# Greaslider

**David Raya Conesa**
Máster en desarrollo de aplicaciones para dispositivos móviles

**Eduard Martin Lineros**
**Carles Garrigues Olivella**

13/03/2019


Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

**FICHA DEL TRABAJO FINAL**

| | |
|---|---|
| **Título del trabajo:** | *Greaslider* |
| **Nombre del autor:** | *David Raya Conesa* |
| **Nombre del consultor/a:** | **Eduard Martin Lineros** |
| **Nombre del PRA:** | |
| **Fecha de entrega (mm/aaaa):** | 06/2019 |
| **Titulación::** | *Máster universitario de Desarrollo de aplicaciones para dispositivos móviles* |
| **Área del Trabajo Final:** | *Aplicaciónes Android* |
| **Idioma del trabajo:** | *Castellano* |
| **Palabras clave** | *Android, Bluetooth, Juego* |


**Resumen del Trabajo (máximo 250 palabras):**

El mercado de las aplicaciones móviles está en auge, en concreto las relacionadas con los videojuegos. Son muchos los usuarios potenciales que juegan en sus ratos de ocio para entretenerse y competir. Pero se conoce que el mercado es muy competitivo y es difícil lanzar una aplicación y que triunfe.

Por lo que el objetivo de este trabajo es el de intentar sacar una aplicación original. Al no disponer de una importante inversión económica ni un equipo de desarrollo, el juego se centrará en su originalidad para diferenciarse del resto. Además, es necesario que esté disponible en la mayoría de dispositivos. También que sea fácil de jugar y compartir con amigos para darse a conocer.

Con este fin, en el presente documento se explica cómo se ha desarrollado el juego Greaslider para Android. Un juego de mesa original adaptado para estos dispositivos digitales. Para facilitar la forma de jugar se han implementado distintos modos de juego para cubrir diferentes escenarios de uso que pudiese necesitar el usuario. Permite jugar de forma individual, con una amigo en el mismo dispositivo o en distintos dispositivos mediante la tecnología Bluetooth. Además de tener unos diseños que permite que el juego se adapte a la mayoría de dispositivos.

El juego se probó con distintos usuarios y cumple lo esperado. Pero para poder lanzarlo al mercado es necesario incorporarle mejoras que los usuarios esperan que tengan los juegos actualmente como permitir jugar contra otros a través de Internet.

**Abstract (in English, 250 words or less):**

The market for mobile applications is booming, specifically the video games. There are many potential users who play in their leisure time to entertain and compete. But it is known that the market is very competitive and it is difficult to launch an application and succeed.

So the goal of this work is to try to get an original application. By not having an important economic investment or a development team, the game will focus on its originality to differentiate itself from the rest. In addition, it needs to be available on most devices. Also make it easy to play and share with friends to make themselves known. To this end, this document explains how the Greaslider game for Android has been developed. An original board game adapted for these digital devices. To facilitate the way of playing, different game modes have been implemented to cover different usage scenarios that the user may need. Allows you to play individually, with a friend on the same device or on different devices using Bluetooth technology. In addition to having some designs that allows the game to adapt to most devices. The game was tested with different users and meets expectations. But to be able to launch it to the market, it is necessary to incorporate improvements that users expect games to have nowadays, such as allowing players to play against others through the Internet.

# Índice

## Lista de figuras

# 1. Introducción

## 1.1 Contexto y justificación del Trabajo

Con el motivo de profundizar más en los conocimientos adquiridos con las asignaturas del máster se ha pensado en la creación de un videojuego para dispositivos móviles.

Es un juego con reglas muy básicas en el que su complejidad radica en la cantidad de jugadas posibles, como el ajedrez y las damas, que te permiten obtener la victoria.

Para hacerlo se pensó en crear una aplicación móvil, si fuese posible en 3D. Que implementase toda la lógica del juego y permitiese competir contra otro de forma inalámbrica. La idea de este juego es que el usuario pueda enfrentarse tanto contra una IA como contra otros usuarios. La conexión que se utilizará entre ellos en un principio será Bluetooth.

Con el objetivo de que este juego esté disponible para la mayoría de usuarios se ha implementado para dispositivos con el API mínimo 23. Y para que se pueda ver bien en los diferentes tipos de pantalla, uno de los principales problemas de Android, al haber tantos tipos de dispositivos, se ha utilizado ConstraintLayout.

## 1.2 Objetivos del Trabajo

Implementar un juego de tablero original para dispositivos móviles.

Diseñar la interfaz, así como las reglas del juego para que sean intuitivas.

Adaptar el juego al mayor número de dispositivos, pero sin perder calidad, para que sea accesible por la mayoría de usuarios

Desarrollar una IA simple capaz de enfrentarse al usuario.

Integrar la conexión por Bluetooth para permitir jugar contra otros.

Aplicar los mecanismos de seguridad necesarios para proteger al usuario de una conexión por Bluetooth maliciosa.

## 1.3 Enfoque y método seguido

### Estrategia 1

Una de las posibles opciones consistía en empezar el proyecto basándolo en un juego de ajedrez o damas realizado en Unity. Ya que el tablero similar, adaptando este tipo de proyectos al nuestro, se puede abordar con una alta probabilidad de éxito.

Puesto que no se tienen conocimientos de modelado 3D, basándonos en otros proyectos, se puede aprender cómo han sido creadas las piezas y el tablero y extraer ese conocimiento para personalizar las nuestras.
De esta forma también se consigue mitigar el riesgo de no conocer cómo crear diseños en 3D.

Sin embargo, para diferenciar esta aplicación del resto, la dinámica del juego es totalmente nueva. Por lo que tanto los movimientos y la IA encargada de jugar contra el usuario serán puntos clave del éxito de la aplicación

En cuanto a la posibilidad de jugar contra otros usuarios se establecerá una conexión entre los dispositivos con Bluetooth. Para esto serán necesarios plugins de Unity.

### Estrategia 2

Esta opción es igual que la primera pero se ha planteado aplicar la opción multijugador sobre un único dispositivo en lugar de utilizar Bluetooth.
De esta forma, en cada uno de los turnos se cambia la perspectiva desde la que se ve el tablero para mostrar la del jugador al que le toca jugar, permitiéndole mover sus piezas.

### Estrategia 3

Se pensó también, como posible solución, crear la aplicación en nativo, concretamente para Android. De esta forma se podría hacer uso de la conexión Bluetooth en nativo sin necesidad de usar plugins de terceros.

Sin embargo con esta opción no resulta viable aplicar gráficos en 3D, por lo que el juego se verá en 2D. La parte positiva es que es un entorno más controlable al solo centrarse en los dispositivos Android. No obstante, no podremos cubrir el resto de mercado, como por ejemplo IOS.

### Estrategia escogida

Debido a que para desarrollar la conexión por Bluetooth en Unity se necesitan plugins de pago y, según las criticas y valoraciones de Internet, algunos no eran muy fiables, se decidió descartar la primera de las opciones.[1]

Una vez descartada por la dificultad de implementar la conexión Bluetooth en Unity se estuvieron valorando las otras dos opciones.

La que se escogió al final fue la tercera: la implementación del juego para Android en Nativo. Se descartó la segunda porque se quiere crear una conexión inalámbrica real y aprender cómo funciona. Al seleccionar esta opción también se pretende que el juego se adapte a las diferentes pantallas de los dispositivos móviles de Android utilizando constraints. También, por el hecho de hacerla en nativo el proyecto se podrá desarrollar de una forma más controlada y robusta.

## 1.4 Planificación del Trabajo

Para abordar este proyecto se necesita de un ordenador, Android Studio y acceso a la API oficial de Android. Además de conexión a Internet para poder consultar información relevante.

El tiempo de dedicación será de unas 17 horas semanales aproximadamente. Repartidas en 3 horas diarias entre semana y una hora diaria los fines de semana. De esta forma, si fuesen necesarias más horas, fácilmente se podrían cubrir los fines de semana.

3h diarias de lunes a viernes = 15h
1h diaria fines de semana = 2h

**Fechas Importantes**

Plan de trabajo 13/03
Diseño 03/04
Implementación 15/05
Entrega final 05/06
Tribunal 21/06

**Tareas**

Redacción de la memoria del trabajo final. Esta tarea se realiza en paralelo con las PECs 2 y 3, documentando todo lo que se ha hecho en éstas.

PEC2 Diseño

Estudio de mercado. Para comparar esta aplicación con juegos parecidos.

Investigar usuarios potenciales.

Definir requisitos funcionales y no funcionales necesarios para cubrir las necesidades de los usuarios y obtener un producto de calidad.

Elaborar casos de uso y así entender mejor cómo se comportará la aplicación frente a la interacción de los usuarios.

Diseño del menú de juego con las opciones de juego individual y multijugador.

Prototipo de alto nivel con NinjaMock.

PEC3 Implementación

Implementar la conexión Bluetooth entre dos dispositivos y el envió de datos.

Diseño 2D del tablero y las piezas.

Animaciones de los movimientos de las piezas.

Gestión de la interacción del usuario. La implementación de todas las mecánicas de juego, el control de las piezas, los movimientos, determinar cuándo ha finalizado la partida y el vencedor...

Testing del juego

Diseño y análisis del algoritmo de la IA contra la que jugará el usuario.

Desarrollo de la IA.

Testeo de la IA.

PEC 4 Entrega final

Resolución de incidencias. Margen de tiempo de holgura reservado por si surgen incidencias y así poder completar todas las tareas del proyecto.

Repaso y corrección de Memoria.

Manual de usuario. Para explicar cómo se usa la aplicación e instrucciones de como compilar.

Presentación.

**Tabla de Hitos**

| Nombre de la tarea | Duración | Inicio | Finalizar |
|---|---|---|---|
| PEC2 - Diseño | 15,667d | 13/03/19 | 03/04/19 |
| Estudio de mercado | 6h | 13/03/19 | 14/03/19 |
| Investigar usuarios potenciales | 2h | 15/03/19 | 15/03/19 |
| Definir requisitos | 8h | 16/03/19 | 19/03/19 |
| Elaborar casos de uso | 8h | 20/03/19 | 22/03/19 |
| Diseño del menú de juego | 8h | 23/03/19 | 26/03/19 |
| Prototipo de alto nivel | 20h | 26/03/19 | 03/04/19 |
| PEC3 Implementación | 30,667d | 02/04/19 | 14/05/19 |
| Implementar la conexión Bluetooth | 8h | 02/04/19 | 04/04/19 |
| Diseño 2D del tablero y las piezas | 4h | 04/04/19 | 05/04/19 |
| Animaciones de los movimientos | 8h | 05/04/19 | 09/04/19 |
| Gestión de la interacción del usuario | 28h | 09/04/19 | 22/04/19 |
| Testing del juego | 12h | 19/04/19 | 24/04/19 |
| Diseño y análisis del algoritmo de la IA | 16h | 25/04/19 | 02/05/19 |
| Desarrollo de la IA | 20h | 02/05/19 | 10/05/19 |
| Testeo de la IA | 8h | 10/05/19 | 14/05/19 |
| PEC 4 Entrega final | 15,667d | 15/05/19 | 05/06/19 |
| Resolución de incidencias | 24h | 15/05/19 | 24/05/19 |
| Repaso y corrección de Memoria | 10h | 23/05/19 | 28/05/19 |
| Manual de usuario | 8h | 25/05/19 | 28/05/19 |
| Presentación | 20h | 28/05/19 | 05/06/19 |
| Redacción de la Memoria | 130h | 13/03/19 | 13/05/19 |

**Diagrama de Gantt**

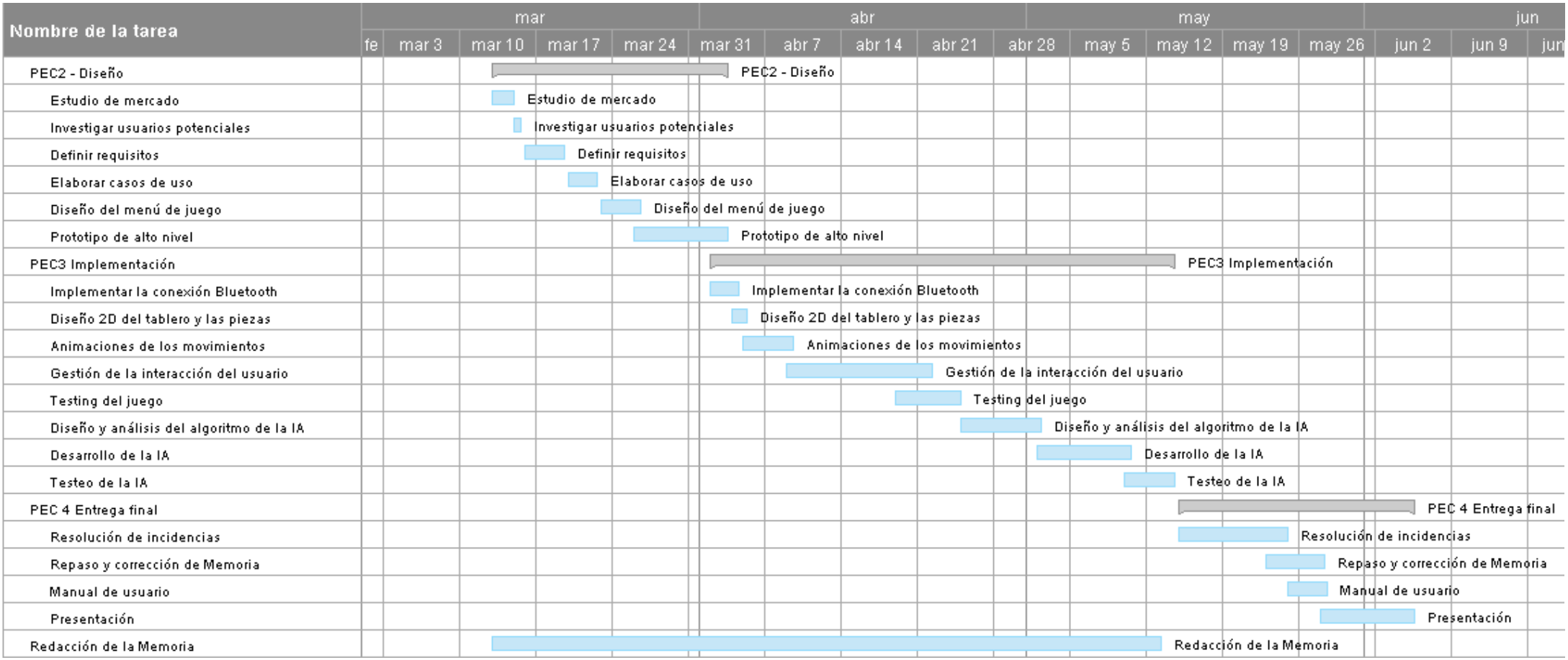


**Ilustración 1: Diagrama de Gantt**

## 1.5 Breve sumario de productos obtenidos

Los productos que se obtendrán en las entregas serán:

PEC2 Diseño:
Documentación de la viabilidad del proyecto, requisitos a cubrir y los diseños de las pantallas de la aplicación.

PEC3 Implementación:
Primera versión del juego funcional y la primera versión de la memoria.

PEC 4 Entrega final
Juego completo, la presentación, el manual de usuario y la memoria completa y revisada.

## 1.6 Breve descripción de los otros capítulos de la memoria

En el capítulo 2 se muestra todo el análisis previo a la implementación.
Empieza con un análisis de la competencia, aplicaciones similares a la que se quiere desarrollar. Así, conociendo los puntos fuertes y débiles del resto, se puede hacer un plan estratégico de como implementar la nuestra, evitando los fallos que cometen e intentando emular sus puntos fuertes.
Una vez conocida la competencia se estudian los usuarios potenciales y en qué escenario usarían nuestra aplicación.
Después se han definido los requisitos y los respectivos casos de uso.
Finalmente en este capítulo 2 se muestran los prototipos y los diagramas que representan la arquitectura del sistema.

En cuanto al capítulo 3 , explica el frontend de la aplicación que finalmente se ha implementado usando como referencia los diseños del prototipo del anterior capítulo.

Dentro del capítulo 4 se encuentra todo lo referente al backend, los distintos desarrollos de todos los modos de juego implementados.

Finalmente, en el 5 se definen algunas de las pruebas realizadas para probar la aplicación.

# 2. Diseño

## 2.1 Competencia

Se han estudiado las siguientes aplicaciones de la competencia. Puesto que es un juego de tablero parecido al ajedrez, se ha optado por analizar ese tipo juegos.
Lo que se ha extraído de cada una de ellas se ha obtenido a partir de la descripción de la aplicación de Google Play, los comentarios de sus usuarios y la instalación y prueba de éstas en un dispositivo.

**Chess.com**[2]
La aplicación móvil de una de las páginas web para jugar ajedrez más famosas chess.com.
El punto fuerte de esta aplicación:

- Permite jugar contra otros usuarios.
- Tiene lecciones interactivas y vídeos
- Se puede jugar individualmente contra la máquina.
- Puzles tácticos para practicar.
- Artículos diarios de ajedrez
- Foros

**Lichess**[3]

Es una de las mejores aplicaciones de ajedrez, la cual utilizan más de 150.000 usuarios diariamente.
El punto fuerte de esta aplicación:

- Ofrece una gran variedad de modos de juego, torneos...
- Permite desafiar a otros jugadores
- Se puede jugar de forma individual contra la máquina
- Incluye puzles para practicar.
- También hay una sección de aprendizaje
- Sin publicidad

**Chess Via Bluetooth**[4]

Aplicación muy parecida a la que se quiere realizar. Permite que dos usuarios jueguen al ajedrez via Bluetooth, cada uno desde su propio dispositivo.
La distribución de los elementos en la pantalla es muy parecida a la idea que tenía de cómo hacerlo en la actual aplicación: el tablero en el centro aprovechando todo el ancho del dispositivo y colocar arriba y abajo los nombres y el recuento de piezas.
El punto fuerte de esta aplicación:

- Está centrada en el multijugador por Bluetooth.

Debilidades:

- No se puede jugar de forma individual al no tener una IA contra la que competir.
- No notifica el cambio de turno
- El usuario no recibe el feedback de si ha ganado la partida
- Anuncios


**Ilustración 2: Chess Via Bluetooth**

## 2.2 Usuarios potenciales

Debido a que el juego a desarrollar es de estrategia y es parecido al ajedrez y a las damas, se supone que compartirá el mismo nicho de mercado.
Puesto que es un juego que no distingue de edades, género y estatus social, lo puede practicar todo el mundo.

Sin embargo, leyendo un artículo en el que hicieron un estudio sobre una muestra de 71.000 jugadores de ajedrez en chess.com se puede apreciar la edad de los usuarios potenciales.
En este artículo se muestra tanto la relación de la edad del jugador con respecto a sus habilidades de juego, como la distribución de los jugadores por edad.[5]

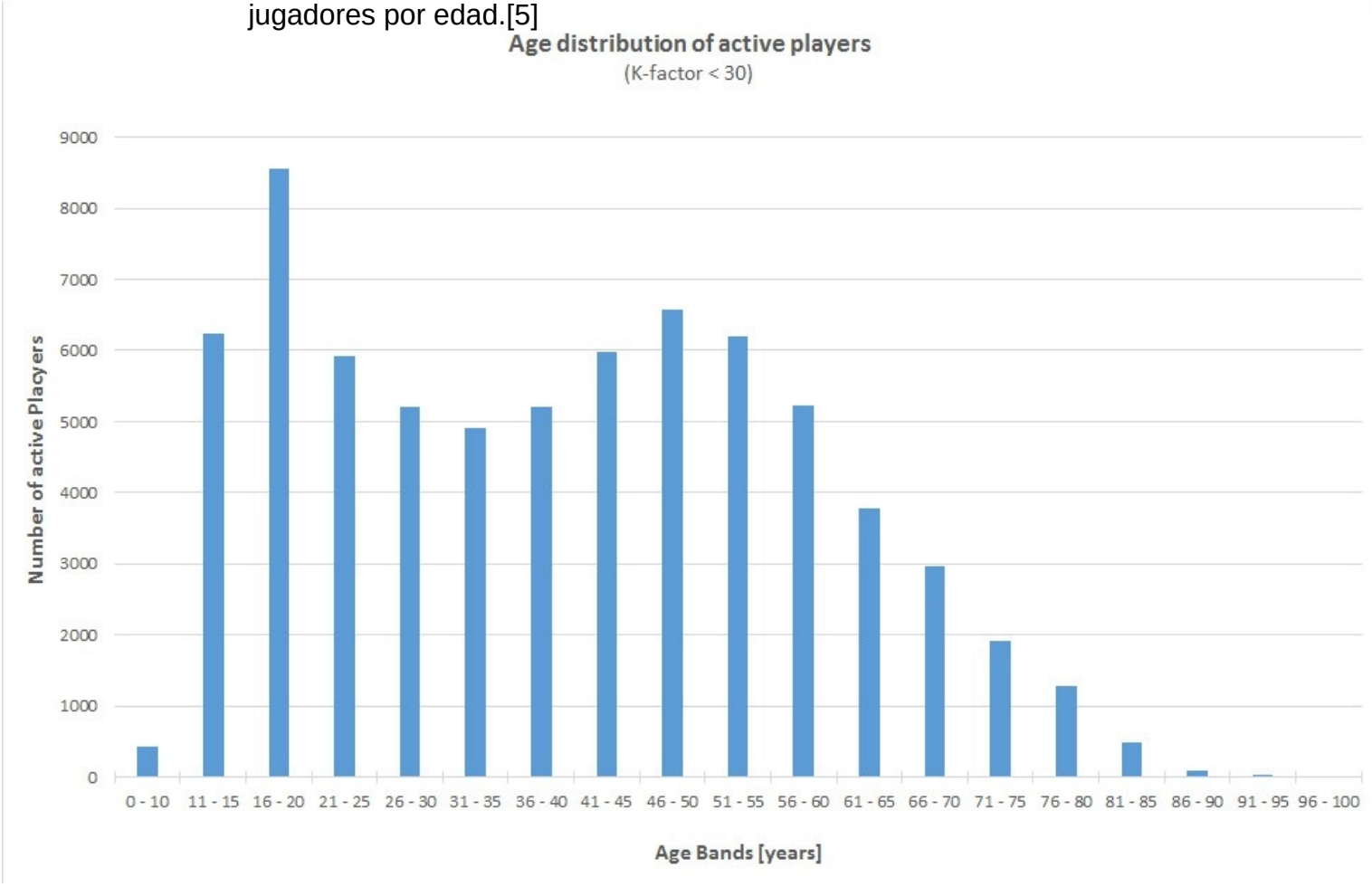


**Ilustración 3: Distribución de la habilidad de los jugadores por su edad**

Se puede ver que está muy distribuido, ya que es un juego al que cualquiera puede jugar. Es cierto que de adolescente entusiasma más este juego, pero tampoco pierde público adulto hasta edades muy avanzadas.

Las siguientes gráficas obtenidas de la página "business of apps" muestran las horas invertidas en utilizar aplicaciones móviles por edad. [6]

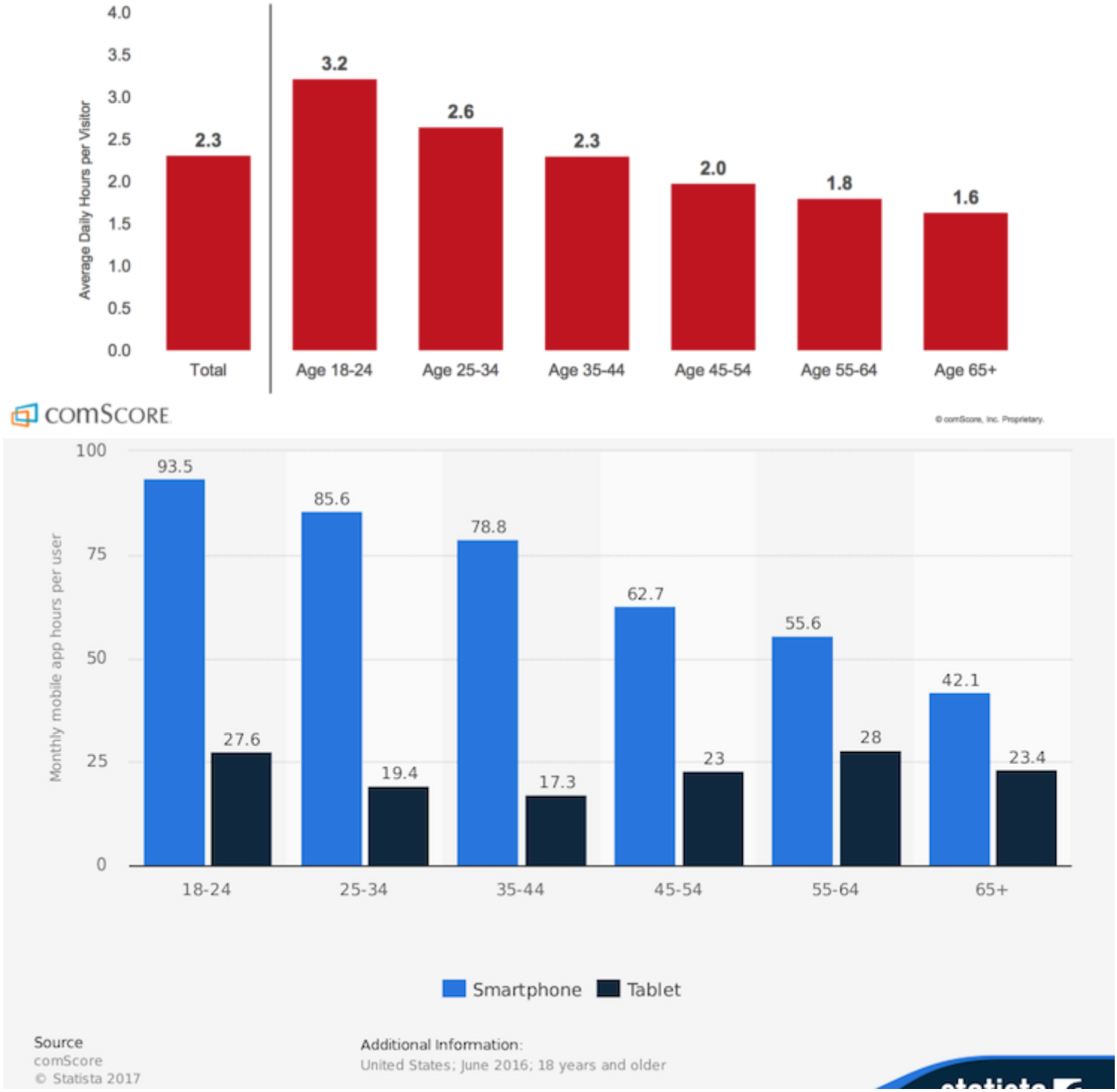


**Ilustración 4: Horas invertidas en aplicaciones móviles**

Teniendo en cuenta ambos gráficos los usuarios potenciales serán jóvenes de entre 11 y 24, ya que son los que más juegan al ajedrez y más tiempo emplean en usar aplicaciones. Aunque se espera que atraiga también al público adulto que desee probar nuevos juegos de estrategia.

## 2.3 Escenario de Uso

PERSONA

**Ilustración 5: Persona**

Nombre: Erick
Edad: 25

Erick vive actualmente en Barcelona en un piso de alquiler con un par de amigos. Ha vivido toda su vida en Málaga pero se ha mudado por la creciente demanda de informáticos en la capital de Cataluña. Trabaja en una multinacional como desarrollador de servicio web haciendo jornada completa. Además también se forma en su tiempo libre, siempre le ha gustado aprender seguridad informática y crear aplicaciones móviles, aunque no ha tenido ninguna experiencia laboral en esos ámbitos. Hace poco se apuntó a una cursa y desde entonces entrena tres días a la semana, corriendo entre siete y ocho kilómetros. Recientemente se ha aficionado a ir al teatro y a la biblioteca. Siempre que puede aprovecha para ir con los amigos al parque y jugar al ajedrez. Le encantan los juegos de lógica y de estrategia y prueba todos los que puede. Sin embargo últimamente se le están haciendo repetitivos. Un problema que ve en esos juegos es que no se pueden disfrutar sin Internet. Además sus amigos, a pesar de ser aficionados del mismo género de estrategia son más de juegos de mesa, por lo que tampoco juega con ellos.
Busca alguna aplicación con la que poder jugar sin necesidad de Internet y con la que poder jugar con sus amigos con poco esfuerzo.

ESCENARIOS

Erick se encuentra en el metro sin cobertura, dirigiéndose a su casa después de una dura jornada de trabajo. Necesita desconectar un rato por lo que enciende su dispositivo y se pone a jugar a Greaslider, ya que no necesita Internet. De esta forma se ha podido entretener durante el trayecto y ha podido llegar más descansado a casa.

Un día le enseña la aplicación a sus amigos. Como estos son reacios a bajarla Erick les deja jugar con su dispositivo, ya que permite jugar a dos personas a la vez. Después de probarla y ver lo sencilla y divertida que es terminan descargándosela.

Por la tarde dirigiéndose a un bar con un amigo en bus deciden jugar a Greaslider. Al no disponer de mesa y no encontrar asientos contiguos para jugar en un mismo dispositivo el amigo se descarga la aplicación y se ponen a jugar a distancia mediante Bluetooth. El viaje se les hizo corto y a la vez habían encontrado un juego para jugar en cualquier sitio con amigos.

## 2.4 Requisitos

**Funcionales**

- En el menú debe permitir escoger si queremos jugar contra la máquina o contra otro jugador.
- Una pantalla en la que se expliquen las reglas del juego.
- Derrotar a la IA debe ser un reto pero sin resultar imposible
- El jugador debe ser capaz de seleccionar una pieza pulsando sobre ella y ver los posibles movimientos.
- Al pulsar sobre uno de los movimientos disponibles de una pieza se debe ejecutar ese movimiento y actualizar el estado del tablero.
- Se debe controlar cuando una pieza ha sido rodeada para eliminarla del tablero.
- Determinar el ganador de la partida.
- Ofrecer la posibilidad de rendirse para que la partida no se haga eterna si el usuario no ve posibilidad de remontar.
- Notificar los cambios de turno para que el usuario sepa cuando tiene que jugar.
- Debe capturar que pieza selecciona el usuario.
- Mostrar los movimientos disponibles de la pieza seleccionada.
- Ejecutar el movimiento escogido entre los disponibles.
- Debe permitir la conexión entre dispositivos Android por Bluetooth.
- Se debe poder escoger, en el caso de que se escoja jugar mediante Bluetooth, quien será el anfitrión y el huésped de la partida.
- Permitir jugar a dos usuarios en un solo dispositivo.
- La conexión por bluetooth debe ser segura .

**No funcionales**

- El tablero y el resto de elementos deben adaptarse al tamaño de la pantalla.

- Las piezas deben mostrar un movimiento hacia su nueva posición y no aparecer de repente.

## 2.5 Casos de uso

| **Caso:** | Mover una pieza | **Fecha :** | Marzo 2019 |
|---|---|---|---|
| **Descripción:** El jugador puede seleccionar una pieza y moverla a una de las casillas disponibles. | | | |
| **Actores:** Jugador | | | |
| **Precondiciones:** El actor ya se encuentra en una partida y es su turno para jugar. | | | |
| **Flujo Normal:**<br>1.El actor pulsa sobre la pieza que desea mover.<br>2. El sistema comprueba si la pieza pertenece al actor y qué movimientos tiene disponibles esta pieza y los resalta en el tablero.<br>3.El actor pulsa sobre uno de los movimientos<br>6. El sistema muestra la animación de ese movimiento y actualiza la posición de las piezas en el tablero eliminando las piezas si alguna se ha acorralado. | | | |
| **Flujo Alternativo:**<br>3a. El actor pulsa sobre otra pieza en lugar de pulsar sobre un movimiento disponible.<br>3b. El sistema, después de comprueba qué movimientos tiene disponibles esta nueva pieza y los resalta en el tablero. | | | |
| **Poscondiciones:** El sistema comprueba el estado de las piezas para ver si ha acorralado alguna para eliminarla del tablero. | | | |

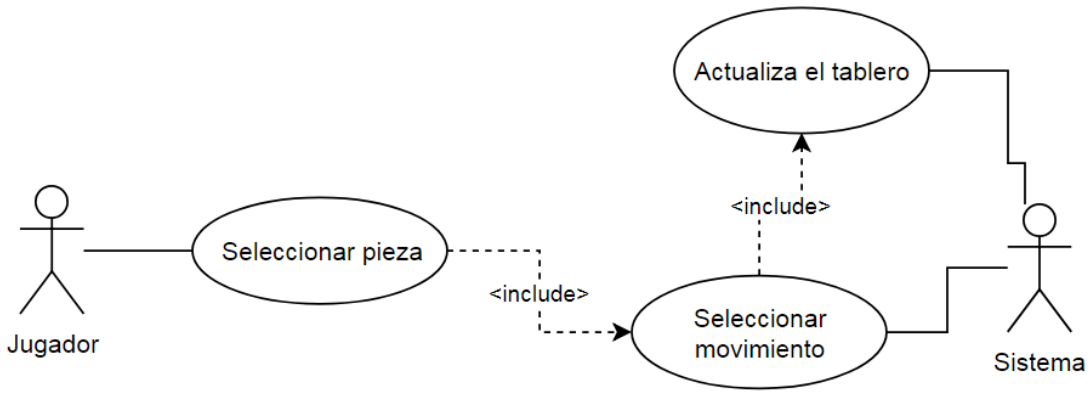


**Ilustración 6: Diagrama de Flujo. Mover una pieza**

| **Caso:** | Jugar contra la IA | **Fecha :** | Marzo 2019 |
|---|---|---|---|
| **Descripción:** Permite jugar de forma individual contra la IA | | | |
| **Actores:** Jugador | | | |
| **Precondiciones:** El usuario ya conoce como se juega y está en la pantalla principal de la aplicación. | | | |
| **Flujo Normal:**<br>1. El actor pulsa sobre el botón "Empezar partida", para empezar una partida individual.<br>2. El sistema muestra una nueva pantalla al usuario en la que aparece el tablero con las piezas.<br>3.El actor mueve una pieza.<br>4. El sistema calcula un movimiento mediante un algoritmo y lo ejecuta.<br>5. Se repite desde el punto 3 hasta que haya un vencedor. | | | |
| **Flujo Alternativo:**<br>3a. El actor desea abandonar la partida y pulsa sobre la opción "Salir del menú"<br>3b. El sistema lo devuelve al menú principal | | | |
| **Poscondiciones:** Se muestra por pantalla si el usuario ha ganado o perdido la partida y se le devuelve al menú principal. | | | |

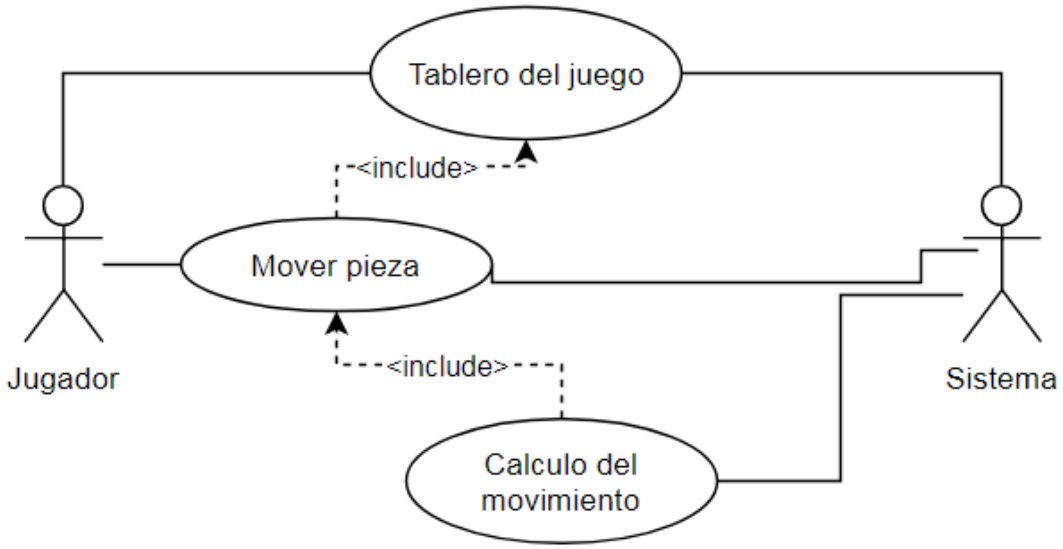


**Ilustración 7: Diagrama de Flujo. Jugar contra la IA**

| **Caso:** | Establecer conexión con otro usuario | **Fecha :** | Marzo 2019 |
|---|---|---|---|
| **Descripción:** Permite mediante la conexión Bluetooth empezar una partida con otro usuario. | | | |
| **Actores:** Jugador 1, jugador 2 | | | |
| **Precondiciones:** los usuarios se encuentran en la pantalla principal de la aplicación y tienen activado el Bluetooth. El jugador 1 quiere crear la partida a la que el jugador 2 se unirá. | | | |
| **Flujo Normal:**<br>1. Los actores pulsan sobre el botón "Multijugador".<br>2. El sistema muestra otro menú en el que permite escoger si quiere ser el anfitrión o conectarse a uno.<br>3. El jugador 1 selecciona "Anfitrión".<br>4. El sistema hace visible el dispositivo del jugador 1 para otros dispositivos bluetooth.<br>5. El jugador 2 selecciona ser huésped.<br>6. El sistema muestra la lista de anfitriones detectados a través de Bluetooth.<br>7. El jugador 2 pulsa sobre el dispositivo del jugador 1 que aparece en la lista.<br>8. El sistema establece una conexión entre ambos dispositivos y carga la pantalla de juego, en la que aparece el tablero con las piezas. | | | |
| **Flujo Alternativo:**<br>1. Un actor no tiene la conexión Bluetooth activada al pulsar sobre el botón "Multijugador".<br>2. El sistema notifica al actor de que la active para poder jugar y lo devuelve a la pantalla principal. | | | |
| **Poscondiciones:** Una vez mostrada la pantalla de juego los actores pueden comenzar la partida empezando por el anfitrión. | | | |

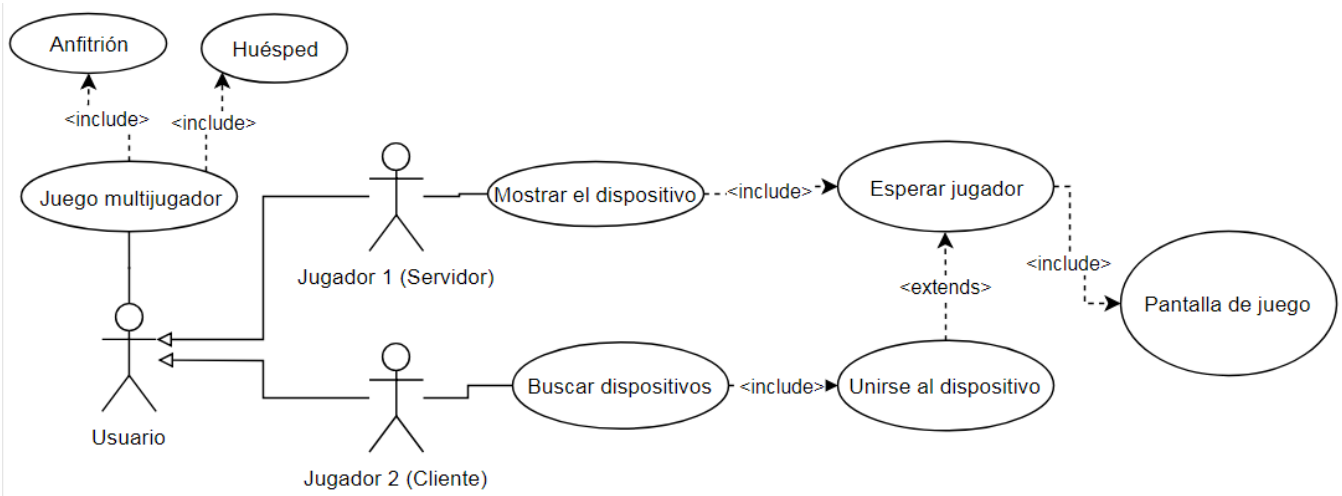


**Ilustración 8: Diagrama de Flujo. Establecer conexión**

| **Caso:** | Jugar contra otro usuario | **Fecha :** | Marzo 2019 |
|---|---|---|---|
| **Descripción:** Permite enfrentarse contra otro usuario. | | | |
| **Actores:** Jugador 1(Anfitrión), jugador 2(Huésped) | | | |
| **Precondiciones:** Los actores ya han establecido una conexión Bluetooth. | | | |
| **Flujo Normal:**<br>1. El jugador 1 empieza la partida, por lo que posee el turno.<br>2.El actor que posee el turno pulsa sobre la pieza que desea mover.<br>3. El sistema comprueba qué movimientos tiene disponibles esta pieza y los resalta en el tablero.<br>4.El actor pulsa de nuevo, pero esta vez sobre uno de los movimientos<br>5. El sistema muestra la animación de ese movimiento y actualiza la posición de las piezas en el tablero eliminando las piezas si alguna se ha acorralado.<br>6. El sistema envía el movimiento realizado por el jugador ese turno y lo envía al otro jugador a través de Bluetooth.<br>7. El sistema del otro jugador al recibir el turno muestra la animación del movimiento por pantalla, actualiza el tablero y pasa a tener el turno<br>8. Se repite desde el punto 2 hasta que haya un vencedor. | | | |
| **Flujo Alternativo:**<br>1. El usuario pierde la conexión o apaga el dispositivo.<br>2. El sistema finaliza la partida y vuelven al menú principal. | | | |
| **Poscondiciones:** Se muestra por pantalla a cada actor si ha ganado o perdido la partida y se le devuelve al menú principal. | | | |

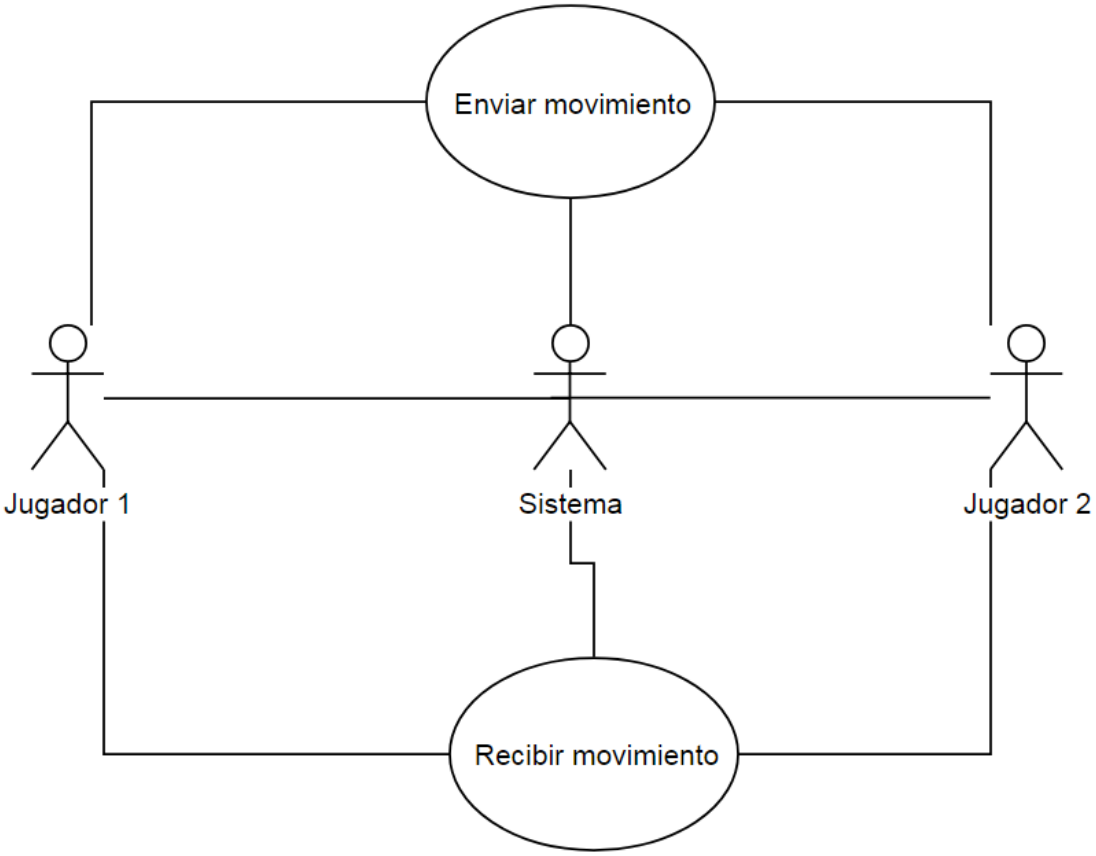


**Ilustración 9: Diagrama de Flujo. Jugar contra otro usuario**

## 2.6 Diseños de las pantallas

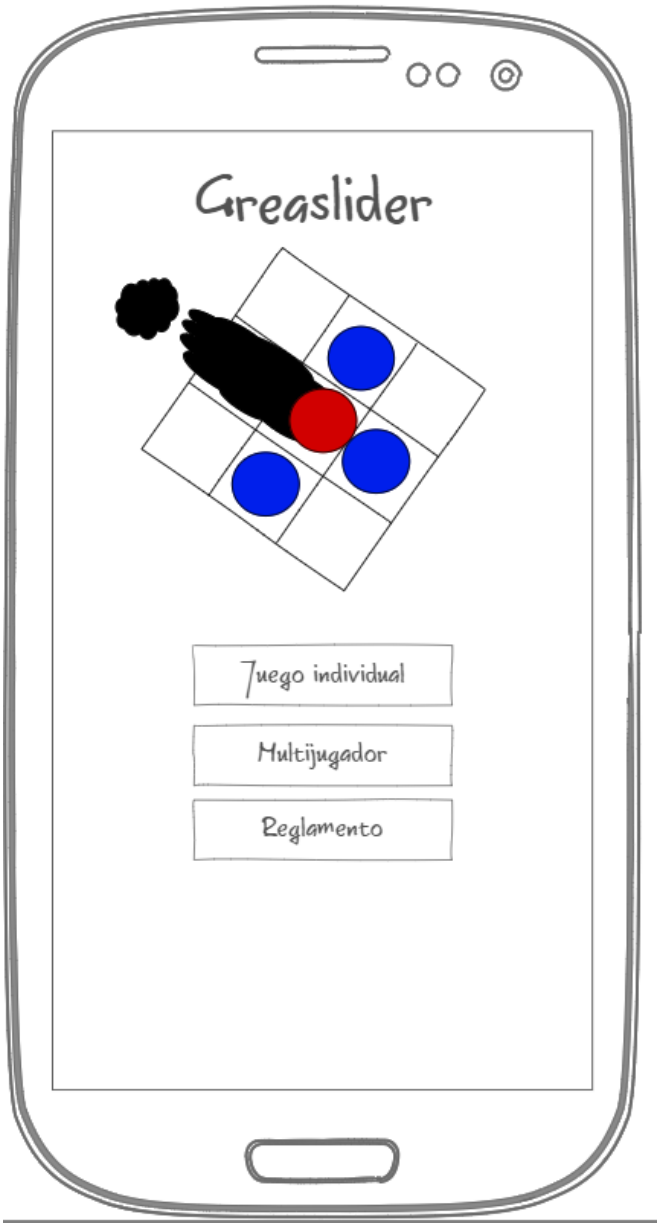


**Ilustración 10: Diseño. Menú principal**

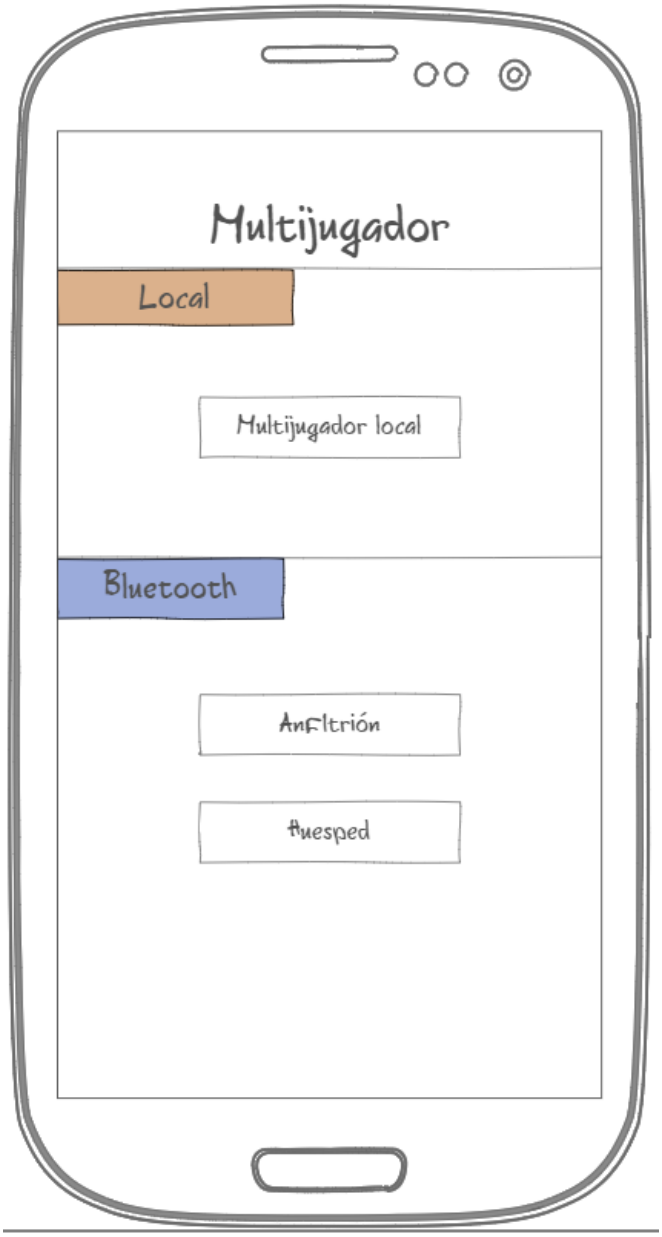


**Ilustración 11: Diseño. Menú multijugador**

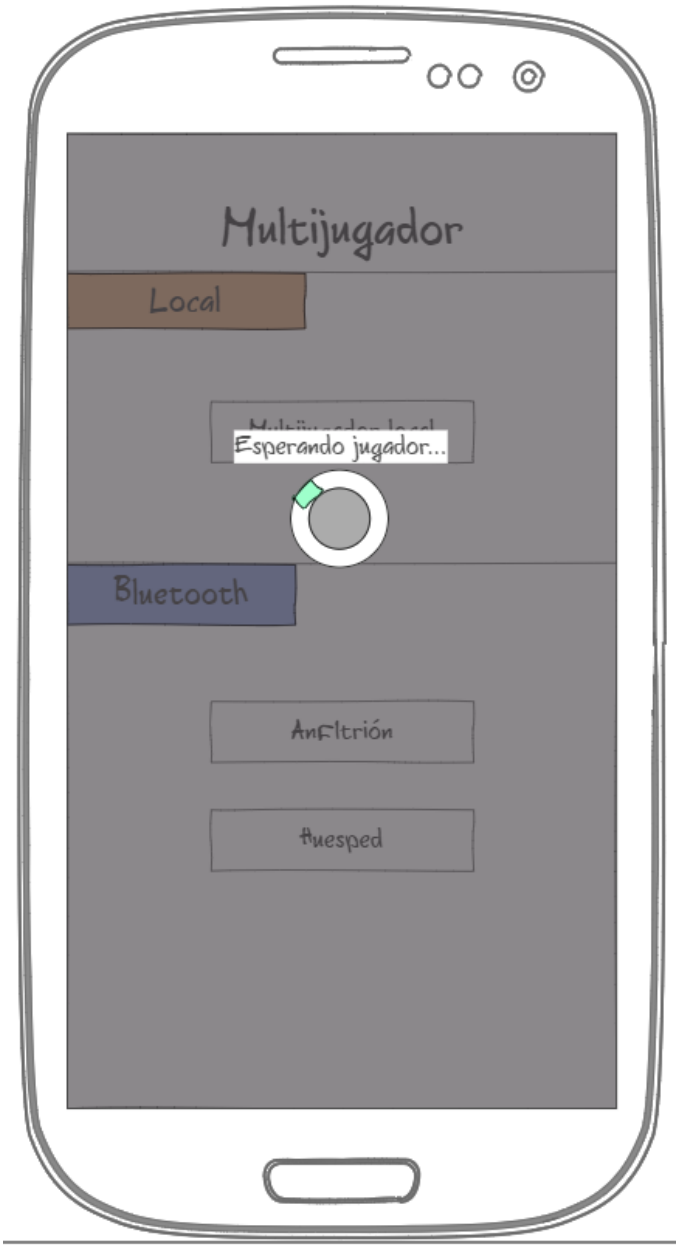


**Ilustración 12: Diseño. Menú Multijugador( pop up loading)**

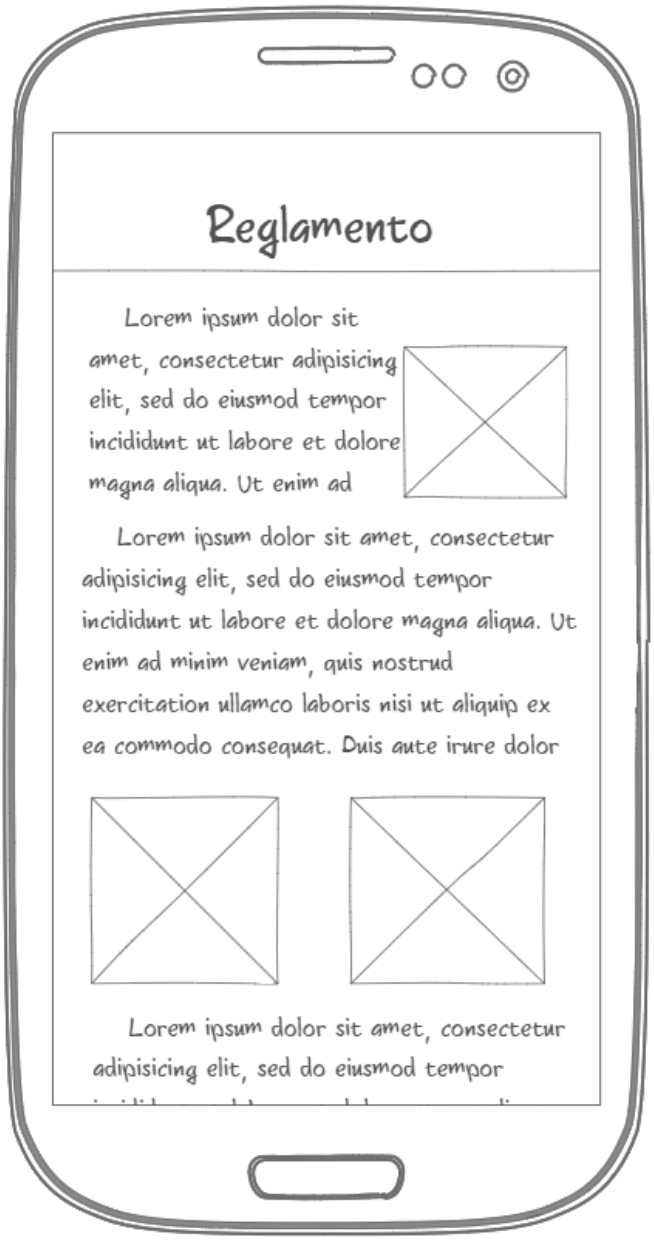


**Ilustración 13: Diseño. Pantalla del reglamento del juego**

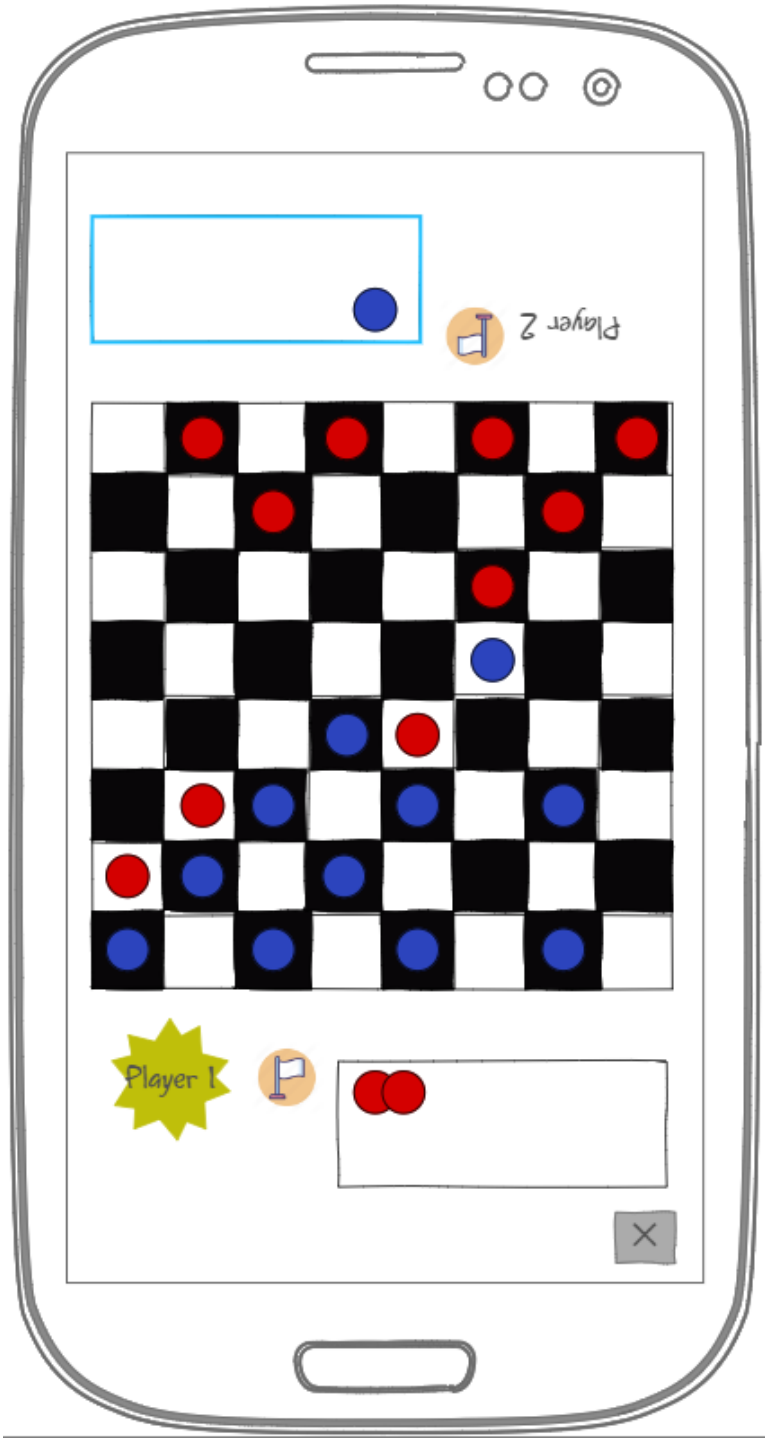


Ilustración 15: Diseño. Pantalla de juego


Ilustración 14: Diseño. Mensajes de victoria y derrota

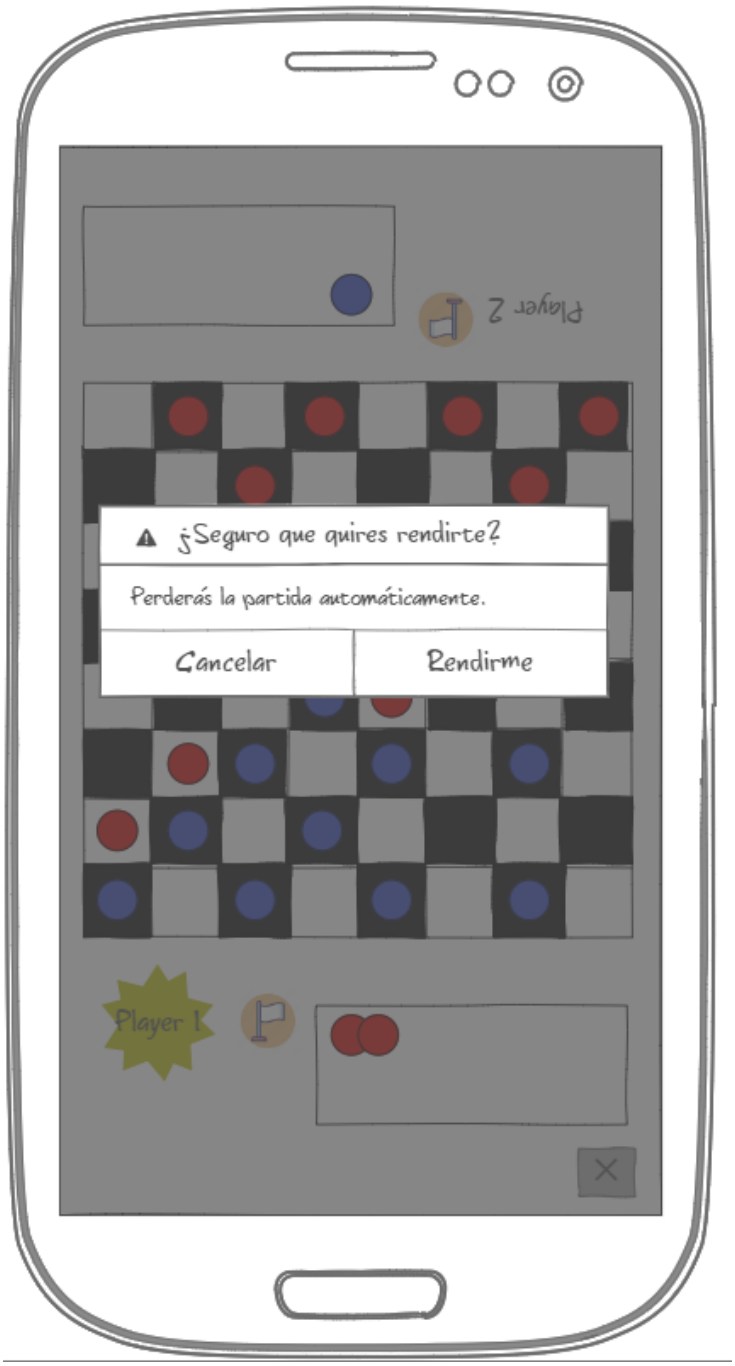


**Ilustración 17: Diseño. Dialogo para rendirse**


**Ilustración 16: Diseño.Dialogo para abandonar la partida**

El diseño se ha creado a partir del análisis de las aplicaciones de la competencia. Extrayendo de cada una de ellas las funcionalidades más útiles que se pueden incorporar a esta aplicación.

El diseño del menú principal es simple y se ha limitado a las tres opciones esenciales para poder disfrutar el juego:

- Juego individual : Para jugar solo contra la máquina.
- Multijugador: Permite acceder a la pantalla del menú que permite jugar contra otros jugadores.

- Reglamento: Muestra la pantalla con las reglas del juego.

El menú multijugador es la pantalla que contiene todas las diferentes opciones disponibles para que puedan jugar dos usuarios:

- Multijugador Local: Jugar en el mismo dispositivo dos usuarios, jugando por turnos.
- Bluetooth: Para jugar cada uno desde su dispositivo mediante Bluetooth.
    - Anfitrión: Opción para ser el creador de la partida.
    - Huésped: Permite buscar un anfitrión y unirse a su partida.

En un principio solo se contemplaba la opción de Bluetooth pero se ha añadido una opción para permitir jugar a dos usuarios en el mismo dispositivo.

La siguiente pantalla importante es la pantalla de juego. Se ha colocado el tablero justo en el centro de la pantalla, aprovechando la anchura del dispositivo.

Se ha decidido colocar el nombre de los jugadores, el cual se resalta si es su turno, y las cajas en las que irán las piezas eliminadas orientadas hacia donde empieza cada jugador. Así el diseño se adapta bien y queda mejor distribuido en las partidas multijugador sobre un mismo dispositivo. Cada jugador tiene un botón para rendirse, justo al lado de su nombre, para poder retirarse y perder la partida.
Además hay un botón para finalizar de inmediato la partida.
Para evitar que perdamos la partida, al pulsar alguno de los últimos botones de manera accidental, se muestra un dialogo para verificar que se ha hecho intencionalmente.

La ultima pantalla es la del reglamento. La primera opción fue la de que las instrucciones de como se juega solo se mostrarán la primera vez que se abre la aplicación. Sin embargo, al ofrecer la opción del multijugador en el mismo dispositivo, el usuario que no es propietario no sabrá jugar. Por ese motivo se ha decidido tenerlas accesibles desde el menú principal, ya que el juego es nuevo y no hay ninguno de similar.

## 2.7 Reglas del juego

Se trata de un juego de estrategia que se juega sobre un tablero 8x8 igual que se utiliza para el ajedrez o las damas.
La distribución inicial de las piezas es la que muestra la imagen siguiente.

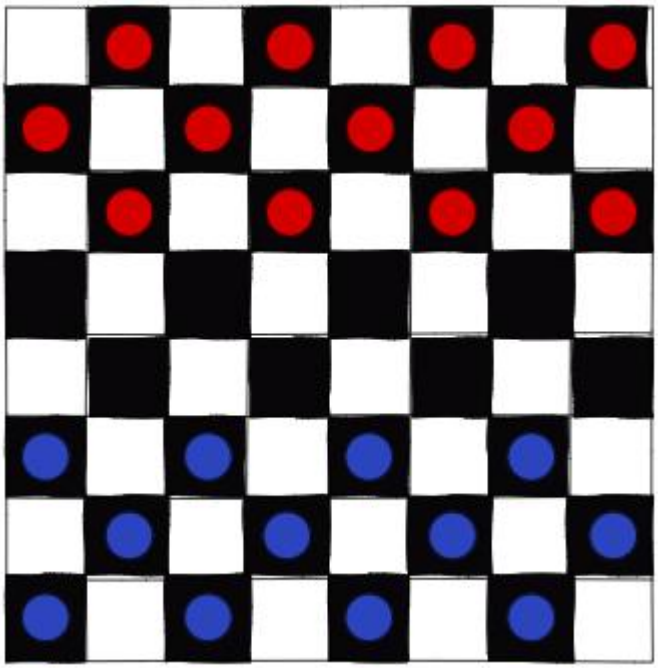

**Ilustración 18: Reglas del juego 1**

El objetivo consiste en eliminar las piezas del rival hasta que no le quede ninguna o hasta que el rival se rinda.
La forma principal de mover las piezas en este juego consiste en desplazarlas vertical u horizontalmente hasta que choquen. Pueden colisionar tanto con otras piezas, ya sean amigas o enemigas, como contra los márgenes. En la imagen siguiente se muestra gráficamente los tres movimientos disponibles de la pieza roja.

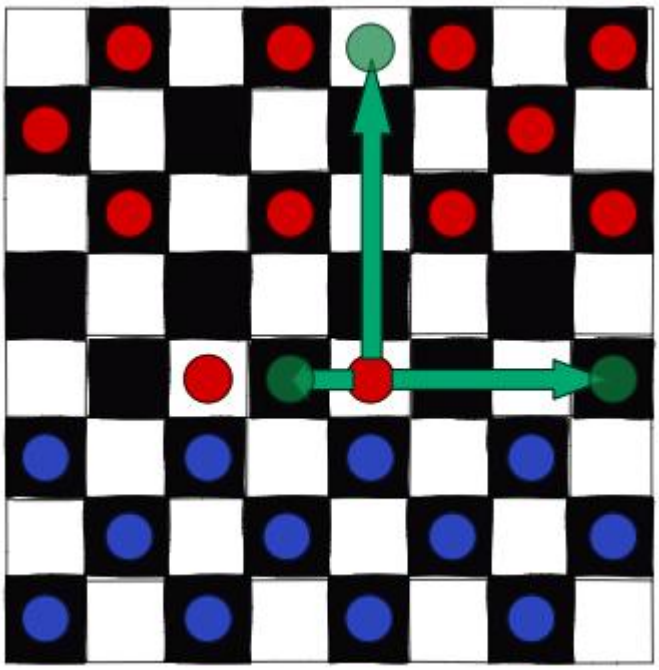

**Ilustración 19: Reglas del juego 2**

Para eliminar una pieza esta debe encontrarse completamente rodeada, es decir, que no pueda moverse. Independientemente de que la esté rodeando,

pueden ser tanto los márgenes del tablero como otras piezas, tanto enemigas como aliadas.
En la siguiente imagen se muestra como un movimiento puede eliminar una pieza, incluso si es aliada, ya que queda totalmente rodeada.

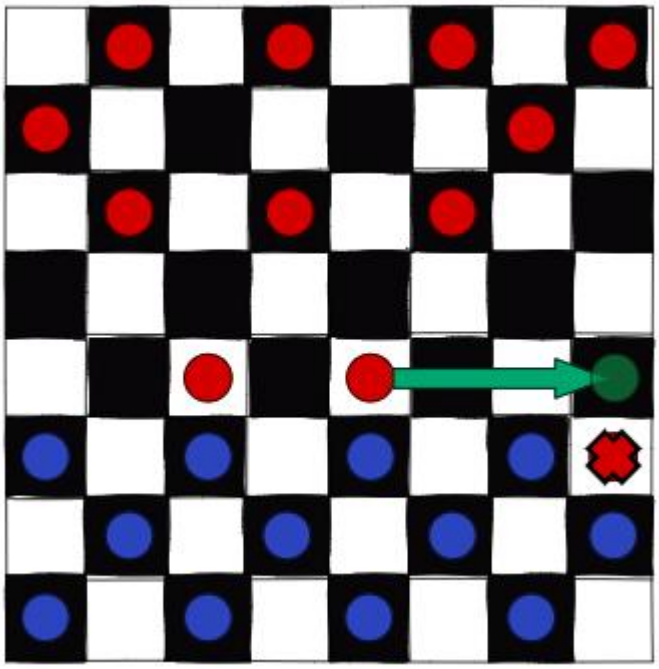

**Ilustración 20: Reglas del juego 3**

El otro movimiento disponible es el de empujar. En el caso en que tengamos más de una pieza de las nuestras en fila y contiguas frente a un número menor, en esa misma fila, de piezas enemigas podemos empujarlas una única casilla hacia atrás. Esto se podrá hacer siempre y cuando haya una casilla libre detrás,
Ha sido necesario añadir este movimiento para evitar estrategias posibles con solo el movimiento anterior que dejaban en punto muerto la partida.

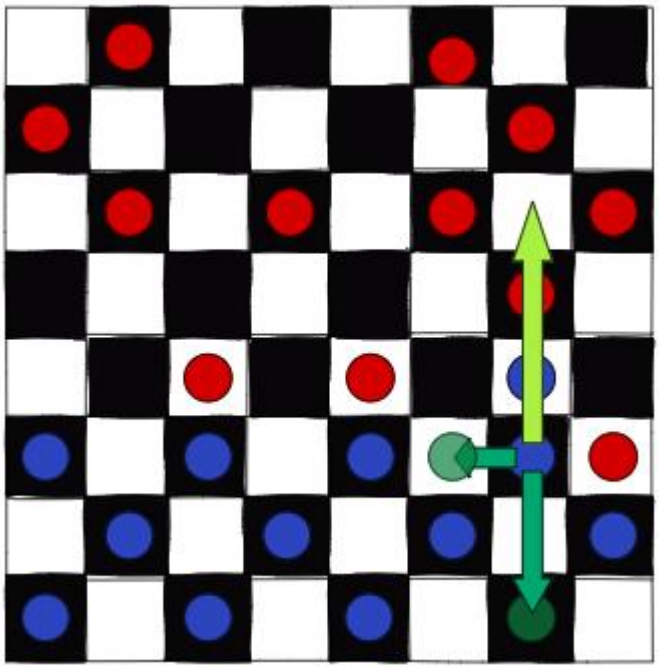

**Ilustración 21: Reglas del juego 4**

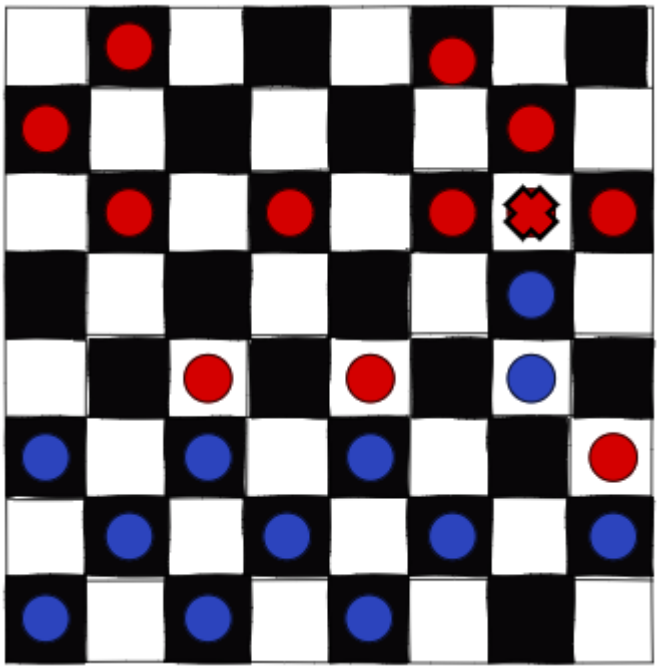

**Ilustración 22: Reglas del juego 5**

## 2.8 Arquitectura del sistema

Se ha escogido la arquitectura MVC para controlar las interacciones del usuario y mostrar la vista con los datos. El motivo por el que se ha escogido esta ha sido por lo fácil que resulta adaptar esta estructura en Android, ya que es la habitual. También para tener bien separados los distintos módulos de la aplicación. Además, como hay una comunicación con otros dispositivos, utilizando esta arquitectura se puede aislar esa gestión en el controlador.

En este caso las vistas corresponden a todos los XMLs que definen las pantallas de la aplicación Android. El controlador es el código Java que carga la vista y los datos, y controla las eventos. En cuanto al modelo, al no necesitar de base de datos, estará integrado junto al controlador y corresponde a todas las estructuras de datos necesarias para el correcto funcionamiento del sistema.

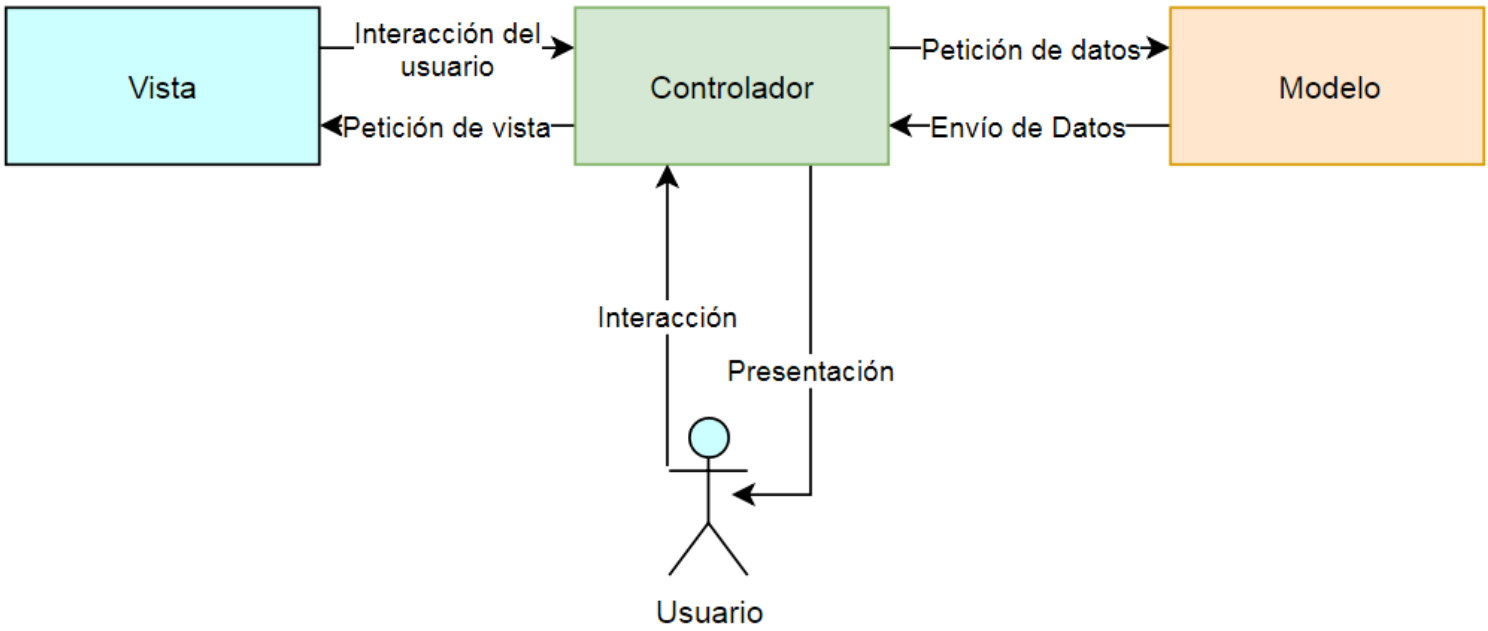


**Ilustración 23: Diagrama MVC**

Al no disponer de una Base de datos, los pocos que se manipulan ya se encuentran en el mismo controlador o en los recursos del sistema. En este caso el modelo corresponde a los recursos de texto del sistema. Los cuales, según el idioma del dispositivo carga un fichero u otro de texto. Por el momento está localizado para tres idiomas: español, inglés y francés.

También se ha implementado la arquitectura Cliente-Servidor para establecer la comunicación por Bluetooth. Ya que para establecer la conexión en un inicio se necesita que uno de los dispositivos se prepare para recibir la conexión y otro para hacerla. Una vez conectados se encuentran al mismo nivel y pueden empezar a enviar y recibir datos.

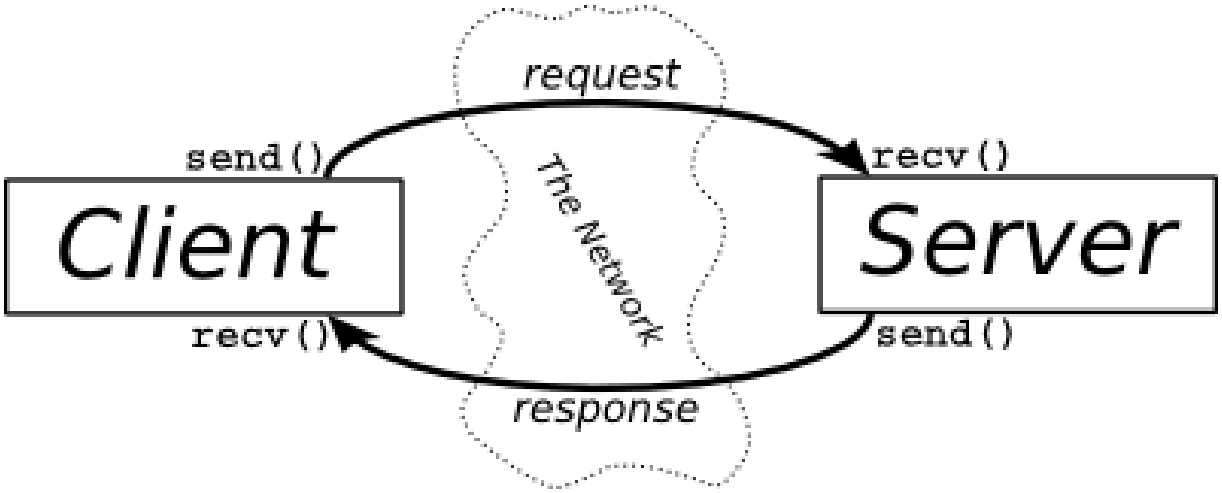


**Ilustración 24: Arquitectura Cliente-Servidor**

## 2.9 Diagrama UML de clases

A continuación se muestra el diagrama de las entidades y clases principales.

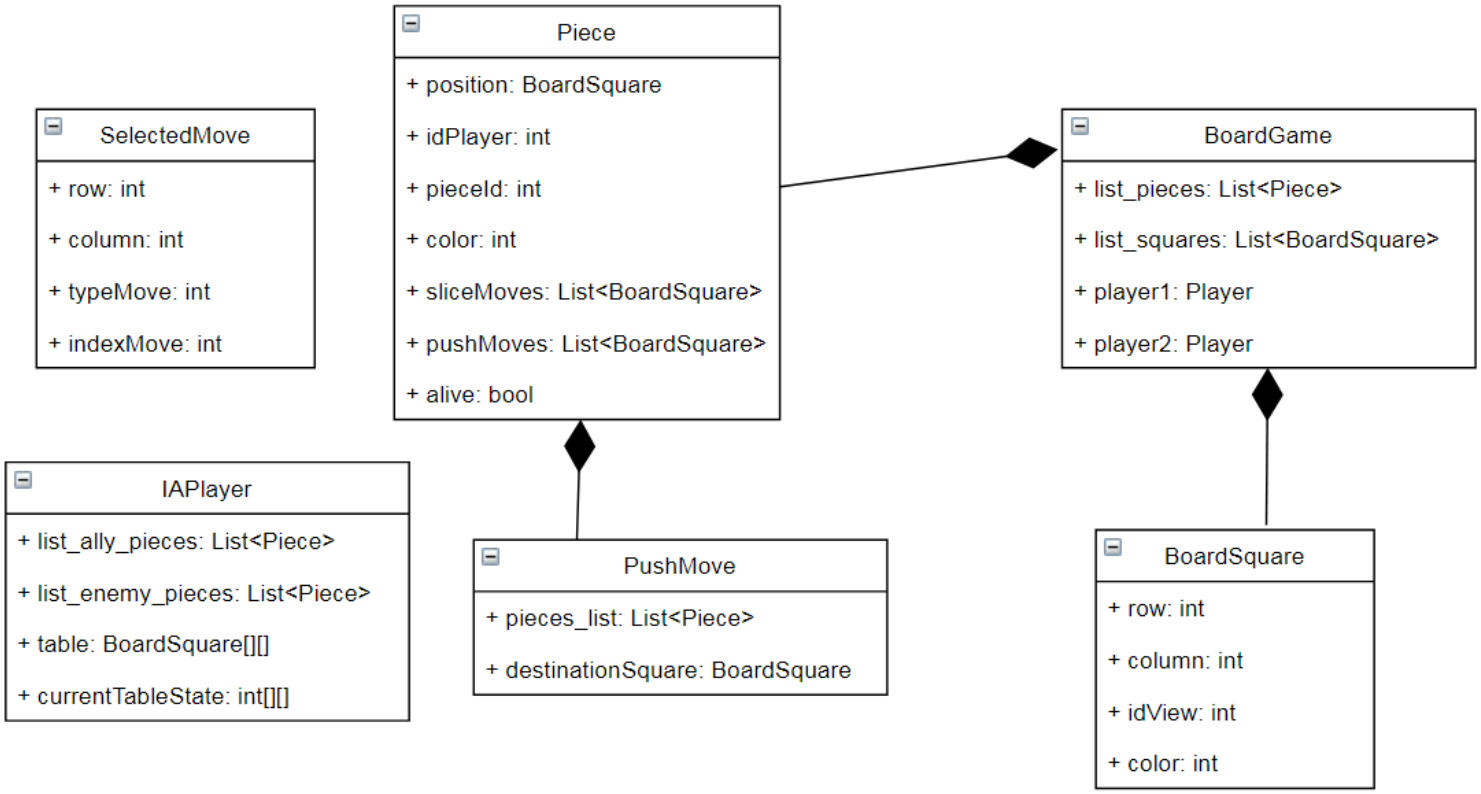


**Ilustración 25: Diagrama de clases**

# 3. Desarrollo Frontend

## 3.1 User Interface

Para hacer el diseño de la aplicación se han utilizado como referencia los diseños que se crearon en Ninjamock.

Con el objetivo de que las pantallas se pudiesen adaptar correctamente en la mayoría de dispositivos, se estudiaron los diferentes Layouts que ofrece Android. Entro todos ellos, los que se han analizado son:

- LinearLayout
  Una layout que posiciona las vistas que contiene una debajo de otra o una al lado de otra, dependiendo de la orientación especificada. Una vista muy útil para hacer el tablero, si se utiliza el atributo "android:layout_weight", para que cada vista ocupe el mismo espacio. [7]

- RelativeLayout
  Permite crear pantallas situando las vistas en una posición relativa a otras. Por lo que, utilizando el panel de diseño que ofrece Android Studio, se puede crear prácticamente cualquier interfaz. [8]

- ConstraintLayout
  Esta última da la posibilidad de crear diseños más complejos y evitar el exceso de vistas anidadas. Es similar al layout anterior, porque también utiliza relaciones con las demás vistas para colocarlas. Sin embargo, es más flexible. [9]

Después de valorar las distintas posibilidades se ha decidido utilizar ConstraintLayout. Los motivos por los que se ha escogido este diseño son:

1. Por tener una mayor flexibilidad que el Relativelayout, se adapta mejor a los distintos tipos de pantalla.
2. Es más eficiente al no anidar vistas.[10]
3. Es un tipo de diseño que no se ha trabajado mucho en el curso y me ha parecido interesante profundizar.

El resultado final de los diseños se muestra a continuación.


**Ilustración 26: App. Menú principal**


**Ilustración 27: App. Menú multijugador**


**Ilustración 28: App. Reglamento del juego**


**Ilustración 29: App. Pantalla de Victoria/Derrota**

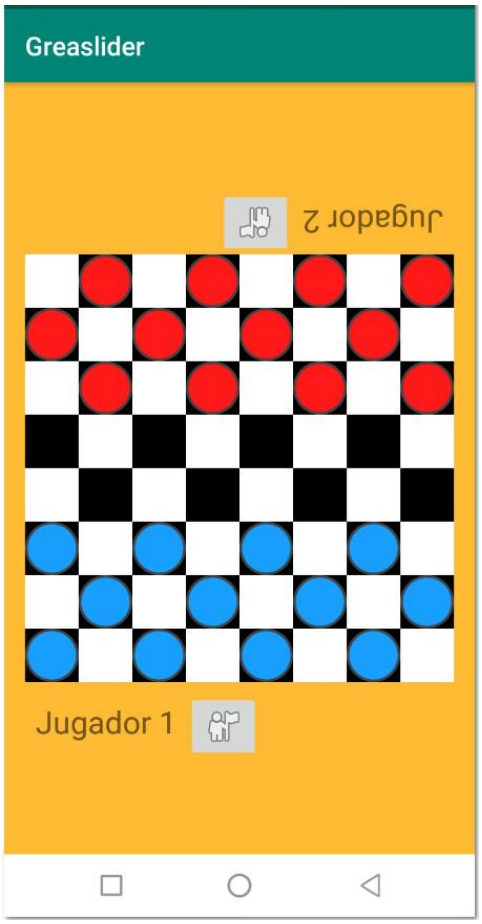


**Ilustración 30: App. Pantalla de juego**

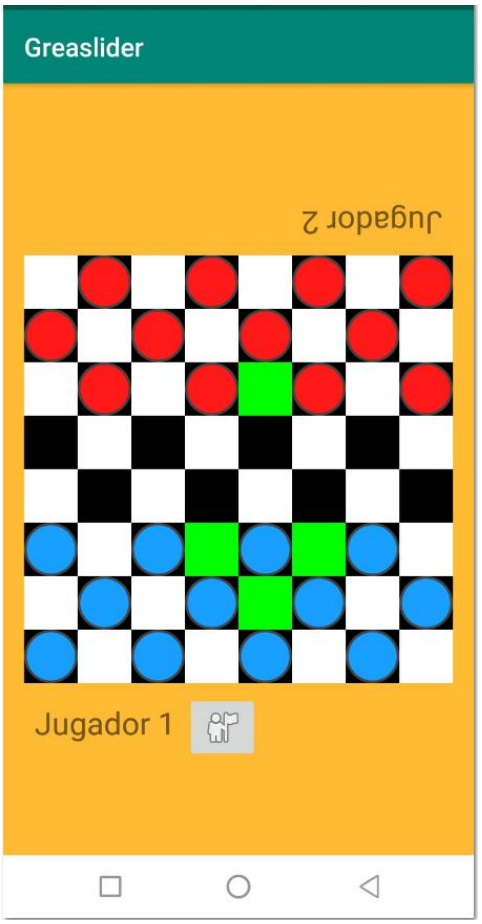


**Ilustración 31: App. Movimiento de pieza**


**Ilustración 32: Confirmación de rendición(Orientado al jugador 1)**

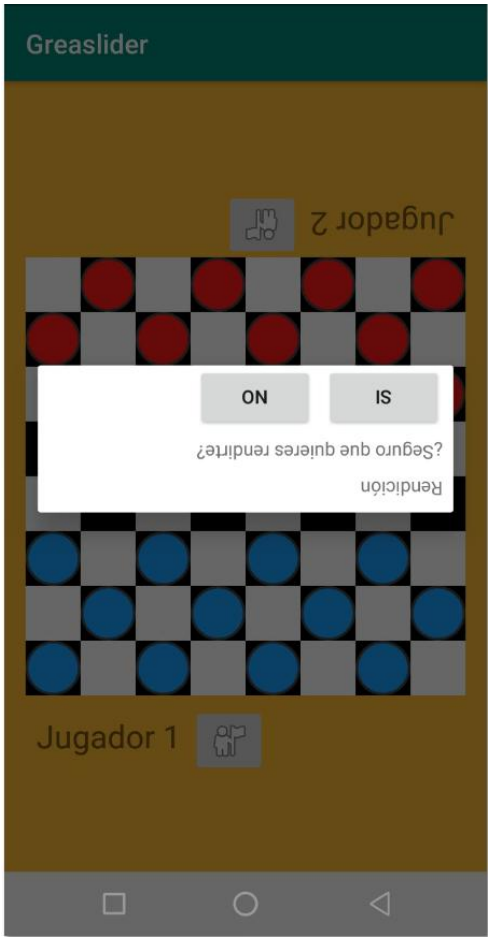


**Ilustración 33: Confirmación de rendición(Orientado al jugador 2)**

Todos se han hecho con ConstraintLayout, el único layout de más que se ha incluido es el ScrollLayout, para la pantalla en la que se explican las reglas del juego.
Incluso el AlertDialog, para rendirse en mitad de la partida, se ha tenido que diseñar con ConstraintLayout, ya que se necesitaba cambiar su rotación en el momento de mostrarlo en partidas de dos jugadores en el mismo dispositivo.

El diseño es trivial en la mayoría de vistas, lo único que se ha necesitado ha sido especificar las posiciones de los elementos y su aspecto en relación al tamaño de la pantalla. No obstante, el tablero se crea dinámicamente.
La distribución de las casillas y las piezas se crean y añaden al layout desde el Backend, creando una a una las vistas que los representan y los constraints para que se adapten bien al tamaño del dispositivo.

Para optimizar la aplicación, se ha querido limitar la cantidad de imágenes que utiliza a las necesarias para la explicación del reglamento y el icono del menú principal. Las casillas son vistas rectangulares a las que se les ha coloreado el fondo y las piezas se han definido en diseños XMLs.

También se ha decidido eliminar las cajas de los jugadores en las que se mostrarían las piezas eliminadas. El motivo de eso es porque en realidad no ofrece información que los usuarios ya no conozcan viendo el tablero y puede que en dispositivos más pequeños no quepa esa caja. Así que se ha decidido reservar este espacio para mejoras futuras.

# 4. Desarrollo Backend

## 4.1 Requisitos del sistema

El juego se ha desarrollado para la plataforma Android por lo que para probarlo es necesario un dispositivo con ese sistema operativo.

En la siguiente imagen se muestra  la distribución de la fragmentación de Android cuando se empezó a desarrollar el proyecto.

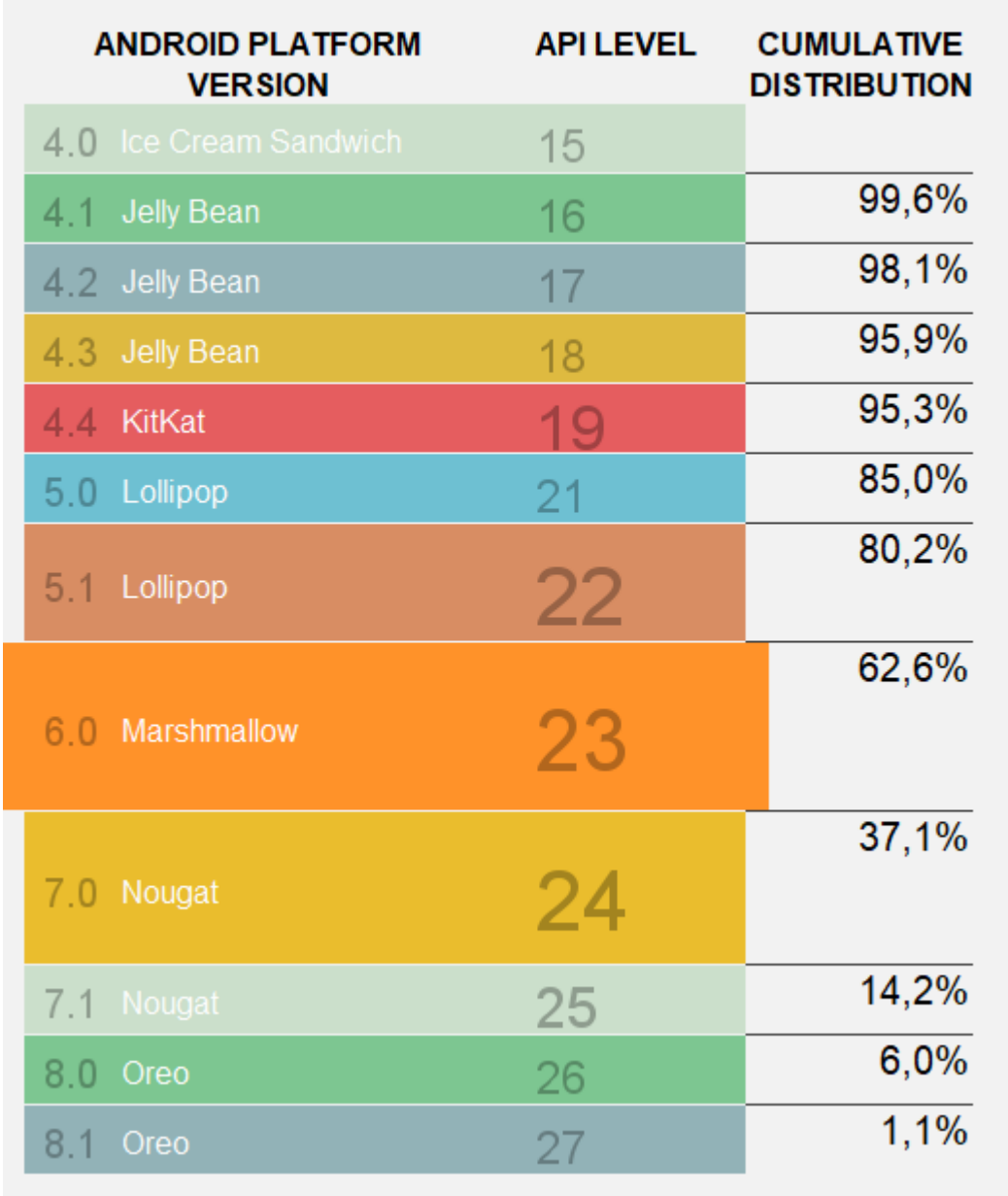


**Ilustración 34: Fragmentación de Android**

Con el objetivo de que este juego esté disponible para la mayoría de usuarios pero sin perder eficiencia se ha decidido implementar para dispositivos con el API 23 como mínimo. Pero se ha compilado para el último API el 28 para aprovechar todas sus mejoras.
Por eso, para poder instalarse este juego, el dispositivo necesitará la versión de Android 6.0 Marshmallow o una superior.

## 4.2 Movimiento de las piezas

**Selección de Movimiento**

Se han valorado tres posibles formas en que el usuario podría indicar qué movimiento quiere hacer y con qué pieza.

1. Pulsando sobre una pieza para escogerla y, una vez seleccionada, se muestre en la parte inferior de la pantalla flechas que indican sus movimientos disponibles.
2. Como se trata de un juego de deslizar la otra alternativa es que el usuario realice exactamente eso. Pulsando sobre una pieza y deslizando el dedo. En el momento en el que el usuario lo levanta se puede calcular la dirección que ha escogido.
3. Finalmente tenemos la opción que se utiliza en la mayoría de juegos de este estilo. El jugador selecciona una pieza y se le muestran las casillas a las que puede ir. Después pulsa sobre esa casilla y se efectúa el movimiento.

El primer tipo de selección de movimiento se ha descartado porque no es muy intuitivo e interrumpe al usuario obligándolo a apartar la vista del tablero para centrarse en la de los botones de debajo.

El segundo no permite al usuario rectificar la pieza escogida. Cuando se ha pulsado sobre ella, en el momento en que levante el dedo, se genera el movimiento. Además, es más fácil que el movimiento que queramos realizar no sea el que al final se haga, si no somos lo suficiente precisos en el momento de hacerlo.

Finalmente, nos quedamos con el tercero, porque aporta además una visión estratégica del tablero permitiéndonos ver todos los posibles movimientos si no estamos familiarizados con el juego.

Una vez definida la acción del usuario, como se explicó al inicio, se quiere que el movimiento de la pieza sea lo más real posible y que no desaparezca simplemente de una casilla y aparezca en otra. Es decir, que muestre la traslación que hace.

Para hacer este movimiento nos apoyamos en los constraints. En el momento que se ha solicitado el movimiento se conoce la pieza que lo ha solicitado y la casilla destino. Por lo que se actualizan los constraints de la pieza ,que estaban relacionados con la casilla en la que se encuentra actualmente, y se relacionan con la casilla destino. Todo eso junto con la llamada "beginDelayedTransition" del TransitionManager permite ver la traslación de la pieza de una casilla a otra.

**Tipos de Movimientos**

Existen dos tipos de movimientos que pueden hacer las piezas.

- Slide

  Es el básico que consiste en desplazarse en una dirección hasta chocar con los límites del tablero u otra pieza.

- Push

  Si se tienen varias piezas aliadas contiguas en fila, contra un número inferior de piezas enemigas adyacentes en esa misma fila, se pueden empujar una posición si hay una casilla disponible detrás.

La manera en la que se han representado estos movimientos al jugador es marcando las casillas de selección de movimiento de distintos colores. Las casillas verdes son para un movimiento Slide y las azules para los Push, como se muestra en la siguiente imagen

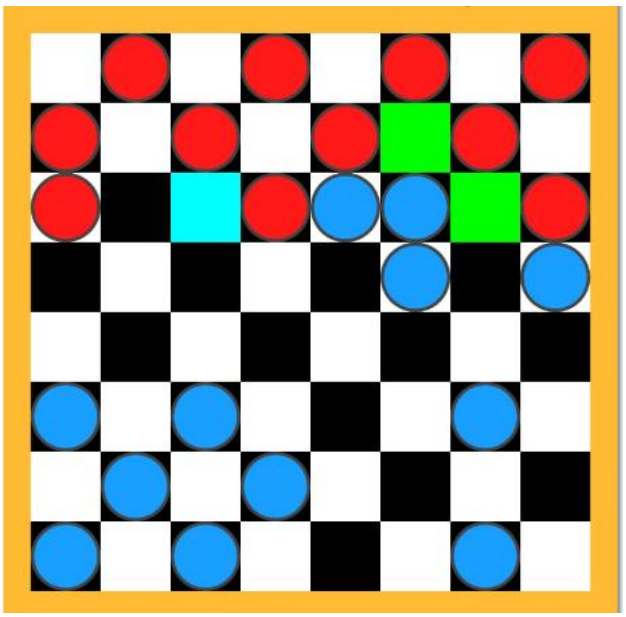

**Ilustración 35: Movimientos del juego**

## 4.3 Lógica común del Juego

Dentro de los diferentes modos de juego hay código común en todos ellos, la lógica interna de la partida.

Una vez generado el tablero y guardadas todos los identificadores asociados a esas vistas dinámicas, empieza el juego.
Siempre empiezan las piezas azules que son las del jugador 1. Y, al ser una partida por turnos, sólo puedes seleccionar las piezas cuando te toca el turno. Una vez hecho el movimiento, el turno cambia y le toca al otro jugador.

En cuanto a los movimientos que puedes seleccionar de cada pieza, se calculan previamente. La primera vez que se evalúan es cuando se genera el tablero y, a partir de ahí, se recalculan al final de cada turno, ya que la distribución del tablero cambia.

Antes de decidir que se evaluasen todos los movimientos de todas las piezas en cada turno, se ha probado calculando sólo los de las piezas en el momento en el que son seleccionados. Viendo que la diferencia de tiempo era imperceptible y que tener todos los movimientos ayuda al cálculo posterior del mejor movimiento de la IA, se optó a hacer una evaluación de todos en cada turno.

Explicado el funcionamiento de los turnos y el de los movimientos, sólo queda el de cómo vencer la partida. Al final de cada turno se eliminan las piezas que han sido rodeadas y, si uno de los jugadores se queda sin piezas, el juego termina, llamando a otra pantalla que muestra un mensaje de victoria y derrota orientado hacia los jugadores, según quién de los dos haya ganado. La otra forma de vencer es que uno de los dos se retire pulsando sobre el botón de la bandera blanca y confirmando la acción.

## 4.4 Juego Individual

Para el desarrollo de este modo de juego se ha necesitado crear la clase IAPlayer con la lógica necearía para que, conociendo las piezas y su distribución, sea capaz de escoger movimientos. Estos movimientos deben ser lo suficientemente acertados para que la partida suponga un reto para el jugador.

Al ser un juego totalmente nuevo y no haberlo podido probar lo suficiente, la elaboración de una heurística compleja, como la que disponen los juegos de ajedrez, no es plausible. La implementación de esta heurística es una posible mejora futura de esta aplicación.

A pesar de eso, se ha desarrollado un pequeño algoritmo para escoger movimientos con un mínimo de criterio para que la IA sea capaz de atacar y matar las piezas del usuario, así como también defender las suyas cuando se ven amenazadas.

El algoritmo consiste en cuatro pasos que se ejecutan en el siguiente orden hasta encontrar un movimiento. Es decir, pasa al siguiente paso si en el actual no ha encontrado ninguno.

1. Prioriza localizar una pieza enemiga que pueda matar.
2. Si no puede matar a ninguna, salvará sus piezas, por lo que si hay una que esté prácticamente rodeada, la moverá.
3. Realiza un PushMove, un movimiento que empuja varias piezas, si hay alguno disponible. Para arrinconar de esta forma al enemigo.
4. Finalmente, si no se cumple ninguna de las condiciones anteriores, se hace un movimiento aleatorio.

Para identificar el movimiento seleccionado se ha creado la clase SelectedMovement que almacena todos los datos mínimos necesarios para poderlo procesar en la pantalla principal.

Una vez la IA ha sido inicializada actúa de segundo jugador al final de cada turno, si se está jugando en el modo individual, permitiendo jugar solo.
Finalmente, para conseguir que los movimientos realizados por la IA, no sean instantáneos y se muestren a la vez que los del jugador, cosa que puede llevar a confusión, se calculan en un hilo de ejecución alternativo. Así, con un delay de un segundo se consigue que los movimientos se vean por separado. Además, realizando el cálculo del movimiento más óptimo en un hilo a parte, permite en un futuro mejorar la IA con heurísticas y cálculos más complejos sin afectar al rendimiento del hilo principal, por lo que el usuario no se vería afectado.

## 4.5 Multijugador Local

El primer modo de juego que se ha implementado. Ofrece la posibilidad de que dos usuarios jueguen un uno contra uno en un mismo dispositivo.

Está compuesto por todo el desarrollo común del juego y en lugar de ceder el turno a la IA permite seleccionar las piezas del Jugador 2 cuando le llega su turno, de la misma forma en que lo hace el primer jugador.

## 4.6 Bluetooth

Esta ha sido la implementación más compleja del proyecto.
En las siguientes imágenes obtenidas de[11] se explica la comunicación entre dispositivos Android mediante Bluetooth.

### Configuración previa

Lo primero: para poderse comunicar se han declarado los permisos en el manifest de Bluetooth y el "ACCESS_COARSE_LOCATION". Este último se tiene que pedir en tiempo de ejecución por temas de seguridad.
Una vez solicitados los permisos se pide la activación del Bluetooth del dispositivo si no lo tiene activo.

### Identificador único universal

Para que la comunicación sea segura se ha especificado un identificador único universal (UUID) para la conexión. Es un formato estandarizado de 128 bits para definir un identificador lo suficiente grande para generarse al azar sin que haya conflictos. En este caso se ha utilizado para identificar de forma única el servicio Bluetooth de esta aplicación, de

forma que sólo se puedan conectar a él dispositivos que conozcan ese UUID.

**Establecimiento de la comunicación**

Finalizado esto, el dispositivo ya está preparado para iniciar la conexión Bluetooth. Para realizarlo se ha consultado la guía de Bluetooth de la documentación oficial de Android.[12]

En esta aplicación se activa el Bluetooth y los permisos en la pantalla del menú multijugador. Una vez hecho eso, en uno de los dispositivos se ejecuta el código para hacer de servidor y en el otro el de cliente.

El servidor empieza abriendo el socket de servidor y haciendo visible el dispositivo al resto durante un tiempo.
El cliente, en cambio, inicia una búsqueda de dispositivos Bluetooth y trata cada uno que encuentra en un hilo diferente. Se ha creado de forma que sólo admite una única conexión, la primera en aceptar.

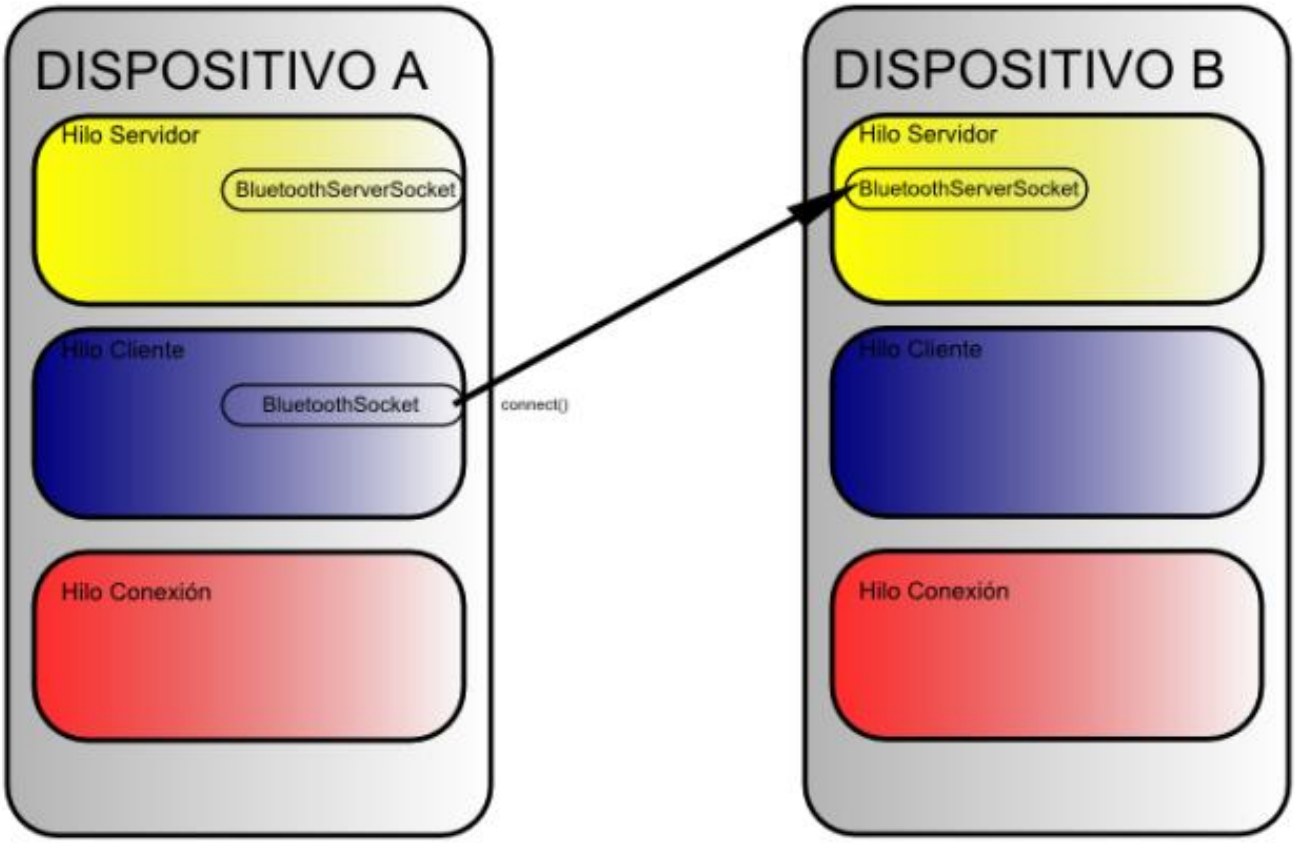


**Ilustración 36: Busqueda de dispositivo Bluetooth**

Al encontrar al que hace de servidor, a ambos les aparece una alerta para vincular los dispositivos. Cuando la aceptan, el servidor crea el BluetoothSocket en el hilo que utilizará para la comunicación.

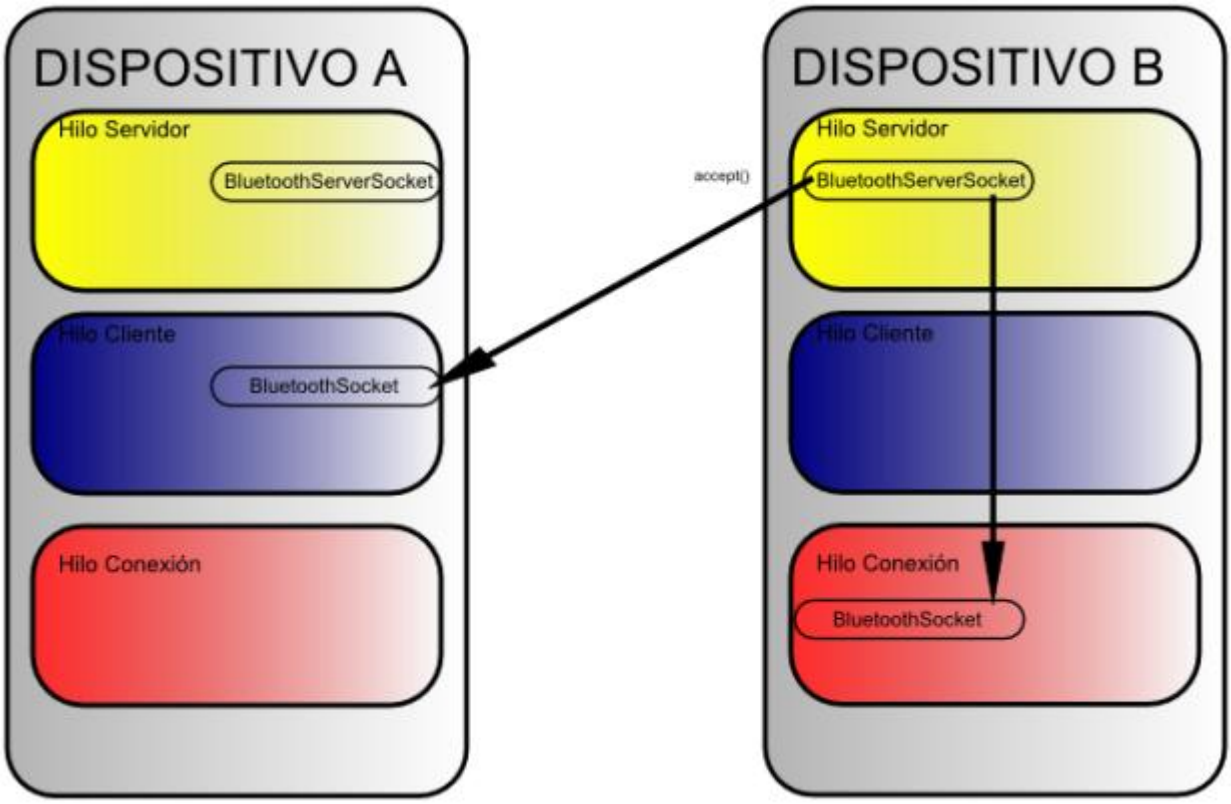


**Ilustración 37: Vinculación de los dispositivos**

El cliente, al aceptarse su conexión, crea otro BluetoothSocket igual que el servidor.
En el caso de que ya estén vinculados, se salta la petición de vinculación de dispositivos, y se crea automáticamente el BluetoothSocket sin preguntar.
Una vez que ambos tienen su BluetoothSocket pueden comunicarse. Sin embargo, la pantalla de juego está en otra Activity y con el objetivo de conservar ese socket se ha implementado el patrón de diseño Singleton.
La clase que lo implementa hereda de Aplication, por lo que es accesible desde todas las pantallas de la aplicación y en esa clase se guarda la referencia al BluetoothSocket.

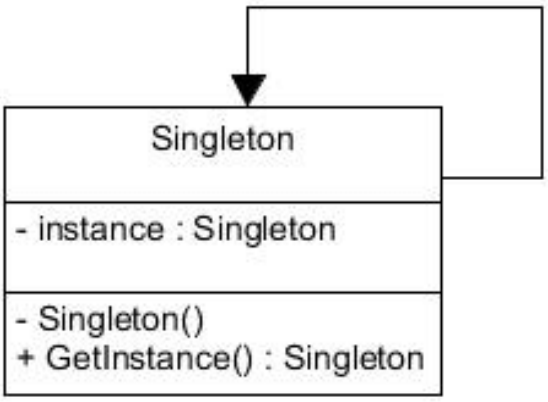




**Ilustración 38: Patrón de diseño Singleton**

Se llama a la pantalla del juego en Bluetooth y una vez cargada se crea e inicia el hilo de conexión con el Socket recuperado del Singleton para empezar la transferencia de datos. Como se puede ver en la imagen siguiente.

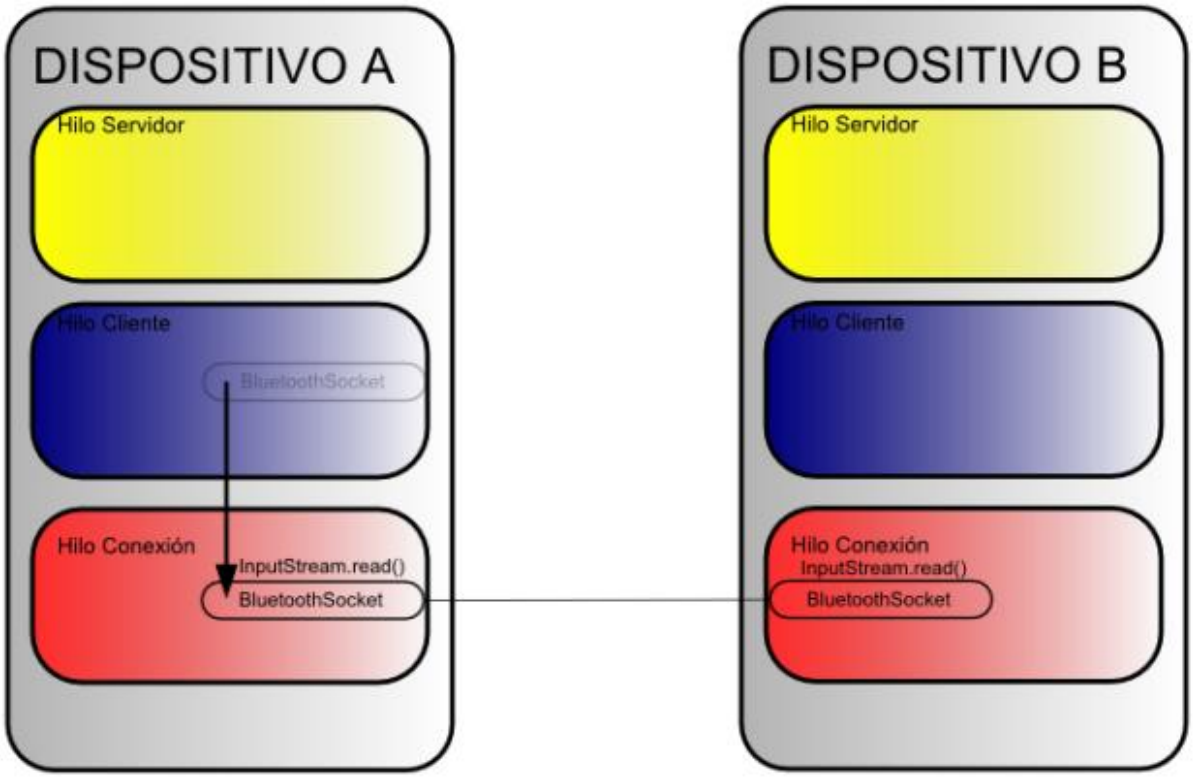


**Ilustración 39: Conexión Bluetooth establecida**

Para enviar información al otro dispositivo se hace directamente. Sin embargo, para recibirla, se ha creado un Handler para tratar los datos leídos del buffer.

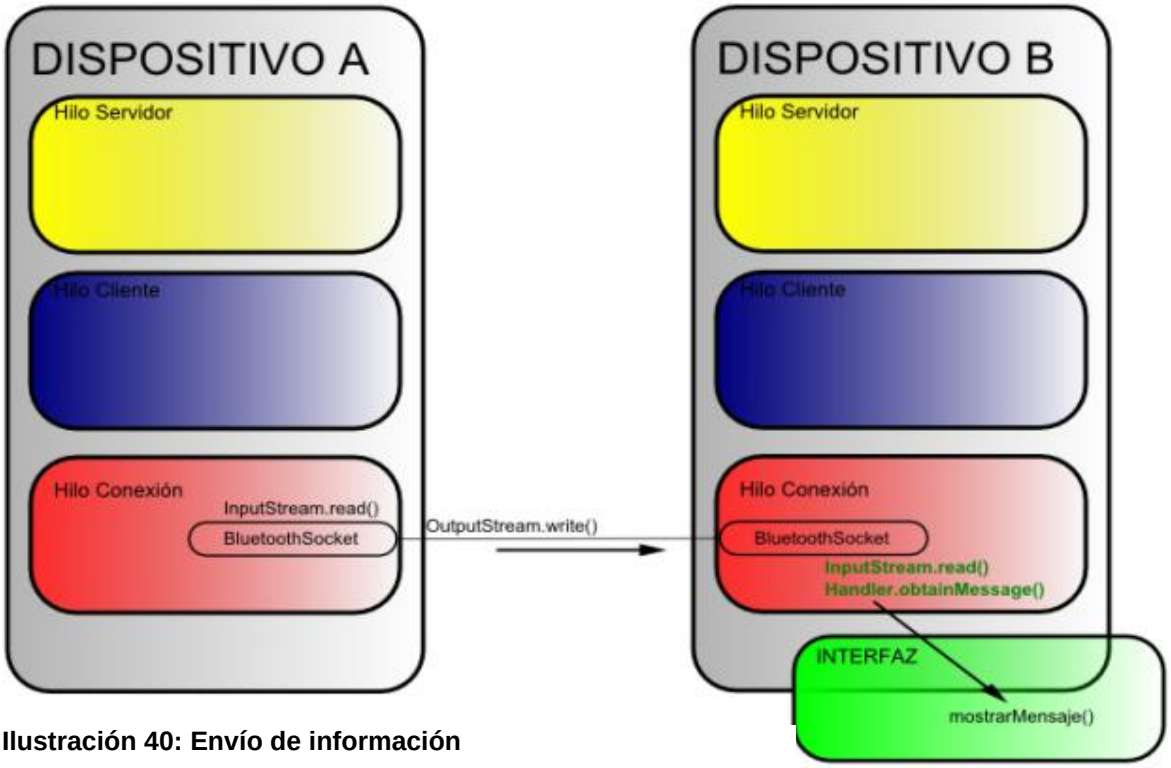


**Ilustración 40: Envío de información**

**Formato de los mensajes**

Con el objetivo de optimizar el intercambio de información y su tratamiento, lo que se intercambian los dispositivos es un mensaje en formato JSON. Concretamente un array de enteros de cuatro posiciones.

Cuando un usuario selecciona un movimiento en su dispositivo, no sólo lo ve él sino que a la vez se envía un array codificado en JSON al otro dispositivo.

El formato es:
"[**fila**, **columna**, **tipo de movimiento**, **índice de la posición del movimiento**]"

Con la fila y la columna se puede localizar la pieza. El tipo de movimiento puede indicar tres cosas:

- Un SlideMove, para buscar en la lista de estos movimientos.
- Un PushMove, también para saber en qué lista buscar
- Si se rinde y automáticamente ganas la partida

Finalmente, si el tipo de movimiento no es la petición de rendición, se utiliza el índice final para coger el movimiento de la lista especificada.

### 4.7 Optimización

Una vez implementado todo lo anterior se reestructuró el código. Ya que para cada tipo de juego se implementó en clases diferentes, porque ,a pesar de tener código en común, había suficientes diferencias que complicaban el desarrollarlo todo en una única clase. Por si se daba el caso de que por falta de tiempo no se llegase a completar todo.

Se ha conseguido separar todo el código común relacionado con el cálculo de movimientos . Este código no necesita tener acceso a las vistas, por lo que se ha centralizado dentro de la clase estática "MovementsUtils", para simplificar el código del flujo del juego principal.
Además, se han unificado los tres códigos de los diferentes tipos de juego en sólo uno totalmente comentado.

Todo esto con el fin de mejorar la calidad del proyecto y facilitar el mantenimiento y reutilización del código en futuras mejoras.

# 5. Pruebas

## 5.1 Juego Individual

Desarrollada la IA, se probó jugando contra ella, de forma que generase movimientos en los cuatro pasos posibles. Se jugó hasta vencerla y probar así que no salta ninguna excepción.

Hubo problemas con la utilización de un hilo alternativo para calcular el movimiento de la IA, debido a que las vistas de Android no se pueden modificar desde otro hilo que no sea el principal. por lo que se solucionó enviando un mensaje desde el hilo alternativo al principal, el cual lo recupera con un objeto Handler, para mostrar el movimiento por pantalla.

## 5.2 Juego Multijugador

En este desarrollo, al ser el primero en implementarse, se ha probado el flujo principal del juego.

- El cálculo correcto de movimientos de cada pieza.
- La interacción con el usuario.
- La animación del desplazamiento de las piezas.
- El control de las piezas muertas.
- La actualización de movimientos.
- Comprobar si uno de los jugadores se ha quedado sin piezas.
- La acción de rendirse.
- El control adecuado de los turnos para que solo pueda mover piezas a quien le toque.

## 5.3 Bluetooth

Para probar la comunicación y el correcto funcionamiento de las conexiones, así como el envío de datos, no se ha podido hacer dentro del entorno de Android Studio.
Las máquinas virtuales no tienen la capacidad para probar las comunicaciones Bluettoth, por lo que las pruebas se han hecho instalando el juego en dos dispositivos físicos distintos mientras se analiza uno de ellos.

Para las pruebas se han utilizado los siguientes dispositivos:

- Oukitel Mix 2
- Xiaomi

Se ha podido probar el establecimiento de la conexión, tanto siendo servidor como cliente, y el envío de mensajes a través del Socket.

Ha habido muchas incidencias en este desarrollo por falta de permisos, como el nuevo "ACCESS_COARSE_LOCATION", no tratar bien los

diferentes hilos de ejecución( threads) y el control del buffer. Pero se han podido solucionar todas las encontradas hasta ahora.

## 5.4 Pruebas Unitarias

La mayoría de funcionalidades requieren de acceso a un layout y vistas. También hay otros métodos que por el hecho de estar pensados para ser usados sólo en este juego y haber sido centralizados, para que el código se mantenga más fácilmente, requieren de una gran cantidad de parámetros. Por ese motivo, se han probado sólo un par de funciones con test unitario, ya que para hacer unos fiables del resto de la aplicación se necesitarían más horas de las que se disponen para hacer el proyecto.

## 5.5 Pruebas Localización

Se ha probado la aplicación en un dispositivo cambiándole el idioma para probar el funcionamiento del cambio de los textos de la interfaz de la aplicación.

# 6. Conclusiones

## 6.1 Conclusiones del trabajo

He aprendido que no siempre se dispone de todo el tiempo que se cree, que pueden surgir imprevistos y por eso hubiese valido la pena haber limitado algo más el trabajo y disponer de más tiempo para tratar incidencias.
También, que es muy arriesgado proponerse objetivos y utilizar tecnologías nuevas sin tener ninguna experiencia en ellas. Por mucho que se analizó en un principio el proyecto, surgieron muchas incidencias inesperadas.
El haber podido desarrollar un proyecto de la nada, con una idea de juego todavía sin madurar, ya que parte de la mecánica del juego se pensó durante el desarrollo, me ha enseñado que vale la pena dedicar el máximo de tiempo posible al inicio del proyecto. Ya que lo que no se define bien en un inicio, se acaba arrastrando en resto de las fases de desarrollo ralentizándolo.
Ha servido para darme cuenta de todo lo aprendido durante el curso y profundizar en temas pendientes como la utilización del ConstraintLayout y la conexión Bluetooth.

## 6.2 Objetivos cumplidos

Los objetivos que se especificaron en el inicio se han podido completar.

Se ha podido crear un juego de tablero con una mecánica de juego distinta a los que hay actualmente para la plataforma Android

La interfaz de juego, después de valorar diferentes alternativas, se ha diseñado lo más intuitiva posible. Además de incluir un reglamento para que los usuarios puedan consultarlo cuando lo necesiten.

Al desarrollarse finalmente para Android, en lugar de usar Unity, el cual permitía exportar para múltiples plataformas, se ha limitado el número de dispositivos que podrán acceder al juego. Sin embargo, dentro de Android se ha desarrollado de forma que esté disponible para la mayoría de sus usuarios. Esto se ha conseguido utilizando diseños que se adaptan fácilmente a diferentes tipos de pantalla e indicando el API 23 como el mínimo, ya que de esta forma se tiene acceso a la mayoría del mercado sin perder demasiado en eficiencia.

La IA que se ha desarrollado es muy simple, no se han utilizado heurísticas ni el algoritmo Minimax para implementarla. Sin embargo, después de probarla con distintos usuarios se ha podido ver que, a pesar de su simpleza, puede suponer un reto vencerla.

El punto clave que se quería hacer en este trabajo y ha podido completarse es la conexión mediante Bluetooth. Ha sido lo más complicado de todo el trabajo y donde se fueron la mayoría de las horas.

En cuanto al último punto de los objetivos, referente a la seguridad de la conexión por Bluetooth, se ha hecho en parte. Se ha utilizado un identificador único universal (UUID) para que sólo se puedan conectar al socket Bluetooth abierto dispositivos que lo conozcan. Sin embargo, como una medida extra de seguridad, se podrían haber cifrado los mensajes que se intercambian utilizando una clave sobre los JSON enviados.

## 6.3 Planificación u metodología

La planificación inicial fue demasiado optimista y no se incluyó suficiente tiempo para solucionar incidencias, imprevistos y mejorar la calidad del producto.
La metodología aplicada, el trabajo en cascada, ha sido eficaz por ser el proyecto con un tiempo tan limitado y realizarlo una sola persona. Ya que permitía centrarse en sólo una cosa concreta en cada fase del desarrollo.

Debido a imprevistos e incidencias se han tenido que introducir cambios en la planificación. Un ejemplo de ello son los tests que estaban planificados. La idea es que fueran unitarios pero por la cantidad de trabajo requerida y la complejidad de los tests necesarios para garantizar la calidad del producto, se optó por probar la aplicación manualmente.

## 6.4 Trabajos Futuros

Elaboración de tests unitarios que no se pudieron completar para garantizar la calidad del juego y poder detectar errores rápidamente en futuras modificaciones del código.

Se podría hacer extender el alcance del juego y permitir a usuarios jugar a través de Internet. Cosa que no se ha realizado en este trabajo por la necesidad de tener unos servidores dedicados para ello y falta de tiempo.

Con la capacidad de jugar a través de la red se podrían crear competiciones y rankings para que los jugadores compitan entre ellos y mejorar así el atractivo del juego, al poder comparar y compartir tus progresos con otros jugadores.

También la posibilidad de añadir opciones de personalización para el usuario. Permitiéndole personalizar las piezas y el nombre con el que quiere aparecer en las partidas.

La creación de una IA con heurísticas tras el estudio de un número significativo de partidas, permitiría ofrecer al jugador un enfrentamiento

individual más realista. Incluso se podrían indicar niveles de dificultad para que, en el momento de calcular el movimiento, la IA lo haga en función de ese nivel y se adapte a las habilidades del usuario.

Mejorar la interfaz de juego añadiendo efectos o incluso rediseñar el tablero en 3D, idea que se tenía al comienzo del trabajo.

Exportar el juego a otras plataformas. Exportando el juego a IOS estaría disponible prácticamente para la totalidad de dispositivos móviles actuales.

Crear una página web como Chess.com, pero propio de este nuevo juego. En esta web se podrá publicitar la aplicación y dar a conocer el juego además de permitir jugar utilizando un navegador web.

# 7. Glosario

**App**: Aplicación móvil.

**Android**: Sistema operativo móvil desarrollado por Google.

**IA** : Inteligencia Artificial.

**Bluetooth**: es un protocolo de comunicaciones que sirve para la transmisión inalámbrica de datos.

**NinjaMock**: Una plataforma online para hacer diseños de pantallas de las aplicaciones.

**Testing**: Conjunto de pruebas para determinar la calidad del código.

**Backend**: Parte oculta de la aplicación que se encarga de procesar los eventos solicitados por el usuario.

**Frontend**: Parte visible de la aplicación con la que interactúa el usuario.

# 8. Bibliografía


1. Unity Asset Store "Bluetooth Android Plugin" [en línea] [Consulta: 13 Marzo 2019] < https://assetstore.unity.com/packages/tools/network/bluetooth-android-plugin-15179 >

2. Google Play "Ajedrez. Jugar y aprender" [en línea] [Consulta: 23 Marzo 2019] < https://play.google.com/store/apps/details?id=com.chess >

3. Google Play "Lichess. Ajedrez gratis" [en línea] [Consulta: 23 Marzo 2019] < https://play.google.com/store/apps/details?id=org.lichess.mobileapp&hl=es >
4. Google Play "Chess via Bluetooth" [en línea] [Consulta: 23 Marzo 2019] < https://play.google.com/store/apps/details?id=stefan.chesstv >

5. Chess.com "Age vs Elo - Your battle against time" [en línea] [Consulta: 23 Marzo 2019] < https://www.chess.com/blog/LionChessLtd/age-vs-elo---your-battle-against-time >

6. Business of Apps "App download and usages statistics(2018)" [en línea] [Consulta: 28 Marzo 2019] < http://www.businessofapps.com/data/app-statistics/>

7. Android Developers "Linear Layout" [en línea] [Consulta: 15 Mayo 2019] < https://developer.android.com/guide/topics/ui/layout/linear?hl=es-419>

8. Android Developers "Relative Layout" [en línea] [Consulta: 15 Mayo 2019]< https://developer.android.com/guide/topics/ui/layout/relative>

9. Android Developers " Build a Responsive UI with ConstraintLayout" [en línea] [Consulta: 15 Mayo 2019] < https://developer.android.com/training/constraint-layout>

10. Android Developers " Understanding the performance benefits of ConstraintLayout" [en línea] [Consulta: 15 Mayo 2019] < https://android-developers.googleblog.com/2017/08/understanding-performance-benefits-of.html>

11. Daniel Garcia Wordpress" BLUETOOTH (III): EL ESQUEMA CLIENTE-SERVIDOR" [en línea]

[Consulta: 15 Mayo 2019] < https://danielggarcia.wordpress.com/2013/10/23/bluetooth-iii-el-esquema-cliente-servidor/>

12. Android Developers " Bluetooth" [en línea]
[Consulta: 15 Mayo 2019] < https://developer.android.com/guide/topics/connectivity/bluetooth?hl=es-419>