Trabajo de Fin de Grado

Escuela Superior de Ingeniería y Tecnología

Sección de Ingeniería Informática

Plataforma de ludificación de un simulador didáctico de arquitectura de computadores

Gamification framework for a didactic simulator of computer architecture

Antonio Jesús López Garnier

La Laguna, 30 de junio de 2018

D. **Iván Castilla Rodríguez**, con N.I.F. 78.565.451-G profesor Ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Adrián Abreu González**, con N.I.F. 54.111.250-R profesional externo a la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Plataforma de ludificación de un simulador didáctico de arquitectura de computadores"

ha sido realizada bajo su dirección por D. **Antonio Jesús López Garnier**,

con N.I.F. 78.855.494-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 30 de junio de 2018

Agradecimientos

A mi tutor, Iván Castilla, por su predisposición para ayudar en todo y la confianza depositada.

A mis amigos, por ayudarme siempre que estaba perdido.

A Irene, por acompañarme todos los días y darme fuerzas para continuar.

Licencia

Resumen

A veces olvidamos que la mejor manera de transmitir y adquirir conocimientos es a través de la emoción. Realizar actividades diferentes que conecten con la persona y consigan captar su interés es la clave para aprender rápidamente.

En este proyecto se realiza el desarrollo de una plataforma de ludificación para apoyar el aprendizaje de los alumnos en el campo de las arquitecturas que explotan el paralelismo a nivel de instrucción y, más concretamente, sobre las máquinas superescalares.

El objetivo es presentar una actividad atractiva en la que se pueda llevar a cabo una competición sana entre alumnos; potenciando el esfuerzo y la concentración; y aprovechando el juego como medio para alcanzar un aprendizaje.

El resultado ha sido una aplicación web que se integra con un simulador de arquitecturas previamente desarrollado y que complementa a este simulador con actividades lúdicas para aprender el funcionamiento de las máquinas superescalares.

Palabras clave: Ludificación, Arquitectura de computadores, Superescalar, Simulador, plataforma web, Javascript, React, Redux

Abstract

Sometimes we forget that emotion is the best way to transmit and acquire knowledge. The key to learn quickly is making activities that connect with the person and capture their attention.

In this project, I have developed a gamification platform to support student learning in the field of instruction level parallelism architectures and, more specifically, about superscalar machines.

The objective is to present an attractive activity in which students can carry out a healthy competition; improving effort and concentration; and using the game as a way to learn.

The result is the integration of a previously developed simulator into a web platform that compliments the simulator with fun activities intended to learn about superscalar machines.

Keywords: Gamification, Computer arquietcture, Superscalar, Simulator, web platform, Javascript, React, Redux

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Introducción

El uso generalizado de las nuevas tecnologías, tales como internet, redes sociales y móviles, han producido un gran impacto en el sistema educativo, pero sobre todo en el ámbito universitario. Las Tecnologías de la Información y la Comunicación (TIC) han hecho posible tanto una mejora en la comunicación como en la implementación de nuevos sistemas de información, útiles para el aprendizaje y la tutorización. Desde 2010 en adelante, ha emergido una nueva tendencia conocida como ludificación. El objetivo es incrementar la experiencia de usuario, eficacia y efectividad del aprendizaje, además de la motivación de los alumnos. Esta tendencia ha evolucionado hasta convertirse en una gran herramienta de apoyo en las aulas, pues cuenta con innumerables ventajas como estimular, enseñar y reforzar conocimientos, pero también fomentar habilidades y aptitudes como son la resolución de problemas, el trabajo en equipo y la comunicación. Así se hace posible motivar a los participantes con el placer de jugar y participar con otros.

1.2 Ludificación

Durante la década de 1980 comenzaron los primeros estudios de autores como Loftus & Loftus (1983) [1] que relacionaban la motivación intrínseca de los juegos, con la capacidad de mantener motivados a los individuos sin la necesidad de recompensas extras.

A lo largo de los años, el concepto de ludificación (gamification en inglés) se ha ido consolidando, convirtiéndose en una teoría que se basa en el diseño de juegos y que ha sido objeto de estudio de numerosas disciplinas (ciencias sociales, humanidades e informática).

De esta forma, el concepto definido por Deterding et al., (2011) [2] hace referencia a la ludificación como el uso de elementos de diseño de juegos en contextos ajenos a éste. Es decir, la ludificación propone hacer uso del pensamiento y dinámicas de los juegos con el fin de aumentar el compromiso de los individuos; además de estimular la participación produciendo una mejora en los resultados. A través del empleo de

estas mecánicas y elementos de juego en entornos considerados no lúdicos, se consigue modificar el comportamiento de las personas. Esto es posible al convertir las tareas en una actividad mucho más atractiva, potenciando el esfuerzo y concentración en aras de conseguir unos objetivos concretos por medio de juegos cuya finalidad es aprender. Como numerosas investigaciones han reconocido, el enfoque basado en juegos resulta altamente eficaz, pues repercute positivamente en las personas ya que facilita el proceso de aprendizaje, es más centrado en el individuo, divertido e interesante. Estudios como el de Papastergiou (2009) [3] han profundizado en las razones que hacen que los juegos favorezcan un entorno de aprendizaje más eficaz y eficiente: su éxito se debe a que admiten un aprendizaje mucho más activo, es experimental, basado en problemas que promueven el uso del conocimiento previo adquirido y que, a su vez, fomentan la creatividad.

Con el fin de comprender el poder y uso de la ludificación como sistema de aprendizaje, es necesario conocer aquellos elementos del juego que pueden extraerse y aplicarse en este contexto. En esta línea, Werbach y Hunter (2012) [4] distinguen entre:

- **Componentes**: rankings y puntos, logros, avatares y niveles.
- **Mecánicas**: retos, competición, cooperación, feedback y recompensas.
- **Dinámicas**: limitaciones, emociones, narrativa, progresión y relaciones.

Estos elementos mencionados anteriormente contribuyen a mejorar el "engagement" (compromiso) vde los participantes en aquellas actividades en las que se empleen técnicas de ludificación.

De esta forma, en una sociedad cambiante, en la que predomina la tecnología además de una cultura del entretenimiento, es necesario que la ludificación sea flexible y capaz de adaptarse a cada entorno para fomentar un aprendizaje constante en el tiempo.

1.2.1 Ludificación en aulas

La ludificación se ha convertido en una estrategia pedagógica como método para superar los nuevos retos a los que se enfrenta el sector de la educación en un mundo que en la actualidad se encuentra altamente digitalizado. En consecuencia, los docentes se enfrentan a un gran desafío a la hora de motivar a los estudiantes. En este contexto el uso de juegos como herramienta para fomentar el aprendizaje se ha convertido en un planteamiento conveniente debido a su capacidad para reforzar y enseñar conocimientos e impulsar la resolución de problemas. Así, el sistema educativo ha experimentado la transmisión de conocimiento de la mano de las TIC con la evolución de las tecnologías y comunicación en la interacción entre sujetos.

Esto ha dado lugar a la aplicación de estos elementos con el fin de crear componentes que resulten más interactivos y atractivos en situaciones que requieren una mayor formalidad, como es la educación. Por otro lado, la introducción de las tecnologías plantea un cambio no solo de escenario (en cuanto a la educación se refiere) sino también de las características de los sujetos de la educación (Prensky, 2007) [5]. Surge por lo tanto un campo relativamente nuevo y que se encuentra hoy en día en auge en lo que se refiere al ámbito de la docencia en general.

La ludificación en las aulas aplica esas mecánicas y elementos de juego mencionadas anteriormente con el fin de mejorar el rendimiento de los alumnos, además de servir como herramienta de apoyo para la adquisición y refuerzo del conocimiento. Por otra parte, promueve el pensamiento lógico-matemático y crítico, el desarrollo de habilidades personales y sociales, habilidades del lenguaje, comunicación y colaboración, capacidades creativas y de resolución de problemas.

Además, Pesare et al., (2016) detallaron que la ludificación apoya los siguientes principios pedagógicos:

- **Individualización**: se refiere al hecho de que el nivel se adapta en función de las habilidades del jugador.
- **Feedback**: se proporciona "feedback" o retroalimentación instantánea y contextualizada de acuerdo con la sesión del juego.
- **Social**: el juego contempla varias modalidades de juego, entre ellas multijugador o social, mejorando la comunicación.
- **Aprendizaje activo**: el juego involucra al jugador en el descubrimiento activo.
- **Motivación**: los jugadores persiguen un objetivo.
- **Superación**: los jugadores son desafiados gradualmente.
- **Transferencia**: Se fomenta la capacidad de transmitir el aprendizaje desde u contexto de juego a un contexto real.
- **Evaluación**: el jugador puede evaluar el conocimiento o habilidad adquirida.

La utilización de esta estrategia puede variar de acuerdo con los objetivos de cada docente, ya que se puede optar por ludificar o bien toda la asignatura o sólo aquellas partes del curso que se considere no funcionan correctamente. Admite versatilidad y flexibilidad, pues el profesor puede implantarla total o parcialmente.

A continuación, se detallas los pilares fundamentales de la ludificación educativa:

- **Diversión**: Resulta más entretenido estudiar cuando se añaden elementos de juego.
- **Significado**: Se reconocen los obstáculos/retos superados y se fomenta el refuerzo positivo de los alumnos.
- **Progresividad**: El juego al resultar atractivo influye en la manera en la que los alumnos ven las tareas, mejoran su organización, pueden ver como progresan y obtener retroalimentación.
- **Autorregulación**: Se observan fortalezas y aquellas áreas en las que los alumnos deben mejorar, por lo que permite sobreponerlas antes de la evaluación final. La ludificación además se adapta a los ritmos de aprendizaje.

Con los fundamentos mencionados anteriormente se pueden obtener conclusiones cruciales sobre el rendimiento académico y percepción de los estudiantes sobre la asignatura. Además de aumentar la satisfacción del alumnado, se consigue que el aprendizaje traspase el aula para ser constante. Así, la educación a través del uso de técnicas de gamificación o ludificación se ha convertido en una estrategia de futuro.

1.3 Motivación para el trabajo

Como se ha señalado, el uso de técnicas lúdicas en el ámbito educativo ha ido ganando fuerza con la evolución de las tecnologías. Se ha convertido en un campo muy interesante para realizar proyectos grandes y apoyar la educación digital.

Actualmente está siendo utilizado un simulador de arquitecturas que explotan el paralelismo a nivel de instrucción para la docencia en la asignatura de Arquitectura de Computadores, que ayuda en gran medida a comprender el funcionamiento de dichas máquinas.

Es por esto qué se ha querido construir una plataforma que envuelva este simulador y futuros proyectos, que pueda proporcionar una forma de aprender diferente y divertida, y que sirva de base o referente para futuras ampliaciones o mejoras.

Capítulo 2
Antecedentes

2.1 SIMDE

SIMDE es un simulador didáctico para la enseñanza de la asignatura arquitectura de computadores, desarrollado en 2004 por el estudiante Iván Castilla Rodríguez, actualmente profesor y tutor de este proyecto [6]. Se trata de una herramienta capaz de simular arquitecturas que tratan de aprovechar el paralelismo a nivel de instrucción (ILP) mediante técnicas tanto de planificación dinámica como planificación estática de instrucciones. Es una herramienta con una perspectiva puramente educativa, que rehúye las complejidades tecnológicas y se centra en los conceptos básicos para comprender el ILP. Utiliza un repertorio de instrucciones similar al MIPS, pero considerablemente simplificado, que se puede simular tanto sobre una máquina superescalar, como sobre una máquina de planificación estática mediante instrucciones muy largas (Very Long Instruction Word – VLIW)

Se trataba de un desarrollo usando C++ (más específicamente, la plataforma C++ Builder de Borland) que cumplía los requisitos como aplicación de escritorio cuando fue desarrollada pero que ha quedado tecnológicamente obsoleta, lo que dificulta su mantenibilidad y el planteamiento de ampliaciones o integración con otras aplicaciones.

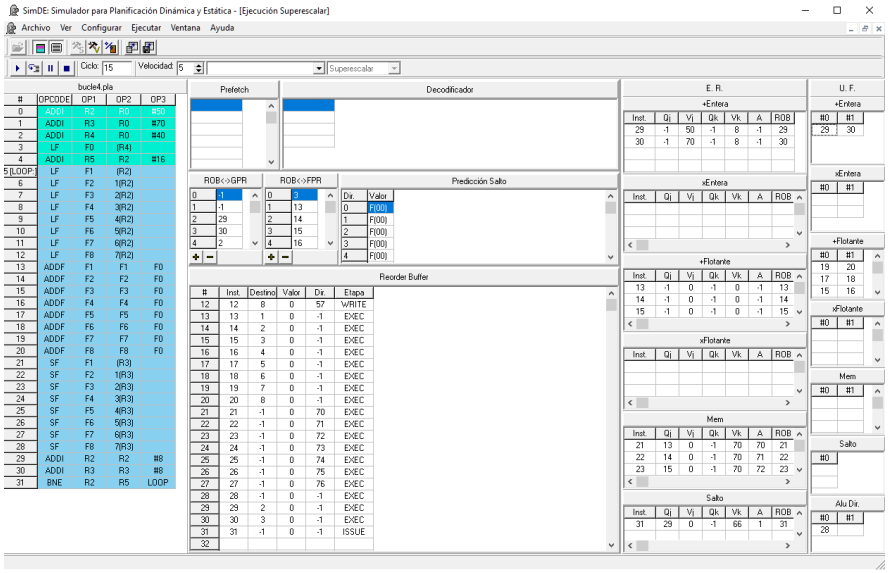

Figura 2.1: SIMDE original

2.2 SIMDE Web

Con el doble objetivo de actualizar SIMDE y permitir su ampliación y su interacción con otras plataformas y herramientas, se desarrolló una nueva versión de la aplicación en 2017, como Trabajo de Fin de Grado de Adrián Abreu González, un alumno de la Universidad de La Laguna. [7]

Esta nueva versión es una aplicación web que está desarrollada con tecnologías actuales (Typescript, Redux…), y disponible en un repositorio github (https://github.com/etsiiull/SIMDE). Se trata de una versión que reproduce las funcionalidades del simulador original para la máquina superescalar pero que tiene aún en desarrollo la simulación de la máquina VLIW. Además, incorpora nuevas utilidades como la simulación en "batch" o la vuelta atrás en el estado de la máquina.

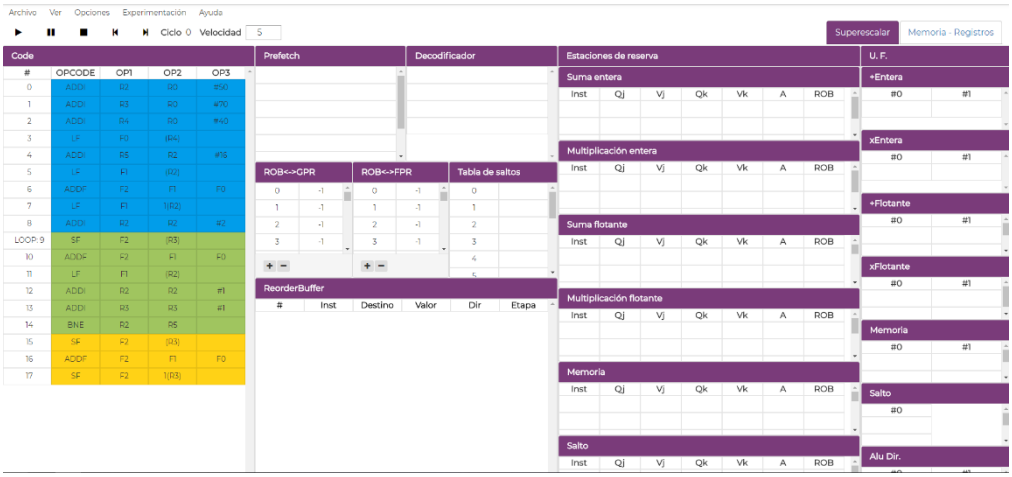

Figura 2.2: SIMDE Web

2.3 Uso actual de la herramienta en el aula

El uso actual de la herramienta en las clases prácticas de arquitectura de computadores puede ser definido en los siguientes pasos:

1. Los alumnos trabajan en grupos. Se plantea un problema como, por ejemplo, la ordenación de un vector que se encuentra en memoria.
2. Se pide implementar en el código pseudo MIPS una solución y comprobarlo con las instancias que se facilitan.
3. Utilizando el simulador de la máquina superescalar, se comprueba la influencia de diferentes parámetros, tratando de reducir al máximo el número de ciclos de ejecución del programa desarrollado.
4. Se realizan tareas similares con la máquina VLIW.
5. Finalmente, se prepara una presentación con los resultados obtenidos.

El profesorado de la asignatura ha observado que, de forma natural, los alumnos suelen plantearse a veces la actividad como una especie de competición, fijándose en quién reduce más el número de ciclos. De esta percepción surge la idea de introducir el simulador en un entorno ludificado.

Capítulo 3
Objetivos y fases

3.1 Objetivo

La idea principal del proyecto era el desarrollo de actividades de ludificación, utilizando como medio el simulador didáctico SIMDE [8]. Lograrlo requería de una base sólida para implementar elementos lúdicos. Es por ello qué se definieron los siguientes objetivos:

- Implementar una interfaz de usuario.
- Implementar un método de control y gestión de usuarios.
- Implementar la sincronización entre aplicaciones.
- Integrar el simulador de máquinas superescalares.

3.2 Fases

El desarrollo del proyecto se ha llevado a cabo en cinco fases:

1. **Documentación**: El sistema que se pretendía implementar requería de mucho conocimiento acerca de las tecnologías empleadas. Por lo que la primera fase del proyecto fue documentarse.
2. **Arquitectura interna de la plataforma**:
    a. **Implementación de registro de usuarios:** Es decir, identificar usuarios y mantener una sesión activa.
    b. **Implementación de un método de control de estado:** Una forma de controlar el estado de la aplicación para facilitar el tratamiento de datos y la comunicación entre componentes.
    c. **Implementar la sincronización y actualización de la aplicación:** Sincronización entre aplicaciones cliente y actualizaciones de datos en tiempo real.
3. **Desarrollo de la interfaz gráfica.**
4. **Integración del simulador con la plataforma**: Permitir una conexión entre el sistema de juegos implementado en la plataforma y el simulador de máquinas superescalares.
5. **Testeo**: Comprobación del correcto funcionamiento de la plataforma.

Capítulo 4
Tecnologías

En este capítulo se expondrán las diferentes tecnologías utilizadas durante el desarrollo del proyecto.

4.1 Lenguaje para el desarrollo de la aplicación

Gracias a la evolución de las tecnologías web, tenemos a nuestro alcance una serie de herramientas que nos permiten desarrollar nuestros proyectos de una forma muy cómoda y priorizando nuestros gustos sobre lenguajes de programación. Es por esto qué se ha valorado realizar el proyecto en dos lenguajes: Typescript y Javascript.

4.1.1 **Typescript**

Typescript es un lenguaje libre y de código abierto desarrollado por Microsoft. Es una extensión de Javascript que añade el tipado estático y objetos basados en clase como podemos ver en la figura 4.1. El antecesor a este proyecto fue realizado utilizando este lenguaje ya que aportaba una serie de características que facilitaban el desarrollo de toda la lógica. Una de ellas es la comprobación de tipos en tiempo de compilación. Además, este tipado estático no se ve reflejado en el código final, ya que typescript es transpilado a ECMAScript5 que es el lenguaje que entienden la mayoría de los navegadores actuales. [9]

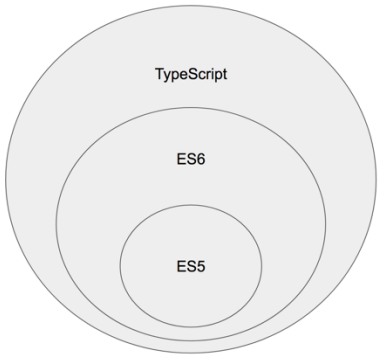

Figura 4.1: Typescript - Superconjunto de Javascript

4.1.2 **Javascript**

Javascript es el lenguaje de programación que actualmente se utiliza en la mayoría de las páginas y aplicaciones web. Ofrece mucha flexibilidad a la hora de desarrollar y actualmente podemos utilizar funcionalidades que aún no han sido incluidas en el

estándar pero que lo estarán muy pronto gracias a compiladores externos de Javascript. [10]

Sin embargo, el principal motivo por el que ha sido elegido cómo lenguaje de desarrollo para este proyecto es la extensa documentación encontrada con relación a las tecnologías empleadas. Y, por otra parte, el tipado estático no fue considerado una gran ventaja en este desarrollo ya que React proporciona otras herramientas para la validación de datos.

4.2 Tecnologías para la gestión y almacenamiento de datos

4.2.1 Firebase

Firebase es una plataforma de desarrollo en la nube creada por Google. Esta plataforma nos provee de una API para guardar y sincronizar datos en tiempo real.

Es una plataforma que proporciona muchos servicios para analizar, desarrollar, escalar y monetizar una aplicación. Sin embargo, para el desarrollo de este proyecto se ha elegido por dos servicios clave: Firebase Authentication y Cloud Firestore.

La extensa y ordenada documentación existente, y su compatibilidad con todas las tecnologías empleadas en este proyecto, hacen de Firebase el mejor candidato para la gestión de usuarios y almacenamiento de datos.

4.2.2 Firebase Authentication

Este servicio de Firebase nos provee un sistema de registro de usuarios muy completo y fácil de implementar e integrar con cualquier aplicación. Además, permite la comunicación con otros servicios de interés como Cloud Firestore.

El registro en nuestra aplicación utilizando este sistema, nos permite obtener información del usuario como su nombre y apellidos, email, foto, fecha de primer registro, fecha de última conexión, UID y otros datos más. Este UID es generado automáticamente y solo cambia si el usuario es borrado manualmente del sistema de registro de Firebase Authentication. [11]

También resulta útil para determinar la persistencia de la autenticación del usuario. Es decir, mantener la sesión activa incluso después de cerrar el navegador o, por el contrario, terminar la sesión.

La autenticación se puede realizar con distintos proveedores, en este proyecto se ha optado por utilizar Google. En la figura 4.2 podemos ver un diagrama sencillo de un registro a través de este proveedor.

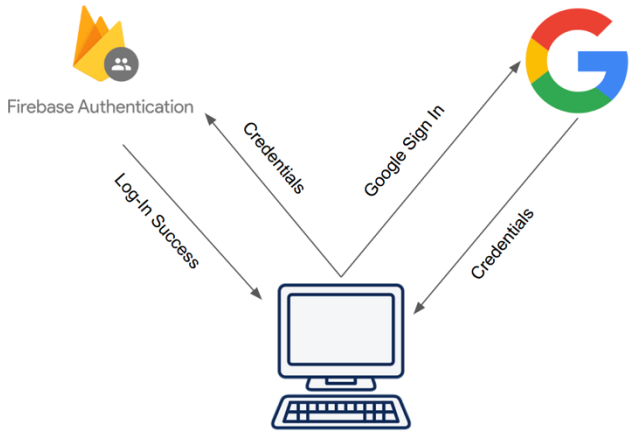

Figura 4.2: Autenticación con Google Sign In en Firebase

Todo esto nos proporciona unas herramientas ideales para manejar la entrada de usuarios en la plataforma.

4.2.3 **Firebase Cloud Firestore**

Este servicio pone a nuestro alcance una base de datos NoSQL muy flexible, escalable y que permite mantener nuestros datos sincronizados en tiempo real entre aplicaciones cliente. [12]

Los datos son guardados en estructuras jerárquicas. Se almacenan los datos en documentos y éstos son organizados en colecciones. Y estos documentos a su vez, pueden contener objetos anidados o subcolecciones. Ver figura 4.3.

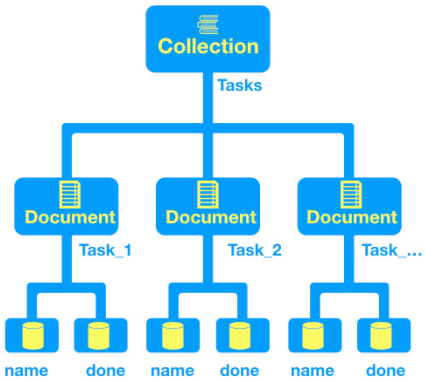

Figura 4.3: Estructura de una colección en Cloud Firestore [13]

Cloud Firestore te permite realizar consultas expresivas en las que puedes filtrar, ordenar, obtener uno o más documentos, recuperar colecciones enteras, etc.

Sin embargo, las principales características por la que se decidió utilizar este servicio son:

1. **Actualizaciones en tiempo real:** Proporciona agentes de escucha que se suscriben a los cambios en la base de datos y los comunican. De esta forma se consigue mantener las aplicaciones cliente sincronizadas y actualizadas.
2. **Reglas de seguridad:** Estas reglas, junto con Firebase Authentication, proporcionan acceso, autenticación, autorización y validación de datos. De esta forma podemos proteger el acceso a datos sensibles, por ejemplo, podemos hacer que solo un usuario autenticado con un determinado UID puede acceder a unos determinados datos.

Por todo esto y, de nuevo, gracias a la extensa documentación que nos proporciona Google, se ha elegido cómo base de datos para la plataforma.

4.2.4 **Local Storage**

Esta tecnología es una propiedad de HTML5 que nos permite almacenar datos en nuestro navegador de forma indefinida, incluso después de cerrarlo. [14]

Esto nos permite mantener una copia del estado de la aplicación, o una parte del estado para poder preservar los datos y obtener una carga inicial más rápida al partir de un estado anterior.

4.3 Tecnología para el desarrollo de la interfaz

En esta sección se describirán las tecnologías utilizadas para el desarrollo de la interfaz de usuario.

4.3.1 **React**

React es una librería Javascript de código abierto y creada por Facebook para el desarrollo de interfaces. [15]

Está basada en componentes, es decir, pequeños bloques reusables que incorporan cierta funcionalidad.

Es una librería muy potente que permite una actualización del DOM diferente a la convencional. Esto es gracias a que implementa el **Virtual DOM**, que copia los cambios en memoria para luego compararlos con el DOM y aplicar cambios exclusivamente en las partes que varían.

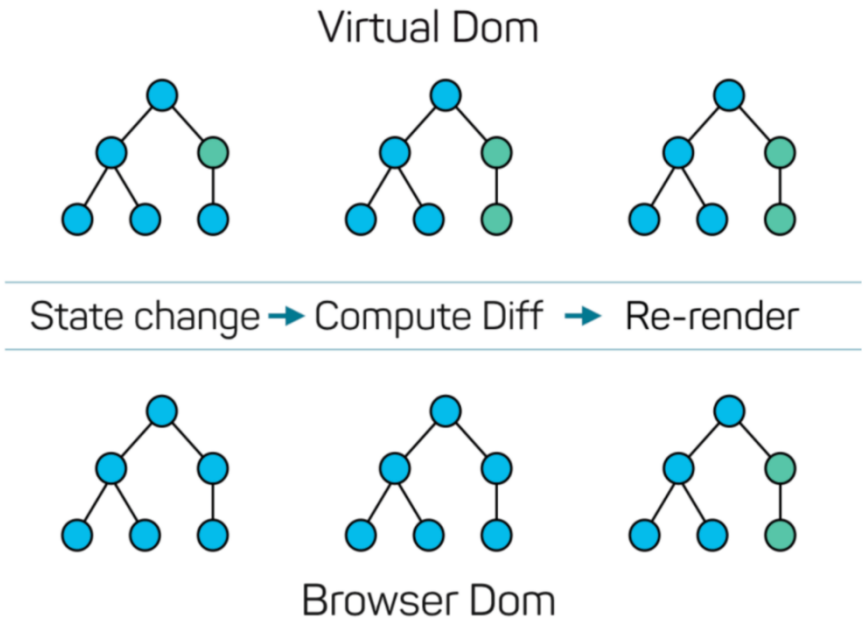

Figura 4.4: Funcionamiento del Virtual DOM de React [16]

El lenguaje que se utiliza para desarrollar componentes en React es un híbrido entre Javascript y HTML, llamado JSX.

Tiene una de las comunidades más activas actualmente y es utilizado por grandes empresas como Netflix o Airbnb. Incluso los propios empleados de estas empresas participan activamente en la comunidad de React. Por ejemplo, Dan Abramov, empleado de Facebook y creador de Redux, y Jay Pelphs, un trabajador de Netflix y creador de Redux-Observable invierten su tiempo resolviendo dudas en páginas como GitHub o StackOverflow.

Debido a todo lo anterior y a la compatiblidad con el antecedente SIMDE Web, React es la mejor opción para el desarrollo de la interfaz de la plataforma.

4.3.2 **Redux**

Redux se define como un contenedor predecible del estado de aplicaciones Javascript. Permite manejar y centralizar el estado entero de una aplicación y, además, puede ser utilizado con otras librerías. [17]

Se compone principalmente por cuatro elementos:

1. Un **Store** donde se almacena todo el estado de la aplicación en formato JSON.
2. Para cambiar el store es necesario lanzar **Actions**, que son simples objetos JSON que describen el cambio. Contienen dos propiedades principales que son el “type” o tipo de acción y el “payload” o la carga de datos que transporta la acción.

3. Finalmente, para aplicar el cambio al estado están los **Reducers**, funciones que toman como argumentos el estado actual y la acción, para devolver el nuevo estado de la aplicación.
4. Como concepto avanzado en Redux nos encontramos con los **Middlewares**, que proporcionan puntos de extensión entre el lanzamiento de una acción y su llegada al reducer. Se utilizan, generalmente, para realizar operaciones asíncronas como el registro de eventos, informe de fallos, llamadas a una API asíncrona, etc.

Podemos ver un diagrama completo en la figura 4.5, donde cabe indicar que el middleware puede capturar la acción y realizar operaciones asíncronas o simplemente ignorarla. Pero las acciones disparadas siempre llegan al reducer, aunque no realicen ningún cambio en el store o sean capturadas por un middleware. Podemos tener tantos middlewares como queramos.

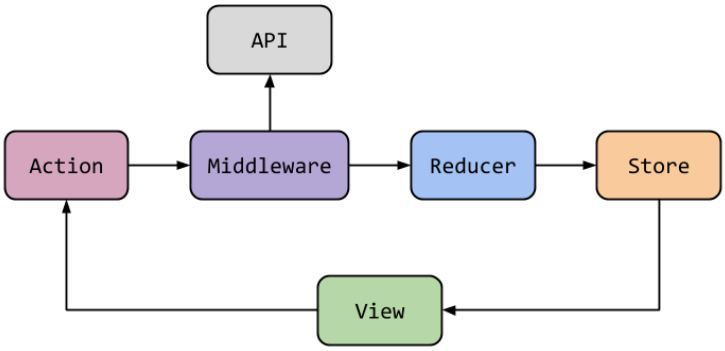

Figura 4.5: Diagrama de la estructura Redux [18]

Con un único estado de la aplicación podemos compartir datos entre componentes de una manera más limpia y ordenada. Y los cambios en un componente que afectan a más componentes se comunican a un único lugar, el Store. El resto de los componentes se actualizan al leer los cambios del store como vemos en la figura 4.6.

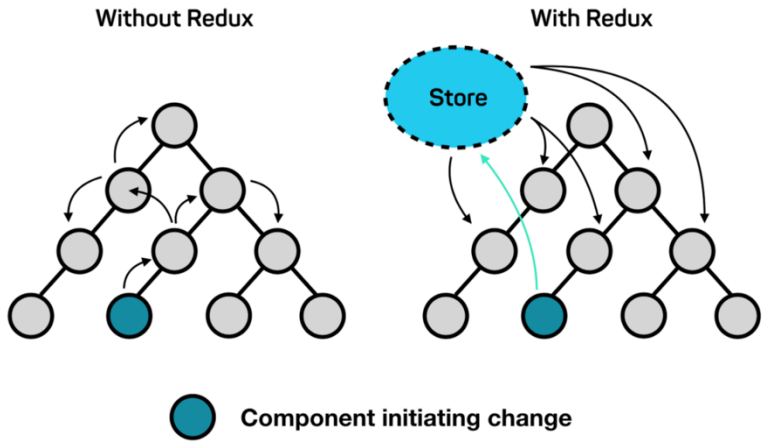

Figura 4.6: Estado centralizado usando Redux [19]

En este proyecto, el uso de Redux es fundamental para controlar el estado de la aplicación, así como el uso de middlewares para las llamadas a la base de datos y los agentes de escucha para las actualizaciones en tiempo real.

4.3.3 **ReactiveX y Redux-Observable**

ReactiveX [20] es una API de programación reactiva basada en el patrón observer que te permite utilizar observables. Los observables se pueden ver como una corriente, stream o array de datos cuyos valores llegan de forma asíncrona a lo largo del tiempo. Estos valores pueden ser desde números, hasta estructuras de datos e incluso eventos. Y nos permite manipular esos valores con las funciones que normalmente usaríamos en los arrays (map, filter, reduce, etc.).

Para que los observables puedan emitir estos valores tienen que estar suscritos a unos "observer" que son, simplemente, una colección de funciones que saben cómo tratar los datos de ese observable.

Para tratar con datos asíncronos, tenemos las promises y los callbacks. Sin embargo, utilizando los Observables podemos, por ejemplo, cancelar una petición asíncrona a una API, cosa que no podríamos hacer en una promise.

Redux-Observable [21] nos provee de una versión de middleware basado en ReactiveX que recibe una stream de acciones y devuelve otras acciones.

Veamos un ejemplo de uso:

- Tenemos un botón para enviar datos a una base de datos.
- Podemos retrasar ese envío un segundo, después de haber pulsado el botón.
- Si en ese intervalo de tiempo, se pulsara nuevamente el botón de enviar datos, se cancelaría la petición anterior y se procesaría la nueva.
- Con esto evitamos que un usuario envíe más de un dato cada segundo.

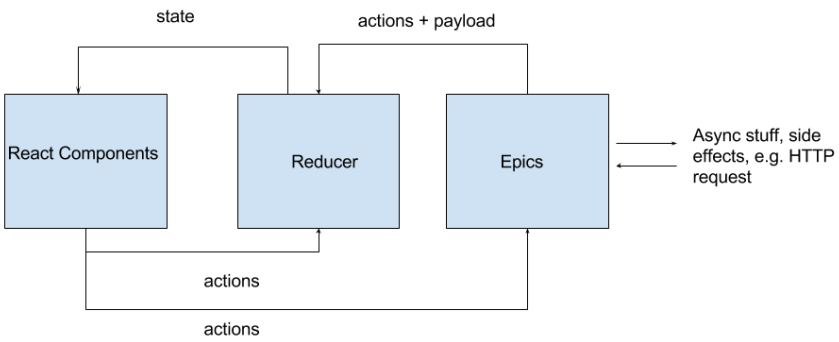

Figura 4.7: Redux-Observable usando Epics [22]

Gracias a esto evitamos hacer envíos innecesarios a la base de datos. De cara al futuro de la plataforma esto permitirá su escalabilidad y aumento de usuarios sin mucha sobrecarga en la base de datos.

4.3.4 **React Router**

Es una librería que nos permite realizar un enrutado dinámico en nuestra aplicación. [23]

Mientras el enrutado estático define las rutas antes de renderizar la aplicación, el dinámico lo hace mientras está renderizando.

Posee muchas características entre las cuales cabe destacar:

- Nos permite saber de forma rápida las vistas que está mostrando.
- Podemos utilizar el historial del navegador, es decir, sin estas rutas, al pulsar el botón de “retroceder” saldría de la aplicación en lugar de ir a una vista anterior.

Por estas características ha sido necesario incluirlo en este proyecto, con el fin de proteger la aplicación y mejorar la experiencia de usuario.

Figura 4.8: Logo React Router v4

4.3.5 **Otras**

En esta sección se describirán otras tecnologías utilizadas en el desarrollo de la interfaz cuya descripción es muy breve, aunque no menos importante. Se trata de:

- **Material UI:** Librería de componentes en React que implementa Material Design de Google. [24]
- **React-Quill:** Quill es un editor de texto enriquecido, personalizable y ligero. Y React-Quill es Quill como componente de React. [25]

4.4 Tecnologías para el empaquetado

Para el empaquetado de la aplicación se ha utilizado la herramienta create-react-app [26] que automáticamente genera un entorno de desarrollo que posteriormente puedes configurar, ya que te provee internamente de Babel y Webpack.

4.4.1 **Babel**

Esta herramienta nos permite transformar nuestro código en Javascript que utiliza funcionalidades nuevas que aún no se han incluido en el estándar, a código Javascript que los navegadores entienden. [27]

Funciona mediante plugins para indicarle las transformaciones que quieres hacer. Por ejemplo, podemos utilizar las Arrow Functions de ECMAScript6 y utilizar un plugin para que babel las transforme a funciones normales cuando se empaquete la aplicación.

La gran potencia de esta herramienta es que podemos escribir código que en un futuro estará estandarizado, y cuando eso ocurra simplemente eliminamos babel de nuestro proyecto. Ya no sería necesario transformar tu código.

Figura 4.9: Logo Babel JS

4.4.2 **Webpack**

Webpack [28] es un sistema de empaquetado de módulos para preparar la salida de una aplicación a producción. También realiza otras cosas como gestión de dependencias, ejecución de tareas, conversión de formatos, servidor de desarrollo, carga y uso de módulos.

En definitiva, podemos definir Webpack como una herramienta de compilación

que coloca en un grafo de dependencias a todos los elementos que forman parte de tu proyecto y luego las empaqueta todas en orden.

Su ciclo de funcionamiento puede ser resumido en:

- Parte de un punto de entrada en tu proyecto. Y para cada tipo de archivo existen unas reglas que activan unos loaders.
- Estos loaders se encargan, por ejemplo, de transformar un código de Typescript a Javascript. Es posible concatenarlos, por lo que podríamos realizar transformaciones sucesivas sobre un tipo de archivo.
- También se pueden añadir plugins para el desarrollo de tareas más complejas

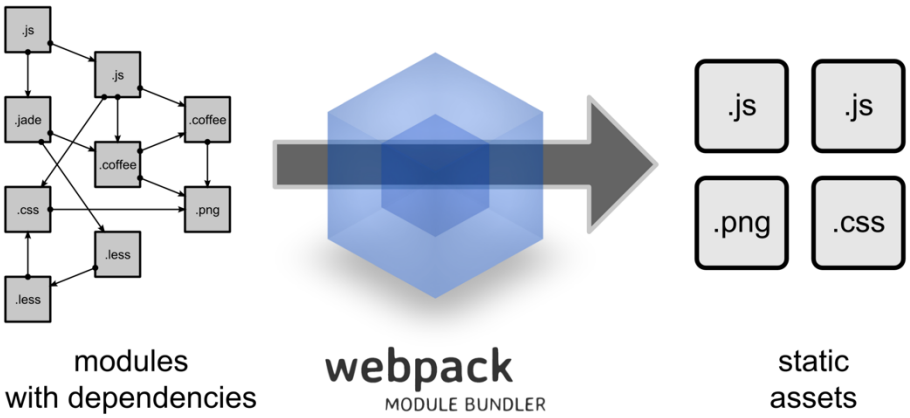

Figura 4.10: Funcionamiento general de Webpack [29]

Capítulo 5
Arquitectura del sistema

Para el desarrollo del proyecto se ha optado por una arquitectura web que se encuentra actualmente en auge, las SPA (Single Page Application) [30]. Habitualmente en las aplicaciones web convencionales, toda la lógica es ejecutada en el servidor y se apoya en el sistema de URLs con peticiones al servidor para mostrar unas vistas u otras, Esto trae consigo un problema de latencia al tener que procesar todo desde cero al cambiar de vistas, ya que para el navegador estas URLs son completamente diferentes entre sí, aunque tengan los mismos estilos.

En un SPA, toda la interfaz de usuario esta implementada casi en su totalidad en el navegador. Contiene todas las vistas de la web por lo que solo requiere de una carga inicial. Con esto se elimina el problema de la latencia al cambiar de vistas y se consigue una aplicación web con una velocidad similar a la de una aplicación nativa en lo que a cambiar de vistas se refiere. También se obtienen otras ventajas como poder gestionar el estado de la aplicación desde el cliente, evitando así sobrecargas en el servidor y saturación del ancho de banda.

La arquitectura propuesta para la construcción de la plataforma es la que se muestra en la figura 5.1.

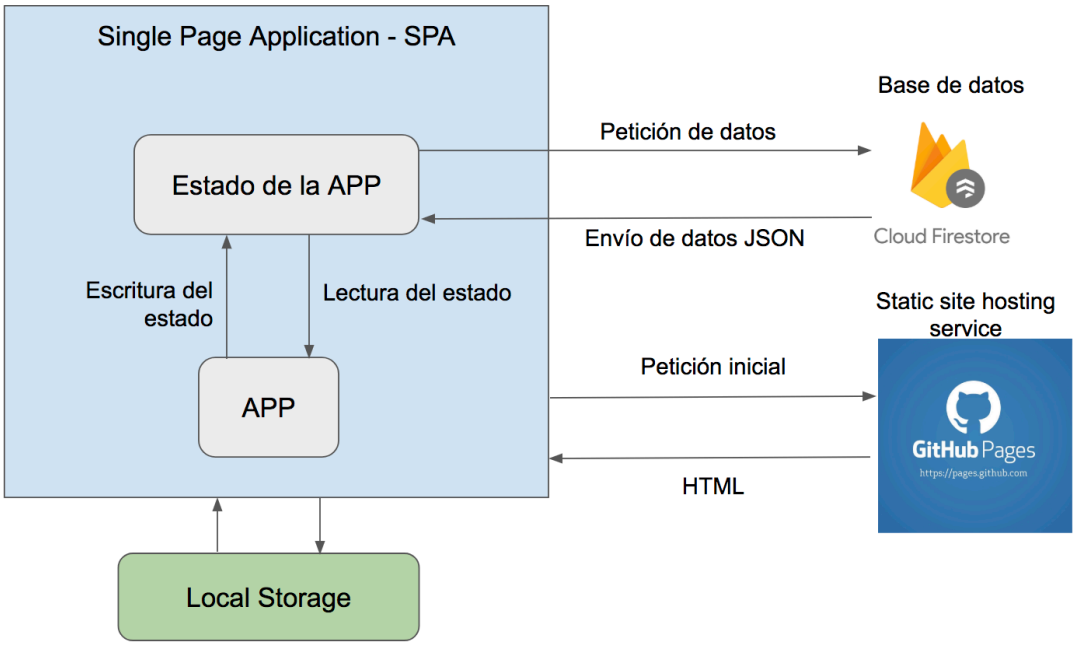

Figura 5.1: Arquitectura general de la plataforma

Tenemos a GitHub Pages para el hosting de nuestro código. De aquí descargamos

todo, una única vez.

La base de datos, Cloud Firestore que se comunica directamente con el cliente.

Y finalmente la SPA, que es ejecutada en el lado del cliente.

Internamente, la SPA gestiona su estado utilizando Redux, ReactiveX y Redux-Observable. Para la persistencia de datos, la SPA se apoya siempre en el Local Storage del navegador. Guardando una copia del estado de la aplicación cada vez que este cambia. De esta forma, al refrescar la página, la SPA recupera su estado del Local Storage para tener siempre algo que mostrar al usuario.

Podemos ver una aproximación en la figura 5.2.

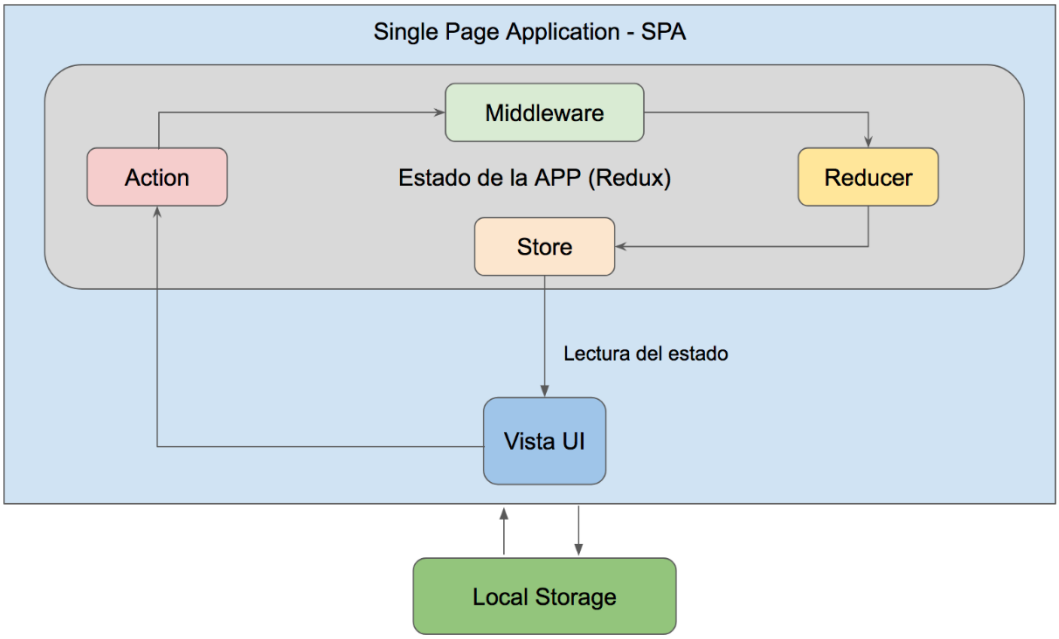

Figura 5.2: Arquitectura interna de la SPA

5.1 Gestión de usuarios y datos.

5.1.1 **Autenticación**

La autenticación de usuario se hace a través de Firebase Authentication, usando como proveedor Google. Es decir, para poder registrarse en la plataforma se ha de tener una cuenta de este proveedor.

Se ha limitado el registro de usuario a cuentas de Google aprovechando que las cuentas institucionales de la ULL pueden autenticarse con este método.

Este proceso nos proporciona información básica del usuario que utilizamos en la plataforma para identificar al mismo. Uno de los datos más importantes que obtenemos es el identificador de usuario o UID, único para cada usuario autenticado.

5.1.2 **Base de datos**

Cloud Firestore es una base de datos no relacional, con un modelo de datos orientado a documentos, los objetos JSON que manipulamos en la aplicación son almacenados en documentos y estos a su vez en colecciones, que son el equivalente a las tablas en el modelo relacional.

La base de datos es la encargada de almacenar todos los datos de la plataforma en las siguientes colecciones: usuarios, grupos de usuarios, salas, problemas, instancias de problemas, historial, ranking y contraseñas para acceder a salas y grupos.

Por ejemplo, para los datos de un usuario que estén almacenados en la lista de usuarios, el nombre de la colección es “userList” y el documento asociado al usuario tiene como ID el UID del usuario que se obtiene mediante la autenticación descrita anteriormente. Y el documento son los campos asociados a ese UID. Podemos ver un ejemplo en la figura 5.3.

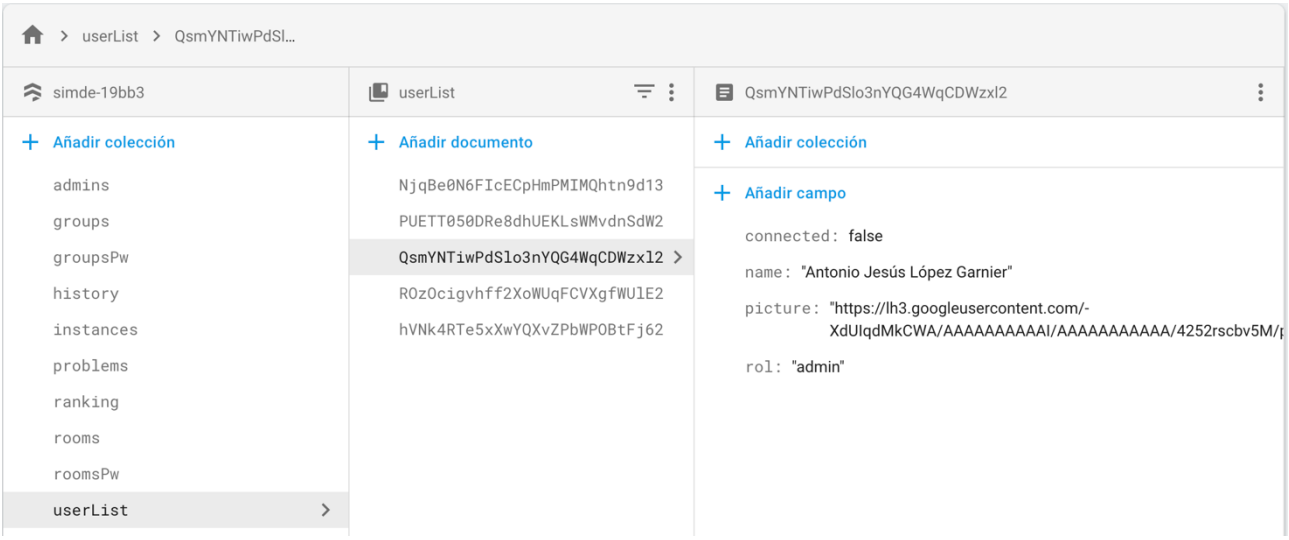

Figura 5.3: Lista de usuarios en Cloud Firestore

A continuación, una breve descripción de las colecciones mencionadas anteriormente. así como una explicación de los campos internos de cada documento:

- **UserList:** Es la colección para la lista de usuarios que almacena información básica que es compartida al resto de usuarios de la plataforma. Los datos son: nombre, foto, rol y estado de conexión o desconexión.
- **Groups:** Es la colección para la lista de grupos creados en la plataforma. Cada documento de la colección tiene un identificador único que es el ID del grupo. Los datos almacenados son: nombre del grupo, líder del grupo, fecha de creación y miembros. Este último es un objeto que tienen como propiedad los UID de los usuarios que pertenecen a dicho grupo y un valor de “true” o

"false" asociado.

- **Instances:** Es la colección para la lista de instancias de los problemas. Una instancia es la información sobre el contenido inicial en memoria del simulador y el contenido final para comprobar el resultado. También tienen un identificador único y contiene la siguiente información: nombre de la instancia, estado inicial, estado final y fecha de creación.
- **Problems:** Es la colección para la lista de problemas. Tienen asociado un ID único y contiene los siguientes campos: nombre del problema, definición del problema, fecha de creación e instancias asociadas a dicho problema. Este último campo es un objeto con las ID de las instancias asociadas al problema y un valor de "true" o "false". Con esto podemos determinar ¿qué instancia es asignada?, y ¿cuál lo fue?
- **Rooms:** Es la colección para la lista de salas. Es aquí donde se llevarán a cabo las actividades lúdicas. Tienen un ID único y contiene la siguiente información: nombre de la sala, tipo de sala ("single" o "group"), visibilidad de la sala, fecha de creación, miembros de la sala y problemas asociados. Estos últimos, miembros y problemas, son dos objetos que contienen como propiedades, las IDs de los miembros que pertenecen a la sala y los problemas asociados a dicha sala, respectivamente, y un valor "true" o "false".
- **History:** Es la colección para la lista de historiales. Cada historial se identifica por el UID del usuario, y almacena el ID y nombre de la sala, ID y nombre del problema, código usado en la sala y cantidad de ciclos realizados al resolver el problema con dicho código.
- **Ranking:** Es la colección para la lista de rankings. Existe un ranking por cada sala, es decir, el identificador único de cada ranking es el propio ID de la sala a la que está asociado. La información que guarda es la UID del usuario, la ID del problema y el número de ciclos que ha tardado en resolverlo.
- **Admins:** Es la colección que almacena la lista de administradores. Cada documento tiene un ID que es el UID del usuario, y se almacena un valor de "true" o "false" en el campo "admin". Con esto podemos determinar ¿quién es administrador?, y ¿quién lo fue?
- **GroupsPw y roomsPw:** Son las colecciones para almacenar las contraseñas de los grupos y salas. Tienen como identificador de cada documento el ID del grupo y sala respectivamente, y almacenan la contraseña en el campo "password".

5.1.3 Reglas de seguridad

Para la protección de los datos, se utilizan las reglas de seguridad de Cloud Firestore. Podemos restringir el acceso a los datos a personas que no estén

autenticadas y hacer reglas más específicas para situaciones concretas. Por ejemplo, la colección de “admins” es de lectura solo para personas autenticadas mediante Firebase Authentication, y de escritura para las personas autenticadas y que además estén en esa colección, es decir, sean admins. Podemos ver esta regla en la figura 5.4

```
service cloud.firestore {
  match /databases/{database}/documents {
    function isSignedIn() {
      return request.auth != null;
    }

    function isAdmin() {
      return exists(/databases/$(database)/documents/admins/$(request.auth.uid));
    }


    // Admins:
    // Read: Only signed can read.
    // Write: Only admins can write.
    match /admins/{admin} {
      allow read: if isSignedIn();
      allow write: if isAdmin();
    }
}
```

Figura 5.4: Regla de seguridad de Cloud Firestore

5.2 Single Page Application

Para poder describir con exactitud la arquitectura interna de la SPA, se va a dividir en tres partes: inicio de la aplicación, sincronización de datos, y cierre de la aplicación. A lo largo de estas tres partes se explicará cómo se han protegido el acceso a ciertas vistas, aun conociendo la URL directa, y el acceso a los datos.

5.2.1 **Inicio de la aplicación**

Una vez completado el registro y justo antes de iniciar la aplicación, se determina el rol del usuario y se recuperan datos de la base de datos.

Firebase Authentication nos provee de un agente que permanece a la escucha del login y logout de un usuario. Al comenzar a renderizar la vista principal, se muestra un estado de carga mientras se comprueba si el usuario esta autenticado. Vamos a ver lo que ocurre con las rutas y la recuperación de datos.

Rutas

En este punto, pueden darse dos situaciones:

1. El usuario **NO** está autenticado. En este caso se renderizan unas rutas para usuarios anónimos que sólo te permiten acceder al simulador y a la

pantalla de login.

2. El usuario **SI** está autenticado. Se procede a comprobar el rol del usuario:
    a. Es **estudiante**: Se renderizan unas rutas para los estudiantes que te permiten acceder a parte de la plataforma.
    b. Es **administrador**: Se renderizan unas rutas para los administradores que te permiten acceder a todas las funcionalidades de la plataforma.

En cualquiera de los casos anteriores, si se intentara acceder a una ruta distinta a las que te permite tu nivel de acceso, tu vista se redirige al inicio de la aplicación. Esto es gracias al enrutado dinámico que nos ofrece React-Router que, al no renderizar todas las rutas, es como si no existieran.

Recuperación de datos

La información del usuario una vez logueado es disparada con una **acción** para almacenarse en el **store** de Redux. Es decir, un objeto JSON con dos propiedades "type: USER_LOGIN" y "payload: datos de usuario". Ver figura 5.5

```
{
    "type": "USER_LOGIN",
    "user": {
        "displayName": "Antonio Jesús López Garnier",
        "email": "aluXXXXXXXXX@ull.edu.es.",
        "creationTime": "Fecha de creación",
        "LastSignInTime": "Última vez que inició sesión",
        "picture": "Url foto cuenta de google",
        "rol": "student / admin",
        "UID": "Identificador de usuario"
    }
}
```

Figura 5.5: Acción disparada en formato JSON

Esta acción llega al **reducer** y al **epic** (middleware de redux-observable) al mismo tiempo. Una vez en el reducer, se almacena la información en el store. Pero al pasar por el epic, da como resultado otra serie de acciones diferentes. Ver figura 5.6.

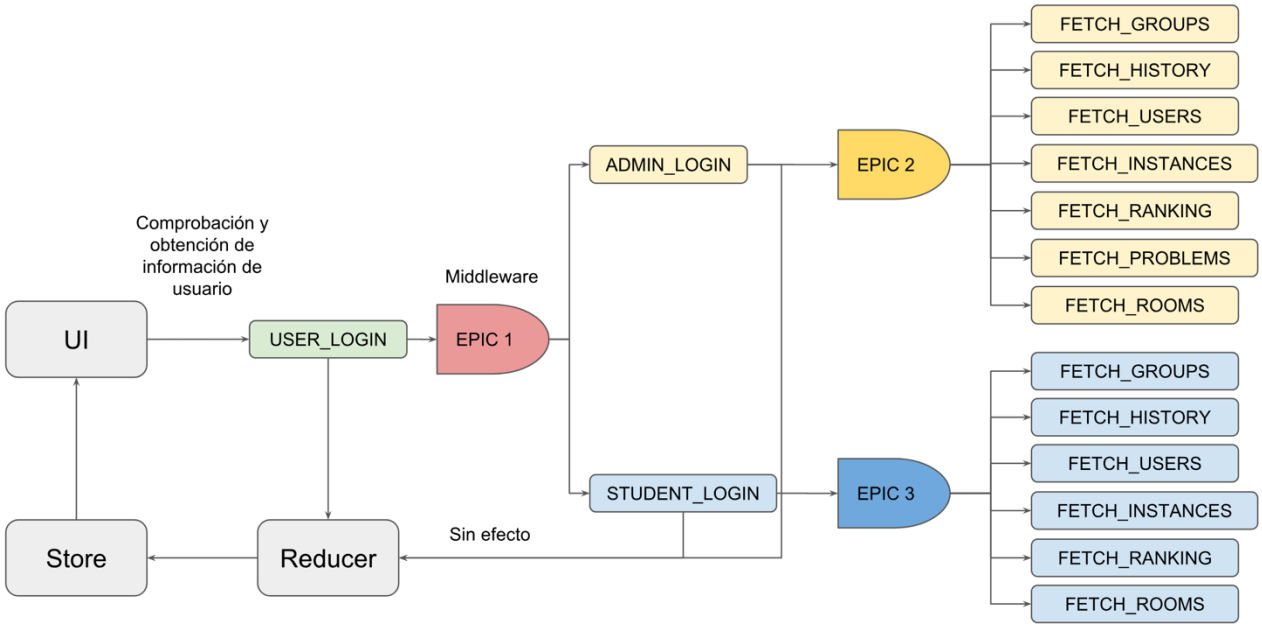

Figura 5.6: Acción de Login de usuario y sus resultados

Como vemos en el diagrama el proceso para recuperar los datos según el nivel de acceso de un usuario sigue el siguiente ciclo:

1. Se dispara una acción "USER_LOGIN" con la información del usuario autenticado.
2. Llega al "EPIC 1" y según el campo "rol" se lanza la acción "ADMIN_LOGIN" o "STUDENT_LOGIN".
    a. Caso **administrador**: La acción "ADMIN_LOGIN" llega al "EPIC 2" y lanza siete acciones más para la recuperación de datos.
    b. Caso **estudiante**: La acción "STUDENT_LOGIN" llega al "EPIC 3" y lanza seis acciones más para la recuperación de datos.
3. La diferencia en este caso es que el alumno no descarga los problemas de la base de datos ni las salas cuya visibilidad sea "false". Los problemas solo se descargan al entrar a una sala y solo recupera los asociados a esa sala.

Redux te permite escuchar cambios en el store y actuar en consecuencia. Gracias a esto, cada vez que ocurre un cambio, hacemos una copia del estado de la aplicación en el Local Storage del navegador. Al refrescar la página, Redux se reinicia y se pierden los datos. Sin embargo, cada vez que esto ocurre, si existen datos de la aplicación en el **Local Storage**, se recuperan. De esta forma, podemos mostrar al usuario, algo de información mientras se recuperan nuevos datos de forma asíncrona.

5.2.2 **Sincronización de datos.**

Cloud Firestore proporciona agentes de escucha para obtener actualizaciones en

tiempo real y mantener los datos sincronizados. Estos agentes se activan de formas diferentes según el rol del usuario. Por ejemplo, para los alumnos, que solo pueden recuperar datos de salas visibles, se activa un agente de escucha que filtra las salas y observa los cambios solo en las salas visibles.

Estos agentes se activan según la acción disparada anteriormente "ADMIN_LOGIN" o "USER_LOGIN", como podemos ver la figura 5.7.

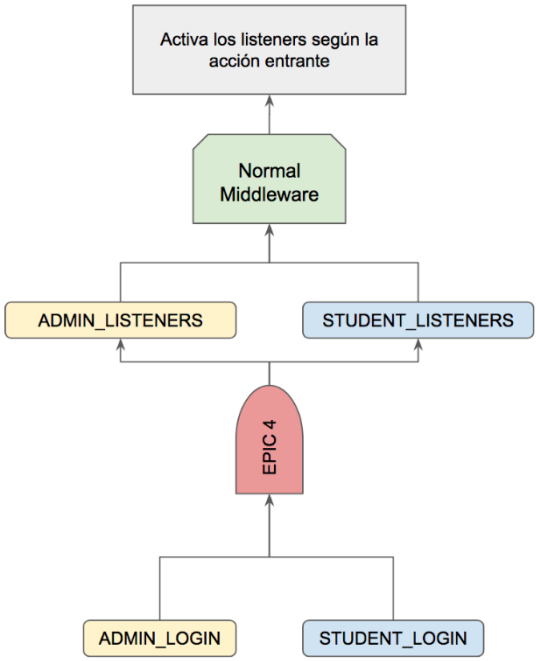

Figura 5.7: Activación de los agentes de escucha

Estas dos acciones continúan su camino y pasan a través de otro Epic que lanza la acción que indica el tipo de agentes de escucha que se deben activar. Estas últimas acciones pasan por un middleware que finalmente activa los correspondientes agentes.

Finalmente, para mantener la sincronización en tiempo real, para cada colección de datos tenemos tres observables que se encargan de añadir, actualizar o borrar datos, según se lo indiquen los agentes de escucha. Podemos ver una aproximación para el caso de las salas en la figura 5.8.

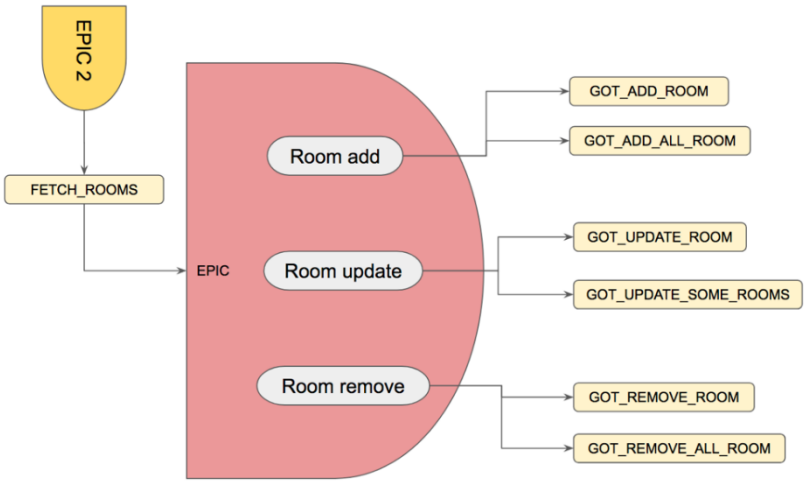

Figura 5.8: Observables emitiendo acciones dentro de un epic

Como vemos en la figura, dentro del epic tenemos tres observables, cada uno encargado de transformar los datos entrantes en acciones para emitir.

Sin embargo, para que los observables puedan emitir acciones, tienen que estar suscritos a un "observer" (conjunto de funciones que saben cómo tratar los datos de ese observable) y esta suscripción no se realiza hasta que la acción "FETCH_ROOMS" alcance el epic.

Cuando la acción llega al epic, los observables se suscriben a las funciones definidas dentro del epic. Y cada vez que un agente de escucha recibe un cambio, lo envía al observable correspondiente que se encargará de generar y lanzar la acción necesaria para guardar los nuevos datos en el store. Ver figura 5.9.

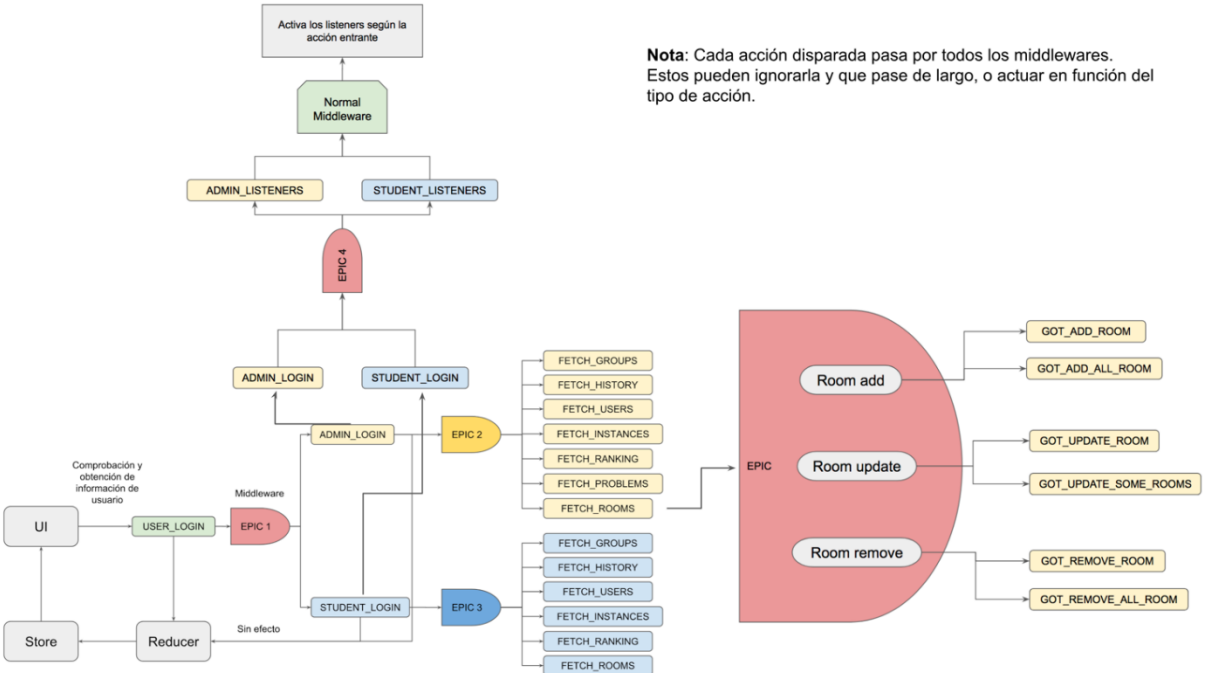

Figura 5.9: Ecosistema para el inicio y mantenimiento de los datos

Otra de las utilidades de utilizar los epics es la implementada para controlar el envío de resultados desde una sala a la colección de ranking de la base de datos.

El epic puede recibir la acción de enviar resultados y no admitir más envíos hasta que haya pasado un periodo de tiempo determinado. También podría enviar los resultados si al cabo de un tiempo determinado no ha recibido una nueva acción de enviar resultados.

Con esto conseguimos que no se abuse del envío de resultados y no sobrecargamos la base de datos.

5.2.3 **Cierre de la aplicación**

Al cerrar la aplicación pueden ocurrir dos cosas: que el usuario cierre el navegador sin terminar la sesión, o que el usuario termine la sesión antes de cerrar el navegador

1. **El usuario cerró el navegador sin terminar sesión**: En este caso, los datos almacenados en Redux se pierden ya que la aplicación deja de funcionar. Sin embargo, la copia que teníamos guardada en el Local Storage se mantiene. Es decir, al abrir de nuevo la aplicación, se encontrará en el mismo estado que cuando la cerramos.
2. **El usuario termina la sesión antes de cerrar el navegador**: En este caso, se realiza un borrado del Local Storage donde esta almacenada la copia del estado para liberar espacio de memoria del cliente.

Capítulo 6
Descripción funcional

Tras explicar las tecnologías empleadas y la arquitectura del sistema, en este capítulo se abordarán los resultados obtenidos, los ejercicios de ludificación que podemos realizar y las funcionalidades de la plataforma.

6.1 Ludificación en la plataforma

En esta sección se explicarán las funciones lúdicas de la plataforma [33], detallando en profundidad las tres modalidades existentes.

La plataforma permite a los alumnos unirse a salas o salas de grupos, donde se les presenta un problema a resolver con una información de la que partir.

En el apartado 2.3 vimos cómo es usado actualmente el simulador de máquinas superescalares en el aula. La idea consiste en resolver problemas con el fin de obtener un aprendizaje. En este caso, la plataforma ofrece tres modalidades de juego para conseguirlo: modalidad individual en las salas individuales, modalidad en grupo en las salas de grupos y modalidad “Adelanta a tu profesor” que puede ser jugada en ambas salas.

6.1.1 **Competición**

Cada alumno deberá acceder a la sala indicada por el profesor con la correspondiente contraseña. Una vez dentro se les presenta uno o varios problemas a resolver con sus instancias asociadas.

El alumno comenzará entonces a desarrollar el código que resolverá el primer problema planteado. Una vez lo tenga, puede probarlo directamente con las instancias pulsando en el botón correspondiente. Ver figura 6.1.

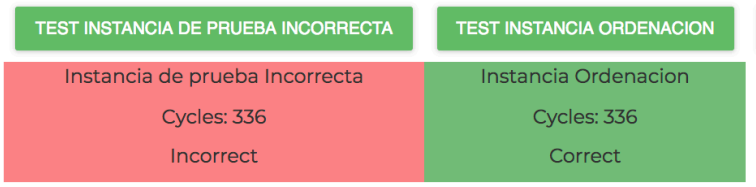

Figura 6.1: Resultados al probar el código con las instancias

Como vemos en la imagen, una prueba puede ser correcta, o incorrecta en el caso de que el estado final de la memoria en el simulador no coincida con el estado final de la instancia. Para poder superar el problema, todas las instancias deben estar superadas satisfactoriamente. Obteniendo como resultado final la media de ciclos de todas ellas. Ver figura 6.2.

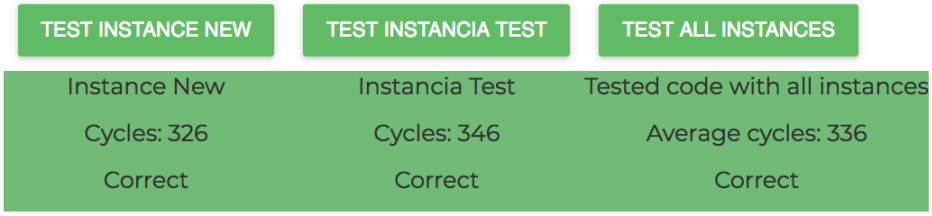

Figura 6.2: Instancias superadas satisfactoriamente

Cuando nuestro código resuelva todas las instancias, podremos enviar los resultados del problema a la base de datos, para posteriormente calcular nuestra puntuación actual y mostrarnos en el ranking. Ver figura 6.3.

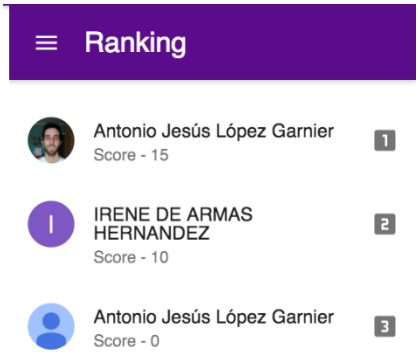

Figura 6.3: Ranking de puntuaciones

6.1.2 **Modalidad individual**

En esta modalidad el alumno compite contra el resto en las salas individuales. Desarrollando su código y enviando los resultados obtenidos para escalar posiciones en el ranking.

6.1.3 **Modalidad en grupo**

En este caso, los alumnos tendrán que ser miembros de un grupo para poder acceder a las salas de esta modalidad. Una vez dentro, deben cooperar para obtener el mejor resultado posible y comunicárselo al líder del equipo. Este último es el único que tiene permiso para enviar resultados al ranking. De esta forma se busca la cooperación de los estudiantes para aprender en grupo.

6.1.4 **Modalidad “Adelanta a tu profesor”**

Esta modalidad puede ser jugada en ambas salas. Se parte de una situación inicial como la que se describe a continuación:

1. El profesor desarrolla un código que resuelve el problema y sus instancias en un número determinado de ciclos.
2. Éste envía los resultados al ranking.
3. El profesor no es miembro de la sala, ni miembro o líder de un grupo. Es una entidad aparte. Pero puede realizar las mismas acciones que un estudiante.

Dada esta situación inicial, los alumnos ingresarían en la sala y tratarían de escalar posiciones para situarse por delante del profesor.

6.1.5 **Integración con el simulador**

Para facilitar el desarrollo de código, el simulador de máquinas superescalares, se encuentra integrado en la plataforma. Gracias a esto, el alumno tiene la posibilidad de probar el código que está desarrollando con la instancia que desee.

Para ello, bastaría con pulsar el botón correspondiente a una instancia, y automáticamente en la barra lateral, sección “Simulator”, se tendría el código y el estado inicial de la memoria cargados en el simulador.

De esta forma, si una instancia da un resultado incorrecto, se puede volver a pulsar su botón y realizar una traza en el simulador para depurar el código.

6.2 Inicio

6.2.1 **Login**

Para acceder a la plataforma es necesario hacer un login, mientras tanto, la aplicación mostrará directamente al simulador de máquinas superescalares. Aquí podemos seleccionar la opción para loguearnos. Figura 6.4.

Figura 6.4: Login en el simulador.

Una vez en la vista de Login, solo tenemos que seguir los pasos y seleccionar una cuenta de google que tengamos activa, o registrar otra. Ver figura 6.5.

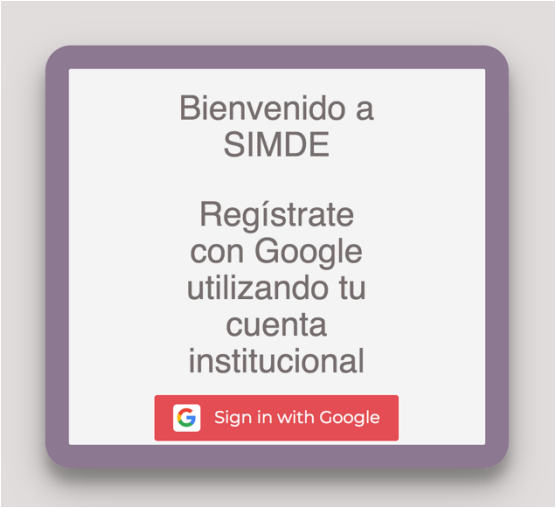

Figura 6.5: Pantalla de Login

6.2.2 **Welcome**

Al terminar el proceso de login, podemos visualizar la plataforma de ludificación. Que nos proporciona una visión inicial de la interfaz de usuario, así como nuestra foto, nombre de usuario y el botón de logout en la esquina superior derecha. Ver figura 6.6.

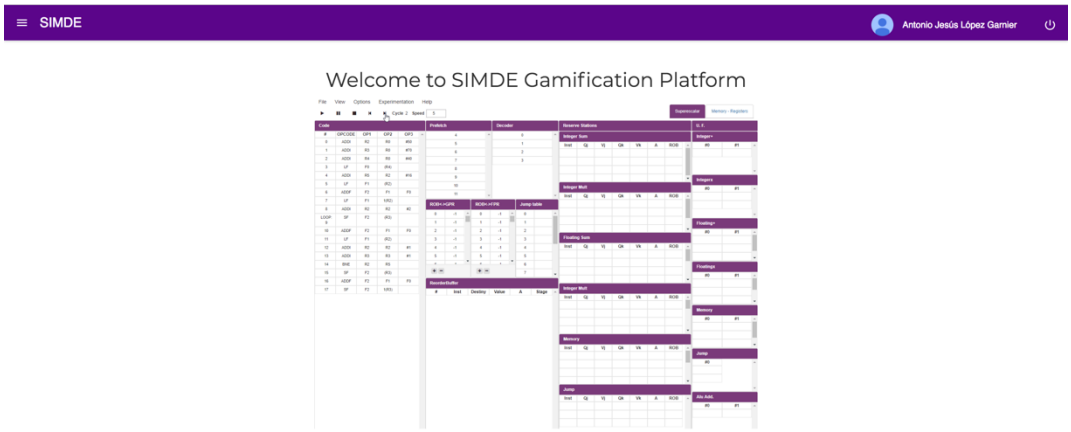

Figura 6.6: Vista principal de la plataforma

6.3 Barra lateral de secciones

La barra lateral de la aplicación nos proporciona un menú para navegar por la plataforma y acceder a sus funcionalidades. El uso de este menú está parcialmente restringido para los alumnos.

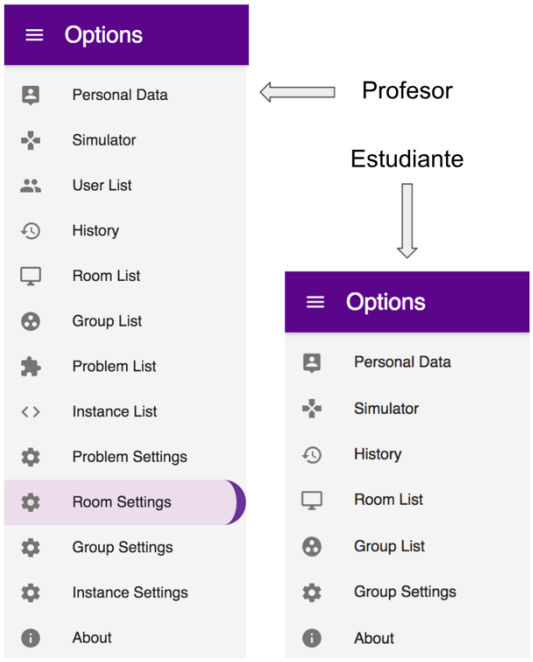

Figura 6.7: Barra lateral de la plataforma

Como podemos ver, el profesor puede acceder a todas las funcionalidades. Sin embargo, el alumno no tiene acceso a los problemas, lista instancias, lista de usuarios, o a las configuraciones de los problemas, instancias y salas.

6.4 Secciones

En este apartado se detallarán cada una de las secciones, así como sus funcionalidades. En cada una de las secciones, indicaré, si fuera necesario, las limitaciones para los alumnos.

6.4.1 **Personal Data**

En esta vista se muestran los datos personales del usuario: nombre y apellido, correo electrónico, fecha de alta en la plataforma y última conexión.

También tenemos un listado con las salas individuales, las salas de grupos y los grupos de alumnos de los que el usuario es miembro.

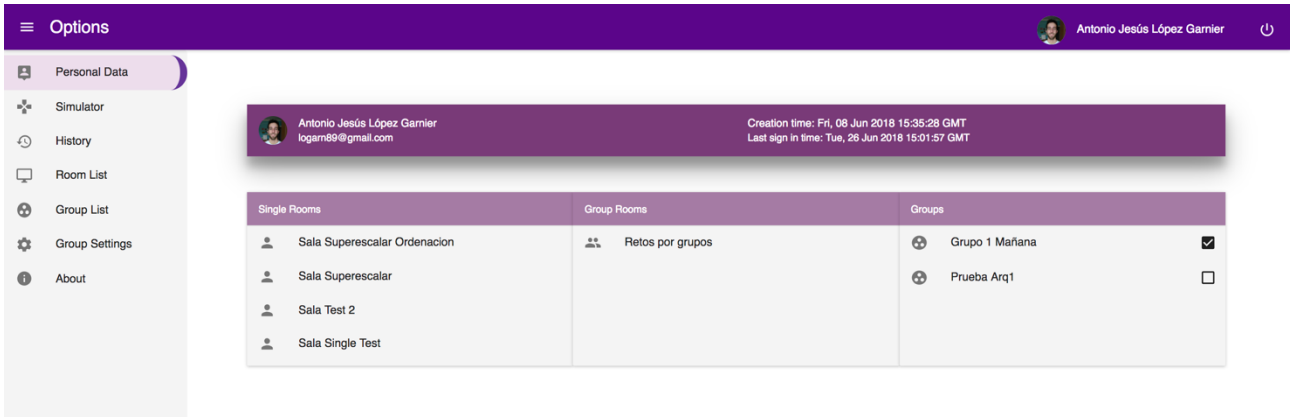

Figura 6.8: Sección de datos personales

Esta sección no solo nos proporciona una visión global del usuario, sino que nos permite acceder directamente a las salas o grupos a los que pertenecemos.

Para poder acceder a las salas de grupos es necesario tener un “Grupo activo” que se selecciona en esta vista, marcando el recuadro situado al lado del nombre del grupo.

6.4.2 **Simulator**

En esta vista podremos hacer uso del simulador de máquinas superescalares de la misma forma que se hace cuando no estas dado de alta en la plataforma. Si en esta vista decidiéramos ir a Archivo – Login, la plataforma nos redirigirá a la pantalla de bienvenida, ya que la ruta de login deja de existir una vez entras en la aplicación.

El simulador está completamente integrado con la plataforma. En la sección 6.1 vimos cómo interactuar con el simulador mientras competimos para hacer trazas de nuestro código y buscar posibles fallos.

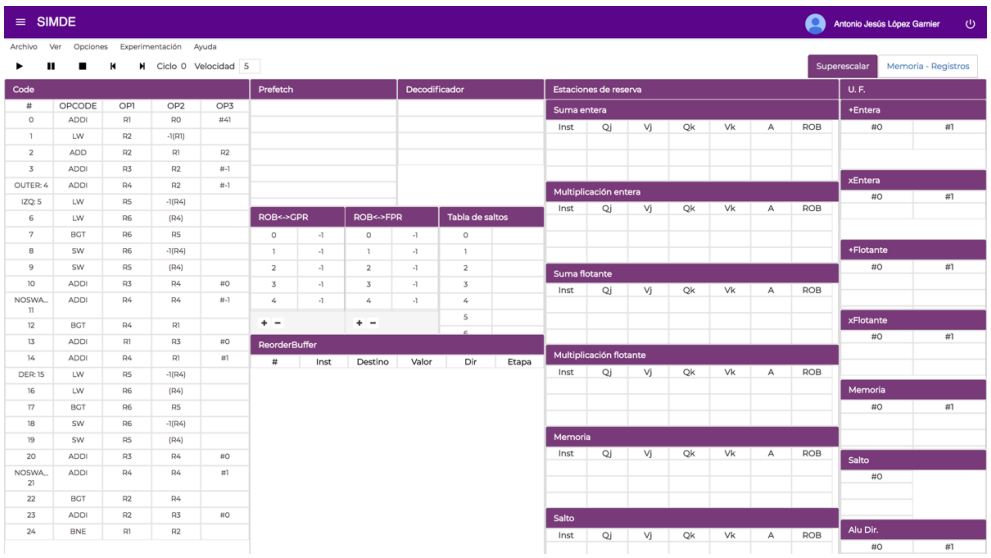

Figura 6.9: Sección del simulador

6.4.3 **User List**

Esta vista es únicamente accesible para los administradores de la plataforma. Permite saber en todo momento los usuarios existentes y ver su estado de conexión. Esta vista es meramente informativa.

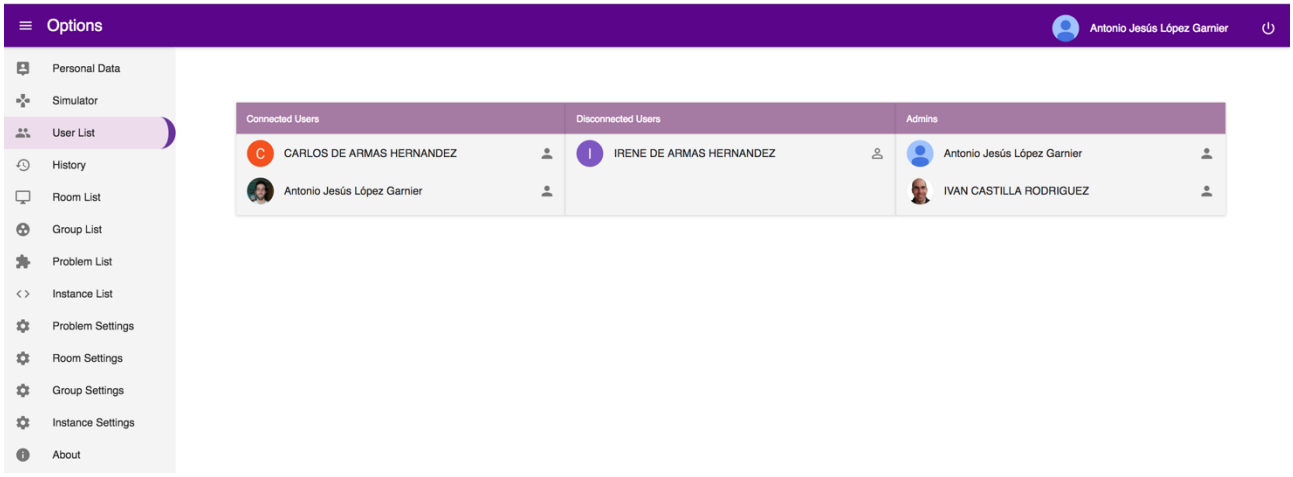

Figura 6.10: Sección de lista de usuarios

6.4.4 **History**

En esta sección podemos ver nuestro historial de código y resultados asociados a un problema de una sala.

Como vemos en la figura 6.11, nos muestra una lista de salas que podemos expandir para visualizar los problemas asociados, que también pueden expandirse para mostrarnos el código que se ha utilizado para resolverlo.

Para mejorar la experiencia de usuario al navegar por el historial en busca de un código para su posterior prueba, se puede copiar al portapapeles simplemente pulsando con el ratón en cualquier parte del código.

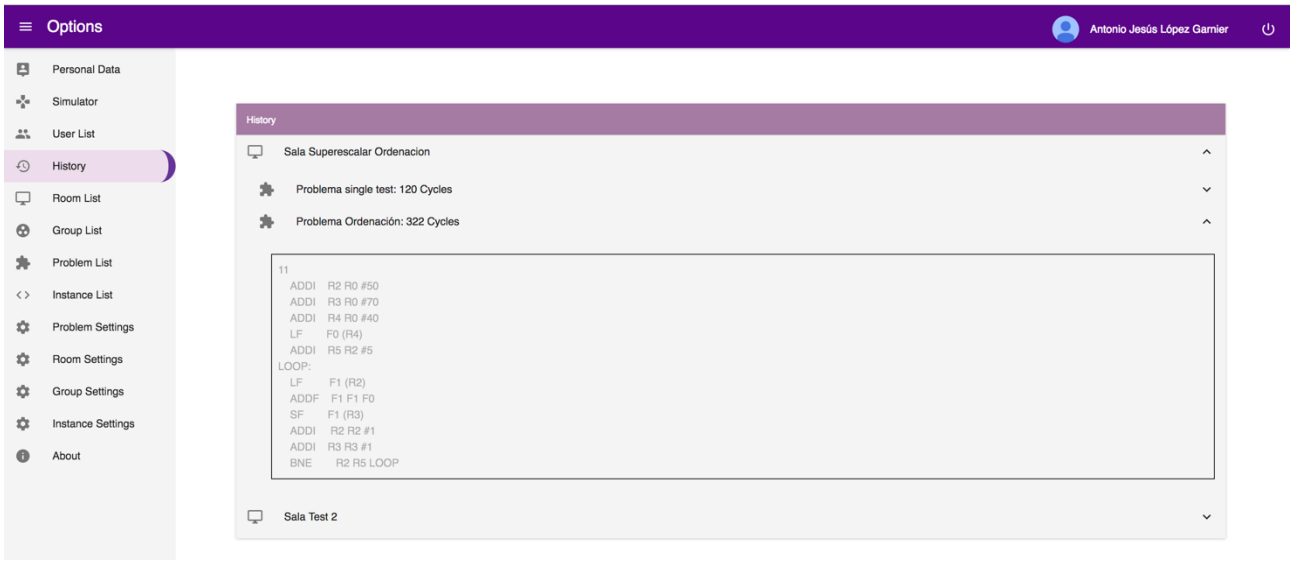

Figura 6.11: Sección de historial

6.4.5 **Room List**

Para describir esta sección es necesario dividirla en dos partes: la lista de las salas y la vista de una sala.

Lista de salas.

En esta sección tenemos una lista de las salas individuales y las salas de grupos. Los alumnos verán en esta vista únicamente las salas que el administrador haya seleccionado como visibles. Por otro lado, el administrador verá el listado de todas las salas independientemente de su visibilidad.

Al pulsar con el ratón en cualquiera de ellas pueden ocurrir dos cosas:

1. No eres miembro, por lo que se pide ingresar la contraseña de la sala para acceder. Una vez dentro, el estudiante descarga los problemas asociados a dicha sala.
2. Eres miembro de la sala y entras directamente a su visualización.

Si eres administrador, entres directamente en la sala sin necesidad de ingresar una contraseña.

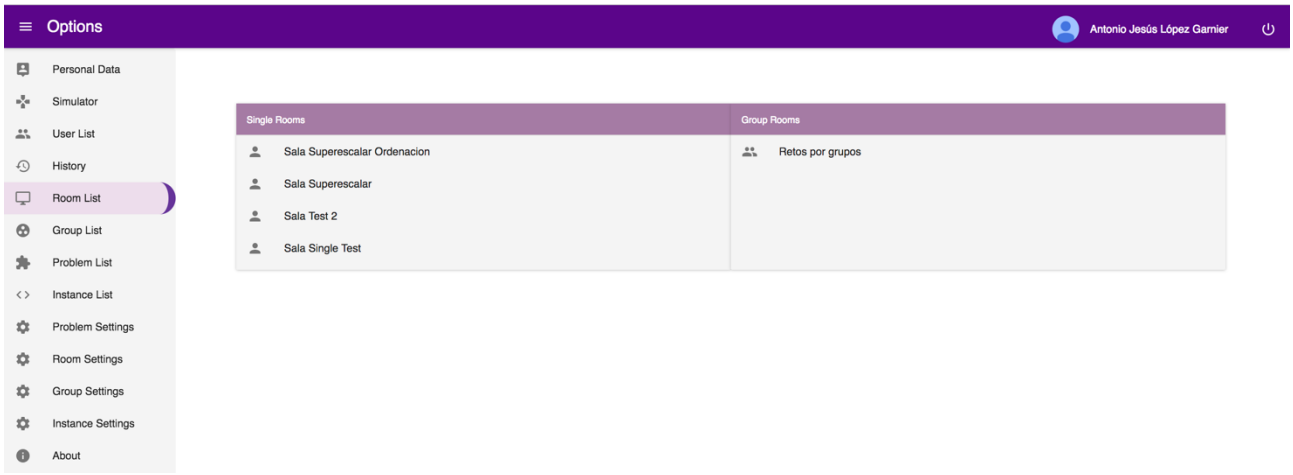

Figura 6.12: Sección de lista de salas

Vista de sala.

Una vez se accede a la sala, podemos ver los problemas asociados a dicha sala con sus respectivas instancias. Podemos realizar las siguientes acciones:

- Un botón “Next” y otro “Back” para movernos entre los problemas.
- Un botón “Reset Instances” para reestablecer los resultados obtenido en las instancias.
- Un botón “Send Results” para enviar los resultados obtenidos a la base de datos.
- Un botón simbolizado por una estrella, para mostrar u ocultar el Ranking de resultados.
- Un botón por cada instancia para probar el código introducido individualmente.
- Un botón para probar el código con todas las instancias y calcular la media de los resultados. Es este resultado el que se envía, siempre que el resultado de las instancias sea correcto.
- Un campo de texto donde escribir el código y un botón para guardarlo en el historial.

Cualquier usuario puede dejar de ser miembro de una sala. Sin embargo, en las salas de grupo solo el líder puede tomar dicha decisión para su grupo.

En esta sala, la vista para alumnos y profesores sólo difiere en la información referente a las instancias. Los alumnos sólo ven el estado inicial de la instancia, mientras los profesores pueden ver el estado inicial y el final.

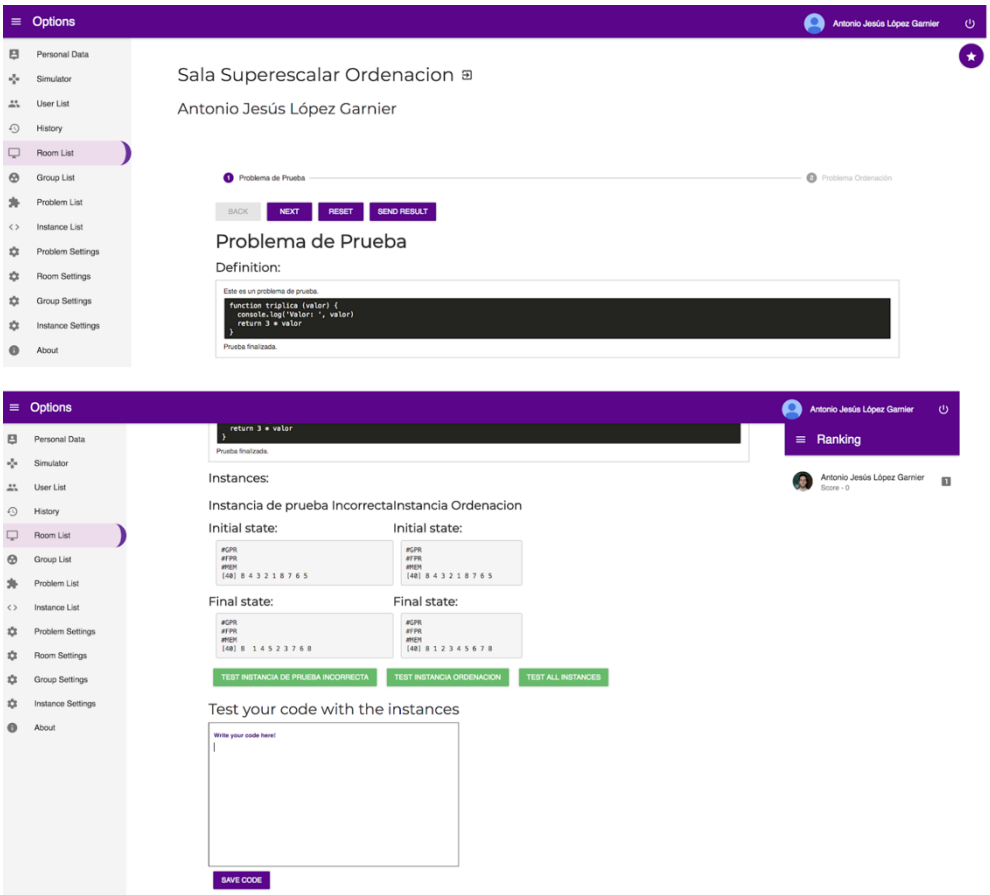

Figura 6.13: Vista individual de una sala

6.4.6 **Group List**

Esta sección se divide en la lista de los grupos y la vista individual de un grupo.

Lista de grupos

En esta vista se muestra un listado con todos los grupos existentes. Estos grupos son liderados por la persona que lo crea, que es el encargado de tomar las decisiones importantes en las salas de grupos.

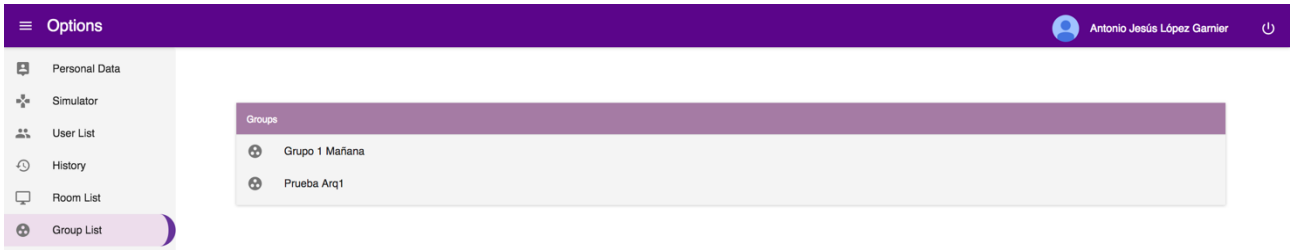

Figura 6.14: Sección lista de grupos

Vista de grupos

Para acceder a esta vista es necesario que seamos líder o miembro del grupo. Si no es así, se nos solicita una contraseña para unirnos. Los administradores pueden acceder sin necesidad de pertenecer al grupo. Dentro podemos ver los miembros del grupo, así como sus roles. Ver figura 6.15.

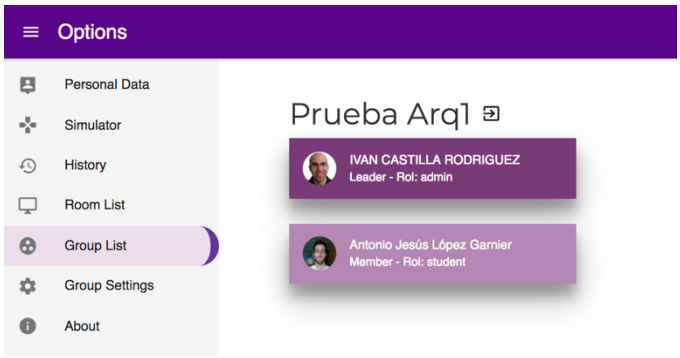

Figura 6.15: Vista individual de un grupo

6.4.7 **Problem List**

Esta sección se divide en la lista de problemas y la vista individual de un problema. Esta lista no es accesible para los alumnos. Únicamente los administradores pueden acceder a ella y ver la definición de los problemas. Ver figuras 6.16 y 6.17.

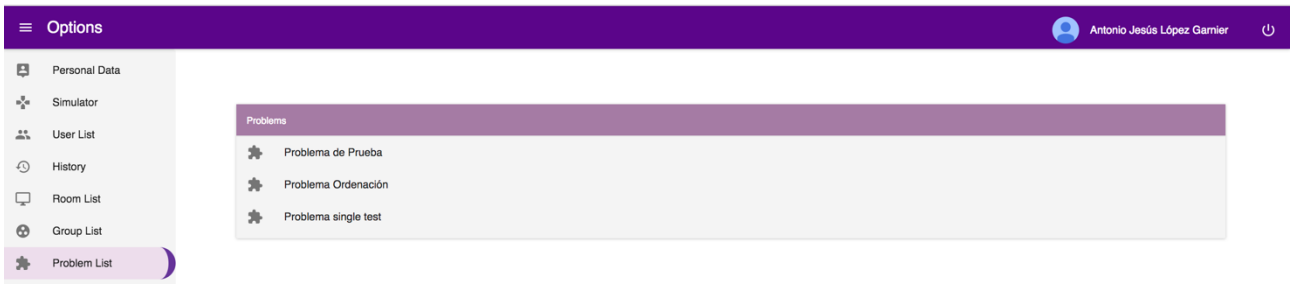

Figura 6.16: Sección listado de problemas

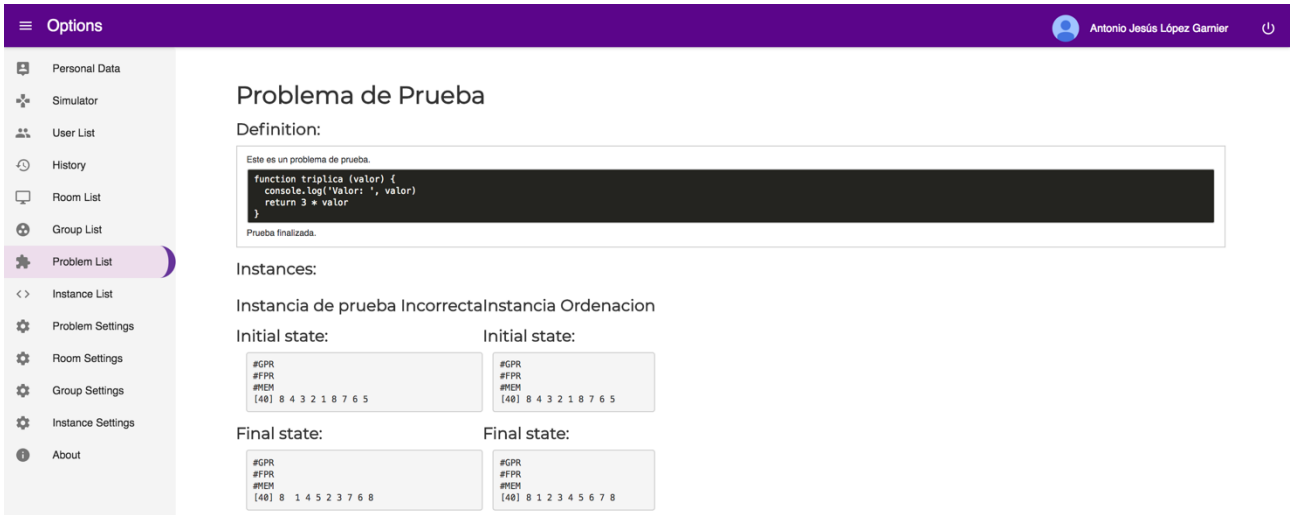

Figura 6.17: Vista individual de un problema

6.4.8 **Instance List**

En esta sección se muestra un listado de las instancias que han sido creadas. Esta sección tampoco es visible para los alumnos. Ver figuras 6.18 y 6.19.

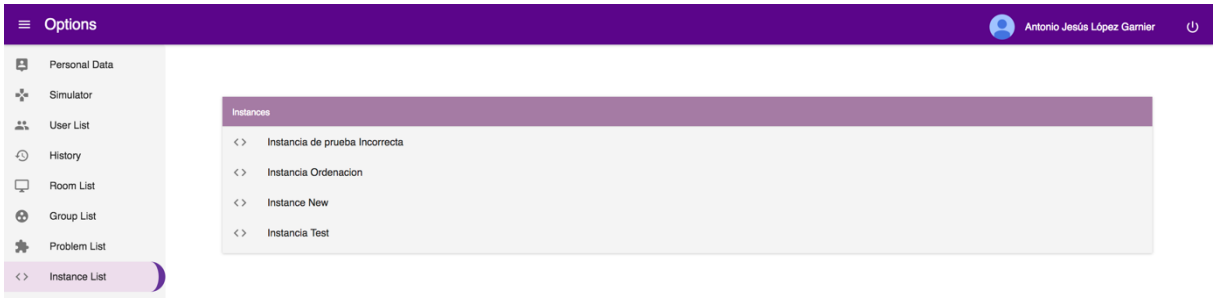

Figura 6.19: Sección lista de instancias

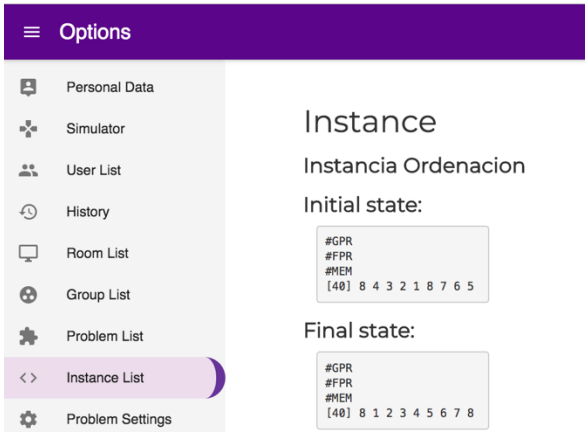

Figura 6.18: Vista individual de una instancia

6.4.9 **Room Settings**

Esta vista no es accesible para los alumnos, no puede crear salas. Éstas son gestionadas en su totalidad por los administradores.

En esta sección se crean, modifican o borran las salas. Está formada por un panel de pestañas que nos permite movernos entre vistas para realizar las acciones anteriores.

Para crear una sala es necesario completar los pasos que se indican en la parte superior de la vista:

- Nombre de la sala
- Problemas asociados a la sala.
- Propiedades de la sala:
  - Tipo de sala: Individual o de grupos.
  - Visibilidad.
  - Contraseña.
- Revisión de los datos.

Para actualizar algunos datos de la sala es necesario cambiar de pestaña, seleccionar la sala de la lista que aparece y proceder a cambiar sus propiedades. Los datos que podemos modificar son el nombre, los problemas asociados y la visibilidad.

Para borrar una sala, nos movemos a la última pestaña, seleccionamos la sala de una lista y pulsamos el botón “Remove” para completar la acción.

Podemos ver la vista principal de esta sección en la figura 6.20.

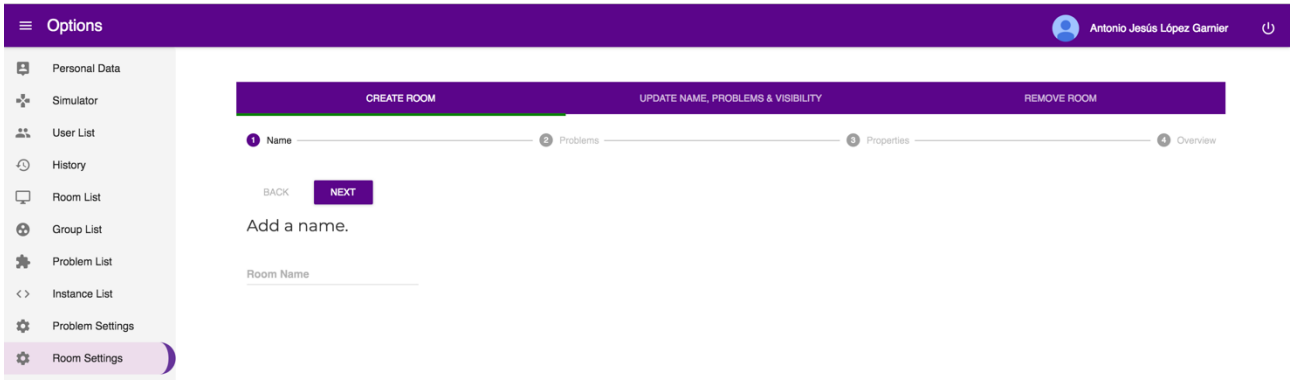

Figura 6.20: Sección de configuración de salas

6.4.10 **Group Settings**

Esta vista es accesible por los alumnos, que son los encargados de crear los grupos

y organizarse para entrar en las salas de grupo.

La información que se ha de rellenar es únicamente el nombre del grupo y una contraseña. Los líderes de cada grupo son los propios creadores de dicho grupo.

La vista está formada por dos pestañas:

1. **Crear grupo**: En esta pestaña podemos crear un grupo. Simplemente añadimos un nombre y una contraseña, revisamos la información del grupo, y confirmamos.
2. **Borrar grupo**: En esta pestaña podemos borrar un grupo. Únicamente se puede borrar un grupo si se es líder de dicho grupo. Por otro lado, un administrador puede eliminar cualquier grupo.

La vista principal de la configuración de grupos es la mostrada en la figura 6.21.

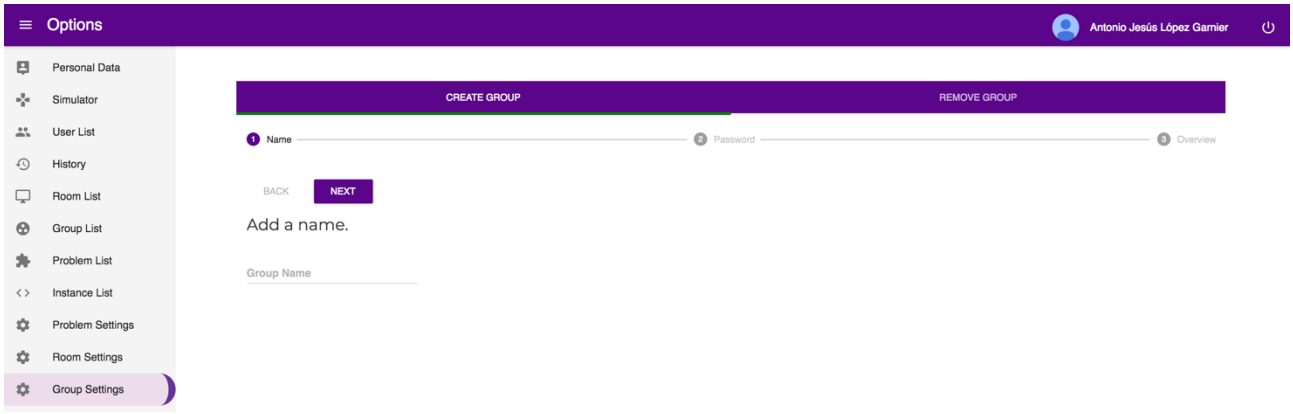

Figura 6.21: Sección de configuración de grupos

6.4.11 Problem Settings

Esta sección es para configurar problemas. Sólo los administradores pueden acceder a esta vista.

Se compone de un panel de tres pestañas que te permiten crear, modificar o borrar problemas.

Para crear uno, es necesario completar los siguientes pasos:

1. Nombre del problema.
2. Definición del problema.
3. Instancias asociadas al problema.
4. Revisar y confirmar los datos.

Para actualizar un problema, nos movemos a la pestaña central. Que nos permite modificar el nombre, la definición y las instancias del problema.

Para borrar el problema, en la última pestaña, seleccionamos el problema de una lista y confirmamos el borrado.

La vista principal de la configuración de los problemas es la mostrada en la figura 6.22.

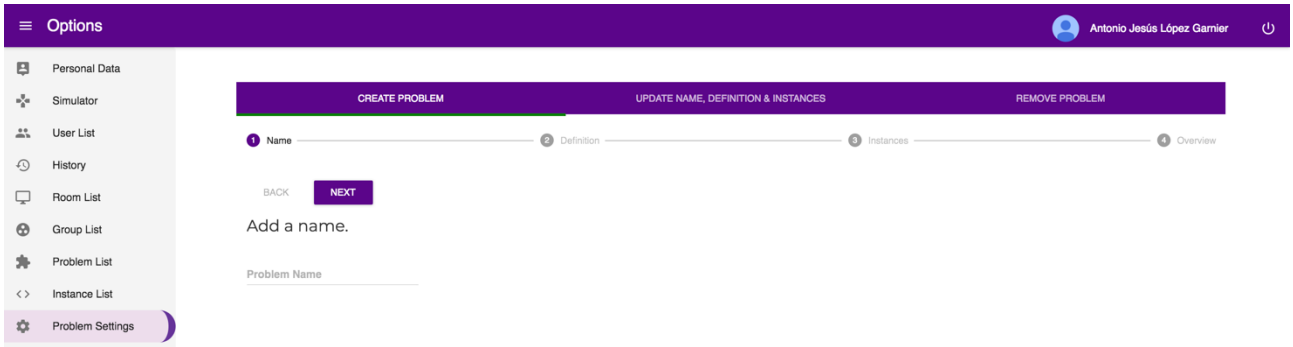

Figura 6.22: Sección de configuración de problemas

6.4.12 **Instance Settings**

En esta sección se crean las instancias. Solo los administradores pueden acceder a esta vista.

Se compone de un panel con dos pestañas:

1. Crear instancia. Se pide completar los pasos para rellenar la información de la instancia.
2. Borrar instancia. Se selecciona la instancia de una lista y se elimina.

La vista principal de la configuración de instancias es la mostrada en la figura 6.23.

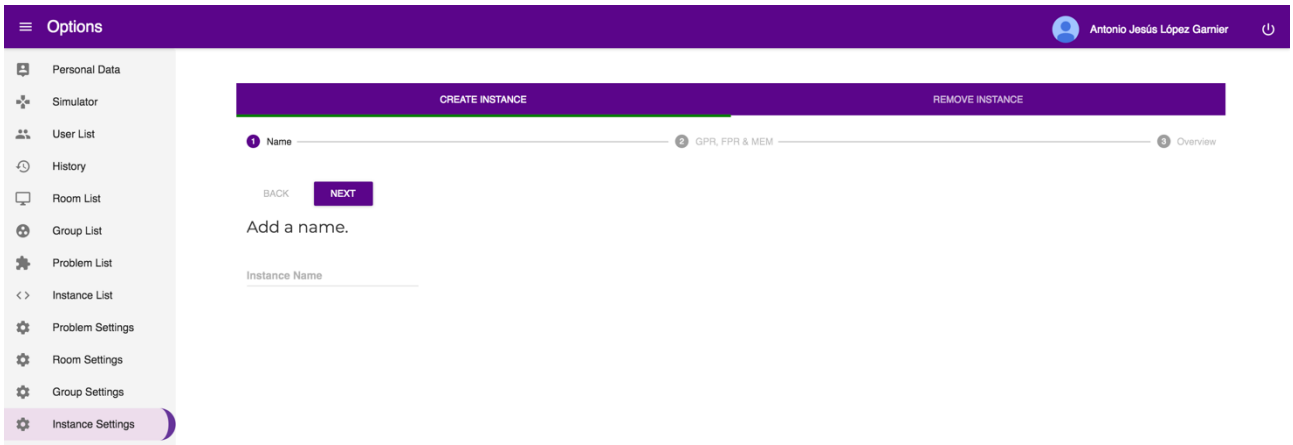

Figura 6.23: Sección de configuración de instancias

6.4.13 **About**

En esta sección se muestra información sobre el desarrollador y el tutor del proyecto.

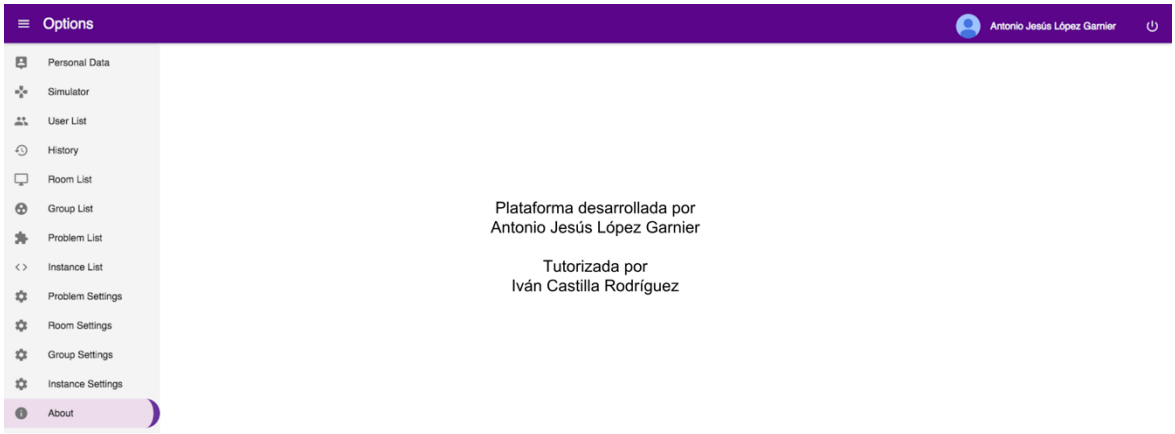

Figura 6.24: Sección de información del proyecto

6.5 Otras funcionalidades

6.5.1 **Avisos**

La plataforma ofrece un sistema de avisos para notificar al usuario de acciones realizadas con éxito, acciones erróneas o mensajes de alerta.

Se utiliza un único componente al que se le cambia el mensaje y el color para diferenciar cada una de las notificaciones:

1. Verde: Acciones con éxito.
2. Rojo: Acciones erróneas.
3. Naranja: Mensajes de alerta o “warnings”.

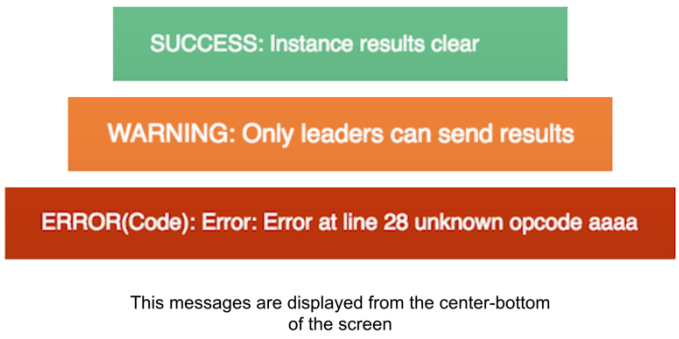

Figura 6.25: Notificaciones al usuario

6.5.2 **Solicitudes**

Cuando un usuario trata de visualizar la información de una sala o un grupo al que no pertenece, se activa una ventana emergente que solicita la contraseña para unirse a dicha sala o grupo. El componente utilizado es el mismo, lo único que cambia es el mensaje.

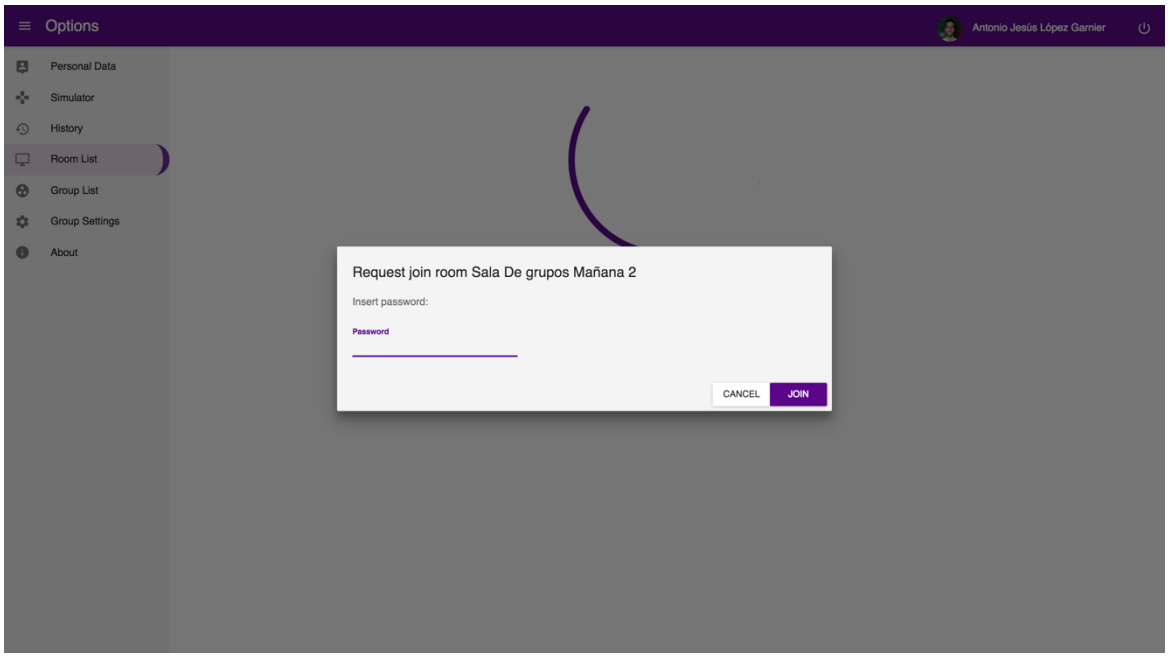

Figura 6.26: Intento de acceso a una sala

Capítulo 7
Conclusiones y líneas futuras

7.1 Conclusiones

Con el desarrollo de este proyecto se han adquirido y perfeccionado muchos conocimientos sobre tecnologías web y buenas prácticas de programación. Como resultado se dispone de una plataforma de ludificación que permitirá a los alumnos de Arquitectura de Computadores aprender de una forma diferente.

Cabe destacar el uso de las tecnologías empleadas, que han permitido desarrollar una aplicación con una base sólida, eficiente y escalable. Preparada para futuras ampliaciones o mejoras.

7.2 Líneas futuras

Al finalizar el trabajo se han abierto varios caminos:

- **Migración del control de usuarios**: Continuar el estudio y documentación de realizar la gestión de los usuarios a través del Moodle de la Universidad. Es decir, convertir la plataforma de ludificación en una Learning Tool Interoperability (LTI) para poder utilizarla como herramienta externa desde el Campus Virtual de la ULL.
- **Integrar el simulador VLIW:** Incluir este simulador en la plataforma de ludificación. Proporcionar una "Landing page" donde los usuarios puedan seleccionar qué simulador utilizar y opción a loguearse en la plataforma.
- **Mejorar las reglas de seguridad**: Las posibilidades que ofrece Cloud Firestore son muy grandes y flexibles. La posible ampliación y mejora de este proyecto, necesitará que estas reglas se mantengan actualizadas y mejoradas siempre que se posible.
- **Añadir nuevas características**: Compartir código entre usuarios de un grupo, mensajería instantánea entre miembros de un grupo, aportar feedback y guiar al alumno hacia la solución.
- **Incluir nuevas modalidades de juego**: Con una base sobre la que construir, es posible incorporar nuevas ideas sobre ludificación a la plataforma.
- **ImmutableJS**: Utilizar la librería Immutable.JS para ser usado como estado en Redux. Cuanto mas grande es la aplicación más complicado es mantener su estado. Esta librería nos ayuda a realizar los cambios de una forma más sencilla.

Capítulo 8
Summary and Conclusions

8.1 Summary

Developing this Project I have acquired and improved a lot of knowledge about web technologies and good practices. As a result, there is a gamification platform that will allow students to learn in a different way.

The use of this technologies is worthy of mentions, as they have contributed to have a strong, efficient and scalable base to start developing new features or extensions.

8.2 Future work lines

There are several future lines opened:

- **Migration of users control**: Continuing the research and documentation of managing users through the Moddle platform at the Universidad de La Laguna. Modifying the platform to get a Learning Tool Interoperability (LTI) so we can use it as an external tool from Moddle.
- **Integrating VLIW simulator**: Including this simulator in the gamification platform. Providing a "Landing Page" where users can select what to do: Superscalar, VLIW or Login.
- **Improving security rules**: Cloud Firestore offers a big and flexible set of possibilities. These rules will need to keep up to date and it is recommended to improve them if it is possible.
- **Adding new features**:
    - Sharing a code between users of a group.
    - Instant messaging between members of a group to facilitate communication.
    - Feedback: When failing to solve a problem, guide the student to the solution by providing advice.
- **Adding new game modes**: The developed framework allows new games to be included easily.
- **Immutable:** Using ImmutableJS API to manage Redux Store change.

Capítulo 9
Presupuesto

Descripción	**Coste por hora**	**Total**
241 horas de trabajo	35€	8435€
Nota: En el precio por hora se incluye la cuota de autónomo.		

Tabla 1: Presupuesto

Bibliografía

[1] LOFTUS, E. y LOFTUS, G. 1983. Mind at Play: The Psychology of Videogames. Basic Books. Nueva York (USA).

[2] DETERDING, S; SICART, M; NACKE, L; O'HARA, K y DIXON, D. 2011. "Gamification. using game-design elements in non-gaming contexts". In Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems. ACM

[3] Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education, 52*(1), 1-12.

[4] Werbach, Kevin y Dan Hunter. For the Win: How Game Thinking Can Revolutionize Your Business. Harrisburg: Wharton Digital Press, 2012.

[5] Prensky, M. (2007). Digital Game-Based Learning. McGraw-Hill, New York.

[6] Iván Castilla Rodríguez. Simulador didáctico de arquitectura de computadores. PhD thesis, Universidad de La Laguna, 2004.

[7] Adrián Abreu González
https://github.com/adrianabreu

[8] Simulador SIMDE Web
https://etsiiull.github.io/SIMDE/

[9] Typescript
https://www.typescriptlang.org/

[10] Javascript
https://www.javascript.com/

[11] Firebase Authentication
https://firebase.google.com/docs/auth/?hl=es-419

[12] Firebase Cloud Firestore
https://firebase.google.com/docs/firestore/?hl=es-419

[13] Doron Katz – Getting started with cloud firesrtore for iOS
https://code.tutsplus.com/tutorials/getting-started-with-cloud-firestore-for-ios--cms-30910

[14] Local Storage
https://developer.mozilla.org/es/docs/Web/API/Storage/LocalStorage

[15] React
https://reactjs.org/

[16] Sandy Weck & Marcel Birkner. Developing modern offline apps with ReactJS, Redux and Electron – Part 2 – ReactJS basics
https://blog.codecentric.de/en/2017/11/developing-modern-offline-apps-reactjs-redux-electron-part-2-reactjs-basics/

[17] Redux
https://es.redux.js.org/

[18] Xavi Rigau - Introduction to redux in flutter
https://blog.novoda.com/introduction-to-redux-in-flutter/

[19] Sandy Weck. Developing modern offline apps with ReactJS, Redux and Electron – Part 3 – ReactJS + Redux
https://blog.codecentric.de/en/2017/12/developing-modern-offline-apps-reactjs-redux-electron-part-3-reactjs-redux-basics/

[20] ReactiveX
https://rxjs-dev.firebaseapp.com/

[21] Redux-Observable
https://redux-observable.js.org/

[22] Simen Andresen - Using redux observable for asynchronous actins
https://medium.com/imersotechblog/using-redux-observable-for-asynchronous-actions-1afb31cbc01c

[23] React Router
https://reacttraining.com/react-router/core/guides/philosophy

[24] Material UI
https://v0.material-ui.com/#/

[25] Quill
https://quilljs.com/

[26] Create-React-App
https://github.com/facebook/create-react-app

[27] Babel
https://babeljs.io/

[28] Webpack
https://webpack.js.org/

[29] Sergio Xalambrí – Compilando el fron-end con webpack

https://medium.com/@sergiodxa/compilando-el-frontend-con-webpack-d251f7a632ec

[30] SPA

https://www.roiting.com/que-son-spa-single-page-applications

[31] ReactiveX How

http://reactive.how/

[32] Netflix UI Engineering – Jay Phelps – Netflix JavaScript Talks – RxJS + Redux + React = Amazing!

https://www.youtube.com/watch?v=AslncyG8whg

[33] Plataforma de ludificación

https://antoniogarnier.github.io/SIMDE.JS/