Planificación inicial - Prototipos - Esquemas
  - **Realización de diagramas de BD:** *

Estructura inicial de la base de datos.

  - **Prototipado inicial de la aplicación:** *

Prototipos de la pantalla de inicio, login y signup, perfiles y listados.

  - **Tareas sin historia:**

Diagramas de base de datos, E/R y de clases. Puntos del 1.1 al 1.4 de la memoria del proyecto.

- Despliegue de la aplicación
  - **Tareas sin historia:**

Despliegue inicial de la aplicación.

- Primera versión básica de la web
  - **Pantalla principal:** *

- **Listado de Mundos/Niveles: ****
- **Login + Signup: ****
- **Tarea sin historia:**

Diseño inicial de las pantallas. Estructura de la aplicación en Spring.

- Necesidades avanzadas de la web
  - **Tarea sin historia:**

- Despliegue final - Revisión de funciones
  - **Tarea sin historia:**

- Revisión final
  - **Demo final: ***

Demostración del producto final.

- **Tarea sin historia:**

Revisión de la documentación, código e interfaz.

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 es identificar los factores que participarán en este. Al ser una web en la cual van a interactuar mínimo un tipo de usuario, podemos partir desde este y observar las interacciones y necesidades que tendrá dentro de nuestra plataforma. Por ejemplo: Si un usuario necesita una identificación exclusiva para la web, si un nivel está habilitado para este, cuantas recompensas ha desbloqueado, etc. Entonces, la base de datos para este proyecto debe comprender un mínimo de entidades:

➔ **Usuarios:** Se identificarán por un código de alumno. Su nombre, correo, icono y contraseña que podrán determinar 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. Puede tutelar a más de un alumno.

➔ **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 cuenta con el atributo de puntuación mínima para superar el nivel. Los niveles contienen una lección y un test único. Puede 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 formados 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 ayudarán a sumar los puntos totales del test.

Diagramas finales

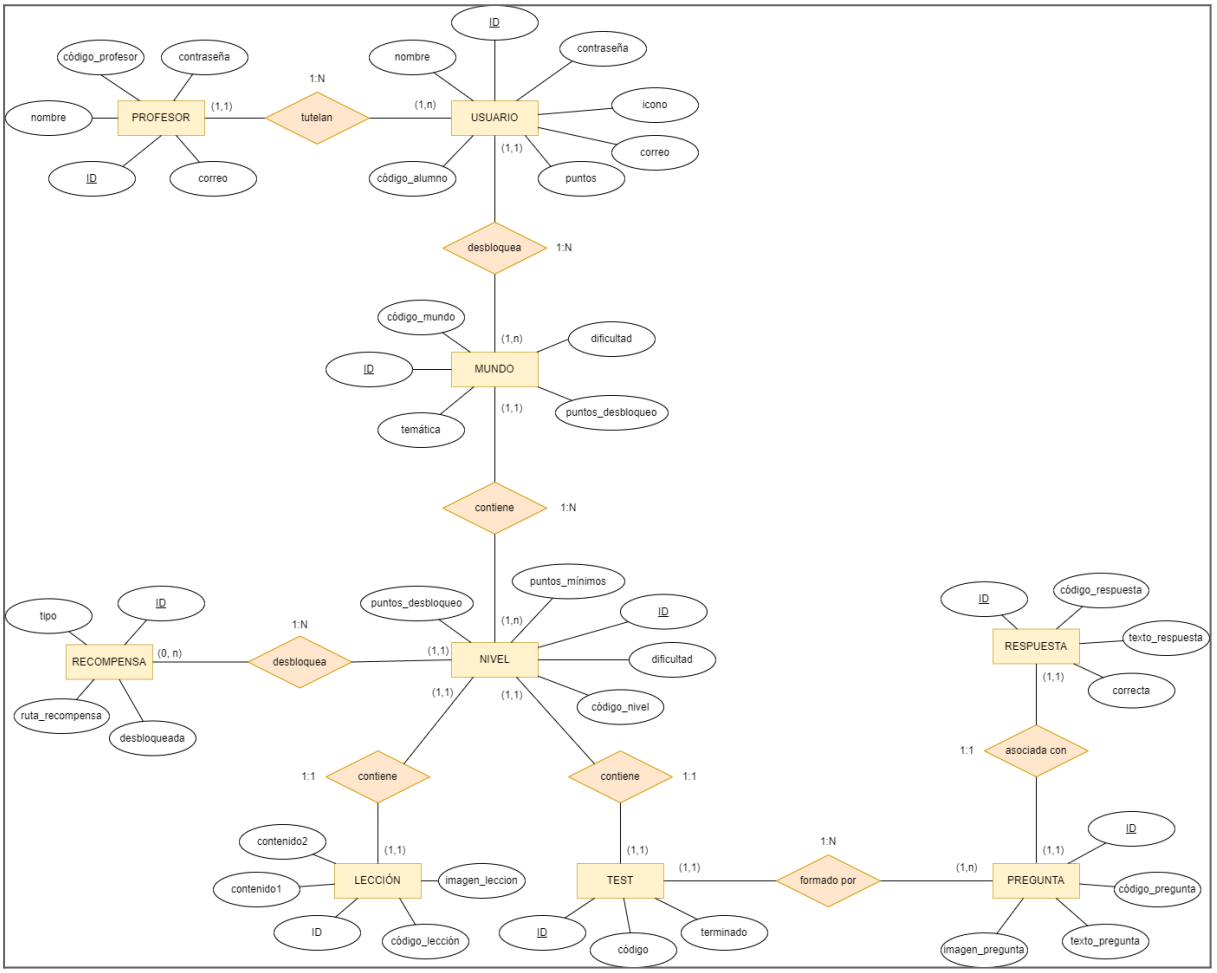


Diagrama de Entidad/Relación

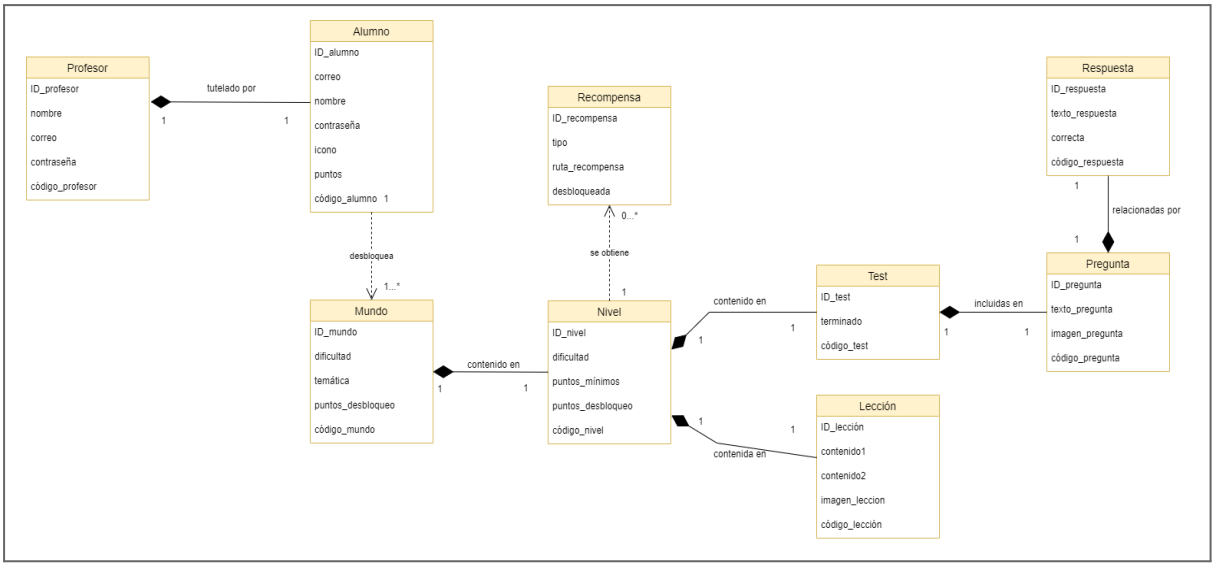


Diagrama de clases

- **Prototipado: Diseño de pantallas**

Para este punto, he decidido incorporar el uso de una tecnología que no hemos visto durante el curso. Esta sería la aplicación de Adobe XD, que 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, dado a las múltiples guías, agrupaciones de componentes y demás herramientas de personalización que posee el editor, pudiendo incluir nuestras propias preferencias y elementos que queramos usar.

Muy similar a las demás suites de Adobe como Photoshop o Illustrator, este editor permite crear estructuras y diseños a través de variadas herramientas de forma o texto. La gran diferencias 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 aportará mucho a esta primera fase, no solo permitiendo una visualización básica de la plataforma creada, si no 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 crear una imagen más sólida de nuestra aplicación, muy útil para la identificación de la marca que hemos creado.

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 contendrá la información acorde a su función.

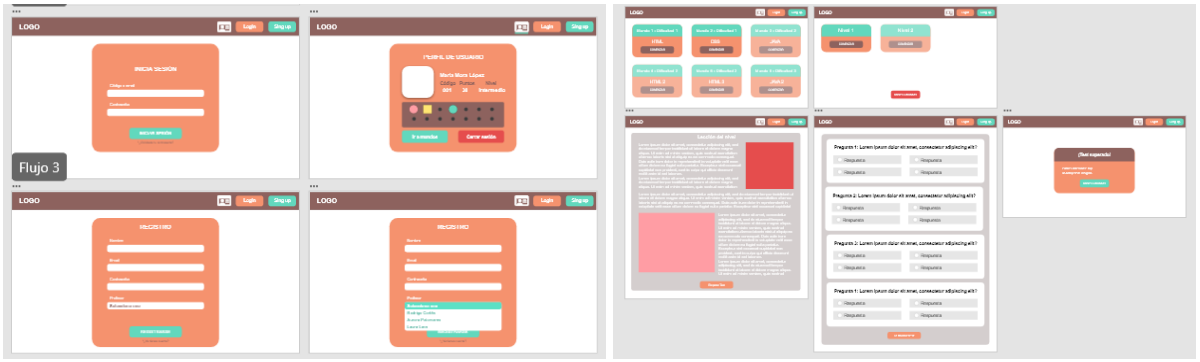
Una pauta que he tenido en cuenta a la hora del diseño en estas ha sido el sistema de rejillas de Bootstrap. Al hacer esto, la estructura será simétrica, ordenada y facilita la lectura de la web. También hace que el diseño responsive sea más fácil de lograr, simplemente agregando unas clases extras a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

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

Las muestras finales son las siguientes:

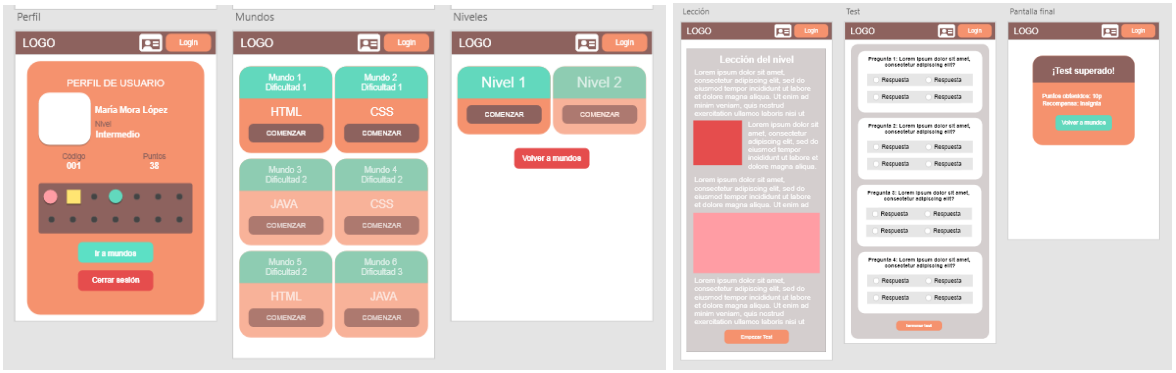
Vista en Web - 1920x1080

Pantalla inicial | Paleta de color empleada



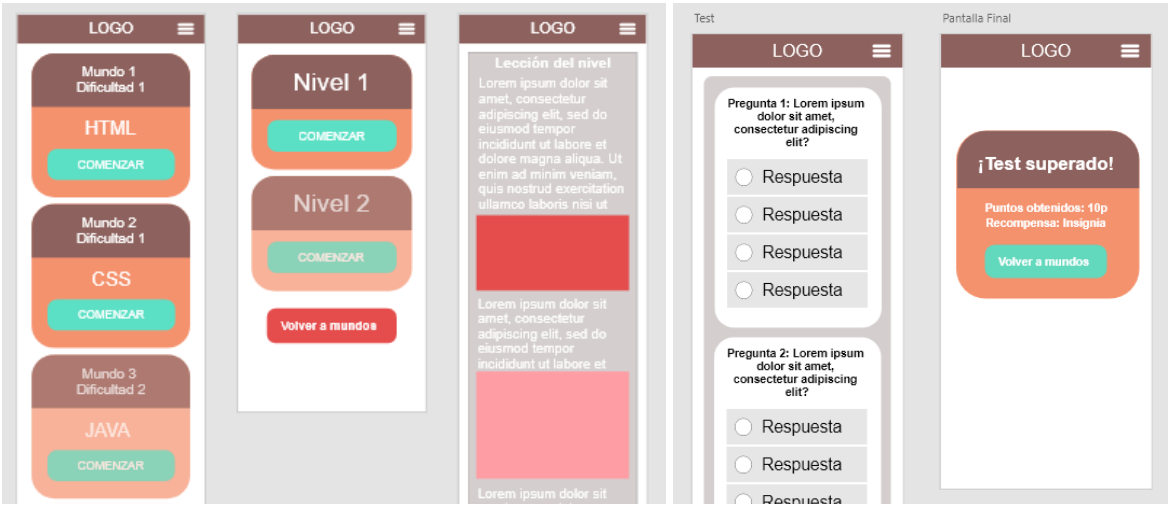
Vista en Tablet - Ipad Air 820x1180

Pantallas iniciales | Registro



Pantallas de perfil | Dinámica de mundos

Vistas en Dispositivo Móvil - Iphone 13/12 390x844



Vistas generales de móvil

1.4.2. 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, he decidido utilizar el formato de MySQL para la creación de esta y gestionarla a través del software de DBeaver.

DBeaver es una herramienta de administración de bases de datos. Para las bases de datos relacionales, utiliza la API de JDBC para interactuar con las bases de datos a través de un controlador JDBC.

Las tablas principales serían:

```
CREATE TABLE profesor(
        id INT CONSTRAINT profesor_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_profesor VARCHAR(1) NOT NULL UNIQUE,
        nombre VARCHAR(50) NOT NULL,
        correo VARCHAR(50) NOT NULL,
        contrasena VARCHAR(50) NOT NULL
);

CREATE TABLE alumno(
        id INT CONSTRAINT alumno_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_alumno VARCHAR(4) NOT NULL,
        correo VARCHAR(50) NOT NULL,
        contrasena VARCHAR(10) NOT NULL,
        icono VARCHAR(300) DEFAULT NULL,
        nombre VARCHAR(50) NOT NULL,
        puntos int DEFAULT NULL,
        CONSTRAINT alumno_profesor_id_profesor_fk FOREIGN KEY (id_profesor)
    REFERENCES profesor (id_profesor) ON DELETE CASCADE ON UPDATE
CASCADE
);
```

```
CREATE TABLE mundo (
        id INT CONSTRAINT mundo_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_mundo VARCHAR(2) NOT NULL,
        dificultad INT NOT NULL,
        puntos_desbloqueo INT NOT NULL,
        tematica VARCHAR(50) NOT NULL
);

CREATE TABLE nivel (
        id INT CONSTRAINT nivel_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_nivel VARCHAR(4) NOT NULL,
        dificultad INT NOT NULL,
        puntos_desbloqueo INT NOT NULL,
        puntos_minimos INT NOT NULL,
CONSTRAINT nivel_recompensa_id_recompensa_fk FOREIGN KEY (id_recompensa)
    REFERENCES recompensa (id_recompensa) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT nivel_test_id_test_fk FOREIGN KEY (id_test)
    REFERENCES test (id_test) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT nivel_leccion_id_leccion_fk FOREIGN KEY (id_leccion)
    REFERENCES leccion (id_leccion) ON DELETE CASCADE ON UPDATE
CASCADE,
CONSTRAINT nivel_mundo_id_mundo_fk FOREIGN KEY (id_mundo)
    REFERENCES mundo (id_mundo) ON DELETE CASCADE ON UPDATE
CASCADE
);

CREATE TABLE recompensa (
        id INT CONSTRAINT recompensa_pk PRIMARY KEY AUTO_INCREMENT,
        tipo VARCHAR(30) NOT NULL,
        ruta_recompensa VARCHAR(300) NOT NULL,
        desbloqueada BOOLEAN NOT NULL
);
```

```
CREATE TABLE leccion (
        id INT CONSTRAINT leccion_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_leccion VARCHAR(4) NOT NULL,
        contenido1 VARCHAR(500) NOT NULL,
        contenido2 VARCHAR(500),
        imagen_leccion VARCHAR(300)
);

CREATE TABLE test (
        id INT CONSTRAINT test_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_test VARCHAR(4) NOT NULL,
        terminado BOOLEAN NOT NULL
);

CREATE TABLE pregunta (
        id INT CONSTRAINT pregunta_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_pregunta VARCHAR(6) NOT NULL,
        texto_pregunta VARCHAR(100) NOT NULL,
        imagen_pregunta VARCHAR(300),
CONSTRAINT pewgunta_test_id_test_fk FOREIGN KEY (id_test)
      REFERENCES test (id_test) ON DELETE CASCADE ON UPDATE CASCADE,
);

CREATE TABLE respuesta (
        id INT CONSTRAINT test_pk PRIMARY KEY AUTO_INCREMENT,
        codigo_pregunta VARCHAR(6) NOT NULL,
        texto_respuesta VARCHAR(100) NOT NULL,
        correcta BOOLEAN NOT NULL,
CONSTRAINT respuesta_pregunta_id_pregunta_fk FOREIGN KEY (id_pregunta)
      REFERENCES pregunta (id_test) ON DELETE CASCADE ON UPDATE CASCADE,
);
```

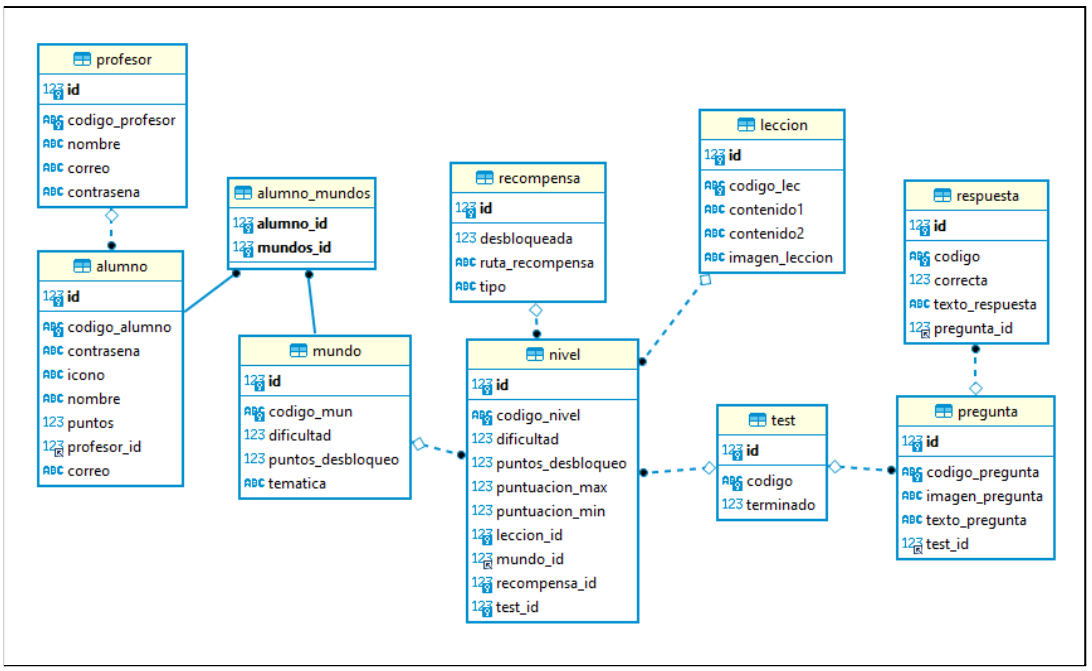


Diagrama de clases generado por DBeaver

- **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 que hay en las pantallas.

➔ **Visual Studio Code:** Editor de código fuente desarrollado por Microsoft. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.

Usado para la parte Front de la aplicación, 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. Comprende todo tipo de diagramas de base de datos relacionales, UML, esquemas de tablas. etc.
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 desde un sistema de almacenamiento 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.