UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TREBALL DE FI DE GRAU

TÍTULO DEL TFG: Juego de avatares para una herramientas de gamificación

TITULACIÓN: Grado en Ingeniería de Sistemas Aeroespaciales, mención Aeronavegación

AUTOR: Eva Clemente Escobar

DIRECTORES: Miguel Valero García
Roc Meseguer Pallarès

FECHA: 5 de septiembre de 2019

Título: Juego de avatares para una herramientas de gamificación

Autor: Eva Clemente Escobar

Directores: Miguel Valero García
Roc Meseguer Pallarès

Fecha: 5 de septiembre de 2019

Resumen

El desarrollo de este proyecto se basa en la aportación de un módulo a una aplicación de gamificación para la educación llamada Classpip, que está compuesta de varios módulos. Todos los proyectos que forman Classpip se han implementado siguiendo una arquitectura de software común, desarrollada con diferentes herramientas de código abierto.
El objetivo de este proyecto es el desarrollo de un juego destinado a la creación de avatares. El docente gestiona el juego de avatares de las diferentes clases desde el dispositivo móvil, para poder poner en marcha el juego durante la clase. Con el material seleccionado por el profesor, el alumno tendrá a su dispoción material suficiente para crear el avatar con diferentes elementos y que sea de su agrado.
El concepto base de Classpip es la gamificaión, una herramienta que utiliza puntuaciones, niveles, recompensas y otras dinámicas que usan los videojuegos para cualquier ámbito no lúdico. Los principales ámbitos en los que se utiliza son el académico y el laboral, que aunque difieren los objetivos de un entorno a otro, las técnicas son muy similares. Un estudio aplicado a estos dos contextos refleja el impacto de la gamificación y los aspectos que ésta potencia.
Para dar una idea cómo nació Classpip se explica toda la arquitectura de la que parten todos los módulos. Se necesita un aporte de diferentes tecnologías externas a la aplicación con el fin de poder hacer un proyecto completo, ya que no se extiende sólo a una plataforma web, sino que también ha sido versionada para tablets y smartphones.
Una vez se ha introducido Classpip, será necesarió seguir presentado nociones nuevas como la de avatar. Concebir un avatar no es algo sencillo y menos cuando no se tiene una base teórica de qué es. Por eso se especifica de dónde proviene esta idea, qué finalidad tiene y qué aplicaciones la utilizan más actualmente. De las aplicaciones que la utilizan hoy en día se recogen las más atractivas para poder llegar a desarrollar algo similar que pueda captar al público de Classpip.
Teniendo los objetivos a cumplir es mucho más sencillo comenzar a construir el módulo. Primero se habla de las funciones que quieren implementarse estableciendo un diseño, sujeto a modificaciones, para cada una de ellas. Sobre estas funciones y el diseño se desempeña la implementación real, dando detalle sobre todos los conceptos y finalmente se presenta el resultado final, especificando tanto diseño como funciones con su respectiva mecánica.
Esencialmente, este documento trata de analizar y explicar todas las funcionalidades que desempeña el juego de avatares. Además se evalúa el módulo completo en cuestiones de diseño, funciones y lo intuitivo que es para orientarlo a cualquier usuario independientemente de su experiencia con la tecnología.

Title : Avatar game for a gamification tool

Author: Eva Clemente Escobar

Advisors: Miguel Valero García
Roc Meseguer Pallarès

Date: September 5, 2019

Overview

The development of this project is based on the contribution of a module to a gamification application for education called Classpip, which is composed of several modules. All the projects that compose Classpip have been implemented following a common software architecture, developed with different open source tools.
The main objective of this project is to achieve the development of a game for avatars creation. The teacher manages the avatars game of the different classes from the mobile device, to be able to launch the game during the class. With the material selected by the teacher, the student will dispose from enough material to create the avatar with different elements, fulfilling his will.
The concept base of Classpip is the gamification, a tool that uses scores, levels, rewards and other dynamics that games use for any non-recreational environment. The main areas in which it is used are at academic and at work, although the objectives differ from one environment to the other, the techniques are very similar. A survey applied to these two contexts reflects the impact of gamification and all the aspects that it enhances.
To give an idea about how Classpip was born, it is explained all the architecture from which all the modules begin. A contribution of different technologies external to the application are needed in order to be able to build a complete project, since it does not extend only to a web platform, but has also been versioned for tablets and smartphones.
Once Classpip has been introduced, it is necessary to introduce new notions such as avatar. Conceiving an avatar is not a simple thing, moreover if you do not have a theoretical basis of what it is, it's even more difficult. That is the reason why it is explained where this idea comes from, which is its purpose and what applications are currently used for avatars. From the applications that are in use it today, the most attractive ones are collected to be able to develop something similar that can capture the public of Classpip.
When the objectives to meet have been set is much easier to begin building the module. First we talk about the functions that we want to get implemented by establishing a design (attached to modifications) for each of them. About these functions and the design established, the real implementation is performed, giving details on all the concepts and by last the final result is presented, specifying both the design and the functions with their respective mechanics.
Essentially, this document tries to analyze and explain all the features played in the avatars game. Furthermore, the complete module is evaluated in terms of design, functions and how intuitive it is to guide any user regardless of his experience with technology.

A mi familia,
por haber hecho posible que pueda llegar a ser lo que siempre quise ser

A mis compañeros de piso,
por haberse convertido en familia y por hacer de un lugar extraño un hogar

A Anna Ruíz,
una gran amiga y uno de mis puntos de apoyo más fuertes durante toda la carrera

A Miguel y Roc
por haber invertido tiempo en aconsejarme y guiarme durante todo este trabajo

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La tecnología es una herramienta de la que cada día se hace más uso. Además las nuevas teconologías de hoy en día están en constante actualización, por lo que a veces se quedan rápidamente obsoletas. La invasión del gigante tecnológico es tal que desde hace unos años ha pasado a estar al servicio del aprendizaje.

Desde temprana edad los niños ya interactúan con tablets y smartphones, por lo que introducir la tecnología en las aulas no ha causado un gran impacto para ellos. Sin embargo la brecha digital entre los estudiantes de las nuevas generaciones y los docentes es muy grande todavía, por tanto muchos docentes tendrán que especializarse y conocer cómo aplicar el uso de las nuevas tecnologías en la educación.

Una de las herramientas tecnológicas enfocadas a un ambiente de aprendizaje es la Gamificación. Utiliza las mismas dinámicas que los juegos aplicadas al ambiente académico, aplica puntuaciones y todo tipo de recompensas en competiciones individuales y grupales y la intención de este útil es mantener la motivación. Qué es la gamificación, cómo funciona, qué aplicaciones existen y qué consecuencias tiene utilizar la gamificación son cuestiones sobre las que habla en profundidad en el capítulo 1, donde además de estas cuestiones se aportan información y datos de estudios reales relacionados con su impacto.

El propósito de este proyecto es hacer un aporte a Classpip, una aplicación de gamificación desarrollada con enfoque a la educación. Esta aplicación es un proyecto colaborativo, desarrollado por diversos estudiantes de la EETAC. Classpip está compuesto por diferentes piezas, que son módulos autónomos creados de manera independiente que siguen una misma línea de desarrollo. Puesto que este proyecto es una pieza más que se suma a la aplicación, la plataforma ya está implementada, con una arquitectura de software y unas tecnologías de desarrollo establecidas. Las herramientas con las que se ha trabajado para este proyecto forman parte de la tecnología moderna de desarrollo para aplicaciones en internet. Todo lo relacionado con la implementación de Classpip y la línea de desarrolo que siguen los diferentes módulos que la componen aparece en el capítulo 2.

La finalidad es desarrollar un módulo orientado a la creación de avatares. La aplicación de Classpip, previamente al desarrollo de este módulo ya disponía de avatares, pero venían predeterminados con cada tipo de usuario. Con el aporte de un juego de avatares, no sólo se aporta creatividad a la aplicación, sino que también se busca la motivación del alumno, que para poder decorar el avatar tendrá que cumplir con los requisitos establecidos por el docente. El docente gestionará el material para el avatar y concederá los permisos de personalización a los alumnos. Mientras tanto los alumnos tendrán que estar al tanto de cumplir con los requisitos establecidos si quieren sacarle todo el partido al juego.

Las secciones de administración que están disponibles desde el usuario del docente se han diseñado e implementado de manera que el mecanismo para crear el juego de avatares sea sencillo. Las ideas para implementar y diseñar estos apartados pueden verse con detalle en el capítulo 4.

El documento se ha organizado de tal manera que se pueda seguir el hilo sobre el que se ha desarrollado el proyecto. Para empezar es primordial la introducción a la Gamificación, que es el concepto en torno al que gira el proyecto. Otro de los conceptos con mayor importancia es Classpip y las herramientas de software que han permitido su desarrollo.

Además se introduce la idea de avatar en el capítulo 3, a partir de dónde nace este nuevo módulo. Por útilmo se hace incapié en el desarrollo del juego de avatares, incluyendo detalles de software y herramientas externas.

Para culminar, con expectativas de poder mejorar en un futuro, se recogen datos a cerca de una evaluación de la aplicación. Qué tan intuitiva es, si el diseño es apropiado para el público al que va dirigida son algunas de las cuestiones que se exponen en la evaluación.

CAPÍTULO 1. LA GAMIFICACIÓN

El creciente uso de smartphones, tablets y otros muchos dispositivos tecnológicos entre los niños ha provocado que los métodos de enseñanza tradicionales queden algo obsoletos.

La brecha digital que se ha abierto entre docentes y estudiantes es evidente, y ha llevado a combinar la educación con la tecnología. El objetivo es implicar a una nueva generación de estudiantes digitales para hacer las clases más motivadoras. Los niveles, puntos, rankings, las metas y objetivos o crear equipos para el desarrollo de tareas como si de un videojuego se tratase, son métodos para gamificar un aula.

Teniendo en cuenta esta nueva necesidad de convertir el aprendizaje en una especie de juego nace Classpip, una aplicación del Departamento de Arquitectura de Computadores de la Universidad Politécnica de Cataluña desarrollada por estudiantes universitarios. La aplicación es accesible desde cualquier ordenador, dispositivo móvil o tablet.

El objetivo de este proyecto es crear un juego de avatares que se incorpora a Classpip para que los usuarios puedan tener una representación gráfica en la aplicación. De esta manera se puede aportar un toque más personal al juego.

1.1. Qué es gamificación

La gamificación es una técnica que integra estrategias de juego en diferentes ámbitos no lúdicos, y tiene como objetivo motivar y amenizar las tareas llevadas a cabo en dichos ámbitos. Los sectores que han implementado mayoritariamente esta técnica son: el laboral y el académico. La gamificación en los negocios potencia aspectos psicológicos dentro de los procesos empresariales, mientras que en la educación hace del aprendizaje algo divertido y llevadero. Mediante la motivación, se consigue un desarrollo eficiente de las tareas llevadas a cabo por los sujetos que trabajan con herramientas de gamificación. Las recompensas, puntuaciones y los rankings permiten dar reconocimiento, incentivando así el ánimo de superación.

Para mantener la expectación de los jugadores, es necesario estar constantemente innovando en el juego. Competiciones, insignias y privilegios son algunos elementos que pueden utilizarse. Las dinámicas de juego se basan en actividades con una determinada estructura, orientadas a un propósito u otro. Dichas actividades hay que dinamizarlas, ambientarlas para que resulten agradables y divertidas de cara al público.

Las competiciones se pueden utilizar como método de evaluación cada cierto tiempo. Puesto que muestran un ranking de puntuación, queda reconocido el mérito de los competidores y conlleva a recompensar a los mejores posicionados. Esta recompensa obtenida es el es el incentivo para despertar la motivación del usuario que, si surte efecto, implica la progresión de las cualidades del individuo.

Para tener una idea menos abstracta, pongamos un ejemplo. El docente puede asignar puntos a los estudiantes por diferentes motivos como puntualidad o buen comportamiento. La suma de todos estos puntos durante un periodo de tiempo lleva a clasificar a los alumnos dentro de un ranking. Los tres primeros podrán optar a diferentes privilegios como sentarse con el compañero que quieran en clase o poder repetir una pregunta de

exámen.

En los siguientes apartados se exponen diferentes entornos en los que puede aplicarse la gamificación.

1.2. Gamificación en el sector laboral

El mundo empresarial y el sector docente son los principales consumidores de métodos de gamificación, sin embargo los objetivos finales son diferentes.

En los últimos años grandes empresas de todo el mundo han incorporado la gamificación para poder alcanzar varios objetivos, ya sea para mejorar la eficiencia de los trabajadores, aumentar las ventas de un producto, optimizar los procesos de onboarding o mejorar la formación del personal, es una herramienta que puede resultar útil en multiples áreas.

Buscando información y documentandose sobre la gamificación, no hay artículo que de una mala opinión sobre esta herramienta, pero para ir un poco más allá es preferible saber la opinión directa de la persona que realmente trabaja en un entorno gamificado. El estudio ”The 2018 Gamification At Work” de Talent MLS recoge encuestas a diferentes usuarios de gamificación para que podamos visualizar por qué es tendencia.

Como resultado general, un 80 % de los entrevistados se divierte aplicando la gamificación en sus tareas del trabajo. Dentro de este entorno se utilizan las clasificaciones por niveles, puntos, insignias, recompensas y tablas de clasificación. De todo esto lo valorado como más motivador fueron las clasificaciones por niveles, mientras que lo peor puntuado fueron las recompensas. La clasificación aparece en el siguiente gráfico, donde además de clasificar de mejor al peor puntuado, a la derecha aparece la valoración sobre 5 que ha recibido cada elemento.

Figura 1.1: Ranking de elementos de gamificación

Lo que las empresas pretenden es que la gamificación tenga un impacto en sus empleados, tal que su productividad y motivación aumente, así que a continuación veremos qué

es lo que opinan sobre esto los que han respondido a las encuestas para este estudio.

1.2.1. Productividad

La definición de productividad es la cantidad de trabajo que un empleado es capaz de realizar en un determinado período de tiempo. La opinión subjetiva de los usuarios, tal y como reflejan los gráficos que aparecen a continuación es que la gamificación les hace más productivos.

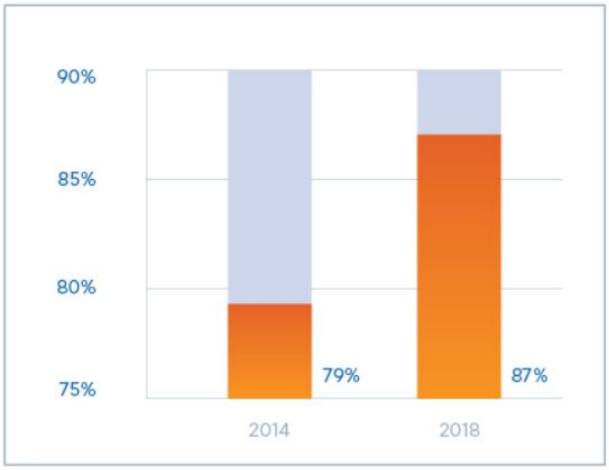


Figura 1.2: Impacto potenciador de la productividad

No es sorprendente que al hacer de una tarea un juego, sea más amena, y es que el 87% de los sujetos que han participado en el estudio de Talent MLS han afirmado que la gamificación sí que ha potenciado su productividad. En un estudio similar del 2014, el 79% ya arfirmaban esta cuestión,por tanto a medida que la gamificación se va abriendo paso en el mundo laboral los empleados se comprometen más con la causa.

1.3. Gamificación en las aulas

En la educación todo lo que se pretende es mantener motivado al estudiante para que la tarea de estudiar no suponga un esfuerzo. Hacer un examen o hacer los deberes dejan de ser tareas serias y aburridas. Aplicando la gamificación, como si de un juego se tratase, tiene como consecuencia conseguir una recompensa para intentar estar entre los mejores clasificados del ranking. Todas las dinámicas de juego que se aplican, ofrecen como finalidad llevar al alumno a implicarse en el juego para completar sus objetivos educativos.

Para valorar la experiencia del alumnado, se tiene como referencia un artículo que estudia el uso de la gamificación en la enseñanza superior, concretamente en la Asignatura de Derecho Romano de la Universidad Autónoma de Madrid.

Para fomentar la atención en las clases Magistrales y la motivación para el aprendizaje de la asignatura Se llevaron a cabo dos tipos de actividades, una de ellas con la herramienta Kahoot, durante el final de las clases Magistrales y otra de ellas es la creación de un Juego a través de un Moodle para hacer un repaso final de la asigantura de cara al Examen final.

Figura 1.3: Pregunta de evaluación realizada con Kahoot

Ambas pruebas tienen como objetivo mejorar el aprendizaje, para así tener un método de evaluación complementario a las pruebas calificatorias y de evaluación. Pero, ¿Cuál es el objetivo en torno al que gira esta invertigación? Pues bien hay varios estudios aplicando la gamificación en diferentes categorías de educación, pero los datos que aportaban eran escasos. Así que el propósito de éste estudio es comprobar empíricamente cuáles eran los efectos de estas nuevas técnicas de aprendizaje.

1.3.1. Diversión

A la pregunta ”¿Ha sido divertido competir con otros?”, un 65 % de estudiantes está completamente de acuerdo, otro 31 % está más bien de acuerdo, mientras que sólo hay un estudiante que ha respondido estar más bien en desacuerdo y otro que se muestra neutral.

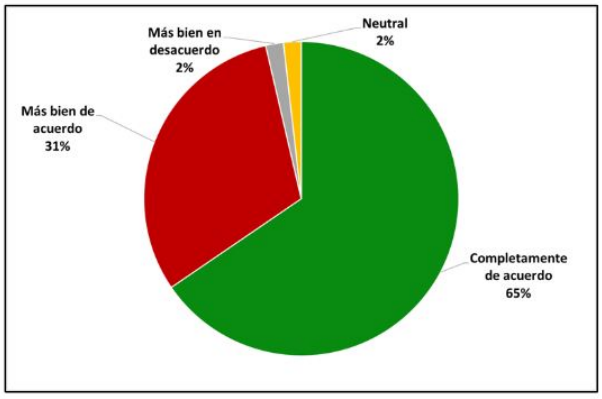


Figura 1.4: Respuestas a la pregunta ”¿Ha sido divertido competir con otros?”@

Los resultados recogidos muestran que la diversión es uno de los factores principales que llevan al estudiante a involucrar las herramientas de gamificación. Dar un añadido de diversión tiene un efecto que disminuye la tensión del ambiente universitario, que suele ser mucho más serio que el ambiente de cualquier estudio de educación inferior.

1.3.2. Motivación

Contestando la pregunta “El uso de Kahoot me ha motivado a asistir a clase”, un 38,4 % delos encuestados está completamente de acuerdo, un 18,2 % de los participantes están

más bien de acuerdo, mientras que un 38,2% se mantiene neutral y un 5,5% está en desacuerdo.

Respecto a la motivación, los diferentes datos recogidos reflejan que una mayoría de los alumnos ha conseguido sentirse más motivada para asistir a clase y mantener una mayor antención. Dependiendo de la persona el nivel de motivación era mayor o menor, a algunos les proporcionaba un nível de motivación alto mientras que en otros no surtía efecto.

1.3.3. Mejora del aprendizaje

La afirmación referida al aprendizaje, “Creo que Kahoot ha mejorado mi aprendizaje de la materia Derecho Romano”, es apoyada por el 32,7% que están completamente de acuerdo y por un 41,8% que dice estar más bien de acuerdo, contrastando con un 3,6% que opina que está más bienen desacuerdo y un 20% que se mantiene neutral respecto a lacuestión.

La mayoría de los estudiantes afirma haber logrado una mejora en las técnicas de aprendizaje de Derecho Romano gracias a las herramientas de gamificación. Hay que tener en cuenta que estos resultados se basan en la experiencia personal del usuario, no en los resultados reflejados por las pruebas de evaluación, motivo por el que se pretende hacer un estudio a posteriori de la evolución basado en las pruebas de evaluación para ver el veradero impacto de la evolución.

Concluyendo con los resultados generales, la experiencia de introducir una aplicación de gamificación en la materia de Derecho Romano, ha resultado una experiencia divertida, motivadora e incluso ha conseguido mejorar el aprendizaje de los usuarios.

1.4. Aplicaciones móviles para la gamificación

Desde que el uso de la gamificación se ha ido extendiendo, han aparecido numerosas aplicaciones para móvil que servían como herramienta de gamificación. Primero hablaremos de Kahoot, la aplicación utilizada en el experimento de la asignatura de Derecho Romano del que hemos hablado en el apartado anterior. Después presentaremos Atrivity, una plataforma que se utiliza como herramienta de gamificación en empresas.

- Kahoot

  Kahoot es una herramienta de gamificación que utiliza para aprender y repasar conceptos de cualquier tipo de una manera muy entretenida. Puntua no sólo el acertar la respuesta correcta, sino también la velocidad con laque se responde, puntuando mucho más alto conforme menos tiempo se haya tardado en responder. De manera análoga, si se comete un fallo la puntuación resta y cuanto menor sea el tiempo en cometer el fallo, más son los puntos negativos. La finalidad de esto es crear un ranking por puntos en el que aparezcan todos los participantes.

  Cualquier persona puede crear ün Kahoot”, un test de preguntas cualesquiera. Para hacer que otras personas participen en el test creado Kahoot asigna un número PIN al test creado, para que éste aparezca al introducirlo en la aplicación móvil.

Es una aplicación muy utilizada tanto en la docencia como en el ámbito laboral dado que está al acceso de cualquiera.

- Atrivity

  Atrivity es una plataforma que promueve el desarrollo de los empleados en su entorno de trabajo. Un conjunto de juegos recogen los contenidos formativos para potenciar las habilidades de los usuarios y hacer más agradable las fases de formación de un empleado.

  El administrador tiene el privilegio de acceder a informes analiticos detallados en la interfaz web mientras los jugadores compiten desde sus smartphones y tablets. Basado en los resultados, aparecerá un ranking que le de vida a la competición e impulse la participación. Informes de eficiencia, comparativas de los datos con las pérdidas y ganancias de la empresa y muchas otras herramientas para medir el impacto de la gamificación en la empresa son las que ofrece Atrivity.

  Es uno de los juegos que las empresas están usando más para mejorar el rendimiento, hacer el onboarding de forma rápida y motivar a los empleados.

- Zombies, Run!

  Una manera muy buena de sacar todo el potencial de un corredor es hacerle creer que está siendo perseguido por Zombies.

  Al iniciar la aplicación, comienza a narrarse una historia muy convincente sobre un apocalipsis zombie durante la que se escuchan gruñidos de zombies y compañeros que utilizan la aplicación, todo esto mientras un operador de radio dirige al deportista.

  Los papeles que desempeña el corredor dentro de la aplicación son los de repartidor de suministros y combatiente de villanos para intentar salvar la raza humana.

- fold.it

  Foldit es una plataforma dedicada a la investigación científica que busca diferentes patrones de doblaje de proteínas.

  Es un proceso largo y costoso que requiere de mucha investigación. Por este motivo se lanza este juego que permite construir estructuras viables. El juego consiste en hacer puzzles, que completandose con éxito contribuyen a la investigación científica del proyecto.

  Cualquier usuario interesado en el proyecto puede unirse al juego cuando lo desee.

CAPÍTULO 2. INTRODUCCIÓN AL PROYECTO

Classpip es una aplicación para la gamificación escolar. Ha sido creada por el Departamento de Arquitectura de Computadores de la Universidad Politécnica de Cataluña y han participado en el desarrollo del software diversos alumnos de la Escola d'Enginyeria de Telecomunicacions i Aeroespacial de Castelldefels (EETAC). Por lo general los alumnos aportan su contribución a la aplicación para realizar el trabajo de fin de grado.

La aplicación está compuesta por diferentes módulos que se han desarrollado de manera independiente, cada uno de ellos con diferentes funcionalidades y propósitos. El primer proyecto de Classpip fue el de arquitectura de software, que sentaría las bases de las herramientas para desarrollar el software de la aplicación.

Antes que nada, para poder introducir el proyecto es necesario conocer cómo está estructurada la aplicación de Classpip, y su arquitectura, que serán los puntos a explicar en los siguientes apartados.

2.1. Arquitectura del Proyecto

La arquitectura en programación, indica la estructura, funcionamiento y la manera en la que interaccionan las diferentes partes del software.

La arquitectura software en la que está basada el proyecto se escogió en el primer proyecto de Classpip y está compuesta fudamentalmente por una aplicación móvil, ideada para hacer actividades de corta duración en horario de clase, una interfaz para el administrador, creada para profundizar en los apectos relacionados con el software de la aplicación, y una arquitectura orientada a servicios, donde están creados los métodos que permiten el manejo de la información de la base de datos.

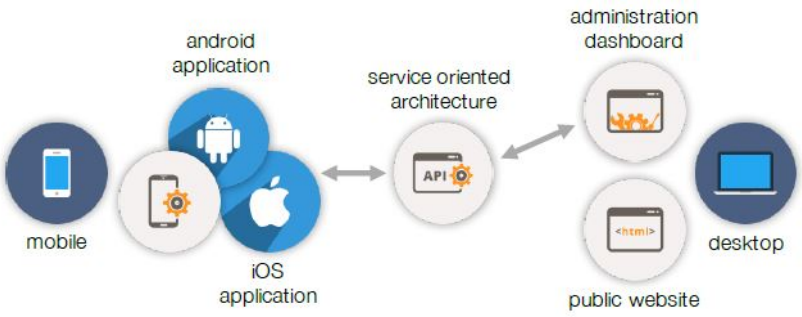


Figura 2.1: Arquitectura Classpip

Todas las herramientas utilizadas para desarrollar software son de código abierto, pero como ya hemos comentado, hay más de una pieza en el software y no todas ellas necesitan de las mismas tecnologías para ser desarrolladas. Ahora haremos una pequeña introducción a las diferentes partes en las que se divide la aplicación y en este apartado profundizaremos en las tecnologías que se usan en cada una de ellas y cómo están relacionadas entre sí. La parte de funcionalidades y aspecto será comentada en la 2.2.

Para ver de manera gráfica esta arquitectura, a continuación aparecen los componentes y las respectivas herramientas que se utilizan en cada uno.

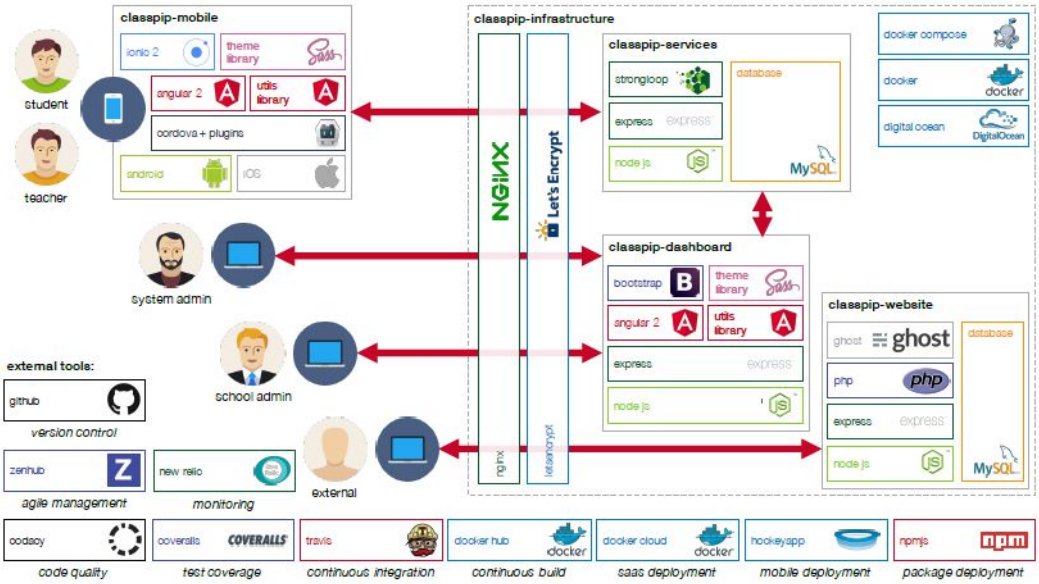


Figura 2.2: Arquitectura Classpip

- Arquitectura software para móvil

  En la aplicación móvil el software se desarrolla con ionic, un marco de desarrollo o ”framework”que proporciona la interfaz que va a conectar el código con la vista de una aplicación para móvil. El código se desarrolla con Angular, otro framework que se utiliza para el desarrollo web. Con estas dos herramientas parece ser suficiente, pero nos estamos olvidando de que actualmente existe más de una plataforma para móvil: android, iOs, Windows.... y eso es algo que se soluciona añadiendo Cordova, un entorno de desarrollo de código libre que permite crear aplicaciones para móvil utilizando CSS, HTML y JavaScript.Así no hay que usar una plataforma específica que sirva para un único tipo de servidor.

  El CSS es la parte donde se codifica el estilo de la vista, de qué color es un elemento, de qué tamaño son los botones o de qué color es el fondo. El HTML lleva el código de la vista, lo que el cliente va a ver a través de la pantalla. y por último el JavaScript es un lenguaje de programación que permite crear funciones.

  Para acompañar un poco a los estilos CSS, se hace uso de Sass, una herramienta para hacer que los estilos CSS sean más dinámicos y haya que escribir menos código.

- Arquitectura software para web o Dashboard

  El desarrollo de código en el Dashboard también es mediante Angular, así puede utilizarse tanto para web como para móvil y los estilos CSS se complementan con Sass de nuevo.

  Se añade NodeJs, un entorno de ejecución de múltiples plataformas. Es complicado de clasificar, porque sirve para desarrollar diferentes tipos de tareas a la vez, permite elaborar APIs, aplicaciones web o hasta aplicaciones de mensajería push. Con toda esta gama de funciones, es un componente básico si se pretende desarrollar una aplicación del tipo que sea. Se añade un framework que conecta con la interfaz llamado Express, útil para desarrollar la aplicación en versión móvil y web.

Otra novedad que aparece en esta parte de la aplicación es Bootstrap, una interfaz que permite que el código se adapte a la pantalla que sea y trabaja con estilos predefinidos para CSS muy sencillos.

- Arquitectura web orientada a servicios

  La parte de servicios es la pieza central de la arquitectura, es el canal de comunicación entre el proyecto Dashboard y la aplicación móvil. Comunicar datos requiere hacerlo mediante direcciones IP, y una interfaz de red que haga de conexión entre los puntos que se quiere comunicar.

  Se implementa el código utilizando Express y NodeJs como en la web, y se añade Loopback StrongLoop, una interfaz de red virtual que es capaz de crear varias interfaces virtuales para una conexión física. Esta interfaz a su vez precisa de un sistema de gestión de base de datos, que es MySQL, uno de los sistemas de código abierto para gestión de datos más utilizado. Se explican con más detalle estas dos herramientas de Loopback en el apartado de Herramientas Externas.

2.1.1. Herramientas externas

Para poder construir la aplicación desde cero son necesarias diferentes herramientas, ya sea para administrar la base de datos, añadir elementos de diseño o poder guardar un repositorio del código.

- Git y GitHub

  Git es uno de los sistemas de control de versiones más conocidos. Para administrar cualquier proyecto de desarrollo de software es fundamental tener un control sobre las diferentes versiones del código. De esta manera se consigue no perder el estado del trabajo cuando se van a introducir cambios.

  Hay que hacer una distinción importante entre Git y GitHub, y es que Git es el sistema que administra las versiones, mientras que GitHub proporciona el hosting o alojamiento de los repositorios que administra Git.

  Este sistema permite ver y comprender rápidamente los cambios que tienen las versiones nuevas respecto a las anteriores, además de poder desechar los cambios y volver atrás gracias al historial de versiones, que permanecen siempre registradas.

- LoopBack

  LoopBack es una interfaz que sirve para crear la API REST que se utilizará en el proyecto. Definir el concepto API-REST, lleva introducir los conceptos API y REST por partes.

  Por lado una API (Application Programming Interface) recoge métodos, funciones o protocolos, para ser un poco más concretos una API define cómo un módulo de software interactúa con otro. Lleva implícitas una serie de funciones ya definidas para interactuar con el programa que sea, lo que resulta muy útil porque de no llevar funciones pre-definidas habría que programarlas desde cero.

  Por otra parte esta el concepto REST (del inglés, Representational State Transfer)se utiliza para obtener datos, realizar operaciones con datos en diferentes formatos.

Las operaciones básicas que utiliza un sistema REST con protocolos HTTP (protocolos de transferencia de datos mediante internet) son POST (crear un elemento nuevo de un modelo existente), GET (consulta de información ya almacenada), PUT (editar un campo de información específico) y DELETE(eliminar datos).

Uniendo los dos conceptos tenemos una interfaz que comunica los diferentes módulos de software y un sistema REST que es el que se encarga de hacer las operaciones con los datos transferidos entre módulos. Una solución sencilla para evitar tener que adaptar todo a una plataforma de desarrollo.

- Bootstrap

  Para la parte de diseño se utiliza Bootstrap, un conjunto de librerías de código abierto para diseño web. Pensado para adaptar la interfaz de usuario a todo tipo de pantalla (desarrollo web responsive), combinando elementos de diseño del contenido de sus librerías.

  Desde el lanzamiento de la versión 3, Bootstrap es totalmente compatible con:

  - Google Chrome (en todas las plataformas).
  - Safari (tanto en iOS como en Mac).
  - Mozilla Firefox (en Mac y en Windows).
  - Internet Explorer (en Windows y Windows Phone).
  - Opera (en Windows y Mac).

  Toda esta serie de características y compatibilidades convierten a Bootstrap en una buena herramienta para desarrollo web responsive.

2.2. Estructura de Classpip

Classpip está formada por tres piezas fundamentales, una aplicación móvil, un panel de administración y una arquitectura orientada a servicios que hace de conector entre las otras dos piezas.

Cada bloque tiene funcionalidades diferentes, y está pensado para un tipo de usuario concreto.

2.2.1. Aplicación móvil

El proyecto para aplicación móvil permite hacer uso de la aplicación desde cualquier smartphone o tablet. Esta parte está pensada para utilizarla en clase y los principales usuarios serán los alumnos. Realizar actividades mediante la aplicación móvil es rápido e intuitivo.

Hay tres tipos de rol que puede tener un usuario de classpip, docente, estudiante y administrador de la escuela. Dependiendo del tipo de usuario que inicie sesión. Como ejemplo veremos las diferentes pantallas y funciónes que aparecen si inicias sesión con todos los roles de usuario, el proyecto que se utiliza es el de Joan Valverde y Anna García, que fueron los alumnos que desarrollaron la primera versión vendible de Classpip.

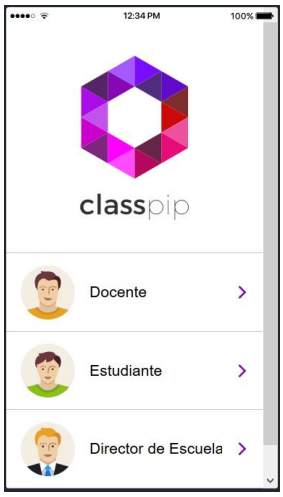
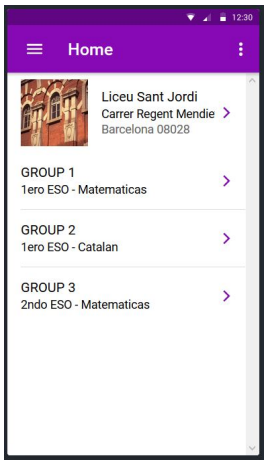


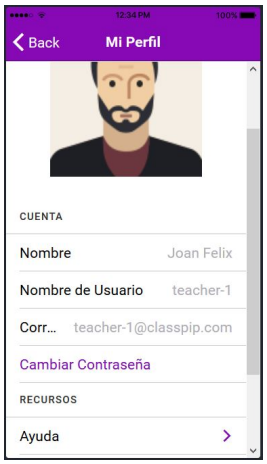
Figura 2.3: Pantalla de inicio de sesión Classpip

- Docente

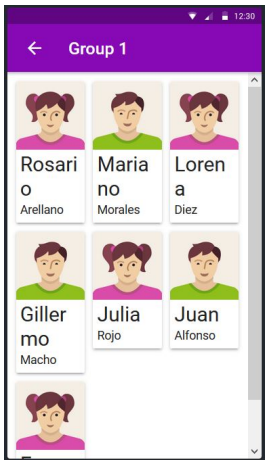
  Al inciar sesión como docente aparece un menú con diferentes apartados, en ”Mi perfil¨ se muestran todos los datos del docente: nombre personal, nombre de usuario, contraseña e incluso tiene una serie de funcionalidades como Ayuda, Privacidad y términos y hasta permite cambiar la contraseña del usurio si lo desea. En ”Mi Escuela¨ Adicional a estos dos apartados hay una vista de las clases que imparte el docente, cada una con su respectivo listado de alumnos y sus correspondientes datos.



(a) Menú inicio



(b) Apartado Mi Perfil



(c) Lista de alumnos de clase

Figura 2.4: Vista aplicación móvil para Docente

- Estudiante

  El inico de sesión desde un usuario de estudiante también contiene los apartados ”Mi perfil¨ , que incluye las mismas funcionalidades que se ofrecen en la vista del docente, y ”Mi Escuela¨ , que también muestra lo mismo que presenta para el do-

cente. Sin embargo un estudiante no puede ver la lista de alumnos de toda la clase ni mucho menos interactuar con ésta.

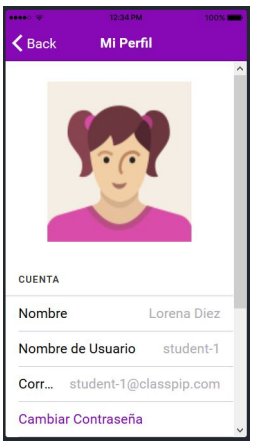


Figura 2.5: Apartado Mi Perfil

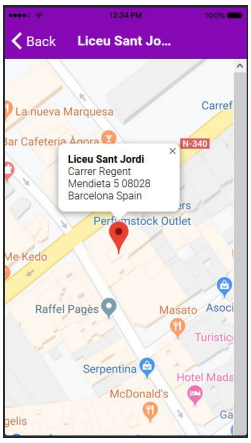
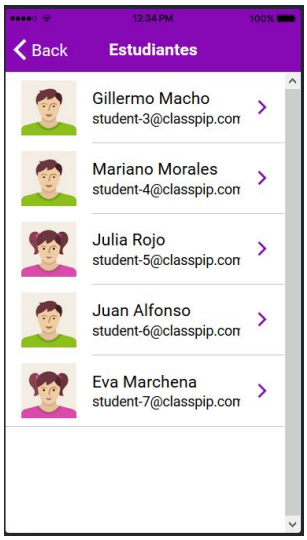


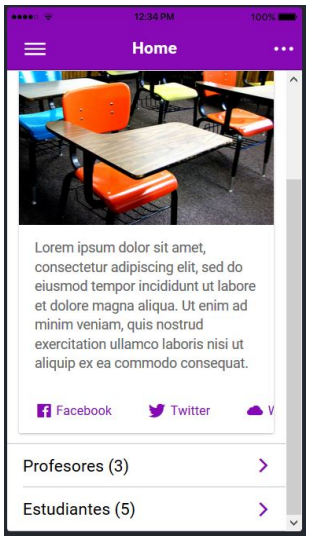
Figura 2.6: Localización Escuela

- Administrador

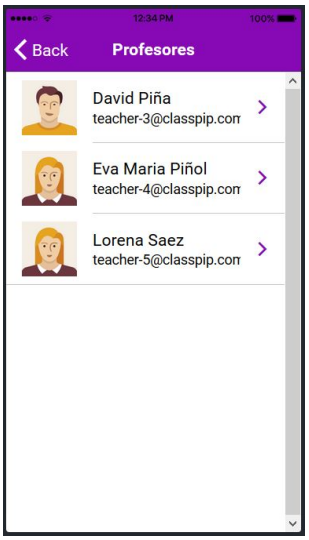
  El administrador de la escuela tiene el mismo menú que los dos roles explicados anteriormente, pero éste tiene le posibilidad de ver todos los estudiantes de la escuela registrados en la aplicación (ya no de una sola clase) y además también puede ver la lista de docentes del centro.



(a) Menú inicio



(b) Apartado Mi Perfil



(c) Lista de alumnos de clase

Figura 2.7: Vista aplicación móvil para Docente

2.2.2. Panel de administrador

Ya hemos visto las funcionalidades que tiene el panel del administrador en la aplicación móvil, pero el administrador, puesto que su papel conlleva tareas diferentes a las activi-

dades que se realizan en clase y le pueden llevar bastante tiempo, es más cómodo que acceda desde la versión para ordenador, o lo que llamamos ”dashboard”.

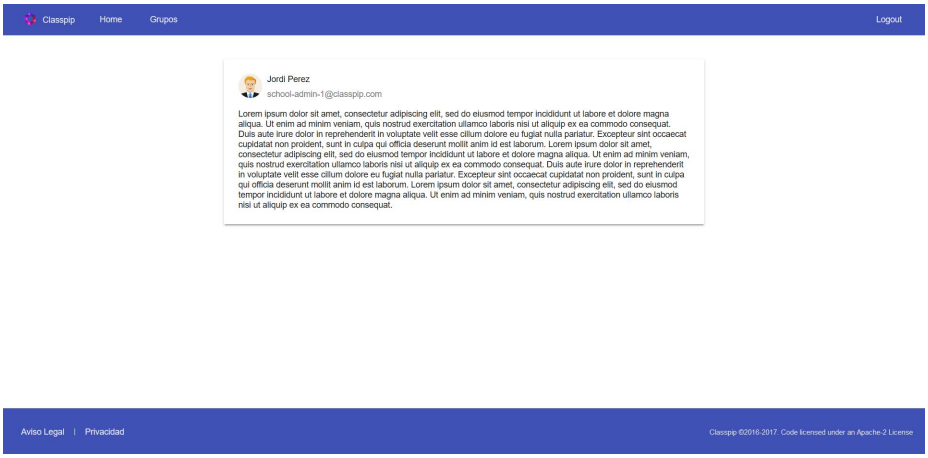


Figura 2.8: Dashboard de Classpip

2.2.3. Arquitectura orientada a servicios

La arquitectura orientada a servicios es uno de los componentes más importantes de la aplicación. La aplicación de Loopback aporta una REST API que nos permite administrar y realizar operaciones con la base de datos. En la API se exponen los modelos que se van creando y para cada uno de ellos contiene operaciones post, get, patch y todo tipo de funciones de protocolo http.

2.3. Motivación

El motivo de desarrollar un módulo de avatares para la aplicación viene del primer bloque que fué desarrollado para Classpip. Ferrán es el alumno que aportó el desarrollo de la arquitectura para la aplicación, sentando las bases y componentes necesarios para construir la plataforma de gamificación.

En su memoria, Ferran explica qué es un Producto Viable Mínimo (MVP, del inglés Minimum Viable Product), un producto con una serie de características hechas para satisfacer al cliente y proporcionar una reacción positiva para un futuro desarrollo del mismo. Todo esto forma parte de la estrategia a seguir para comercializar un producto. Las carácterísticas que definen al MVP están enfocadas a los clientes que se pretenden captar, así el producto tendrá una mayor acogida.

Uno de los requisitos que se definen en ese MVP es que cada alumno debería poder cambiar su avatar, por tanto de aquí nacela necesidad de crear un juego de avatares para la clase. En los primeros proyectos desarrollados para Classpip ya había avatares, pero eran una simple foto. Se pretende que el avatar sea algo más que eso, el alumno podrá construirlo dentro de la aplicación y dependiendo de su participación en clase, tendrá opción a personalizarlo más o menos. La finalidad del avatar es darle juego a la aplicación de la misma manera que lo hace cualquier otro módulo pero con funcionalidades distintas.

2.4. Objetivos del módulo

Un paso previo al desarrollo del proyecto es plantear una serie de objetivos. En este caso el objetivo del proyecto es implementar una aplicación móvil para aportar a Classpip un juego de avatares. Las funcionalidades que pretenden desarrollarse, son las listadas a continuación:

- Crear clases con un listado de alumnos propio. Cada clase tendrá un identificador (nombre) propio y un juego de avatares que por defecto estará desactivado.
- Añadir y eliminar alumnos de la lista de clase.
- Poder ver el listado de alumnos que pertenece a cada clase con algunos datos.
- Activar y desactivar el juego de avatares de cada clase cuando se crea conveniente.
- Añadir una familia de avatares. Cada juego tendrá opción a crear una familia de personas o de otro ente como puede ser un monstruo.
- Diseñar unos requisitos para administrar privilegios que permitan personalizar el avatar. El docente se encargará de establecer cuáles son los objetivos a cumplir para poder elegir un peinado o cualquier otro elemento disponible.
- Conceder los privilegios a los alumnos manualmente, cuando el profesor crea necesario.
- Crear un avatar, cuyo diseño dependerá de los requisitos que el docente haya establecido. Por ejemplo si el privilegio de cambiar los ojos no ha sido concedido, el alumno no podrá ponerle ojos al avatar.
- Cargar una vista de la lista de clase con los avatares personalizados de cada alumno.
- Añadir una galería de material, desde donde el profesor pueda cargar y ver los archivos de imágen para crear el avatar.

La estructura del proyecto tiene dos pilares, que son los roles de usuario que existen en la aplicación, el alumno y el profesor. Una vez dentro de la aplicación, encontramos todas las funciones para administrar el juego en la parte del profesor, mientras que en la parte del estudiante la tarea más importante que se desempeña es la de decorar el avatar. Ambos roles permiten modificar algo de la información personal de la cuenta como la contraseña, que es la única acción común.

La imágen que aparece justo en esta sección (**??**) sirve para tener una ligera idea de qué está al alcance de cada usuario y qué aporta este nuevo módulo. Cómo funciona el módulo y qué mecánica lleva el código de la aplicación se detallan más adelante en los capítulos 5 y 6, respectivamente.

CAPÍTULO 3. ESTADO DEL ARTE

3.1. Qué son avatares

Avatar es un concepto cuyo significado tiene varias definiciones, dependiendo del ámbito en el que se utilice. Mientras que antaño, en el hinduismo un avatar se considera una encarnación del dios Visnú, pasando a tomar apariencia humana o animal; Actualmente en el mundo occidental se asocia a un elemento gráfico que representa a un usuario en el ámbito de la informática y la tecnología.

La variante del concepto de avatar que se utiliza es la que se extiende al mundo digital. Aplicaciones, redes sociales y programas informáticos permiten que sus usuarios se identifiquen mediante algún recurso gráfico.

La Web 2.0 o web social fue la primera herramienta en utilizar este recurso, creando una red en la que los usuarios interactúan entre sí y se convierten en usuarios activos. El fenómeno social empezaba a desarrollarse con la sucesora de la Web 1.0, dando paso a blogs, redes sociales, espacios web colaborativos como Wikipedia, plataformas como Google Drive o Youtube. Toda esta cantidad de plataformas y servicios han conseguido crecer gracias a que los propios usuarios eran a la vez partícipes del contenido de éstas.

Desde este boom de la web 2.0 se han visto diferentes tipos de avatares, desde los más básicos hasta los que se mueven, tienen mascotas y complementos y tienen voz propia.

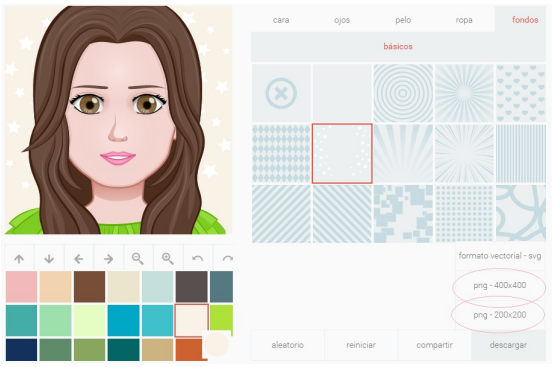


Figura 3.1: Busto de avatar

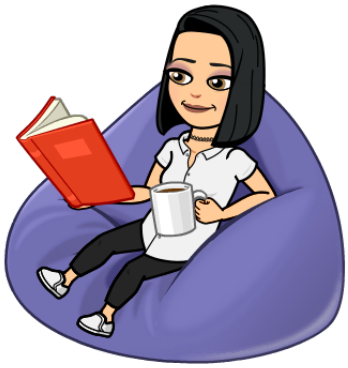


Figura 3.2: Avatar con complementos

3.2. Herramientas para avatares

En esta sección se han recopilado diferentes aplicaciones de avatares que utilizan actualmente plataformas de docencia o redes sociales.

El propósito de ver las cualidades de cada una de ellas, es poder extraer ideas para a partir de éstas, idear los avatares que se incorporarán en Classpip.

3.2.1. Voki

Voki es una herramienta que facilita el diseño de avatares personalizados en 2 dimensiones. Una de las características de esta aplicación es que sus personajes pueden moverse e incluso se les puede añadir voz. Se nos ofrece la posibilidad de elegir un personaje y un fondo. De entre los personajes se puede escoger una persona, político, famoso, animal o personaje de cómic. La ropa del avatar también es personalizable, dentro de la variedad que muestra la propia aplicación y el fondo puede ser uno de los proporcionados por la propia herramienta, o bien puede ser una aportación propia subiendo una imagen cualquiera. Podemos cambiar el color de la piel, de los ojos, la forma del pelo, etc.

Figura 3.3: Editor de avatar en Voki

Una de las ventajas que ofrece Voki es que permite organizar clases, para que se pueda utilizar en el ámbito docente. Se pueden configurar clases y permite que los estudiantes accedan a sus tareas.

3.2.2. Bitmoji

Bitmoji es una aplicación que permite crear un avatar personalizado. Está disponible en la tienda de Chrome, y hay una aplicación disponible para Android e iOS. También existe la posibilidad de acceder a la cuenta de Bitmoji a través de la web.

Muchas otras aplicaciones de las más utilizadas por los usuarios, tienen integrada la aplicación de Bitmoji, como Snapchat, Facebook, Facebook Messenger, Gmail, Gboard, Slack e iMessage. En caso de que la aplicación no esté integrada directamente, simplemente se puede copiar y pegar el avatar para utilizarlo.

Para hacer un Bitmoji, tan solo hay que descargar la aplicación. Luego, la imagen se puede guardar y compartir, o se puede sincronizar con Snapchat. Una vez creada se puede seguir editando tantas veces como se quiera. Además de la propia creación del avatar personalizado, permiten bastantes opciones para dejarlo al gusto del usuario. Se puede cambiar la ropa del avatar, cuyo armario va cambiando con el tiempo y en relación

a eventos. Los eventos deportivos suelen traer ropa deportiva que se añade al vestuario, y empresas como Pixar sacan packs de extensión de los personajes de sus películas.

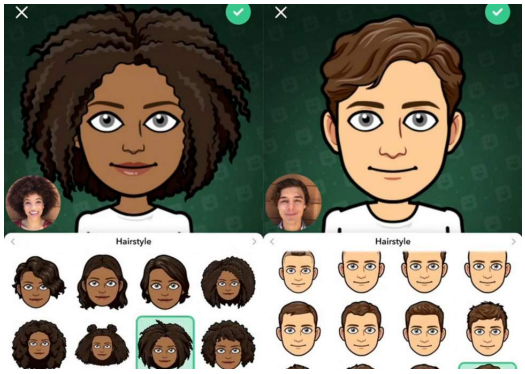


Figura 3.4: Editor de avatar en Bitmoji

3.2.3. Classdojo

Classdojo es una plataforma creada para gestionar el aula con la que pueden interactuar profesores, padres y estudiantes. Que pueda haber tres tipos de usuarios quiere decir que las funcionalidades varían, así que a continuación vermeos en qué consiste el papel de cada uno de ellos.

- Cuenta de docente

  Un aula puede estar administrada por uno o varios profesores, y de la misma manera un profesor puede estar al cargo de varias aulas.

  El profesor añade a los estudiantes y administra las aulas, teniendo la posibilidad de crear tantas clases como crea necesarias. Como todo juego tiene un sistema de asignación de puntos, que depende en todo momento del profesor. Si los puntos restan o suman, y qué cantidad de puntos se atribuye a cada alumno recae directamente sobre el docente. Otra funciones disponibles para el profesor son la mensajería directa con los padres y asignar insignias como las de la figura 3.5 a los alumnos.

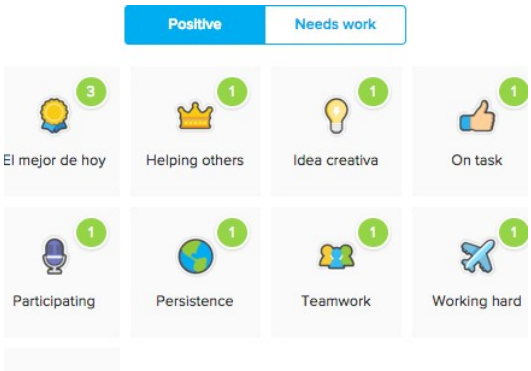


Figura 3.5: Panel de insignias

Algo que es de gran utilidad para el docente son los informes que genera Classdojo. Reflejan la evolución de la clase tomando como datos referentes el comportamiento y la puntuación de los estudiantes. Los datos de estos informes pueden exportarse a un fichero excel en caso de que se desee.

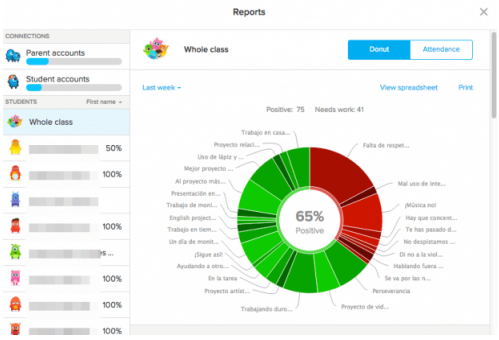


Figura 3.6: Informe de evolución de la clase

- Cuenta de alumno

  El estudiante es el rol que menos juego tiene en la aplicación. Puede personalizar un pequeño monstruo a modo de avatar y también ver las puntuaciones e insignias que le ha asignado el profesor.

Figura 3.7: Monstruo personalizado

- Cuenta de padre

  La familia del estudiante puede participar también en Classdojo. Un padre puede tener hasta varios alumnos a su cargo. Estar al tanto de cuáles han sido las valoraciones del docente, ponerse en contacto con el profesor a través de mensajes privados y visualizar la información online son las opciones al alcance de este tipo de usuario. Cabe decir que todas estas funcionalidades van ligadas a los alumnos de los que se hace cargo el padre, es decir, no es posible que vea la información de otras asignaturas, ni las calificaciones de otros estudiantes.

Una aplicación gratuita y accesible desde cualquier plataforma web, iOS y Android con versiones en siferentes idiomas. Todas estas carácterísticas convierten a Classdojo en una herramiente muy útil para la docencia.

3.3. Aspectos distinguidos

En esta sección se recogen algunas de las características que podrían resultar de inspiración para desarrollar el módulo de avatares.

Bitmoji, dispone de una función única de la que pocas aplicaciones de creación pueden fardar. No solo tiene muchisima variedad de caras, peinados y complementos, además es capaz de construir un avatar a partir de un ”selfie”. Es quizá la parte más atractiva del módulo, pese a que supone tener un manejo extraordinario de librerías de reconocimiento facial y podría llevar mucho tiempo desarrollar algo similar.

También cabe distinguir que las celebraciones especiales merecen una vestimenta o complementos especiales a veces. La Navidad, verano, Pascua, o incluso carnaval pueden pasar a formar parte del mundo del avatar en Bitmoji. Rompe un poco con la monotonía del avatar y podría mantener a los estudiantes más motivados si el juego va cambiando dependiendo de la época del año.

De Classdojo lo que destaca por su sencillez es el editor de avatares. Es un diseño muy fácil de implementar, funciona con coordenadas de profundidad, así superponiendo unos elementos sobre otros se construye el avatar con todos sus componentes. Es una de las características que se tendrán en cuenta a la hora de implementar el diseño del avatar.

Algo de Classdojo que también puede resultar atractivo a la hora de escoger una aplicación de gamificación para el entorno docente es incluir a los padres en una aplicación. Para estudiantes de temprana edad sobre todo, puede ser reconfortante que los padres puedan estar al tanto del comportamiento de sus hijos y tener un canal de comunicación directo con el docente. Pero este matiz no tiene sentido en un juego de avatares, quizá pueda resultar útil de cara al desarrollo de futuros módulos de Classpip.

Lo más fascinante de Voki es que los avatares puedan tener voz. No es una voz pregrabada, es la voz del usuario que grabada con el micro interno del ordenador queda registrada en la aplicación. Otra función que le daría un papel al avatar que no fuese sólo visual.

CAPÍTULO 4. CONCEPTO DE DISEÑO

Como concepto básico de un buen diseño es imprescindible tener en cuenta las opciones a las que pueden acceder los dos tipos de rol que acoge el módulo.

Los menús del profesor y del alumno son completamente diferentes. De hecho, el rol del profesor está pensado para hacer de administrador realmente, y por tanto tendrá el poder de conceder permisos y administrar las bases de datos a las que accede el juego.

4.1. Modelo de juego

Son varias las maneras de darle juego a este módulo. El usuario que tiene autorizado poner en marcha el módulo de avatares es el que tiene el rol de profesor. Tras iniciar sesión, el profesor puede ver sus clases y crear un juego de avatares para cada una de ellas.

El módulo está pensado para añadir una imagen característica a cada alumno, que podrá estar diseñada como este quiera. El avatar podrá tener tres elementos que irán variando los usuarios a su gusto, como el pelo, los ojos y algún complemento. La gracia del juego reside en que para modificar estos elementos, los usuarios deben tener permiso del administrador, y es que cada uno de los elementos tiene una condición a cumplir para conseguir el permiso.

En el apartado mi perfil aparecen los datos del usuario, que podrá modificar siempre y cuando crea necesario mediante un formulario de entrada. galería de material contendrá todos los elementos que podrán usarse para personalizar los avatares, existiendo la posibilidad de añadir y eliminar cuantos elementos sean necesarios. Una vez esté activado el módulo de avatares, los alumnos que utilicen la plataforma, encontrarán un busto de avatar desnudo al entrar por primera vez a ésta, que se le asigna por defecto cuando se activa el módulo de avatares en la aplicación.

De manera análoga a tomar la decisión de activar el módulo de avatares para cada clase, el docente tambien decide, dentro de cada clase, qué alumnos pueden modificar los elementos del avatar. Para esto tendremos 3 elementos para modificar: pelo, ojos y complementos. La vista que aparece de la lista de alumnos de una clase muestra, a parte de los datos personales de cada alumno, estas 3 casillas de elementos pertenecientes al avatar. Partiendo de la información que contiene dicha lista, el profesor puede generar una vista de los avatares, para poder mostrar a sus alumnos en clase cómo se ven todos los avatares de los miembros de clase, lo supone que cada alumno puede ver su avatar pero no el del resto.

4.2. Diseño funcional

En esta sección se explica cómo se van a diseñar las funciones de la aplicación de manera que cubra lo que se ha especificado en los objetivos del proyecto.

4.2.1. Rol del profesor

El rol de profesor permite acceder a un menú que permite ver sus datos personales, que podrá modificar cuando crea conveniente. Verá también las diferentes clases que tenga registradas en la aplicación, así como los alumnos que pertenecen a cada una de ellas. Para cada clase podrá crear un juego de avatares.

- Añadir clases

  El aministrador de la aplicación puede crear tantas clases como crea que son necesarias. Cada clase tiene un nombre único, que es el identificador de la asignatura. Para cada asignatura se dispone de una lista de alumnos y un juego de avatares que viene desactivado por defecto.

- Crear un juego de avatares

  Todas las clases disponen de un juego de avatares, que podrá activarse y desactivarse con un sólo clic. Cada juego de avatares viene con un set de avatares de personas predeterminado que puede cambiarse. Activar el juego, además de permitir que los estudiantes de la asignatura creen su propio avatar, incluye otras tantas funciones que se listan a continuación.

- Asignar familia de avatares

  Para que el juego no sea monótono, es posible cambiar la familia de personas que aparece predeterminada la primera vez que se activan los avatares. Las familias añadidas pueden ser animales, seres fantásticos o personajes de ciencia ficción.

- Permisos

  Una vez se crea un juego de avatares para la clase, es necesario definir los tres permisos que conceden los privilegios de modificar el pelo, los ojos o los complementos del avatar. Serán tres notas de texto en las que aparecerán las normas para que cada alumno pueda recibir el permiso. La activación de los permisos no es automática, sino que es el profesor, quien manualmente activa o desactiva los permisos de cada usuario registrado en la clase.

  Si el alumno cumple los requisitos para que se le conceda el permiso, el profesor irá a la lista de alumnos de clase para poder activar el permiso que corresponda.

- Lista de avatares

  Esta es una de las partes fundamentales del juego, tener una vista de la lista de clase con avatares, para que así todos los alumnos puedan ver cómo quedan los avatares de todos sus compañeros, y de alguna manera motivar al mayor número de personas posibles a que consigan los privilegios para poder ajustar el avatar a su gusto.

- Material de diseño

  Algo que es imprescindible para darle juego a los avatares, son las imágenes con las que se diseñan. Como administrador, el profesor debe suplir una galería de material a la que puede acceder desde el menú de inicio, colocando las imágenes que le parezcan adecuadas para que se construyan los avatares. Y de la misma manera que puede añadir material, también podrá eliminarlo o bloquearlo.

4.2.2. Rol del alumno

El juego visto desde el rol de alumno es diferente. De manera análoga al profesor, tiene un apartado en el que aparece su información personal, que podrá modificar a sus anchas cuando lo necesite. Por lo demás, las funciones que están al alcance del estudiantes son las siguientes:

- Permisos

  Puesto que personalizar el avatar tiene unos requisitos, el alumno podrá ver por escrito cuáles son para apresurarse a conseguirlos. Por cada requisito cumplido, se desbloquea un apartado para personalizar el avatar. Además habrá un marcador a la derecha de cada permiso que le muestre si ha conseguido el permiso, cuando aparezca en color verde el administrador le habrá activado el permiso. Los permisos son cambiar el pelo del avatar, los ojos, los complementos y poder ver la lista de clase con el avatar del resto de compañeros de clase.

- Diseñar el avatar

  Conforme el estudiante haya conseguido los privilegios que permiten cambiar elementos del avatar, los apartados de los elementos que decoran el avatar se desbloquearán para poder acceder al material que contienen.

  Aquí se recogen 3 de los 4 privilegios, concretamente poder cambiar de peinado, elegir lols ojos del personaje y añadir un complemento.

- Ver la lista de avatares

  Este es el cuarto privilegio que puede tener el usuario siendo alumno. Un apartado da acceso a ver la lista de avatares que tiene toda la clase. Cuando se haya conseguido este privilegio, el apartado aparecerá disponible para cceder, de lo contrario estará bloqueado.

- Lista de asignaturas

  En el menú habrá un acceso directo a todas las asignaturas que estén registradas en la aplicación, que el alumno esté cursando. Accediendo a cada una de ellas podrá ver la información como el nombre la asignatura, las notas y el nombre del docente que imparte la clase, y también podrá crear un avatar personalizado siempre que el juego de avatares de la asignatura haya sido creado previamente por el profesor.

- Miniatura del avatar

  En el apartado de información de cada asignatura, si está activo el juego de avatares se mostrará una miniatura del avatar, que en un principio aparecerá como un busto desnudo.

4.3. Diseño de las pantallas

Las pantallas de profesor y alumno serán completamente diferentes, ambas tendrán un menú principal con diferentes secciones que acceden a otras vistas. La pantalla del login sí

que será común. A partir de esta, ni el menú, ni el resto de pantallas tienen prácticamente nada en común. Más adelante se muestra cómo se va de una pantalla a otra en los menús asociados a cada rol.

Podemos ver una aproximación de este diseño a continuación creado con Mockplus, una aplicación orientada al diseño de prototipos de interfaces. Todos los diseños, comparados con los implementados tendrán ligeras variaciones, pero lo que no cambiará son los elementos que se utilizan.

Partimos de un inico de sesión, con dos entradas de texto para introducir el usuario y la contraseña, acompañadas de algún identificativo para que el usuario sepa dónde poner cada cosa.

Figura 4.1: Pantalla de inicio de sesión

Los menús de inicio son los de la figura 4.10, para el profesor el de la izquierda y para el alumno el de la derecha. Cada uno de estos menús está adaptado a las necesidades que le corresponden al tipo de usuario que accede a la aplicación. Las secciones que aparecen en cada tipo de usuario serán reproducidas de la misma manera a la hora de implementarse, como mucho aparecerá algún apartado adicional.

Para el profesor, los apartados de clases y las galerías serán labase para darle dinámica a los avatares. Activar el juego es una opción que permanece dentro de las funciones de clase y el mterial con el que se desarrolla el juego se administra desde las galerías.

Pese a que los apartados que aparecen en cada menú, como hemos dicho, no tienen nada que ver, sí que es cierto que hay una pantalla que es la misma para ambos, que es la de “Mi perfil”. Mediante un clic accedemos a ver los datos con los que está registrado el usuario en la aplicación, independientemente de su rol. Esto sirve para modificar en cualquier momento los datos si el usuario lo ve necesario, que es la funcionalidad para la que está diseñada.

A partir de aquí tenemos que diferenciar en las pantallas del profesor y las del alumno. Al no tener más apartados en común, las opciones restantes del menú nos llevarán a vistas diferentes.

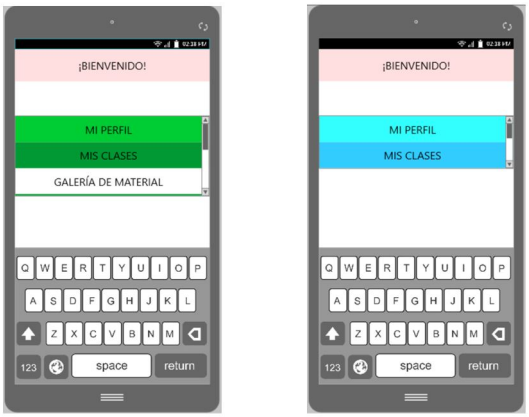


Figura 4.2: Menú de inicio para profesor y alumno

Figura 4.3: Apartado Mi Perfil

4.3.1. Pantallas para el rol de profesor

La pantalla de inicio del profesor muestra 3 secciones, basadas en todo lo descrito en el previo apartado 4.2.1..

Desde “Mis clases” se accede a una pantalla que muestra todas las clases de las que el profesor es administrador. En cada clase puede haber diferentes alumnos y un juego de avatares con sus respectivos permisos, y teniendo esto en cuenta cada clase debe tener su propio panel para poder visualizar bien los usuarios y elementos que le corresponden. La idea es un nuevo menú para cada clase, con dos opciones que cubran los requisitos de este módulo: “Ver listado de alumnos” y “Juego de avatares”.

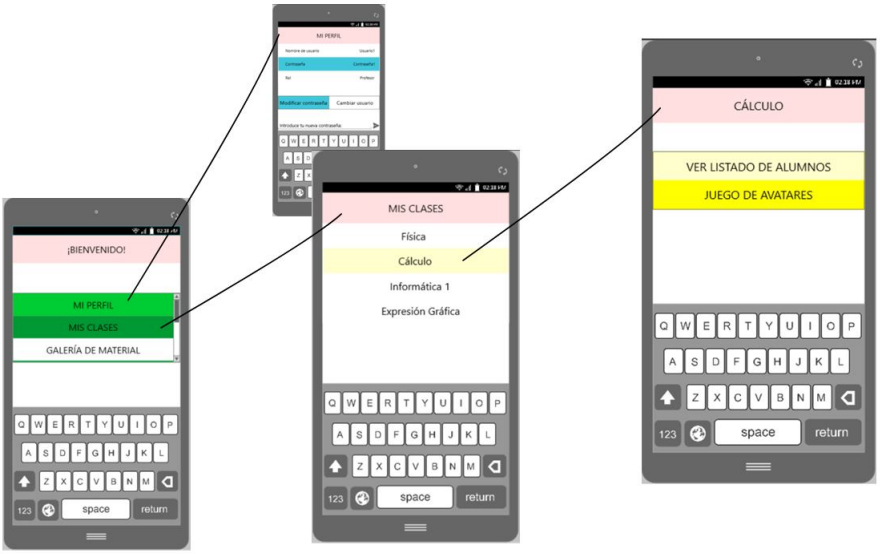


Figura 4.4: Despliegue de pantallas rol profesor

El juego de avatares se puede activar o desactivar, que es una de las principales acciones de las que dispone la pantalla de esta sección. Los permisos que venimos mencionando desde algún apartado anterior se gestionan desde esta vista, en “Permisos”, aquí se puede seleccionar un archivo de texto para cada uno de los cuatro e imponer cuáles van a ser las condiciones para conseguirlos. Y por último, la vista del listado de alumnos con sus avatares también tiene que estar ligada a esta pantalla. Desde “Lista de avatares” se abre una nueva pantalla que exhibe el avatar y el nombre del usuario al que le corresponde, mostrando los de toda la lista de la clase en la que esté activo el juego.

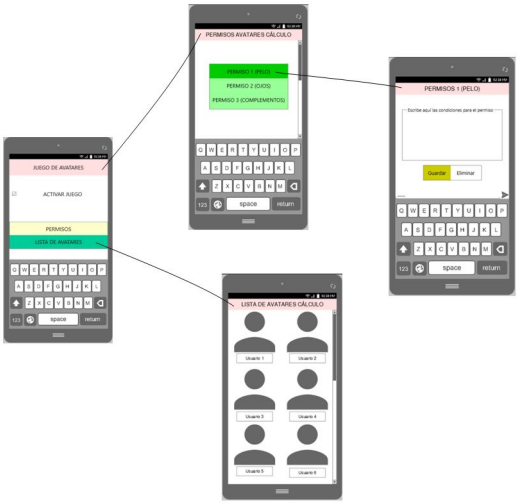


Figura 4.5: Pantallas juego de avatares desde el rol de profesor

Para el listado de alumnos, puesto que el profesor tiene la autoridad de administrar, debe haber unos botones que le permitan hacer de administrador de la lista, como añadir alumnos, eliminarlos, o bien conceder y retirar permisos a un alumno específico. Todo esto en paralelo a poder ver la lista completa con todos los datos registrados en ésta. A continuación puede visualizarse qué apariencia podría tener la lista de alumnos.

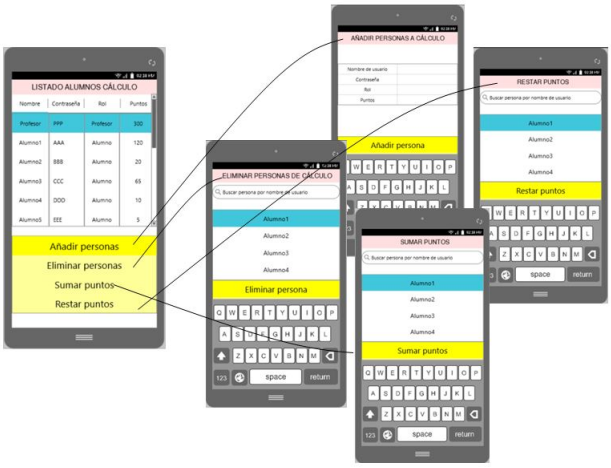


Figura 4.6: Despliegue de pantallas para “Lista de alumnos”

Ahora bien, la pantalla anterior es la que se abre cuando el módulo de avatares no está activo. A partir del momento en el que el juego esté activo, además de ver los datos y puntos de cada alumno de la lista, aparecerán 4 casillas desactivadas por defecto, que representarán los 4 permisos que van relacionados con el pelo, los ojos, los complementos del avatar y ver la lista completa de avatares de la asignatura.

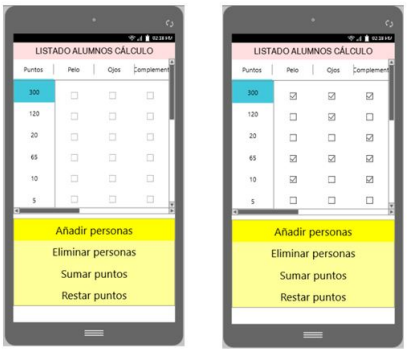


Figura 4.7: Lista de alumnos con permisos

4.3.2. Pantallas para el rol de alumno

Al entrar en la aplicación con el rol de alumno, los apartados que aparecen son “Mi perfil” y “Mis asignaturas”. La pantalla de “Mis asignaturas” despliega las asignaturas en las que está registrado el alumno, cuyo docente utiliza la aplicación. En cada clase, se abre una nueva pantalla que contiene los datos del docente, la puntuación del alumno, una vista del avatar y un apartado titulado “Mi avatar” si es que el juego está en marcha. Además una miniatura del avatar aparecerá e irá variando de aspecto según éste se personalice. No es es aquí donde se personaliza el avatar, pero como cada asignatura tiene su juego de avatares, es una manera de saber qué apariencia tiene cada avatar en cada clase.

Desde la sección de “Juego de avatares” se abre una nueva vista que muestra un sección para personalizarel avatar, que además incluye un botón que da acceso a ver los permisos.

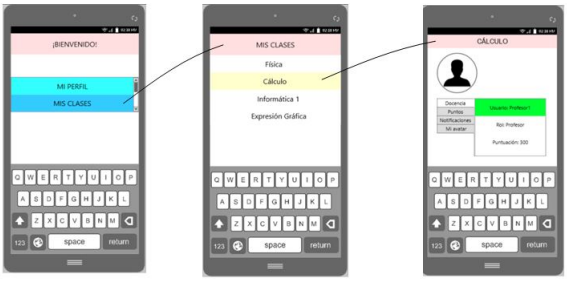


Figura 4.8: Despliegue de pantallas para Mis clases

Figura 4.9: Apartado ”Juego de avatares”para el rol del alumno

La sección permisos simplemente accede a una pantalla en las que aparecen las condiciones para conseguir los permisos para cambiar el pelo, los ojos, complementos del avatar o ver la lista completa de avatares, con un indicador verde que indica que está activo en rojo que significa que no está activado.

Figura 4.10: Vista de ”Permisos¨ desde el rol de alumno

Para personalizar el avatar, el panel es muy intuitivo. Se muestra en una imagen central el busto desnudo del avatar y en un menú colocado en la parte más baja de la pantalla 3 botones sin texto, con iconos que representan los ojos, el pelo y los complementos del avatar. Cada botón despliega una galería de imágenes que contien peinados, ojos y complementos respectivamente. El usuario sólo podrá acceder a las galerías que este autorizado.

Figura 4.11: Menú para personalizar el avatar

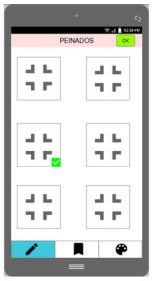


Figura 4.12: Elementos para personalizar el avatar

Al seleccionar una imagen de la galería de cualquiera de los apartados, aparecerá un tick verde sobre la imagen que se haya seleccionado, y al pulsar en el OK que aparece arriba a la derecha, se retrocede a la vista principal que mostrará el avatar con el pelo, ojos o complemento que se haya elegido.

Estas son todas las pantallas diseñadas para implementar en la aplicación, cuyo aspecto y funciones podrán variar en función de las necesidades que vayan surgiendo.

CAPÍTULO 5. ENTIDADES DE LA BASE DE DATOS

Mediante una aplicación llamada Loopback, es posible crear una API (Application Programming Interface) que además de contener los datos de todos los usuarios, utiliza servicios que recogen una serie de protocolos y funciones para intercambiar datos entre aplicaciones. Dentro de esta aplicación hay varios sistemas de gestión de bases de datos de código abierto, de los que he escogido “In Memory”, uno de los sistemas de código abierto que almacena los datos en la memoria local.

5.1. Modelos

Los modelos representan información con la que va a trabajar el controlador, están definidos con atributos y se especifica qué tipo de información tiene cada atributo. Si tenemos un archivo con las características del modelo, el controlador y la vista HTML pueden conocer también qué propiedades tiene y cómo está estructurado.

Los modelos que han sido necesarios para poder alcanzar los objetivos propuestos en la aplicación son cinco: persona, matrícula, clase, imagen, contenedor y permiso. A continuación se dan todo tipo de detalles a cerca de los atributos de cada uno de ellos.

- Persona

  La aplicación pide como datos de entrada un usuario y una contraseña, datos que deberían ser únicos para cada persona que utilice la aplicación y un modelo persona es lo ideal para poder almacenar estos datos.

  En este modelo se disponen de tres atributos,un nombre de usuario, una contraseña y un atributo que represente el rol de la persona. La función del rol es que la aplicación pueda detectar si la persona que ha iniciado sesión es profesor o alumno, que esos serán los dos únicos valores que pueda tomar este atributo.

- Clase

  Todas las personas con sus respectivos datos estarán distribuidas en clases. Así que un modelo llamado clase va a contener los datos de todas las asignaturas que imparte el profesor. En caso de que haya que administrar más de una clase al ir a ver el listado de alumnos se pedirá como parámetro de entrada el identificador de la clase, para poder mostrar los datos de todos los miembros de la clase que se corresponda con el identificador.

  La clase tendrá como atributos un identificador, para que el profesor pueda saber de qué clase se trata, un administrador, una atributo llamado avatares, para indicar si el juego está activado o no, una familia de avatares, acompañada de un atributo al que llamaremos busto, para guardar el fichero del busto de la familia de avatares, y por último cuatro atributos para guardar los cuatro privilegios que se conceden en el juego.

- Matrícula

  Una vez presentados los modelos de persona y clase, hace falta un tercer modelo para relacionar a cada persona con su clase. Este modelo es la matrícula, que tiene como atributos un identificador de alumno y un identificador de asignatura, así podemos ubicar a la persona en una clase concreta, y no sólo eso, sino que una persona puede estar matriculada de varias asignaturas siempre y cuando tenga una matrícula para cada una de ellas. Además de estos dos atributos, en matrícula aparecen 10 atributos más. Cuatro de ellos son booleanos que sirven para saber si la persona tiene los permisos del juego de avatares que son 4: cambiarse el peinado, escoger los ojos, añadir un complemento y poder ver la lista de avatares de toda la clase. Otros cuatro atributos van ligados a estos últimos, porque en caso de que el estudiante tenga permiso para cambiar el peinado del avatar implica que puede escoger un archivo de imágen, y que éste tendrá que guardarse en alguna parte. Así que atributos para fichero de peinado, fichero de ojos, fichero de complemento y fichero de boca, que puede cambiarse siempre.

  Por último, los dos atributos restantes que no son relevantes en el juego de avatares pero tienen su función, son la nota y el identificador de la matrícula, que es un número único para cada matrícula.

- Contenedor Trabajar con los contenedores de almacenamiento implica hacer la declaración de algún tipo de modelo para trabajar de una forma más cómoda. Los contenedores tienen un nombre y archivos con un nombre concreto dentro, por tanto como vamos a trabajar con contenedores por separado, el único atributo que vamos a asignarle al contenedor es un nombre de tipo string. El modelo contenedor nos será útil para asignar una familia de avatares, la familia tendrá el mismo nombre que el del contenedor. Conocer el nombre del contenedor es suficiente para tener acceso a todas los archivos que contenga.

- Imágen El modelo para hacer referencia a las imagenes se va a llamar Imagen para no confundirlo con el del contenedor que solo hace referencia al lugar donde se almacenan estas. Como atributos tiene un nombre, haciendo referencia al nombre del archivo y un segundo atributo que es la dirección, para almacenar la cadena de caracteres que servirá para ver el archivo.

  Dentro de cada contenedor habrá archivos tipo 'png' o imagenes que pertenezcan a la familia de avatares que representa cada contenedor. Trabajar con estas imagenes para mostrarlas en HTML es un poco complejo, hay una función que será la encargada de convertir los datos del fichero png en una URL. La URL es una secuencia de caracteres que nos lleva a un sitio concreto de internet y permite recuperar un tipo de información determinada. Una URL lo que hace es redirigirnos a mostrar la imágen. La función que genera la URL debe saber de qué archivo tiene que generar la dirección URL, así que lo mejor para trabajar con los datos de la imágen es crear un modelo nuevo sobre el que se sustenta el archivo de foto.

- Permiso

  Además de un contenedor para las imagenes que componen el avatar, hay un contenedor que almacena los archivos de texto que se utilizan para establecer los requisitos para conseguir los permisos. Los permisos permiten cambiar diferentes elementos del avatar y cada uno de ellos se concede si el usuario cumple unos

requisitos que el profesor establece. Pues bien, estos requisitos van recogidos en un archivo de texto. Para eso declaramos un nuevo modelo de almacenamiento que llamaremos permiso y que no tendrá ningún atributo

en la siguiente imágen se muestra un diagrama de estos con sus atributos y cómo están relacionados unos con otros.

5.2. Relaciones entre modelos

Una vez ligados todos los elementos del juego de avatares con los diferentes modelos, es momento de introducir las relaciones. Las relaciones establecen la dependencia que existe entre dos modelos.

- Relaciones 1:1

  En este tipo de relaciones solamente se relaciona un elemento de un modelo con otro elemento de otro modelo. Por ejemplo, una matrícula puede pertenecer solo a una persona.

- Relaciones 1:N

  Un elemento de un modelo A se relaciona con ninguno o con varios elementos de otro modelo B, pero los elementos del modelo B solo pueden relacionarse con un modelo de A. Este es el caso de la matrícula y la persona, una persona puede tener varias matrículas, mientras que una matrícula puede pertenecer a una sola persona. En loopback este tipo de relaciones se llaman “belongsTo”.

- Relaciones N:M

  Varios elementos de un modelo pueden estar relacionados con un solo modelo, por ejemplo, tenemos el modelo “persona” y el modelo “clase”. Una clase puede estar formada por varias personas y a su vez, una persona puede estar en varias clases diferentes. Este tipo de relación en la aplicación se llama Hasmany.

Traducido al lenguaje que utiliza loopback, encontramos las siguientes relaciones:

1. clase-HasMany-persona
2. persona-HasMany-clase
3. persona-belongsTo-matrícula
4. matrícula-belongsTo-clase

Que dos modelos estén relacionados supone que se conectan a través de una clave, que puede ser propia o de un modelo ajeno. Las claves que aparecen en las relaciones detalladas anteriormente son ajenas o también llamadas “foreignKey”. El claro ejemplo lo aporta el modelo “matrícula”, donde sus atributos “idAlumno” e “idAsignatura” son ambos “foreignKeys”.

5.3. Ejemplos de acceso a la base de datos

En este apartado mostramos efectos prácticos que nos permiten acceder a la base de datos mediante los métodos HTTP explicados en el apartado 5.1..

Desde un archivo utilizado para servicios, que contiene funciones que hacen operaciones y peticiones a la base de datos, creamos una función que muestre los datos de una persona. Para evitar consultar constantemente la base de datos, trabajaremos con “Observables” que reflejan automáticamente los cambios realizados en la misma.

Obtener la información de una persona requiere del identificador que tiene el modelo “persona”. Todas las instancias de este modelo se encuentran en una dirección http concreta, así para obtener los datos de una sola de ellas es necesario tener el parámetro único de cada persona (identificador), añadiendo a la dirección de la http donde se encuentran todas las personas una / seguida del identificador se consiguen los atributos de la persona deseada obviando la información del resto.

Estos datos van a utilizarse para autentificar al individuo concreto en la aplicación. Recapitulando, en el apartado 5.1 los atributos del modelo “persona” son, nombre, que además sirve de identificador, pass haciendo referencia a la contraseña, y rol que especifica si el usuario es estudiante o docente.

En el componente para entrar en la aplicación, se piden el nombre de usuario y la contraseña del individuo. Al hacer click en el botón “iniciar sesión” la función “Autentificar” llama al servicios que accede a la base de datos utilizando la función “DamePersona” que toma como parámetro el nombre de usuario introducido, esta función se encarga solo de hacer una petición get a la base de datos y retorna un “Observable” de la persona. La función “autentificar” es la que recoge este observable y comprueba que la contraseña introducida coincide con la persona retornada, y si la información concuerda, dependiendo del rol del usuario, le dará acceso a una página o a otra dentro de la aplicación.

Otro caso en el que se consulta la base de datos mediante un servicio ocurre cuando el alumno accede a l apartado de “mis asignaturas”. Aquí se llama a la función “DameClases” del archivo de servicios, que lanza una petición get a la dirección donde se encuentran todas las clases, y retorna un observable de todas ellas. Un último escenario en el que el archivo de servicios media entre la interfaz de la aplicación y el almacenamiento de datos se da a la hora de crear una clase nueva, la función “CreaClase” hace una operación post con toda la información necesaria para cubrir los atributos requeridos de la clase que se recogen de un formulario cumplimentado por el profesor.

CAPÍTULO 6. IMPLEMENTACIÓN

La implementación de la aplicación móvil se ha llevado a cabo con angular e ionic. La aplicación está formada por componentes, que son pequeños módulos del software compuestos por tres partes, la vista HTML, el controlador y el estilo además de utilizar servicios para conectar con la base de datos.

6.1. Componentes

En este capítulo se detallan los componentes más importantes y se explica qué implicación tienen en el desarrollo del software y de la aplicación. Dichos componentes están orientados a la parte del juego de avatares, el resto de vistas HTML, estilos y controladores no juegan el mismo papel en este módulo, son detalles adicionales que sirven para complementar el módulo. A modo introducción vamos a enumerar los componentes que forman esta lista de elementos que dan juego a los avatares:

- avatares

  En la página avatares tenemos la opción de activar y desactivar el juego si el inicio de sesión en la aplicación ha sido hecho con usuario que tenga el rol de profesor. Activar el juego supone seleccionar una familia de avatares, que por defecto es de personas. Las otra familia disponible es de minions.

  Cuando el juego está en funcionamiento, aparecen dos botones: “Permisos” y “Lista de avatares”. Estos nos llevarán a los componentes de listaavatares y permisos, respectivamente. Estas dos secciones se explican más adelante.

```
export class AvataresPage implements OnInit {

  estado: boolean;
  idClase: string;
  clase: Clase;
  familias: Familia[] = new Array();
  nombrefamilias = new Array();
  private APIFotos = 'http://localhost:3000/api/imagenes';
  familiaescogida: string;
  familiaBDD: string;

  constructor(private http: HttpClient,
              private router: Router,
              private datosService: DatosService,
              private dbService: DbServiceService,
              private http2: Http) { }

  ngOnInit() {
    this.idClase = this.datosService.DameIDClase();
    this.dbService.DameClase(this.idClase).subscribe( clase => {this.clase = clase;
                                                                console.log('Datos de clase: ' + clase);
                                                                this.estado = clase.avatares;
                                                                this.familiaBDD = clase.familia;
                                                                this.myChange();
                                                                this.Mostrar();
                                                                });

    this.http.get<any[]>(this.APIFotos + '/Bustos/files')
    .subscribe( fotoscontainer => {console.log (fotoscontainer);
                                   var i;
                                   for ( i = 0; i < fotoscontainer.length; i ++) {
                                    this.familias[i] = fotoscontainer[i].name;
                                   }
                                   console.log('Aquí están las familias: ' + this.nombrefamilias);
                                   this.DivideFamilias(this.familias);
                                  });
```

Figura 6.1: Código del controlador

```
<ion-header>
  <ion-toolbar>
    <ion-title>Juego de avatares</ion-title>
    <ion-back-button (click)="IrAClases();"></ion-back-button>
  </ion-toolbar>
</ion-header>

<ion-content>
    <div class="estado">

    <ion-item>
      <ion-label>Juego avatares {{idClase}}</ion-label>
      <ion-toggle slot="end" id="estado" color="success" checked [(ngModel)]="estado" (ionChange)="CambiaEstado(); Mostrar();" ></ion-toggle>
    </ion-item>

    <div id="activado">

      <!-- <p>
        Juego activado
      </p> -->

      <ion-list>
        <ion-list-header>Familia de avatares</ion-list-header>
        <ion-item>
          <ion-label>Escoge una familia</ion-label>

          <ion-select [(ngModel)]="familiaescogida">


            <a *ngFor="let item of familias">

              <ion-select-option value="{{item}}">{{item}}</ion-select-option>


            </a>

          </ion-select>
```

Figura 6.2: Código vista HTML

Para ver de qué manera trabaja un componente relacionando la vista con el controlador, en el ejemplo de la figura 6.1 se obseva un parámetro que se llama idClase, de tipo string. Este parámetro reaparece en la vista HTML de la figura 6.2 entre una doble llave. Esto significa que el valor que toma este parámetro en el controlador, se mostrará en la vista HTML al usuario.

A continuación se enumeran más componentes que pertenecen a la aplicación.

- galeria

  La galería es una página exclusiva para el profesor, donde carga los archivos de imágen que constituyen diferentes elementos del avatar. Encontramos diversas secciones, una para cada elemento del avatar.

  Cada sección tiene dos subapartados, uno para añadir material y otro para visualizar el material que además da la opción de eliminarlo si se desea.

- complementos, ojos y pelos

  Las funcionalidades y vistas de estas tres páginas pueden explicarse en conjunto, ya que todas ellas tienen la misma función. Están disponibles unicamente para el usuario con rol de alumno. Una vez se ha explicado la página de la galería es mucho más sencillo comentar su funcionamiento.

  Tienen la misma funcionalidad que la galería del profesor, mostrar los peinados, ojos, complementos y ropa, pero en páginas por separado.

- home

  Home es el primer componente que venía por defecto con la aplicación, que se ha aprovechado para personalizar el avatar. Desde un usuario con rol de alumno se puede acceder, siempre y cuando el juego de avatares esté activo.

Al entrar aparece el avatar en el centro de la pantalla se muestran los ficheros del avatar que tenga el alumno, en caso de no tener ninguno se verá el busto del avatar vacío o con los elementos que tenga. Desde esta sección podrá entrar a cambiar los elementos del avatar y tendrá acceso a la lista de permisos para que el juego sea completo.

- listaavatares

  La lista de avatares carga los avatares de todos los alumnos que estén matriculados en una asignatura con juego de avatares.

  Teniendo un avatar personalizado para cada alumno, la lista de alumnos puede cobrar un poco de vida.

- permisos

  En “Permisos” el profesor tiene las funcionalidades de subir archivos de texto a la base de datos y de cargarlos en la aplicación para que se muestre por pantalla el texto que contienen. Los archivos de texto para cada permiso tienen una sección propia.

- vistapermisos

  A la página vistapermisos se accede desde “home”. Aquí aparecen clasificados los requisitos que el profesor ha seleccionado para conceder los permisos. Es una sección a la que acceden los usuarios que son alumnos y no tiene ningún elemento para interactuar. Se muestra qué se exige para conseguir cada permiso y además, hay un indicador para que el alumno sepa si ya ha cumplido el requisito o no.

6.2. Servicios

Un servicio es un método que se utiliza para comunicar dos interfaces, mediante un conjunto de protocolos establecidos. En este proyecto los servicios que se utilizan son dos, ”datos.servicez ”db-service.service”. Uno de ellos, db-service, tiene alrededor de 400 líneas de código y comunica esencialmente con la base de datos, mientras que el servicio datos tiene unas 140 líneas de código y sirve para pasar información de un componente a otro dentro de la aplicación.

En la imágen que aparece bajo este párrafo, se expone el código de una función del archivo de servicios ”db-service”, que elimina una matrícula de la base de datos mediante una operación ”delete”de protocolo http. toma como parámetro de entrada la matrícula que va a eliminar para poder localizar la matrícula en la base de datos por su identificador.

```
EliminarMatricula(matricula: Matricula): Observable<any> {
  console.log('Voy a eliminar la matrícula de: ' + matricula.idAlumno);
  console.log('El id que elimino es: ' + matricula.id);
  return this.http.delete<any>('http://localhost:3000/api/matriculas/' + matricula.id);
}
```

Figura 6.3: Caption

Para ver más operaciones de este tipo, que se utilizan para acceder a la base de datos, en el anexo F aparece el código de el archivo “db-service” para tener una buena referencia.

6.3. Juego de Avatares

El componente más importante de todos los que forman la aplicación es el que activa el juego de avatares, ya que de eso se trata este módulo. Se ha descrito brevemente en el apartado 6 el componente “avatares”. Teniendo la idea de las funcionalidades y aspecto que tiene, pasamos a detallar en la siguiente sección el mecanismo lógico que lleva detrás.

6.3.1. Crear un juego de avatares

Cuando se activa el juego aparece un cuadro para seleccionar qué familia de avatares es la que se quiere añadir, si no se selecciona ninguna, por defecto se crea una familia de avatares de personas.

Un botón toggle, que es un elemento HTML con apariencia de un botón circular que se desliza por una ranura a derecha o a izquierda cada vez que se clica sobre él, es el que da la órden de activar el juego. En la figuras 6.5 y 6.4 se puede ver cómo es este tipo de botón. Una función del controlador de la página, se activa cada vez que el toggle cambia de estado y toma como parámetro de entrada el estado del toggle. Cada vez que el toggle cambia de color y de estado se llama a una función que comprueba el estado del juego de avatares en el modelo clase de la base de datos, la tarea de esta función es cambiar el estado del juego.

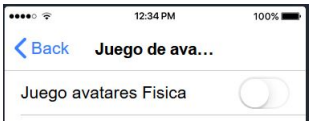


Figura 6.4: Botón toggle desactivado

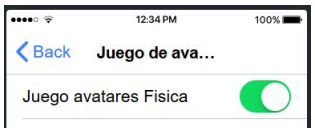


Figura 6.5: Botón toggle activado

Hasta aquí todo lo explicado es lo que se ve desde el rol de profesor, pero para el alumno la cosa cambia. El alumno dentro de cada asignatura, tiene un apartado que se llama Juego de avatares, pero que este apartado exista no quiere decir que pueda entrar siempre que le apetezca.

Al entrar a las opciones de una asignatura, se carga la información que tiene la clase en la base de datos y si en el atributo avatares hay un valor “false”, una función se encarga de desactivar el boton. Desactivar el botón no significa eliminarlo, pero los botones están pensados para hacer click sobre ellos y que algo ocurra, pero si el botón está desactivado por más que se haga click no pasa absolutamente nada. Es una manera de que el alumno se dé cuenta de si el juego de avatares está activado o no, a simple vista se nota cómo las letras pierden el color y toman un color grisáceo. En las imagenes 6.6 y 6.6 tenemos el claro ejemplo para ver la diferencia.

El paso posterior a activar el juego es elegir una familia de avatares. Por defecto la familia de avatares es de personas, pero si se quiere establecer otra familia, existe la opción de cambiarla. Se explica con más detalle en el apartado 6.3.2.

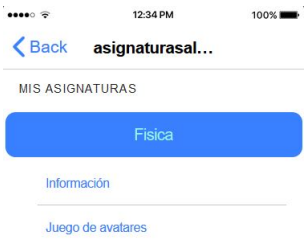


Figura 6.6: Juego de avatares activado

Figura 6.7: Juego de avatares desactivado

6.3.2. Crear una familia de avatares

Una de las funcionalidades más originales de este proyecto es que no hay un sólo tipo de avatares. Pueden recopilarse diferentes familias, para no tener que utilizar siempre la misma y convertir el juego en algo repetitivo.

Por defecto, cuando se activa el juego de avatares la familia que se asigna es del tipo “persona”, y así se muestra por pantalla en la aplicación, sin embargo aparece un desplegable acompañando al texto “Escoge una familia” que extiende una lista de opciones para elegir. Una vez seleccionada la familia, el botón que aparece bajo la lista “guardar” utiliza el servicio que accede la base de datos para cambiar la familia de la base de datos y también el busto que representa a la familia.

6.3.3. Normas de nomenclatura

Una familia tiene un sólo busto, sobre el que pueden colocarse cierta variedad de ojos, peinados, bocas y complementos. Cambiar de familia, supone cambiar el busto del avatar y para no complicar el asunto, añadir una familia también supone añadir un busto nuevo de avatar. Pero, ¿cómo detecta la aplicación qué archivos pertenecen a qué familia?

Esta es la pregunta que se ha intentado resolver de la manera más sencilla posible. Las librerías de la mayoría de lenguajes de programación incorporan una función llamada “split”, que sirve para trocear variables de tipo “string”. Esta función que se aplica a un string toma como parámetros de entrada un separador, que es el elemento a partir del cual divide la cadena de texto, y devuelve un vector con los diferentes caracteres que ha encontrado a un lado y al otro del elemento separador.

Esto conlleva establecer unas normas de nomenclatura para todos los archivos que pertenecen a una familia. El nombre de la familia va a ser el que encabece el nombre del archivo, seguido de una barra baja a partir de la que se pondrá el nombre del elemento. Un ejemplo para nombrar al busto sería la nomenclatura “persona_busto”.

El busto es un elemento único para cada familia, pero no el resto de elementos, así que en caso de tener más de un peinado para la familia persona, la nomenclatura sería la misma, añadiendo un número al final, algo así como “persona_peinado1”.

6.3.4. Galería de material

La galería de material contiene todos los archivos con extensión png que son las piezas para construir el avatar. Tiene un apartado para cada elemento de los avatares, que es un contenedor diferente (peinados, ojos, bocas...). Al hacer clic en un botón, una función toma como parámetro de entrada el nombre del contenedor al que pertenece el botón. Anteriormente en la sección 5.1. se introducía el modelo Contenedor, que tenía un atributo que hacía referencia al nombre del contenedor. Pues bien, sólo se necesita este nombre del para que la función que carga los archivos muestre la vista de las imagenes.

En esa misma sección donde se introduce el modelo contenedor, se introduce el modelo imágen, que tenia los atributos nombre y dirección. Para la galería sólo es necesario que se pueda ver el contenido de la imágen, así que la función que carga la vista de las fotos, se encarga de convertir todos los archivos del contenedor y obtener una URL de cada uno para poder mostrar por pantalla el archivo, que viene a ser una foto en este caso.

El usuario con rol de profesor es quien puede entrar a esta página para visualizar, añadir y eliminar el material. Para añadir material hay un botón que activa la función de carga de archivos del controlador cuyo parámetro de entrada es el nombre del contenedor al que tiene que ir a parar para dejar el archivo que se envía.

Las fotos aparecen una bajo la otra, en el lado izquierdo de la pantalla, dejando a la derecha un recuadro que sirve para seleccionar la imágen. De cada archivo seleccionado, el controlador guarda el nombre de éste en un vector que puede contener los nombres de todos los archivos que se seleccionen. Este vector de nombres se utiliza en caso de que se quieran eliminar, haciendo click en otro botón que hará una petición a la base de datos para hacer una operación DELETE.

6.3.5. Personalizar el avatar

Solo los usuarios con rol de alumno pueden acceder a la página home, que aporta las funcionalidades para adornar el avatar. En la parte central de la pantalla aparecerá un busto sin ojos, pelo ni boca, que podrá personalizarse más o menos dependiendo de los privilegios de los que se habla en el apartado 6.4., donde se detalla qué son y cómo se conceden. Para estos privilegios, en la parte superior de la pantalla, tal y como se puede ver en la foto C.14, aparece un botón que permite al usuario visualizar la página de estos privilegios, denominada permisos dentro de la aplicación. Por último en la parte inferior de la página hay botones que representan los elementos para personalizar el avatar.

Los botones morados abren galerías para poder escoger el peinado, los ojos, los complementos, la boca o la vestimenta del avatar. Dentro de estas secciones el mecanismo para escoger el elemento que se desea lo desempeñan dos funciones.

Una función sirve para saber el nombre del elemento que selecciona el usuario, guarda siempre el nombre del último elemento seleccionado por si no se decide a la primera. La otra función entra en acción cuado se clica en el boton azul que aparece en la parte superior derecha de la pantalla con un el texto “OK”, que se dedica a coger el nombre que ha guardado la función explicada anteriormente y lo almacena en la matrícula del alumno.

La matrícula tiene un identificador para la asignatura y otro para el alumno, para saber de quién es el avatar y en qué clase está. Estos no son los únicos datos que contiene

la matrícula, sino que hay un atributo para cada pieza que decora el avatar. En estos apartados hay nombres de ficheros que están en el contenedor de imágenes, los mismos nombres que la segunda función explicada en esta sección ha guardado.

Al volver a la página home los elementos escogidos se habrán colocado sobre el busto o cuerpo del avatar. Los elementos no se colocan por arte de magia, sino que una función se encarga de descargar los archivos de la base de datos y de descodificar la información, tal y cómo se hacía en la galería del profesor. De hecho el método que hace esto es el mismo que usaba la página galería. Es una manera de aprovechar los recursos de los servicios que hay en el código.

6.3.6. Lista de avatares

Las matrículas de todos los alumnos de una misma asignatura son recogidas en un vector. De aquí la información útil es el nombre del alumno al que pertenece la matrícula y ya que es una lista de avatares, qué aspecto tiene el avatar personalizado. Se muestran por pantalla los archivos de imágen de los elementos del avatar que tiene el alumno, colocandolos bajo su nombre de usuario para poder identificar a quién pertenece.

Un método se encarga se crear una sección para cada alumno, poniendo en la parte superior el nombre. Es tan simple como crear un bucle que recorra las matrículas de la asignatura y muestre el atributo nombre de la matrícula, dejando un espacio bajo cada nombre en el que otra función coloque el avatar.

Reutilizando la función que descarga archivos de imágen de la galería y muestra qué apariencia tienen, mostramos todas las imágenes cuyo nombre está alacenado en la matrícula del alumno, combinandolas para que aparezca el avatar al completo.

No es una página que aporte dinámica al juego, no permite hacer absolutamente ninguna acción, más bien es un elemento de decoración.

6.4. Privilegios

Los privilegios los establece el profesor a su gusto, ningún usuario con rol de alumno puede interactuar con ellos sino que solo puede verlos.

Hay tantos permisos como elementos para crear el avatar, uno para los peinados, otro para los ojos, para la boca, los complementos y un último permiso que permite que el alumno vea la lista completa de clase con los avatares de sus compañeros.

De manera análoga a la galería, antes de mostrar el material hay que cargarlo, por eso existe un apartado llamado permisos avatares en el que el profesor puede añadir y eliminar archivos de texto. Una vez la base de datos dispone de material, ya se pueden dar a conocer los permisos. Una función para cargar el archivo, muy similar a la que carga imágenes a la galería, hace una operación POST, enviando el fichero al contenedor donde residen los archivos de permisos. La dirección de los ficheros de texto es uno de los parámetros con los que trabaja la función, además de conocer el nombre del fichero.

Para mostrar a los alumnos los requisitos de cada permiso, el profesor debe dirigirse a la sección “Permisos dentro del juego de avatares de cada clase. Junto a cada permiso apa-

rece un desplegable que muestra los nombres de los archivos de texto que previamente se han introducido en la base de datos. Al clicar en el botón guardar de cada permiso, una función envía el nombre del archivo a la base de datos, concretamente al atributo del permiso que sea dentro del modelo de la clase.

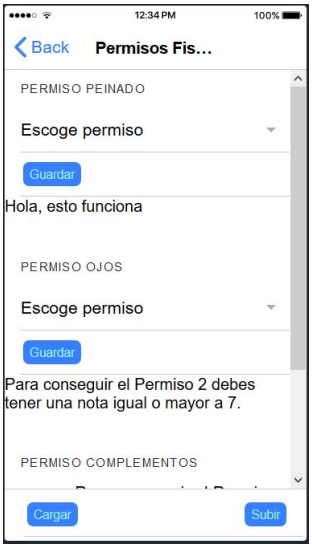


Figura 6.8: Caption

Otra función hace que automáticamente aparezca por pantalla el texto del archivo. Esta función actúa de la misma manera que el método que se usa para obtener la URL de la imágen, pero esta vez la información que tiene codificada el fichero no es una imágen sino texto, por tanto lo que se retorna es una cadena de texto lista para mostrar por pantalla.

Aquí finalizan las funcionalidades del profesor respecto a los permisos. El alumno, pese a que no puede interactuar de ninguna de las maneras con los permisos, sí que puede verlos y saber si se le han concedido mediante un toggle como el de la figura 6.5. Esta vez, el toggle aparece desactivado, para que no se pueda interactuar con éste.

Como el toggle está desactivado un método se encarga de cambiarle el color, verde si el permiso se ha concedido y blanco en caso contrario. El método accede a la matrícula del alumno que ha iniciado sesión, donde está la información de los permisos. Como se ha comentado en el capítulo sobre la base de datos (5), es un booleano que sólo puede tomar dos valores, true o false. Solo en caso de que el permiso tenga un valor true, el toggle aparece verde.

Que el valor del permiso sea true o false no es casualidad. Se profundiza sobre la concesión de privilegios en el siguiente apartado.

6.4.1. Concesión de privilegios

El profesor tiene el poder de cambiar este valor cuando crea conveniente. En la lista de alumnos aparecen casillas para marcar los permisos que tiene cada alumno, si la casilla de un permiso está marcada, el valor de este permiso en la base de datos va a ser un true, de lo contrario seguirá siendo false, que es el valor por defecto que aparece al matricular a un alumno.

La manera de conceder el privilegio o permiso es bastante intuitiva y no da lugar a confusión. Además estas casillas sólo aparecen en la lista de alumnos si el juego de avatares está activado, cosa que se encarga de revisar una función específica. Tan sencillo como comprobar que el atributo “avatares” de la clase tiene un valor “true”.

La clase “persona” tiene 4 atributos para cada uno de los persmisos, todos ellos booleanos. Cada uno de ellos es un booleano, y en cuanto el profesor haga clic sobre la casilla que corresponde, un método cambia el valor del booleano.

Con la intención de que todo lo descrito en este capítulo de implementación funciona correctamente, el siguiente capítulo detalla cómo se puede poner a prueba la aplicación para hacer un programa robusto.

CAPÍTULO 7. PRUEBAS Y EVALUACIÓN

Para comprobar que el código sea robusto y no haya ninguna incoherencia que pueda dar lugar a errores, se han escogido cinco pruebas para comprobar el correcto funcionamiento de la aplicación.

7.1. Batería de pruebas

- Contraseña errónea

  Para iniciar sesión en la aplicación, se requiere de un usuario y contraseña. En caso de que el usuario exista en la base de datos, pero la contraseña no sea la correcta, no se debería poder iniciar sesión. Tal y como se muestra en la figura XX, no sólo se impide que el usuario inicie sesión, sino que además aparece un mensaje que especifica el motivo que no le permite entrar en la aplicación.

- Añadir un usuario ya existente

  Tener dos usuarios con el mismo nombre puede llevar a confusiones, y no debe estar permitido. Así, cuando el profesor registra un nuevo alumno, se comprueba que no haya otro con el mismo nombre que haya sido registrado previamente. Si ya se había registrado a una persona con el mismo nombre, otra notificación aparece en la pantalla sugiriendo al profesor que utilice un nombre nuevo.

- Conceder y retirar permisos

  Los permisos se conceden desde un usuario con rol de profesor, sin embargo afectan a un usuario con rol de alumno. Así que una buena manera de comprobar que esta función sigue el curso correcto, es ver si el clic del profesor, ha repercutido en la matrícula del alumno. Para esto se colocan unos marcadores en el usuario del alumno, que aparecen en color verde para indicar que el permiso está activado.

- Eliminar una clase que no existe

  Hacer una petición “delete” a la base de datos relacionada con un elemento que no existe puede desencadenar un error fatal. Por esta razón, se alaerta al docente cuando intenta eliminar una clase inexistente.

Todas estas pruebas pueden darse por superadas, y aparecen en el apéndice D.

7.2. Evaluación de usuario

Siempre es conveniente conocer la opinión de disferentes personas que podrían hacer uso de la aplicación. La opinión de diferentes individuos a los que se les ha mostrado la aplicación con sus respectivas funcionalidades viene recogida en la encuesta anexada en la sección E de los apéndices.

Las preguntas realizadas están ligadas a las funcionalidades del profesor y del alumno, se pide opinión personal a cerca de qué encuentran a faltar para hacer del proyecto un

módulo más completo, se consultan qué tan intuitivo es nvegar por la aplicación y se pide escoger una funcionalidad favorita.

Los resultados generales reflejan que es un módulo muy completo en opinión de los encuestados, bastante intuitivo para cualquier usuario y que las funcionalidades favoritas por igual, son añadir diferentes familias de avatares y poder conceder privilegios para personalizar el avatar.

CAPÍTULO 8. CONCLUSIONES

Utilizar tecnologías completamente desconocidas, de las que no se ha trabajado a lo largo de los años de carrera ha supuesto un reto de aprendizaje autodidacta. Los vídeo-tutoriales de youtube y las webs de material de referencia en programación orientada al desarrollo de aplicaciones web y móvil han sido las guías del proyecto.

Además el uso de ionic como framework ha facilitado muchísimo la programación de la aplicación móvil, ya que incluye un montón de documentación sobre material en su página web. Por otra parte Git, la herramienta de control de versiones, ha sido otra de las piezas clave a lahora de realizar cambios de la manera más segura posible, pudiendo deshacerlos en cualquier momento.

8.1. Objetivos conseguidos

El juego de avatares para una herramienta de gamificación se ha intentado desarrollar de manera que cumpliese con el mayor número de objetivos propuestos posibles. De todos los objetivos propuestos, hay dos que no se han llegado a alcanzar. El nulo conocimiento de las tecnologías de desarrollo ha sido el motivo principal, puesto que tanto la instalación de todas lasherramientas, como el conocimiento de programación con éstas han llevado mucho más tiempo de lo esperado al inicio del proyecto. La lista de objetivos está definida en el apartado 2.4., de la que no se ha completado lo siguiente:

- Eliminar alumnos de la lista de clase

El tiempo dedicado a implementar el proyecto se ha centrado más en las funcionalidades de los avatares, por tanto detalles de este tipo se han dejado para última hora, sabiendo que la persona que tome el relevo de la aplicación será capaz de añadir el detalle.

8.2. Valoración personal

El proyecto ha sido todo un desafío, ya que en los diferentes cursos de la mención de aeronavegación tan sólo se imparten dos asignaturas dedicadas plenamente a la programación. Las librerías y el tipo de lenguaje que se han utilizado en este proyecto son orientados a objetos, que es la clase de programación que se utiliza en una de las asignaturas mencionadas previamente, lo que proporciona una buena base. Sin embargo, el tipo de tecnología que se utiliza para el desarrollo nada tiene que ver.

Uno de los mayores obstáculos para mí ha sido la instalación de todas y cada una de las herramientas para poder desarrollar el proyecto. Muchas veces, la versión instalada no era la correcta y daba errores muy poco específicos que no daban a entender que las vesiones entre dos o más herramientas no fuesen compatibles. Otra de las dificultades que se me han presentado es el uso de la versión 4 de Angular e Ionic, que no se habían utilizado en proyectos previos y eso supone que haya variaciones en algunas de las librerías, por tanto lo que en la versión 3 funcionaba perfectamente en la versión 4 probablemente no lo hiciese.

Cada error que daba el código, al principio era todo un mundo para corregirlo. Si que es verdad, que con la ayuda del resto de alumnos implicados en proyectos de Classpip y con los tutoriales de Miguel, poco a poco fui aprendiendo a evitar tantos fallos en el sistema.

El proceso de aprendizaje, pese a que ha sido costoso y algo largo, me ha aportado conocimientos suficientes como para que el resultado sea sacticsfactorio.

8.3. Líneas abiertas

El módulo de avatares añade una característica visual y muy atractiva a Classpip, sobretodo de cara al público más jóven.

Esta primera versión del proyecto, deja muchas puertas abiertas a la comunicación con otros módulos, como por ejemplo conceder privilegios especiales a los ganadores de un torneo perteneciente al módulo de competiciones, o desbloquear un permiso al llegar a un número de puntos determinado.

Quedan pendiente muchas posibles mejoras como funcionalidades nuevas o incluso alguna guía para usuarios recién iniciados, ya que puede ser un poco confuso el funcionamiento del proyecto. Todas las secciones pueden mejorar sobretodo en cuanto a diseño, que es la parte en la que menos se ha profundizado. La aplicación se ha hecho lo más sencilla posible para poder cumplir con los objetivos.

Para terminar, es imprescindible incluir más familias en el módulo y material exclusivo para determinadas épocas. Todo esto hará que el juego no sea repetitivo, mantenga motivados a los usuarios y satisfaga al cliente.

BIBLIOGRAFÍA

[1] Avatar [online]: *https://blog.ensalza.com/diccionario/que-es-un-avatar-en-informatica/*.

[2] Web 2.0. [online]: *https://iweb2o.webnode.com.co/avatar/*

[3] Voki [online]: *http://www.educacontic.es/blog/voki-personajes-que-hablan-text-speech*

[4] Git y GitHub [online]: *https://desarrolloweb.com/articulos/introduccion-git-github.html*

[5] LoopBack [online]: *https://loopback.io/*

[6] Gamificación [online]: *https://blog.atrivity.com/es/que-es-gamificacion-y-como-usarla*

[7] Gamification at Work Survey [online]: *https://www.talentlms.com/blog/gamification-survey-results/*

[8] Classdojo [online]: *https://www.educaciontrespuntocero.com/recursos/classdojo-que-es-como-empezar/33376.html*

[9] Gamificación en la asignatura de Derecho Romano [online]: *https://www.researchgate.net/profile/Sonia*$_{Pamplona}$

[10] API-REST [online]: *https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos*

[11] Bootstrap [online]: *https://raiolanetworks.es/blog/que-es-bootstrap/*

[12] Ionic [online]: *https://ionicframework.com/*

[13] Angular [online]: *https://angular.io/*

[14] Kahoot [online]: *https://www.xataka.com/basics/kahoot-que-es-para-que-sirve-y-como-funciona*

[15] Bitmoji [online]: *https://www.bitmoji.com/*

[16] NodeJS [online]: *https://www.netconsulting.es/blog/nodejs/*

[17] Cordova [online]: *https://es.wikipedia.org/wiki/Apache*$_{Cordova}$

[18] Zombies Run! [online] *https://zombiesrungame.com/*

APÉNDICES

APÉNDICE A. MANUAL DE INSTALACIÓN

En esta sección aparecen los pasos a seguir para instalar todas las herramientas que forman parte de la arquitectura del software.

A.1. Node y NPM

La versión de NODE con la que se ha desarrollado el proyecto es la 8.6.0, sin embargo esta versión no es compatible con Angular CLI, así que utilizaremos la versión 8.9.4.

Para esto accederemos a la web del link adjuntado a continuación:

https://nodejs.org/es/download/releases/

Figura A.1: Opciones instalación NODE

Entrando en la sección de"Todas las versiones", se abrirá el siguiente enlace https://nodejs.org/dist/, donde visualizaremos las versiones disponibles hasta la fecha. Ahí haremos click en la versión que queremos descargar.

Figura A.2: Versión de instalación NODE

El NPM se instala automáticamente tras realizar la instalación del NODE. Para ver qué versión se ha instalado de cada uno de ellos, podemos abrir una línea de comandos o como aparece en Windows "Símbolo del sistema".

Por tanto la versión del NPM no es importante, ya que viene ligada a la del NODE, que es la que nos interesa.

Mediante las órdenes node –v y npm –v, aparece en pantalla la versión correspondiente a cada una de las herramientas.

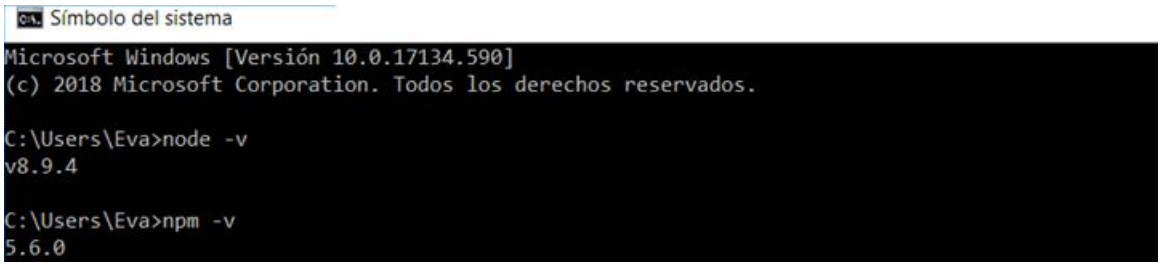


Figura A.3: Comprobación de versiones NODE y NPM

A.2. Git

Github es una herramienta para alojar proyectos utilizando el sistema de control de versiones de Git. Justamente se ha usado anteriormente para guardar los repositorios de las versiones desarrolladas del proyecto hasta la fecha. El sitio para descargar la versión adecuada es el siguiente: https://git-scm.com/downloads

Con el comando git –version, comprobamos qué versión se ha instalado. La versión del Git, de la misma manera que la versión del NPM, no es importante.

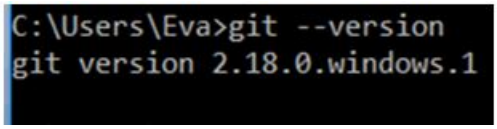


Figura A.4: Versión Git

A.3. Sourcetree

Sourcetree es un programa que sirve para manejar los repositorios de Git. Se utiliza para ver las diferentes modificaciones y versiones que han añadido los contribuidores del proyecto.

Sólo está disponible para Windows y Mac OS, en el siguiente link se accede a la descarga: https://www.sourcetreeapp.com/

A.4. Visual Studio Code

Visual Studio Code es el programa recomendado para desarrollar código, aunque también existen otras opciones, por ejemplo WebStorm de JetBrains. Ambas herramientas son gratuitas, simplemente hemos procedido con Visual Studio, que está disponible para sistemas operativos de Windows, Linux y Mac OS. Para descargarlo utilizamos el enlace que aparece a continuación:

https://code.visualstudio.com/alt-downloads

Si el sistema operativo que se utiliza es Linux, hay que ejecutar el siguiente comando desde el cmd: sudo dpkg -i nombrearchivo.deb. No se ejecutará el comando sin antes haber

descargado el archivo de Visual Studio Code. Una vez descargado abrimos la carpeta en la que se haya descargado dicho archivo, y entonces procedemos a ejecutar la orden.

Para desarrollar código en HTML5, basado en Typescript y Angular2 se tienen que instalar los siguientes plugins del Visual studioCode, tras haber realizado la instalación de éste.

- Plugins Angular 2

  https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2

- Beautify

  https://marketplace.visualstudio.com/items?itemName=HookyQR.beautify

- EditorConfig

  https://marketplace.visualstudio.com/items?itemName=EditorConfig.EditorConfig

- EsLint

  https://marketplace.visualstudio.com/items?itemName=dbaeumer.vscode-eslint

- Sass-Lint

  https://marketplace.visualstudio.com/items?itemName=glen-84.sass-lint

- TsLint

  https://marketplace.visualstudio.com/items?itemName=eg2.tslint

- TypeScript

  https://marketplace.visualstudio.com/items?itemName=DSKWRK.vscode-generate-getter-setter

A.5. Android Studio

Android Studio no es una herramienta del todo necesaria, pero puede resultar útil para debuggear la aplicación de Android creada por cordova. En todo caso, la que sí que es necesaria es Android SDK.

El programa que soportan Windows, Mac OS y Linux es el que aparece en este link:

https://developer.android.com/studio/index.htmldownloads

A.6. Repositorios

En el previo apartado del Git se habla de los repositorios de las versiones anteriores. Para entender un poco mejor qué quiere decir esto, tenemos que saber que el código desde el que se parte se clona, es decir, no se modifica directamente.

Es necesario tener un usuario propio en Github para trabajar con el código y hacer el “Fork” de los tres repositorios existentes, que más adelante se especificará qué es. Por ahora se focaliza la instalación.

Si se utiliza una máquina Windows, primero debe verificarse que se dispone de complementos como el Python o el .NET Framework. En caso de no disponer de ellos, pueden instalarse mediante la siguiente instrucción: npm install –global –production windows-build-tools.

A continuación se indicará cómo hacer el set up, o cómo poner en marcha los tres repositorios.

A.7. Classpip-Services

1. Entramos en el siguiente link de github https://github.com/classpip/classpip-services, y hacemos click en el botón que pone Fork.

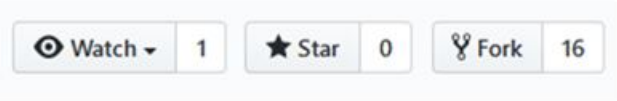


Figura A.5: Opciones disponibles en Git

2. El botón que aparece en verde, en el que pone “Clone or download” da dos opciones, clonar el repositorio o descargarlo como archivo ZIP.

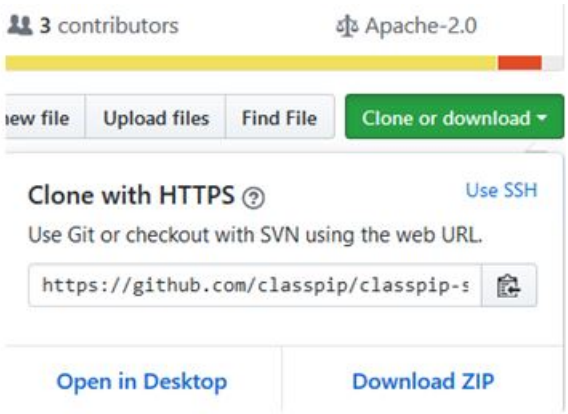


Figura A.6: Repositorio Classpip-Services en Git

Previo a tomar cualquiera de las dos opciones, se crea una carpeta donde guardar los repositorios. Es preferible que se cree dentro de la carpeta del usuario desde el que se trabaje con el cmd.

A

Para clonar el repositorio desde el cmd tenemos que conocer algunas de las herramientas y comandos que se utilizan con Git. Pese a que no es muy recomendable escoger esta vía sin conocer Git, también es útil para familiarizarse con esta herramienta de trabajo.

El cmd se inicializa por defecto en la carpeta del usuario desde el que se ejecuta. Si la carpeta ha sido creada en otra carpeta diferente, mediante la instrucción cd

nombre carpeta. En este caso se ha escogido el escritorio para clonar directamente la carpeta, de ahí que aparezca por pantalla el comando cd Desktop.

Hecho esto inicializamos Git con el comando git init , órden a la que el cmd responde con un mensaje de Initialized empty Git repository in C:/Users/NombreUsuario/CarpetaDondeClonamo

Lo siguiente es enviar la orden que clona el repositorio. Para esto el comando que se utiliza es git clone –o repo –branch NombreDeLaRama URLDeLaRama.

El nombre de la rama se localiza en el mismo usuario de GitHub, en este caso está comprobado que se llama master.

Figura A.7: Rama máster del repositorio en Git

La URL del repositorio se copia del usuario de Git haciendo click en el icono del portapapeles que aparece al lado de la dirección.

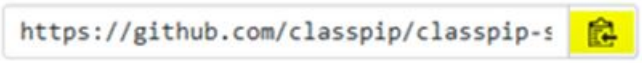


Figura A.8: Icono para copiar repositorio en Git

Todas las órdenes ejecutadas en cadena se muestran en pantalla de la siguiente manera.

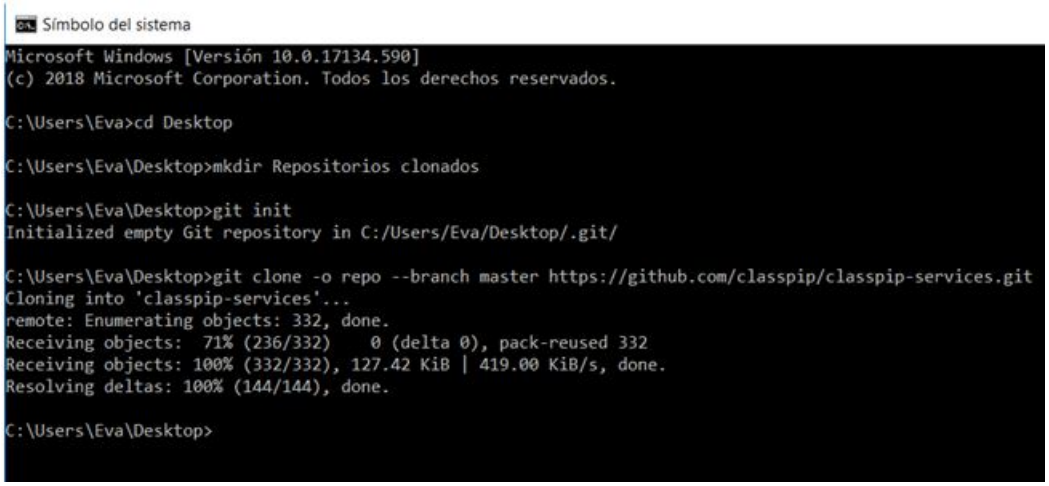


Figura A.9: Ordenes cmd para copiar repositorio en el PC

Lo que se ha conseguido es descargar la carpeta completa del repositorio classpip-services en el escritorio, que es lo mismo que se necesita hacer con el resto de repositorios.

B

La opción alternativa es mucho más simple. Consiste en descargar la carpeta en formato ZIP dándole al botón de “Download ZIP”. A partir de aquí el archivo puede descomprimirse y ser extraído en la carpeta que más convenga.

3. Descargar strongloop, un útil que sirve para crear relaciones en el código.

   Las instrucciones en las que aparece una –g no se ejecutan si no se abre el cmd como administrador. Presionando con el botón derecho del ratón sobre “Símbolo del sistema” aparece la opción de “Ejecutar como administrador”.

   La instrucción es la siguiente:

   ```
   npm install -g strongloop@6.0.3
   ```

4. Instalar las dependencias del proyecto mediante la siguiente instrucción:

   ```
   cd CarpetaDondeEstáElRepositorio/classpip-services npm install
   ```

5. Ejecutar el servicio classpip con la instrucción que aparece a continuación:

   ```
   npm run start
   ```

   Tras ejecutar esta orden, se abre automáticamente una pestaña con la URL siguiente:

   http://localhost:3000/explorer/

   Este es el servidor, que dispone de una base de datos. De aquí en adelante, siempre que sea necesario ejecutar órdenes desde el cmd para otros repositorios, se debe hacer desde otra ventana de cmd diferente. La ventana cmd usada para este repositorio se deja abierta.

A.8. Classpip-Mobile

1. Hacer el Fork del repositorio, clonarlo o bien descargarlo en formato ZIP, tal y como se ha explicado en el apartado anterior, del siguiente enlace:

   https://github.com/classpip/classpip-mobile

2. Instalar las dependencias globales (ejecutar cmd con permisos de administrador), de Ionic y Cordova usando las siguientes órdenes:

```
npm install -g ionic@2.2.2

npm install -g cordova@6.4.0
```

3. Instalar las dependencias locales para la aplicación móvil mediante estas órdenes, ejecutadas una a una:

```
cd CarpetaDondeEstáElRepositorio/classpip-mobile

npm install

mkdir www

cordova prepare
```

La aplicación se ejecuta con la siguiente instrucción:

```
ionic serve
```

Y si se desea ver cómo queda la aplicación en una pantalla de un dispositivo móvil, se puede visualizar mediante la siguiente orden:

```
Ionic serve --lab
```

Para empezar a hacer pruebas programando en un servicio local, como un usuario cualquiera, deben comentarse algunas líneas del fichero:

`classpip-mobile/src/app/app.config.ts`

Concretamente las siguientes líneas:

```
// public static get SERVER_URL(): string { return 'https://api.classpip.com'; } // PRO
   public static get SERVER_URL(): string { return 'http://localhost:3000'; } // DEV
```

Figura A.10: Líneas a comentar

A.9. Classpip-Dashboard

Este repositorio es el que contiene todos los archivos de código con los que se trabaja, independientemente de cuál sea su extensión. Siguiendo pasos análogos a los de los anteriores repositorios se accede a trabajar con éste.

1. Hacer el Fork, clonar y abrir el repositorio del siguiente enlace:

   https://github.com/classpip/classpip-dashboard

2. Instalar todas las dependencias locales que hay en la carpeta del classpip-dashboard con la instrucciones que aparecen a continuación, ejecutadas una a una:

   ```
   cd CarpetaDondeEstáElRepositorio/classpip-dashboard
   ```

   ```
   npm install
   ```

3. ejecutar la aplicación utilizando la siguiente instrucción:

   ```
   npm run start
   ```

   Con tal de abrir la aplicación, hay que abrir el navegador e introducir la siguiente dirección:

   http://localhost:4200/

Estos son todos los elementos que deben instalarse en caso de querer emprender unnuevo proyecto relacionado con Classpip.

APÉNDICE B. IMAGENES DE LAS FUNCIONALIDADES PARA EL DOCENTE

La aplicación muestra diferentes vistas, aquí se recogen las que corresponden al usuario de rol profesor.

Figura B.1: Pantalla de inicio de sesión

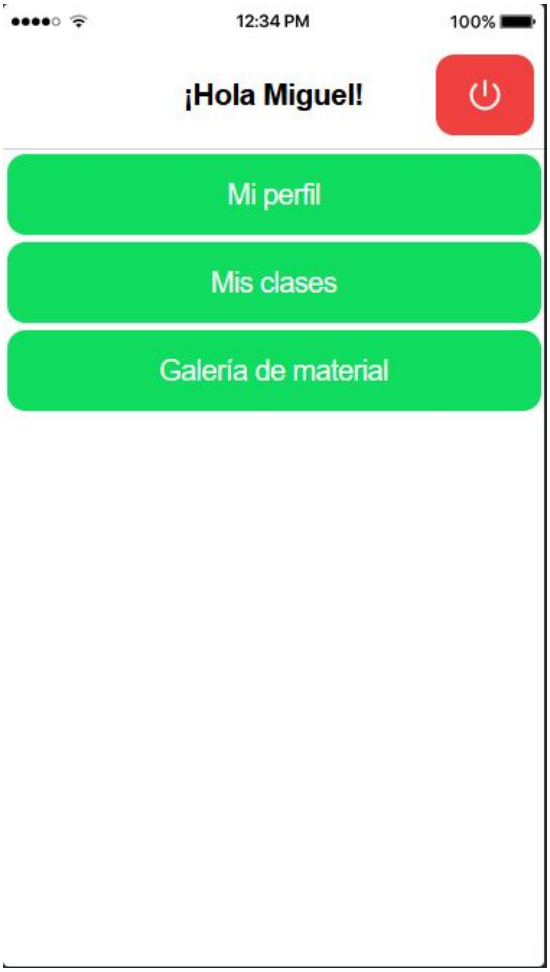


Figura B.2: Menú de inicio para profesor

Tras el inicio de sesión, como puede verse aparecen tres opciones en el menú principal y un botón rojo arba a la derecha que sirve para cerrar la sesión.

A continuación aparecen las diferentes páginas que se despliegan en cada una de las opciones del menú.

B.1. Mi perfil

Empezando por “Mi Perfil”, vemos todos los datos del docente, y además hay una función para cambiar la contraseña, que extiende una entrada de texto, tal y como se observa en la figura B.4.

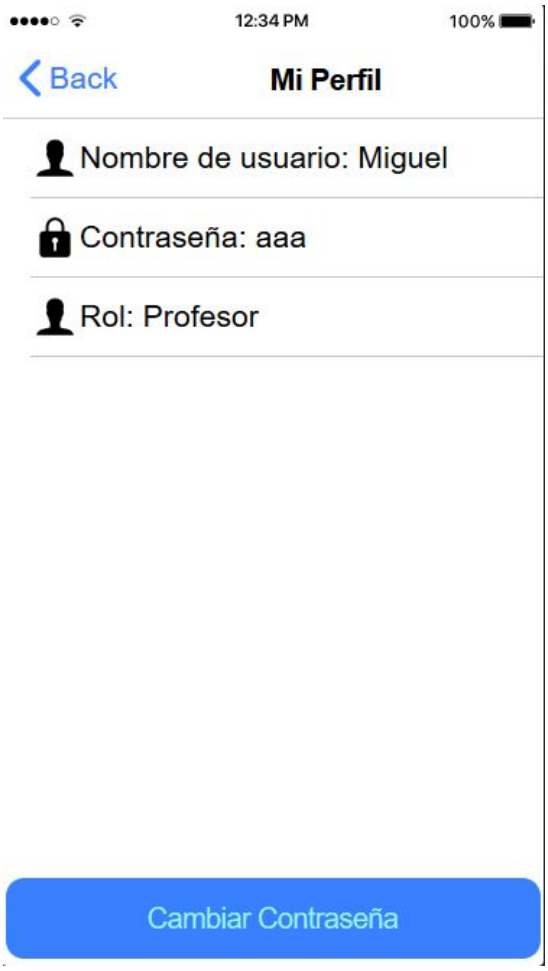


Figura B.3: Apartado “Mi perfil”

Figura B.4: Elección de unanueva contraseña

B.2. Mis clases

Volviendo al inicio, el siguiente apartado, con título “Mis clases”, muestra la siguiente apariencia:

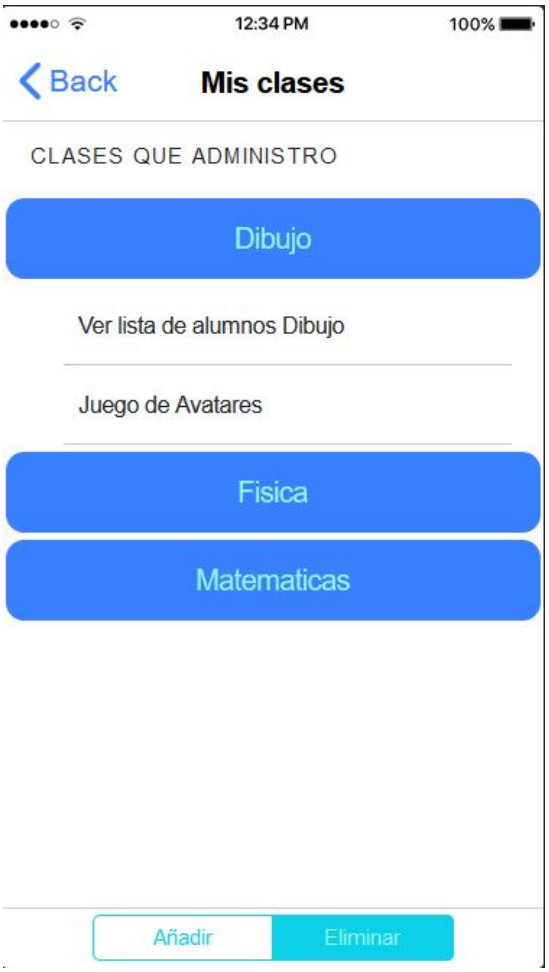


Figura B.5: Apartado “Mis clases” para rol de profesor

La opciones de añadir clase y eliminar clase están disponibles en la barra inferior dela pantalla. Para añadir una clase aparece un formulario a rellenar mientras que para eliminar la clase simplemente se pide introducir el nombre que identifica a la asignatura.

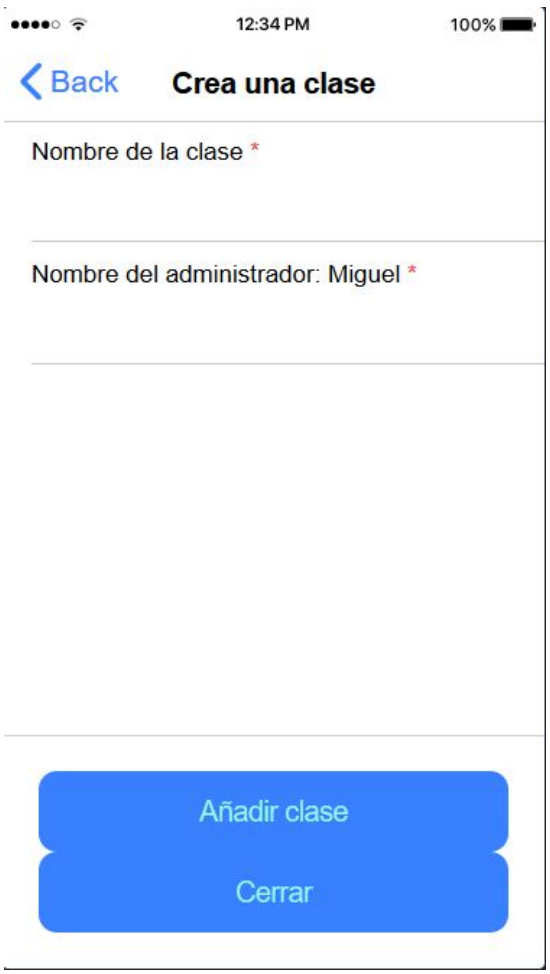


Figura B.6: Añadir clase

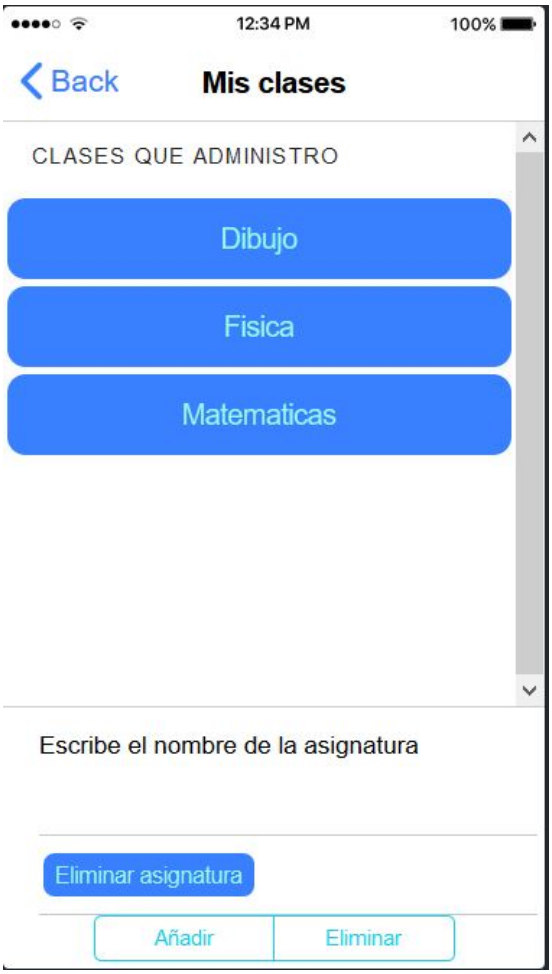


Figura B.7: Eliminar clase

El siguiente paso es visualizar las páginas de las opciones “Ver lista de alumnos” y “Juego de Avatares” que aparecen para cada clase.

La lista de alumnos varía ligeramente si el juego de avatares está activado, comparada con la versión de la lista que aparece cuando no hay avatares.

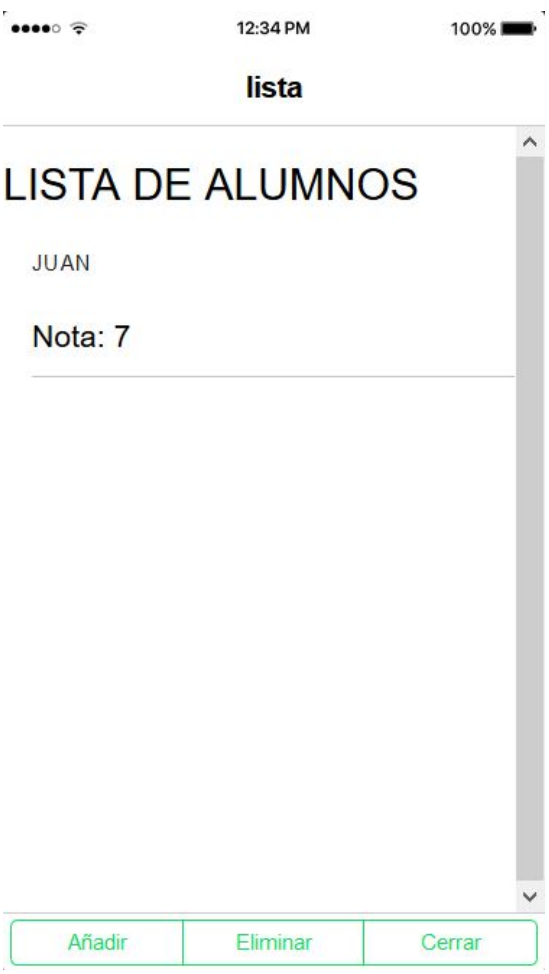


Figura B.8: Lista de alumnos con el juego de avatares desactivado

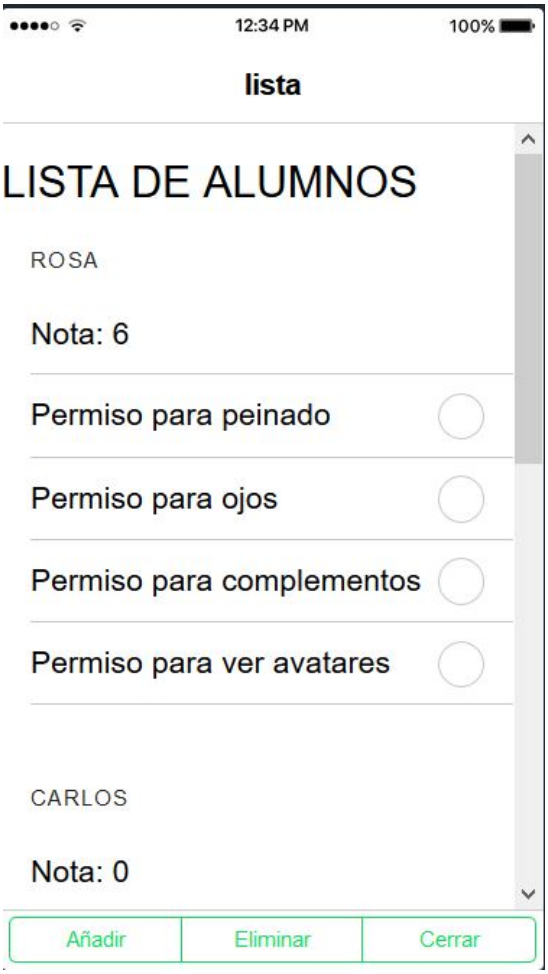


Figura B.9: Lista de alumnos con el juego de avatares activo

Para añadir alumnos en la clase, aparece un formulario muy similar al de añadir clases como se muestra en la figura a continuación.

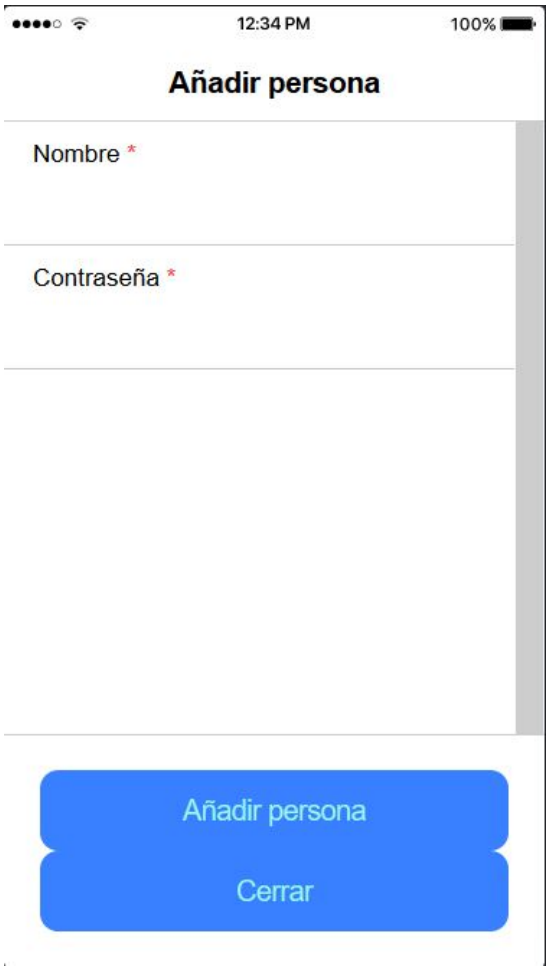


Figura B.10: Formulario para añadir alumnos a la clase

B.3. Juego de avatares

Para activar el juego de avatares, es necesario entrar en la sección “Juego de Avatares”, que es única para cada clase. Aquí se activa el juego con un botón “toggle”. Al clicar en el botón, éste se vuelve de color verde y aparecen todos los elementos relacionados con el juego en un apartado centrado en la pantalla.

Figura B.11: Juego de avatares desactivado

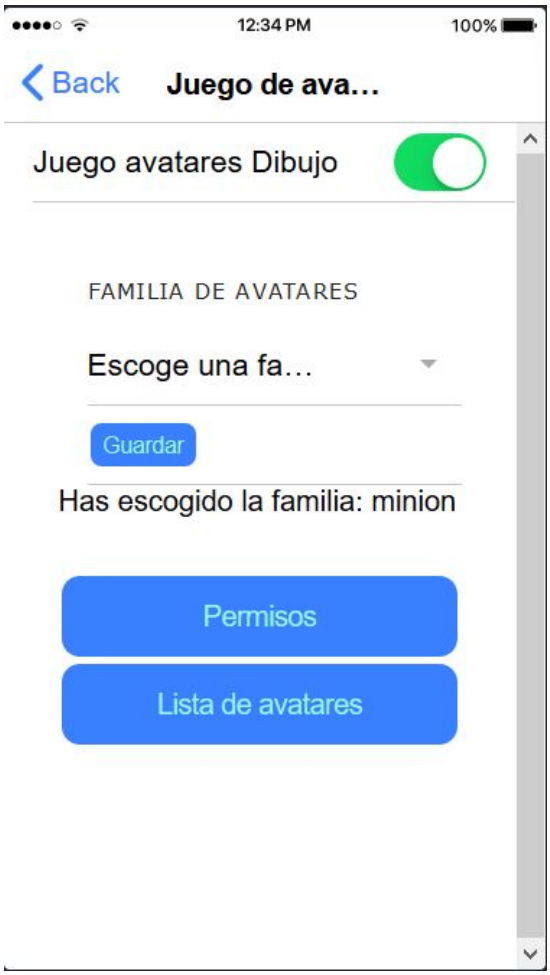


Figura B.12: Juego de avatares activado

Escoger la familia de avatares, como ya habíamos comentado en capítulos anteriores, es muy simple, el desplegable que aparece en la plantalla muestra las opciones que hay.

Tras escoger familia y clicar en el botón “Guardar” de la figura B.12, tras la frase “Has escogido la familia:” aparece el nombre de la última familia guardada.

Figura B.13: Formulario para añadir alumnos a la clase

Respecto a la sección de “Permisos” encontramos una página donde se pueden cargar archivos de texto. En la barra de la parte inferior de la pantalla hay dos botones, uno a cada lado. El botón cargar extiende una pantalla que permite seleccionar archivos del dispositivo en el que se utiliza la aplicación. Una vez seleccionado el archivo, haciendo clic en “Subir”, éste se carga en la base de datos.

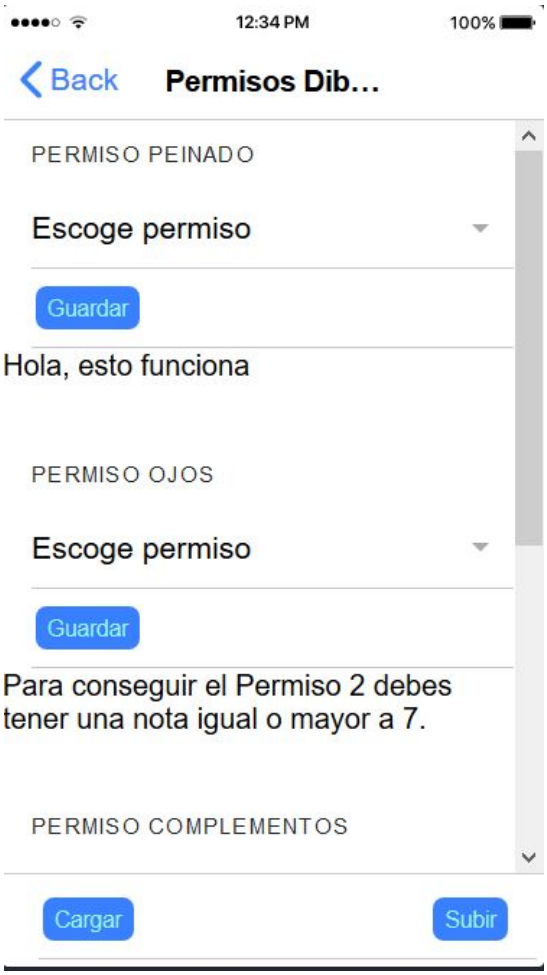


Figura B.14: Carga de archivos de texto para mostrar en permisos

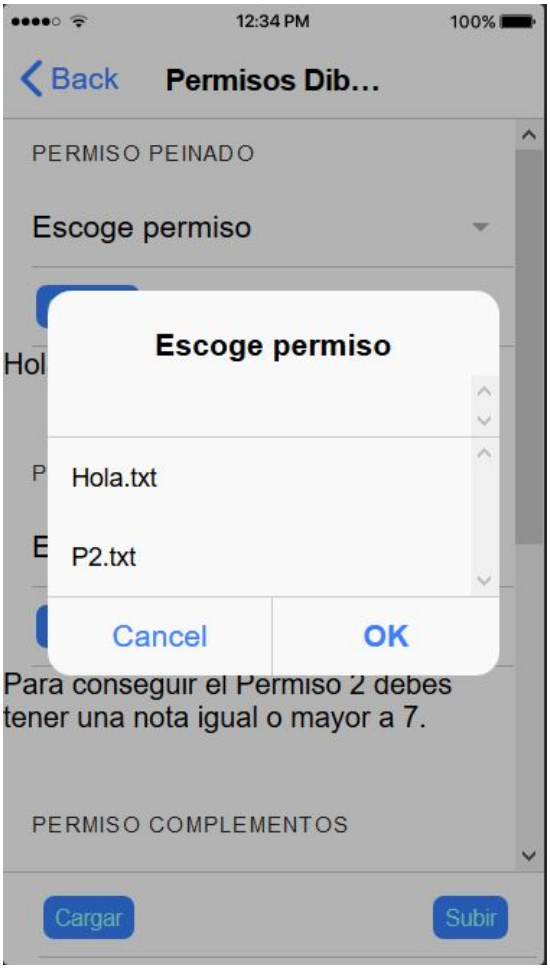


Figura B.15: Selección de archivo para mostrar por pantalla

Para cada permiso hay un desplegable, que permite ver todos los archivos que se han cargado previamente. Eligiendo un archivo y clicando en el botón de Guardar, el texto del archivo aparece por pantalla. Puede apreciarse la secuencia de elementos descrita en la figura B.14.

Dentro de esta sección de juego de avatares, se permite que el docente pueda tener un listado de todos los avatares de la clase, que toma la apariencia de las siguiente figuras.

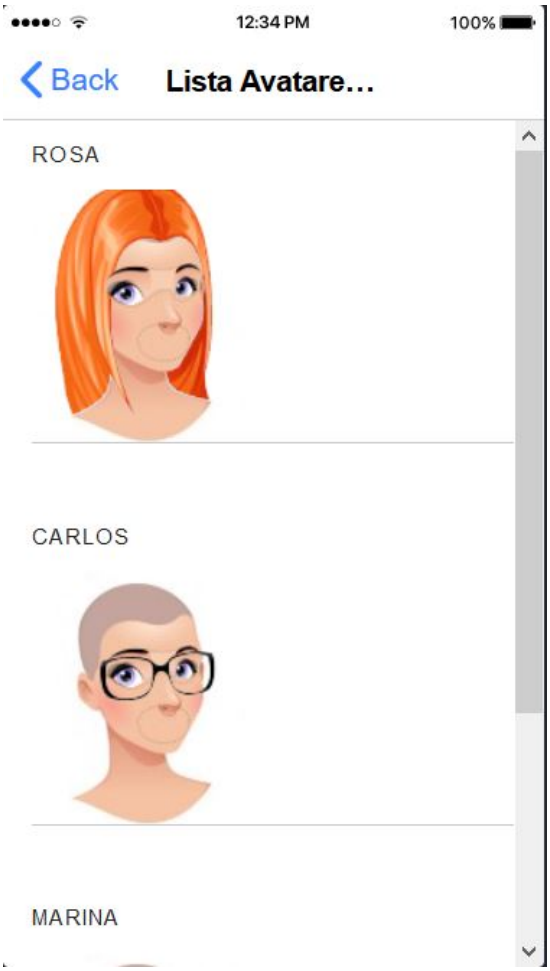


Figura B.16: Lista de avatares familia “persona”

Figura B.17: Lista de avatares familia “minion”

B.4. Galería de material

La galería de material es el último apartado que queda por mostrar. Está organizda de tal manera que cada elemento del avatar tiene su propio apartado y dos botones, el de “Añadir” da opción a cargar una nueva foto en la base de datos, y el de “Mostrar” permite ver lo que ya hay en esa sección de la base de datos.

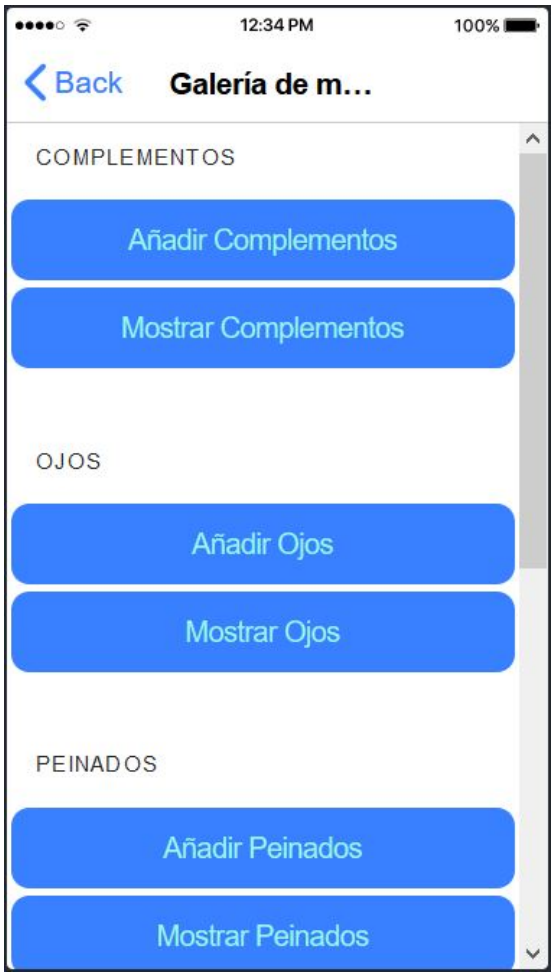


Figura B.18: Galería de material a)

Figura B.19: Galería de material b)

En las siguientes figuras aparece todo el material que se ha diseñado para construir los avatares.

- Material para complementos y bustos del avatar

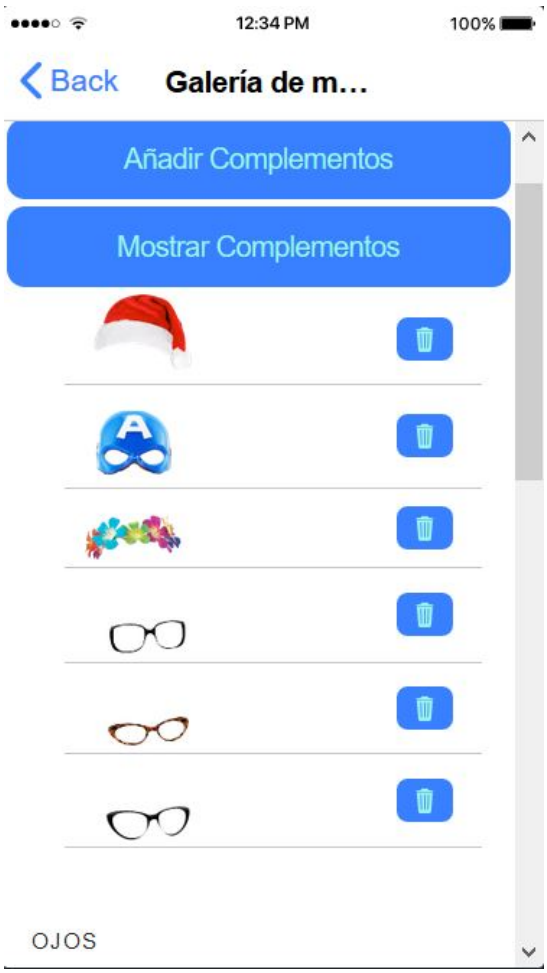


Figura B.20: Apartado de complementos

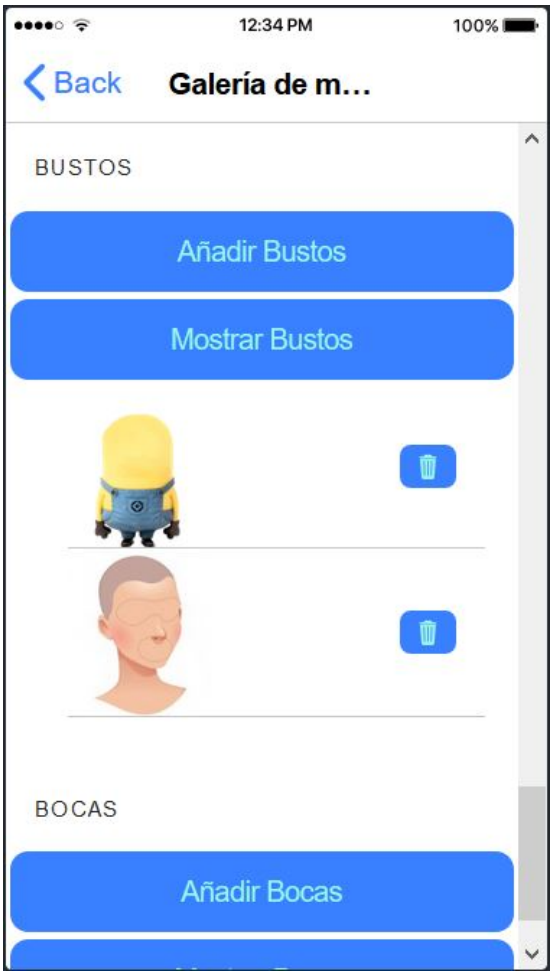


Figura B.21: Apartado de bustos

- Material para peinados:

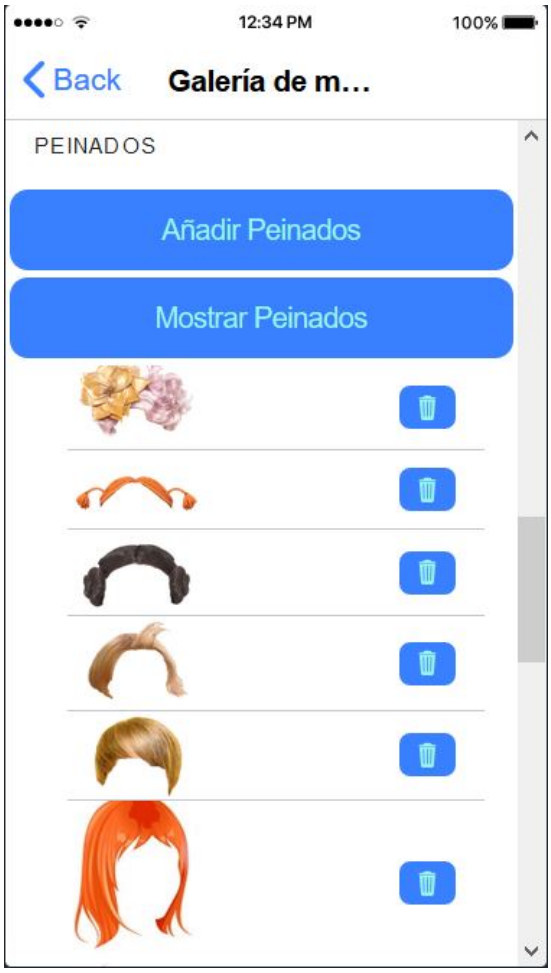


Figura B.22: Apartado de peinados a)

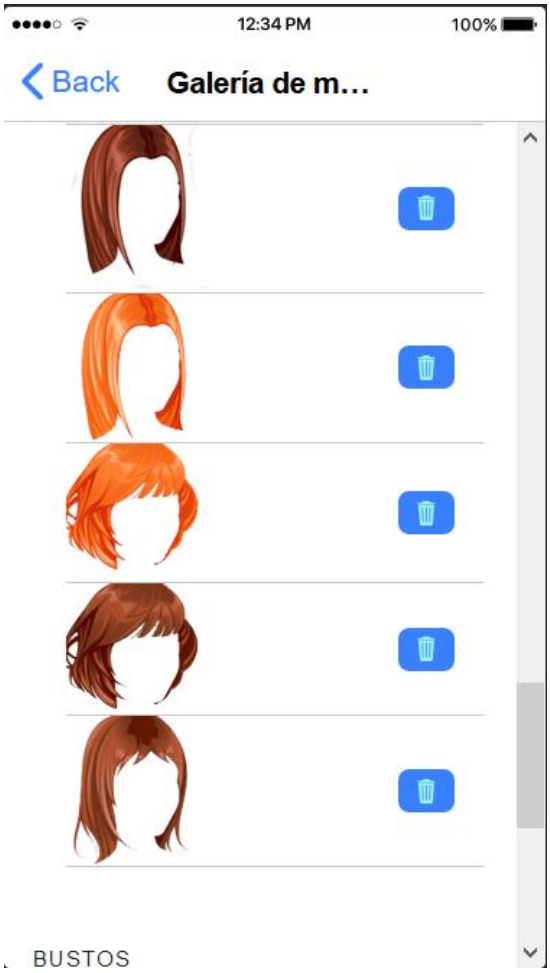


Figura B.23: Apartado de peinados b)

- Material para ojos:

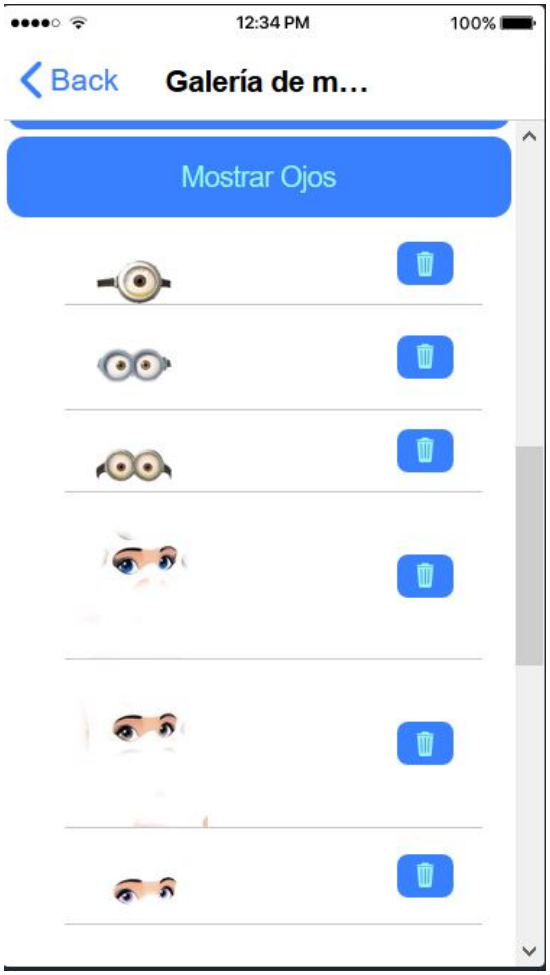


Figura B.24: Apartado de ojos a)

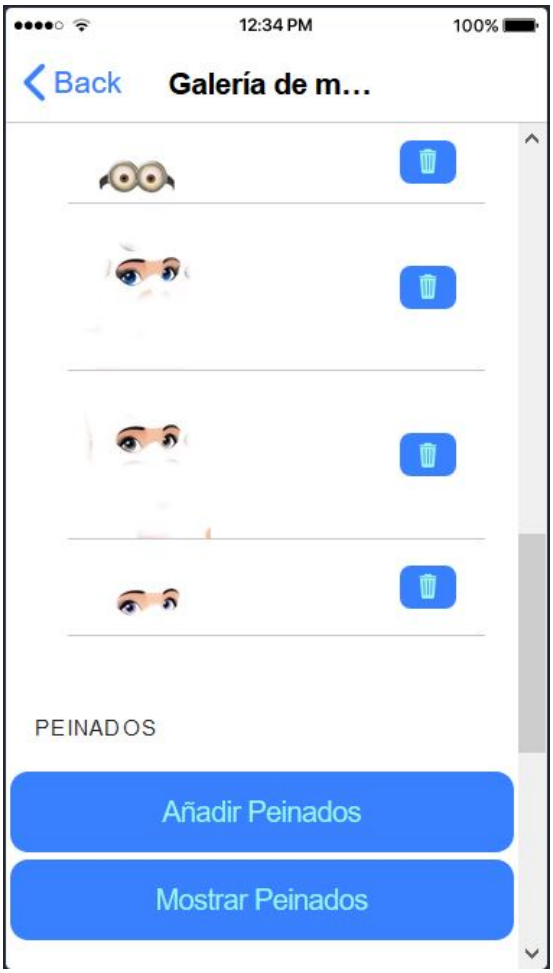


Figura B.25: Apartado de ojos b)

- Material para bocas:

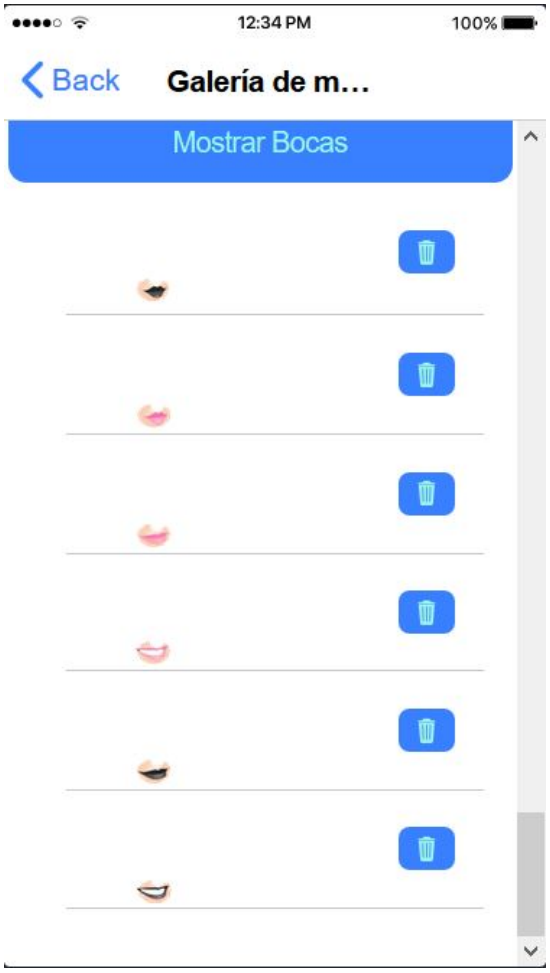


Figura B.26: Apartado de bocas

Estas son todas las pantallas de las que dispone el usuario con rol de profesor. A continuación se muestran todas las que recoge el usuario alumno.

APÉNDICE C. IMAGENES DE LAS FUNCIONALIDADES PARA EL ESTUDIANTE

El estudiante tiene bastantes menos funcionalidades que el profesor. La pantalla para iniciar sesión en la aplicación sin embargo, es la misma.

Figura C.1: Pantalla para iniciar sesión

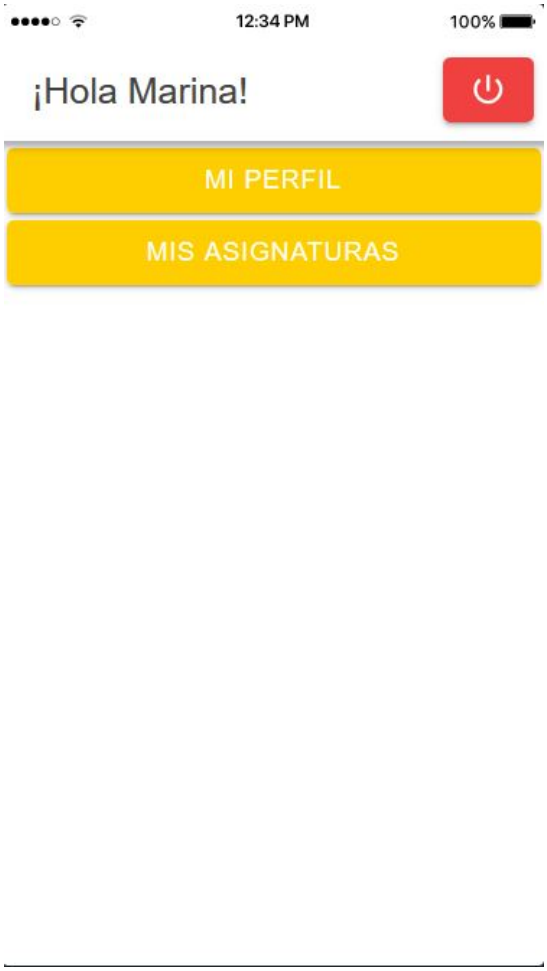


Figura C.2: Menú de inicio Alumno

Los dos apartados que se muestran parecen idénticos a los del profesor, y de hecho la primera impresión es esa, no hay nada diferente a simple vista, sin embargo como veremos a continuación en el apartado de “Mis asignaturas” no se ofrecen las mismas funcionalidades. El apartado “Mi perfil” sí que es idéntico.

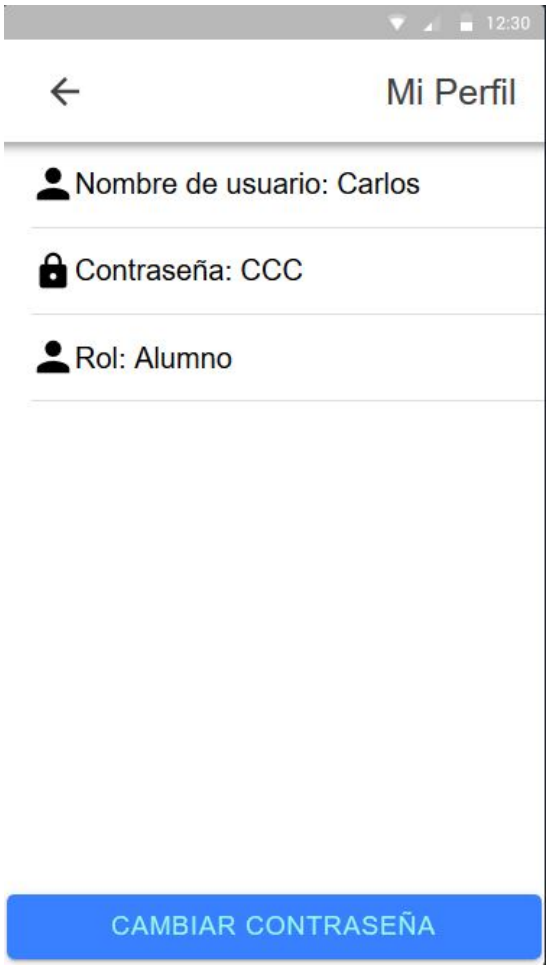


Figura C.3: Apartado “Mi perfil” para rol de alumno

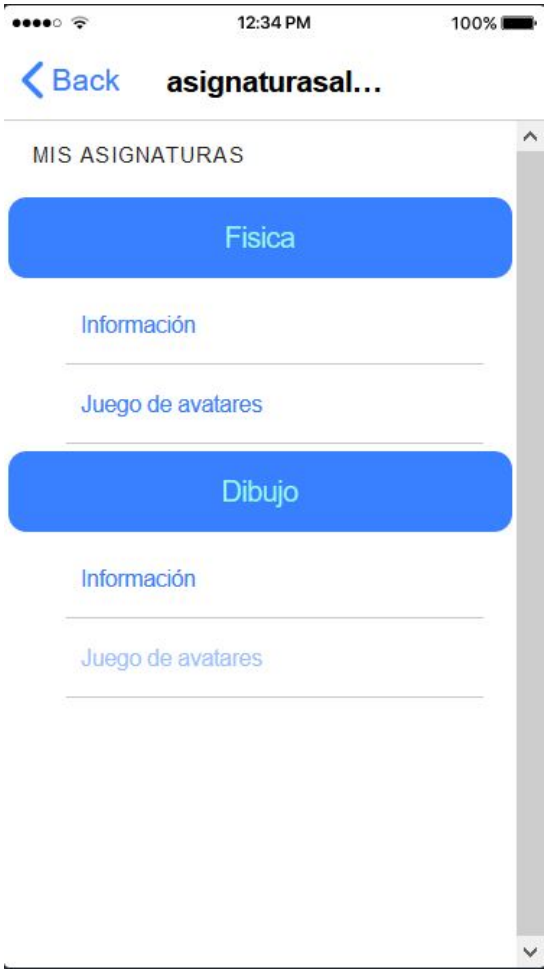


Figura C.4: Apartado de asignaturas para el alumno

Como puede apreciarse en la figura C.4, la sección“Juego de avatares” bajo el botón en el que pone Dibujo tiene un color muy apagado, lo que significa que para la clase de Dibujo el juego no está activado. Puede verse la diferencia de color comparado con la clase de Física, que está justo encima.

Dentro de cada asignatura, los apartados de “Información” y “Juego de avatares” se ven como en las imagenes expuestas a continuación.

Figura C.5: Apartado “Información”

Figura C.6: Sección “Juego de avatares” para alumno

Las siguientes imágenes pertenecen a los diferentes apartados de material para el avatar. La diferecia de estos apartados con la galería de material del profesor es que aquñi se hace un filtro de material por familias. El programa detecta si el archivo pertenece a una familia o a otra gracias a las reglas de nomenclatura.

- Sección de peinados:

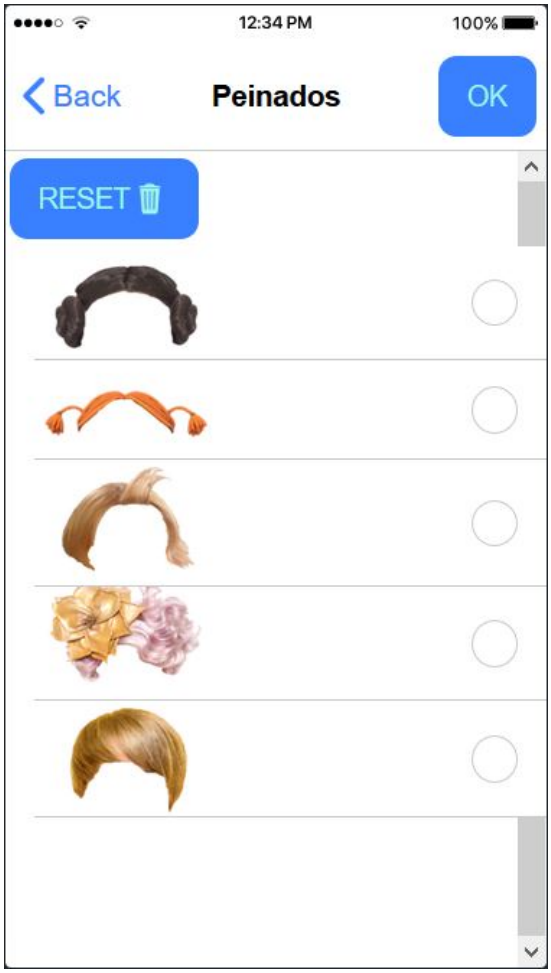


Figura C.7: Peinados familia “minion”

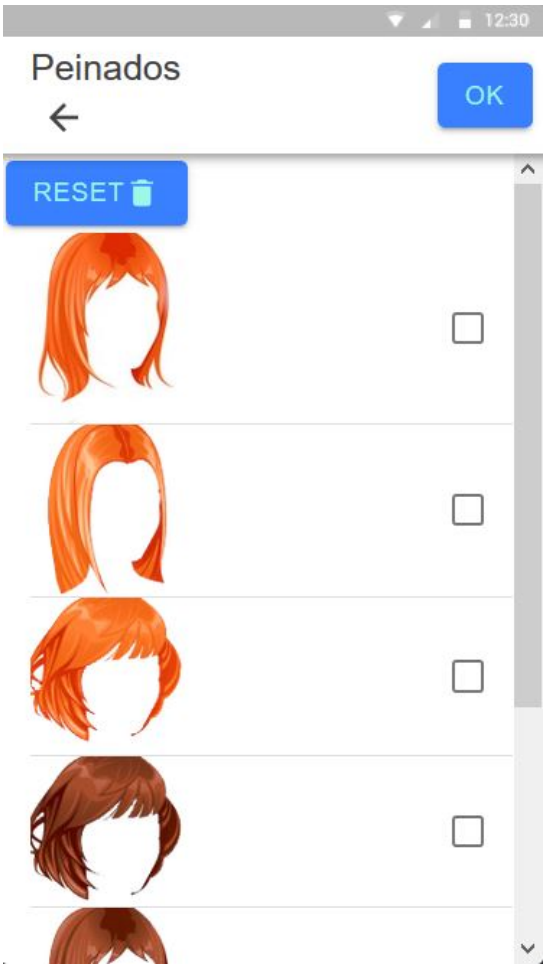


Figura C.8: Peinados familia “persona”

- Sección de ojos

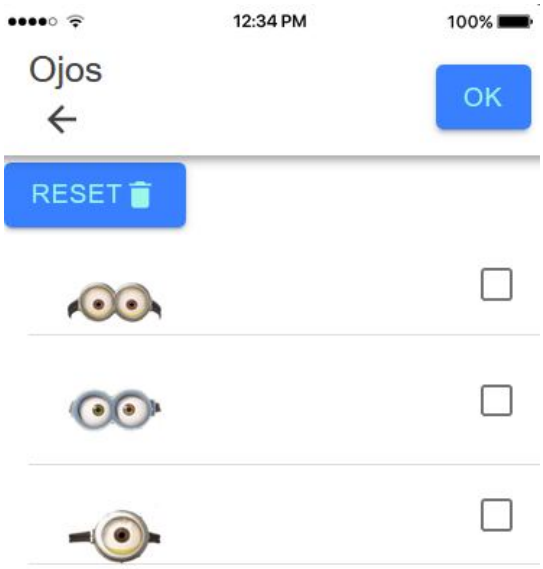


Figura C.9: Ojos familia “minion”

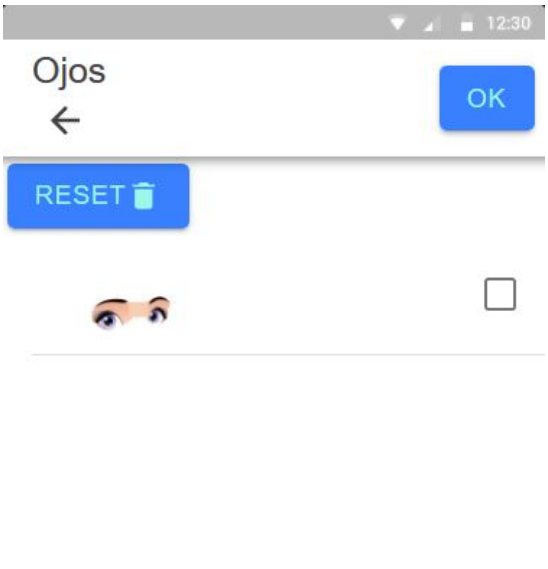


Figura C.10: Ojos familia “persona”

- Sección de complementos:

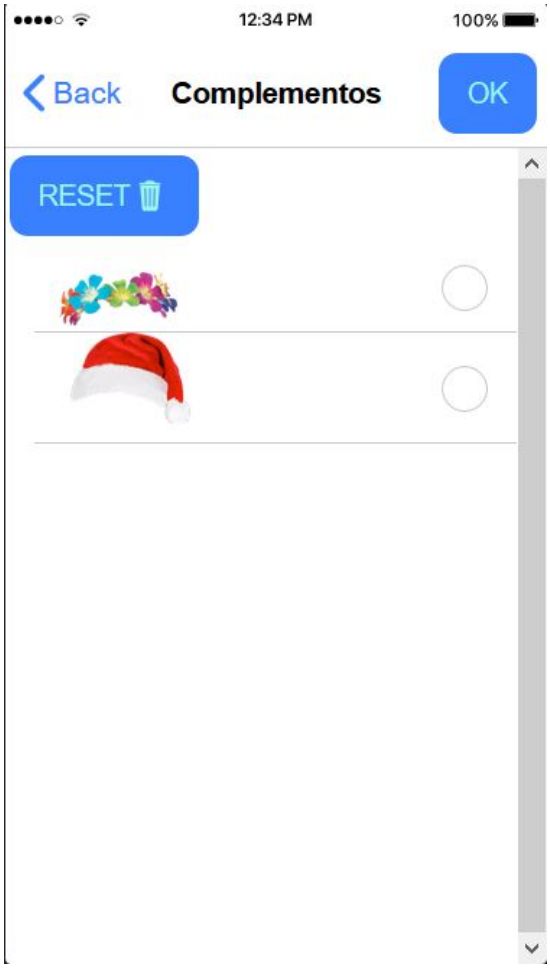


Figura C.11: Complementos familia “minion”

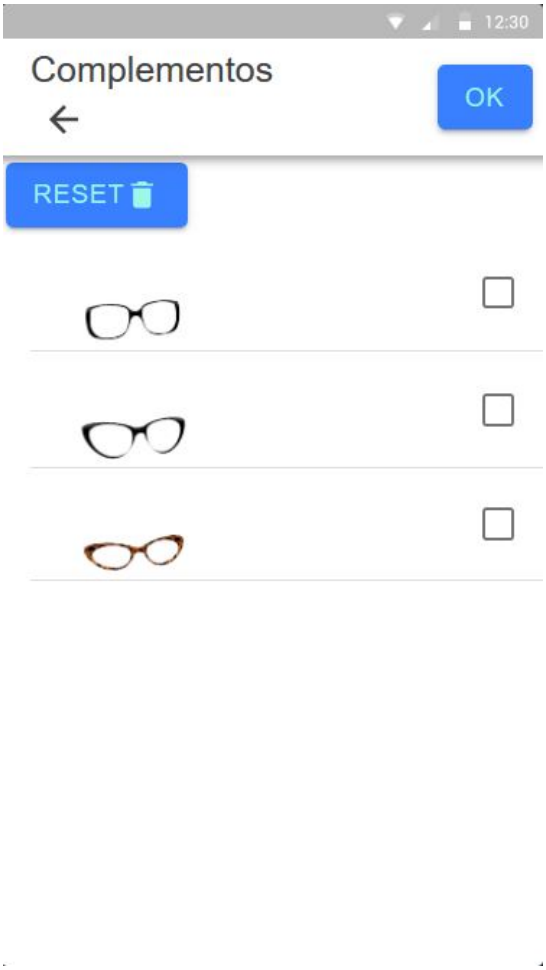


Figura C.12: Complementos familia “persona”

- Sección de bocas:

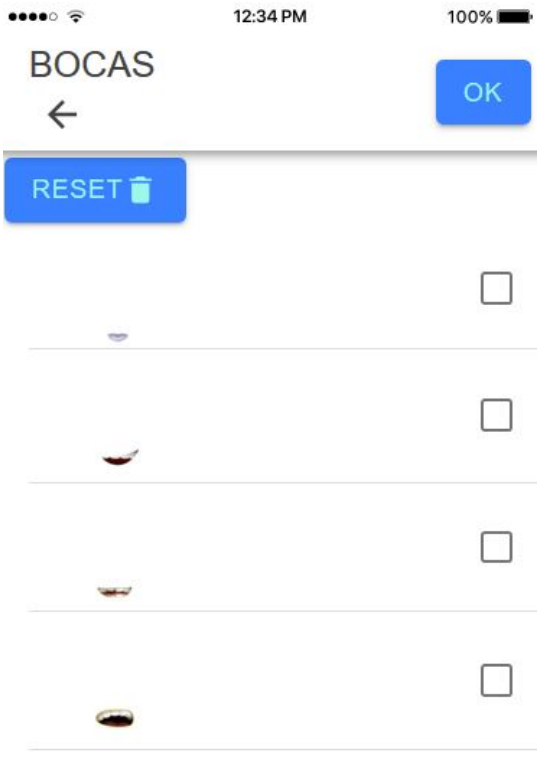


Figura C.13: Bocas familia “minion”

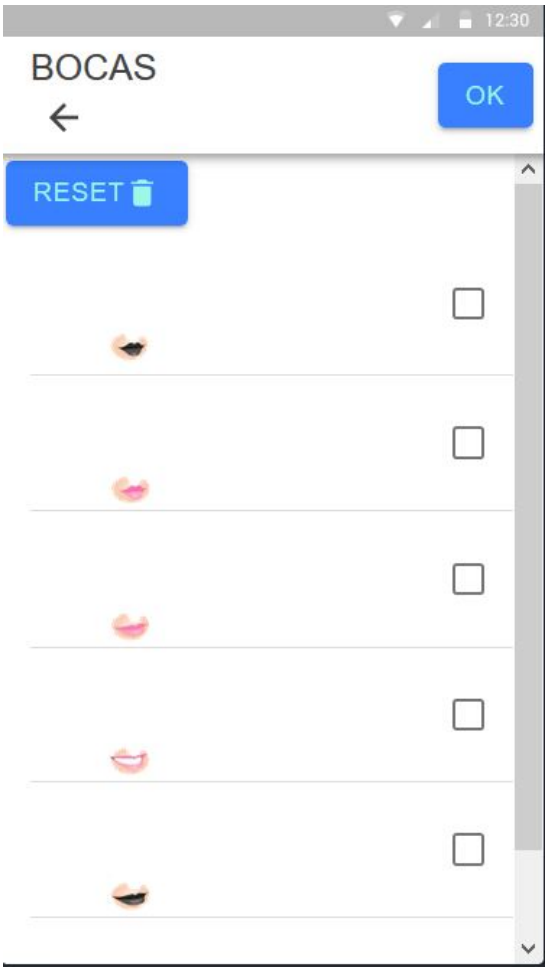


Figura C.14: Bocas familia “persona”

Por último sería necesario explicar las últimas vistas a la que se puede acceder desde el apartado de personalización del avatar.

Por una parte tenemos la lista de avatares. Si el botón morado que aparece en la parte superior de la derecha está activado, el alumno tiene permiso para visualizar la lista de avatares. La diferencia entre un alumno que puede acceder y otro que no está en el color del botón, que se ve más apagado si no está accesible.

Figura C.15: Permiso para ver la lista de avatares concedido

Figura C.16: Permiso para ver la lista de avatares denegado

La versión de la lista a la que se accede es exactamente la misma que la que puede ver el profesor, aparece en las figuras B.16 y B.17.

Con la siguiente vista se termina por completo el abanico de páginas a las que accede el alumno.

En la sección “Permisos” aparecen los requisitos que el alumno debe cumplir para poder acceder a los apartados de peinados, ojos, complementos y ver la lista de avatares. Cada permiso tiene un indicador que aparece en verde si éste ha sido condedido.

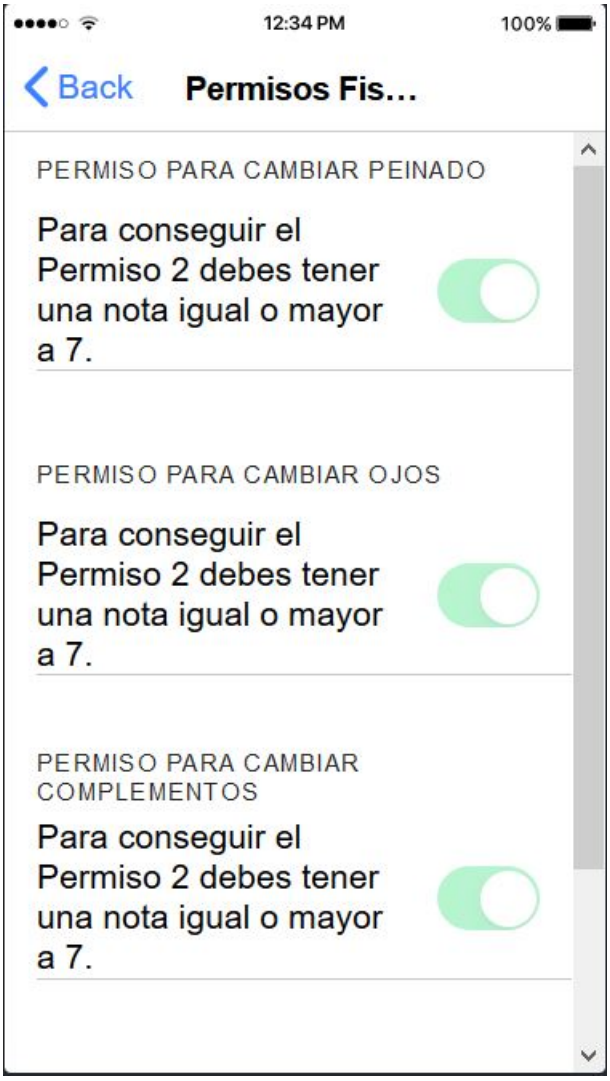


Figura C.17: Vista de permisos para alumno

APÉNDICE D. IMÁGENES BATERÍA DE PRUEBAS

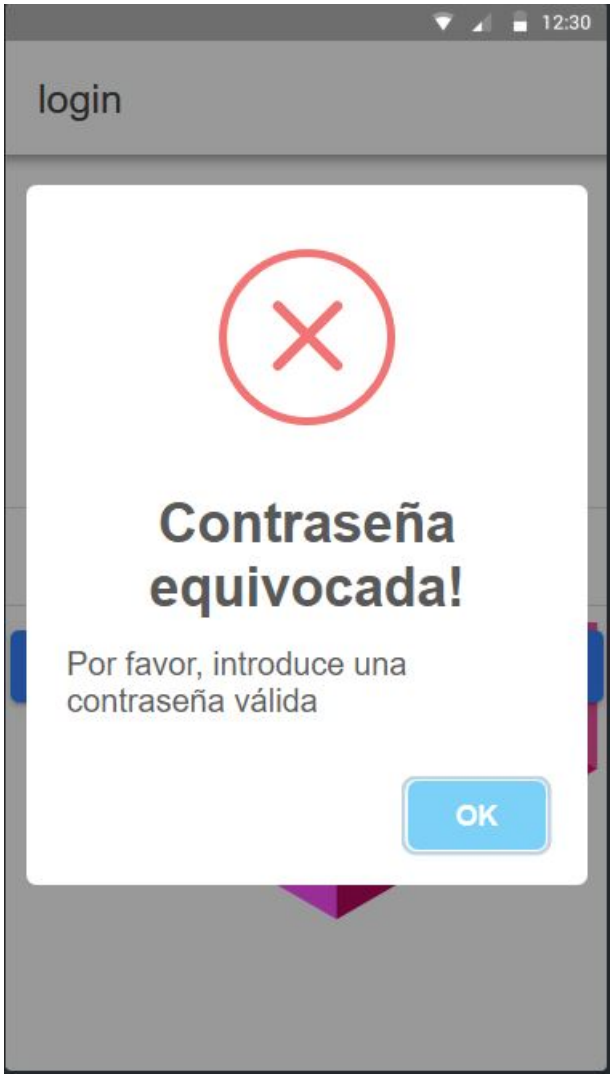


Figura D.1: Contraseña equivocada

Figura D.2: Añadir un usuario existente

Figura D.3: Añadir clase existente

APÉNDICE E. IMAGENES DE LOS RESULTADOS DE LA ENCUESTA DE VALORACIÓN

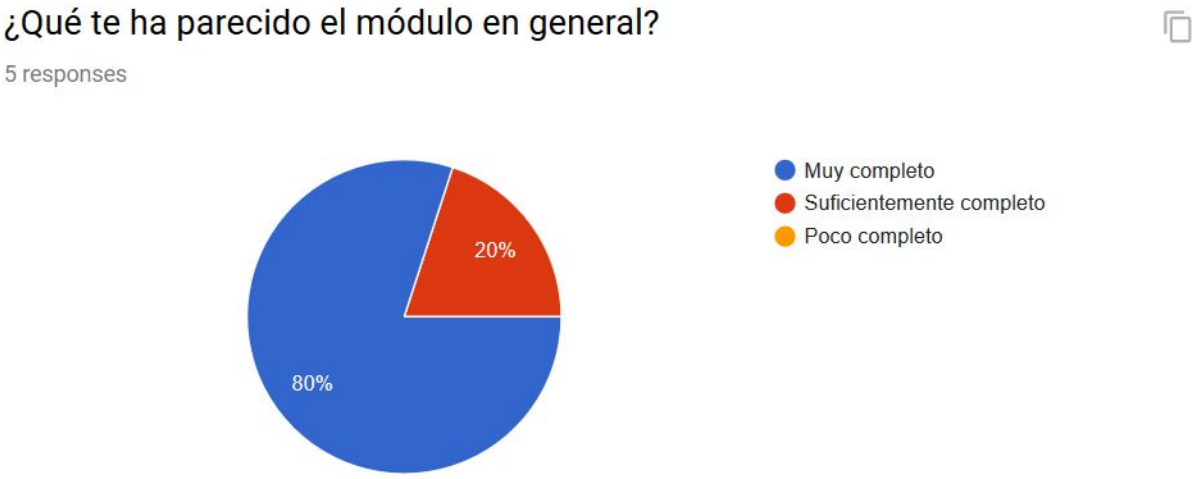


Figura E.1: Valoraciones generales del módulo

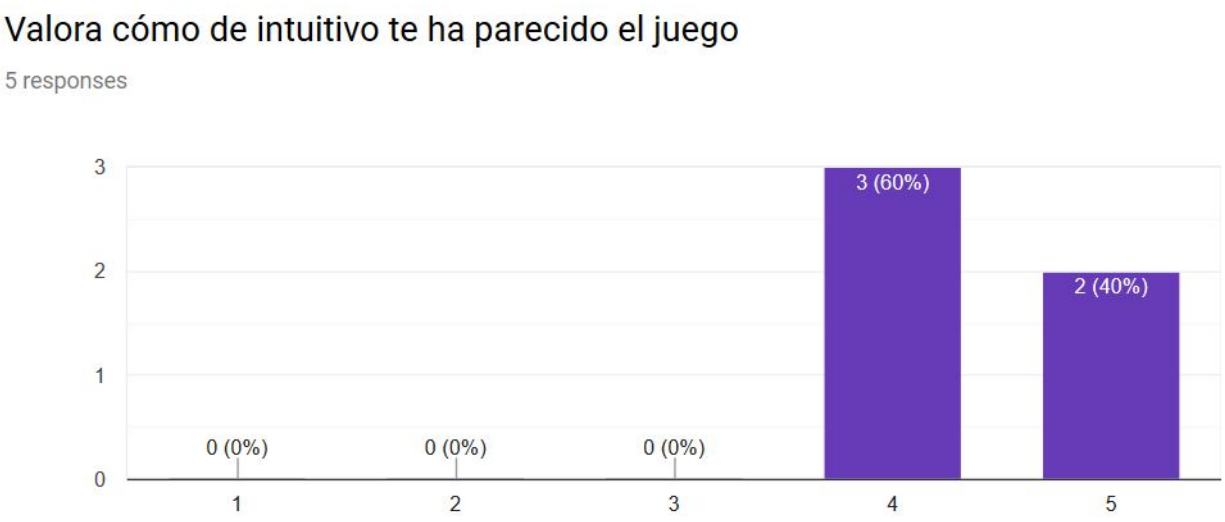


Figura E.2: Valoración de la intuitividad del módulo

Figura E.3: Elección de la función favorita del juego

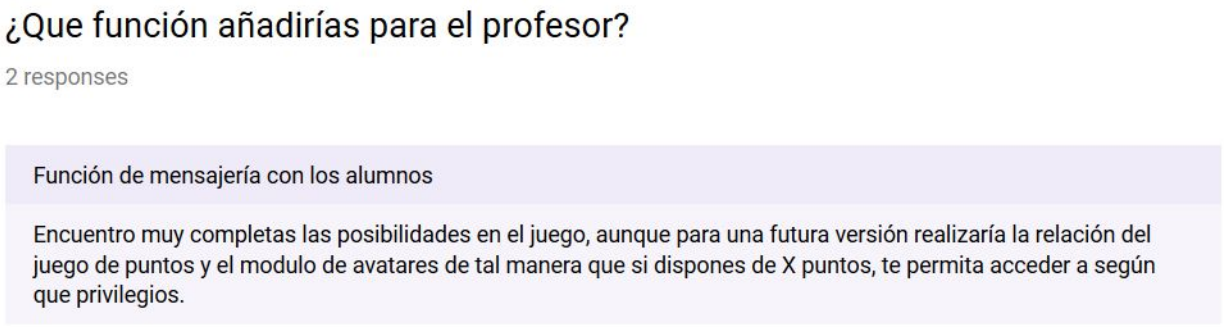


Figura E.4: Opiniones sobre funcionespara añadir al rol de profesor

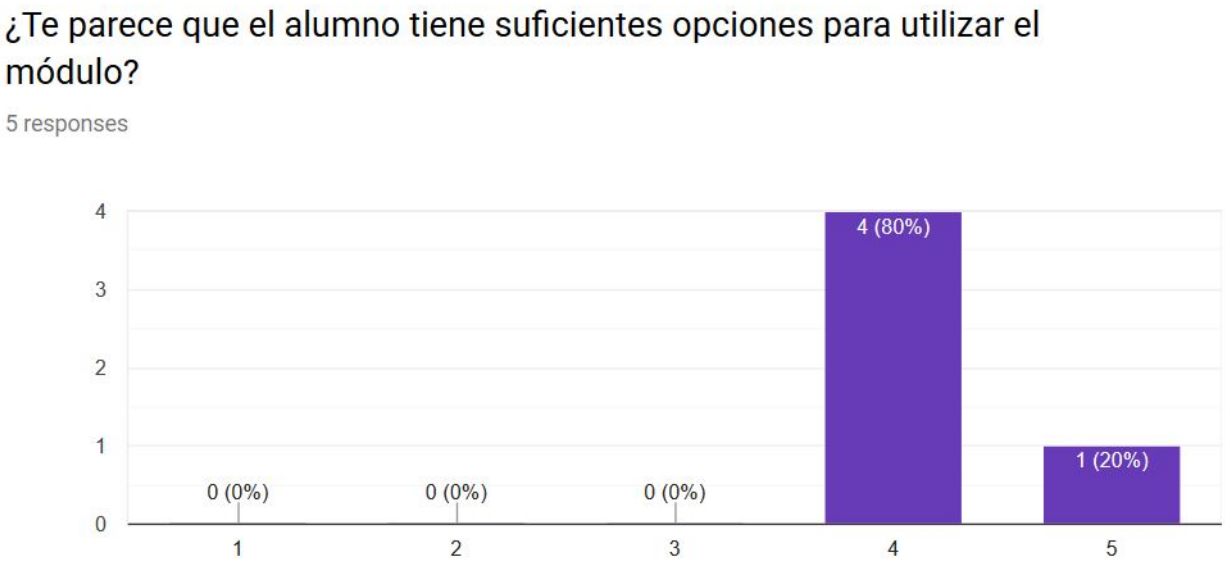


Figura E.5: Valoración de las funciones del alumno

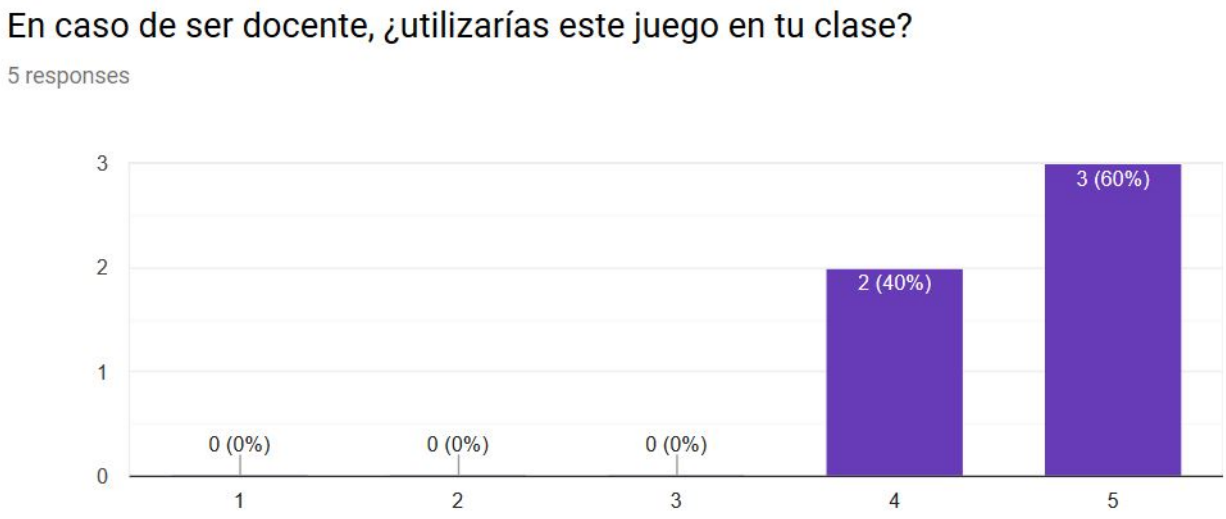


Figura E.6: Respuestas a “¿Utilizarías este juego en tu clase?”

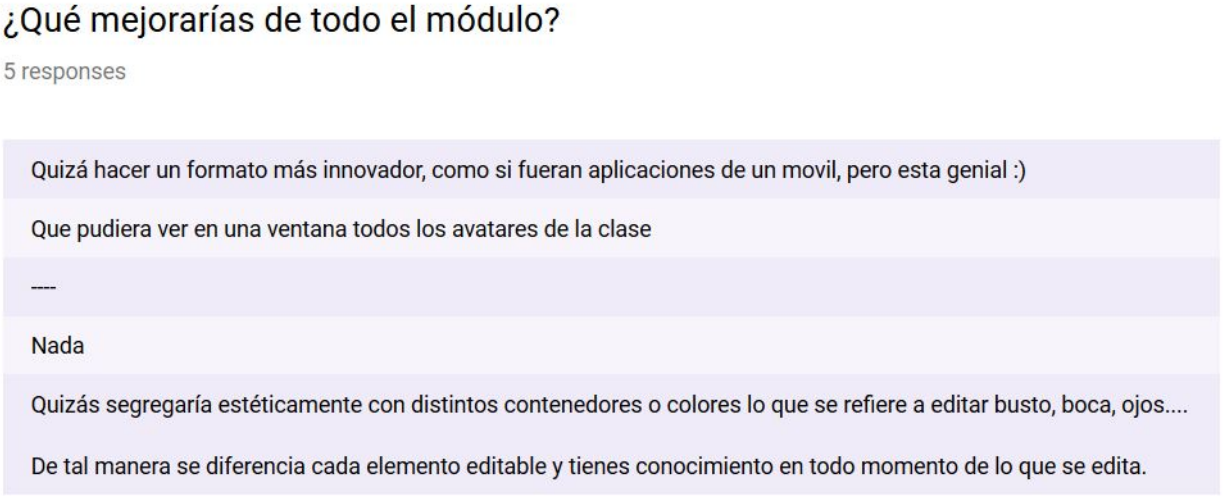


Figura E.7: Propuestas de mejora para el módulo de avatares

APÉNDICE F. CÓDIGO DE ARCHIVO DE SERVICIOS PARA LA BASE DE DATOS

```
import { Injectable } from '@angular/core';
import { Observable, of, from } from 'rxjs';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Persona } from './Persona';
import { Clase } from './Clase';
import { Imagen } from './Imagen';
import { Container } from './Container';
import { Img } from './Img';
import { Matricula } from './Matricula';
// Las siguientes librerías importadas son para poder realizar operaciones Http
import { Http, ResponseContentType, RequestOptions, Response, Headers } from '@angul



@Injectable({
  providedIn: 'root'
})
export class DbServiceService {

  // Creo un array en el que voy a guardar imágenes
  // que vendran de un contenedor concreto

  imagenesPelos: Img[] = new Array();
  imagenesOjos: Img[] =  new Array();
  imagenesComplementos: Img[] = new Array();
  idclase: string;
  NPersona: string;
  matricula: Matricula;

  // Declaro como string la URL de la BDD a la que me quiero conectar
  private APIUrl = 'http://localhost:3000/api/Personas';
  private APIClases = 'http://localhost:3000/api/Clases';
  private APIFotos = 'http://localhost:3000/api/imagenes';
  APIPermisos = 'http://localhost:3000/api/permisos/ArchivosTexto';


  // Inserto en el constructor el servicio Http para poder hacer las operaciones
  //necesarias
  constructor(private http: HttpClient,
              private http2: Http) { }

  // A partir de aquí declaro las operaciones que va a ofrecer este servicio
```

```
// La siguiente función lama a un observable de la lista de personas
// Por esto mismo hemos importado arriba la clase Observable

dameTodos(): Observable<Persona[]> {
  // La operacion get del protocolo http devuelve lo que tiene
  // entre "< >", en este caso una lista de personas.
  return this.http.get<Persona[]>(this.APIUrl);
}

dameFotosContainer(container: string): Observable<any[]> {

  return this.http.get<any[]>(this.APIFotos + '/' + container + '/files');

}

DameMatriculaAlumno(idAsignatura: string) {

  console.log( idAsignatura );

  return this.http.get<Matricula[]>('http://localhost:3000/api/matriculas?filter[w

}

PonMatricula(matricula: Matricula ): Observable<any> {

  return this.http.post<any>('http://localhost:3000/api/matriculas', matricula);
}

CuentaMatriculas(): Observable<any> {
  return this.http.get<any>('http://localhost:3000/api/matriculas/count');
}


Eliminar(nombre: string): Observable<any> {
  return this.http.delete<any>(this.APIUrl + '/' + nombre);
}

EliminarClase(idClase: string): Observable<Clase> {
  return this.http.delete<Clase>(this.APIClases + '/' + idClase);
}

EliminarMatricula(matricula: Matricula): Observable<any> {
  console.log('Voy a eliminar la matrícula de: ' + matricula.idAlumno);
  console.log('El id que elimino es: ' + matricula.id);
  return this.http.delete<any>('http://localhost:3000/api/matriculas/' + matricula
}

DamePersona(nombre: string): Observable<Persona> {
```

```
  console.log(nombre);
  return this.http.get<Persona>(this.APIUrl + '/' + nombre);
  console.log(Persona);
}

SetIdClase(clase: string) {
  this.idclase = clase;
}

SetNombrePersona(npersona: string) {
  this.NPersona = npersona;
}

SetMatricula(matri: Matricula) {
  this.matricula = matri;
}


ReturnNombrePersona() {
  return this.NPersona;
}

ReturnIdClase() {
  return this.idclase;
}

ReturnMatri() {
  return this.matricula;
}


DameClase(idclase: string): Observable<Clase> {
  this.idclase = idclase;
  console.log('Te doy los datos de: ' + idclase);
  return this.http.get<Clase>(this.APIClases + '/' + idclase);
}

DameClases(): Observable<Clase[]> {
  return this.http.get<Clase[]>(this.APIClases);
}


ColocoPelo(elementoP: string) {

  console.log('Me llega un: ' + elementoP);
  if (elementoP === undefined || elementoP === '') {
    console.log('No has seleccionado ningún pelo');
  } else {
```

```
    console.log('Entro a colocar');

    var imagen = document.createElement('img');

    imagen.style.position = 'absolute';
    // imagen.style.zIndex = '1';
    imagen.style.left = '0px';
    imagen.style.top = '0px';
    imagen.src = elementoP;
    document.getElementById('avatar').appendChild(imagen);
  }


}

CambiaEstadoJuego(clase: Clase) {

  if (clase.avatares === true) {
    clase.avatares = false;
  } else {
    clase.avatares = true;
  }
  console.log('Ahora el estado es este:' + clase.avatares);
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);
}

CreaClase(clase: Clase): Observable<Clase> {
  // clase.admin = admin;
  return this.http.post<Clase>(this.APIClases + '/' + clase.id, clase);
}

PonPass(alumno: Persona, nuevopass: string): Observable<any> {
  alumno.pass = nuevopass;
  return this.http.put<any>(this.APIUrl + '/' + alumno.nombre, alumno);
}

// Añadir una persona a la BBDD es una operación post
// requiere la URL y en este caso la persona que debemos añadir

// Esta función también devulve un observable de cualquier tipo
// Va a ser un método usado únicamente por el profesor

PonPersona(persona: Persona): Observable<any> {
  return this.http.post<any>(this.APIUrl, persona);
}

// Creo una función que sirve sólamente para la carga de archivos
// como imágenes o archivos de texto, servirá para que el profesor
```

```
// cargue el material en la galería y los archivos de texto que irán ligados
// a los permisos 1, 2 y 3 para construir el avatar



DameContenedores(): Observable<any[]> {
  return this.http.get<any[]>(this.APIFotos);
}



DameFoto(idconte: string) {
  // this.VaciarArray();
  var i;
  this.http.get<any>(this.APIFotos + '/' + idconte + '/files')
  .subscribe( fotoscontainer => { console.log('Tengo los archivos del container: '
                                  // this.nombreslogos = fotoscontainer;
                                  for (i = 0; i < fotoscontainer.length; i++) {
                                    console.log(fotoscontainer[i].name);
                                    this.http2.get(this.APIFotos + '/' + idconte +
                                    {responseType: ResponseContentType.Blob} )
                                    .subscribe(response => {
                                                              console.log(response);
                                                              this.CargarLogos(respo
                                  }
                                  console.log('Ye he acabado');
                                 });
}

DameLogosPelo() {
  // console.log('Ya he rellenado el modelo');
  return this.imagenesPelos;
}

DameLogosOjos() {
  return this.imagenesOjos;
}

DameLogosComp() {
  return this.imagenesComplementos;
}

CargarLogos(response: Response, idconte: string) {


  const blob = new Blob([response.blob()], {type: 'image/jpg'});

  const reader = new FileReader();
```

```
  reader.addEventListener('load', () => {

    if (idconte === 'Pelos') {
      // console.log('No sé si entra');
      this.imagenesPelos.push(reader.result.toString());
    }

    if (idconte === 'Ojos') {
      this.imagenesOjos.push(reader.result.toString());
      console.log(this.imagenesOjos);
    }

    if (idconte === 'Complementos') {
      this.imagenesComplementos.push(reader.result.toString());
      console.log(this.imagenesComplementos);
    }

  }, false);

  if (blob) {
    reader.readAsDataURL(blob);
  }
}

GuardarPelo(matricula: Matricula, p1: string): Observable<Matricula> {

  console.log(p1);
  matricula.URLpelo = p1;
  console.log('He llegado al servicio');
  console.log('http://localhost:3000/api/matriculas/' +  matricula.id);

  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i
}

GuardarOjos(matricula: Matricula, p2: string): Observable<Matricula> {

  matricula.URLojos = p2;

  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i
}

GuardarComp(matricula: Matricula, p3: string): Observable<Matricula> {

  matricula.URLcomplemento = p3;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i
}

GuardarBoca(matricula: Matricula, p4: string): Observable<Matricula> {
```

```
  matricula.URLboca = p4;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i
}

GuardarP1(matricula: Matricula, valorp1: boolean): Observable<Matricula> {
  matricula.pelo = valorp1;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i
}

GuardarP2(matricula: Matricula, valorp2: any): Observable<Matricula> {

  matricula.ojos = valorp2;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

}

GuardarP3(matricula: Matricula, valorp3: any): Observable<Matricula> {

  matricula.complemento = valorp3;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

}

GuardarP4(matricula: Matricula, valorp4: any): Observable<Matricula> {

  matricula.verclase = valorp4;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

}

MuestraFicheros(): Observable<Container[]> {
  return this.http.get<Container[]>(this.APIPermisos + '/files');
}

GuardaFicheroPermiso1(clase: Clase, archivo: string): Observable<any> {
  clase.p1 = archivo;
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);
}

GuardaFicheroPermiso2(clase: Clase, archivo: string): Observable<any> {
  clase.p2 = archivo;
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);
}

GuardaFicheroPermiso3(clase: Clase, archivo: string): Observable<any> {
  clase.p3 = archivo;
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);
```

```
}

GuardaFicheroPermiso4(clase: Clase, archivo: string): Observable<any> {
  clase.p4 = archivo;
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);
}

// Añadir familias de avatares va a ser tan simple como subir imágenes a la carpet
// Cada imágen tendrá una nomenclatura así: "nombrefamilia_pelo1" por ejemplo.

GuardaFamilia(familia: string, clase: Clase, archivobusto: string): Observable<Cla

  clase.familia = familia;
  clase.busto = archivobusto;
  return this.http.put<any>(this.APIClases + '/' + clase.id, clase);

}

EliminarFoto(galeria: string, idfoto: string): Observable<any> {

  console.log('Voy a eliminar la foto de: ' + galeria);
  console.log('La foto es: ' + idfoto);
  return this.http.delete<any>(this.APIFotos + '/' + galeria + '/files/' + idfoto)

}

// Este método borra el archivo que haya
// para poner ojos en el avatar
ResetOjos(matricula: Matricula): Observable<any> {
  var reset = '';
  console.log('Reseteando ojos');
  matricula.URLojos = reset;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

}

ResetPelo(matricula: Matricula): Observable<any> {
  var reset = '';
  console.log('Reseteando pelo');
  matricula.URLpelo = reset;
  return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

}

ResetComp(matricula: Matricula): Observable<any> {
  var reset = '';
  console.log('Reseteando complemento');
  matricula.URLcomplemento = reset;
```

```
    return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

  }

  ResetBoca(matricula: Matricula): Observable<any> {
    var reset = '';
    console.log('Reseteando ojos');
    matricula.URLboca = reset;
    return this.http.put<any>('http://localhost:3000/api/matriculas/' +  matricula.i

  }


}
```