UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA INFORMÁTICA

SISTEMA PARA LA GESTIÓN DE PARTIDAS Y TORNEOS DE AJEDREZ

ALUMNO: ÁLVARO SÁNCHEZ RODRÍGUEZ

TUTOR: ISRAEL GONZÁLEZ CARRASCO

FECHA: JUNIO 2018

Agradecimientos

Quisiera dar las gracias a mi familia por el apoyo recibido, ya que nunca han dudado de que pudiera llegar hasta aquí. Siempre que yo comentaba que era muy difícil, su primera respuesta es que lo voy a conseguir, y que todo me va a venir muy bien como experiencia, lo cuál es cierto. La experiencia lograda ha sido lo más valioso que una persona puede obtener, y con el esfuerzo siempre se obtiene algún tipo de resultado y mucho aprendizaje.

También quiero agradecer a mi tutor, Israel, su disponibilidad para resolver las dudas que me han surgido a lo largo del proyecto. Siempre ha mostrado interés en los avances que conseguía, por lo que sólo puedo tener palabras positivas para él.

Y, por último, agradecer a todos mis amigos su interés por el proyecto, los cuáles siempre me han preguntado por los detalles, en especial aquellos con los que he compartido momentos en nuestras partidas de ajedrez por Internet.

¡¡¡Gracias a todos!!!

Abstract

The goal of this document is to show another way to do a chess software that allow users to play online chess.

There are some web sites that do this actually. We will learn about that, and purpose another alternative, which is free for all type of options, and there isn't special options for pay to use. Is totally free and easy to use, because is based on Java language.

The users don't need to install nothing, because they will download a client executable, hosted in a website, that connects directly to the server program. This is the reason why players don't need to install, because Java is installed in almost all computers. If is not the case, they can install Java easily from this page: https://java.com/en/download/

Once Java is installed, all must work, because the software doesn't use nothing special. It's a client executable that connects to the server. The server provides all resources to the players, so they can play matches between them without problems.

The reason to base all software in Java is because his oriented-object paradigm. It's efficient and easy to detect bugs and to fix them.

The proyect is not that young. For me, I thought about that 3 years ago, and I designed first steps, which include the FIDE rules, for the moves on the chess board for a single game between two players. Once that goal was completed, the rest of the things were a bit more difficult. I had to investigate about client-server theory, and the way to program it with Java.

I learned a lot about that method to program client-server based programs, and I realized that it can be use for all online game.

Resumen

El objetivo de este documento es dar a conocer una nueva forma de hacer un software de ajedrez que permita a los usuarios jugar a través de Internet.

Actualmente hay páginas web dedicadas a ello. Vamos a aprender de ellas, y proponer otra alternativa, la cual será gratuita para todas las opciones de juego. Es totalmente gratis y fácil de usar, porque está basada en el lenguaje Java.

Los usuarios no tienen que instalar nada, porque se descargarán un programa cliente ejecutable, alojado en una página web, el cual se conecta directamente con el programa servidor. Ésta es la razón por la que los usuarios no tienen que instalar, porque Java ya viene instalado en la mayoría de ordenadores. Si este no es el caso, pueden instalarlo fácilmente desde aquí:

https://www.java.com/es/download/

Una vez instalado, todo debe funcionar, porque el programa no usa nada especial para funcionar. Es un cliente ejecutable, que se conecta con el servidor, el cual provee todos los recursos necesarios a los jugadores, por lo que podrán jugar sin ningún tipo de problema, con ejecutar con doble clic en el programa cliente.

La razón por la cual se ha desarrollado en Java es gracias al paradigma de programación de orientación a objetos, el cual es eficiente y fácil de depurar para corregir los errores.

El proyecto no es tan reciente. Me resultó costoso diseñar e implementar durante cerca de 3 años un programa que refleje todas las reglas de ajedrez oficiales de la FIDE, para reflejar una partida de ajedrez en un mismo ordenador, para dos jugadores. Una vez esto se completó, se quiso ir más allá y hacer algo un poco más difícil. Para que se pudiera jugar partidas entre ordenadores remotos a través de Internet, tuve que aprender los modelos cliente-servidor y la teoría que había al respecto, sobre todo para programar en Java. Gracias a ello, he aprendido mucho sobre la sencillez de esta técnica, y me he dado cuenta de que sirve para cualquier tipo de programa o juego que funcione con este sistema cliente-servidor, para comunicar dos programas remotos a través de Internet.

ÍNDICE DE CONTENIDOS

Índice de tablas

Índice de ilustraciones

1. Introducción

El presente documento recoge todos los aspectos que han sido necesarios a la hora de desarrollar un sistema que permite a los usuarios, mediante la descarga de una aplicación en Java, alojada en una página web, jugar partidas y torneos online de ajedrez.

Vamos a comentar todas las fases de desarrollo, que son las siguientes:

- Estudio de la viabilidad del sistema.
- Análisis del sistema.
- Diseño del sistema.
- Fase de pruebas.

Además, vamos a comentar el marco regulador relativo a aspectos de la legislación a tener en cuenta a la hora de desarrollar el trabajo. También hablaremos sobre el impacto socio-económico, con el presupuesto disponible, los costes sufridos y el beneficio esperado.

1.1. Glosario de términos

Término	**Definición**
router	Dispositivo de red encargado de dirigir los mensajes (paquetes de datos) desde un dispositivo de red de origen a otro dispositivo de destino.
lag	Se dice del tiempo de retardo experimentado por un jugador de videojuegos online, generalmente. Este normalmente se refiere al producido por el tiempo de respuesta entre una acción del jugador y la respuesta del servidor.
servidor	Ordenador remoto que provee los datos solicitados por parte de otras computadoras, en un entorno de red [15].
cliente	Ordenador que requiere un servicio mediante el envío de peticiones al servidor [16].
puerto	Número que identifica un programa ejecutándose en una máquina determinada.
socket	Canal de comunicación entre dos programas de máquinas remotas.
forwarding	Método para acceder a una máquina con una IP privada, a través del router, gracias a la IP pública del router.
red	Se denomina red a Internet, habitualmente.

sistema	Conjunto de hardware y software integrados.
software	Programas creados en un lenguaje de programación que son ejecutadas gracias a un hardware concreto.
hardware	Máquinas que ejecutan aplicaciones.

1.2. Acrónimos

Acrónimo	**Definición**
TCP	Protocolo de Control de Transmisión
HTTP	Hyper Text Transfer Protocol
LAN	Local Area Network
WAN	Wide Area Network
IP	Internet Protocol
NAT	Network Address Translation
LOPD	Ley Orgánica de Protección de Datos [19] [20]
PGN	Siglas de “Portable Game Notation”. Es la notación algebraica para registrar las jugadas de una partida de ajedrez [2] [3].

1.3. Definiciones

Nombre	**Definición**
Modelo cliente-servidor	Sistema que permite la comunicación a través de diferentes usuarios de la red, por medio de un servidor que centraliza todos los recursos web [9].
Heroku	Servicio que ofrece alojamiento web de forma gratuita, bajo ciertas limitaciones de uso [6].
hosting	Proceso que permite alojar en un servidor, recursos web. Estos incluyen, páginas web, archivos alojados en la web, y ejecutar software remoto con alta disponibilidad.
TCP	Protocolo de red que permite el envío de datos de forma fiable [17].
HTTP	Protocolo de red que se encarga del envío de datos a través de la web [18].
LAN	Modo de funcionamiento de un sistema cliente-servidor de red local.
WAN	Modo de funcionamiento de un sistema cliente-servidor de red pública a nivel mundial [23].
IP	Dirección de una máquina en Internet.
NAT	Protocolo que permite comunicarse con máquinas que no tienen una IP pública, sino que pertenecen a una subred con IP's privadas [21] [26].

1.4. Objetivo

1.4.1. Objetivos generales

El objetivo general es poder permitir a los usuarios jugar partidas de ajedrez entre ellos de forma remota, a través de Internet. Una vez conseguida esta funcionalidad, servirá de base para poder albergar torneos, ya que se componen de un conjunto de partidas.

El usuario descargará una aplicación alojada en la web, para poder jugar. Esto tiene múltiples ventajas, entre ellas las siguientes:

- No depende del navegador, por lo que, si éste se estuviera actualizando, no va a detener el juego.
- La mayor parte de la complejidad computacional no la hace el lado del servidor, ya que es la parte gráfica del cliente quien hace todas las tareas. El servidor sólo hace de mediador, por lo que su carga de trabajo es simplemente procesar y enviar textos (jugadas realizadas) y es bastante rápido.
- No penaliza un jugador que tiene muy mala conectividad o conectividad lenta, ya que su tiempo para realizar jugadas se corrige, y no tiene en cuenta el "lag" de conexión.

1.4.2. Objetivos específicos

- Poder jugar partidas en un mismo ordenador.
- Poder jugar partidas entre ordenadores remotos de una misma subred (LAN). Esta funcionalidad no requiere que el lado del servidor tenga un puerto abierto, aunque de ser así, hay que utilizar direcciones IP's privadas.
- Poder jugar partidas entre ordenadores de subredes diferentes (WAN). Esta funcionalidad sirve tanto para LAN como para WAN, ya que al usar la dirección IP pública del lado del servidor a la que el jugador (cliente) se conecta, ésta es fija, no cambia y es única. Para ello, como hemos mencionado, el programa del lado del servidor, funcionará en un puerto concreto, elegido acorde a los parámetros de seguridad recomendados [11], ya que hay puertos que no deben abrirse por ser de uso de programas maliciosos. Aunque por lo visto, el puerto 80 que es el puerto web es usado por esos programas y siempre viene abierto por defecto, con lo que nos plantea serias dudas de lo que realmente es considerado seguro.
- Poder jugar partidas sin necesidad de usar un ordenador que haga de servidor, y sustituyendo el protocolo TCP por HTTP, usando Heroku como servidor (objetivo aún por desarrollar).

1.5. Motivación

Este es un proyecto que permitirá a los jugadores de ajedrez, poder utilizar una nueva plataforma de juego online. Es el inicio de un proyecto que va a ir creciendo a medida que se van a ir incluyendo nuevas funcionalidades. Creemos que, para el tiempo de desarrollo de un Trabajo de Fin de Grado, hemos cubierto las funcionalidades más básicas, que consisten en permitir a los usuarios jugar partidas online. Es el inicio de algo todavía mayor, como puede ser jugar torneos serios, poder obtener toda la información en la web de los torneos, premios, etc.

Además, es un proyecto que surgió como afición a la programación, en particular la programación en lenguaje Java. Y yo como jugador de ajedrez, siempre he tenido curiosidad de saber más acerca de la programación de una partida, así que, en un momento dado, me puse manos a la obra, y decidí investigar al respecto. Al principio, los primeros diseños no eran para nada exitosos. Es cierto que, el hecho de saber cómo permitir el movimiento de una determinada pieza en el tablero, no era para nada difícil. Pero la cosa se complicaba cuando había que contemplar posibilidades complejas, tales como son el hecho de no permitir saltos de una pieza sobre otra, el cálculo del jaque mate, saber cuándo el rey está en jaque, y evitar permitir una jugada que ponga al rey en jaque, entre otras. Además, la regla del enroque, y la captura al paso, son dos de las reglas más complicadas de contemplar, puesto que se componen de particularidades más simples, y hay que saber integrarlas en su conjunto.

Otras reglas, como las de las tablas por triple repetición de una posición en el tablero, también tienen peculiaridades que no son tan evidentes de programar, por lo que el tiempo dedicado a ello ha sido de unos 2 a 3 años en poder contemplar todos los casos posibles.

Así pues, es el punto de partida del proyecto. Una vez conseguido que se pueda reflejar el hecho de jugar una partida en un mismo ordenador por dos personas, se decidió ir más allá, y adaptarlo para poder disputar partidas online, lo cual vamos a ver más en detalle a lo largo de este documento.

También es interesante comentar, que el software se ha desarrollado en Java con el entorno de desarrollo eclipse [8].

1.6. Contexto

Actualmente ya existen aplicaciones similares. Sin embargo, muchas de ellas presentan limitaciones; entre ellas, que hay opciones de pago, y no permiten una experiencia agradable para el usuario. Otras de ellas requieren por el simple hecho de usarlas, pagos mensuales. Y las que son gratuitas, presentan una serie de limitaciones, o errores de programación, de los cuales muchos usuarios se quejan habitualmente. Es por ello que presentamos una nueva alternativa, la cual creemos que es una buena iniciativa, puesto que el hecho de jugar partidas de ajedrez por Internet, no debería ser ningún problema. El ajedrez es un juego mental, y en realidad, no debería haber una diferencia importante para poder permitir simplemente jugar y disfrutar entre amigos de una partida online.

2. Planteamiento del problema

Queremos desarrollar un software (sistema), que permita a los usuarios poder competir al ajedrez online, de forma gratuita y sencilla, con la mejor experiencia posible. Hemos visto que hay diferentes plataformas existentes. Sin embargo, algunas de ellas (por no decir, todas) presentan una serie de limitaciones o inconvenientes que, para la mayoría de los usuarios, limita el aprendizaje, o requiere para el uso de determinados servicios, cuentas tipo premium o de pago. Además, muchos usuarios reportan bugs a la hora de jugar, justificados cabe destacar, es decir, conocen perfectamente las reglas del juego, y estos servicios, no las respetan en todos los escenarios y casos posibles.

Nuestro sistema no funcionará directamente en la web, sino que, para poder jugar, el usuario se descargará un archivo ejecutable, y sin necesidad de instalación, ya podrá utilizar para jugar.

Esto tiene como ventaja que, al funcionar en una aplicación, aunque tendrá que acceder a la web para poder descargar el ejecutable, evitará algunos problemas, como puede ser, que su conectividad penalice su experiencia de usuario. Ya que al jugar una partida directamente en la propia página web, no se puede controlar el retraso producido entre que se envía la jugada y llega al servidor, por lo que muchas veces esto penaliza su tiempo restante durante la partida.

Nuestro sistema evitará esto, ya que aparte de la jugada realizada, se enviará al servidor el tiempo restante, por lo que se corrige in situ en el servidor. Esto es debido a que es necesario enviar el tiempo restante si se quiere corregir el “lag”. De otra manera, el tiempo de cola (tiempo en el que los paquetes permanecen en los routers hasta que son procesados) es impredecible, por mucho que se quiera tratar de corregir o compensar. Nuestro software será innovador en ese aspecto.

Cabe destacar que, para poder realizar toda la labor, se ha desarrollado previamente en Java una aplicación que permitía en un mismo ordenador, poder jugar entre dos personas, con todas las reglas oficiales de ajedrez [1]. Sin embargo, se quiso dar un paso más, añadiendo la posibilidad de reutilizar este programa para objetivos mayores, y uno de ellos es crear un sistema capaz de albergar partidas de ajedrez online tomando como base el código desarrollado durante 2-3 años, incluyendo todas las reglas de ajedrez, y la parte gráfica para que, mediante el ratón se puedan realizar las jugadas. También se incluyó la funcionalidad de poder descargar la partida en formato PGN. De esta manera, otras aplicaciones de ajedrez pueden interpretar la partida, puesto que es un formato estándar que guarda la información de una partida en notación algebraica [2].

Sin embargo, adaptar esto para poder albergar, primero una partida de un jugador a otro, con ordenadores remotos usando para ello Internet, y posteriormente permitir jugar torneos online, trajo una serie de implicaciones y dificultades.

Lo primero, había que pensar en qué cambios había que incluir para poder conseguir estas funcionalidades. Una vez conseguido, surgió un problema adicional. Dicho problema radicaba en que no había ningún problema cuando, una vez adaptado el

código para conectar dos programas remotos para representar una partida online, estos estaban en dos ordenadores dentro de una misma subred (mismo router). Sin embargo, entre ordenadores de subredes diferentes, no se podían comunicar, puesto que los routers que funcionan mediante NAT, necesitan conocer la IP de la máquina a la que conectarse, con lo cual el protocolo NAT no nos daba esa opción, puesto que la IP pública era la del router y no la de la máquina concreta.

Por ello, surgió la solución de utilizar un sistema cliente-servidor. Un ordenador hará de servidor, y el router asociado al servidor, necesita un puerto abierto, para que los jugadores (clientes) puedan comunicarse entre sí gracias a este servidor, que hará de mediador. Como los clientes conectan a esta dirección IP del servidor, y es accesible mediante dicho puerto, los jugadores no tienen que instalar ni configurar nada para poder usar la aplicación. Hemos utilizado el protocolo TCP, puesto que, aunque se han hecho pruebas con mensajes HTTP utilizando Heroku como servidor, trae una serie de limitaciones, por lo que, en lugar de utilizarlo en primer lugar, decidimos realizar este sistema mediante el esquema antes mencionado. Así, nos podemos extraer de la parte del servidor, y centrarnos en la funcionalidad. De todas maneras, una vez terminada una partida concreta, es factible que automáticamente se suba dicha partida al servidor antes mencionado mediante el protocolo HTTP [5].

De esta manera, mediante el sistema desarrollado los jugadores se registrarán y obtendrán un perfil de usuario, con el que será identificado en la aplicación, para poder tener un seguimiento de partidas, puntuación, etc.

Por lo tanto, se ha utilizado Java como herramienta para crear el entorno de juego del usuario, y otras tecnologías para poder disponer de una página web que aloje dicho programa para descargar.

Además, la propia web permitirá en un futuro almacenar información acerca de las partidas, perfiles de usuario y los torneos jugados, sin necesidad de iniciar sesión para poder verlo.

Por lo tanto, sin duda la parte central y más compleja, pese a lo explicado anteriormente, fue el trabajo previo que duró aproximadamente 3 años en diseñar la programación del reglamento de ajedrez. Algunos aspectos, sin embargo, aprovechaban la gran capacidad de los ordenadores de resolver problemas de búsqueda; por ejemplo, hallar si es jaque mate una posición, y aunque era aparentemente difícil de plantear, esto resolvía muchos problemas, la capacidad de simular las diferentes jugadas posibles en copias del propio tablero en memoria. Una vez conseguido esto, las adaptaciones, pese a requerir esfuerzo, son muy factibles por el hecho de que gran parte del trabajo lo realizan los routers y se confía en que enviarán mediante el protocolo que elijamos, los mensajes a sus destinos, que almacenarán la información relacionada con las jugadas que se envían.

3. Estudio de la Viabilidad del Sistema

3.1. Establecimiento del alcance del sistema

En este apartado vamos a comentar a quiénes va dirigido el sistema, y todos los usuarios implicados. Con el fin de poder estudiar los requisitos necesarios para llevar a cabo nuestro software.

3.1.1. Estudio de la solicitud

Vamos a analizar un estudio de las necesidades que los clientes esperan cubrir. De esta manera, podemos predecir la viabilidad del desarrollo de nuestro proyecto, ya que necesitamos garantizar que disponemos del presupuesto, equipo de trabajo, personal y tiempo necesarios para cumplir todos los objetivos del cliente. Además, tenemos que conocer cómo de variable puede llegar a ser la necesidad de estos clientes. Si es un software que no suele cambiar con el paso del tiempo, el proyecto implica un riesgo menor, que si suele incluir actualizaciones.

Una vez realizados estos pasos, procederemos a identificar aquellos requisitos que el usuario demanda.

3.1.2. Alcance del sistema

Este sistema permitirá jugar al ajedrez online. Para ello, el usuario requerirá de un entorno de juego. Este puede ser un entorno web o un entorno gráfico. Nosotros proporcionamos un entorno web para descargar la aplicación, que contiene a su vez un entorno gráfico, el cuál funcionará en su máquina, como si de una aplicación se tratase. El usuario requiere para ello, conectividad a Internet.

3.1.3. Identificación de los usuarios

Vamos a explicar todos los interesados (stakeholders) en nuestra aplicación.

- Álvaro Sánchez será el responsable de llevar a cabo, el desarrollo del proyecto, y será el máximo responsable.
- Israel González es el cliente principal, el cuál va a utilizar el sistema. Para ello, se entrevistará periódicamente con Álvaro, para acordar las características que debe cumplir el sistema de torneos de ajedrez.
- Otros usuarios que deseen utilizar el software, ya que es público y de carácter gratuito.

3.2. Estado del arte

Para el desarrollo de nuestro sistema, hemos tenido en cuenta las siguientes aplicaciones existentes:

- lichess.org [12]
- chess.com [13]
- ICC (Internet Chess Club) [14]

Estas son las aplicaciones que utilizan el esquema más similar, puesto que sirven para jugar y albergar torneos de ajedrez online. En particular, ICC utiliza un sistema cliente

servidor, muy parecido al nuestro, puesto que usa una web para poder albergar la aplicación cliente, que descargándola se podrá jugar. Las dos primeras usan la propia web para poder jugar directamente en ella. Más adelante explicaremos la decisión tomada en el apartado de justificación de la misma.

Ahora vamos a mostrar una tabla con las características que proporcionan estas aplicaciones:

NOMBRE	Gratis	Requiere descarga	Prestaciones (A=Altas, M=medias, B=bajas)	P=Para profesionales principalmente M=Para jugadores medios/aficionados y profesionales
lichess.org	SI	NO	M	M
chess.com	SI	NO	M	M
ICC	NO	SI	A	P

Tabla 1: Comparativa de aplicaciones similares

Podemos ver como ICC tiene como público objetivo, jugadores profesionales de ajedrez. Esto es debido a que jugar online trae consigo una serie de implicaciones, entre ellas, que hay personas que suelen hacer trampa. Esto es debido a que utilizan software para copiar jugadas durante la partida, por lo que, aunque sean analizadas sus partidas y se detecten dichas trampas, estos permanecen en el anonimato. Sin embargo, ICC exige registrar los datos personales puesto que, en caso de trampa, la persona se responsabiliza mucho más. Además, ICC evita muchos problemas relativos a la conectividad. Por mencionar los más importantes, tenemos que, en ajedrez, el tiempo es un factor, por lo que una mala conexión puede hacer que la jugada se envíe con retraso y se penalice al jugador con menos tiempo, indirectamente. La ventaja de un programa descargado, de escritorio pero que mantiene conectividad a Internet, es que no hay problemas en enviar la jugada y el tiempo restante, para que sea éste corregido a tiempo real. De esta manera, nadie va a notar dicho retardo de conexión.

Además, podemos ver que ICC no es gratuito, pese a que utiliza un esquema similar al nuestro. En este caso, la intención es que tenga las mismas prestaciones, y además sea un sistema gratuito para los usuarios, lo cual es novedoso.

Vamos a ver un ejemplo de cada una de estos sistemas:

- lichess.org:

Ilustración 1: lichess.org

Esta es la interfaz de una partida en lichess.org. Aunque, además, se pueden editar algunas características, como el set de piezas, el tamaño, ofrecer tiempo de compensación al rival, etc. Podemos ver que se está jugando una partida a 3 minutos por jugador. Los relojes aparecen en la parte derecha, y cuando el jugador realiza su jugada, para su reloj y pone en marcha el reloj del adversario. Sin embargo, a pesar de que en las últimas actualizaciones lo han sabido corregir en parte, muchas personas se quejan en ocasiones del "lag", es decir, cuando por algún motivo el servidor se satura, o la conexión del jugador es algo pobre, notan cómo al realizar la jugada, ésta tarda en quedar reflejada, por lo que sigue corriendo el tiempo. La gran ventaja, es que es un sitio totalmente gratuito, y no hay ninguna opción extra que deje de serlo. Por lo que, además, gusta tanto a aficionados como a profesionales. En general, esta web tiene una fama, que ha progresado en los últimos meses debido a que el mejor jugador del mundo de ajedrez, Magnus Carlsen, ha ultilizado diferentes cuentas de usuario de manera "sorpresa", entre ellas las siguientes:

https://lichess.org/@/MagnusCarlsen

https://lichess.org/@/DannyTheDonkey

https://lichess.org/@/DrDrunkenstein

Además, tiene como característica que las partidas pueden ser analizadas por un software en busca de errores durante la partida, y cómo solventarlos, lo cual ayuda en gran medida a jugar un ajedrez más sofisticado.

- Chess.com
  Aquí ya hay a menudo jugadores más profesionales, por lo que genera menos misterio que la web anterior. La apariencia de una partida es la siguiente:

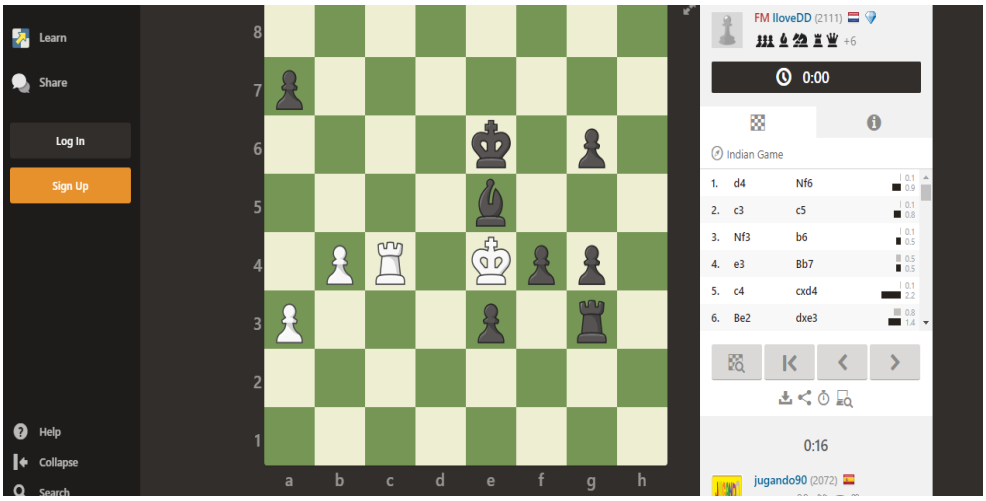


Ilustración 2: chess.com

Los jugadores profesionales más habituales en esta página son, entre otros, Hikaru Nakamura y Daniel Rensch.

- ICC
  Tenemos una primera página web donde se presenta el sistema, y se da la opción de descargar:

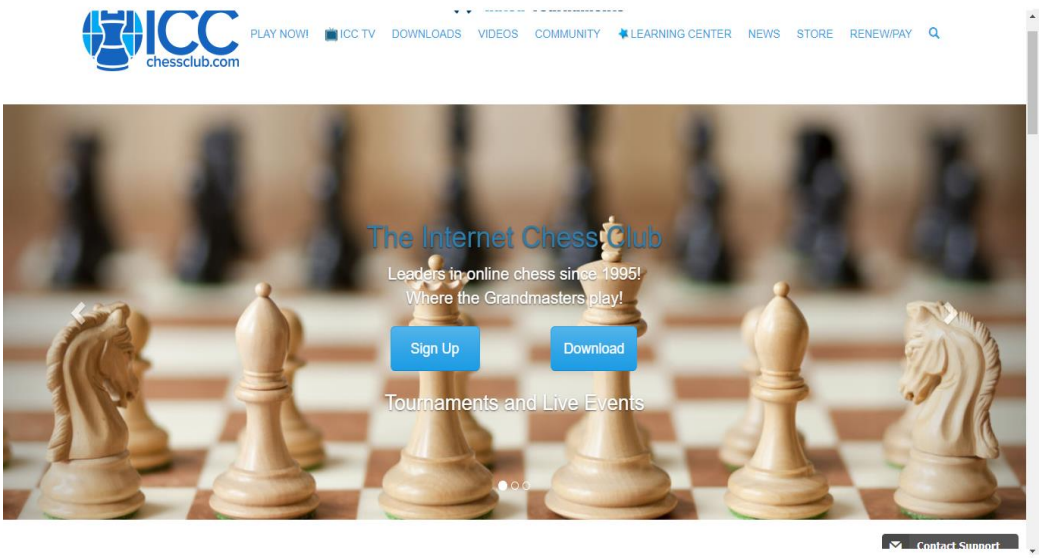


Ilustración 3: ICC (I)

Y una vez hacemos clic en “Download” aparece lo siguiente:

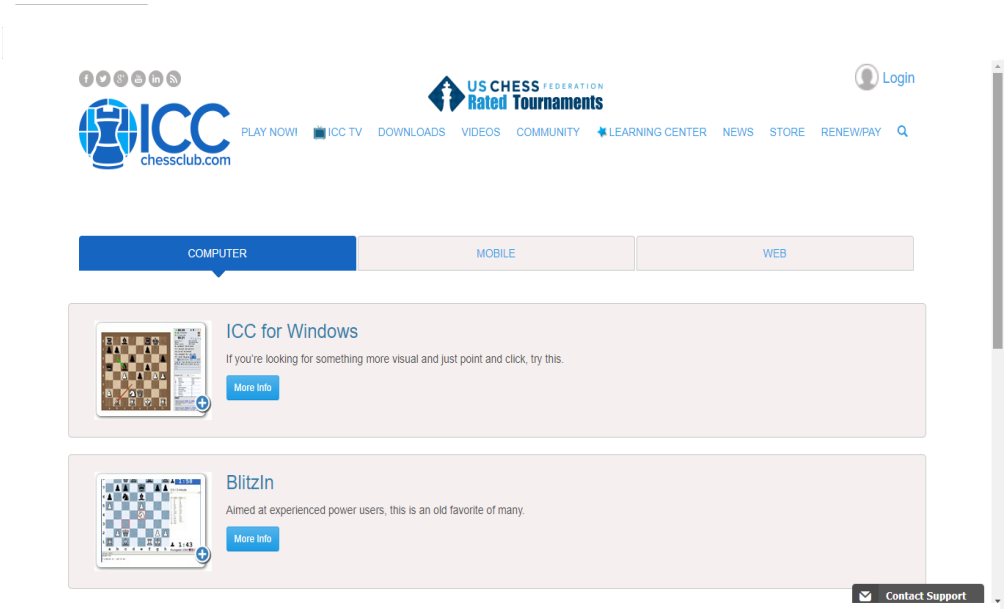


Ilustración 4: ICC (II)

Esta última es la página en la cual se basa nuestra aplicación. Es decir, las partidas funcionan gracias a un archivo ejecutable, aunque utilice recursos web.

3.3. Definición de los requisitos de usuario

Estos son los requisitos de usuario más importantes. Se describe un identificador de requisito, su prioridad (A=alta, M=media, B=baja), el título del requisito, los requisitos de los que depende y una descripción.

El formato de los requisitos de usuario es el siguiente:

ID: RU-XXX donde XXX es el número identificador del requisito.

Prioridad: alta, para requisitos esenciales. El resto son de prioridad media o baja.

Título: descripción breve del requisito.

Dependencias: requisitos de los cuales depende.

Descripción: describe de forma completa el requisito.

ID	RU-001
PRIORIDAD	A
TITULO	Jugar partidas de ajedrez online
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario iniciar una partida online con otros jugadores

Tabla 2: RU-001

ID	RU-002
PRIORIDAD	A
TITULO	Registrarse en el sistema
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario registrarse en la aplicación para poder tener un seguimiento de sus partidas jugadas y su puntuación

Tabla 3: RU-002

ID	RU-003
PRIORIDAD	A
TITULO	Tener una web de referencia para descargar la aplicación
DEPENDENCIAS	-
DESCRIPCION	El sistema tendrá una página web para descargar el software que permita al usuario jugar partidas y torneos de ajedrez online

Tabla 4: RU-003

ID	RU-004
PRIORIDAD	A
TITULO	Jugar partidas de ajedrez online con todas las reglas de la FIDE aplicadas
DEPENDENCIAS	-
DESCRIPCION	El programa tiene implementadas todas las reglas de la FIDE para que el jugador no tenga problemas a la hora de jugar.

Tabla 5: RU-004

ID	RU-005
PRIORIDAD	A
TITULO	Poder elegir el ritmo de juego deseado al jugar una partida de ajedrez
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario elegir el tiempo de juego deseado en una partida de ajedrez

Tabla 6: RU-005

ID	RU-006
PRIORIDAD	A
TITULO	Poder descargar y usar gratis la aplicación
DEPENDENCIAS	-
DESCRIPCION	El programa permite descargar gratis la aplicación para los usuarios.

Tabla 7: RU-006

ID	RU-007
PRIORIDAD	B
TITULO	Permitir chat durante la partida contra un jugador
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario hablar con su contrincante durante la partida.

Tabla 8: RU-007

ID	RU-008
PRIORIDAD	B
TITULO	Permitir elegir el estilo de piezas y tablero
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario elegir el estilo de piezas y tablero.

Tabla 9: RU-008

ID	RU-009
PRIORIDAD	A
TITULO	Permitir jugar torneos
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario jugar torneos de ajedrez

Tabla 10: RU-009

3.4. Estudio de alternativas de solución

Se proponen diferentes alternativas de solución. Una de ellas es la siguiente:

- Página web para descargar la aplicación cliente.jar. Utilizaremos como servidor de páginas web Heroku, que ofrece hosting gratuito, aunque con ciertas limitaciones, que no nos afectan para nuestro proyecto.
- cliente.jar utilizará librerías Java para el funcionamiento a través de Internet, mediante el uso de Sockets.
- Base de datos MySQL para almacenar la información de los usuarios de la aplicación.
- Utilizamos Java como lenguaje de programación. Esto es debido a que es un lenguaje multiplataforma y orientado a objetos, ideal para facilitar la programación. También utilizaremos Eclipse como entorno de desarrollo, ya que facilita mucho la depuración del software, al hacer el proceso de compilación a tiempo real. Esto ayuda a detectar en fases tempranas los errores de sintaxis. Además, ofrece sugerencias de cómo se pueden solucionar estos errores.
- Utilizaremos en principio un servidor dedicado para la aplicación que funciona en el lado del servidor, que gestiona los perfiles y las partidas de los usuarios, para evitar que la aplicación servidora necesite utilizar otros tipos de lenguaje diferentes a Java. Esto es debido a que, principalmente, ningún servicio gratuito ofrece alojar un archivo .jar y ejecutar remotamente. Con Amazon esto sí es posible, pero se ha preferido evitar esta opción, por la necesidad de tener que ceder datos a terceros, y porque para conseguir la funcionalidad y realizar las pruebas, no se necesita que realmente esté ofreciendo servicio las 24 h del día. Aunque en un futuro se considerará migrar a una solución de la aplicación del lado de servidor mediante node.js. En las primeras fases de desarrollo no se contempló esta posibilidad, aunque más adelante comprendimos la manera de conseguir la misma funcionalidad, utilizando un servidor gratuito como Heroku, que alojase una aplicación node.js. Sin embargo, no se han hecho demasiadas pruebas al respecto, por lo que es una buena opción de futuro. Esto es debido a que hay una brecha de aprendizaje a la hora de utilizar tanto sockets TCP como sockets HTTP en Java, y comprender del todo el envío y recepción de mensajes en ambos protocolos de red.

  Ahora vamos a comentar cómo han sido resueltas las aplicaciones mencionadas en el apartado del estado del arte, para tener una idea de los recursos que han necesitado para desarrollarlas.

  -lichess.org: Ha sido desarrollado en lenguaje de programación Scala [22]. Como base de datos utiliza MongoDB, ya que tiene la ventaja, de que manejar gran cantidad de datos en una base no relacional, es muy ventajoso y de veloz acceso. Tiene por contra, la dificultad en el aprendizaje de las técnicas de programación de este lenguaje, al haber menor información.
  También utiliza web sockets mediante nginx 1.9.

-chess.com. Aquí no se deja claro el lenguaje de programación utilizado ni las tecnologías de base de datos aplicadas. Probablemente han preferido dejarlo en el anonimato.

-ICC. Tampoco especifica los recursos utilizados. Probablemente, al ser software que no es libre, al contrario que en lichess.org que si es de software libre.

3.5. Valoración de las alternativas

A continuación, vamos a mostrar tablas comparativas para cada recurso (software/hardware) necesarios para el funcionamiento de nuestro sistema.

Lenguaje de programación

Nombre del lenguaje	Multiplataforma (SI/NO)	Paradigma de programación
Java	SI	Orientado a objetos
Scala	NO	Multi-paradigma/ orientado a objetos
C	NO	Imperativo/Procedural
C++	SI	Orientado a objetos

Tabla 11: Valoración de los lenguajes de programación

Sin embargo, al parecer para que C++ sea multiplataforma, necesita ciertos complementos, mientras que Java es puramente multiplataforma, es decir, válido y compatible para cualquier sistema operativo.

Alojamiento web

Vamos a ver diferentes tipos de servidores para alojar nuestra página web de referencia.

Nombre	Gratuito/De pago	Limitaciones
Heroku	Gratuito	Escasas
000webhost	Gratuito	Limitación de disponibilidad

Tabla 12: Valoración de los servidores web

3.6. Selección de la solución

Por lo tanto, gracias a lo analizado en el punto 3.5. nuestra selección será la siguiente:

- Servidor web: Heroku.
- Lenguaje de programación para la aplicación cliente para jugar: Java.
- MySQL como base de datos.

La aplicación servidora, también usará Java y se ejecutará en un servidor dedicado. En futuras mejoras, se optará por alojarlo en Heroku mediante node.js y así evitar depender de un servidor dedicado.

4. Análisis

En esta fase de desarrollo, se van a analizar en profundidad todos los requisitos que deben funcionar en la aplicación. En la fase anterior, se detectaron todos los requisitos que en general deben ser aplicados.

4.1. Casos de uso

Ahora vamos a analizar los casos de uso necesarios para obtener dichos requisitos funcionales. Los casos de uso tienen los siguientes elementos:

ID: es el identificador del caso de uso. En este caso utilizamos las siglas CU y a continuación un número que los identifica unívocamente.

Título: describe de forma resumida el objetivo del caso de uso.

Actores: son los elementos que interactúan con el sistema. Tenemos dos tipos principales. Los usuarios registrados, y los usuarios invitados. Los usuarios registrados podrán tener un perfil de sus partidas jugadas, mientras que los usuarios invitados, no tendrán esta característica. Por lo tanto, para las características básicas, se puede hacer uso del rol del usuario invitado. Mientras que para aquellas que impliquen el registro de partidas, se hará uso del rol del usuario registrado.

Objetivos: cubren las necesidades de los actores.

Precondición: aquello que se tiene que cumplir previamente para poder alcanzar el caso de uso.

Postcondición: es la salida generada por el caso de uso.

Escenario principal: aquel que se prevé en la mayor parte de los casos, principalmente cuando todo funciona correctamente.

Escenario alternativo: aquel que puede ocurrir con menor frecuencia, habitualmente asociado a casos de error.

ID	CU-001
TÍTULO	Dos jugadores pueden jugar una partida en el mismo ordenador
ACTORES	Usuario invitado A, Usuario invitado B
OBJETIVOS	Los usuarios A y B quieren jugar una partida de ajedrez en el mismo ordenador, para dos jugadores.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción “2 jugadores” 3. Elegir el ritmo de juego.
POSTCONDICIÓN	1. Se inicia la partida con el ritmo de juego elegido. 2. Ambos jugadores pueden mover sus piezas en su turno.
ESCENARIO PRINCIPAL	
ESCENARIO ALTERNATIVO	-

Tabla 13: CU-001

ID	CU-002
TÍTULO	Dos jugadores pueden jugar una partida, cada uno en su propio ordenador mediante LAN.
ACTORES	Usuario invitado A, Usuario invitado B
OBJETIVOS	Los usuarios A y B quieren jugar una partida de ajedrez en ordenadores remotos, mediante la conexión a Internet que les proporciona el mismo router (red local).
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción "Jugar como Invitado" 3. Elegir el ritmo de juego por parte de uno de los usuarios (A). 4. El otro usuario (B) se une a la partida.
POSTCONDICIÓN	1. Se inicia la partida con el ritmo de juego elegido. 2. Ambos jugadores pueden mover sus piezas en su turno.
ESCENARIO PRINCIPAL	La partida se inicia sin problemas y los dos jugadores tienen conectividad.
ESCENARIO ALTERNATIVO	1. Uno o ambos jugadores pierde conectividad, con lo cual, el que pierde la conexión, pierde la partida. 2. El servidor no funciona correctamente.

Tabla 14: CU-002

ID	CU-003
TÍTULO	Dos jugadores pueden jugar una partida, cada uno en su propio ordenador mediante WAN.
ACTORES	Usuario invitado A, Usuario invitado B
OBJETIVOS	Los usuarios A y B quieren jugar una partida de ajedrez en ordenadores remotos, mediante la conexión a Internet que les proporciona a cada uno en redes distintas (distancia de más de un router, diferente localización geográfica).
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción "Jugar como Invitado" 3. Elegir el ritmo de juego por parte de uno de los usuarios (A). 4. El otro usuario (B) se une a la partida.
POSTCONDICIÓN	1. Se inicia la partida con el ritmo de juego elegido. 2. Ambos jugadores pueden mover sus piezas en su turno.
ESCENARIO PRINCIPAL	La partida se inicia sin problemas y los dos jugadores tienen conectividad.
ESCENARIO ALTERNATIVO	1. Uno o ambos jugadores pierden conectividad. El que pierda conectividad, pierde la partida. 2. El servidor no funciona correctamente, el puerto asociado al programa servidor está deshabilitado.

Tabla 15: CU-003

ID	CU-004
TÍTULO	Un usuario invitado puede registrarse en el sistema
ACTORES	Usuario invitado A
OBJETIVOS	El usuario A quiere registrarse en el sistema.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción “Soy Nuevo” 3. Elegir el nombre de usuario. 4. Elegir la contraseña
POSTCONDICIÓN	1. Se registra su nombre de usuario. 2. Se registra su contraseña
ESCENARIO PRINCIPAL	Se añade al sistema el nuevo usuario, y éste puede iniciar sesión sin problemas
ESCENARIO ALTERNATIVO	1. El nombre de usuario no es válido. Se solicita al usuario que elija otro nombre. 2. La contraseña no es válida. Se solicita al usuario que escoja otra contraseña.

Tabla 16: CU-004

ID	CU-005
TÍTULO	Un usuario registrado puede iniciar sesión
ACTORES	Usuario registrado A
OBJETIVOS	El usuario A quiere iniciar sesión.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción “Iniciar Sesión”
POSTCONDICIÓN	1. Se puede comprobar que el usuario aparece en el sistema con su nombre de usuario, que aparece en el menú de la sala de juego.
ESCENARIO PRINCIPAL	Se inicia sesión con éxito, y el usuario puede acceder al menú de opciones para usuarios registrados.
ESCENARIO ALTERNATIVO	1. El nombre de usuario no es válido. 2. La contraseña no es válida. 3. La combinación de nombre de usuario y contraseña no son válidas

Tabla 17: CU-005

ID	CU-006
TÍTULO	Un usuario registrado puede ver su perfil
ACTORES	Usuario registrado A
OBJETIVOS	El usuario A quiere ver su perfil de usuario.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción “Iniciar Sesión” 3. Elegir la opción “Ver mi perfil”
POSTCONDICIÓN	1. Se puede comprobar que el usuario aparece en el sistema con su nombre de usuario, que aparece en el menú de la sala de juego y puede acceder a su perfil, donde aparece su registro de partidas jugadas y su puntuación.
ESCENARIO PRINCIPAL	Se inicia sesión con éxito, el usuario puede acceder al menú de opciones para usuarios registrados y puede entrar a su perfil.
ESCENARIO ALTERNATIVO	1. Error al iniciar sesión. 2. El servidor no funciona correctamente.

Tabla 18: CU-006

ID	CU-007
TÍTULO	Los usuarios pueden mover las piezas durante la partida según el reglamento FIDE
ACTORES	Usuario invitado A, Usuario invitado B
OBJETIVOS	Los usuarios quieren poder mover sus piezas durante la partida que están jugando.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción "Jugar como invitado" 3. El usuario A inicia una partida en la sala de juego. 4. El usuario B se une a esa partida.
POSTCONDICIÓN	1. Se comprueba que ambos jugadores pueden mover sus piezas en su turno, siempre que respeten las reglas de la FIDE.
ESCENARIO PRINCIPAL	Se inicia la partida con éxito y la jugabilidad es buena.
ESCENARIO ALTERNATIVO	1. Error de conexión en alguno de los jugadores. 2. Error del servidor.

Tabla 19: CU-007

ID	CU-008
TÍTULO	Los usuarios pueden inscribirse a un torneo en la sala de juego
ACTORES	Usuario invitado A
OBJETIVOS	Un usuario quiere poder inscribirse en un torneo.
PRECONDICIÓN	1. Ejecutar el programa 2. Elegir la opción "Jugar como Invitado" 3. En la sala de juego elegir la opción "Torneos" 4. Inscribirse en un torneo de la lista de torneos.
POSTCONDICIÓN	1. Se comprueba que el jugador aparece en la lista de participantes de ese torneo.
ESCENARIO PRINCIPAL	El usuario se puede inscribir correctamente
ESCENARIO ALTERNATIVO	1. Error del servidor

Tabla 20: CU-008

4.2. Requisitos del sistema

Se hará una clasificación por tipos de requisito. Estos son requisitos funcionales, de restricción, y requisitos no-funcionales.

Además, se detallan los requisitos en el siguiente formato:

ID: es el identificador del requisito. Los requisitos funcionales, comienzan con las siglas RF, seguidos de un guión y un número consecutivo. Los requisitos de restricción aparecerán con las siglas RES, seguidos también de un guión, y un número consecutivo para esos requisitos de restricción. Y los requisitos no funcionales, aparecerán con sus siglas RNF, seguidas de un guión y también otro número secuencial.

Prioridad: A, para requisitos con prioridad alta; M, para requisitos con prioridad media; y B, para requisitos con prioridad baja.

Requisitos funcionales

ID	RF-001
PRIORIDAD	A
TITULO	Ajedrez 2 jugadores mismo ordenador
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario iniciar una partida en el mismo ordenador para dos jugadores

Tabla 21: RF-001

ID	RF-002
PRIORIDAD	A
TITULO	Ajedrez varios jugadores online
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario iniciar una partida online con otros jugadores

Tabla 22: RF-002

ID	RF-003
PRIORIDAD	A
TITULO	Ajedrez varios jugadores online en la misma subred
DEPENDENCIAS	RF-002
DESCRIPCION	El programa permitirá al usuario iniciar una partida online con otros jugadores de una misma subred, en un mismo router (LAN).

Tabla 23: RF-003

ID	RF-004
PRIORIDAD	A
TITULO	Ajedrez varios jugadores online en cualquier subred
DEPENDENCIAS	RF-002
DESCRIPCION	El programa permitirá al usuario iniciar una partida online con otros jugadores en cualquier subred o un número de routers de distancia ilimitados (WAN).

Tabla 24: RF-004

ID	RF-005
PRIORIDAD	A
TITULO	Registro de usuario
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario registrarse en el sistema

Tabla 25: RF-005

ID	RF-006
PRIORIDAD	A
TITULO	Inicio de sesión
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario iniciar una sesión en el sistema

Tabla 26: RF-006

ID	RF-007
PRIORIDAD	A
TITULO	Reglamento FIDE
DEPENDENCIAS	-
DESCRIPCION	El programa de ajedrez utilizará las reglas FIDE para validar las jugadas realizadas

Tabla 27: RF-007

ID	RF-008
PRIORIDAD	A
TITULO	Elegir ritmos de juego
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario iniciar una partida a un ritmo de juego determinado

Tabla 28: RF-008

ID	RF-009
PRIORIDAD	A
TITULO	Jugar torneo en una sala de juego
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario inscribirse en una sala de un torneo determinado

Tabla 29: RF-009

ID	RF-010
PRIORIDAD	B
TITULO	Elegir estilo de piezas
DEPENDENCIAS	-
DESCRIPCION	El usuario podrá elegir el estilo de piezas

Tabla 30: RF-010

ID	RF-011
PRIORIDAD	B
TITULO	Elegir estilo de tablero
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario elegir el estilo del tablero

Tabla 31: RF-011

ID	RF-012
PRIORIDAD	M
TITULO	Opciones durante el juego
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario elegir determinadas opciones durante el juego, aparte de mover las piezas. Estas incluyen: rendirse, ofrecer tablas, incrementar el reloj del rival, iniciar conversaciones, saludar, abandonar la ventana de juego, cerrar sesión

Tabla 32: RF-012

ID	RF-013
PRIORIDAD	A
TITULO	Mover piezas durante el juego
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al usuario poder mover las piezas con el ratón, pulsando sobre ella y arrastrándola hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 33: RF-013

ID	RF-014
PRIORIDAD	A
TITULO	Mover alfil durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover el alfil con el ratón, pulsando sobre él y arrastrándolo hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 34: RF-014

ID	RF-015
PRIORIDAD	A
TITULO	Mover caballo durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover el caballo con el ratón, pulsando sobre él y arrastrándolo hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 35: RF-015

ID	RF-016
PRIORIDAD	A
TITULO	Mover dama durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover la dama con el ratón, pulsando sobre esta y arrastrándola hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 36: RF-016

ID	RF-017
PRIORIDAD	A
TITULO	Mover peón durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover el peón con el ratón, pulsando sobre él y arrastrándolo hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 37: RF-017

ID	RF-018
PRIORIDAD	A
TITULO	Mover rey durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover el rey con el ratón, pulsando sobre él y arrastrándolo hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 38: RF-018

ID	RF-019
PRIORIDAD	A
TITULO	Mover torre durante el juego
DEPENDENCIAS	RF-013
DESCRIPCION	El programa permitirá al usuario poder mover la torre con el ratón, pulsando sobre él y arrastrándolo hacia la casilla destino. Al soltar el ratón, se validará la legalidad de la jugada

Tabla 39: RF-019

ID	RF-020
PRIORIDAD	A
TITULO	Comprobación de posición de jaque mate
DEPENDENCIAS	RF-007
DESCRIPCION	El programa calculará si la posición actual es de jaque mate, y producirá los resultados pertinentes (1-0 para la victoria del jugador de las blancas, y 0-1 para la victoria del jugador de las negras Si un jugador pierde por tiempo o se rinde, el resultado producido será equivalente).

Tabla 40: RF-020

ID	RF-021
PRIORIDAD	A
TITULO	Comprobación de posición de tablas
DEPENDENCIAS	RF-007
DESCRIPCION	El programa comprobará si se han producido tablas (empate) y emitirá un resultado de ½-½. Se comprobará si son tablas por material insuficiente, por tiempo agotado, por triple repetición, por la regla de los 50 movimientos, por rey ahogado, o por mutuo acuerdo.

Tabla 41: RF-021

ID	RF-022
PRIORIDAD	A
TITULO	Comprobación de posición de tablas por rey ahogado
DEPENDENCIAS	RF-007, RF-025
DESCRIPCION	El programa comprobará si se han producido tablas (empate) por rey ahogado y emitirá un resultado de ½-½

Tabla 42: RF-022

ID	RF-023
PRIORIDAD	A
TITULO	Comprobación de posición de tablas por material insuficiente
DEPENDENCIAS	RF-005, RF-023
DESCRIPCION	El programa comprobará si se han producido tablas (empate) por material insuficiente y emitirá un resultado de ½-½

Tabla 43: RF-023

ID	RF-024
PRIORIDAD	A
TITULO	Comprobación de posición de tablas por triple repetición
DEPENDENCIAS	RF-005, RF-021
DESCRIPCION	El programa comprobará si se han producido tablas (empate) por triple repetición y emitirá un resultado de ½-½

Tabla 44: RF-024

ID	RF-025
PRIORIDAD	A
TITULO	Abandonar partida/rendirse
DEPENDENCIAS	RF-012
DESCRIPCION	El programa permitirá al jugador rendirse durante una partida en juego.

Tabla 45: RF-025

ID	RF-026
PRIORIDAD	A
TITULO	Ofrecer tablas al rival durante la partida
DEPENDENCIAS	RF-012
DESCRIPCION	El programa permitirá al jugador ofrecer tablas al rival en una partida en curso.

Tabla 46: RF-026

ID	RF-027
PRIORIDAD	A
TITULO	Elegir la pieza a coronar cuando un peón promociona
DEPENDENCIAS	RF-007
DESCRIPCION	El programa permitirá al jugador elegir entre Dama, Caballo, Alfil o Torre, cuando un peón promociona.

Tabla 47: RF-027

ID	RF-028
PRIORIDAD	B
TITULO	Permitir chat durante la partida con el rival
DEPENDENCIAS	-
DESCRIPCION	El programa permite enviar mensajes de chat al rival durante la partida

Tabla 48: RF-028

ID	RF-029
PRIORIDAD	A
TITULO	Elegir nombre de usuario al registrarse
DEPENDENCIAS	RF-005
DESCRIPCION	El programa permite al usuario registrarse con un nombre de usuario, que no exista previamente, ni contenga espacios. La longitud será de 2 a 25 caracteres.

Tabla 49: RF-029

ID	RF-030
PRIORIDAD	A
TITULO	Elegir contraseña al registrarse
DEPENDENCIAS	RF-005
DESCRIPCION	El usuario elegirá una contraseña de una longitud entre 8 y 15 caracteres, que al menos contenga una letra minúscula, una letra mayúscula y un dígito. Se permiten los siguientes símbolos alfanuméricos: _ -

Tabla 50: RF-030

ID	RF-031
PRIORIDAD	A
TITULO	Aceptar tablas al rival durante la partida
DEPENDENCIAS	RF-012, RF-026
DESCRIPCION	El programa permitirá al jugador aceptar tablas al rival en una partida en curso.

Tabla 51: RF-031

ID	RF-032
PRIORIDAD	A
TITULO	Rechazar tablas al rival durante la partida
DEPENDENCIAS	RF-012, RF-026
DESCRIPCION	El programa permitirá al jugador rechazar tablas al rival en una partida en curso.

Tabla 52: RF-032

ID	RF-033
PRIORIDAD	M
TITULO	Ver perfil de usuario
DEPENDENCIAS	-
DESCRIPCION	El programa permitirá al jugador ver su perfil de usuario.

Tabla 53: RF-033

ID	RF-034
PRIORIDAD	A
TITULO	Descargar la aplicación cliente
DEPENDENCIAS	-
DESCRIPCION	Habrá una página web para poder descargar la aplicación cliente de forma gratuita.

Tabla 54: RF-034

Requisitos de restricción

ID	RES-001
PRIORIDAD	A
TITULO	Invalidar saltos de piezas sobre otras
DEPENDENCIAS	RF-007
DESCRIPCION	El programa no permite hacer jugadas ilegales, como intentar arrastrar una pieza delante de la trayectoria de otra pieza.

Tabla 55: RES-001

ID	RES-002
PRIORIDAD	A
TITULO	Invalidar capturar piezas propias
DEPENDENCIAS	RF-007
DESCRIPCION	El programa no permite hacer jugadas ilegales, como intentar capturar una pieza propia

Tabla 56: RES-002

ID	RES-003
PRIORIDAD	A
TITULO	Invalidar poner al rey propio en jaque
DEPENDENCIAS	RF-007
DESCRIPCION	El programa no permite hacer jugadas ilegales, como poner al rey propio en jaque (una pieza rival podría capturarlo).

Tabla 57: RES-003

ID	RES-004
PRIORIDAD	A
TITULO	Invalidar realizar una jugada en el turno del rival
DEPENDENCIAS	RF-007
DESCRIPCION	El programa no permite hacer jugadas ilegales, como hacer una jugada en el turno del rival.

Tabla 58: RES-004

ID	RES-005
PRIORIDAD	A
TITULO	Invalidar realizar una jugada al finalizar la partida
DEPENDENCIAS	RF-007
DESCRIPCION	El programa no permite realizar una jugada cuando ya hay un resultado producido en la partida.

Tabla 59: RES-005

ID	RES-006
PRIORIDAD	A
TITULO	Invalidar registrarse con un nombre de usuario incorrecto
DEPENDENCIAS	RF-005
DESCRIPCION	El programa no permite elegir nombres de usuario incorrectos, y se informará para que elija otro nombre de usuario.

Tabla 60: RES_006

ID	RES-007
PRIORIDAD	A
TITULO	Invalidar registrarse con una contraseña incorrecta
DEPENDENCIAS	RF-005
DESCRIPCION	El programa no permite elegir contraseñas incorrectas, y se informará para que elija otra.

Tabla 61: RES-007

Requisitos no funcionales

ID	RNF-001
PRIORIDAD	A
TITULO	Límite de usuarios simultáneos
DEPENDENCIAS	-
DESCRIPCION	El programa no permite jugar a la vez a más de 1000 usuarios, debido a la eficiencia y el uso de threads para almacenar todas las partidas y sockets asociados.

Tabla 62: RNF-001

ID	RNF-002
PRIORIDAD	A
TITULO	Límite de partidas simultáneas
DEPENDENCIAS	-
DESCRIPCION	El programa no permite jugar a la vez a más de 500 partidas simultáneas, debido a la eficiencia y el uso de threads para almacenar todas las partidas y sockets asociados.

Tabla 63: RNF-002

ID	RNF-003
PRIORIDAD	A
TITULO	Disponibilidad limitada
DEPENDENCIAS	-
DESCRIPCION	El programa funcionará en momentos determinados, debido a que el servidor no puede ofrecer disponibilidad completa.

Tabla 64: RNF-003

4.3. Matrices de trazabilidad

Las matrices de trazabilidad nos ayudan a verificar que todos los requisitos, tanto de usuario como del sistema final, son cubiertos. Esto ayuda a asegurar la calidad del sistema, puesto que, si algún requisito no está cubierto, es por algún defecto en el análisis o el diseño.

4.3.1. Casos de uso-Requisitos funcionales del sistema

Tiene como finalizar saber si hemos cubierto todos los requisitos funcionales con algún caso de uso.

CU\RF	RF-001	RF-002	RF-003	RF-004	RF-005	RF-006	RF-007
CU-001	X						
CU-002			X				
CU-003		X		X			
CU-004					X		
CU-005						X	
CU-006							
CU-007							X
CU-008							

Tabla 65: Matriz de trazabilidad Caso Uso-Requisitos Funcionales (I)

CU\RF	RF-008	RF-009	RF-010	RF-011	RF-012	RF-013	RF-014
CU-001							
CU-002	X		X	X	X		
CU-003							
CU-004							
CU-005							
CU-006							
CU-007						X	X
CU-008		X					

Tabla 66: Matriz de trazabilidad Caso Uso-Requisitos Funcionales (II)

CU\RF	RF-015	RF-016	RF-017	RF-018	RF-019	RF-020	RF-021
CU-001							
CU-002							
CU-003							
CU-004							
CU-005							
CU-006							
CU-007	X	X	X	X	X	X	X
CU-008							

Tabla 67: Matriz de trazabilidad Caso Uso-Requisitos Funcionales (III)

CU\RF	RF-022	RF-023	RF-024	RF-025	RF-026	RF-027	RF-028
CU-001							
CU-002					X		X
CU-003							
CU-004							
CU-005							
CU-006							
CU-007	X	X	X	X		X	
CU-008							

Tabla 68: Matriz de trazabilidad Caso Uso-Requisitos Funcionales (IV)

CU\RF	RF-029	RF-030	RF-031	RF-032	RF-033	RF-034
CU-001						
CU-002			X	X		X
CU-003						X
CU-004	X	X				
CU-005						
CU-006					X	
CU-007						
CU-008						

Tabla 69: Matriz de trazabilidad Caso Uso-Requisitos Funcionales (V)

4.3.2. Requisitos de usuario-Requisitos funcionales del sistema

RU\RF	RF-001	RF-002	RF-003	RF-004	RF-005	RF-006	RF-007
RU-001	X	X	X	X			
RU-002					X	X	
RU-003							
RU-004							X
RU-005							
RU-006							
RU-007							
RU-008							
RU-009							

Tabla 70: Matriz Requisitos de usuario-Requisitos funcionales (I)

RU\RF	RF-008	RF-009	RF-010	RF-011	RF-012	RF-013	RF-014
RU-001							
RU-002							
RU-003							
RU-004					X	X	X
RU-005	X						
RU-006							
RU-007							
RU-008			X	X			
RU-009		X					

Tabla 71: Matriz Requisitos de usuario-Requisitos funcionales (II)

RU\RF	RF-015	RF-016	RF-017	RF-018	RF-019	RF-020	RF-021
RU-001							
RU-002							
RU-003							
RU-004	X	X	X	X	X	X	X
RU-005							
RU-006							
RU-007							
RU-008							
RU-009							

Tabla 72: Matriz Requisitos de usuario-Requisitos funcionales (III)

RU\RF	RF-022	RF-023	RF-024	RF-025	RF-026	RF-027	RF-028
RU-001							
RU-002							
RU-003							
RU-004	X	X	X	X	X	X	
RU-005							
RU-006							
RU-007							X
RU-008							
RU-009							

Tabla 73: Matriz Requisitos de usuario-Requisitos funcionales (IV)

RU\RF	RF-029	RF-030	RF-031	RF-032	RF-033	RF-034
RU-001						
RU-002	X	X			X	
RU-003						X
RU-004			X	X		
RU-005						
RU-006						X
RU-007						
RU-008						
RU-009						

Tabla 74: Matriz Requisitos de usuario-Requisitos funcionales (V)

5. Diseño

A continuación, vamos a explicar nuestro sistema más en detalle. En esta fase, justificaremos la solución encontrada.

Primero tenemos que explicar cómo hemos desarrollado la jugabilidad de ajedrez. El elemento principal es una partida entre dos jugadores.

Tenemos que diferenciar entre la parte de codificación de los elementos de una partida (tablero, piezas, jugadas legales, resultado…) de la parte gráfica (cómo se representa gráficamente el estado de una partida.

5.1. Codificación

La codificación se podría decir que es el alma del programa. Una buena codificación resuelve gran parte del software, puesto que representa las distintas combinaciones de estados posibles.

Por ejemplo, podemos representar el tablero de la siguiente forma:

t	c	a	d	r	a	c	t
p	p	p	p	p	p	p	p
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-
P	P	P	P	P	P	P	P
T	C	A	D	R	A	C	T

Tabla 75: Codificación del estado del tablero de ajedrez

Se codifican en letras mayúsculas las piezas blancas, y en minúsculas las piezas negras. Utilizamos la letra inicial del nombre de la pieza para codificar las diferentes piezas.

Además, hay que tener en cuenta que además, las posiciones (x,y) del tablero son parte de esa codificación:

(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)

Tabla 76: Codificación de las posiciones del tablero

Con estos ingredientes, es cuestión de seguir las reglas de ajedrez, y hacer uso de estos elementos, para poder representar el conjunto de todas las posiciones y movimientos legales, así como conocer el resultado de la partida (jaque mate para las blancas, jaque mate para las negras y tablas). Aunque hay que analizar todos estos casos más en detalle, todas las reglas de ajedrez serán programadas en base a esta codificación.

Además, la codificación debe ser lo más flexible posible. Es decir, podríamos codificar, por ejemplo, las casillas dándole un número del 1 al 64; sin embargo, esto nos proporciona muy poco juego a la hora de realizar comprobaciones sobre la fila o la columna en la que está una determinada pieza. Por lo que se puede representar de muchas formas posibles, pero siempre algunas de ellas son mejores que otras, o más eficientes. Hay que tener siempre en cuenta, que el coste computacional en el tiempo es siempre un recurso más limitado que el espacio en memoria, por lo que es mejor almacenar el tablero en un array bidimensional, en el que las coordenadas x e y quedan más claras, que en un array unidimensional, el cual da muy poca flexibilidad y requiere más cálculos.

5.2. Movimiento de una pieza en el tablero

Un movimiento consiste en abandonar una determinada casilla, para ocupar otra. La casilla de destino puede estar libre, o bien estar ocupada por una pieza de color contrario. Por lo tanto, tenemos los siguientes casos:

- La casilla de destino está libre. Se codifica la casilla de origen a "-" y se coloca en la casilla de destino la pieza. Además, en la pieza se cambian los valores de los atributos de posición x e y para facilitar la gestión de consultar posiciones sin necesidad de tener que consultar siempre el tablero en busca de un determinado carácter que represente dicha pieza. Esto es más útil para mostrar el tablero por pantalla. Para hacer comprobaciones, es mejor consultar los valores de estos atributos, ya que hay reglas que necesitan la herencia de la clase Pieza. De otra forma, habría que consultar de una forma más compleja.
- La casilla de destino está ocupada por una pieza de color contrario, con lo que se captura esta pieza. Se anulan los valores x e y de la pieza anterior (valor -1), se sustituye la casilla de origen por "-" y se cambia la casilla de destino del tablero por la letra de la nueva pieza, además de actualizar la posición x y la posición y de dicha pieza.

A modo de ejemplo, si tenemos la siguiente posición inicial:

8	t	c	a	d	r	a	c	t
7	p	p	p	p	p	p	p	p
6	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-
2	P	P	P	P	P	P	P	P
1	T	C	A	D	R	A	C	T
	a	b	c	d	e	f	g	h

Ilustración 5: Codificación de la posición inicial del tablero

Es el turno de las blancas, y quieren jugar el caballo de g1 a la casilla f3. El estado del tablero quedaría de la siguiente manera:

8	t	c	a	d	r	a	c	t
7	p	p	p	p	p	p	p	p
6	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-
3	-	-	-	-	-	C	-	-
2	P	P	P	P	P	P	P	P
1	T	C	A	D	R	A	-	T
	a	b	c	d	e	f	g	h

Ilustración 6: Ejemplo de jugada realizada 1. Cf3

Vemos cómo en g1 se vacía dicha casilla, para ocupar la casilla f3 por el caballo blanco.

Ahora supongamos que las negras juegan d5, es decir, mueven el peón de d7 a d5. La posición quedaría de la siguiente manera:

8	t	c	a	d	r	a	c	t
7	p	p	p	-	p	p	p	p
6	-	-	-	-	-	-	-	-
5	-	-	-	p	-	-	-	-
4	-	-	-	-	-	-	-	-
3	-	-	-	-	-	C	-	-
2	P	P	P	P	P	P	P	P
1	T	C	A	D	R	A	-	T
	a	b	c	d	e	f	g	h

Ilustración 7: Ejemplo de jugada realizada 1…d5

De esta manera, podemos tener codificado el juego del ajedrez, en base a una especie de matriz. Vemos como no es necesario mezclar la parte de codificación, de la parte gráfica del juego, puesto que la parte gráfica simplemente se irá encargando de consultar esta matriz, que estará en memoria principal, para ir dibujando las piezas sobre el tablero.

Codificación del turno de la partida

Para codificar el turno del jugador (blancas o negras), se emplea un atributo numérico. Es cierto que se podría codificar con un atributo booleano (true/false). Sin embargo, un atributo numérico aporta más información, acerca del número de jugadas que se han realizado, y que posteriormente se utilizará para almacenar la partida jugada en el formato PGN.

Así, tenemos que, si el turno es impar, comenzando desde 1, el turno es de las blancas; y si es par, el turno es de las negras.

Por lo tanto, un jugador sólo puede mover las piezas de su color, cuando es su turno.

Aunque en ajedrez, hay que distinguir entre el valor de este atributo, denominado habitualmente “ply” (utilizado en motores de ajedrez para calcular cuánto de profundo es el algoritmo que predice jugadas), del número de jugadas propiamente, el cual se suele considerar como jugadas de las blancas. Es decir, en el ejemplo anterior, la secuencia sería 1. Cf3 d5, ambos han realizado 1 jugada, pero corresponde a 2 plys.

Los motores de ajedrez, suelen tener como referencia este último valor (ply), para comparar cómo de profundo analiza.

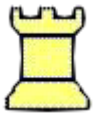
5.3. Creación gráfica de las piezas

Las piezas se crearon gracias a un estilo existente llamado linares [24]. En base a este estilo, se recortaron los fondos de las piezas con un software de la siguiente manera:

https://www.javierrguez.com/transparencia-en-png-en-corel-photo-paint/

El resultado final es el siguiente:

Torre blanca sobre fondo blanco:



Torre blanca sobre fondo marrón:

Torre negra:

Torre negra sobre fondo marrón:

Caballo blanco:

Caballo blanco sobre fondo marrón:

Caballo negro:

Caballo negro sobre fondo marrón:

Dama blanca:

Dama blanca sobre fondo marrón

Dama negra:

Dama negra sobre fondo marrón

Rey blanco:

Rey blanco sobre fondo marrón:

Rey negro:

Rey negro sobre fondo marrón:

Alfil blanco:

Alfil blanco sobre fondo marrón:

Alfil negro:

Alfil negro sobre fondo marrón:

Peón blanco:

Peón blanco sobre fondo marrón:

Peón negro:

Peón negro sobre fondo marrón:

Ilustración 8: Diseño gráfico de las piezas de ajedrez

5.4. Modelos de clases

Tenemos un modelo de clases que van a representar la codificación de los elementos de una partida. Dicho modelo es el siguiente:

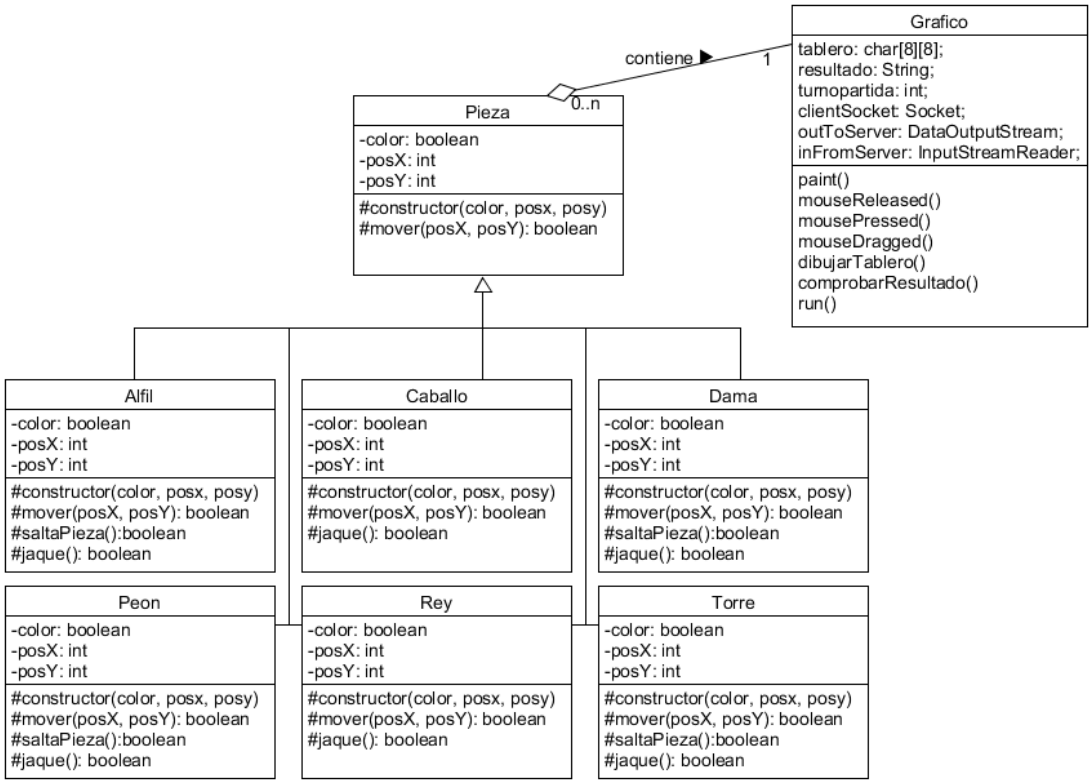


Ilustración 9: Diseño de clases

Se puede ver que, hay una clase importante, llamada Pieza, que es la clase padre de todos los tipos de piezas posibles (Alfil, Caballo, Dama, Peón, Rey, Torre). La razón principal de la existencia de esta clase es la siguiente: hay una regla en ajedrez muy importante, que dice que una pieza no puede saltar a otra pieza. Por lo que esta clase tiene sentido contemplarla, para que las condiciones de que una jugada sea legal, tengan en cuenta cualquier tipo de pieza, independientemente del color (blancas, negras) o el tipo (mencionado arriba).

Con esto, tenemos una parte de la implementación bastante importante, puesto que es una regla difícil de imaginar cómo se programa, sin tener en cuenta esta herencia.

5.5. Movimiento de las piezas

Ahora vamos a explicar en líneas generales cómo se programa la movilidad de las piezas.

Los alfiles mueven en diagonal, y para ello existe una regla bien sencilla. Al tener las casillas numeradas (del 0 al 7) para los ejes x e y, cuando un alfil mueve, mueve en diagonal. Y lo que siempre se cumple es que el desplazamiento a lo largo de un eje es el mismo que el del eje contrario. Esto es, el número de casillas que se desplaza en el eje x coincide con el número de casillas que se desplaza esta pieza en el eje y. Además, se debe comprobar siempre que la jugada no ponga al rey en jaque, ni salte ninguna otra pieza, aunque si la casilla de destino es una en la que existe otra pieza, si es de color contrario, puede capturarla por lo que, la posición de la pieza enemiga se coloca a un valor negativo, como -1 (fuera del tablero) y se actualiza la posición de ese alfil en el tablero (se borra la anterior y se incluye la de destino).

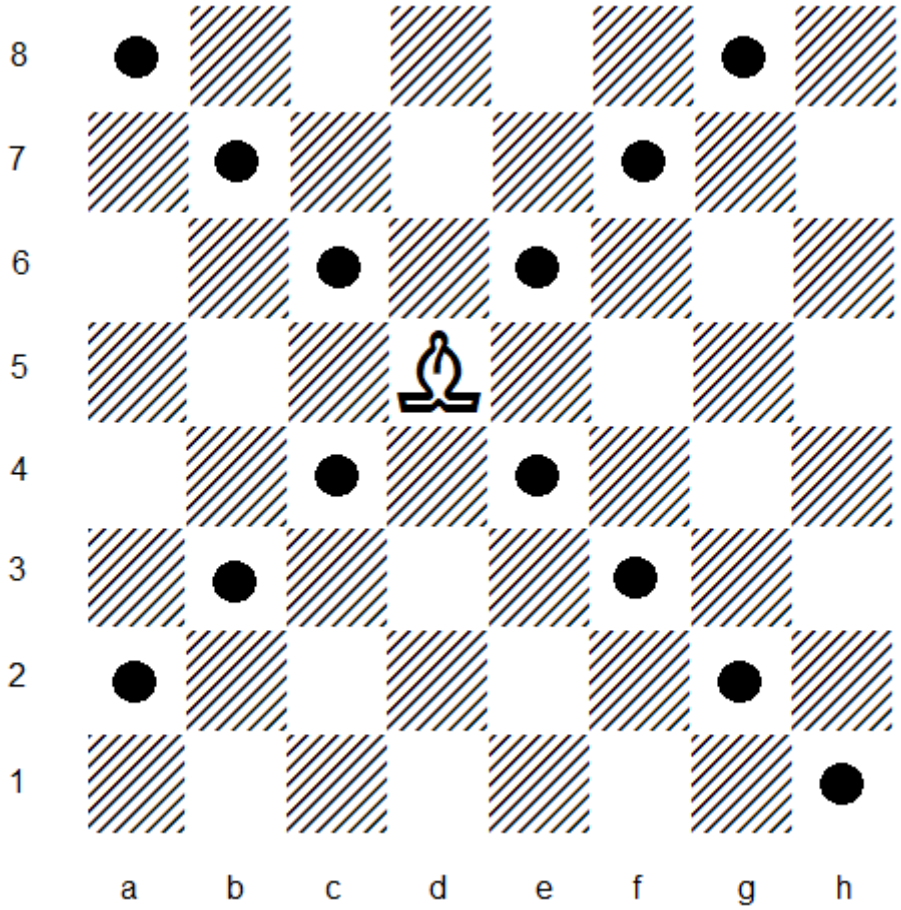


Ilustración 10: Jugadas de alfil

En este caso, como la trayectoria es una línea recta, mediante la ecuación y=mx donde m es la pendiente, y es igual a 1, tenemos que y=x. Denominamos dx al desplazamiento a lo largo del eje x, y dy al desplazamiento a través del eje y de coordenadas. En este caso tenemos, en concreto, |dy|=|dx| (el desplazamiento en el eje y es el mismo que en el del eje x, aunque esto da las siguientes posibilidades:

dy=-dx

dy=dx

-dy=dx

-dy=-dx

El caballo tiene una peculiaridad y es que, no necesita implementar además la regla de que no pueda saltar a otras piezas, porque su movimiento en realidad, no sigue una trayectoria lineal. Todas las demás piezas deben implementar esa regla salvo el caballo (y el rey). Por lo que, si un jugador decide mover el caballo, sólo se comprobará que el rey propio no quede en jaque y que la casilla de destino esté libre, o haya una pieza de color contrario; y que se mueva una casilla de diferencia en uno de los ejes, y dos en el eje contrario (además de estar en los límites del tablero). Esto hace que potencialmente

un caballo pueda tener un máximo de 8 jugadas disponibles teniendo como (dx, dy) las siguientes combinaciones: (-1, 2), (1, 2), (-1,-2), (1, -2),

(-2, 1), (2, 1), (-2, -1), (2, -1).

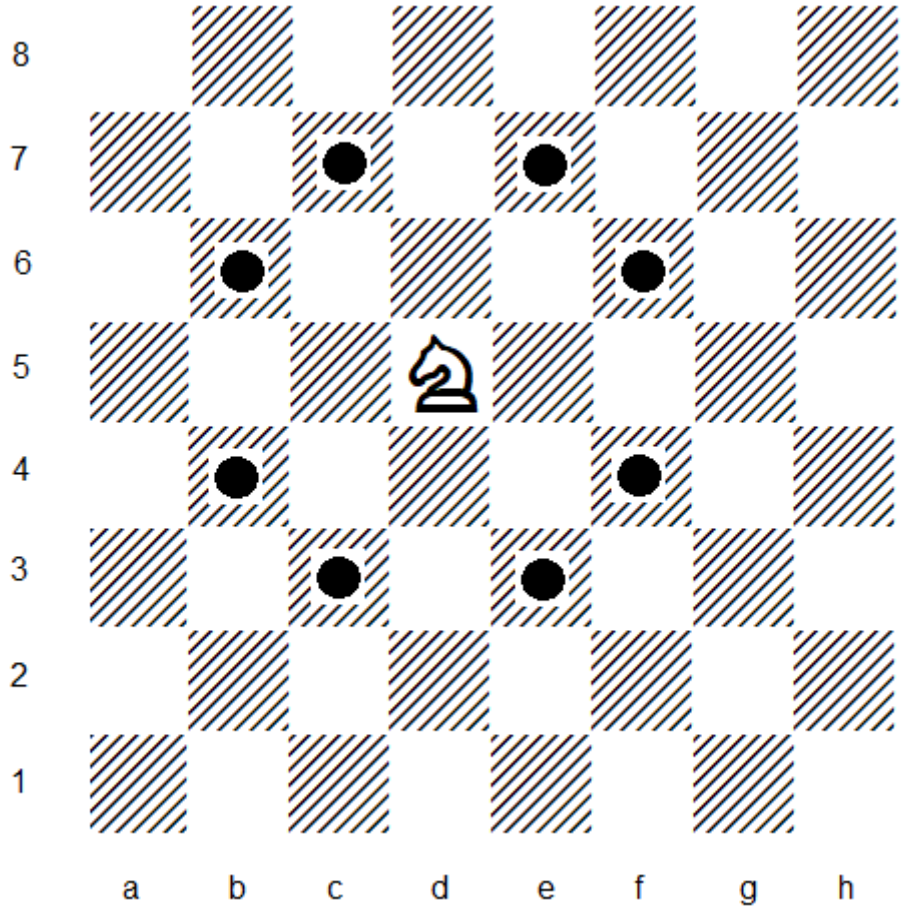


Ilustración 11: Jugadas de caballo

De hecho, su movilidad se define como la casilla más cercana que no se encuentra ni en diagonal ni en horizontal.

El peón es sin dudas, la pieza más extraña del juego. Para empezar, todas las demás piezas comen una pieza rival de la misma manera que al mover. Los peones tienen una serie de peculiaridades que hacen que algunos ajedrecistas, distingan entre piezas (normales) y peones.

Los peones se sitúan en la segunda fila (fila 2 para el blanco y fila 7 para el negro) al inicio del juego. En su primera jugada, cada peón puede avanzar una o dos casillas hacia adelante, a lo largo del eje vertical. Pero después sólo pueden avanzar de uno en uno. Sin embargo, al capturar, no capturan como avanzan. Es decir, sólo pueden capturar una pieza enemiga cuando ésta se sitúa delante de este peón pero en las casillas adyacentes (izquierda o derecha). Y por lo tanto, adoptaría la posición de esa pieza enemiga, por lo que cambiaría de columna al finalizar la captura:

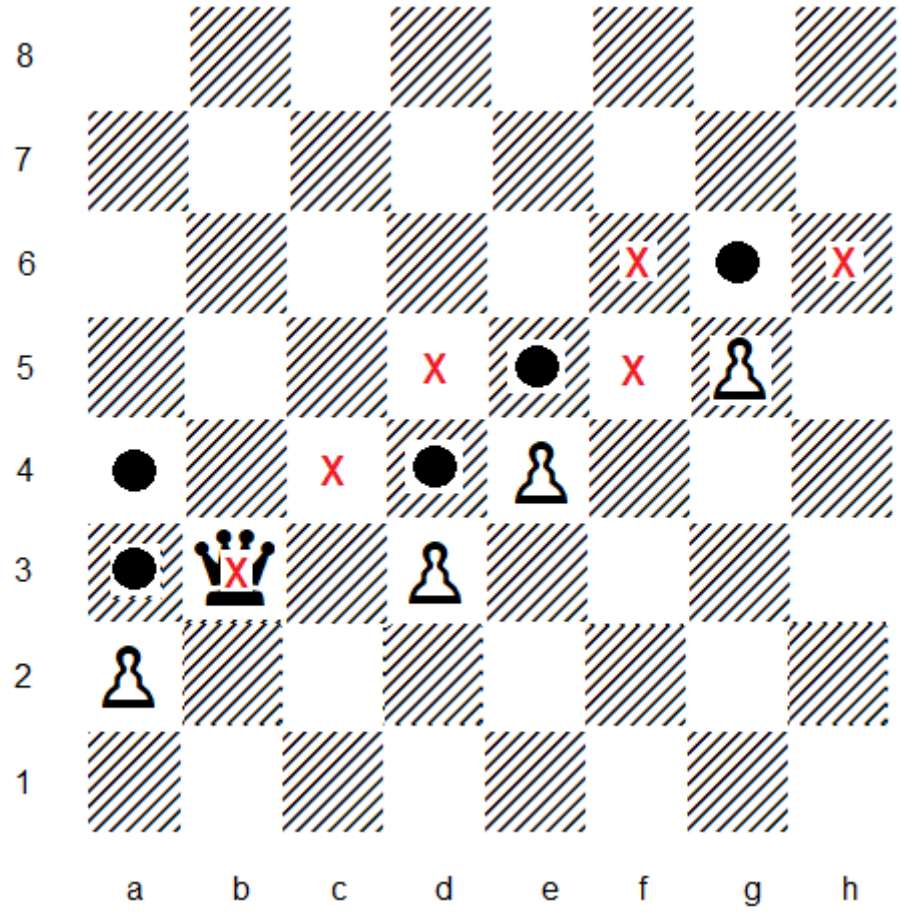


Ilustración 12: Jugadas de peón

En este caso, tenemos la siguiente situación. El peón de a2 podría o bien capturar la dama de b3, con lo cual, pasaría de a2 a ocupar la casilla b3 con lo que la dama se retiraría del tablero; o bien avanzar una o dos casillas en adelante, puesto que ocupa la casilla de salida. El peón de d3 sólo puede avanzar una casilla en adelante, hacia d4, puesto que las casillas adyacentes de captura no contienen piezas enemigas. Pero si en d4 hubiera una pieza enemiga, no podría ni siquiera mover. El peón de e4 sólo puede avanzar a e5, y el de g5 sólo puede avanzar a g6. Se marcan con una "x" aquellas casillas en las que si hubiera una pieza enemiga, un peón podría capturarla y ocupar su posición. En concreto, el peón puede capturar en diagonal, pero avanza de frente.

Hay una regla especial, que se denomina captura "en passant" o al paso. Esta es la excepción, y ocurre cuando un peón aún no ha salido de su casilla de origen y avanza dos casillas adelante, y un peón rival está justo a su izquierda o derecha. Entonces este peón, puede capturarlo también, como si sólo hubiera avanzado una casilla (en lugar de las dos que puede avanzar inicialmente).

Además, si el peón avanza hacia la última fila del tablero (fila 8 si es blanco, o fila 1 si es negro), el jugador elegirá la pieza de su mismo color con la que se va a sustituir (Dama, Caballo, Alfil o Torre).

Ahora tenemos la torre, que se desplaza a lo largo de un solo eje, las casillas que se deseen. Por lo que hay cuatro posibilidades:

(dx, 0), (-dx, 0), (0, -dy) y (0, dy) como desplazamientos.

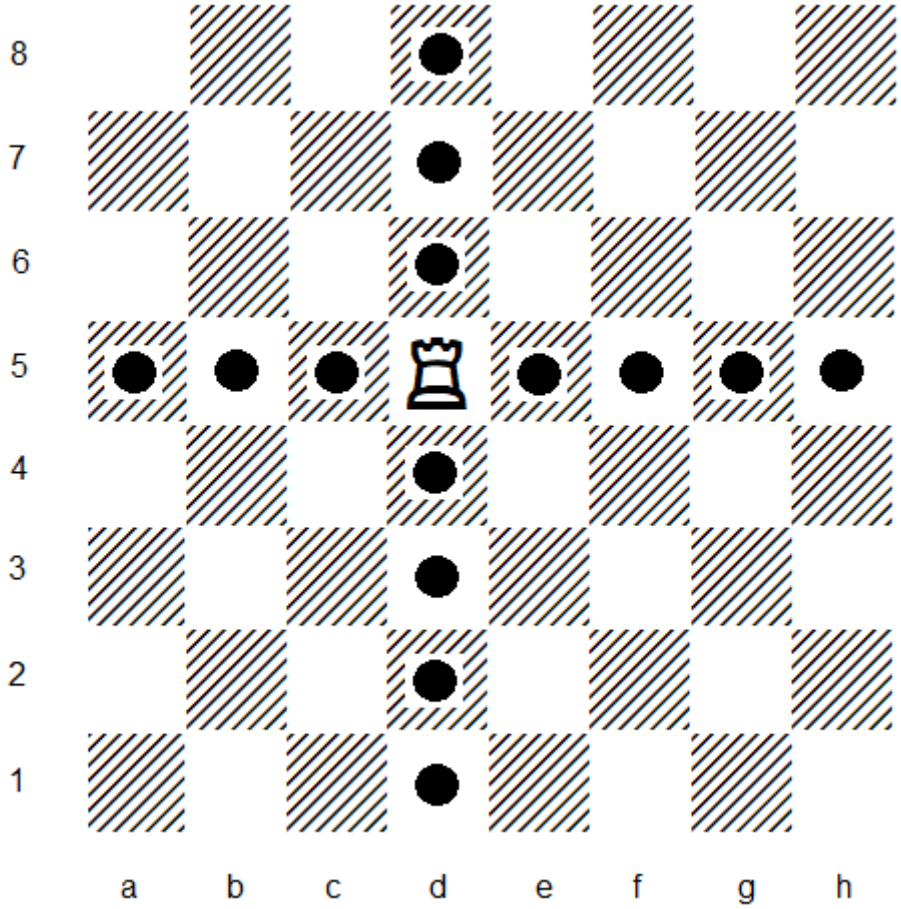


Ilustración 13: Jugadas de torre

La dama se puede implementar como una combinación entre un alfil y una torre, en la misma pieza.

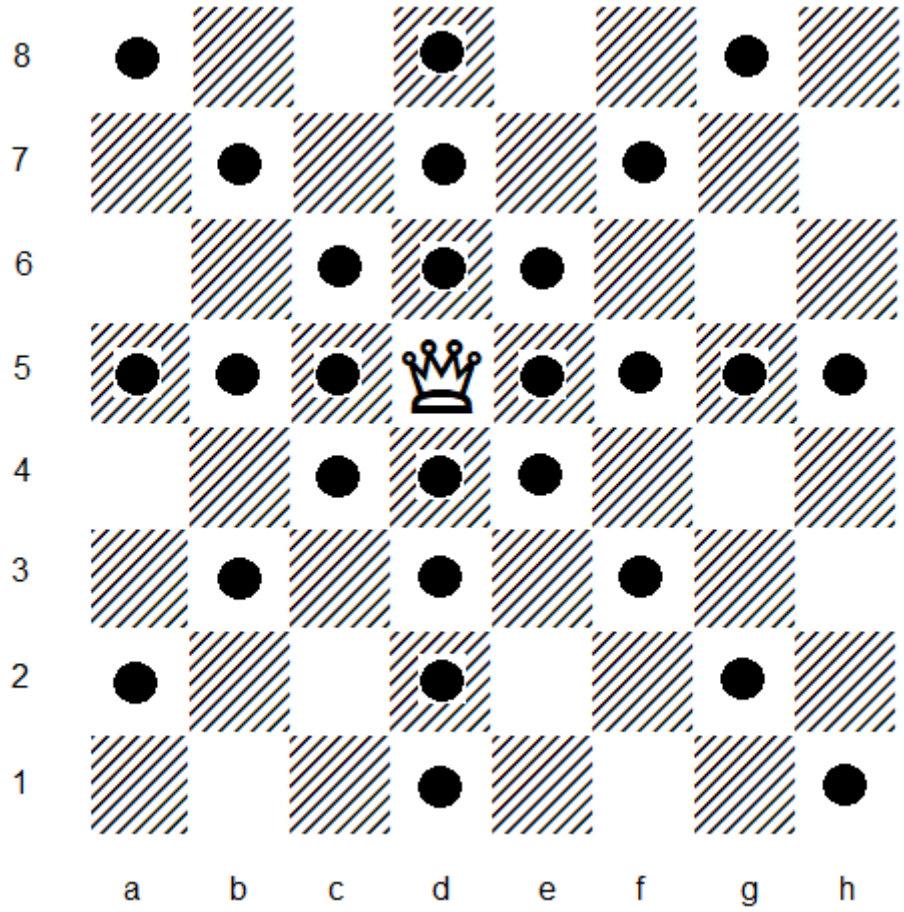


Ilustración 14: Jugadas de dama

El rey es la pieza más importante y que también, tiene una serie de peculiaridades.

Primero, su movimiento es de una casilla a lo largo de cualquier eje, y existen potencialmente 8 tipos de desplazamientos: (0, 1), (0, -1), (1, 0), (-1, 0), (1, 1), (1, -1), (-1, 1), (-1, -1). Esto hace que el rey pueda incluso romper las leyes de la geometría. Es decir, se considera un desplazamiento de una casilla, cuando se desplaza en diagonal, por lo que en verdad se habría desplazado dos casillas. Esto desconcierta mucho puesto que, muchas veces es decisivo poder llegar a capturar un peón con el rey si su distancia es suficiente como para poder detenerlo y que el peón no pueda convertirse en dama porque no le dé tiempo a llegar a la última fila. Por lo que el rey puede realizar maniobras de zig-zag sin perder distancia.

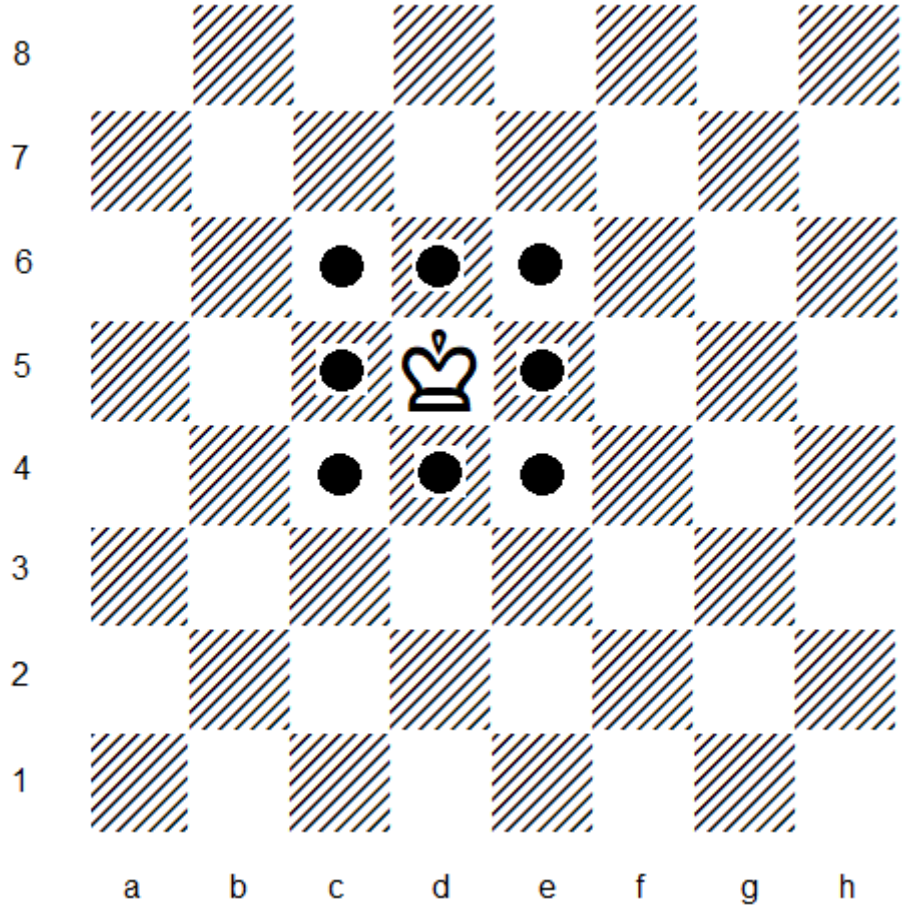


Ilustración 15: Jugadas de rey

Además, tenemos la jugada enroque, que tiene su complejidad de programación, puesto que no es evidente implementarla, ya que hay que dividirla en varias posibilidades (enroque largo y enroque corto, con todas sus peculiaridades).

Enroque

El enroque es una regla de ajedrez, que hay que analizar más, puesto que es una de las excepciones que cuesta entender a la mayoría de personas que están aprendiendo por primera vez a jugar.

Es una jugada que involucra al rey y a una de sus torres. En inglés se denomina “castling”, que proviene de castillo, o torre. Es decir, el rey se refugia en su castillo.

Es una jugada novedosa, que antiguamente no existía, pero que fue introducida para que las torres tomasen más rápidamente actividad por las columnas centrales, en lugar de permanecer abandonadas en las esquinas del tablero, la mayor parte del tiempo. Además, el rey toma una casilla más segura, con esta jugada, puesto que se forma una especie de “castillo” con su torre y su barrera de peones, lo cual le da mucha protección. Ahora vamos a proceder a explicar la jugada, para poder comprender cómo se programa.

Tenemos el siguiente diagrama:

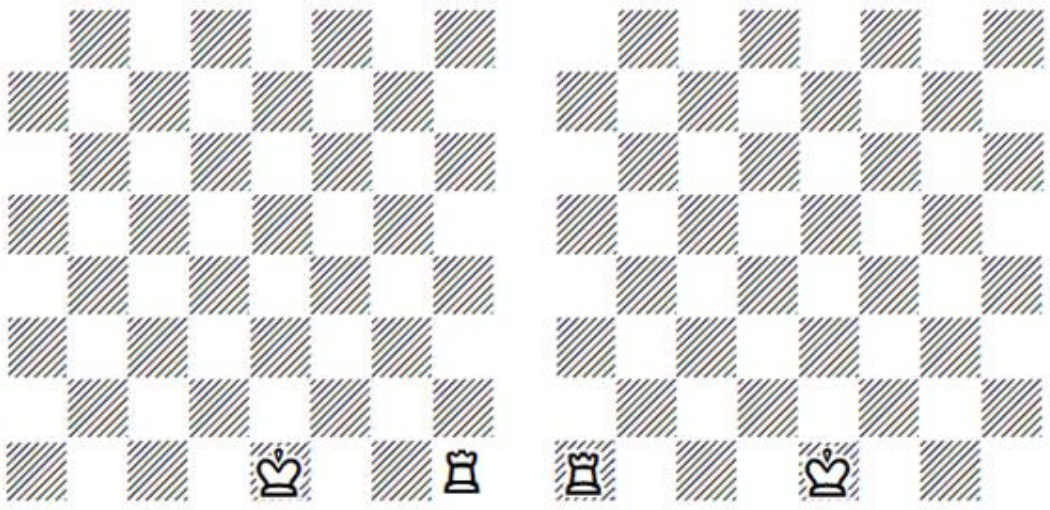


Ilustración 16: Enroque, disposición inicial

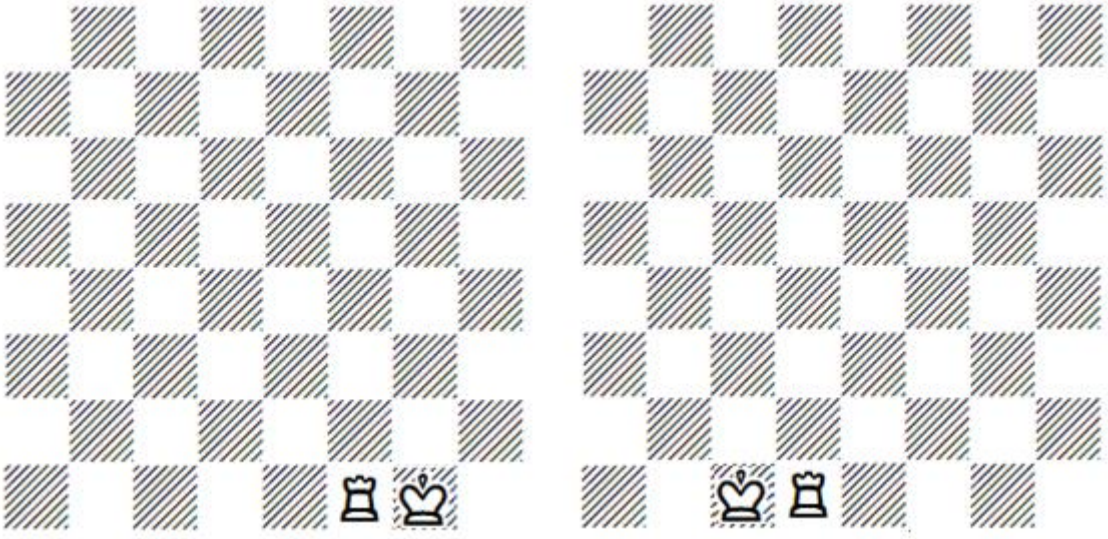


Ilustración 17: Enroque, disposición final

En la parte izquierda, tenemos lo que sería el enroque corto. El rey mueve dos casillas a la derecha, y la torre pasa hacia el otro lado, a la casilla adyacente al rey.

En la parte derecha, tenemos el enroque largo. De la misma manera, el rey mueve dos casillas, pero esta vez a la izquierda, pasando la torre a la casilla adyacente al rey.

Es la única jugada en la cual, el rey puede mover dos casillas, y a su vez, se mueven dos piezas en una misma jugada. Es decir, se considera como una jugada de rey, aunque involucre el hecho de mover además la torre. De hecho, en un torneo oficial de forma presencial, el jugador debe primero mover el rey y con la misma mano, posteriormente, pasar la torre hacia el otro lado, para que la jugada sea considerada correcta. De otra forma, si tocase antes la torre, se considera que no va a enrocar, por lo que automáticamente debe mover esa torre, pero ya no está habilitado para realizar la

jugada que acabamos de explicar, aunque en un futuro sí podría enrocarse, pero tendría que ser con la otra torre.

Para poder enrocar durante una partida, además debemos considerar según las reglas oficiales de la FIDE, los siguientes casos:

1. El rey no ha movido previamente. En el momento que el rey mueve, ya no puede enrocar nunca más.
2. Si mueve una de las torres, sólo puede enrocar con la otra torre. Si ha movido las dos torres, no podrá enrocar más.
3. Entre el rey y la torre con la cual se va a enrocar, no debe haber piezas de ningún tipo.
4. Las casillas de tránsito del rey para enrocarse no deben estar atacadas. Esto quiere decir, que el rey no puede estar en jaque, ni las dos casillas hacia el enroque deben ser atacadas por piezas enemigas. Atacar significa amenazar con capturar una pieza enemiga. Aquí imaginamos como si existieran piezas imaginarias, aunque las casillas estén libres, para considerar este punto.
5. Obviamente, tiene que ser el turno del jugador, y no haber finalizado la partida.

Por lo tanto, la regla del enroque se podría decir que es la más difícil de todas ellas, puesto que involucra numerosas pruebas. Por lo tanto, la simulación de validar que un jugador puede enrocarse, tiene que cumplir todos estos puntos. En el momento que uno de ellos no se cumple, se debe invalidar esta posibilidad.

En inglés, se considera dos casos, que corresponde a lo visto anteriormente:

- Castle kingside: Enrocarse en el flanco del rey, es decir, enroque corto.
- Castle queenside: Enrocarse en el flanco de dama, es decir, el enroque largo.

Los flancos son los sectores en los que se divide el tablero. Tenemos la columna d en la cual las damas (reinas) se sitúan al inicio de una partida. Y la columna e, en la cual los reyes se sitúan al comenzar. La barrera imaginaria entre la dama y el rey es lo que divide los dos flancos, y de ahí la nomenclatura “kingside” y “queenside”.

Tablas por triple repetición

Esta es otra de las reglas más complicadas a la hora de programar, puesto que tenemos que ir almacenando cada estado del tablero (incluyendo el turno, el derecho de enrocar, en ambos lados y la posibilidad de capturar al paso), y verificar que si se repite este estado 3 veces (no necesariamente de forma consecutiva), la partida se considera tablas automáticamente.

Evitar dar validez a saltos de una pieza sobre otra

Se aprovecha la codificación numérica de las casillas para resolver esta regla del juego.

Tenemos siempre una posición (x,y) de origen, y otra posición (x,y) de destino, a la hora de querer mover una determinada pieza. Además, cada pieza se sitúa en una posición (x,y) del tablero.

Ahora vamos a analizar en detalle, cómo evitaríamos permitir una jugada en la cual la pieza quiera saltar en su trayectoria a cualquier otra.

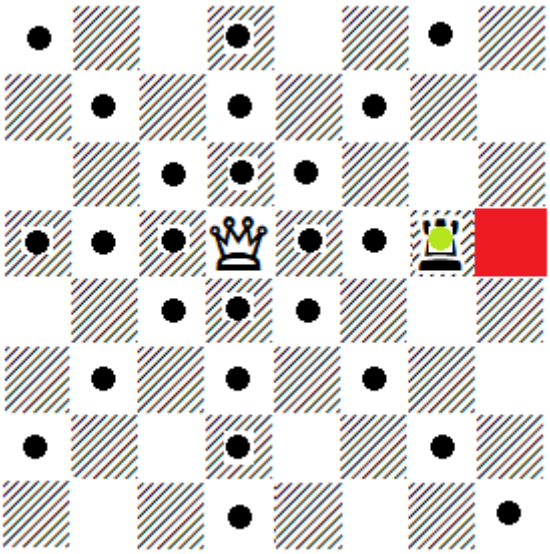


Ilustración 18: Evitar saltos de piezas sobre otras. Caso I

Imaginemos que la dama que ahora mismo está en la casilla d5, quiere ir a h5. La casilla original de esa dama, en d5, tiene como coordenadas (3,4). La casilla de la torre negra tiene como coordenadas (6,4). Y la casilla de destino, a la cual desea ir la dama blanca, tiene como coordenadas (7,4). Aquí podemos ver rápidamente cuál es el patrón de salto de la dama blanca sobre la torre negra. Es decir, la posición x de destino de la dama blanca es mayor que la posición actual de la torre negra (7>6). Si la posición “x” de origen es menor a alguna pieza y la de destino es mayor, y ambas piezas están en la misma fila (en este caso, “y”=4), hay que invalidar esta posibilidad, y en ese caso, se devuelve false en el método que desea mover esta pieza. Por lo tanto, el tablero no se actualiza. En ajedrez online, se permite que el jugador se equivoque y tenga la posibilidad de elegir otra jugada, por lo que no cambia el turno ni el resultado.

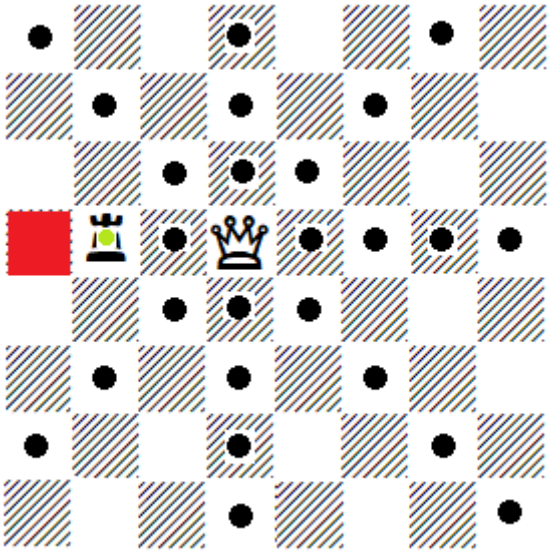


Ilustración 19: Evitar saltos de piezas sobre otras. Caso II

Este caso es similar al anterior. Aquí la casilla de la dama de origen tiene como coordenadas (3,4), la casilla de la torre negra tiene como coordenadas (1,4) y la casilla de destino de la dama blanca es (0,4). Como la casilla de origen de la dama y de la torre tienen la misma componente “y”, y la casilla de origen de la dama blanca tiene como componente “x” 3 que es mayor que la de la torre (x=1), pero la casilla de destino de la dama blanca es menor a la de la torre (x=0), hay que invalidar esta jugada.

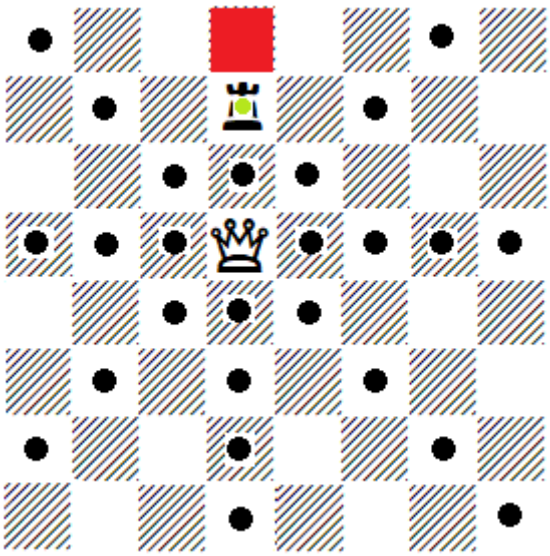


Ilustración 20: Evitar saltos de piezas sobre otras. Caso III

En este caso, la dama está en la posición (3,4) y la torre en la posición (3,6). La columna de ambos es la misma (x=3). La casilla de destino de la dama es la posición (3,7). Como la casilla de origen de la dama, tiene como componente “y” un valor de 4, que es menor al de la componente “y” de la torre, que es 6; pero la casilla de destino tiene como componente “y” un valor mayor a la de dicha torre (y=7), de nuevo hay un cambio de

signo (pasar de un valor mayor, a un valor menor, o viceversa, entre piezas), y se invalida la jugada.

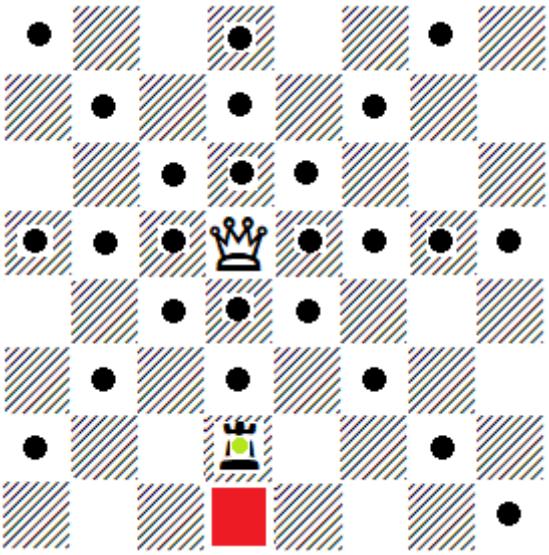


Ilustración 21: Evitar saltos de piezas sobre otras. Caso IV

En este caso, tenemos la casilla origen de la dama, en la posición (3,4), la torre está en la posición (3,1) y la dama quiere ir a la posición (3,0). Actualmente, ambas piezas están en la misma columna (x=3) y la dama tiene una componente "y" mayor a la de la torre. Como la dama quiere ir a una casilla cuya componente "y" es menor a la de la otra pieza, se tiene que invalidar la jugada.

A continuación, vamos a explicar los otros 4 casos, los cuales tienen que ver con piezas que están en la misma diagonal.

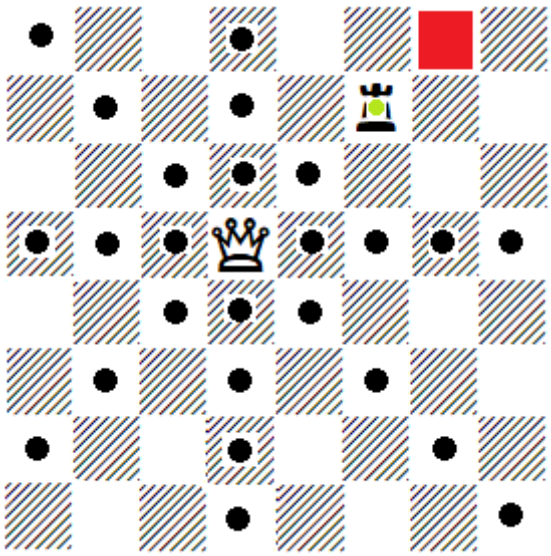


Ilustración 22: Evitar saltos de piezas sobre otras. Caso V

La dama se sitúa en la posición (3,4), y una torre está en la posición (5,6). Sabemos que están en la misma diagonal, porque si comparamos la diferencia a lo largo del eje x, y la diferencia a lo largo del eje y, ambas son iguales, en este caso, 2. Es decir, |5-3|=|6-4|.

Pues bien, en este caso, la posición x de origen de la dama es menor a la posición x de la torre. La posición y de la dama es menor a la posición y de la torre. Pero la casilla de destino de la dama tiene ambas componentes mayores a la de la torre (la casilla de destino es (6,7)). En este caso, hay que invalidar la jugada.

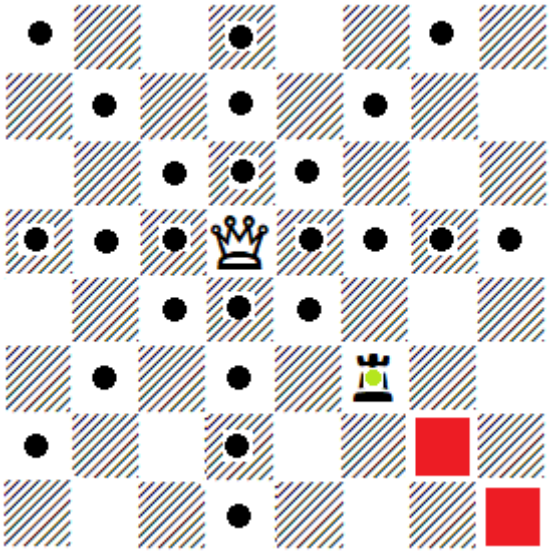


Ilustración 23: Evitar saltos de piezas sobre otras. Caso VI

La dama se sitúa en la posición (3,4), y una torre está en la posición (5,2). Sabemos que están en la misma diagonal, porque si comparamos la diferencia a lo largo del eje x, y la diferencia a lo largo del eje y, ambas son iguales. Es decir, |5-3|=|2-4|.

Pues bien, en este caso, la posición x de origen de la dama es menor a la posición x de la torre. La posición y de la dama es mayor a la posición y de la torre. Pero la casilla de destino de la dama tiene la componente x mayor a la de la torre, y la componente “y” menor (la casilla de destino es (6,1)). En este caso, hay que invalidar la jugada, puesto que hay de nuevo intercambio de condiciones (pasar de una posición menor a otra mayor).

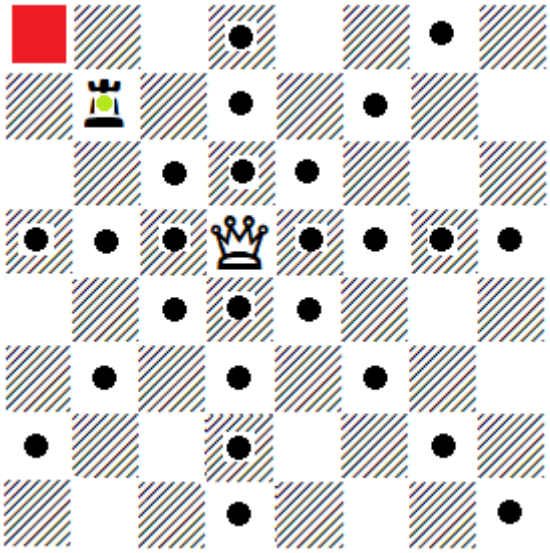


Ilustración 24: Evitar saltos de piezas sobre otras. Caso VII

La dama se sitúa en la posición (3,4), y una torre está en la posición (1,6). Sabemos que están en la misma diagonal, porque si comparamos la diferencia a lo largo del eje x, y la diferencia a lo largo del eje y, ambas son iguales. Es decir, |3-1|=|4-6|.

Pues bien, en este caso, la posición x de origen de la dama es mayor a la posición x de la torre. La posición y de la dama es menor a la posición y de la torre. Pero la casilla de destino de la dama tiene la componente x menor a la de la torre, y la componente "y" mayor (la casilla de destino es (0,7)). En este caso, hay que invalidar la jugada.

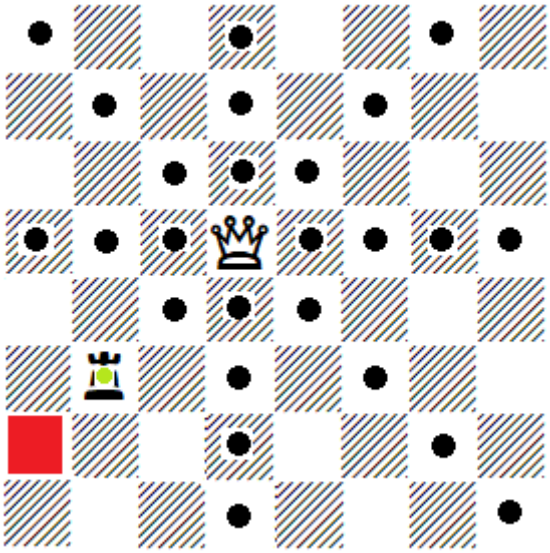


Ilustración 25: Evitar saltos de piezas sobre otras. Caso VIII

La dama se sitúa en la posición (3,4), y una torre está en la posición (1,2). Sabemos que están en la misma diagonal, porque si comparamos la diferencia a lo largo del eje x, y la diferencia a lo largo del eje y, ambas son iguales. Es decir, |3-1|=|4-2|.

Pues bien, en este caso, la posición x de origen de la dama es mayor a la posición x de la torre. La posición y de la dama es mayor a la posición y de la torre. Pero la casilla de destino de la dama tiene la componente x menor a la de la torre, y la componente "y" menor también (la casilla de destino es (0,1)). En este caso, hay que invalidar la jugada.

Los peones pueden capturar al paso.

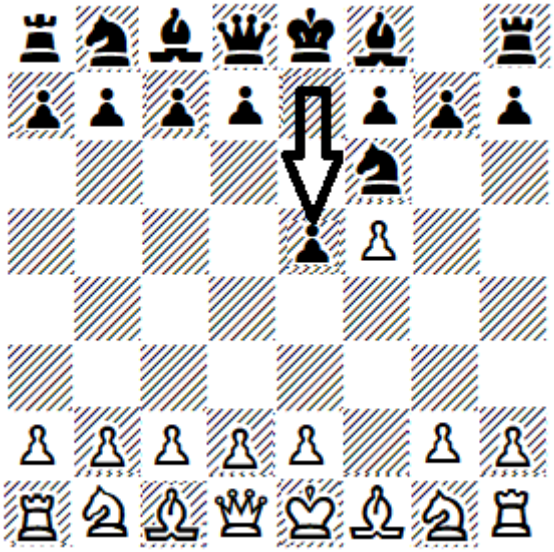


Ilustración 26. Captura al paso (I)

Si un peón avanza dos casillas en su primera jugada, y acaba en una fila en la cual la casilla adyacente hay un peón rival, dicho peón rival puede capturar ese peón como si hubiese avanzado el rival una casilla, en lugar de dos. Esto se produce siempre que la jugada inmediata sea la de la captura. De lo contrario, una vez pasado ese turno, ya no se puede realizar la captura al paso, a menos que el peón de la otra casilla adyacente haga lo propio también. La siguiente imagen ilustra cómo quedaría la posición final de la captura al paso ("en passant").

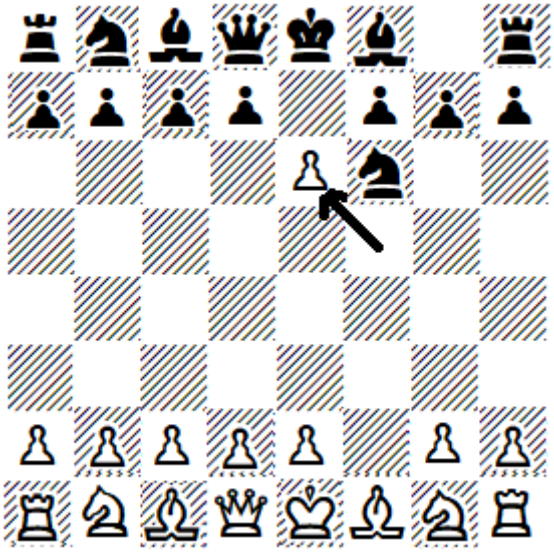


Ilustración 27: Captura al paso (II)

Vemos como se captura el peón negro, como si éste hubiese avanzado una sola casilla. Si las blancas hubiesen realizado otra jugada, ese peón negro ya no podría capturarse al paso.

5.6. Parte gráfica

La parte gráfica consiste en tomar la codificación y representar las piezas y el tablero según el estado de la codificación en memoria principal, de dicho tablero. Por lo que ha sido lo más sencillo de implementar. Mediante una clase con eventos de ratón, para poder mover las piezas, al soltar una pieza en una casilla destino, se llaman a los métodos que validan las jugadas.

Además, se ha incluido la funcionalidad de obtener, en notación algebraica [3] las jugadas realizadas, con el objetivo de, posteriormente, poder descargarlas, y así cualquier programa de ajedrez pueda interpretar dichas partidas, puesto que siguen un estándar, denominado PGN (Portable Game Notation [3]).

5.7. Adaptación para jugar a través de Internet

Ahora pasamos a explicar la segunda fase del desarrollo, que es totalmente diferente, puesto que, a partir de aquí, tenemos que conseguir que dicho software, que de momento sólo sirve para albergar una partida entre dos jugadores, pero desde un mismo ordenador, sea ahora capaz de permitir partidas entre ordenadores remotos. Para ello usaremos Internet. Se hará uso de algunos recursos de la web, como lo son las direcciones IP, los puertos y los protocolos de red. Es cierto que, además, se incluirá una página web principal para poder descargar la aplicación, y además, poder tener un sitio donde subir automáticamente las partidas jugadas una vez finalizadas, y que la propia

aplicación se encargue de ello (mediante mensajes http). Pero nos hemos centrado en el funcionamiento de una partida de ajedrez, puesto que, si una partida entre dos ordenadores remotos funciona, podemos utilizar esto como base para la realización de torneos online.

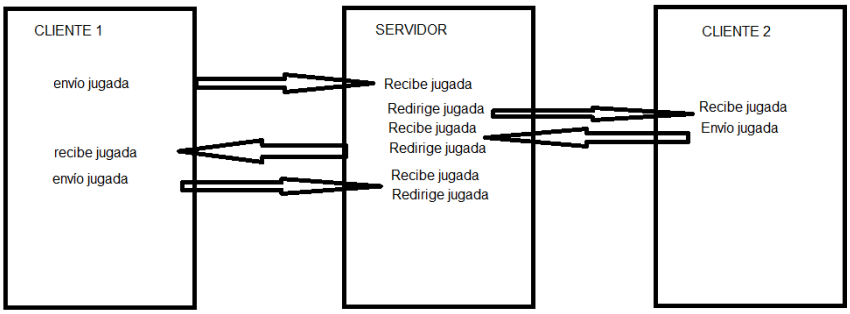


Ilustración 28: Representación del esquema cliente-servidor para una partida de ajedrez

En la imagen podemos ver el esquema de una partida de ajedrez online. La base principal son los sockets. Un socket es un canal de comunicación entre dos aplicaciones. La idea es que, gracias a un lenguaje de alto nivel, como Java, nos podemos abstraer de lo que realmente hacen los routers, y centrarnos en codificar el mensaje que deseamos enviar. La idea es muy similar a un sistema de chat.

Al iniciar una partida, el turno es de sólo un jugador, por lo que el jugador rival deberá esperar a recibir la jugada rival. El programa del lado del servidor se encargará de redirigir los mensajes a sus destinos. Cuando el primer jugador realiza su jugada, se envía al servidor, que hace de intermediario, y la redirige al jugador rival. El rival decodifica en su entorno gráfico dicha jugada, para poder realizar la propia, y se envía de nuevo al servidor que se encargará de conducirla hacia el primer jugador, y así sucesivamente. Al finalizar la partida, quedará registrada por el servidor y se podrán actualizar las puntuaciones.

Las clases en Java que permiten utilizar sockets TCP son la clase Socket y la clase ServerSocket [4].

Tenemos dos tipos de programas.

Uno es el programa servidor, que funcionará en la máquina servidora. Su código de ejemplo es el siguiente [4]:

```
import java.io.*;
import java.net.*;

class TCPServer {
```

```
 public static void main(String argv[]) throws Exception {
  String clientSentence;
  String capitalizedSentence;
  ServerSocket welcomeSocket = new ServerSocket(6789);

  while (true) {
   Socket connectionSocket = welcomeSocket.accept();
   BufferedReader inFromClient =
    new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
   DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
   clientSentence = inFromClient.readLine();
   System.out.println("Received: " + clientSentence);
   capitalizedSentence = clientSentence.toUpperCase() + '\n';
   outToClient.writeBytes(capitalizedSentence);
  }
 }
}
```

Ilustración 29: Código base de la parte del servidor

El segundo tipo de programa es el del lado de los jugadores (clientes). La plantilla para usar sockets TCP del lado del cliente es la siguiente [4]:

```
import java.io.*;
import java.net.*;

class TCPClient {
 public static void main(String argv[]) throws Exception {
  String sentence;
  String modifiedSentence;
  BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
  Socket clientSocket = new Socket("localhost", 6789);
  DataOutputStream outToServer = new
DataOutputStream(clientSocket.getOutputStream());
  BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
  sentence = inFromUser.readLine();
  outToServer.writeBytes(sentence + '\n');
  modifiedSentence = inFromServer.readLine();
  System.out.println("FROM SERVER: " + modifiedSentence);
  clientSocket.close();
 }
}
```

Ilustración 30: Código base de la parte del cliente (jugadores)

De esta manera, disponemos de unos ejemplos de envío de mensajes por TCP, ya que estas clases garantizan que se cumple el protocolo TCP.

Como se puede ver, la máquina servidora está escuchando en un determinado número de puerto. Son los programas cliente quienes necesitan conocer la IP de dicha máquina, y crear un socket para conectarse a esa IP a través del número de puerto acordado. Los sockets son bidireccionales, es decir, sirven tanto para enviar mensajes al servidor, como para recibir mensajes provenientes del servidor, utilizando el mismo socket. Una vez se ha establecido la conexión (el cliente conecta con el servidor), se considera ya

establecida la comunicación, y los routers saben cuál es el origen y el destino de la misma.

En la mayoría de routers actuales, además se requiere de una configuración adicional, que es indicarle al router la IP privada de la máquina donde se va a ejecutar el programa servidor, y a esa entrada se le asocia el número de puerto que elijamos. Con esto, garantizamos que el router va a saber hacer forwarding. Forwarding significa, cuando los paquetes de datos llegan al router, a través de la dirección IP pública de este, en realidad se van a poder redirigir a la máquina concreta que hayamos indicado como IP privada. En concreto, hay que introducir una entrada en el router. Ésta viene dada habitualmente en el siguiente formato:

Protocolo	Número de puerto	IP privada
TCP	XXXXX	XXX.XXX.XXX.XXX

Tabla 77: Configuración Forwarding

Además, necesitamos también conseguir una dirección IP pública para el router que no varíe, es decir, una IP fija, ya que los programas cliente, de los jugadores, tienen que conectar a la misma dirección siempre. Aquí entra en juego el software NoIP [10].

NoIP es un software gratuito que nos da una dirección IP pública adicional a través de un nombre de dominio que elijamos. De esta manera, los sockets del lado del cliente conectarán a través de esa dirección, como si fuese una dirección de una página web.

El principal inconveniente es que se recomienda utilizar un nombre poco conocido e indescifrable, por motivos de seguridad. Además, si no se va a utilizar habitualmente, lo mejor es eliminar la dirección una vez creada. Habitualmente, aunque dispongamos de direcciones IP's dinámicas, en realidad suelen ser fijas la mayor parte de las veces, salvo que se apague y vuelva a encender el router, o cada períodos largos de tiempo (días o meses). Pero de esta manera, en lugar de usar en el código, el número de la IP pública del router, podemos usar este nombre que no cambia nunca y es más manejable.

Ahora tenemos que codificar el mensaje que queremos enviar, que representará una jugada realizada, cuando el rival realice la jugada sobre el tablero. Y además hay que tener en cuenta que al recibir una jugada del rival, ésta se tendrá que descodificar (interpretar) para actualizar el estado del tablero y del tiempo restante.

Podemos codificar el mensaje de la siguiente forma:

Tipo de mensaje	Pieza origen	Pieza destino	Tiempo restante

Tabla 78: Codificación del mensaje

De esta manera, cada campo tiene un tamaño fijo y se puede decodificar sin problemas.

Además, desde la web https://ajedrezynadamasparatodos.herokuapp.com/ se podrá descargar la aplicación. Aunque de momento, estamos en fase de pruebas y momentáneamente, está deshabilitada la opción de descarga.

Éste es el aspecto de la página:

Ilustración 31: Página web de nuestra aplicación

Tecnologías usadas en la página web

- HTML: Es el lenguaje de la parte estática de la página. En ella editamos el texto e incluimos la imagen de la siguiente manera:

<img src="images/chess-316658_640.jpg">

Cabe destacar que es una imagen sin derechos y libre para usar.

- CSS: Es el lenguaje para definir los estilos de la página, tales como el encabezado, el pie, los colores tanto de fondo como de las letras, etc.

Posteriormente, haremos un diseño mejorado, con técnicas de posicionamiento SEO en Google, tales como las siguientes:

- Incluir los términos de consulta en los títulos de la página, y los encabezados.
- Incluir ficheros robots.txt y sitemap.xml
- Incluir más contenido relacionado con el ajedrez.
- Incluir enlaces a otras páginas relacionadas con la temática de ajedrez.

5.8. Diagramas de secuencia

El objetivo de los diagramas de secuencia es reflejar cómo se han de cumplir los requisitos de usuario, en base a todos los escenarios posibles. Estos vienen reflejados en los casos de uso.

De esta manera, se obtiene una representación más visual acerca del funcionamiento de nuestra aplicación, puesto que es lo más importante de comprender, y los usuarios aprenden de forma visual.

En los diagramas de secuencia, hay diferentes elementos importantes, que son los siguientes [25]:

- Actores: aquellos que interactúan con el sistema.
- Lifeline: es una línea vertical que representa la secuencia de eventos de una interacción, mientras el tiempo avanza por la línea.
- Mensajes: aquello que se envía entre los distintos participantes y elementos.
- Interacción: es la colección de mensajes y líneas de vida, reflejados en el diagrama.

En definitiva, los diagramas de secuencia ayudan a obtener los requisitos funcionales, puesto que muestran un detalle mayor acerca de la interacción de los usuarios con el sistema.

Además, visualmente se comprende mejor el funcionamiento, puesto que muchos requisitos necesitan el apoyo visual, de cara a los interesados en el sistema, para negociar su implementación.

Caso de uso 1

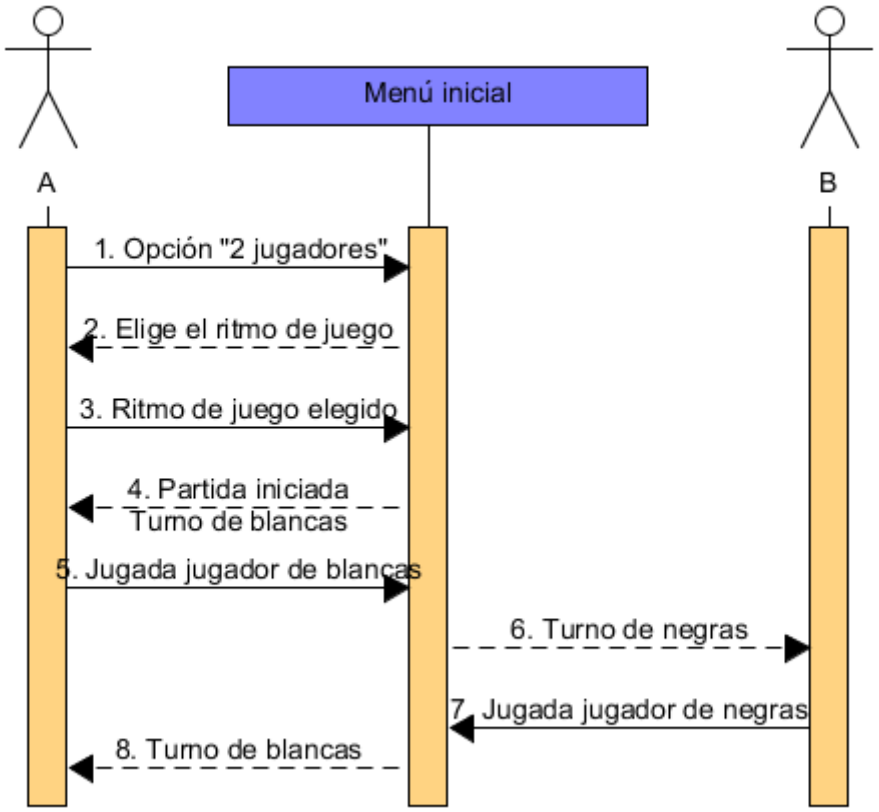


Ilustración 32: Caso de uso 1

Tenemos la funcionalidad más básica, que es jugar una partida en el mismo ordenador, de forma offline, entre dos jugadores. Esto siempre ha de ser posible, puesto que a veces nos interesa analizar incluso de forma individual, una posición determinada, y no deseamos jugar una partida real en ese momento. Por lo que siempre viene bien tener esta opción en nuestro software, aunque sea un software online.

Caso de uso 2 y 3:

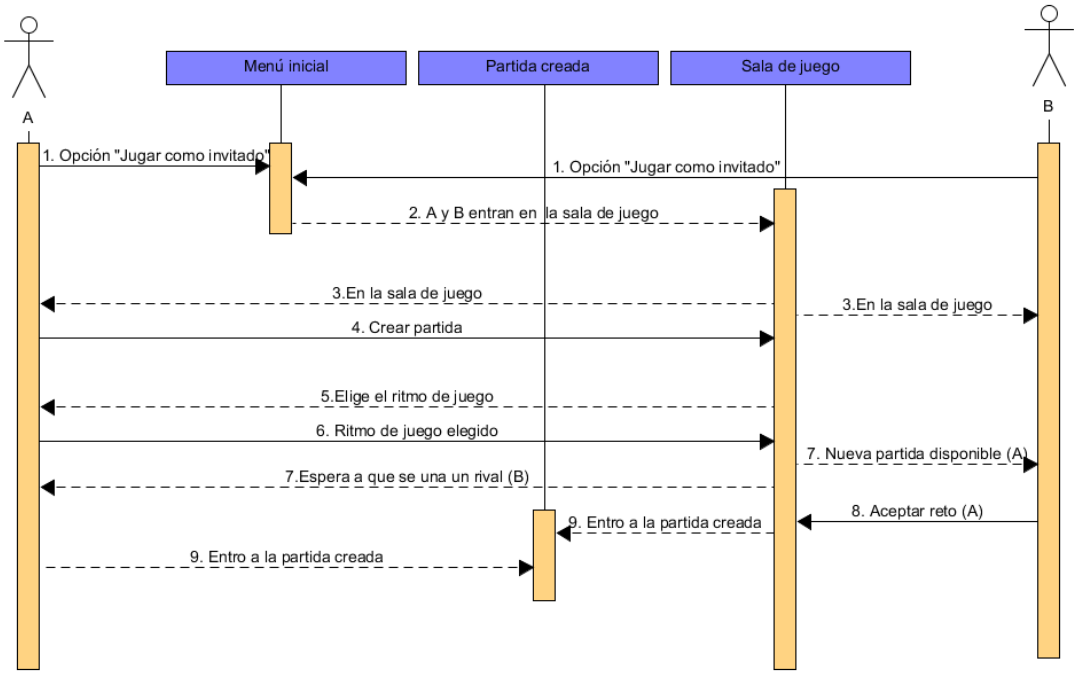


Ilustración 33: Casos de uso 2 y 3

Tenemos dos jugadores que desean disputar una partida, de forma online. Cada uno estará jugando en su propio ordenador, a través de Internet. Por lo tanto, cada uno tendrá su propia vista del tablero, desde la perspectiva en la que juega en ese momento. Para ello, podrán elegir tanto la opción “Jugar como Invitado” como la opción “Iniciar Sesión”, puesto que, para jugar una partida, la diferencia es que la segunda opción permite que la partida quede registrada en su perfil, para actualizar las puntuaciones. Ambos entrarán a la sala de juego y uno de ellos iniciará la partida. El contrincante se unirá a esa partida (aceptar reto).

La razón de agrupar dos casos de uso en uno, es que hay que diferenciar las pruebas para LAN y WAN (red local y red global, respectivamente). La diferencia es mínima, no cambia el esquema gráfico.

Caso de uso 4:

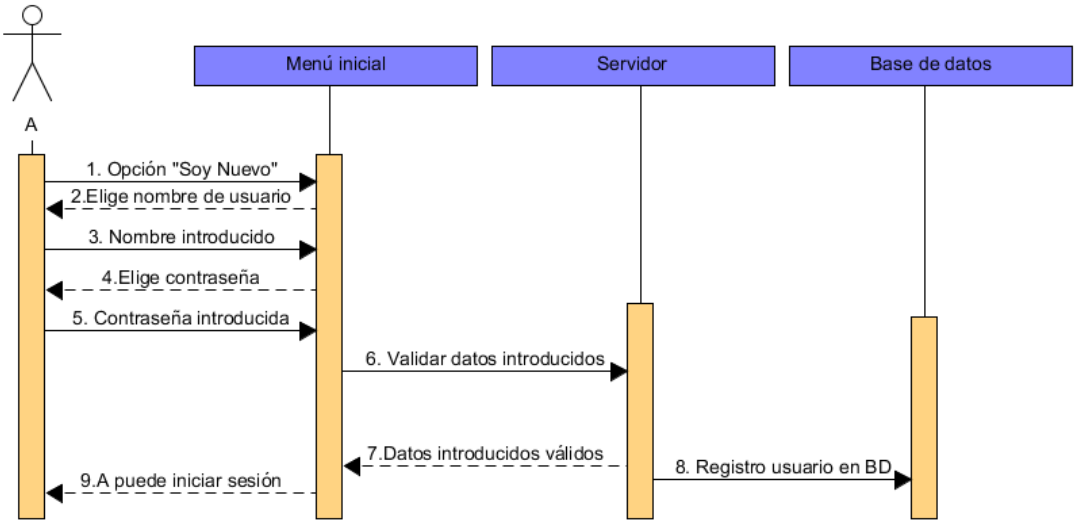


Ilustración 34: Caso de uso 4

La manera de registrarse es muy sencilla. Primero, se elige la opción “Soy nuevo”. A continuación, se introducirá el nombre de usuario y la contraseña. Si son válidos, el usuario podrá iniciar sesión con sus datos.

Caso de uso 5:

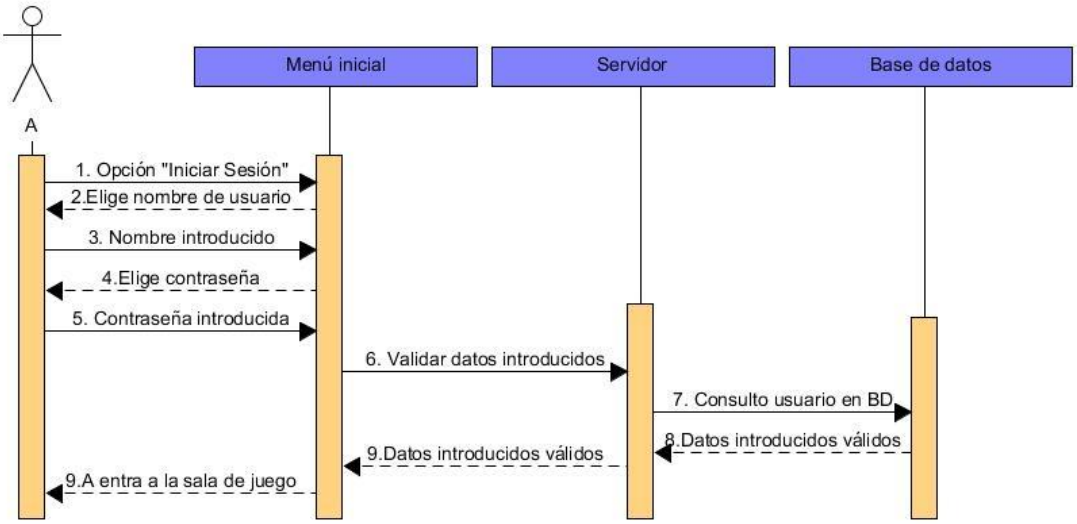


Ilustración 35: Caso de uso 5

Para iniciar sesión hay que estar previamente registrado, y el nombre de usuario y la contraseña han de ser las elegidas en el registro.

Caso de uso 6:

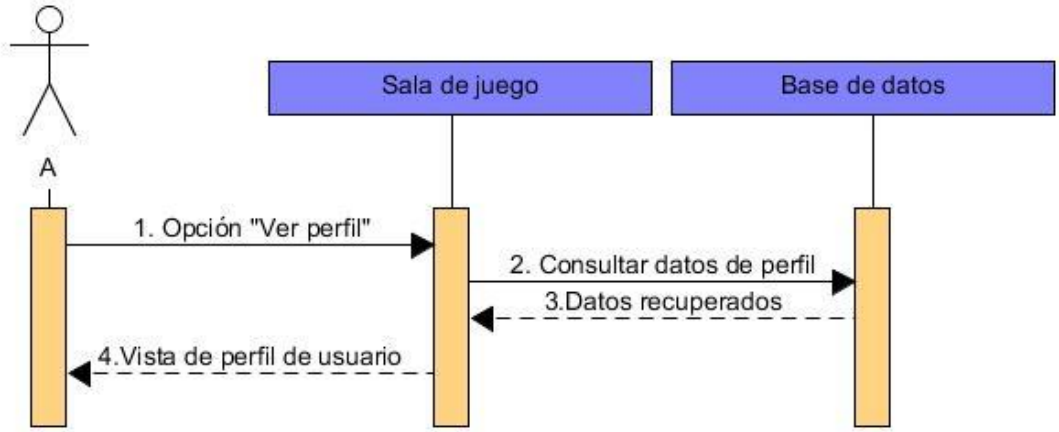


Ilustración 36: Caso de uso 6

El usuario podrá ver su perfil con la opción en la sala de juego “Ver perfil”.

Caso de uso 7:

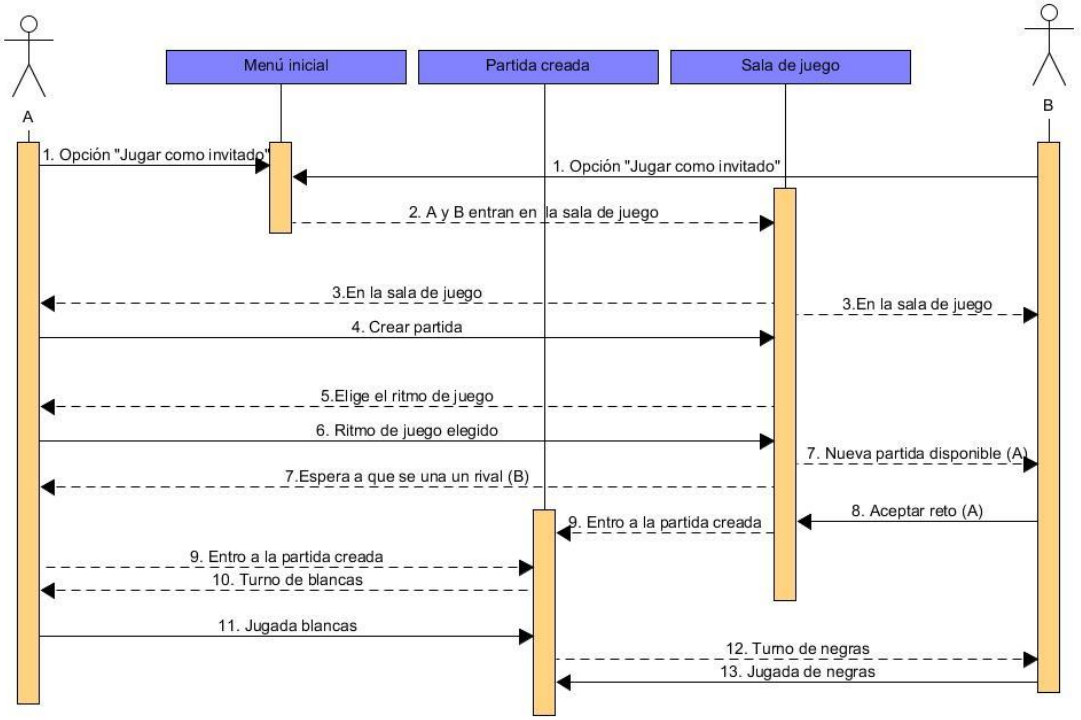


Ilustración 37: Caso de uso 7

En este caso de uso se verifica que ambos jugadores, una vez han entrado a una partida, pueden elegir sus jugadas sin problema, cada uno en el turno que le toque.

5.9. Diseño de las interfaces

En este apartado se va a proceder a diseñar las ventanas gráficas del juego. Aunque es un primer modelo, no tiene por qué ser el definitivo.

Sala de juego

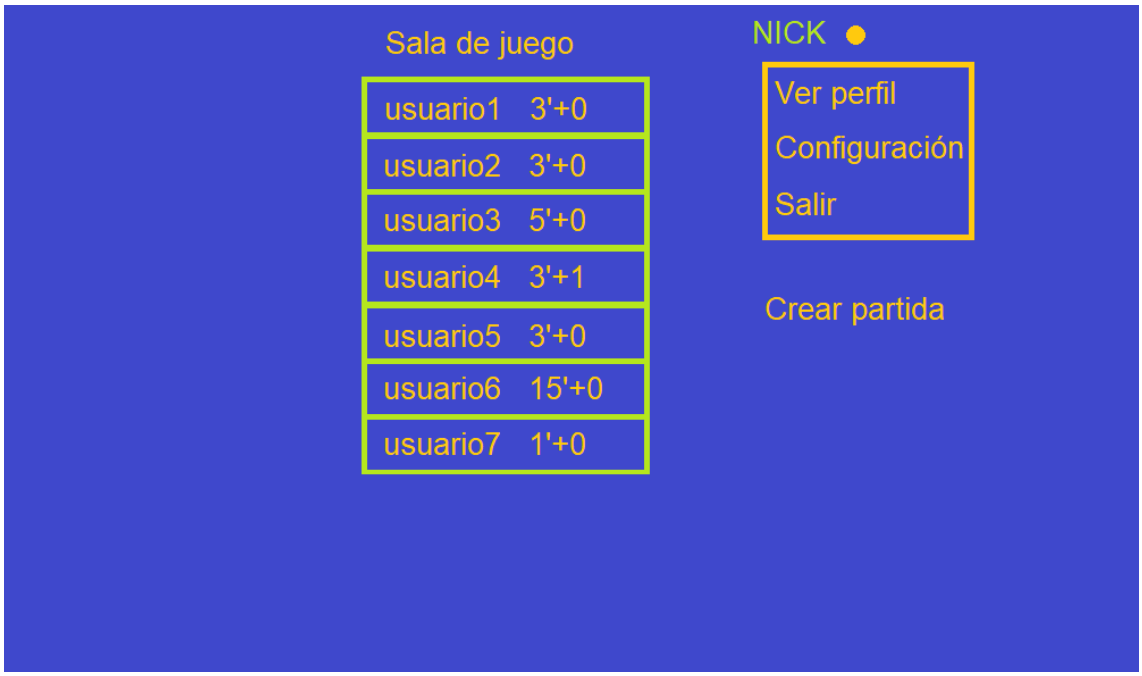


Ilustración 38: Interfaz de la sala de juego

En la sala de juego, irán apareciendo las partidas que los usuarios han creado, para poder unirse a alguna de ellas. Una vez hacemos clic en una de ellas del listado, aparecerá la siguiente ventana, de aceptar reto.

Aceptar reto

Ilustración 39: Interfaz para unirse a una partida creada

Podemos aceptar, o cancelar por si hemos hecho clic accidentalmente.

Ver perfil de usuario

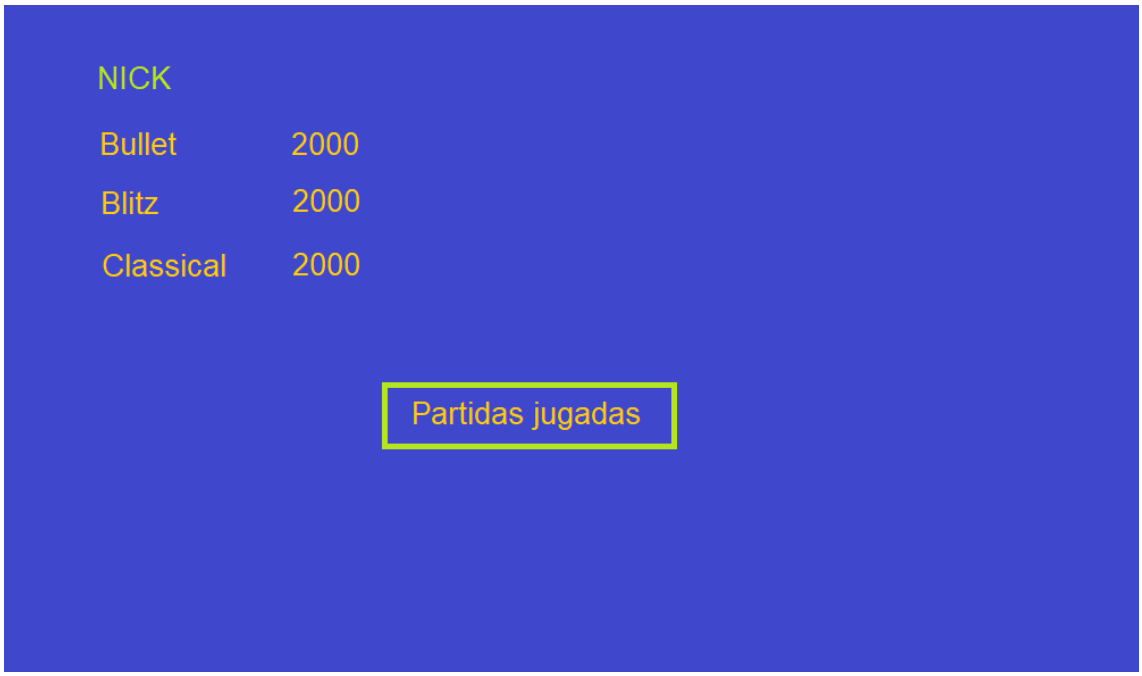


Ilustración 40: Ver perfil de usuario

En esta ventana, el usuario podrá ver su perfil, con las partidas que ha jugado, y sus puntuaciones.

Las puntuaciones siguen el sistema Glicko-2, el cual es un rating, que tiene en cuenta el nivel de los jugadores rivales. Es decir, no es lo mismo ganar a jugadores de rating elevado, que ganar sólo a jugadores de rating más bajo. Esto hace que no sea necesario jugar un número elevado de partidas para obtener un rating fiable.

La teoría de este tipo de puntuación se basa en que habiendo una diferencia de 400 puntos, el resultado esperado es de 10-1 para el jugador con mayor rating. Y en el caso de 800 puntos de diferencia, el resultado esperado sería de 100-1 por lo que obtener ratings más elevados es cada vez más difícil, y se penaliza bastante perder contra jugadores de nivel muy inferior.

5.10. Arquitectura

El esquema de nuestra aplicación es el siguiente.

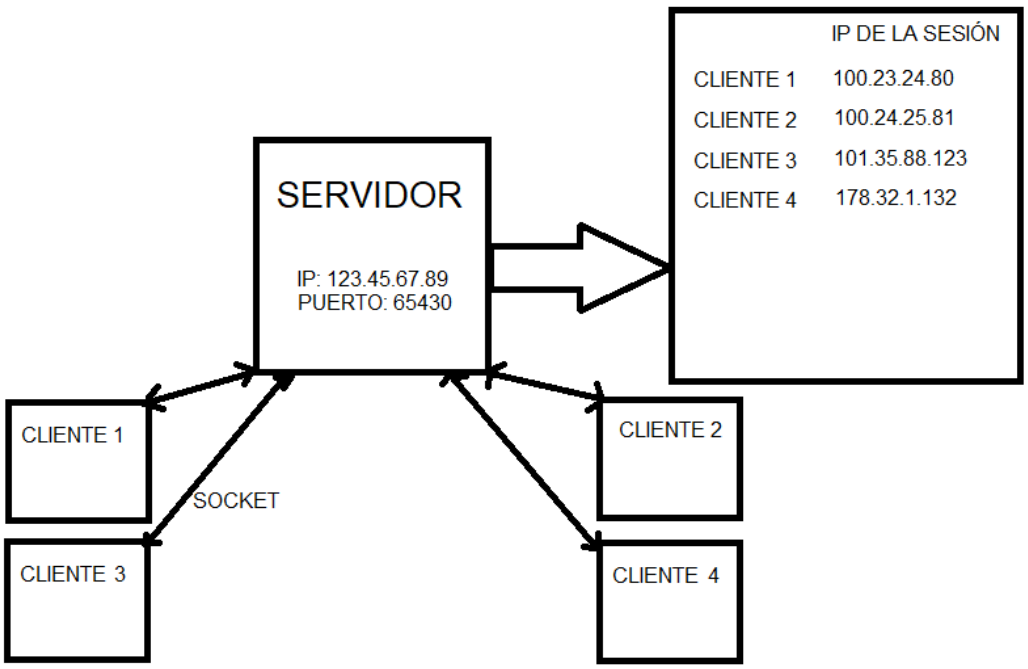


Ilustración 41: Arquitectura del sistema

Se dispone de un servidor que tiene una dirección IP y un puerto, de funcionamiento, a los cuales los clientes (jugadores) se conectan. Una vez el cliente se conecta al servidor, establece un socket bidireccional entre él y el servidor. Las partidas de ajedrez se juegan entre pares de clientes, por lo que habrá un thread o hilo a su vez, por cada partida, de modo que el servidor pueda atender simultáneamente cada jugada realizada, sin tener que esperar un orden determinado entre partidas.

El servidor dispondrá de una estructura de datos, tipo Hashmap, que le permitirá almacenar tanto los sockets como las direcciones IP de los clientes. Esto es eficiente, ya que en todo momento podrá conocer a qué jugador reenviar cada jugada, de forma más inmediata.

6. Pruebas

A continuación, mostraremos las pruebas realizadas para cada requisito. Su objetivo consiste en verificar el correcto funcionamiento de cada requisito, puesto que la fase de pruebas es la más importante en un proyecto de software.

La gran mayoría del tiempo de desarrollo de un proyecto se dedica a probar el software.

ID: Es el identificador de la prueba, comenzando con las siglas PR

Requisito relacionado: es el requisito que se prueba.

Objetivos: la finalidad de la prueba.

Precondición: aquello que se debe cumplir para alcanzar el requisito.

Postcondición: salida adicional de la prueba.

ID	PR-001
REQUISITO RELACIONADO	RF-001
OBJETIVOS	Se comprobará el correcto funcionamiento para una partida que se realice en el mismo ordenador, para dos jugadores. Ambos jugadores pueden mover sus piezas en su turno.
PRECONDICIÓN	Ejecutar el programa
POSTCONDICIÓN	-

Tabla 79: PR-001

ID	PR-002
REQUISITO RELACIONADO	RF-002
OBJETIVOS	Se comprobará el correcto funcionamiento para una partida que se realice entre ordenadores remotos, para dos jugadores. Ambos jugadores pueden mover sus piezas en su turno. Se comprueba que funciona tanto para LAN como para WAN.
PRECONDICIÓN	Cada jugador inicia el programa en su ordenador, y entran a la partida (uno iniciará con piezas blancas y el otro con piezas negras).
POSTCONDICIÓN	En caso de error de conexión aparecerá un mensaje correspondiente.

Tabla 80: PR-002

ID	PR-003
REQUISITO RELACIONADO	RF-003
OBJETIVOS	Se comprobará el correcto funcionamiento para una partida que se realice entre ordenadores remotos, para dos jugadores en la misma subred (LAN).
PRECONDICIÓN	Cada jugador inicia el programa en su ordenador, y entran a la partida (uno iniciará con piezas blancas y el otro con piezas negras, haciendo uso del mismo router).
POSTCONDICIÓN	En caso de error de conexión aparecerá un mensaje correspondiente.

Tabla 81: PR-003

ID	PR-004
REQUISITO RELACIONADO	RF-004
OBJETIVOS	Se comprobará el correcto funcionamiento para una partida que se realice entre ordenadores remotos, para dos jugadores en subredes diferentes (WAN).
PRECONDICIÓN	Cada jugador inicia el programa en su ordenador, y entran a la partida (uno iniciará con piezas blancas y el otro con piezas negras, comprobando que ambos están en subredes diferentes).
POSTCONDICIÓN	En caso de error de conexión aparecerá un mensaje correspondiente.

Tabla 82: PR-004

ID	PR-005
REQUISITO RELACIONADO	RF-005
OBJETIVOS	Se comprobará que el usuario puede registrarse en el sistema, mediante un nombre de usuario y una contraseña. Podrá iniciar sesión y cerrar sesión tantas veces como desee.
PRECONDICIÓN	El usuario elige la opción “Soy Nuevo”
POSTCONDICIÓN	Tras introducir los datos, que son el nombre de usuario y la contraseña, el nuevo usuario puede iniciar sesión.

Tabla 83: PR-005

ID	PR-006
REQUISITO RELACIONADO	RF-006
OBJETIVOS	El jugador podrá iniciar sesión mediante su nombre de usuario y su contraseña.
PRECONDICIÓN	El jugador elige la opción “Iniciar Sesión”
POSTCONDICIÓN	Tras introducir los datos, el usuario aparece logueado en el sistema, en el menú de la sala de juego.

Tabla 84: PR-006

ID	PR-007
REQUISITO RELACIONADO	RF-007
OBJETIVOS	Se verificará que todas las reglas FIDE están programadas, a la hora de jugar una partida.
PRECONDICIÓN	-
POSTCONDICIÓN	-

Tabla 85: PR-007

ID	PR-008
REQUISITO RELACIONADO	RF-008
OBJETIVOS	El jugador puede elegir el ritmo de juego de una partida. De esta manera, entrará en una búsqueda de usuarios que eligen el mismo ritmo de partida. Se comprobará que la partida inicia con el ritmo (tiempo para hacer jugadas) correcto.
PRECONDICIÓN	En el menú de juego, elegir la opción “Iniciar partida” y elegir el ritmo de juego (3 minutos, 5 minutos, 10 minutos, 60 minutos, etc.)
POSTCONDICIÓN	Tras esperar a que un usuario se una a la partida, la partida aparece con el ritmo de juego elegido.

Tabla 86: PR-008

ID	PR-009
REQUISITO RELACIONADO	RF-009
OBJETIVOS	El jugador podrá elegir inscribirse en un torneo de la sala de juego. Se comprobará que aparece en la lista de jugadores.
PRECONDICIÓN	1. Iniciar sesión. 2. Elegir la opción para inscribirse en un torneo. 3. Seleccionar un torneo de la lista de torneos por disputarse, o en juego.
POSTCONDICIÓN	

Tabla 87: PR-009

ID	PR-010
REQUISITO RELACIONADO	RF-010
OBJETIVOS	El jugador puede elegir el estilo de piezas que desee. Elige uno de ellos y se comprueba que las piezas del tablero contienen el diseño elegido.
PRECONDICION	El jugador pulsa el botón que le permite elegir de una lista el estilo de piezas.
POSTCONDICION	-

Tabla 88: PR-010

ID	PR-011
REQUISITO RELACIONADO	RF-011
OBJETIVOS	Se comprobará que el jugador puede elegir el estilo de tablero deseado. Esto incluye el color y el tamaño de las casillas.
PRECONDICION	El jugador elige la opción para elegir el estilo del tablero.
POSTCONDICION	-

Tabla 89: PR-011

ID	PR-012
REQUISITO RELACIONADO	RF-012
OBJETIVOS	Se comprobará que el menú de opciones de juego contiene todas las características necesarias durante la partida. Estas incluyen, poder rendirse, poder abandonar la ventana de partida y poder ofrecer tablas al rival.
PRECONDICION	-
POSTCONDICION	-

Tabla 90: PR-012

ID	PR-013
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover todas las piezas de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón la pieza elegida a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 91: PR-013

ID	PR-014
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover cualquier alfil de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón el alfil elegido a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 92: PR-014

ID	PR-015
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover cualquier caballo de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón el caballo elegido a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 93: PR-015

ID	PR-016
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover cualquier dama de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón la dama elegida a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 94: PR-016

ID	PR-017
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover cualquier peón de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón el peón elegido a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 95: PR-017

ID	PR-018
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover el rey de su color durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón el rey elegido a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 96: PR-018

ID	PR-019
REQUISITO RELACIONADO	RF-013, RF-014, RF-015, RF-016, RF-017, RF-018, RF-019
OBJETIVOS	Se comprobará que el usuario puede mover cualquier torre durante el juego
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Arrastrar con el ratón la torre elegida a la casilla de destino del tablero.
POSTCONDICION	-

Tabla 97: PR-019

ID	PR-020
REQUISITO RELACIONADO	RF-020, RF-007
OBJETIVOS	Se comprobará la verificación de una posición de jaque mate.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Producir una secuencia de jugadas que dan como resultado un jaque mate. Repetir estos pasos para cada combinación posible de color de piezas, tipos de piezas, etc. Es un requisito que se verifica experimentalmente y a lo largo de mucho tiempo, incluyendo muchas combinaciones posibles.
POSTCONDICION	-

Tabla 98: PR-020

ID	PR-021
REQUISITO RELACIONADO	RF-021, RF-007, RF-022, RF-023, RF-024
OBJETIVOS	Se comprobará la verificación de una posición de tablas.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Producir una secuencia de jugadas que dan como resultado tablas. Repetir estos pasos para cada combinación posible de color de piezas, tipos de piezas, etc. Es un requisito que se verifica experimentalmente y a lo largo de mucho tiempo, incluyendo muchas combinaciones posibles.
POSTCONDICION	-

Tabla 99: PR-021

ID	PR-022
REQUISITO RELACIONADO	RF-022, RF-007
OBJETIVOS	Se comprobará la verificación de una posición de tablas por rey ahogado.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Producir una secuencia de jugadas que dan como resultado tablas por rey ahogado. Repetir estos pasos para cada combinación posible de color de piezas, tipos de piezas, etc.
POSTCONDICION	-

Tabla 100: PR-022

ID	PR-023
REQUISITO RELACIONADO	RF-023, RF-007
OBJETIVOS	Se comprobará la verificación de una posición de tablas por material insuficiente.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Producir una secuencia de jugadas que dan como resultado tablas por material insuficiente. Repetir estos pasos para cada combinación posible de color de piezas, tipos de piezas, etc.
POSTCONDICION	-

Tabla 101: PR-023

ID	PR-024
REQUISITO RELACIONADO	RF-024, RF-007
OBJETIVOS	Se comprobará la verificación de una posición de tablas por triple repetición.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Producir una secuencia de jugadas que dan como resultado tablas por triple repetición. Repetir estos pasos para cada combinación posible de color de piezas, tipos de piezas, etc.
POSTCONDICION	-

Tabla 102: PR-024

ID	PR-025
REQUISITO RELACIONADO	RF-025, RF-007
OBJETIVOS	Se comprobará que un jugador puede abandonar la partida/rendirse.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Elegir la opción de rendirse.
POSTCONDICION	-

Tabla 103: PR-025

ID	PR-026
REQUISITO RELACIONADO	RF-026, RF-007
OBJETIVOS	Se comprobará que un jugador puede ofrecer tablas al rival durante una partida.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Elegir la opción de ofrecer tablas.
POSTCONDICION	-

Tabla 104: PR-026

ID	PR-027
REQUISITO RELACIONADO	RF-027, RF-007
OBJETIVOS	Se comprobará que un jugador puede elegir la pieza a coronar durante una partida, cuando un peón llega a la última fila.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Realizar una secuencia de jugadas que permita llevar al peón a la última fila, y elegir la pieza deseada. Repetir el proceso para todas las combinaciones de piezas (Alfil, Caballo, Dama y Torre).
POSTCONDICION	-

Tabla 105: PR-027

ID	PR-028
REQUISITO RELACIONADO	RF-028
OBJETIVOS	Se comprobará que un jugador puede enviar mensajes durante una partida.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Enviar un mensaje de chat y comprobar que el mensaje llega a su destino.
POSTCONDICION	-

Tabla 106: PR-028

ID	PR-029
REQUISITO RELACIONADO	RF-029
OBJETIVOS	Se comprobará que un jugador puede registrarse con un nombre de usuario válido.
PRECONDICION	1. Elegir la opción de registrarse “Soy nuevo”. 2. Elegir un nombre de usuario válido y una contraseña válida. Comprobar para el nombre de usuario que sólo permite aquellos no existentes previamente, con la longitud indicada en el requisito.
POSTCONDICION	-

Tabla 107: PR-029

ID	PR-030
REQUISITO RELACIONADO	RF-030
OBJETIVOS	Se comprobará que un jugador puede registrarse con una contraseña válida.
PRECONDICION	1. Elegir la opción de registrarse “Soy nuevo”. 2. Elegir un nombre de usuario válido y una contraseña válida. Comprobar para la contraseña que cumple con lo indicado en el requisito.
POSTCONDICION	-

Tabla 108: PR-030

ID	PR-031
REQUISITO RELACIONADO	RF-031, RF-007
OBJETIVOS	Se comprobará que un jugador puede aceptar tablas al rival durante una partida.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Elegir la opción de ofrecer tablas por parte del rival. 5. Elegir la opción de aceptar tablas por parte del propio jugador.
POSTCONDICION	-

Tabla 109: PR-031

ID	PR-032
REQUISITO RELACIONADO	RF-032, RF-007
OBJETIVOS	Se comprobará que un jugador puede rechazar tablas al rival durante una partida.
PRECONDICION	1. Haber iniciado sesión 2. Haber elegido iniciar una partida en la sala de juego 3. Haber empezado una partida. 4. Elegir la opción de ofrecer tablas por parte del rival. 5. Elegir la opción de rechazar tablas por parte del propio jugador, o durante su turno, realizar una jugada, ignorando la opción.
POSTCONDICION	-

Tabla 110: PR-032

ID	PR-033
REQUISITO RELACIONADO	RF-033
OBJETIVOS	Se comprobará que un jugador puede ver su perfil de usuario
PRECONDICION	1. Haber iniciado sesión 2. Elegir la opción para ver su perfil de usuario
POSTCONDICION	-

Tabla 111: PR-033

ID	PR-034
REQUISITO RELACIONADO	RF-034
OBJETIVOS	Se comprobará que un usuario puede descargar la aplicación
PRECONDICION	1. Haber entrado a la página de descarga. 2. Elegir la opción de descarga.
POSTCONDICION	-

Tabla 112: PR-034

7. Marco regulador

A continuación, vamos a comentar un aspecto fundamental a la hora de realizar un proyecto software. Las leyes que rigen el buen funcionamiento y el acuerdo entre los clientes y la empresa. Todo software debe estar documentado y firmado tanto por los clientes como por las empresas que venden el producto. Además, en nuestro caso, tenemos que proteger ciertos datos relativos a los usuarios, que ceden información por Internet, muy sensible.

- Ley de protección de datos (LOPD) [19] [20]: Los datos que son recogidos por la aplicación no pueden ser compartidos, y deben ir cifrados, sobre todo los relativos al registro de los usuarios dentro del sistema. Esta versión, al ser una especie de versión demo, no contempla la funcionalidad del cifrado de datos, puesto que está en versión de pruebas, y algunas funcionalidades han de incorporarse antes de poder llevarlas a cabo. Los Artículos 9 y 10 son los que resumen globalmente el tratamiento de estos datos.
  El artículo 9 hace referencia a la seguridad de los datos. Es decir, hay que adoptar las medidas técnicas que preserven la seguridad de la información de terceros. Estos datos no pueden ser alterados, ni manipulados.
  El artículo 10 responde al deber de secreto. Deben conservarse y protegerse del acceso indebido por parte de entidades no autorizadas.

- Marcas registradas. Otro aspecto a tener en cuenta, es que no podemos elegir cualquier nombre a la hora de, por ejemplo, diseñar una página web. Hemos tenido que elegir un nombre poco deseable, para no arriesgarnos a caer en trampas legales.

8. Entorno Socio-Económico

Vamos a ver de forma resumida, y después, de forma más específica, los costes del proyecto, en todas sus partidas.

Resumen de costes:

- Personal: 1 persona
- Coste/tiempo: 12 ECTS: 12*25 horas/ECTS=300 horas

Transporte:

- Tren: 72 €/mes*6 meses=432€

Servidores:

- Usamos un servidor gratuito.

Ordenadores:

- Ordenador propio: 2 ordenadores*400 €=800 €

Conexión a Internet:

- 30 €/mes*6 meses=180 €

Ahora vamos a analizar cada partida por separado, y calcularemos el total de costes.

Costes de personal:

Personal	Rol	Precio/hora (€)	Número de horas totales	Coste total (€)
Álvaro Sánchez	Jefe de proyecto	45	300	13500
Álvaro Sánchez	Viabilidad del sistema y Análisis	30	75	2250
Álvaro Sánchez	Diseñador	30	55	1650
Álvaro Sánchez	Programador	20	100	2000
Álvaro Sánchez	Pruebas	25	70	1750
Total			300	21150

Tabla 113: Costes de personal

Equipos y hardware:

Ordenadores:

- Ordenador propio: 2 ordenadores*400 €=800 €

Nombre	Tiempo dedicado en meses	Período de amortización en meses	Coste sin IVA (€)	Porcentaje de uso en el proyecto	Coste que se imputa (€)
Ordenador Sobremesa	6	48	400	50	25
Ordenador Portátil	6	48	400	50	25

Tabla 114: Costes de equipos

Total: 25+25=50 €

Se ha aplicado la amortización mediante la siguiente fórmula: A/BxCxD, donde A es el tiempo dedicado en meses, B es el período de amortización también en meses, C es el coste sin IVA, y D es el porcentaje de uso en el proyecto (en decimal. Si es 50% se toma 0.5).

Fijos: son los gastos que no varían a lo largo del proyecto. Como la duración del proyecto es de 6 meses, en los cuales se ha necesitado la conexión a Internet, estos son los gastos incluídos.

Conexión a Internet:

- 30 €/mes x 6 meses=180 €

Electricidad:

- 100 €/mes x 6 meses=600 €

Fungible: son los gastos relativos al material desechable, como pueden ser los bolígrafos y el papel.

Nombre	Descripción	Coste/mes (€)
Bolígrafos	Necesario para realizar los diseños, previos a la implementación	3
Papel	Necesario para realizar los diseños, previos a la implementación	2
Pilas	Necesarias para los ratones inalámbricos	3
	Total	8 x 6 meses = 48

Tabla 115: Costes de material fungible

Viajes y dietas: En este apartado debemos contemplar las sesiones de reunión con nuestro tutor, ya que también necesitamos algunos de sus equipos de oficina para realizar las pruebas pertinentes del funcionamiento del software.

Nombre	Descripción	Coste/mes (€)
Abono de transporte	Abono mensual de un adulto	72
	Total	72 x 6 meses = 432

Tabla 116: Viajes y dietas

Costes indirectos: aquellos relativos a posibles imprevistos durante el desarrollo del proyecto. Podemos considerar que se aplica un porcentaje al coste del personal. En nuestro caso, será de un 10%.

Personal	Coste total (€)	Porcentaje (%)	Coste indirecto
Álvaro Sánchez Rodríguez	21150	10	2115

Tabla 117: Costes indirectos

Por lo tanto, los costes totales sin riesgos son de 21150+50+180+600+48+432+2115=**24575 €**

Riesgo: se aplica un porcentaje sobre los costes totales. En este caso, dada la dificultad del proyecto, debemos asumir un riesgo de un 25%.

Costes totales sin riesgo	Riesgo (%)	Total de costes con riesgo (€)
24575	25	**30718.75**

Tabla 118: Costes totales con riesgo

Presupuesto

Calculamos el presupuesto necesario a partir del beneficio esperado y el IVA.

Si queremos un beneficio del 30 % con un IVA del 21% obtenemos los siguientes resultados:

Beneficio: 0.3 x 30718.75 = 9215.63 €

IVA: 0.21 x 30718.75 = 6450.94 €

Total de presupuesto: 30718.75 + 9215.63 + 6450.94 = 46385.32 **€**

Cuarenta y seis mil trescientos ochenta y cinco con treintaidós euros.

9. Conclusiones y líneas futuras

9.1. Conclusiones

En general, ha sido un placer haber desarrollado un modelo cliente-servidor para un juego como es el ajedrez. Debido a que nunca antes había programado una aplicación que funcione a través de Internet, y para los desarrolladores de videojuegos, en general, siempre es interesante, ya que casi nunca se hace hincapié en cómo se puede crear un videojuego que funcione de forma online. La mayoría de tutoriales explican la forma de desarrollar un juego de forma local. Esto parece interesante; sin embargo, si queremos llegar a un público mayor, es necesario poder tener a disposición unas bases de conocimiento que nos permitan ir más allá.

La principal dificultad encontrada es la de obtener un servidor gratuito, de forma que no necesitemos habilitar ningún puerto en nuestro router, para poder disponer de una máquina remota que ejecute nuestro programa servidor. Sin embargo, nos dimos cuenta que se podía utilizar node.js, ya que ningún servicio ofrecía ejecutar un servidor.jar directamente. Más adelante vimos cómo se podía utilizar con el servidor Heroku, con el envío de mensajes HTTP a través de la url de la página, mediante GET. Este es un gran paso para poder sustituir los sockets TCP por sockets HTTP y evitar la limitación mencionada, aunque para mostrar la funcionalidad del trabajo de fin de grado, no lo hemos considerado relevante, pero sí para un futuro.

Creo que el Trabajo de Fin de Grado me ha ayudado bastante a la hora de planificar un proyecto, y las reuniones con el tutor siempre han sido provechosas.

La mayor dificultad a la hora de llevar el proyecto a dimensiones mayores, es el hecho de encontrar un servidor gratuito, sin limitaciones, y que ejecutase programas .java en remoto. Esto no lo hemos encontrado, y aunque se ha investigado y encontrado la manera de conseguir la misma funcionalidad, sin tener que depender de un programa servidor hecho en java, ya no había tiempo para readaptarlo todo, aunque para un futuro es una buena idea. Puesto que los servidores gratuitos tenían muchas limitaciones de uso, y los de pago tampoco daban garantías de que nuestro sistema se adaptara a ese servidor en concreto.

Aun así, pese a las dificultades encontradas, me he visto obligado a aprender más tecnologías, lo cual es siempre muy positivo, puesto que ahora dispongo de alternativas que antes no se me habrían ocurrido, para un sinfín de nuevas necesidades que se me pueden presentar en cualquier tipo de proyecto.

Por lo tanto, siempre el esfuerzo ofrece recompensas, a veces inesperadas.

Estoy muy agradecido con aquellas personas que me han ayudado a lo largo del proyecto, en especial, mi tutor, Israel González Carrasco, y todas aquellas personas que siempre han creído sin dudar que iba a sacar todo adelante.

En general, ha sido apasionante probar a ejecutar un programa servidor desde casa, ir a la universidad, probar a jugar con mi tutor una partida, cada uno en su ordenador, y ver

que ambos programas cliente se comunicaban entre sí, las jugadas que nos enviábamos entre nosotros, y sin ningún retardo apreciable. La verdad es que fue algo impactante.

9.2. Líneas futuras

Como se explicó anteriormente, este es el inicio de un proyecto de un alcance todavía mayor. Hay numerosas formas de ampliar las funcionalidades de este software. Entre ellas están las siguientes:

Foro para opiniones, debates, mejoras…
- Lista de amigos
- Bloqueo de usuarios
- Entrenamiento (táctica, estrategia, vídeos…)
- Verificar título FIDE
- Ver otras partidas como espectador
- Tener perfil de entrenador(en caso de ser entrenador)
- Cuenta de pago (ampliar material, vídeos, contacto con jugadores de alto nivel…)
- Jugar en tablero físico y que mediante reconocimiento por realidad aumentada, se envíe la jugada al rival
- Posibilidad de hacer vídeos en directo mientras se juega, desde la propia aplicación
- Diferentes variantes de ajedrez (normal, chess960, horde, King of the hill…)

Además, tenemos pensado utilizar la tecnología node.js [7], para mejorar el uso del lado del servidor de la aplicación.

Y sin duda, la funcionalidad que me habría gustado desarrollar, y que es seguro que así va a ser en un futuro cercano, es la de poder jugar a través de Internet partidas mediante realidad aumentada. Esto significa, poder hacer las jugadas en un tablero de ajedrez físico, y con ayuda de una cámara, reconocer la jugada y enviarla al contrincante. Es algo novedoso, y serviría tanto para jugar entre dos personas en el mismo tablero, como para jugar a través de Internet.

9.3. Reconocimiento del tablero a través de la cámara por realidad aumentada

El planteamiento para poder jugar mediante realidad aumentada es el siguiente:

1. Preparamos el tablero de forma que cada casilla tenga un círculo con un color determinado. Esto ayuda a la cámara a reconocer el tablero y la posición actual. El proceso se muestra en la siguiente imagen.

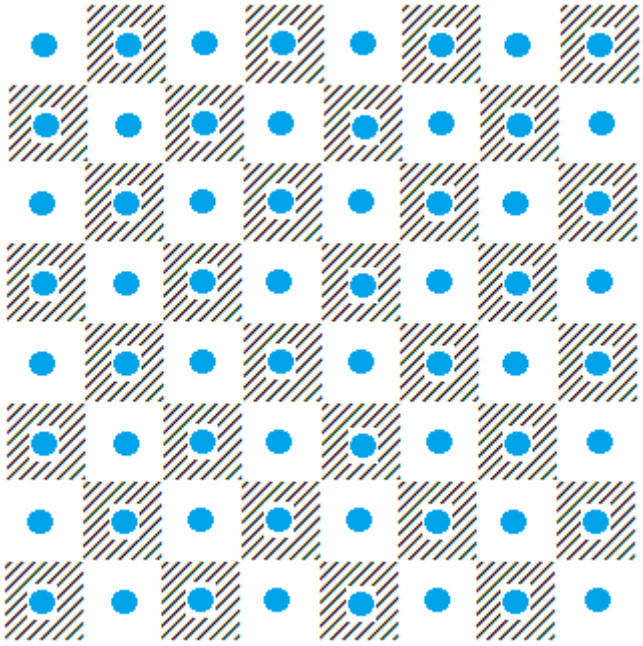


Ilustración 42: Preparación para el reconocimiento del tablero a través de una cámara

2. Como la posición inicial es siempre la misma, no se necesita reconocer ninguna pieza concreta, porque siempre inician en la misma posición. Y cuando una pieza mueve, deja libre una casilla. Esto hace que al dejar libre una casilla y ocupar otra, automáticamente se puede saber qué pieza ha movido, puesto que se dejará libre una casilla (aparecerá el punto azul) y se ocupará otra (se tapará el punto azul de la casilla destino).
3. En caso de que en el punto 2 en lugar de mover sin más una pieza, la jugada sea la captura de otra pieza enemiga, lo que se puede comprobar es que una pieza desaparece del tablero y es ocupada por otra en esa misma posición pero con color opuesto (lo cual es fácil de reconocer).
4. Solamente hay que indicar la pieza de coronación de un peón que llega a la última fila y se va a sustituir por otra pieza. Aunque la mayoría de los casos, por defecto va a ser una dama (reina).

9.4. Detección de tramposos en ajedrez online mediante clustering

Esta es otra de las posibles ampliaciones que se pueden realizar, para mejorar la experiencia de usuario en juegos online, sobre todo en ajedrez. Puesto que es muy desagradable saber que se está jugando una partida contra alguien que utiliza alguna "ayudita" extra.

Este tipo de situaciones se pueden evitar gracias a técnicas de Inteligencia Artificial. Una de ellas se denomina clustering o agrupamiento. La idea consiste en encontrar grupos de comportamiento entre los jugadores, en base a los valores de determinados atributos. Estos hay que seleccionarlos muy bien. Entre ellos, pueden elegirse los siguientes:

- Historial de partidas jugadas.
- Historial de tiempos de respuesta por posición o jugada realizada.
- Histórico de puntuaciones.
- Historial del progreso obtenido en las puntuaciones.
- Porcentaje de veces que se mueve una determinada pieza.
- Tiempos de respuesta ante posiciones complejas. Por ejemplo, muchas piezas atacadas entre sí, o al contrario, posiciones sencillas.
- Ritmos de juego empleados. Es más probable hacer trampas en ritmos de juego lentos, que en ritmos rápidos.
- Jugadas elegidas en apuros de tiempo. Comparar con módulos de ajedrez que analizan los aciertos y errores del jugador.
- Precisión de las jugadas cuando el jugador tiene bastante tiempo para jugar.
- Precisión de las jugadas en apuros de tiempo.
- Cualquier otro atributo que sea relevante.

Preparando bien estos datos, podemos elegir con softwares como Weka, técnicas de agrupamiento, como el K-medias. A partir de ahí, podemos obtener grupos de comportamiento que se salgan de lo común. Y sabiendo a cuál grupo pertenece uno de los tramposos, podemos obtener el resto.

10. Anexo

10.1. Manual de usuario

Una vez descargada la aplicación aparecerá la siguiente ventana al ejecutar la aplicación .jar alojada en la web

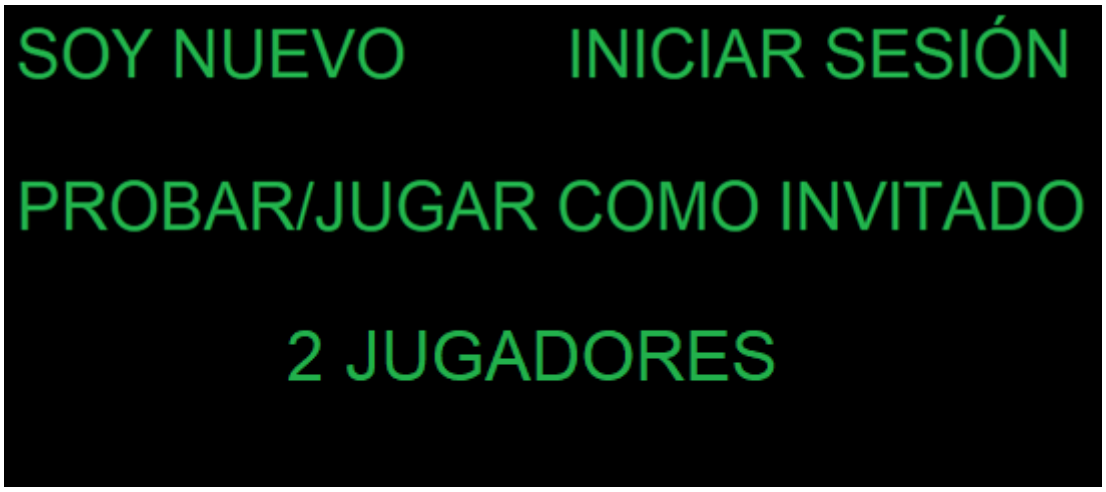


Ilustración 43: Menú inicial

Tenemos cuatro opciones. Podemos registrarnos, en la opción “Soy nuevo”. También podemos iniciar sesión si ya estamos registrados. La tercera opción es jugar como invitado, y de esta manera podemos jugar partidas sin registro. Y por último podemos jugar sin necesidad de conexión a Internet, con la opción “2 jugadores”, con un amigo en el mismo ordenador.

Éste es el menú de registro:

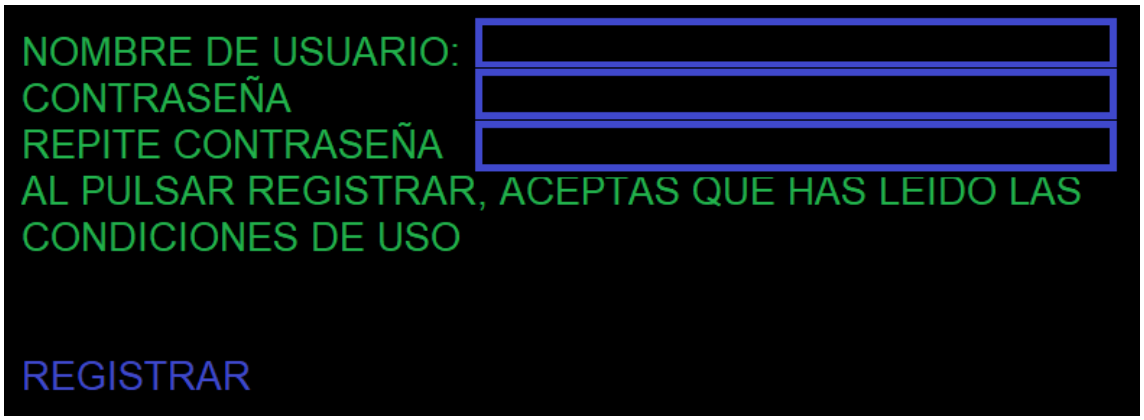


Ilustración 44: Menú de registro del jugador

Tanto si se ha iniciado sesión como si se ha elegido la opción de probar como invitado, aparecerá una sala de juego.

La apariencia de una partida es la siguiente:

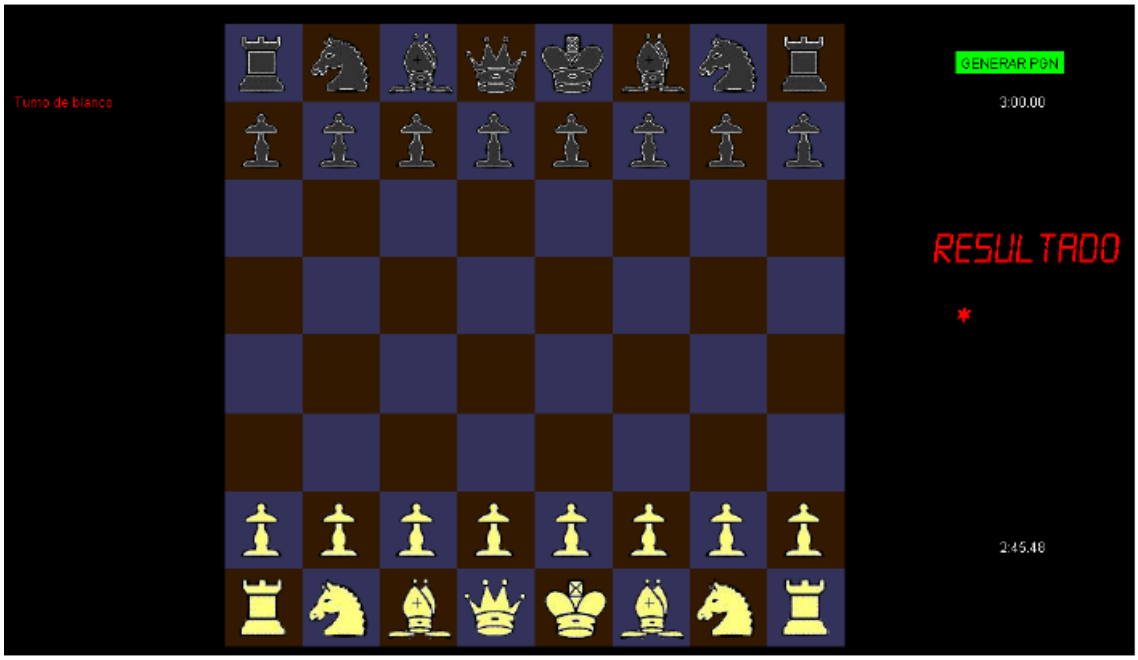


Ilustración 45: Partida en curso

Inicialmente, en la parte inferior disponemos de la perspectiva de nuestras piezas, un reloj a la derecha que nos muestra los minutos, segundos y centésimas de segundo restantes, y a la izquierda se muestra el turno del jugador. Cuando un jugador da jaque mate aparece algo como lo siguiente:

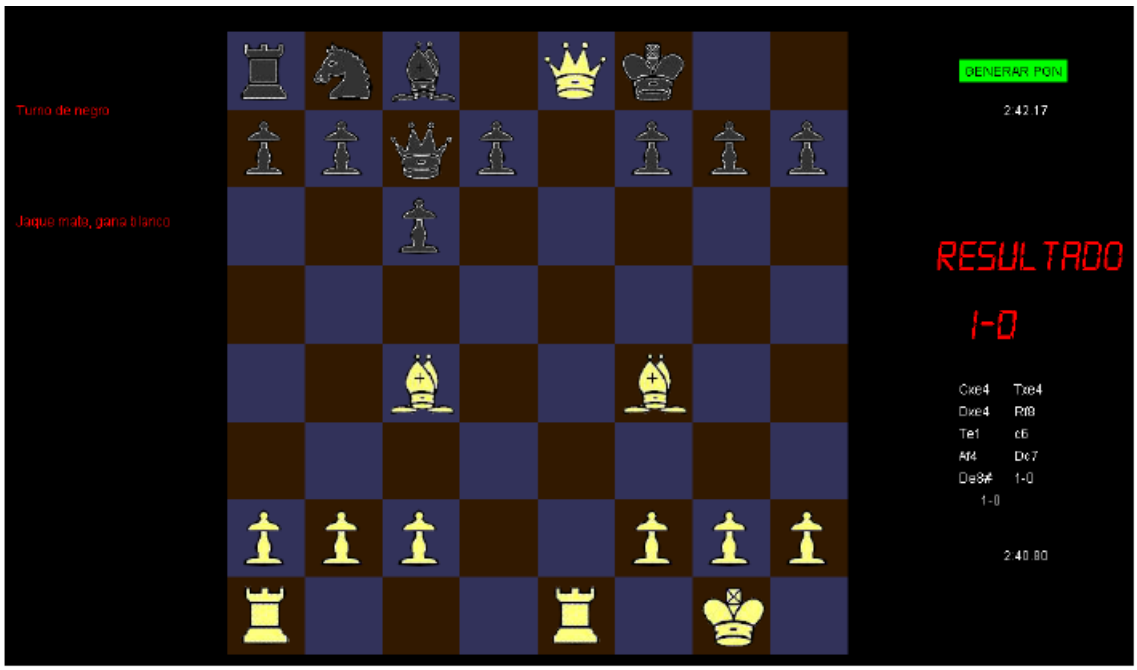


Ilustración 46: Partida finalizada (I)

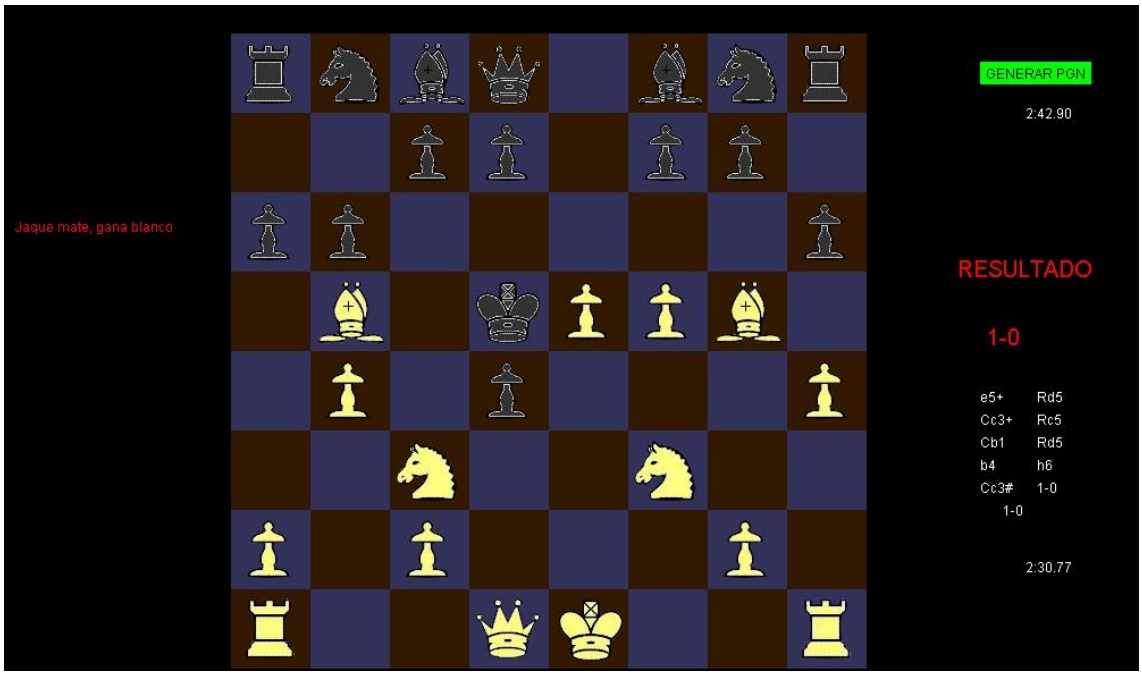


Ilustración 47: Partida finalizada(II)

Aquí podemos ver además que, al realizar las jugadas, a la derecha aparecen reflejadas, en notación algebraica, en la parte derecha de la imagen.

Al jugar como invitado o iniciar sesión, tenemos la siguiente interfaz:

Ilustración 48: Manual: Sala de juego

Y para aceptar retos y unirse a las partidas, tenemos la siguiente ventana:

Ilustración 49: Manual: Aceptar reto

11. Referencias

[1] Reglas FIDE http://feda.org/feda2k16/wp-content/uploads/LEYES-DEL-AJEDREZ-DE-LA-FIDE-2017-bilingue-finales.pdf Junio 2018

[2] Notación algebraica http://ajedrezutea.com/formato-pgn-notacion-pgn-ajedrez Junio 2018

[3] PGN http://portablegamenotation.com/LittlePGN.html Junio 2018

[4] Ejemplo en Java de cliente servidor mediante el protocolo TCP https://systembash.com/a-simple-java-tcp-server-and-tcp-client/ Junio 2018

[5] Mensajes http en Java https://stackoverflow.com/questions/1359689/how-to-send-http-request-in-java Junio 2018

[6] Heroku https://www.heroku.com/ Junio 2018

[7] Node https://nodejs.org/es/ Junio 2018

[8] Eclipse https://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/marsr Junio 2018

[9] Aplicación cliente-servidor http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/cliente-servidor.html Junio 2018

[10] NoIP https://www.noip.com/ Junio 2018

[11] Lista de puertos inseguros https://www.adslzone.net/lista-de-puertos-troyanos.html Junio 2018

[12] lichess.org https://lichess.org/ Junio 2018

[13] chess.com https://www.chess.com/ Junio 2018

[14] Internet Chess Club https://www.chessclub.com/ Junio 2018

[15] Servidor http://www.masadelante.com/faqs/servidor Junio 2018

[16] Cliente https://sistemas.com/cliente.php Junio 2018

[17] Protocolo TCP https://es.ccm.net/contents/281-protocolo-tcp Junio 2018

[18] Protocolo HTTP http://neo.lcc.uma.es/evirtual/cdd/tutorial/aplicacion/http.html Junio 2018

[19] LOPD (1) https://www.boe.es/legislacion/codigos/codigo.php?id=055_Proteccion_de_Datos_de_Caracter_Personal&modo=1 Junio 2018

[20] LOPD (2) https://www.boe.es/buscar/act.php?id=BOE-A-1999-23750 Junio 2018

[21] NAT https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/26704-nat-faq-00.html Junio 2018

[22] “Lichess github” https://github.com/ornicar/lila Junio 2018

[23] LAN vs WAN http://fdryc.blogspot.com/2012/02/diferencia-entre-wlan-lan-y-wan.html

Junio 2018

[24] Chess Diagrams http://www.enpassant.dk/chess/palview/manual/sets.htm Junio 2018

[25] Diagramas de secuencia https://msdn.microsoft.com/es-es/library/dd409377.aspx Junio 2018

[26] NAT (2) https://www.1and1.es/digitalguide/servidores/know-how/nat-asi-funciona-la-traduccion-de-direcciones-de-red/ Junio 2018