KSJ Games

Director: Óscar Brizuela (820773@unizar.es)

Plan de gestión, análisis, diseño y memoria del proyecto
10
Grupo Karen Spärck Jones

ORGANIZACIÓN: GITHUB.COM/30226-2024-10

FECHA DE PRESENTACIÓN DE LA PROPUESTA:
25 / 05 / 2024

Integrantes:
José Miguel de la Ascensión Doñate
Kamal Bouizy Ghazzal
Óscar Brizuela García
Alejandro Calvera Tonin
Jorge Jaime Modrego
Miguel Lacámara Pasamar
Pablo Larraz Jordán
Nicolás Pelegrín Sánchez

Índice general

Introducción

El proyecto consiste en el desarrollo de un juego de ajedrez para navegador web y móvil. Este, por tanto, sigue las reglas del ajedrez permitiendo a los jugadores jugar tanto en línea como en local.

El juego que se ha desarrollado cuenta con dos modalidades: local, de manera que ambos contrincantes pueden jugar entre sí en el mismo dispositivo, y en línea. El modo en línea cuenta, a su vez, con dos modos de partida: síncrono y asíncrono. En el modo síncrono se puede escoger entre diferentes duraciones de partida.

El sistema ofrece la oportunidad al usuario de registrarse, haciendo uso de un nombre de usuario y de una contraseña; de esta manera, si un usuario no está registrado, solo puede jugar en el modo local. Aquellos usuarios que estén registrados pueden hacer uso de funcionalidades como un sistema de emparejamiento por nivel entre jugadores, recompensas según el nivel del usuario, un sistema de *ranking* y personalización de fichas y emoticonos con los que jugar en partida.

En cuanto a las partidas, el sistema dispone de una interfaz gráfica con un tablero de ajedrez, y sus respectivas piezas, cuya apariencia depende del nivel del pase de batalla del jugador. Los jugadores pueden llevar a cabo una conversación gracias a un *chat* de partida a través del cual se pueden enviar tanto emoticonos como mensajes. Además, las partidas cuentan con una opción de rendición de partida: de esta manera, si uno de los jugadores no quiere continuar con la partida, la puede terminar de forma rápida, perdiéndola automáticamente.

El propósito de la aplicación es proporcionar un servicio de juego de ajedrez con un emparejamiento justo, en el que conforme el usuario mejora su nivel de juego y va adquiriendo experiencia se le premia con diferentes recompensas.

El alcance del proyecto incluye el desarrollo de la aplicación de juego de ajedrez para navegadores web y dispositivos móviles.

Como fechas e hitos importantes, estaba previsto contar el día 15 de abril con una aplicación jugable online, tanto para móvil como para *web*, que contara con un único tablero por defecto, *chat* y modo de juego asíncrono. El día 10 de mayo se obtuvo una versión del juego que, además de lo descrito previamente, contaba también con un pase de recompensas, así como con distintas arenas de juego. Por uĺtimo, en la entrega final, presentada el pasado 16 de mayo, se mostró la versión final del juego. Esta contaba con todas las funcionalidades descritas ya implementadas a la perfección.

El documento está dividido en varias secciones, en las cuales se va a comentar la organización del proyecto, el plan de gestión, el análisis y diseño del sistema y la memoria del proyecto. Cada una de estas secciones contará, a su vez, con distintos apartados en los que se hablará de los detalles técnicos del proyecto.

1. Organización del proyecto

El proyecto está conformado por 8 personas, todas ellas estudiantes de Ingeniería Informática de la Universidad de Zaragoza. Óscar Brizuela presenta el puesto de director del proyecto, por lo que ha sido el encargado de asegurar la consecución de los objetivos y el desarrollo dinámico de la actividad, además de ser la persona a la que se ha acudido con dudas de carácter general del proyecto.

El proyecto está dividido en 3 subgrupos (departamentos) principales: *backend*, *frontend-web* y *frontend-móvil*.

El departamento de *backend* está conformado por Miguel Lacámara y Óscar Brizuela. Ambos se han encargado de implementar la lógica del juego, así como proporcionar la *API* mediante la cual se gestionarán las peticiones desde y hacia los demás departamentos. También han llevado a cabo el diseño y la implementación de la base de datos, el despliegue de la aplicación y la integración de la *IA*.

El departamento de *frontend-web* está formado por Pablo Larraz, Alejandro Calvera y José Miguel de la Ascensión. Este último se centra en la jugabilidad de la página, es decir, modos de juego, tableros y personalización del mismo. El segundo ha dedicado su tiempo al diseño de la gran mayoría de componentes *web*. El primero proporciona soporte a ambos compañeros y es el responsable de comunicarse con los distintos subgrupos cuando haga falta. Por otro lado, el último integrante de este departamento ha sido el encargado de diseñar un tablero de juego completamente funcional así como las fichas de este.

El departamento de *frontend-móvil* está formado por Nicolás Pelegrín, Kamal Bouizy y Jorge Jaime Modrego. El primero se ha dedicado al diseño de pantallas y conexión con el *backend* para algunas funcionalidades y se ha mantenido en contacto con el otro grupo frontend para coordinar los diseños. El segundo se ha encargado de desarrollar el tablero y fichas y, en general, la jugabilidad del ajedrez para la *app.* El tercero se ha encargado de la creación de la estructura de pantallas y su funcionamiento en general.

A pesar de la división del proyecto en estos 3 departamentos, es necesario remarcar que las pruebas, en nuestro caso solo de aplicación, han sido llevadas a cabo por todos los integrantes del proyecto a medida que se ha ido construyendo el *software*. Asimismo, los 3 departamentos han estado en constante comunicación para que la integración entre todas las partes fuera correcta y eficiente.

2. Plan de gestión del proyecto

3.1. Inicio del proyecto

3.1.1. Identificación y asignación de recursos

Para el móvil se debe tener simplemente un dispositivo *Android*, que cuente con la versión 13, y el archivo *.apk* a ejecutar.

Para la versión web, es necesario contar con un navegador como *Google Chrom*e y acceso a *internet*.

El principal recurso para llevar a cabo el despliegue es una máquina virtual dentro del servidor *cloud* de *GCloud* proporcionada por la versión gratuita de una cuenta en *Google Cloud Platform*.

Cabe destacar que es probable que la forma de desplegar el sistema durante el desarrollo del mismo y la realización de las pruebas puede variar respecto a la forma de desplegarlo en la entrega final.

3.1.2. Formación inicial de los miembros del equipo

El equipo de este proyecto cuenta con formación en las distintas ramas de Ingeniería Informática. Todos sus integrantes cuentan con conocimientos acerca de las metodologías, herramientas y estándares para la consecución del proyecto.

Algunas de las cualidades que presenta el equipo para este fin son una experiencia previa en desarrollo de sistemas de información *web*, una breve experiencia en el desarrollo de aplicaciones *Android* y breve experiencia en la gestión de proyectos

Debido a la limitada experiencia previa, se ha dedicado una parte importante del esfuerzo a aprender las nuevas tecnologías y sus particularidades. Tanto cursos como vídeos utilizados y otros recursos se pueden observar en el apartado de Anexo V.

3.2. Control de configuraciones, construcción del *software* y aseguramiento de la calidad del producto

Los responsables encargados del control de las configuraciones de manera consistente, así como de construir un software de calidad serán Óscar Brizuela por parte del *backend*, Alejandro Calvera por parte del *frontend-web* y Kamal Bouizy por parte del *frontend-móvil.*

3.2.1. Convenciones de nombrado y guías de estilo del código fuente

En cuanto al nombrado y estilo en nuestro código fuente se ha decidido seguir las convenciones propias de los lenguajes utilizados. Esta decisión se debe a que en algunos casos el lenguaje requiere que las variables tengan ciertos nombres o que la estructura sea de una forma concreta, por lo que de esta manera se evitan errores y se crea un código fácilmente comprensible para cada uno de los grupos.

El idioma tanto de las variables como de los comentarios es en español debido a que es la lengua madre de todos los integrantes del equipo hasta el momento.

Los comentarios solo han sido asignados cuando se ha visto la necesidad de entender mejor el código. Además, estos comentarios son lo suficientemente explicativos, aunque escuetos, para comprender qué realiza cada función o bloque de código.
Los comentarios de funciones y bloques de código son comentarios de bloque, mientras que los comentarios de variables, por ejemplo, son simples comentarios de línea.
Las variables, archivos y carpetas también cuentan con nombres autodescriptivos que permiten deducir de forma directa su intención y cometido.

- *Backend*:
  Puesto que el desarrollo del *backend* se ha llevado a cabo con *Node.js*, así como el framework *Express.js*, se han seguido las convenciones de nombrado y guías de estilo de código aplicadas en *JavaScript*.

  Por tanto, los integrantes del equipo que utilizan *JavaScript* no deben usar tabulaciones, sino preferentemente **indentar con 4 espacios**.
  Además, para una mejor legibilidad, **la longitud máxima de las líneas de código es de 80 caracteres.**
  Las **llaves de aperturas de sentencias o funciones deben ir en la misma línea que dicha sentencia o cabecera de función**. Para la declaración de variables se ha utilizado *const* o *let*, según el caso, en lugar de *var*. Estas variables se declaran a razón de una por línea de la siguiente forma:

```JavaScript
const count = 10;
```

  Para comparar variables **se utiliza por defecto el comparador en JS utilizado para comparar tanto valor como tipo "==="** en vez de solo el que compara valor "==".
  Además, si las variables son *booleanas*, en vez de comparar con los valores *booleanos* "true" o "false" simplemente se compara el valor implícito de la variable de la siguiente forma:

```JavaScript
if(x) {
    // código
}
// ó
if(!x) {
    // código
}
```

Las **variables primitivas no son declaradas como objetos**, puesto que ralentizan el código.
El formato de nombrado de variables y funciones es **camelCase**. Por ejemplo:

```JavaScript
let numeroMovimientos = 10;
function hayJaque();
```

Sin embargo, para nombrar constructores o clases se utiliza PascalCase:

```JavaScript
ConstructorPieza();
```

Los valores constantes se declaran como **palabras mayúsculas separadas por un guión bajo** cada palabra:

```JavaScript
const NUMERO_MOVIMIENTOS_MAXIMO;
```

- *Frontend-Web:*
  Un aspecto importante de *React* es la manera de crear los componentes y exportarlos. Se ha decidido seguir el siguiente modelo:

```JavaScript
// Componente funcional
function MiComponente() {
 return (
```

```
  <div>
   {/* Contenido del componente */}
  </div>
 );
}

export default MiComponente;
```

Además se destaca que todos los componentes deben usar **camelCase** y **nombres descriptivos**.
Al usar el lenguaje de programación JavaScript se utiliza la misma guía de estilo mencionada en el apartado anterior.

- *Frontend-móvil*:
  Para construir una aplicación *web* en *Flutter* los *widgets* son los bloques de construcción fundamentales, y se utilizan para describir tanto la estructura como el diseño de la interfaz de usuario.

Ya que todo el código se estructura a través de *widgets*, a continuación se muestra un ejemplo de un *widget* en *Flutter*:

JavaScript
```
class MiWidget extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar( // Título en la parte superior del widget
    title: Text('Ejemplo'),
   ),
   body: Center(
    child: Text( // cuerpo centrado del widget con su texto en tamaño 24
     'Hola, mundo!',
     style: TextStyle(fontSize: 24.0),
    ),
   ),
  );
 }
}
```

Cabe destacar que para el nombrado de variables se ha utilizado lowerCamelCase, mientras que para el nombrado de clases porque el lenguaje lo requiere se ha utilizado UpperCamelCase. El nombrado de directorios y archivos están en lowercase_with_underscores.

3.2.2. Repositorios de control de versiones y sistema de gestión de incidencias

Se ha utilizado *GitHub* para gestionar el control de versiones. Se ha optado por crear una *Organización,* en la que se centralizan todos los repositorios involucrados en el proyecto, entre otras cosas.

En esta hay 3 repositorios principales, uno por subgrupo. Cualquier miembro del equipo puede acceder a cualquier repositorio, pero la visibilidad de estos es privada, luego cualquier persona ajena a la organización no puede acceder a ellos.
Se ha trabajado de manera desacoplada entre subgrupos, de forma que, por ejemplo, el subgrupo encargado del *frontend-web* no ha tenido que hacer cambios en el repositorio del *backend* en ningún momento.

En cada uno de estos repositorios, la rama *main* se ha utilizado para el código en producción. En el caso del *backend*, cada desarrollador ha podido trabajar en una rama propia (llamada `dev`) hasta que se comprobase el correcto funcionamiento del código a poner en producción. Solo en el momento de esta comprobación se puede realizar el *merge* con la rama principal de este repositorio.

A lo largo del desarrollo del proyecto se ha hecho uso de *Github Issues*. Durante los primeros pasos en el proyecto esta herramienta se ha utilizado únicamente para asignar tareas.
Más adelante, esta herramienta ha tenido el propósito, además, de registrar los *bugs* encontrados y ofrecer un espacio de discusión para abordarlos y solventarlos. Si bien la gestión de las tareas principalmente se ha llevado a cabo con el diagrama de *Gannt*, las incidencias de *Github* también se han empleado para llevar un registro más textual y preciso de dichas tareas.

3.2.3. Métodos, herramientas y técnicas para la construcción, el despliegue y las pruebas del software

Para construir el *software* en el que se trabaja en cualquiera de los subgrupos se ha empleado un entorno de desarrollo, *Visual Studio Code***,** y un gestor de control de versiones, *git*.
En cuanto al control de versiones, el *workflow* con el que se ha trabajado en los repositorios es el comentado en el apartado anterior.

Para automatizar el despliegue y la ejecución de *tests* de la aplicación tras realizar cambios en la rama principal de cualquiera de los repositorios se ha empleado *Github Actions*. Se ha optado por un *workflow CI/CD* (*Constant Integration/Constant Development*) para que el usuario sea privado de nuestro servicio lo mínimo posible.

Pruebas del software

- Tests de aplicación
  Para comprobar el correcto funcionamiento del *software* desarrollado se han diseñado tests automáticos haciendo uso de *Github Actions*. El *backend* ha sido probado de manera exhaustiva haciendo uso de *Postman* para asegurarse, primeramente, de la

corrección de los movimientos que proporciona. Todos los movimientos siguen las reglas oficiales del ajedrez. Además, ha sido probado el *API* que proporciona, pues de este depende la correcta interacción entre el *frontend* y el *backend*.

- Tests unitarios
  No se han realizado tests unitarios debido a la errónea planificación para esta tarea en ninguno de los departamentos (*véase sección 5.2*).

3.2.4. Construcción e integración del software

Respecto a la construcción del *software*, en el desarrollo de la aplicación móvil se ha partido de un proyecto de iniciación proporcionado por el mismo *Flutter*, el cual contiene una carpeta llamada *android* con un *.gradle* que se encarga de automatizar la construcción, la prueba y el despliegue de software.

En el desarrollo web con *React* la construcción e integración de la aplicación se ha gestionado automáticamente. Partiendo del código escrito en formato *.jsx,* la traducción a ficheros *JavaScript* y *HTML* se realiza automáticamente, y la compilación del proyecto se realiza con **"***npm run build"*, la herramienta integrada del framework *create-react-app*, de *Facebook Open Source*.

En el desarrollo del *backend* no ha sido necesario compilar, ya que *Node.js* es capaz de ejecutar directamente archivos *JavaScript* que acaban de ser modificados mediante el uso de *node daemon* siempre que el servidor esté abierto.

La integración de *software* se ha realizado de forma que tanto el equipo que se encarga de *frontend-móvil* como *frontend-web* tengan el repositorio del *backend* en sus carpetas. De esta manera pueden probarlo realizando llamadas a él para comprobar la conectividad y su correcto funcionamiento.

3.2.5. Despliegue del *software*

Para el proceso de despliegue de nuestra aplicación se ha utilizado un servicio de *Cloud Service*, una herramienta de infraestructura en la nube. Este enfoque nos proporciona la escalabilidad, flexibilidad, seguridad y fiabilidad necesarias para ofrecer una experiencia de calidad a nuestros usuarios.

A continuación, se detalla paso a paso el *pipeline* de *CI/CD* que despliegue de la aplicación:

1. En uno de los repositorios con el código fuente (p.ej. el correspondiente al *backend*), se realiza un *push* a la rama *deploy*. El propósito de estas ramas es el de almacenar la versión en despliegue de la aplicación. Los cambios publicados en esas ramas son los que verá el usuario final.

2. Una vez realizado el push, un *trigger* de *Cloud Build* (otra herramienta de *Google Cloud Platform*) detecta ese cambio en la rama *deploy*.

3. Dicho *trigger* es ejecutado en remoto, en las máquinas de la nube de *Google*, y se encarga de construir y subir un contenedor al repositorio integrado de contenedores *Docker* de *GCP*, *Container Registry*. Para construir y ejecutar dicho contenedor emplea un *Dockerfile* incluido en el directorio raíz del repositorio.

4. El mismo *trigger* se encarga de desplegar el contenedor subido al *Container Registry* en la instancia o servicio de *Cloud Service*.

5. Si todo ha ido bien, la aplicación habrá sido desplegada automáticamente en la instancia de *Cloud Run*. Los usuarios podrán acceder a la aplicación mediante sus navegadores web habituales.

El despliegue se puede entender también con un esquema en la *figura 1*:

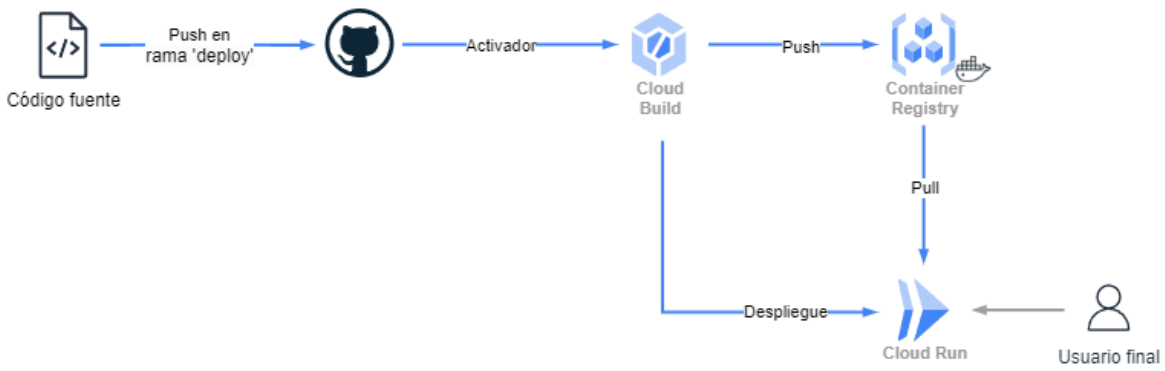


Figura 1: Pipeline del CI/CD

3.2.6. Guías para el diseño gráfico de las *GUI*

A pesar de que no se ha seguido ninguna guía de diseño específica para el desarrollo de las *GUI*, sí que se han tenido en cuenta ciertas reglas:

- Agrupación de los elementos: si por lógica van juntos, físicamente también.
- Orden de los elementos y los grupos: se debe pensar el orden y el flujo de las acciones.
- Decoración: se usan cajas para agrupar elementos con lógica, y tipografía para enfatizar en sitios como las cabeceras, siempre sin excederse.
- Alineación de los elementos: para facilitar la búsqueda y legibilidad de los mismos.
- Uso del espacio en blanco: para separar, organizar o resaltar los distintos elementos.
- Uso de iconos: para acciones comunes dentro de un sistema, como ir al menú principal, volver atrás o deshacer, entre otros.
- Diseño estéticamente agradable y legible: uso de colores vistosos y agradables pero que no dificulten la legibilidad. Evidentemente, no se usan fondos de color similar al texto.

Además de estas reglas, también se han tenido en consideración nociones básicas sobre los principios de la *Gestalt*.

3.2.7. Control de calidad del código

Se ha llevado un control de calidad estricto para el proyecto, por lo que ningún fragmento de código se ha publicado en la rama de producción de los respectivos repositorios de cada subgrupo sin antes haber sido probado completamente y haber sido consensuado por los miembros.

3.2.8. Entrega de resultados a clientes

Primeramente, los clientes han recibido una memoria técnica del proyecto, con detalles relevantes acerca del diseño de alto nivel o la documentación que indica cómo hacer uso del sistema creado. El archivo *.apk* correspondiente a la aplicación móvil ha sido utilizado para realizar la demostración del funcionamiento de la aplicación. En el caso de la versión *web* el cliente es capaz de ejecutar la aplicación en un navegador *Google Chrome* desde cualquier ordenador, puesto que el juego funcional se ha entregado en la plataforma en la que se ha desarrollado desde un principio.

Por otro lado, dichos clientes también recibirán los permisos necesarios para acceder al código fuente mediante *GitHub*, con el objetivo no solo de ver la implementación de cada funcionalidad, sino también de tener la capacidad de mejorar o actualizar, en función de los intereses de los propios clientes, el juego entregado.

3.3. Calendario del proyecto, división del trabajo, coordinación, comunicaciones, monitorización y seguimiento

El responsable de la división del trabajo, la coordinación, las comunicaciones, la monitorización y el seguimiento es el director del proyecto, Óscar Brizuela.

3.3.1. Tareas a realizar del proyecto

La tarea principal a desarrollar en el proyecto ha sido un juego de ajedrez completamente funcional que cumple con los requisitos especificados tanto en la plantilla técnica como en este documento (*tablas 2* y *3*) y que es compatible para jugar tanto en *web* como en dispositivos móviles.

Esta compleja tarea la podemos subdividir en tareas de menor complejidad, como se muestra a continuación:

- **Tormenta de ideas y análisis de requisitos**. Durante la etapa inicial del proyecto, se realizó una sesión de lluvia de ideas y análisis de requisitos para seleccionar la idea que ha sido implementada y definir los requisitos de la aplicación.

- **Diseño de la aplicación**. Esta tarea consistió en definir especificaciones técnicas de la aplicación, arquitecturas, estructuras de datos, interfaces…

- **Desarrollo del *backend*.** Esta tarea implicó la creación del *backend* de la aplicación, un componente clave encargado de gestionar la lógica del juego, gestionar usuarios, persistir datos, proporcionar seguridad y autenticación…

- **Desarrollo de la aplicación móvil.** Constituye una de las etapas más extensas del proceso, puesto que en esta tarea se ha llevado a cabo el desarrollo de la versión destinada a dispositivos móviles.

- **Desarrollo de la aplicación *web*.** De una manera similar al desarrollo de la aplicación móvil, la consecución de esta tarea ha permitido presentar una versión del sistema disponible para *web*.

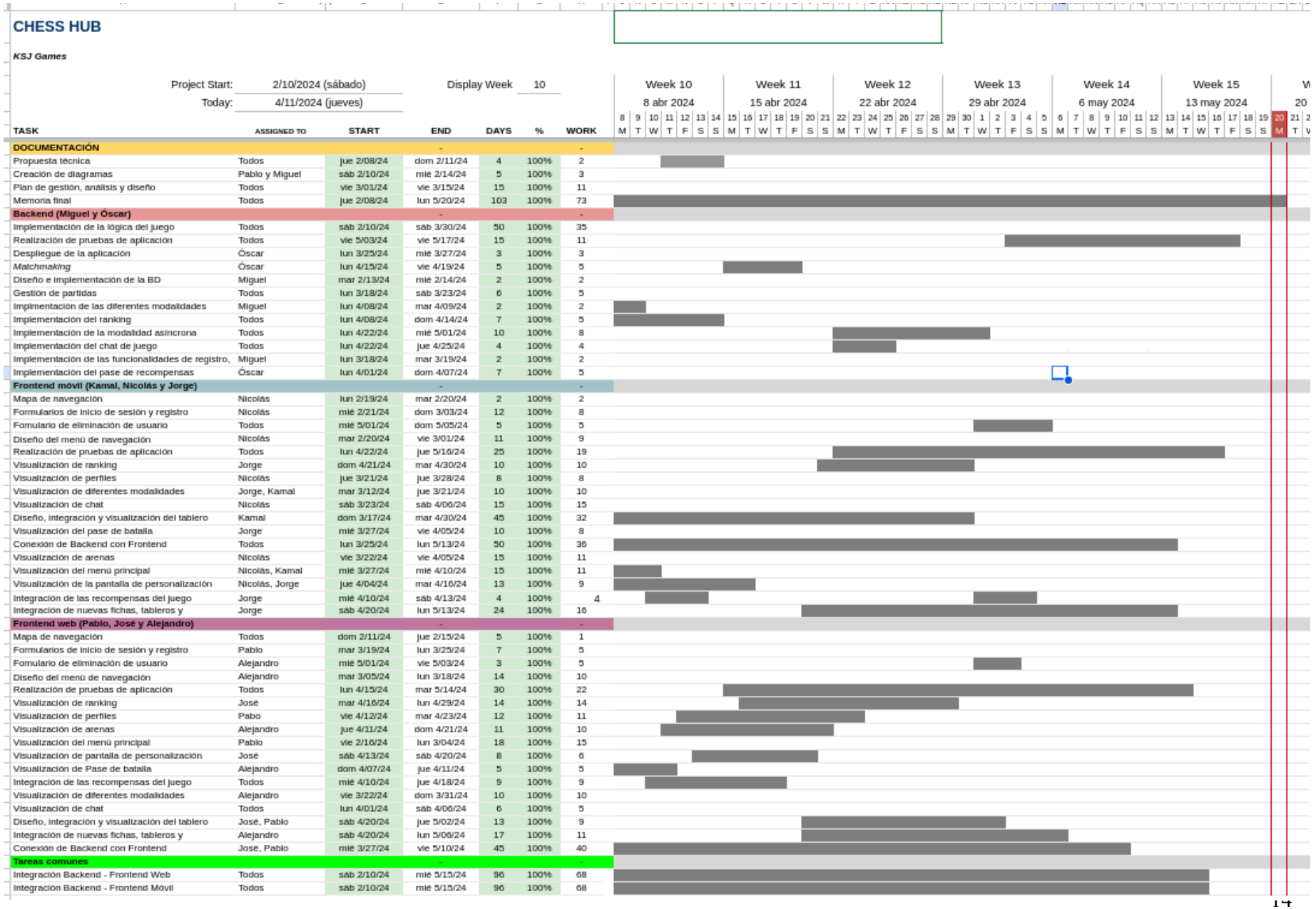
- **Integración de la aplicación móvil y *backend***. Paralelamente al desarrollo de la aplicación móvil se ha llevado a cabo la integración con el *backend*, uniendo así la lógica del juego que representa la funcionalidad con la interfaz de usuario.

- **Integración de la aplicación web y *backend*.** De la misma forma, de manera paralela al desarrollo de la aplicación *web* se ha llevado a cabo la integración con el *backend*, uniendo así la lógica del juego que representa la funcionalidad con la interfaz de usuario.

- **Pruebas de la aplicación.** Constantemente se han ido realizando pruebas que verifican el correcto funcionamiento de cada uno de los tres grandes bloques que componen el proyecto y el cumplimiento con los requisitos. Las pruebas se han realizado tanto en la versión móvil como en la versión *web*.

- **Documentación de la aplicación.** A lo largo de todo el proyecto, se ha documentado todo lo necesario para facilitar el uso y mantenimiento de la aplicación. Además, se ha asegurado de que cualquier persona que se incorpore al proyecto cuente con toda la información necesaria para integrarse al equipo de desarrollo con la mayor preparación posible.

En la *figura 2* se puede apreciar el diagrama de *Gantt* correspondiente a las últimas semanas del proyecto (por petición del tutor del mismo), actualizado a fecha de la entrega de este mismo documento. No obstante, el diagrama de Gantt con todos sus cambios en tiempo real se encuentra aquí.



TASK	ASSIGNED TO	START	END	DAYS	%	WORK
DOCUMENTACIÓN			-			-
Propuesta técnica	Todos	jue 2/08/24	dom 2/11/24	4	100%	2
Creación de diagramas	Pablo y Miguel	sáb 2/10/24	mié 2/14/24	5	100%	3
Plan de gestión, análisis y diseño	Todos	vie 3/01/24	vie 3/15/24	15	100%	11
Memoria final	Todos	jue 2/08/24	lun 5/20/24	103	100%	73
Backend (Miguel y Óscar)			-			-
Implementación de la lógica del juego	Todos	sáb 2/10/24	sáb 3/30/24	50	100%	35
Realización de pruebas de aplicación	Todos	vie 5/03/24	vie 5/17/24	15	100%	11
Despliegue de la aplicación	Óscar	lun 3/25/24	mié 3/27/24	3	100%	3
Matchmaking	Óscar	lun 4/15/24	vie 4/19/24	5	100%	5
Diseño e implementación de la BD	Miguel	mar 2/13/24	mié 2/14/24	2	100%	2
Gestión de partidas	Todos	lun 3/18/24	sáb 3/23/24	6	100%	5
Implmentación de las diferentes modalidades	Miguel	lun 4/08/24	mar 4/09/24	2	100%	2
Implementación del ranking	Todos	lun 4/08/24	dom 4/14/24	7	100%	5
Implementación de la modalidad asíncrona	Todos	lun 4/22/24	mié 5/01/24	10	100%	8
Implementación del chat de juego	Todos	lun 4/22/24	jue 4/25/24	4	100%	4
Implementación de las funcionalidades de registro,	Miguel	lun 3/18/24	mar 3/19/24	2	100%	2
Implementación del pase de recompensas	Óscar	lun 4/01/24	dom 4/07/24	7	100%	5
Frontend móvil (Kamal, Nicolás y Jorge)			-			-
Mapa de navegación	Nicolás	lun 2/19/24	mar 2/20/24	2	100%	2
Formularios de inicio de sesión y registro	Nicolás	mié 2/21/24	dom 3/03/24	12	100%	8
Formulario de eliminación de usuario	Todos	mié 5/01/24	dom 5/05/24	5	100%	5
Diseño del menú de navegación	Nicolás	mar 2/20/24	vie 3/01/24	11	100%	9
Realización de pruebas de aplicación	Todos	lun 4/22/24	jue 5/16/24	25	100%	19
Visualización de ranking	Jorge	dom 4/21/24	mar 4/30/24	10	100%	10
Visualización de perfiles	Nicolás	jue 3/21/24	jue 3/28/24	8	100%	8
Visualización de diferentes modalidades	Jorge, Kamal	mar 3/12/24	jue 3/21/24	10	100%	10
Visualización de chat	Nicolás	sáb 3/23/24	sáb 4/06/24	15	100%	15
Diseño, integración y visualización del tablero	Kamal	dom 3/17/24	mar 4/30/24	45	100%	32
Visualización del pase de batalla	Jorge	mié 3/27/24	vie 4/05/24	10	100%	8
Conexión de Backend con Frontend	Todos	lun 3/25/24	lun 5/13/24	50	100%	36
Visualización de arenas	Nicolás	vie 3/22/24	vie 4/05/24	15	100%	11
Visualización del menú principal	Nicolás, Kamal	mié 3/27/24	mié 4/10/24	15	100%	11
Visualización de la pantalla de personalización	Nicolás, Jorge	jue 4/04/24	mar 4/16/24	13	100%	9
Integración de las recompensas del juego	Jorge	mié 4/10/24	sáb 4/13/24	4	100%	4
Integración de nuevas fichas, tableros y	Jorge	sáb 4/20/24	lun 5/13/24	24	100%	16
Frontend web (Pablo, José y Alejandro)			-			-
Mapa de navegación	Todos	dom 2/11/24	jue 2/15/24	5	100%	1
Formularios de inicio de sesión y registro	Pablo	mar 3/19/24	lun 3/25/24	7	100%	5
Formulario de eliminación de usuario	Alejandro	mié 5/01/24	vie 5/03/24	3	100%	5
Diseño del menú de navegación	Alejandro	mar 3/05/24	lun 3/18/24	14	100%	10
Realización de pruebas de aplicación	Todos	lun 4/15/24	mar 5/14/24	30	100%	22
Visualización de ranking	José	mar 4/16/24	lun 4/29/24	14	100%	14
Visualización de perfiles	Pablo	vie 4/12/24	mar 4/23/24	12	100%	11
Visualización de arenas	Alejandro	jue 4/11/24	dom 4/21/24	11	100%	10
Visualización del menú principal	Pablo	vie 2/16/24	lun 3/04/24	18	100%	15
Visualización de pantalla de personalización	José	sáb 4/13/24	sáb 4/20/24	8	100%	6
Visualización de Pase de batalla	Alejandro	dom 4/07/24	jue 4/11/24	5	100%	5
Integración de las recompensas del juego	Todos	mié 4/10/24	jue 4/18/24	9	100%	9
Visualización de diferentes modalidades	Alejandro	vie 3/22/24	dom 3/31/24	10	100%	10
Visualización de chat	Todos	lun 4/01/24	sáb 4/06/24	6	100%	5
Diseño, integración y visualización del tablero	José, Pablo	sáb 4/20/24	jue 5/02/24	13	100%	9
Integración de nuevas fichas, tableros y	Alejandro	sáb 4/20/24	lun 5/06/24	17	100%	11
Conexión de Backend con Frontend	José, Pablo	mié 3/27/24	vie 5/10/24	45	100%	40
Tareas comunes			-			-
Integración Backend - Frontend Web	Todos	sáb 2/10/24	mié 5/15/24	96	100%	68
Integración Backend - Frontend Móvil	Todos	sáb 2/10/24	mié 5/15/24	96	100%	68

Figura 2: diagrama de Gantt actualizado a 20/05/202

3.3.2. División del trabajo

Respecto a la división del trabajo, esta corresponde a los tres grandes bloques mencionados anteriormente. A continuación se muestra una tabla que representa qué tarea corresponde a cada subgrupo, aunque puede darse el caso que una tarea corresponda a todo el equipo.

TAREA	***Backend***	***Frontend-móvil***	***Frontend-web***
Tormenta de ideas y análisis de requisitos		**X**	
Diseño de la aplicación		**X**	
Desarrollo de la lógica del sistema	**X**		
Desarrollo de la aplicación móvil		**X**	
Desarrollo de la aplicación *web*			**X**
Integración de la aplicación móvil y *backend*	**X**	**X**	
Integración de la aplicación *web* y *backend*	**X**		**X**
Pruebas de la aplicación		**X**	
Documentación de la aplicación		**X**	

Tabla 1: División de tareas para cada subgrupo

3.3.3. Coordinación y comunicaciones

Con el objetivo de mantener un seguimiento continuo todas las semanas, cada lunes a las 11:00h (*CET*) se ha realizado una reunión en un servidor de *Discord* donde hemos puesto al día los avances conseguidos la semana anterior. Esta reunión también ha servido para resolver dudas respecto a la integración del código de los distintos subequipos en casos en los que ha sido necesario, o respecto a los métodos, parámetros o rutas que deben ser utilizados por los subequipos.

Todos los días de práctica presencial también han servido para comprobar el correcto funcionamiento del proyecto en ese momento. No obstante, cuando ha sido necesaria alguna reunión adicional, llevada a cabo por integrantes tanto del mismo subequipo como de subequipos diferentes, está también ha tenido lugar en el mismo servidor de *Discord*, el cual tiene habilitados varios canales de voz para evitar solapamientos. Por otro lado, el canal que se ha utilizado para dudas menos relevantes o puestas en común no demasiado estructurales ha

sido *WhatsApp*, a través del grupo (si el asunto incumbe a todo el equipo) o a través del mensaje privado.

Las horas de reuniones también constan como invertidas en el proyecto dentro de la hoja de cálculo que recoge las horas totales de cada integrante. Al final de cada una de estas reuniones se ha hecho entrega, por parte del director del proyecto, de las tareas realizadas y las horas dedicadas por cada integrante a través del formulario destinado para ello disponible en el *Moodle* de la asignatura.

Finalmente, respecto a la gestión del equipo humano, se abordan las siguientes medidas:

- En la resolución de conflictos, el equipo se ha esforzado por identificar y solucionar cualquier inconveniente lo antes posible. Han surgido diversos tipos de conflictos, como dificultades técnicas que han afectado al rendimiento de un subgrupo y han causado retrasos en otros aspectos del proyecto. Se han abordado estos problemas con la máxima prioridad posible. Este tipo de inconveniente ha sido el más común, pero mediante una comunicación mejorada y una rápida identificación de errores, hemos podido resolverlo eficazmente.

- El director del grupo Óscar Brizuela se ha encargado semanalmente de revisar las horas dedicadas de cada integrante del grupo junto con la tarea realizada. Con esta información ha aportado una pequeña retroalimentación que resalta los logros alcanzados y corrige los errores y las zonas en las que puede mejorar

4. Análisis y diseño del sistema

4.1. Análisis de requisitos

Los requisitos del sistema son los siguientes:

Código	Requisito funcional
RF-1	El sistema ofrece la posibilidad de **registrarse** para monitorizar el nivel del usuario y almacenar las recompensas que se pueden obtener jugando.
RF-1.1	El sistema permite a los usuarios **darse de baja**.
RF-2	El sistema ofrece la posibilidad de **jugar partidas *en línea***. Una partida en línea consiste en 2 usuarios registrados jugando desde navegadores distintos
RF-3	El sistema ofrece la posibilidad de **jugar partidas *locales***. Una partida local ocurre en un solo navegador y los dos jugadores pueden mover por turnos.
RF-4	El sistema permite jugar dos modalidades de partida: **modalidad asíncronas (por correspondencia)** y **modalidad síncrona**. Una partida ***por correspondencia*** se puede abandonar en cualquier momento para retomarla posteriormente donde se dejó. Sin temporizador. Las partidas **síncronas** se pueden jugar a diferentes *ritmos*. Estos ritmos de partida síncrona son *blitz, bullet* y *rapid*, cada ritmo con un temporizador inicial distinto.
RF-5	El sistema garantiza que aquellos jugadores que no se hayan registrado solo tengan la posibilidad de jugar partidas *en **local***: el jugador y su contrincante tendrán que jugar desde el **mismo dispositivo**.
RF-6	El modo de juego *online* cuenta con un sistema de ***ranking*** para realizar los emparejamientos en partidas *online*.
RF-6.1	El sistema de ***ranking*** está basado en puntos (también llamado "**ELO**"). Este "ELO" se otorga **en función de las partidas ganadas y el "ELO" del contrincante**. En caso de perder la partida, se resta puntuación al usuario y, en caso de ganar, se suma puntuación. La puntuación obtenida tras un empate dependerá del nivel de ambos jugadores.
RF-6.2	El *ranking* global de todos los jugadores está dividido en **arenas**. En una arena se agrupan a los jugadores con nivel ("ELO") similar para realizar el emparejamiento de partidas.
RF-6.2.1	Las diferentes arenas se representan a través de diferentes **aspectos visuales** del tablero. Estos serán: *Madera*, *Mármol*, *Oro*, *Esmeralda* y *Diamante*

RF-7	El sistema permite que, en las partidas *online*, el usuario pueda comunicarse con el rival mediante un ***chat* interactivo**.
RF-8	El sistema cuenta con un **"pase de recompensas"**, consistente en recompensar a los usuarios que ganan partidas.
RF-8.1	Los puntos que se requieren para desbloquear recompensas en el **"pase de recompensas"** se otorgan a los usuarios cuando juegan partidas. El usuario ganará puntos de recompensa independientemente de si gana o pierde la partida. Al usuario que gane la partida se le premiará con más puntos que al usuario que pierda la partida.
RF-8.2	El sistema **mostrará visualmente el ranking** del jugador usando el aspecto del tablero. Cuantas **más partidas gane un usuario más recompensas** puede desbloquear. Estas recompensas podrán ser piezas de diferentes personalizaciones o la capacidad para interactuar con diferentes emoticonos.
RF-9	El sistema **mostrará visualmente el ranking** del jugador usando el aspecto del tablero.

Tabla 2: Requisitos funcionales del sistema

Código	**Requisito no funcional**
RNF-1	El **aspecto** del juego es estéticamente **agradable** y dinámico, alterable.
RNF-2	El sistema es **seguro**
RNF-3	El sistema es **estable**, sin caídas.
RNF-4	El sistema es **rápido**.
RNF-5	La aplicación móvil será multiplataforma, por lo que estará disponible para ***Android***, mientras que la aplicación web estará disponible para el navegador ***Google Chrome***.

Tabla 3: Requisitos no funcionales del sistema

4.2. Diseño del sistema

4.2.1 Tecnologías empleadas

- Tecnologías empleadas:

- *Backend:* la base del juego de ajedrez se ha implementado en *Node.js*, utilizándose para la lógica del juego, gestión de usuarios, almacenamiento de datos y comunicación con las aplicaciones cliente. Se han empleado *frameworks* como *Express.js* para manejar las solicitudes *HTTP* y *Socket.IO* para la comunicación en tiempo real entre el servidor y los clientes.

- *Frontend-web:* se ha utilizado el *framework React* para desarrollar la versión *web*, siendo responsable de la interfaz de usuario interactiva, incluyendo el tablero de ajedrez, menús, opciones de configuración y la interacción con el servidor.

- *Frontend-móvil:* se ha empleado *Flutter* para crear la aplicación móvil, por lo que se proporciona una interfaz de usuario nativa para dispositivos *Android*, asegurando un rendimiento óptimo y una experiencia de usuario fluida. Se ha desarrollado la misma funcionalidad que en la versión web, incluyendo el tablero de ajedrez, la lógica del juego y la comunicación con el servidor.

En la sección 4.2.2 se presenta la vista de módulos primaria. En esta se destacan los principales módulos del sistema: *frontend-web*, *frontend-móvil* y *backend*. El *backend* proporciona una interfaz en forma de *API* que permite comunicarse con ambos *frontends*. Estos hacen uso de la interfaz para realizar las operaciones necesarias. El *frontend-web* es el encargado de mostrar la interfaz de usuario en el navegador web que recogerá las peticiones de los usuarios y se comunicará con la *API* proporcionada por el backend. El *frontend-móvil* funcionará de forma similar al otro *frontend*.

4.2.2. Diagramas del sistema

En este punto se muestran las distintas vistas de módulos para comprender la estructura del proyecto.

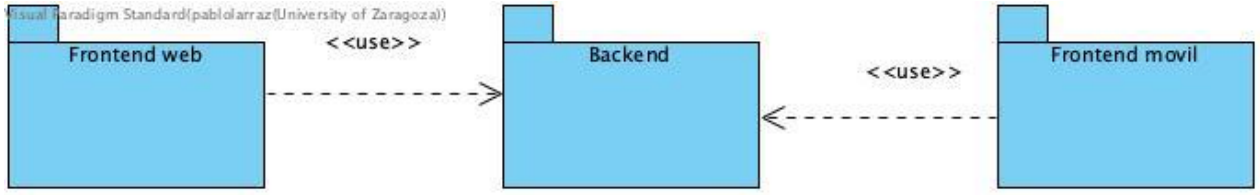


Figura 3: Vista de módulos primaria

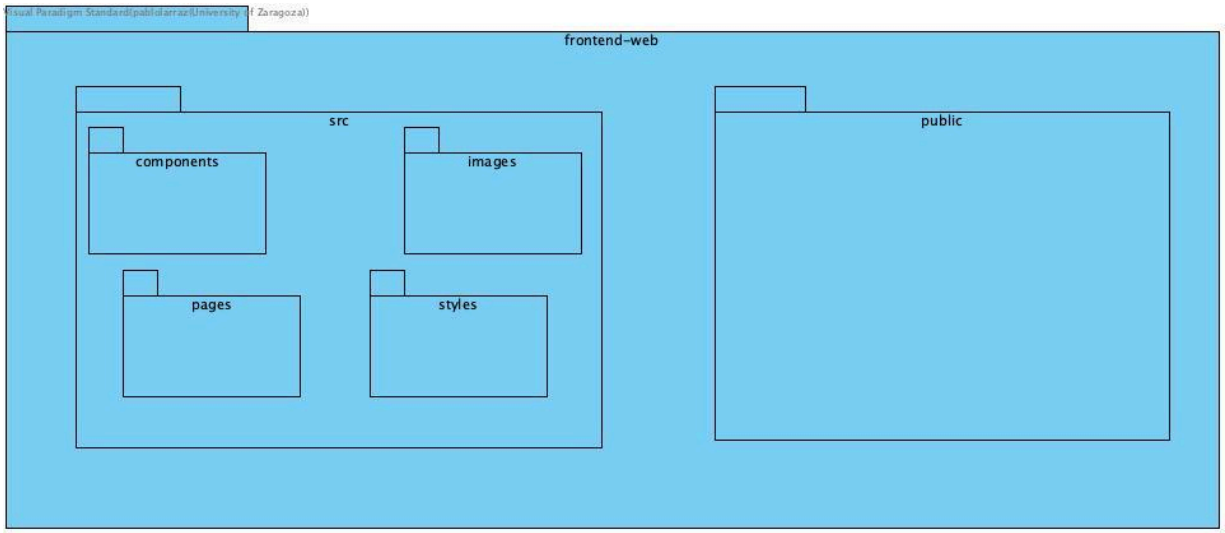


Figura 4: Vista del módulo frontend-web

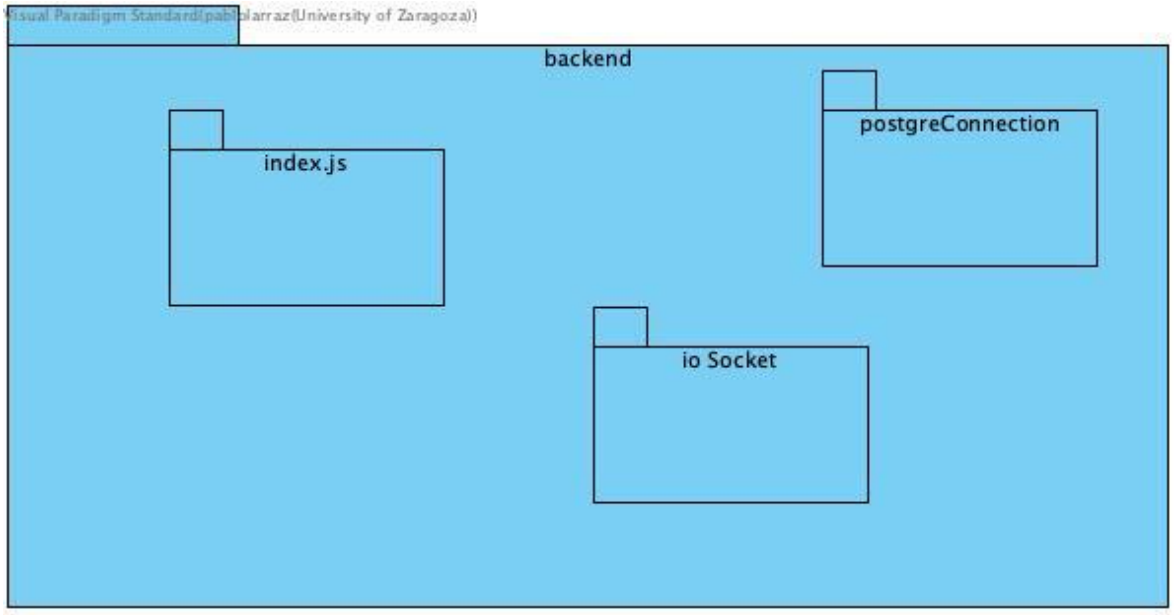


Figura 5: Vista del módulo backend

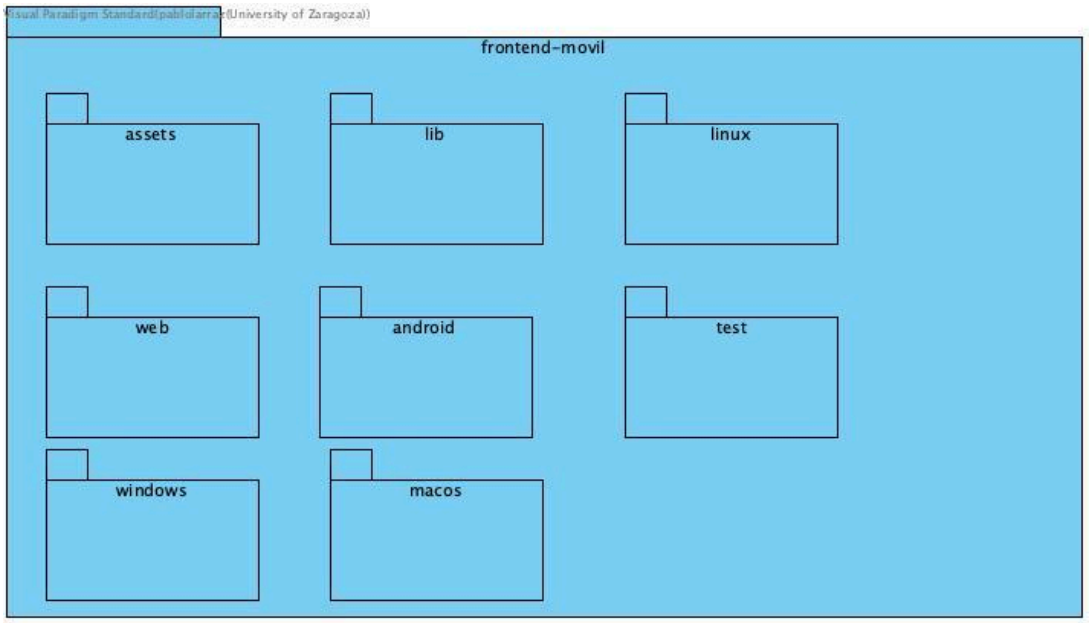


Figura 6: Vista del módulo frontend-móvil

Asimismo se muestra el diagrama de componentes. Como se puede apreciar, el *backend* proporciona una interfaz *API* como punto de encuentro entre los *frontends* y el *backend*.

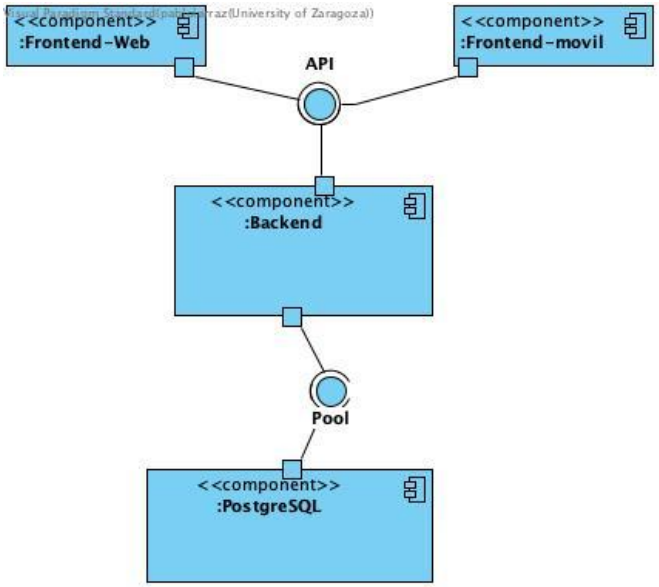


Figura 7: Diagrama de componente-conector

A continuación se describe el diagrama de despliegue en el que el *backend* se despliega sobre una máquina en *GCP*. En el móvil se podrá usar un archivo *.apk* de la aplicación. Sobre el navegador *web* correrá la aplicación *React*.

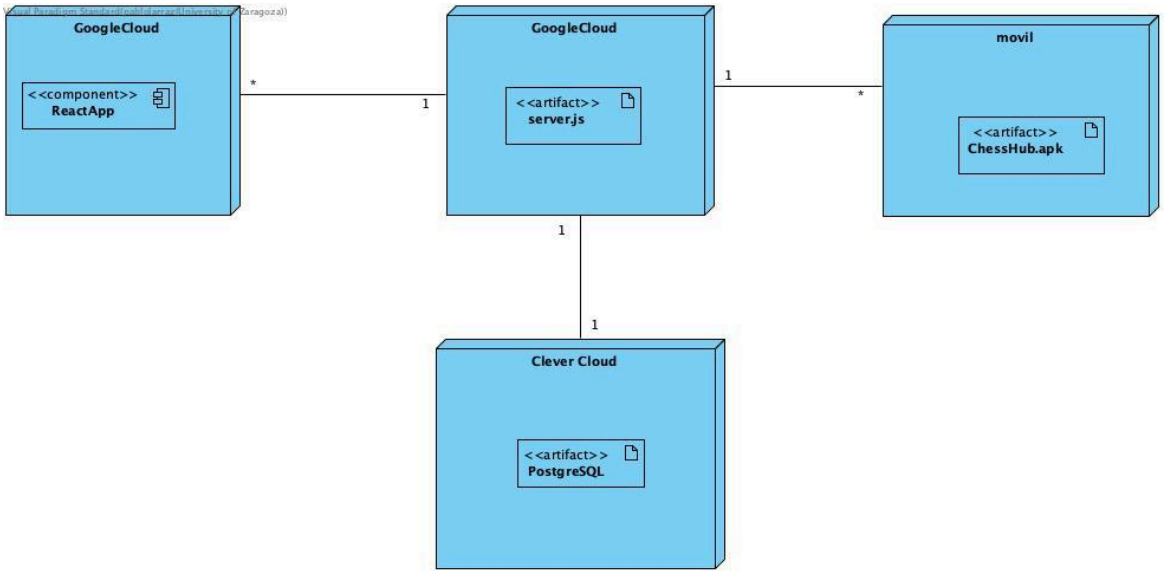


Figura 8: Diagrama de despliegue

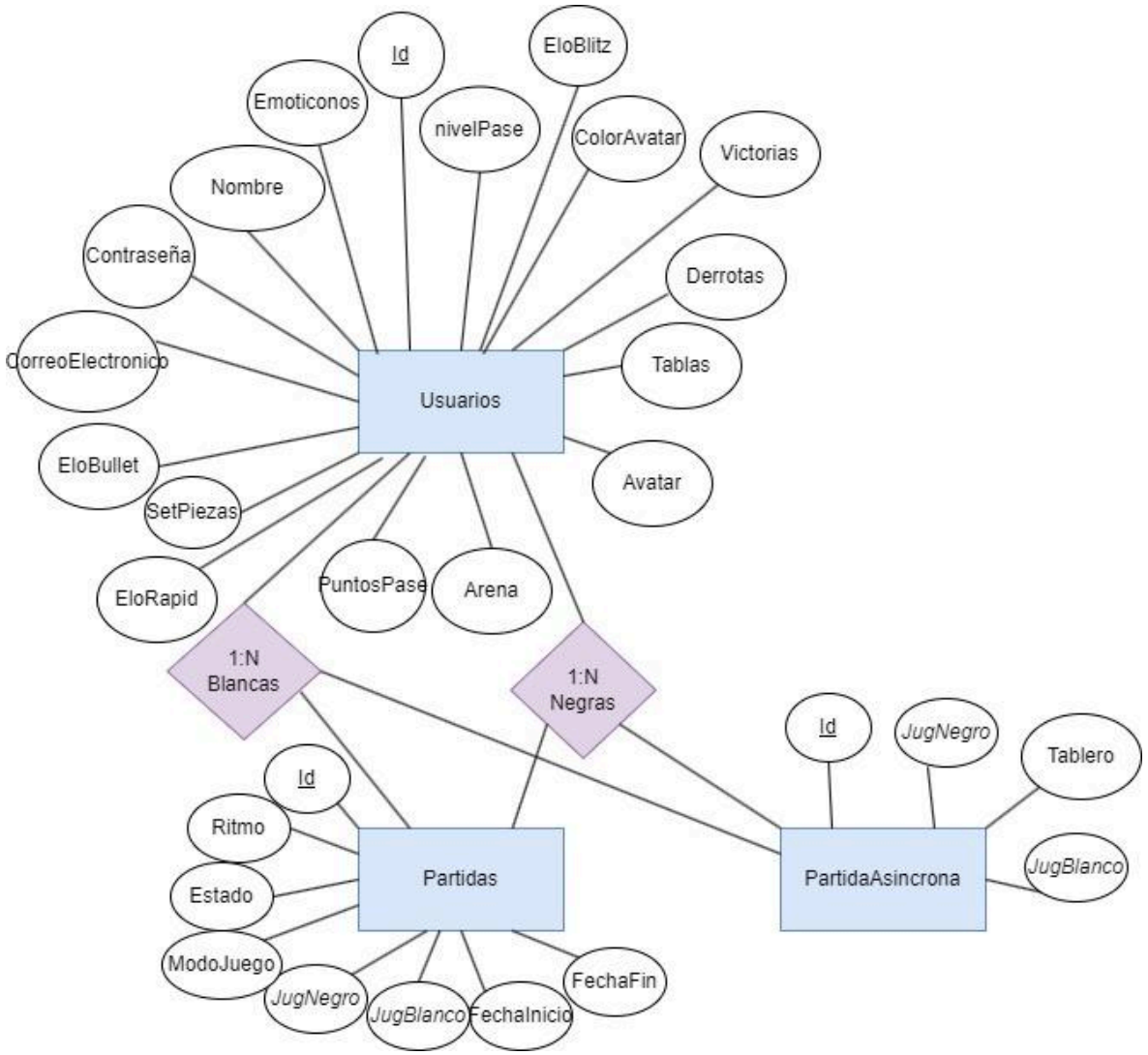


Figura 9: modelo de datos del sistema

Las principales decisiones de diseño de este modelo de datos se detallan a continuación:

- Centralización de información del usuario:
    - La entidad *Usuarios* agrupa toda la información relevante de cada usuario, incluyendo datos personales, configuraciones del juego, estadísticas y preferencias visuales. Esto permite una gestión integral y eficiente del perfil del usuario, facilitando el acceso y la actualización de sus datos.
- Modelado de Partidas:
    - La entidad *Partidas* está diseñada para todos los detalles importantes de cada partida, como el estado, el ritmo, el modo de juego y las fechas de inicio y fin. Además, se incluyen referencias a los usuarios que juegan con las piezas blancas y negras, lo que permite rastrear fácilmente quién participó en cada partida y en qué rol.
- Diferenciación de tipos de partidas:

    - La entidad *PartidaAsincrona* se usa para modelar partidas que no se juegan en tiempo real, diferenciándose de las partidas estándar. Esto permite manejar distintos tipos de partidas con sus propias características y requisitos, ofreciendo una mayor flexibilidad en la aplicación.
- Relaciones claras y específicas:
    - Las relaciones *1:N* entre *Usuarios* y *Partidas* (a través de *Blancas* y *Negras*) indican que cada usuario puede participar en múltiples partidas y que cada partida tiene dos jugadores (uno con piezas blancas y otro con piezas negras). Esto facilita el seguimiento de la participación del usuario en distintas partidas y permite registrar estadísticas detalladas.
- Flexibilidad en configuraciones y estadísticas:
    - La inclusión de varios atributos en la entidad *Usuarios* relacionados con configuraciones de juego (como *ELO* en diferentes modalidades, preferencias de piezas y avatar) y estadísticas (victorias, derrotas, tablas) permite personalizar la experiencia del usuario y mantener un registro detallado de su desempeño.
- Adaptabilidad y Escalabilidad:
    - Al diseñar el esquema con entidades separadas y relaciones bien definidas, se permite una fácil expansión del sistema para incluir nuevas características o tipos de partidas en el futuro. La estructura es modular, lo que facilita la adición de nuevos atributos o relaciones sin una reestructuración completa de la base de datos.

4.2.3. Arquitectura global del sistema

Se ha establecido una arquitectura cliente-servidor, donde los clientes envían solicitudes al servidor para realizar movimientos, unirse a partidas y obtener información actualizada del juego.

Se han utilizado *sockets* para permitir la comunicación en tiempo real entre el servidor y los clientes, lo que permite una experiencia de juego fluida y sincronizada.

Backend

El backend desarrollado en *Node.js* se ha encargado de la lógica del juego, la gestión de los usuarios y las partidas, y la comunicación con los clientes. El código del *backend* está desarrollado en diferentes carpetas:

- `piezas`: contiene todas las clases de las diferentes piezas de la aplicación.
- `routes`: todas las rutas necesarias para la comunicación del *backend* con el *frontend-web* y *frontend-móvil.*
- `database:` contiene la base de datos utilizada en el proyecto.

Frontend-web

Lo primero a destacar sobre el código que conforma el *frontend-web* es la existencia de varias carpetas claves para el desarrollo.

- `src`: carpeta principal que está formada por todas las que se comentan a continuación.
- `components`: componentes de *React* que se podrán reutilizar en distintas páginas.
- `images`: todas las imágenes a usar en la página *web*. Contiene subcarpetas en las que se agrupan elementos relacionados como los diferentes tableros del juego y las diferentes piezas.
- `pages`: páginas *web* que muestran las propias rutas
- `styles`: archivos *CSS* para todos aquellos componentes y páginas que lo requieran.

El mapa de navegación correspondiente al diseño *web* se encuentra en el Anexo III.

Frontend-móvil

El código del frontend-móvil también se encuentra dividido en distintas carpetas:

- `lib`: carpeta principal que contiene otras carpetas y los ficheros *.dart* que implementan las pantallas y los *widgets* utilizados.
- `assets`: carpeta que contiene todos los archivos que no son código referenciados en el proyecto, como imágenes, fuentes de letra, sonidos, etc.

También hay una carpeta por cada página de la aplicación con sus ficheros *.dart* necesarios y carpetas con *widgets* utilizados a lo largo del proyecto.

El mapa de navegación correspondiente a la app móvil se puede encontrar en el Anexo IV.

Por último, en el siguiente enlace se encuentra la documentación de la *API* utilizada en la comunicación entre el *backend* y ambos *frontends*. Esta documentación se ha obtenido a través de una colección de *endpoints* creada con *Postman*, de forma que los integrantes de los *frontends* puedan visualizar rápida y fácilmente las *URLs* a las que tienen que acceder, así como los métodos de acceso y parçametros/cuerpos de solicitud que deben utilizar.

https://documenter.getpostman.com/view/33151640/2sA3QmFFuf

5. Memoria del proyecto

5.1. Inicio del proyecto

Desde el primer momento las diversas tareas del proyecto fueron llevadas a cabo de manera conjunta gracias a los canales de comunicación implantados. Estos canales han sido *Discord* y, para cuestiones no fundamentales, *WhatsApp*. La correcta definición de las tecnologías a utilizar y la clara división de los trabajos fue clave para empezar el proyecto sin imprevistos ni confusiones. Además, como se ha comentado previamente, fue necesario un tiempo considerable invertido en formación de las diferentes tecnologías, recursos y conceptos. No obstante, esta formación no solo ha tenido lugar durante el inicio del proyecto, sino también a lo largo de la consecución del mismo.

5.2. Ejecución y control del proyecto

Dentro del grupo hemos funcionado divididos en 3 subgrupos: *backend*, *frontend-web* y *frontend-móvil*. Cada subgrupo ha ido trabajando independientemente del resto y ha ido publicando sus avances, comunicándose internamente y con el resto de subgrupos cuando fuese necesario poner ideas de acuerdo o aclarar conceptos de diseño.

No obstante, la constante comunicación entre el *backend* y los *frontends* (tanto *web* como móvil) ha sido clave a la hora de desarrollar los diseños y funcionalidades propuestos por el backend. Con el objetivo de evitar confusiones y ambigüedades, fue necesario crear una documentación de la *API* utilizada mediante *Postman* para que ambos frontends pudieran saber a qué rutas acceder, con qué parámetros y con qué *bodies* para desarrollar una nueva funcionalidad (*véase parte final de la sección 4.2.3*).

El grupo ha planteado una serie de hitos con fechas objetivo, con la intención de tener preparados ciertos requisitos en los días acordados. A pesar de que no se ha llegado al 100% de los hitos en los tiempos establecidos, esta gestión sí ha servido para contar con cierto margen en cuanto a la realización de pruebas durante la etapa final. Cuando algún integrante del equipo se ha encontrado por detrás de las fechas previstas ha sido necesario redireccionar recursos a las tareas que iban con retraso.

La forma de trabajar de todos los integrantes del equipo ha sido, siempre que fuera posible, la de implementar por completo una funcionalidad y subir los cambios al repositorio de *GitHub* correspondiente. Acordar esta forma de trabajar desde el primer momento fue crucial para reducir el número de *commits*, aumentar la productividad (evitando dejar funcionalidades a medio hacer siempre que se pudiera) y evitar conflictos en *Git*. Además, mediante el uso de la hoja de cálculo compartida mencionada en el punto 3.3.3, cada vez que se avanzaba o terminaba una funcionalidad se tomaba constancia de las horas que la implementación de esta había llevado. El uso de esta hoja de cálculo, junto al diagrama de *Gannt* compartido (el cual se actualizaba constantemente) permitía conocer el trabajo realizado, el pendiente y en qué se encontraba trabajando cada persona en cada momento.

Debido a la magnitud del proyecto, han tenido lugar tres divergencias principales tanto en el trabajo como en el calendario del proyecto. A continuación se exponen ambos cambios con respecto a la planificación inicial.

5.2.1 Migración de plataforma de despliegue

Durante la mayor parte del desarrollo de este proyecto se ha empleado una instancia del servicio *Elastic Compute Cloud* de *Amazon Web Services* para desplegar el *backend* en una plataforma de despliegue *serverless*.
Sin embargo, esta decisión que en un principio se tomó resultó limitante conforme avanzaba el proyecto.
Además de los objetivos más técnicos como conseguir implementar verdadera asincronía en el ajedrez, con este proyecto se tenía el deseo de familiarizarnos con las metodologías y técnicas más modernas en lo que a desarrollo de *software* se refiere. Y esto incluye la **entrega e integración continuos**.
Automatizar las pruebas y el despliegue del proyecto con *AWS* resultaba imposible (al menos, para los integrantes del equipo). Se intentó emplear *Github Actions* y usar las herramientas que la plataforma pone a la disposición del usuario pero resultó demasiado difícil como para seguir invirtiendo tiempo y esfuerzo.

Por tanto, se optó por estudiar alternativas para el despliegue en remoto que permitieran, sin excesiva complicación, poner en marcha un *pipeline* de integración y entrega continuos. Después de informarnos sobre los principales proveedores de infraestructura en la nube que facilitasen herramientas de *CI/CD*, se estudió a fondo la viabilidad de usar ***Azure***, de *Microsoft*, o ***Google Cloud Platform*** o ***Firebase***, ambos de Google.

Tras realizar varias pruebas y estudiar los planes que puede tener a su disposición un grupo de estudiantes sin intención ni capacidad de gastar dinero, se ha decidido emplear ***Google Cloud Platform***, que además de la escalabilidad, fiabilidad y seguridad que necesitamos para aportar calidad a la aplicación también ofrece herramientas de *CI/CD* integradas. Estas han permitido crear el *pipeline* que finalmente se ha utilizado.
Además, la utilización de estas herramientas es gratuita durante el tiempo suficiente para terminar el proyecto, finalizando su uso en julio de este año.

5.2.2 Integración fallida de la IA

Una vez se habían implementado todos los requisitos, se decidió llevar a cabo la integración de un motor de ajedrez (*Stockfish 16.1*) que devolviera los movimientos elegidos en base a su nivel. Para ello, se completó con éxito las tareas de traducción del tablero que se utilizaba en la comunicación entre el *backend* y los *frontends* a la forma en la que dicho motor de ajedrez lo admitía. De esta forma, también se logró interactuar correctamente de forma local con dicho motor de ajedrez, configurando su dificultad y recibiendo los movimientos para traducirlos posteriormente.

Sin embargo, el hecho de instalar el ejecutable del motor de ajedrez en la máquina utilizada para el despliegue en *GCP* llevó a errores inesperados. Debido a la falta de tiempo para la implementación de este requisito (pues se solapaba con la fase de realización de pruebas), se optó por no finalizar este requisito a pesar de tenerlo implementado a medio camino.

5.2.3 Gestión de las jugadas ilegales

En un primer momento se pretendía gestionar los movimientos ilegales de la misma manera que el resto de jugadas. Sin embargo, debido a la dificultad de comprobar continuamente que se tratase de una jugada válida se añadió un nuevo parámetro a la hora de comunicar el tablero que fuese 'false' cuando se tratase de un movimiento no válido. Esto permite gestionar correctamente que una pieza no pueda moverse en caso de que si se mueve deje en jaque a su propio rey, es decir, no permite que se mueva una pieza que se encuentra 'clavada'.

Por tanto, no ha habido cambios significativos en las decisiones de diseño tomadas al inicio del proyecto. En la fase inicial, se llevó a cabo una investigación exhaustiva y un análisis detallado de diversas opciones de diseño y tecnologías. A partir de este análisis, se seleccionaron aquellas que mejor se adaptan a nuestro contexto y a los requisitos del proyecto.

No obstante, existe un problema de despliegue debido a que la memoria de los *sockets* no se libera correctamente. Esto provoca que se alcance la capacidad máxima de memoria de la máquina, lo que a su vez causa la caída del sistema en determinadas ocasiones, incumpliendo el requisito no funcional RNF-3.

Finalmente, el periodo de pruebas de integración del *software* se llevó a cabo en el tiempo estipulado para ello. Contar con este tiempo permitió llegar de manera adecuada a probar cada requisito, solventando los problemas anteriormente comentados. Por tanto, a pesar de que no se llevaron a cabo pruebas unitarias en ninguno de los subgrupos, las pruebas de integración fueron exitosas, de forma que el proyecto estuviera preparado para el momento de la presentación. Estas pruebas consistieron, principalmente, en ejecutar un gran número de historias de navegación para comprobar el correcto funcionamiento de cada uno de los requisitos.

5.3. Cierre del proyecto

En cuanto a las estimaciones de costes y esfuerzo en el proyecto, se podría decir que se ha pecado de cierta inexperiencia. En algunos casos se estimó más de lo necesario para ciertas tareas y se fue ingenuo para apartados en los que se ha debido invertir una mayor cantidad de esfuerzo y precisión para no cometer errores.

Por ejemplo, en un primer momento el modo asíncrono supuso un gran reto previo a su desarrollo. Una vez abordado, el equipo se dió cuenta de que la complejidad residía únicamente en el guardado del tablero en la base de datos.

Sin embargo, la conexión con la base de datos la cual se predijo trivial en un primer momento ha dado lugar a problemas. Al confiar en el servidor cloud *"Clever Clod"* se ha tenido que tener cuidado con las conexiones abiertas y usar un correcto uso del *pool* de *PostgreSQL*. Ha provocado errores imprevistos en las pruebas y despliegue final.

En la finalización del proyecto todos los miembros del equipo han concluido con amplios conocimientos sobre las tecnologías utilizadas. Al haber completado un juego de ajedrez se ha terminado con un conjunto de habilidades de grado superior a la creación de formularios, en el caso de *frontends*, y el diseño de rutas, en el caso del *backend*.
Entre las competencias adquiridas por subgrupos se pueden destacar las siguientes:

- *Backend*: conocimiento del lenguaje *JavaScript* mediante la creación de algoritmos avanzados de ajedrez, diseño de rutas para la creación de una *API Rest*, conexión con un base de datos remota y uso de la librería *Socket IO* para proporcionar un juego *online*.
- *Frontend-web*: creación de formularios, gestión de estados usando *SessionStorage*, uso de datos y variables especiales de *React*, conexión a una *API* y manejo de errores en un entorno distribuido, cambios en la apariencia dependiendo de variables y manejo del contexto de la aplicación con el uso de *WebSockets*.
- *Frontend-móvil*: manejo de *widgets*, diseño de interfaces móviles, navegación y uso del contexto, llamadas a una *API* y manejo de *Sockets* con parámetros.

Cabe destacar que todos los miembros del equipo han adquirido un amplio dominio sobre la herramienta para control de versiones *Git* y *GitHub*. Añadido a las tecnologías, se ha aprendido (o mejorado) a trabajar en equipo.

Horas invertidas							
José Miguel	Kamal	Óscar	Alejandro	Jorge	Miguel	Pablo	Nicolás
86.5	136.5	120	122	96	93	104.5	100.5

Tabla 4: Control de esfuerzos de los integrantes del equipo

Tarea	Horas dedicadas por integrante							
	José Miguel	Kamal	Óscar	Alejandro	Jorge	Miguel	Pablo	Nicolás
Aprendizaje de tecnologías	10	5	5	5	10	3	1	7.5
Comunicación *backend-frontend*	5	40	15	25	20	10	10	15
Diseño de la interfaz	10	50	0	50	30	0	10	50
Diseño de la lógica del juego	0	0	50	0	0	40	0	0
Chat	0	3	0	10	0	0	5	30
Modo asíncrono	5	2	15	2	15	0	20	0
Gestión de partidas	30	36.5	15	5	6	25	50	0
Gestión de pase de batalla	0	0	5	25	15	10	0	0
Despliegue	21.5	0	5	0	0	0	8.5	0
IA	5	0	10	0	0	7	0	0
Documentación	5	5	15	12	5	5	15	5
Total	**91.5**	**141.5**	**135**	**134**	**101**	**100**	**119.5**	**105.5**

Tabla 5: Desglose por actividades del control de esfuerzos de los integrantes del equipo

Conclusiones

Para concluir este documento se detallarán las conclusiones a las que se hanllegado tras la finalización del proyecto.

Uno de los aspectos más importantes y que al principio menos importancia se le dio fue la comunicación. Si, en este momento, todos los integrantes volvieran a iniciarse en un nuevo proyecto, se le daría prioridad máxima. Al principio todos los miembros daban por hecho aspectos que no eran ciertos, por lo que cuando más se ha comunicado el equipo ha sido en las fechas finales del desarrollo del proyecto. Se podría decir que se ha mejorado en este aspecto y, lo más importante, que se ha aprendido.

Otro de los aspectos a mejorar y que no se volvería a hacer de la misma forma es la metodología de trabajo, ya que no se ha trabajado en paralelo. Esto se debe a que el *frontend* de móvil ha ido bastante por detrás del *frontend-web* durante el transcurso de desarrollo del proyecto debido a una peor organización de tiempo y mayores dificultades a la hora de aprender una nueva tecnología.

Por otra parte, también se han realizado bien otra serie de aspectos, las cuales se considera que son una buena práctica de trabajo:

- Correcta documentación del código, especialmente en la *API,* para que ambos *frontends* sean capaces de entender todo sin "perseguir" a los miembros del *backend*. Además, entre *frontends* también se ha documentado el código para que cualquiera de los 3 integrantes de cada subgrupo sea capaz de entender y modificar el código.

- Realización del desarrollo de forma ordenada, comenzando por crear la parte visual de la aplicación a la que posteriormente se le incluiría una lógica una vez el *backend* estuviera acabado y funcional.

- Cumplimiento de requisitos de forma progresiva, de modo que hasta que un requisito no estuviera totalmente implementado de forma correcta no se procedía a la realización del siguiente.

Aunque no se hayan realizado las tareas de manera perfecta, el equipo se encuentra muy contento con el resultado final del proyecto y con el procedimiento del mismo. Además, se considera que se ha aprendido mucho, sobre todo a trabajar en equipo y a ser autodidactas, dos aspectos muy importantes en el mundo de la informática.

Anexo I. Presupuesto

ESTIMACIÓN DE COSTES DIRECTOS									
ESFUERZOS						COSTES			
Tarea	Descripción	Requisitos relacionados	Estimación horas mínimo	Estimación horas más probable	Estimación horas máximo	Coste/hora	Coste (€) mínimo	Coste (€) más probable	Coste (€) máximo
Autenticación y gestión de usuarios	Permitir a los usuarios registrarse, inicia	RF-1, RF-1.1, RF-5	6	12	20	21,78€	130,70€	261,40€	435,67€
Creación de partidas locales y *online*	Permitir a los usuarios jugar partidas ta	RF-2, RF-3	70	90	120	21,78€	1.524,83€	1.960,50€	2.614,00€
Implementación de partidas asíncro	Permitir a los usuarios jugar por corres	RF-4	16	24	30	21,78€	348,53€	522,80€	653,50€
Implementación de los ritmos para	Permitir a los usuarios jugar con 3 ritm	RF-4	6	15	24	21,78€	130,70€	326,75€	522,80€
Emparejamientos mediante ELO	Emparejar a los usuarios entre sí en fun	RF-6, RF-6.1	8	12	15	21,78€	174,27€	261,40€	326,75€
Creación de arenas	Permitir que los usuarios jueguen en di	RF-6.2, RF-6.2.1	30	40	54	21,78€	653,50€	871,33€	1.176,30€
Implementación del chat online	Permitir a los jugadores de una misma	RF-7	10	16	24	21,78€	217,83€	348,53€	522,80€
Integración de una IA	Permitir a los usuarios jugar contra una	RF-8	10	12	15	21,78€	217,83€	261,40€	326,75€
Gestión del ranking de jugadores	Crear un ranking global de jugadores en	RF-9	5	8	12	21,78€	108,92€	174,27€	261,40€
Creación de "pase de recompensas"	Crear una serie de recompensas que pu	RF-10, RF-10.1	20	30	36	21,78€	435,67€	653,50€	784,20€
Diseño del frontend-web	Diseño de la interfaz gráfica en la web	RNF-1	45	75	105	21,78€	980,25€	1.633,75€	2.287,25€
Diseño del frontend-móvil	Diseño de la interfaz gráfica en el móvil	RNF-1	50	80	110	21,78€	1.089,17€	1.742,67€	2.396,17€
Seguridad y eficiencia	Garantizar la seguridad y la rapidez de r	RNF-2, RNF-3, RNF-4	10	15	20	21,78€	217,83€	326,75€	435,67€
Tiempo de máquinas en *cloud* para	Despliegue en una instancia EC2 de AW	RNF-5	20	24	30	21,78€	435,67€	522,80€	653,50€
Pruebas de aplicación	Pruebas continuas para el correcto desp	RNF-5	25	35	40	21,78€	544,58€	762,42€	871,33€
TOTAL TAREAS/COMPONENTES			**331**	**488**	**655**	**21,78€**	**7.210,28€**	**10.630,27€**	**14.268,08€**

Costes MACROS									
Concepto	Descripción	Dedicación sobre desarrollo	Horas mínimo	Horas Más probable	Horas máximo	Coste/hora	Coste (€) mínimo	Coste (€) más probable	Coste (€) máximo
Gestión proyecto	Gestión de equipos, distribución de tare	15%	49,65	73,2	98,25	21,78€	1.081,54€	1.594,54€	2.140,21€
Aseguramiento de la calidad		5%	16,55	24,4	32,75	21,78€	360,51€	531,51€	713,40€
TOTAL MACROS			**66,2**	**97,6**	**131**	**21,78€**	**1.442,06€**	**2.126,05€**	**2.853,62€**

Otros costes directos									
Concepto	Descripción	Cantidad mínimo	Cantidad más probable	Cantidad máximo	Coste/ unidad	Coste/concepto mínimo	Coste/concepto más probable	Coste/concepto máximo
Máquinas en *cloud* para el despliegu	Alquiler de las máquinas de AWS para el despliegue. Precio por mes	1	3	5	37,2	37,20€	111,60€	186,00€
Viajes para visitas al cliente	Visitas a la universidad para reuniones, prácticas, tutorías y entregas. Cantid	40	80	120	2	80,00€	160,00€	240,00€
	Comidas en días de entregas	8	16	24	7	56,00€	112,00€	168,00€
Uso equipos	Amortización de los equipos. Cálculo hecho a 1.000 €/equipo. Si se quiere, se puede modificar	397,2	585,6	786	0,1388889	55,17€	81,33€	109,17€
TOTAL OTROS COSTES DIRECTOS						**228,37€**	**464,93€**	**703,17€**

Concepto	Costes	Costes/hora	Imputable al proyecto mínimo	Imputable al proyecto más probable	Imputable al proyecto máximo	Detalle
Alquiler instalaciones	6.000,00 €	0,42 €	165,50 €	244,00 €	327,50 €	Presupongo 500 €/mes
Limpieza	5.200,00 €	0,36 €	143,43 €	211,47 €	283,83 €	Presupongo 100 €/semana
Luz y calefacción	1.200,00 €	0,08 €	33,10 €	48,80 €	65,50 €	Presupongo 100 €/mes
Comunicaciones	1.200,00 €	0,08 €	33,10 €	48,80 €	65,50 €	Presupongo 100 €/mes
Proyectos no exitosos	62.736,00 €	4,36 €	1.730,47 €	2.551,26 €	3.424,34 €	Presupongo un 20% de la capacidad de trabajo de la empresa
Servicios de seguridad	1.200,00 €	0,08 €	33,10 €	48,80 €	65,50 €	Presupongo 100 €/mes
Agua	240,00 €	0,02 €	6,62 €	9,76 €	13,10 €	Presupongo 20 €/mes
Formación continua	34.368,00 €	2,39 €	947,98 €	1.397,63 €	1.875,92 €	Presupongo un 10% de la capacidad de trabajo de la empresa
Asesor fiscal y laboral	2.400,00 €	0,17 €	66,20 €	97,60 €	131,00 €	Presupongo 200 €/mes
Seguros	600,00 €	0,04 €	16,55 €	24,40 €	32,75 €	Presupongo 50€/mes
Costes de oficina	360,00 €	0,03 €	9,93 €	14,64 €	19,65 €	Presupongo 30€/mes
Fondo para emergencias	1.000,00 €	0,07 €	27,58 €	40,67 €	54,58 €	Presupongo 1000€
TOTAL INDIRECTOS	**116.504,00 €**		**3.213,57 €**	**4.737,83 €**	**6.359,18 €**	

	Mínimo	Más probable	Máximo
HORAS TOTALES PROYECTO	397	586	786
TOTAL COSTES DIRECTOS	8.880,71€	13.221,25€	17.824,87€
TOTAL COSTES INDIRECTOS	3.213,57€	4.737,83€	6.359,18€
TOTAL COSTES PROYECTO	12.094,28€	17.959,08€	24.184,04€

Anexo II. Resultados de la revisión intermedia

Metodología del Trabajo y documentación	Respuesta (Sí/No)	Si la respuesta es que no, indicad cómo vais a mejorar en este aspecto. Si la respuesta es que sí, añadid información adicional que consideréis relevante.
Se lleva un registro escrito de las principales decisiones tomadas por el equipo.	Sí	En un documento en *Google Drive* se reflejan todas las decisiones importantes.
Los requisitos definitivos del proyecto han sido revisados por el profesor tutor y cuentan con su visto bueno.	Sí	En la primera reunión ya se decidieron los objetivos finales con el tutor, sin ambiguedades.
Hay un diagrama de navegación con los diseños de las interfaces de los frontends.	Sí	Los mapas de navegación se diseñaron al principio del proyecto.
Hay un diagrama de Gantt actualizado a fecha de hoy que refleja las fechas de inicio y fin reales de todas las tareas ya terminadas y la planificación de todas las tareas pendientes.	Sí	*Se encuentra en la página 16 del documento que redacta el plan de gestión*
Las tareas indicadas en el diagrama de Gantt incluyen todas las tareas necesarias para completar el proyecto (incluyendo tests, despliegue, documentación y gestión). Es especialmente crítico asegurarse de que estas tareas cubren el 100% de los requisitos.	Sí	En cada departamento se tiene en cuenta la realizacion de pruebas para la correcta verificación de los requisitos.
Todas las tareas de análisis, diseño, implementación, tests, despliegue y gestión del proyecto tienen un responsable conocido por todos.	Sí	Dentro de cada equipo se ha intentado dar un rol de responsable a uno de los integrantes. En ambos equipos de *frontend* existe una persona responsable de la constante comunicación con el *backend* para el correcto y eficiente uso de la *API* diseñada.
Se está siguiendo una política de nombrado y gestión de documentos acordada (diseños, código, documentos de gestión...).	Sí	Las políticas de nombrado y gestión de documentos quedan reflejadas en el documento que redacta el plan d gestión.
Se está siguiendo alguna guía de estilo de codificación para generar código con un formato homogéneo.	Sí	Las guías de estilo de codificación quedan reflejadas en el documento que redacta el plan de gestión.
Se está siguiendo alguna guía de diseño de GUI para generar interfaces gráficas homogéneas y usables.	Sí	Las guías de diseño de la GUI quedan reflejadas en el documento que redacta el plan de gestión.
Hay un diagrama de arquitectura actualizado con los componentes que forman los frontend y backend del sistema y sus interacciones.	No	A día de hoy, el diagrama de arquitectura deberá ser actualizado correctamente para la entrega del proyecto. Por el momento no ha supuesto un problema debido a la simplicidad de la arquitectura de los componentes.
Hay un documento definiendo la API que expone el backend.	Sí	*Se encuentra definida en un docuemento en Google Drive con rutas, parametros y cuerpos de solicitud posibles, cuyo link se ecuentra en el documento que redacta el plan de gestión.*
Hay documentación arquitectural adicional, describiendo otras vistas del sistema (módulos, incluyendo el modelo de datos y despliegue) y aspectos de la dinámica del mismo.	No	Se pretende redactar una documentación arquitectural actualizada para realizar la entrega final.
Se está usando del sistema de gestión de incidencias de GitHub para asignar tareas de desarrollo.	Sí	Se utiliza GitHub Issues, tal y como se indica en el documento que redacta el plan de gestión.
Se está usando un proceso de control de versiones (workflow) que dicta cuándo, y en qué rama se sube código a cada repositorio de GitHub.	No	Por el momento, no existen workflows debido a que todos los cambios se suben directamente a la rama *main* de cada repositorio.
Se están haciendo pruebas suficientes: unitarias, de integración, de sistema etc. Idealmente al menos una parte de estas pruebas, o todas, son automáticas.	No	Se están realizando pruebas de cada apartado en el que se comprueba el funcionamiento. En el caso del *frontend-web* existen pruebas automáticas, pero no se han acabado de concretar las pruebas definitivas. Lo ideal sería implementar pruebas automáticas con GitHub Actions.
Se ha desplegado al menos una vez el sistema integrado, en su estado actual, en un entorno similar al que se prevea como entorno de despliegue final (entorno de producción). Idealmente este proceso se ha hecho ya varias veces, e idealmente este proceso se ha automatizado al menos parcialmente.	No	A pesar de que sí se han realizado pruebas para comprobar el correcto funcionamiento del código en la instancia de *EC2* utilizada, todavía no se ha lanzado en un entorno completamente real en general. Sí que se han desplegado por separado los distintos componentes. El problema radica en que los servicios despliegue suponen un coste económico imprevisto, aunque pequeño. En cuanto este más avanzada la producción se ejecutará en un entorno real.

Estado del proyecto y grado de ejecución	Respuesta (Valor	Mitigación de posibles problemas
Porcentaje de requisitos funcionales completados.	60%	*El proyecto se encuentra en un estado avanzado. Sin embargo, quedan conceptos de integración entre backend y frontend aún por implementar. Creemos que vamos en buen camino para llegar a tiempo a prácticamente todos los requisitos.*
Horas de trabajo realizadas respecto de las previstas del total del proyecto.		*Si este número es muy bajo, indicad qué vais a hacer para mejorarlo.*
Estimación del porcentaje del proyecto completado, en desarrollo, y pendiente, en función de las tareas completadas, en desarrollo y pendientes.	60% completado, 20% en desarrollo, 20% pendiente	*A pesar de que la lógica del juego está prácticamente finalizada, todavía se estña trabajando en la gestión de usuarios, partidas y recompensas, así como en distintos aspectos del despliegue.*

Actividad de cada integrante del equipo desarrollo. Repetid esta sección las veces que	Respuesta (Valor
Óscar Brizuela García	
Horas dedicadas al proyecto hasta este momento.	60
Horas previstas de dedicación al proyecto en total.	120
Número de commits realizados en Github.	72
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	2160
Kamal Bouizy Ghazzal	
Horas dedicadas al proyecto hasta este momento.	60,5
Horas previstas de dedicación al proyecto en total.	120
Número de commits realizados en Github.	24
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	2534
Alejandro Calvera Tonin	
Horas dedicadas al proyecto hasta este momento.	88
Horas previstas de dedicación al proyecto en total.	110
Número de commits realizados en Github.	68
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	4702

Nicolás Pelegrín Sánchez	
Horas dedicadas al proyecto hasta este momento.	41
Horas previstas de dedicación al proyecto en total.	120
Número de commits realizados en Github.	16
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	1412
José Miguel De La Ascensión	
Horas dedicadas al proyecto hasta este momento.	65,5
Horas previstas de dedicación al proyecto en total.	110
Número de commits realizados en Github.	17
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	721
Jorge Jaime Modrego	
Horas dedicadas al proyecto hasta este momento.	40
Horas previstas de dedicación al proyecto en total.	110
Número de commits realizados en Github.	13
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	1930
Miguel Lacámara Pasamar	
Horas dedicadas al proyecto hasta este momento.	56
Horas previstas de dedicación al proyecto en total.	110
Número de commits realizados en Github.	33

Número de commits realizados en Github.	33
Líneas de código vivas en el Github del proyecto (solo aquellas que existen en la última versión del proyecto). No cuentan las bibliotecas, que no deberían estar almacenadas en vuestros repositorios. Para ello podéis usar git-fame tal y como se indica en el guión de la práctica.	1753

Anexo III. Mapa de navegación *frontend* web

Figura 10: Mapa de navegación del frontend web

Anexo IV. Mapa de navegación *frontend* móvil

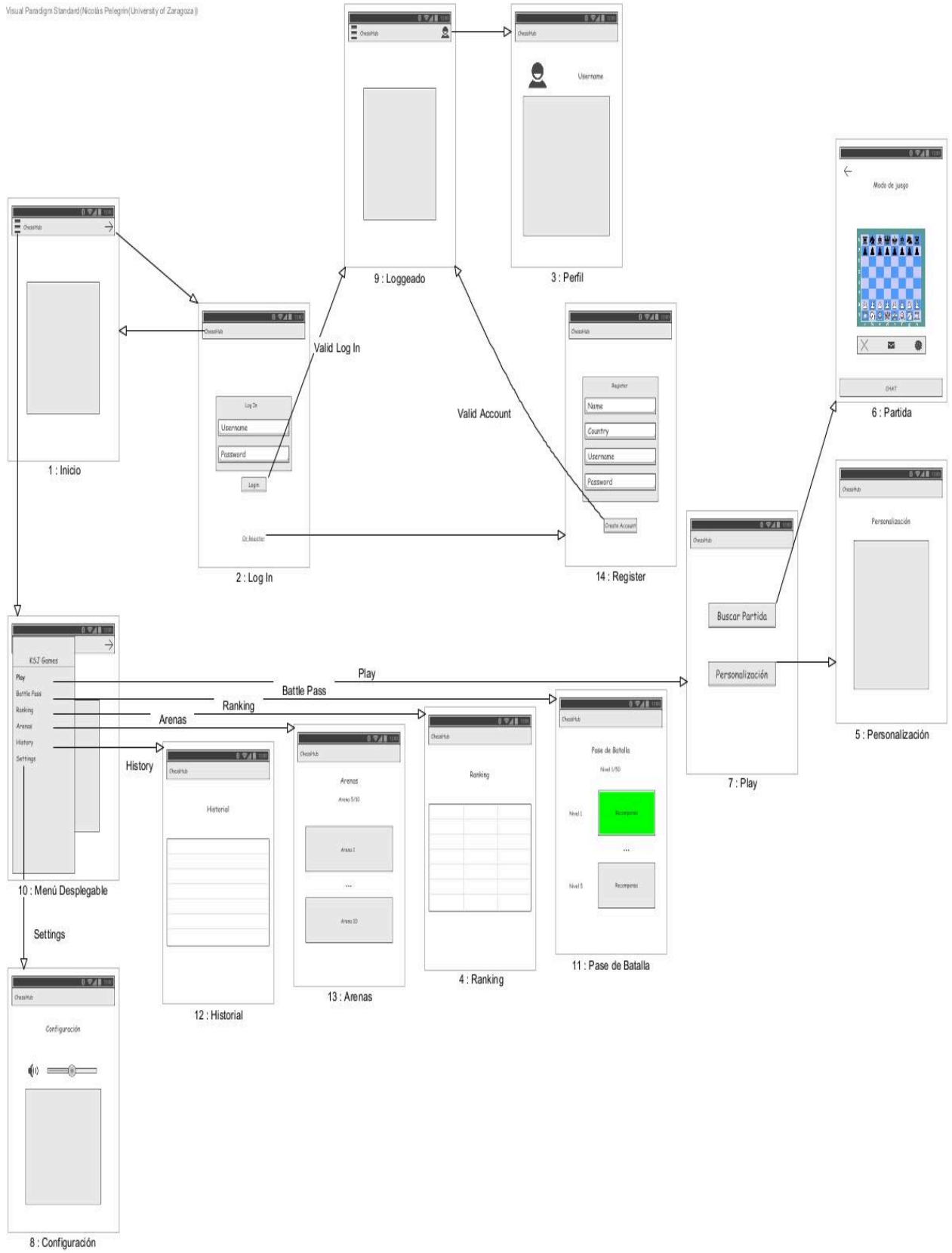


Figura 11: Mapa de navegación de la aplicación móvil

Anexo V. Recursos de formación utilizados por los integrantes

- *Backend*:
  - Curso rápido Express.js
  - Enrutamiento Express.js
  - Middleware y Express.js:
    - https://expressjs.com/en/guide/writing-middleware.html
    - https://expressjs.com/en/guide/using-middleware.html
  - Documentación JavaScript
  - Curso completo de React

- *Frontend-móvil*:
  - Dart Basics: https://dart.dev/language
  - Documentación Flutter: https://docs.flutter.dev
  - Tutoriales oficiales Flutter: https://www.youtube.com/@flutterdev/videos

- *Frontend-web*:
  - Curso React Midulive
  - Razones para aprender React
  - Master React
  - Documentar una arquitectura con React