TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

Aplicació web d'escacs

Memòria

Artur Miralles Hernàndez
Tutor: Catalina Juan Nadal

Curs 2021-2022

TecnoCampus
Mataró-Maresme

Abstract

This project aims to develop a web application to play chess, either against the computer or online against other people. The user can climb the ranking, communicate and share information with other users, challenge his friends to a game, customize his board and many more features. The web incorporates a voice controller to play the game by dictating your moves.

Resum

Aquest projecte té com a objectiu el desenvolupament d'una aplicació web per a poder jugar a escacs, ja sigui contra l'ordinador o de forma online contra altres persones. L'usuari pot ascendir en la classificació, comunicar-se i compartir informació amb els altres usuaris, desafiar als seus amics a una partida, personalitzar el seu taulell i moltes més funcionalitats. La web incorpora un controlador per veu per poder jugar la partida dictant els moviments.

Resumen

Este proyecto tiene como objetivo el desarrollo de una aplicación web para poder jugar al ajedrez, ya sea contra el ordenador o de forma online contra otras personas. El usuario puede ascender en la clasificación, comunicarse y compartir información con los otros usuarios, desafiar a sus amigos a una partida, personalizar su tablero y muchas más funcionalidades. La web incorpora un controlador por voz para poder jugar la partida dictando sus movimientos.

Índex

Índex de figures

Índex de taules

1 Introducció

El projecte consisteix en la creació d'una web per jugar a escacs.

El producte se centra en dos temes:

1. Controlador per veu: facilitar un sistema de control per veu amb el que l'usuari pugui indicar els moviments a realitzar simplement parlant.

   Hi ha persones que, per la seva discapacitat, no poden jugar a escacs ni fer moltes altres coses.

   Poder jugar a escacs amb la veu els permet entretenir-se i alhora aprofitar-se dels molts beneficis dels escacs per al desenvolupament d'habilitats de raonament, memòria, creativitat i anàlisi.

   Alguns exemples en els quals és útil el control per veu són:

   - Tetraplegia.
   - Discapacitat visual.
   - Persones que no necessàriament tenen una discapacitat però que volen practicar de jugar als escacs a cegues.

2. Modernització: Hi ha molts jocs d'escacs, la gran majoria amb el mateix sistema de puntuació, les mateixes peces i els mateixos modes de joc.

   El producte s'enfoca en un públic més casual per tal d'iniciar en el món dels escacs a aquelles persones que no juguen ja sigui perquè els sembla massa difícil o perquè no tenen la motivació.

2 Marc teòric

2.1 Context

2.1.1 Fonaments del joc

Els escacs són un joc de tauler per a dos jugadors. Cada jugador té 16 peces disposades en un taulell de vuit per vuit caselles.

Fig. 2.1.1 Taulell d'escacs amb les peces a la seva posició inicial. Font: Elaboració pròpia.

És un joc d'estratègia i un dels jocs més populars del món. Considerat per molts el joc amb una posició més important en la història [1].

L'objectiu del joc és fer escac i mat al rei contrari seguint una sèrie de regles [2].

Actualment els escacs es poden jugar en línia des d'internet: a través de diferents pàgines webs [3], [4] milers d'usuaris de diferents parts del món poden competir entre ells diàriament.

2.1.2 Beneficis

Els escacs no només prenen importància en l'entreteniment o en l'àmbit competitiu, sinó que també tenen gran impacte en l'educació i la salut.

Hi ha multitud d'estudis que tracten sobre els beneficis dels escacs sobretot en nens i joves però també en adults.

Per exemple, en l'article "Cognitive Benefits of Chess Training in Novice Children" [5] es mostren dades d'un experiment en el que s'avaluen les habilitats cognitives dels nens de dos grups diferents, en un dels grups es realitzen classes d'escacs i en l'altre activitats matemàtiques. Estudiant els resultats abans de les classes i després dels diferents grups s'arriba a la conclusió de que la millora en els resultats del grup d'escacs ha estat major.

Un altre exemple semblant es pot trobar en "The Benefits of Chess for the Intellectual and Social-Emotional Enrichment in Schoolchildren" [6] on també es comparen els resultats en diversos grups, un dels quals participa en activitats d'escacs. I els resultats indiquen una millora en aquest grup tant en les capacitats cognitives com en el la resolució de problemes i el desenvolupament sociopersonal.

2.2 Antecedents

El joc d'escacs és molt popular i com a conseqüència també hi ha moltes webs ja enfocades a aquest joc.

A continuació s'analitzen les webs d'escacs més conegudes i també algunes de més semblants al producte objectiu d'aquest projecte.

2.2.1 Webs més populars

- Chess.com [4]

La web més popular d'escacs en l'actualitat.

Positiu:

- Molts usuaris.
- Col·laboracions amb jugadors de talla mundial.
- Lliçons i explicacions de diferents temes.

- Puzles.
- Anàlisi de partides.

Negatiu:

- Moltes de les característiques (anàlisi, lliçons, etc...) estan limitades o restringides als usuaris prèmium (de pagament).
- Publicitat.

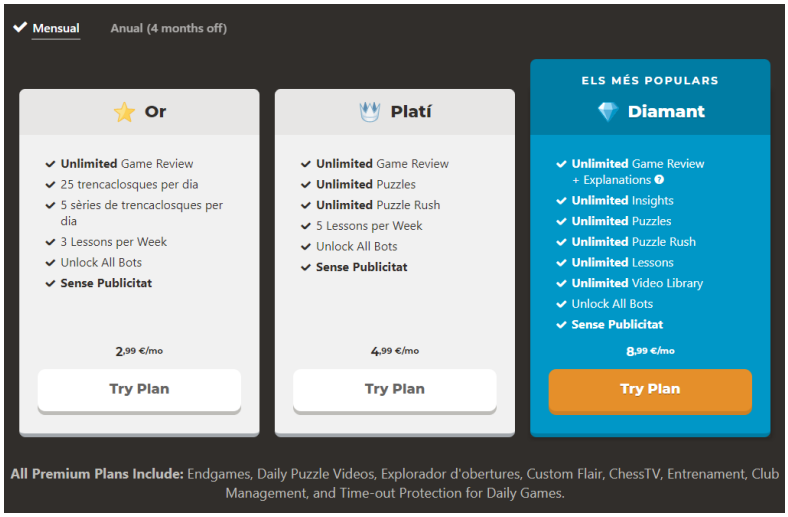

Fig. 2.2.1.1 Plans de pagament de chess.com. . Font: Chess.com

- Lichess [3]

La segona web més popular i amb una evolució molt positiva en els últims anys.

Positiu:

- Puzles.
- Anàlisi de partides.
- Completament gratis.
- No té publicitat.
- Bona interfície d'usuari.

Negatiu:

- No té tantes dades de partides.
- No hi ha tants jugadors de talla mundial.

Com a finançament s'utilitza un model de donacions:

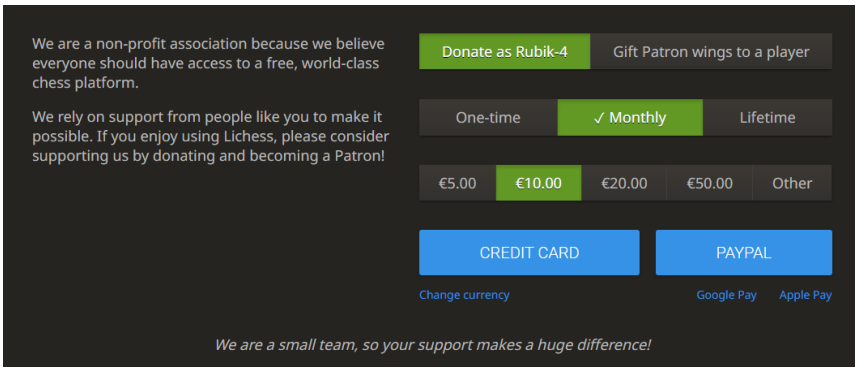

Fig. 2.2.1.2 Interfície per realitzar una donació a Lichess. Font: Lichess.org

2.2.2 Webs amb control per veu

- LChess [7]

Negatiu:

- Anuncis.
- Interfície d'usuari confusa.
- No utilitza la notació normal dels escacs sinó que utilitza números, la qual cosa també dificulta el correcte reconeixement per veu.

- Speak to Lichess

És una extensió de Google Chrome que s'incorpora a la web de Lichess i permet dictar els moviments.

Positiu:

- Fa de complement per una altra web, per tant té tots els avantatges que Lichess ofereix.

- Per facilitar el reconeixement de veu es poden utilitzar paraules: d3c4 → delta 3 charlie 4.

Negatiu:

- És completament dependent de Lichess, això fa que no depèn d'ella mateixa sinó que és sensible als canvis d'una web externa, un petit canvi d'aquesta pot fer que l'extensió deixi de funcionar.
- L'usuari ha de generar un token amb permisos a l'API del seu compte de Lichess i donar aquests permisos a l'extensió.
- Només funciona per Chrome.

2.2.3 Reflexió

Sorprenentment, no hi ha pràcticament webs per poder jugar a escacs amb control per veu. Encara menys amb multijugador online o amb un idioma que no sigui l'anglès.

Per tant, és positiu oferir una nova alternativa amb diferents funcionalitats.

Pel que fa al sistema de puntuació, hi ha webs que directament no en tenen o d'altres on s'utilitza un valor numèric que incrementa en les victòries i disminueix en les derrotes. Pot ser interessant provar nous sistemes.

Com ja es podia intuir, hi ha una gran varietat de webs referents a aquest tema, algunes d'elles molt completes.

3 Objectius i abast

3.1 Objectius del producte

Els objectius principals de l'aplicació web són:

- Atraure noves persones a gaudir dels escacs, amb els beneficis que això comporta.
- Donar una nova font d'entreteniment i motivació a les persones que, per la seva discapacitat, no poden jugar a escacs de forma convencional.
- Permetre a un usuari jugar partides d'escacs de forma multijugador o en solitari.
- Poder dictar els moviments a realitzar amb la veu.

3.2 Objectius del client

Els objectius indicats pel client que ens ha fet l'encàrrec són:

- Desenvolupar una aplicació web funcional que permeti jugar a escacs ja sigui contra la màquina o contra un altre usuari de forma online.
- Implementar la possibilitat de poder indicar els moviments mitjançant la veu per qüestions d'accessibilitat.
- Estudiar i implementar possibles noves funcionalitats al llarg del projecte.
- Crear una interfície amigable i fàcil d'utilitzar.

El codi ha de ser net i entenedor. També ha de ser fàcil de mantenir i estructurat per facilitar la possibilitat d'ampliar-lo o millorar-lo en un futur.

3.3 Target

El públic potencial d'aquest producte és:

- Persones, principalment joves, que veuen els escacs com un joc massa seriós i els manca motivació per jugar-hi.
- Persones amb alguna discapacitat que els impedeix jugar als escacs però que sí que poden parlar per indicar els moviments mitjançant la veu.
- Aquelles persones que, ja sigui per repte personal o interès, volen practicar de jugar als escacs sense mirar.

4 Metodologia

Durant tot el desenvolupament del producte també es documenta tot el procés, avenços i altres dades necessàries per a la correcta escriptura de la memòria.

Per tal de decidir correctament i de forma objectiva en el cas d'haver d'escollir entre diferents tecnologies, funcionalitats o metodologies es fa una llista de punts positius o negatius i s'analitza que encaixa millor amb els objectius del projecte.

Per al control i gestió de canvis del desenvolupament del producte s'utilitza Git, mantenint la branca main el màxim de funcional possible i fent noves branques al desenvolupar.

El projecte es divideix en tres parts:

1. Planificació i definició del projecte:

En aquest apartat s'indiquen les tasques a realitzar, la durada d'aquestes, les dates límit, els requeriments del producte, els objectius, la finalitat del projecte i la realització d'un estudi de mercat.

En aquesta fase es pretén obtenir una visió més clara dels objectius, les aspiracions del projecte i el producte final. També de les passes a seguir per aconseguir-ho i dels terminis a complir.

2. Anàlisi i formació

En aquest apartat s'analitzen, estudien i investiguen les tecnologies, frameworks, llibreries i llenguatges necessaris i útils per al desenvolupament del producte.

La correcte execució d'aquesta etapa és vital per a l'èxit del projecte ja que tota la fase de desenvolupament es veu afectada per les decisions preses aquí.

Encara que es poden fer canvis o acabar de prendre certes decisions un cop començat el desenvolupament, normalment això implica uns majors costos per reajustar la planificació. Per tant, en aquesta etapa es busca aprofundir al màxim per garantir una millor i més fluida etapa de desenvolupament.

Aquesta etapa no consisteix només en l'elecció dels diferents frameworks i llibreries sinó també en la formació i obtenció dels coneixements necessaris.

En el cas d'aquest projecte, la tecnologia, llibreries i format necessaris per al desenvolupament de l'aplicació web són bastant nous per al desenvolupador. Per tant, aquesta fase ha de ser força llarga per aprendre i avançar en la corba d'aprenentatge de les diferents tecnologies.

3. Desenvolupament del producte:

En aquest apartat es desenvolupa el producte, fase per fase, aplicant les metodologies i tècniques apreses tal com s'indica en l'apartat de planificació.

Durant el desenvolupament del producte també es documenta la memòria.

La comprovació del correcte funcionament de l'aplicació web (*Testing*) també es fa en paral·lel en el moment que s'implementen les diferents funcionalitats.

5 Definició de requeriments i casos d'ús

5.1 Requeriments funcionals

- Permetre a l'usuari jugar una partida d'escacs contra un altre usuari de manera online.
- Permetre a l'usuari jugar una partida d'escacs contra la màquina.
- Poder indicar els moviments a realitzar mitjançant la veu.
- Permetre a l'usuari registrar-se i iniciar la sessió.
- Incorporar un sistema de puntuació per lligues a les que l'usuari pot anar ascendint conforme guanya partides.
- Rebre *coins* en guanyar una partida.
- Poder comprar diferents estils de taulell, peces i altres accessoris.
- Poder escollir el límit de temps de la partida.
- Poder jugar sense registrar-se, de forma anònima, però sense conservar el progrés obtingut un cop s'acaba la sessió.
- Poder canviar d'idioma entre anglès, castellà i català.
- La web ha de ser *responsive*, és a dir que s'ha de poder adaptar a les diferents mides i dispositius mantenint una presentació neta i polida.
- El disseny ha de ser intuïtiu i amigable per a l'usuari.

5.2 Requeriments tecnològics

- L'aplicació ha de tenir un temps de resposta suficientment curt perquè no afecti a la partida, sobretot en modes de joc ràpids.
- El codi ha d'estar ben estructurat i ser escalable.

5.3 Casos d'ús

Per poder entendre millor les necessitats que ha de cobrir l'aplicació s'han descrit una sèrie de casos d'ús sobre les diferents activitats que la web ha d'oferir.

Per cada cas d'ús es defineix:

- Nom del cas d'ús.
- Descripció: petit text per descriure l'activitat i el seu objectiu.
- Les precondicions: estat del sistema necessari abans de l'inici del cas d'ús.
- El flux normal d'execució: descriu el comportament ideal del sistema.
- Les postcondicions: possibles estats del sistema en acabar el cas d'ús.
- Flux alternatiu: descriu les excepcions del flux normal.

No s'indica l'actor, ja que per tots els casos d'ús l'actor sempre és l'usuari / jugador de la web.

Nom	Canviar d'idioma.
Descripció	Un usuari canvia l'idioma de la web.
Precondicions	L'usuari es troba a la pàgina principal.
Flux normal	1. Clica a "configuració". 2. Es desplega un menú i clica a l'opció "idioma". 3. Selecciona el nou idioma.
Postcondicions	S'ha canviat l'idioma de tota la web.
Flux alternatiu	

Taula 5.3.1 Cas d'ús de canviar d'idioma

Nom	Registrar usuari.
Descripció	Un nou usuari es registra a la web.
Precondicions	L'usuari entra a la web sense tenir cap sessió iniciada.
Flux normal	1. En entrar a la web es mostra la pantalla principal. 2. Clica a "registrar-se". 3. Introdueix el seu nom d'usuari i contrasenya. 4. El sistema valida que el nom no existeix. 5. Es registra el nou usuari. 6. Es redirigeix a la pàgina principal.
Postcondicions	L'usuari es troba a la pàgina principal amb la sessió iniciada.
Flux alternatiu	5. Si el nom ja existeix, es notifica a l'usuari amb un missatge i torna a permetre introduir un nou nom.

Taula 5.3.2 Cas d'ús de registrar usuari

Nom	Afegir un amic.
Descripció	Un usuari afegeix un altre jugador a la seva llista d'amics.
Precondicions	L'usuari es troba a la pàgina principal.
Flux normal	1. Clica a "amics". 2. Es mostra una pantalla amb un llistat d'amics i l'opció d'afegir nou amic. 3. Clica a "afegir nou amic". 4. Introdueix el nom de l'usuari a afegir i clica a "enviar sol·licitud". 5. L'altre usuari rep una notificació indicant que té una nova petició i se li dona l'opció d'acceptar o declinar la sol·licitud d'amistat. 6. Clica a "acceptar".
Postcondicions	Els dos usuaris tenen un nou usuari a la seva llista d'amistats.
Flux alternatiu	5. L'altre usuari declina la petició i no s'afegeix el nou usuari a la llista.

Taula 5.3.3 Cas d'ús d'afegir un amic

Nom	Desafiar a un amic.
Descripció	Un jugador desafia a un altre jugador de la seva llista d'amics a una partida.
Precondicions	L'usuari es troba a la pàgina principal.
Flux normal	1. Clica a "amics". 2. Clica a "desafiar" sobre l'usuari que es vol desafiar. 3. L'altre usuari rep una notificació indicant que el primer usuari l'està desafiant i se li dona l'opció d'acceptar o declinar. 4. Clica a "acceptar". 5. Es redirigeixen els dos usuaris a la pantalla de joc i comença la partida.
Postcondicions	Els dos usuaris estan a la pantalla de joc amb el format de temps determinat pel primer usuari.
Flux alternatiu	4. L'altre usuari declina la petició i no s'inicia cap partida.

Taula 5.3.4 Cas d'ús de desafiar a un amic

Nom	Canviar estils de peces / tauler.
Descripció	Un usuari canvia l'estil del taulell o de les peces.
Precondicions	L'usuari es troba a la pàgina principal.
Flux normal	1. Clica a "botiga" 2. Es mostra un menú on apareixen tots els estils dels que disposa l'usuari i un taulell al costat per poder veure els canvis. 3. Selecciona l'estil desitjat i clica "acceptar".
Postcondicions	L'usuari es troba a la pàgina inicial amb els nous estils equipats.
Flux alternatiu	

Taula 5.3.5 Cas d'ús de canviar estils de peces/tauler

Nom	Jugar una partida.
Descripció	Un usuari busca una partida per poder jugar contra un altre jugador aleatori.
Precondicions	L'usuari es troba a la pàgina principal.
Flux normal	1. Clica a "multijugador". 2. Tria el format de temps i clica "buscar". 3. S'indica que s'està buscant una partida. 4. Troba una partida i redirigeix a l'usuari a la pantalla de joc, on es veu el taulell. 5. Comença a disminuir el temps de les blanques 6. Cada jugador fa moviments fins que s'arriba a un resultat. 7. El guanyador rep *coins* i ascendeix punts en el rànquing, mentre que el perdedor descendeix en el rànquing. 8. S'activa un botó per tal de poder tornar a jugar.
Postcondicions	L'usuari ha jugat una partida i guanyat/perdut posicions en el rànquing. Es queda a la pàgina del joc amb l'opció de poder tornar a jugar o tornar a la pàgina inicial.
Flux alternatiu	

Taula 5.3.6 Cas d'ús de jugar una partida

6 Desenvolupament

6.1 Apartats de la web

En aquest apartat s'expliquen i mostren les diferents pàgines o apartats de la web. S'expliquen des de el punt de vista d'utilitat per a l'usuari final i per informar de les característiques de la web.

Més endavant s'explica cada funcionalitat de forma més detallada sobre el seu desenvolupament.

6.1.1 Pàgina principal

Aquesta és la pàgina principal de la web, des de aquí l'usuari pot navegar a altres pàgines o esperar mentre busca una partida.

Tal i com es pot veure a la Fig. 6.1.1.1 s'han incorporat diferents botons per poder navegar per la web. També s'ha creat un taulell on l'usuari pot moure les peces dels dos colors, ja sigui per practicar posicions o per distreure's mentre espera.

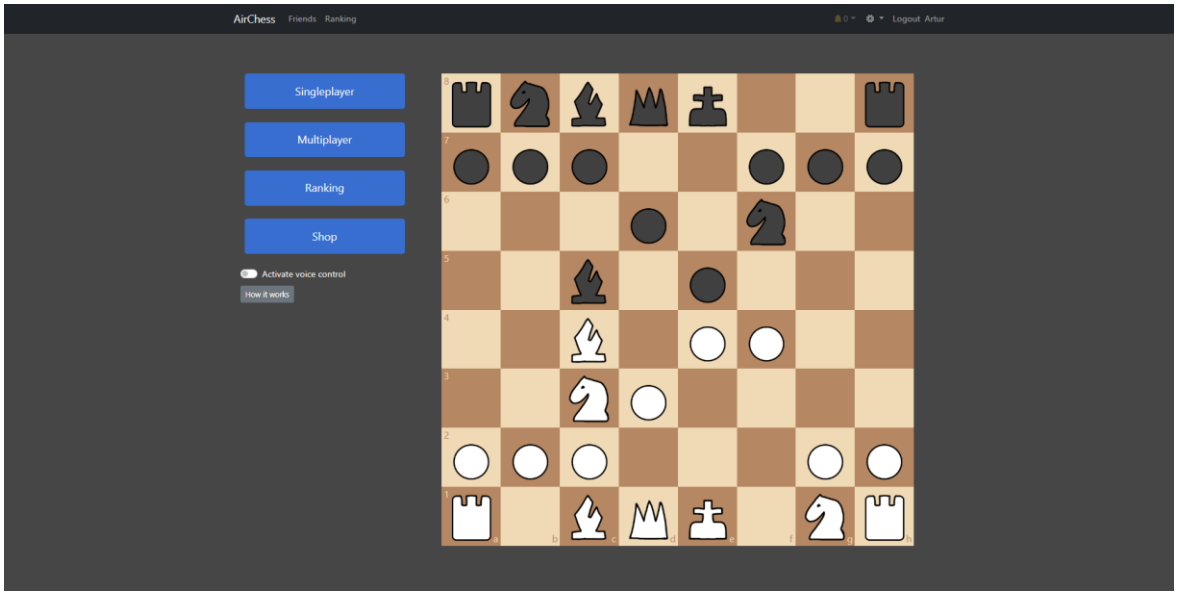

Fig. 6.1.1.1 Pàgina principal. Font: Elaboració pròpia.

La pàgina s'ha fet completament *responsive* per adaptar-se a totes les mides. Al disminuir la mida de la pantalla també ho fa la mida del taulell. L'espai extra disminueix fins que, a partir d'una certa mida, la pantalla es reestructura per adaptar-se millor, tal i com es pot veure en la Fig. 6.1.1.2 i la Fig. 6.1.1.3

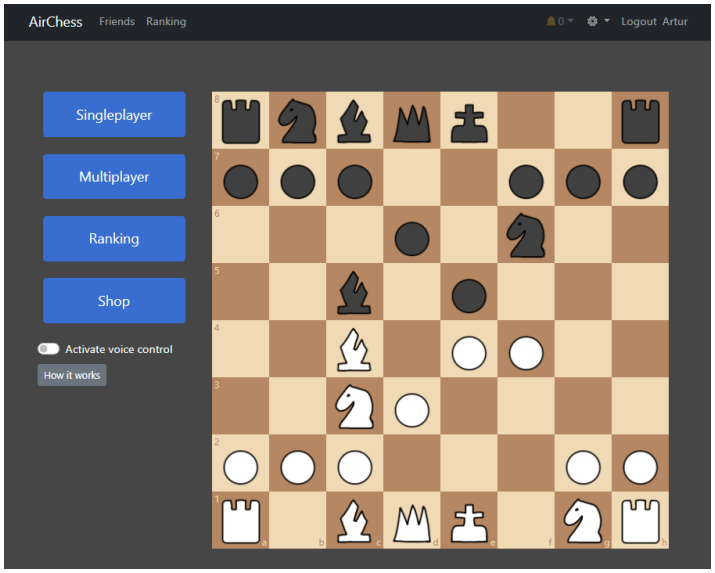

Fig. 6.1.1.2 Pàgina principal mida mitjana. Font: Elaboració pròpia.

Fig. 6.1.1.3 Pàgina principal mida petita. Font: Elaboració pròpia.

En buscar partida, s'obre el modal de la Fig. 6.1.1.4 perquè l'usuari pugui seleccionar el mode i format de joc

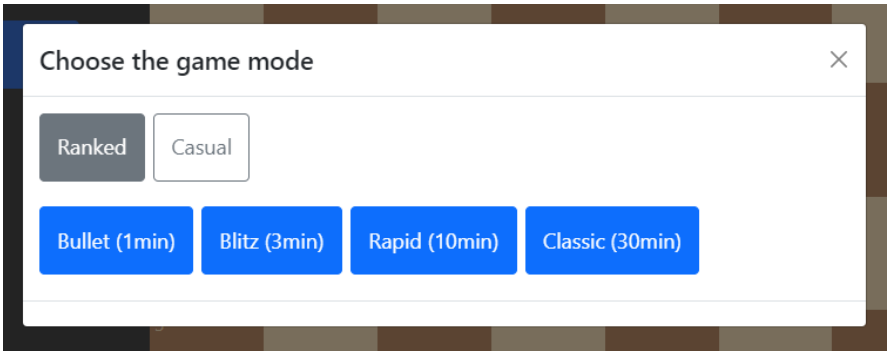

Fig. 6.1.1.4 Modal per seleccionar el mode de joc. Font: Elaboració pròpia.

6.1.2 Pàgina de joc per un jugador

En aquesta pagina l'usuari pot jugar una partida contra la màquina, es a dir, contra una intel·ligència artificial.

En el moment que l'usuari realitza el seu moviment, l'oponent fa el seu automàticament.

En totes les pàgines que contenen un tauler d'escacs hi ha el component de control de veu, que l'usuari pot activar en cas de que vulgui dictar els moviments mitjançant comandes de veu.

També hi ha un boto que obre un modal informatiu sobre l'ús del control per veu i un altre botó per reiniciar la partida.

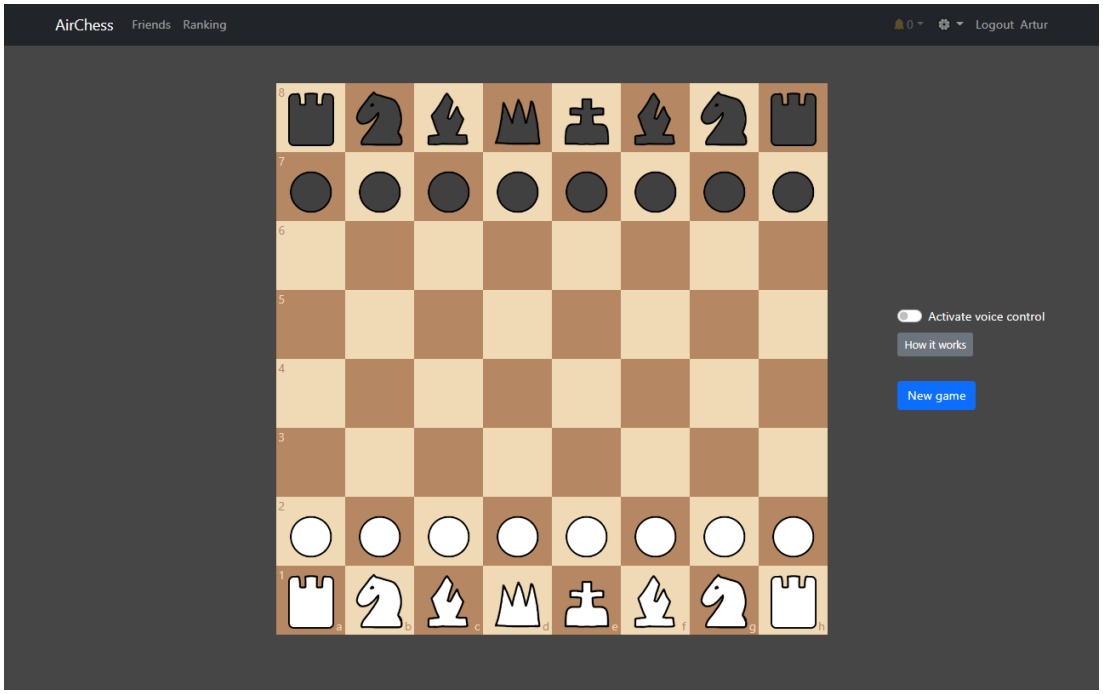

Fig. 6.1.2 Pàgina de joc un jugador. Font: Elaboració pròpia.

6.1.3 Pàgina de joc multijugador

Aquesta és la pàgina on dos usuaris poden jugar una partida de forma online, és molt semblant a la pàgina en mode "un jugador".

S'afegeixen dos rellotges que indiquen el temps restant de cada jugador. El temps corre per al jugador que li toca moure.

Fig. 6.1.3 Pàgina de joc multijugador. Font: Elaboració pròpia.

6.1.4 Pàgina d'amics

Aquesta pàgina només està disponible per als usuaris registrats. L'usuari aquí pot enviar sol·licituds d'amistat a d'altres usuaris.

També pot veure el llistat dels seus amics i desafiar-los a una partida.

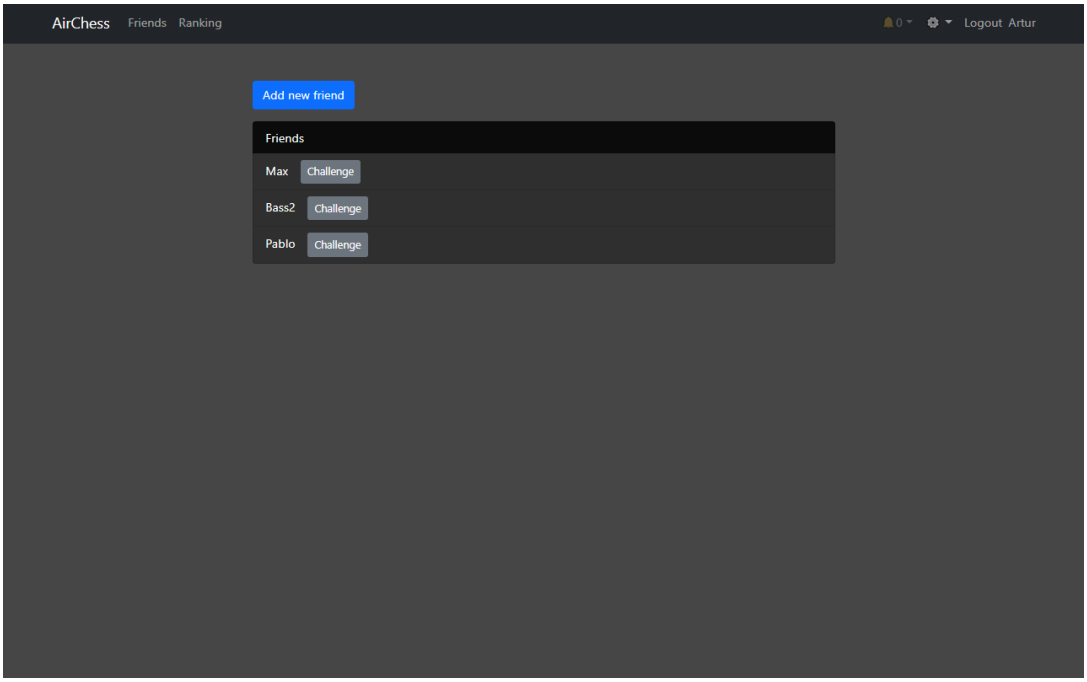

Fig. 6.1.4 Pàgina d'amics. Font: Elaboració pròpia.

6.1.5 Pàgina de classificació

En aquesta pàgina es mostra un llistat de tots els usuaris, els seus punts i la seva distribució en la taula classificatòria.

Els jugadors estan distribuïts en diferents lligues segons els ses punts.

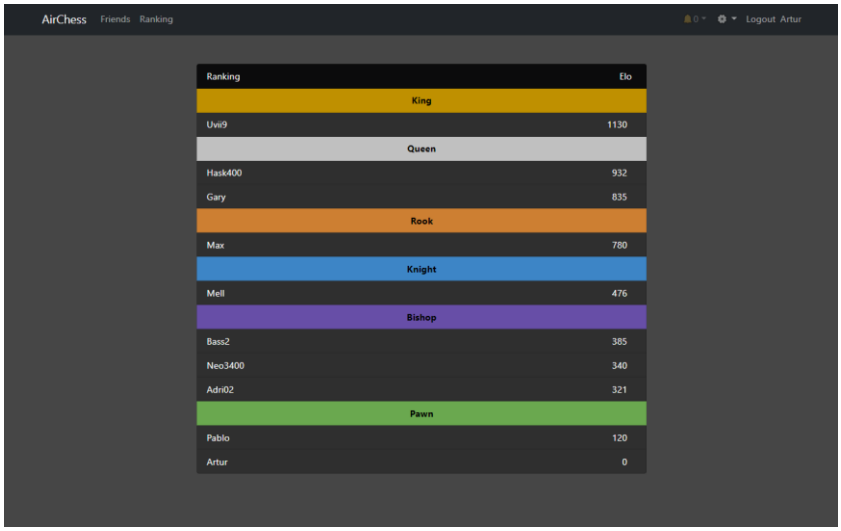

Fig. 6.1.5 Taula classificatòria. Font: Elaboració pròpia.

6.1.6 Pàgina de botiga

Aquí l'usuari pot utilitzar els seus coins per comprar estils de tauler i de peces.

Una vegada comprat, pot equipar aquell estil per utilitzar-lo en totes les altres pàgines en les que hi hagi un tauler de joc.

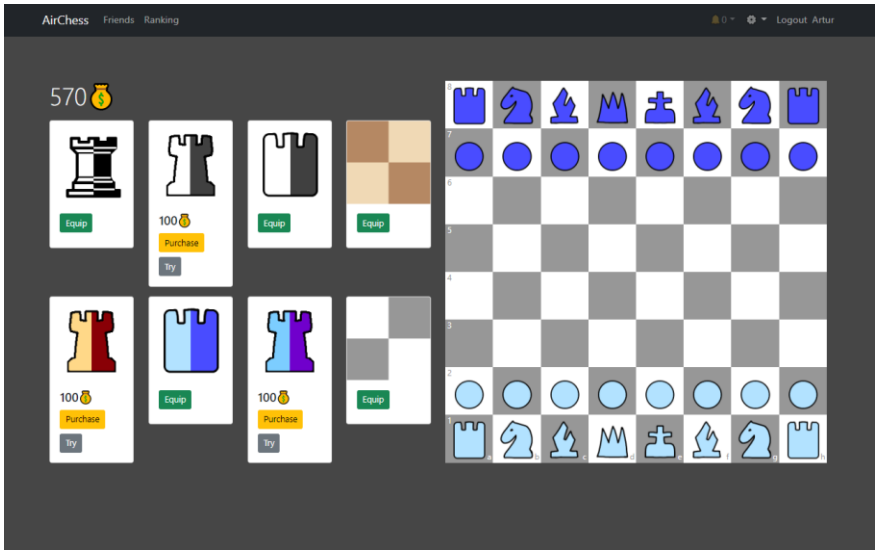

Fig. 6.1.6 Pàgina de botiga. Font: Elaboració pròpia.

6.1.7 Pàgines d'iniciar sessió o registrar-se

Formulari amb camps "nom d'usuari" i "contrasenya" perquè l'usuari pugui registrar-se o iniciar sessió en cas de tenir ja un compte.

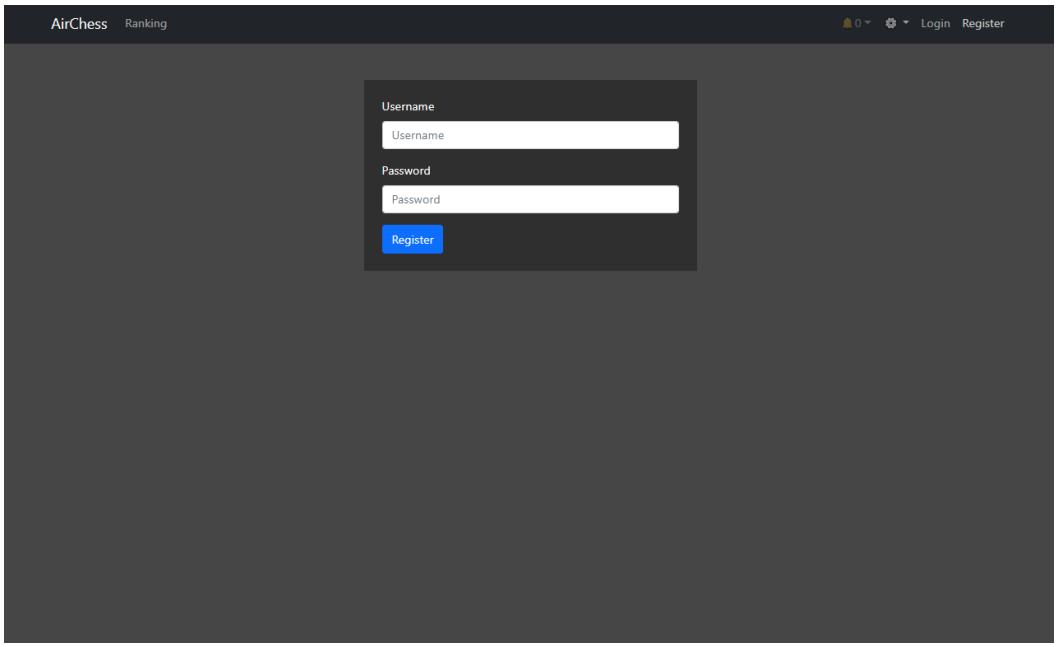

Fig. 6.1.7 Pàgina de registre. Font: Elaboració pròpia.

6.1.8 Barra de navegació

A la barra de navegació hi ha el nom de la pàgina "AirChess". L'usuari pot clicar-hi a sobre per anar a la pàgina principal, també pot accedir a la pàgina del ranking o a la d'amics, aquesta última només un cop l'usuari està registrat.

Si no té cap sessió iniciada, l'usuari té l'opció de registrar-se o iniciar sessió.

També hi ha l'accés a les notificacions, simbolitzat amb la icona d'una campana. Quan l'usuari rep una petició d'amistat o un desafiament podrà veure-ho fent clic sobre la campana.

Per últim, també hi ha la configuració identificada amb un engranatge, mitjançant la qual l'usuari pot canviar d'idioma.

Fig. 6.1.8.1 Barra de navegació sense sessió. Font: Elaboració pròpia.

Fig. 6.1.8.2 Barra de navegació amb sessió. Font: Elaboració pròpia.

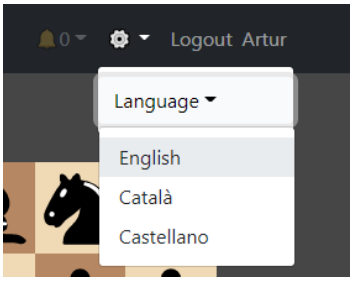

Fig. 6.1.8.3 Configuració d'idioma. Font: Elaboració pròpia.

6.2 Web framework

6.2.1 Elecció

Per tal de poder desenvolupar l'aplicació web de forma correcta i adequada als requisits plantejats és essencial escollir les tecnologies a utilitzar i detallar el motiu de l'elecció.

Per al desenvolupament d'aquesta web s'ha decidit utilitzar React [8]. Encara que realment és considerada una llibreria, també sol anomenar-se dins dels frameworks degut a les funcions que desenvolupa i a la seva utilitat.

React és una llibreria per crear interfícies d'usuari que facilita el desenvolupament d'aplicacions web, sobretot en aplicacions on les dades canvien constantment i és necessari tornar a renderitzar els nous valors.

També permet crear components que es poden reutilitzar en diferents parts de la web i així estructurar millor el codi en components ben definits.

S'utilitza javascript com a llenguatge i també JSX (Javascript Syntax Extension), una sintaxi semblant al HTML que permet estructurar de forma més llegible el component a renderitzar, tal i com es pot veure a la Fig. 6.2.1.1.

```jsx
function App() {
  return (
    <>
      <h1>Exemple JSX</h1>
      <p>És molt semblant a html per fer-ho més llegible</p>
      <CustomNav />
      <Game /> {/* es poden utilitzar components propis */}
    </>
  );
}
```

Fig. 6.2.1.1 Exemple JSX. Font: Elaboració pròpia.

És el més utilitzat d'entre els seus competidors tal i com podem veure a la Fig. 6.2.1.2 extreta de l'enquesta feta al 2021 [9].

És utilitzat per grans empreses com Facebook, Instagram, Netflix, Pinterest, Airbnb, Reddit, Paypal i moltes més.

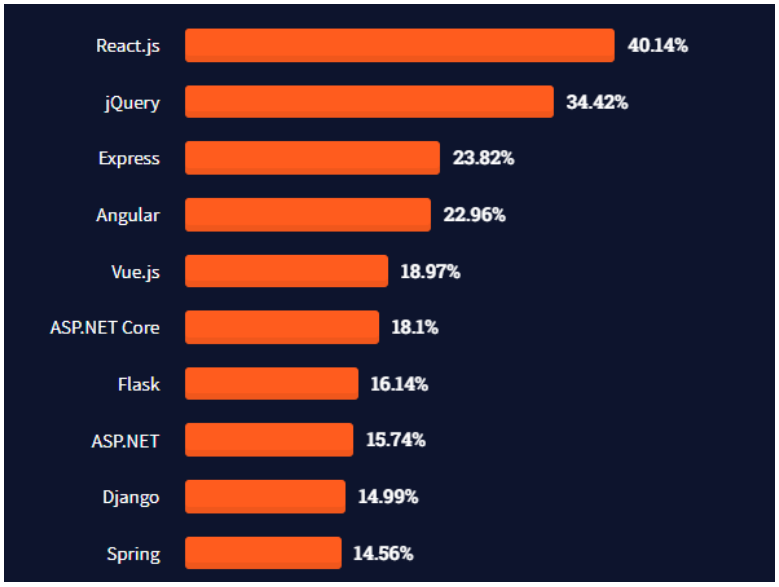

Fig. 6.2.1.2 Gràfic dels Web Frameworks més utilitzats. Font: Stackoverflow Insights Survey, 2021

S'ha decidit React per sobre altres frameworks populars com Angular o Vue per diversos motius:

- La seva popularitat no només mostra que és útil sinó també que hi ha molta documentació i informació sobre els diferents àmbits i problemes amb els que et pots trobar.
- En tenir el suport de grans empreses, com Facebook, garanteix seguir en constant manteniment i actualització en el futur.
- En ser una llibreria i no un framework, dona més llibertat i facilitat per adaptar-se a cada projecte més fàcilment.
- La reutilització de components agilitza el desenvolupament.

Evidentment no tot és perfecte i també hi ha alguns punts negatius de treballar amb React:

- Justament, el fet de donar aquesta llibertat en comparació amb altres frameworks més estructurats fa que si no es manté una bona estructura es dificulti el manteniment a llarg termini.
- En estar constantment en desenvolupament, React va canviant i això és vist com un inconvenient per alguns desenvolupadors.

6.2.2 Conceptes bàsics

Per entendre les futures explicacions, és interesant entendre alguns conceptes bàsics de React.

Components

Els components React estan constituïts per un conjunt d'estats i funcions, i retornen l'element a renderitzar.

En el exemple de la Fig. 6.2.2.1 es crida al component Chessboard. Un component pot rebre paràmetres, en aquest cas se li passen multitud de paràmetres. El component utilitza els valors d'aquests paràmetres per definir les accions del taulell i renderitzar-lo adequadament.

```
<Chessboard
  boardWidth={boardWidth}
  position={game.fen()}
  onPieceDrop={onPieceDrop}
  isDraggablePiece={isDraggablePiece}
  boardOrientation={boardOrientation()}
  onSquareClick={() => cancelPromotion()}
  onPieceDragBegin={() => cancelPromotion()}
  customSquareStyles={{ ...kingInCheckSquare }}
  arePiecesDraggable={arePiecesDraggable}
  customDarkSquareStyle={customDarkSquareStyle()}
  customLightSquareStyle={customLightSquareStyle()}
  customPieces={customPieces()}
/>
```

Fig. 6.2.2.1 Crida al component Chessboard. Font: Elaboració pròpia.

Hooks

Els hooks s'utilitzen per permetre als components tenir accés a l'estat i a altres funcionalitats de React. Alguns exemples són:

- useState

Serveix per declarar una variable d'estat per poder conservar un valor entre les diferents renderitzacions del component.

En la Fig. 6.2.2.2 es mostra la inicialització de la variable d'estat game. Com a valor inicial pren el valor de new Chess() i més endavant es pot actualitzar aquest valor gracies a la funció setGame.

```
const [game, setGame] = useState(new Chess());
```

Fig. 6.2.2.2 Variable d'estat del game. Font: Elaboració pròpia.

- useEffect

Aquest hook permet executar efectes secundaris.

Té dos paràmetres, el primer és una funció i el segon és un array de variables. Cada vegada que es detecta un canvi en alguna de les variables del array s'executa la funció del primer paràmetre.

```
useEffect(() => {
    socket.on("moveDone", ({ from, to, promotion }, oppTime) => {
        opponentTimer.stop(oppTime)
        yourTimer.start()
        let move = doMove(from, to, promotion)
        if (!move.gameOver)
            socket.emit('receivedMove', gameId)
    })
    return () => {
        socket.off('moveDone')
    }
}, [gameId, doMove, yourTimer, opponentTimer]);
```

Fig. 6.2.2.3 useEffect que incorpora el listener encarregat de rebre un moviment. Font: Elaboració pròpia.

Custom hooks

Els custom hooks són els hooks que pots desenvolupar tu mateix per tal de compactar funcionalitats de codi concretes.

6.3 Base de dades

És necessària una base de dades per poder emmagatzemar la informació dels usuaris:

- Les seves credencials.
- Els seus punts o posició en el rànquing.
- Contactes, amics.
- Els seus *coins*.
- Els estils de peces i taulell dels que disposa.

Per aquest projecte s'ha escollit MongoDB com a base de dades.

MongoDB és una base de dades NoSQL orientada a documents. La principal diferència amb una base de dades relacional és que no guarda les dades en taules, sinó que les guarda en documents BSON, un tipus de document semblant als JSON.

És la base de dades NoSQL més popular del mercat i és molt utilitzada sobretot en entorns de desenvolupament web.

S'ha escollit per la seva simplicitat, flexibilitat i fàcil integració amb el producte a desenvolupar, és a dir, amb la pròpia web.

Al crear la base de dades s'ha d'escollir el tipus de servei. En aquest cas s'ha escollit el pla gratuït ja que d'entrada no es preveuen una gran quantitat d'usuaris i per tant és més que suficient. En un futur es pot millorar aquest pla si la situació ho demana.

Un cop creada la base de dades, es genera un *connection string* i aquesta cadena de text es guarda a l'arxiu .env del servidor i s'utilitzarà per establir la connexió.

No obstant això, la gran flexibilitat de MongoDB pot convertir-se en un problema si no es defineix una estructura clara de les dades, ja que a la llarga pot derivar en inconsistències de dades. Per evitar això s'utiltza Mongoose.

Mongoose és un llibreria d'*Object Data Modeling* per MongoDB i Nodejs. Serveix per definir els esquemes de les dades, afegir validació i gestionar les relacions entre dades; i ofereix moltes més funcionalitats per facilitar la comunicació amb MongoDB.

Pel cas d'aquest projecte només és necessària una col·lecció, la d'usuaris.

A la Fig. 6.3.1 es pot veure l'esquema definit pels usuaris.

Es guarda el nom d'usuari, la contrasenya encriptada, l'elo (punts en la classificació), els coins dels que disposa, l'estil de peces i de tauler actual del jugador, els objectes que ha comprat i finalment el llistat amb les ids dels seus amics.

S'indica que els amics fan referencia a User. D'aquesta manera, gràcies al mètode *populate* de Mongoose, després es poden obtenir fàcilment els usuaris de la llista.

```
const userSchema = new Schema({
    username: {
        type: String,
        unique: true,
        required: true
    },
    password: {
        type: String,
        required: true
    },
    elo: Number,
    coins: Number,
    config: {
        pieces: String,
        board: String,
    },
    itemsPurchased: [String],
    friends: [{
        type: mongoose.SchemaTypes.ObjectId,
        ref: 'User'
    }]
})
```

Fig. 6.3.1 Esquema d'User. Font: Elaboració pròpia.

```
_id: ObjectId("62a3883014e451042d24c813")
username: "Artur"
password: "$2b$10$EWKvr30nVI.fTNBQj4Z66.k.4kAlboP4Ky.Am3kEcVSvurbkuge4."
elo: 1250
coins: 870
config: Object
    pieces: "standard"
    board: "standard"
itemsPurchased: Array
friends: Array
    0: ObjectId("62a37eec14e451042d24c806")
    1: ObjectId("62a37eec14e451042d24c807")
    2: ObjectId("62a37eec14e451042d24c808")
    3: ObjectId("62a37eec14e451042d24c803")
__v: 0
```

Fig. 6.3.2 Exemple d'un usuari a la base de dades. Font: Elaboració pròpia.

6.4 API

És necessària una API per fer d'intermediari entre la web i la base de dades.

Per desenvolupar el servidor s'ha utilitzat Node.js, que és un entorn en temps d'execució basat en javascript dissenyat per desenvolupar aplicacions escalables, generalment servidors web.

Per a l'API s'utilitza express.js, que és un framework per Node.js utilitzat per dissenyar i desenvolupar aplicacions web, en aquest cas concretament una API.

Express.js és molt popular degut a la seva senzillesa i ràpid desenvolupament.

Un altre motiu per escollir express és justament que funcioni amb Node.js. Això es indispensable ja que tant Mongoose per la base de dades, com Socket.io per la connexió entre usuaris, funcionen també amb Node.js.

6.4.1 Seguretat

Quan un usuari inicia sessió es validen les seves credencials. En cas que l'usuari i contrasenya siguin correctes es genera un JSON WEB TOKEN (JWT) i es retorna a l'usuari.

Un JWT és un token signat per una clau secreta que es troba en el servidor.

Quan un usuari fa una petició a la API també ha d'enviar el token que ha rebut a l'iniciar sessió en el camp d'Authorization. D'aquesta manera, el servidor pot validar el token per assegurar que la petició l'està realitzant l'usuari en concret.

6.4.2 Rutes

GET /api/users

Descripció: Obté tots els usuaris

Possibles estats:

- 200: Funcionament correcte, retorna la llista d'usuaris
- 500: Error del servidor

GET /api/users/sorted

Descripció: Obté tots els usuaris ordenats per l'elo, de major a menor.

Possibles estats:

- 200: Funcionament correcte, retorna la llista d'usuaris
- 500: Error del servidor

GET /api/users/:id

Descripció: Obté l'usuari identificat per la ID

Possibles estats:

- 200: Funcionament correcte, retorna l'usuari
- 400: Petició incorrecte
- 500: Error del servidor

POST /api/users

Descripció: Crea un nou usuari

Body:

- username
- password

Possibles estats:

- 200: Funcionament correcte, retorna l'usuari
- 400: Petició incorrecte
- 500: Error del servidor

POST /api/users/friends

Descripció: Crea un nou usuari

Body:

- senderId
- receiverId

Possibles estats:

- 200: Funcionament correcte
- 400: Petició incorrecte
- 401: No autoritzat
- 500: Error del servidor

POST /api/users/:id/purchase/:item

Descripció: Incorpora l'objecte a la llista d'objectes comprats per l'usuari

Possibles estats:

- 200: Funcionament correcte, retorna l'usuari
- 400: Petició incorrecte o coins insuficients
- 401: No autoritzat
- 500: Error del servidor

POST /api/login

Descripció: Valida les credencials del usuari i retorna la seva informació amb el token d'autorització corresponent.

Body:

- username
- password

Possibles estats:

- 200: Funcionament correcte, retorna l'usuari i el token
- 400: Petició incorrecte
- 500: Error del servidor

6.5 Disseny i estilització web

Un dels requeriments consisteix en obtenir una web *responsive* que s'adapti a les diferents mides i dispositius.

Això s'ha aconseguit fent ús del Grid System de Bootstrap, aquest sistema utilitza una sèrie de contenidors, files i columnes per tal d'ubicar els diferents continguts de la web.

Bootstrap utilitza *breakpoints* que són bàsicament diferents mides de pantalla tal i com es pot veure a la Fig. 6.5.

Breakpoint	Class infix	Dimensions
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Fig. 6.5 Breakpoints Bootstrap. Font: getbootstrap.com

Per cada *breakpoint* es pot definir quan ha d'ocupar cada contingut de la web, canviar la mida dels marges o espaiats i molt més.

Bootstrap també s'ha utilitzat per estilar els diferents components de forma més senzilla: botons, barra de navegació, modals, llistats i formularis.

6.6 Testing

El nucli de la web i per tant la part més important a testejar es tracta de la patida d'escacs multijugador.

En aquest apartat l'usuari fa moviments sobre un taulell i aquests s'envien mitjançant sockets a un altre jugador.

El fet de que es necessitin dos usuaris jugant simultàniament afegeix molta dificultat per desenvolupar un testing automàtic.

Per tant, degut a aquesta dificultat i al ajustat temps de desenvolupament del projecte, no s'ha pogut realitzar el testing automàtic tal i com hauria sigut ideal.

Per tant, aquesta part s'ha testejat manualment de forma exhaustiva i es considera la implementació d'un testing automatitzat com a activitat prioritària en un futur.

Si que s'ha realitzat algun test automatitzat molt bàsic de comprovació de renderitzat de components i les diferenciacions en funció de si l'usuari està registrat o no.

6.7 Funcionalitats

6.7.1 Rutes i navegació

Per tal de que l'usuari pugui navegar per la web i visitar les diferents pàgines, és necessari un sistema de rutes.

S'ha decidit desenvolupar la web seguint el sistema de Single-page application (SPA), això vol dir que la pàgina web es carrega un sol cop i els recursos necessaris es mostren dinàmicament en funció de les accions del usuari.

D'aquesta manera s'aconsegueix una experiència molt més fluida.

Per aconseguir un sistema de navegació i a l'hora evitar que la pàgina es torni a carregar al canviar de ruta s'ha utilitzat react-router-dom, una llibreria que ofereix la possibilitat de crear un sistema de rutes de forma declarativa, completament dinàmica i mantenint el SPA.

Per aconseguir això, tots els components de la aplicació s'han d'envoltar dins d'un component anomenat BrowserRouter.

A continuació s'ha de utilitzar el component Routes per englobar totes les rutes de l'aplicació. Per cada ruta que es vulgui afegir, s'utilitza el component Route.

Aquest component té diversos paràmetres: s'ha d'indicar el path, que és el camí o ruta, i també s'ha d'indicar l'element, és a dir, el component React a renderitzar quan la ruta coincideixi.

Apart de l'encaminament declaratiu, també es pot utilitzar el hook de useNavigate per redirigir a l'usuari. Aquest hook retorna una funció que permet redirigir a l'usuari indicant el path. Això és útil quan es troba partida per redirigir a l'usuari a la pàgina de joc, o també es utilitzat en la barra de navegació.

Les rutes de l'aplicació web són les següents:

Pàgina principal: /

Pàgina d'una partida multijugador: /game/:id

Pàgina d'una partida contra la maquina: /game

Pàgina d'inici de sessió: /login

Pàgina de registre: /register

Pàgina d'amics: /friends

Pàgina de taula classificatòria: /ranking

Pàgina de la botiga: /shop

6.7.2 Lògica i renderització del joc dels escacs

Per facilitar el desenvolupament de tota la lògica del joc s'utilitza la llibreria chess.js [10].

Aquesta llibreria permet controlar les peces en joc, el seu posicionament i moviment, així com la validació de moviments legals i la detecció d'escac i mat entre moltes altres funcionalitats.

Alguns dels mètodes que ofereix aquesta llibreria són:

- Obtenir la posició actual.
- Comprovar si el joc ha acabat.
- Retornar la peça que hi ha en una determinada casella.
- Comprovar si el rei està en *check*, *checkmate* o *stalemate*.
- Moure una peça.
- Obtenir el jugador (blanc/negre) al que li toca moure.
- Reiniciar el joc.

Aquesta llibreria, però, no ofereix cap mètode per tal de renderitzar i mostrar el taulell, només s'encarrega de la lògica del joc.

Per això s'utilitza també react-chessboard [11], una llibreria que ofereix un component React per renderitzar un taulell d'escacs i que permet personalitzar el taulell, les peces i la mida del taulell, entre altres coses.

Només aporta el suport visual, per tant és necessari complementar-la amb una altra llibreria com la de chess.js per poder dotar el taulell amb la lògica pertinent.

D'altra banda, tampoc disposa d'una interfície per a la promoció d'un peó, per tant aquesta funcionalitat s'ha hagut de desenvolupar completament.

6.7.3 Un jugador

Per el mode de joc d'un jugador s'utilitzen les llibreries de chess.js i react-chessboard mencionades en l'apartat anterior.

S'utilitza el mètode onPieceDrop del component Chessboard per detectar quan l'usuari intenta realitzar un moviment. A continuació es comprova que el moviment sigui vàlid i s'executa sobre l'objecte game de chess.js.

En cas de que sigui un moviment de promoció (un peó arribant a l'última casella) es mostra la interfície de promoció, on l'usuari pot escollir entre dama, cavall, alfil o torre per promocionar el peó. Un cop selecciona la peça, es realitza el moviment.

Després de realitzar el moviment, es fan les comprovacions de fi de joc: escac i mat, empat per repetició, per material insuficient, per estancament, etc...

Sí el joc ha acabat es mostra el resultat al jugador i es dona l'opció de tornar a jugar.

Per tal de realitzar els moviments del oponent s'utilitza la llibreria js-chess-engine, una intel·ligència artificial bastant senzilla d'escacs en javascript.

Aquest tipus d'intel·ligències artificials necessiten un cert temps per calcular el moviment, en funció de la dificultat tarden més o menys temps.

Aquí es troba un problema. Quan s'intenta realitzar el mètode per trobar el moviment, hi ha un període de temps que es bloqueja la web, impedint a l'usuari realitzar cap altre acció i donant una sensació de poca fluïdesa i mal funcionament.

Per solucionar això s'han investigat diferents opcions i s'ha solucionat utilitzant els Web Workers.

Un Web Worker permet executar un script en un altre thread a segon pla, d'aquesta manera es pot executar codi javascript sense afectar al rendiment de la pàgina. Un cop creat pot rebre i enviar missatges.

S'ha creat un worker que pot rebre missatges amb l'estat del joc, posició de les peces i nivell de dificultat desitjat.

Quan el worker rep aquest missatge executa el mètode per trobar el moviment i quan el troba l'envia un altre cop al thread principal.

És rep el missatge i s'executa el moviment obtingut. D'aquesta manera torna a ser el torn de l'usuari i en cap moment s'ha bloquejat el flux normal de la web.

6.7.4 Control de veu

Un dels requeriments de la web consisteix en poder indicar els moviments mitjançant la web.

Per aconseguir això es fa ús de la Web Speech API, que permet incorporar i controlar les dades de veu en aplicacions web. Disposa d'un mòdul SpeechRecognition, que és l'interficie de control per el servei de reconeixement de veu.

Concretament s'ha utilitzat la llibreria react-speech-recognition, es tracta d'una llibreria que ofereix un custom hook de React per realitzar reconeixement de veu, utilitzant per sota la Web Speech API.

S'ha desenvolupat el component que es pot veure a la Fig. 6.7.4.1 i Fig. 6.7.4.2.

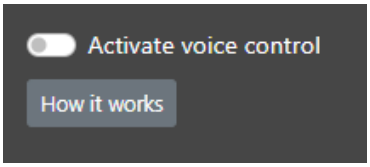

Fig. 6.7.4.1 Component de control de veu desactivat. Font: Elaboració pròpia.

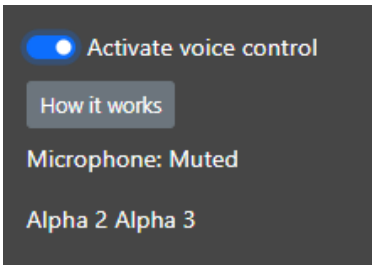

Fig. 6.7.4.2 Component de control de veu activat. Font: Elaboració pròpia.

El jugador pot activar o desactivar la funcionalitat. També hi ha un botó que mostra un modal amb informació al respecte, s'indica si el micròfon està escoltant o no i finalment s'indica el text obtingut per l'ultima frase dita per l'usuari.

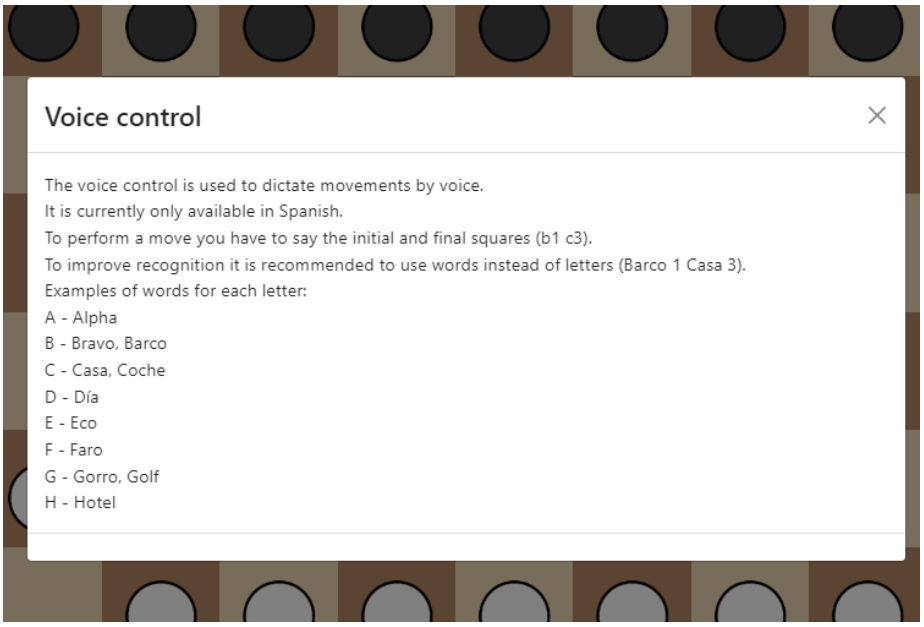

Fig. 6.7.4.3 Modal informatiu. Font: Elaboració pròpia.

A l'hora de captar i processar la frase per obtenir el moviment s'han de tenir en consideració certes situacions.

La situació normal és quan la frase obtinguda és de l'estil "Eco 2 Eco 4". En aquest cas, es divideix la frase en paraules, s'obté el primer caràcter de la primera i tercera paraula i s'uneixen amb la segona i quarta paraula respectivament per tal de formar el moviment "e2 e4"

Un altre situació més problemàtica es troba quan s'obté una frase de l'estil "Eco2 Eco4". El reconeixement per veu identifica que el número va unit a la paraula. Això no es pot processar igual que en l'anterior cas ja que la frase no està formada per quatre paraules.

Per solucionar això es comprova si l'últim caràcter de les paraules és un número i, si es així, es separa per convertir "Eco2 Eco4" a "Eco 2 Eco 4" i, per últim, a "e2 e4"

Una altra situació és quan la frase és "Eco dos Eco tres". El reconeixement per veu interpreta els números en forma de paraula.

En cas de que les paraules dos i quatre no siguin un número, es busca si coincideix amb un número escrit i, en cas afirmatiu, es canvia.

Un cop processada la frase es comprova si el moviment obtingut es vàlid i es pot realitzar en la situació actual del taulell. Si no es així, s'indica i l'usuari pot tornar a dir el moviment.

Tots els moviments, ja siguin els realitzats pel propi usuari o els de l'oponent s'indiquen i reprodueixen en format auditiu. Aquesta funcionalitat només està activa quan l'usuari ha activat el control per veu.

D'aquesta manera si l'usuari té una discapacitat visual pot saber tots els moviments que s'han realitzat.

Finalment, s'ha introduït una altre possible comanda, la de "Nueva partida", que serveix per tornar a buscar partida. D'aquesta manera, un cop l'usuari troba una partida pot realitzar tots els moviments amb la veu i, un cop acabada, tornar a buscar partida també amb la veu.

6.7.5 Connexió entre usuaris

Per poder desenvolupar una aplicació web que permeti jugar a escacs de manera multi jugador és necessari establir una connexió entre els dos usuaris per sincronitzar els seus moviments i mostrar l'estat del taulell actualitzat constantment.

També es necessària aquesta connexió per poder enviar sol·licituds d'amistat i desafiar als teus amics

Per aconseguir això es fa ús dels Web Sockets.

Un Web Socket [12] és una tecnologia que proporciona un canal de comunicació bidireccional i fa possible obrir una connexió entre client – servidor, permet enviar i rebre missatges sense haver de fer una petició al servidor per obtenir les respostes.

En aquest projecte s'utilitza concretament la llibreria Socket.io [13]

Es tracta d'una API bastant potent que permet establir una comunicació bidireccional i de baixa latència entre el client i el servidor per Node.js.

S'utilitzen principalment els mètodes socket.emit per enviar missatges i socket.on per incorporar *listeners.*

6.7.6 Multijugador

Una altre funcionalitat de la web és poder realitzar partides contra altres usuaris de manera online.

Per desenvolupar aquesta funcionalitat han estat necessàries les llibreries de chess.js i react-chessboard juntament amb la de Socket.io.

El primer pas es crear la partida, tal i com es mostra al diagrama de la Fig. 6.7.6.1.

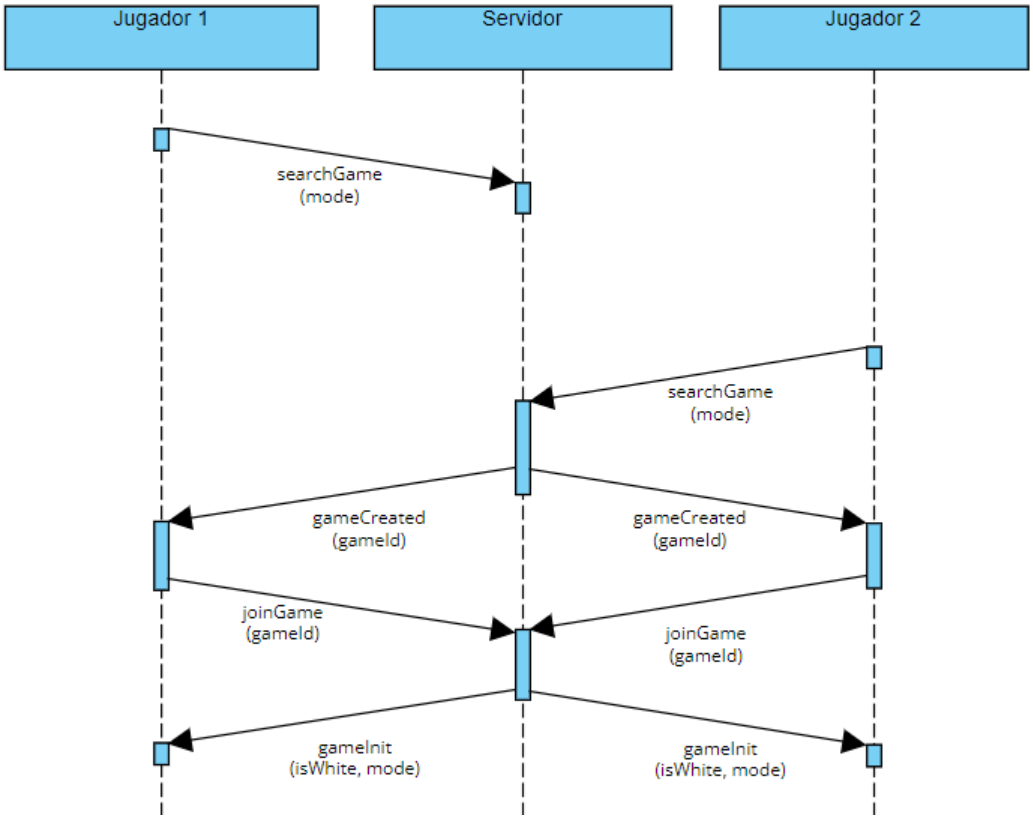

Fig. 6.7.6.1 Diagrama de creació d'una partida. Font: Elaboració pròpia.

Quan un jugador busca partida indica el mode de joc (casual/ranked) i el format de temps (bullet, blitz, rapid, classic). El servidor rep el missatge de "searchGame" i guarda la id del jugador a la llista d'espera d'aquell mode de joc.

Quan un segon jugador busca partida del mateix mode de joc, el servidor emparella als dos jugadors, crea l'objecte partida i envia un missatge de "gameCreated" amb la id de la partida per informar als dos jugadors.

Els jugadors reben el missatge i es redirigeixen a /game/:id, on id és la id de la partida. Quan es carrega la pàgina de la partida, s'envia "joinGame" per indicar que ja estan a la pàgina de la partida.

Un cop el servidor rep la confirmació dels dos jugadors, envia un "gameInit" indicant que la partida comença; també passa el mode de joc i el color del jugador (blanc/negre).

Cada jugador té el seu rellotge. Quan la partida comença, disminueix el temps de les blanques. Quan el jugador de les blanques fa el seu moviment, el seu rellotge para i comença a disminuir el de les negres.

Un dels problemes més importants a l'hora de desenvolupar aquesta funcionalitat ha estat la sincronització del temps dels jugadors. Això es deu a que l'enviament de missatges entre els diferents jugadors no es instantani i tarda un temps.

Per tant, quan un jugador fa un moviment, l'altre jugador el rep uns instants més tard i això fa que els rellotges no indiquin el mateix temps.

Per solucionar això, quan un jugador realitza un moviment, a part d'enviar-se el moviment també s'envia l'hora exacta a la que s'ha fet el moviment.

A més, no inicia el rellotge de l'oponent fins que no rep el missatge conforme l'altre jugador ha rebut el teu moviment. D'aquesta manera s'assegura que el rellotge de l'oponent sempre va per darrera del seu propi rellotge.

És pot observar aquest enviament de missatges en el diagrama de la Fig. 6.7.6.2.

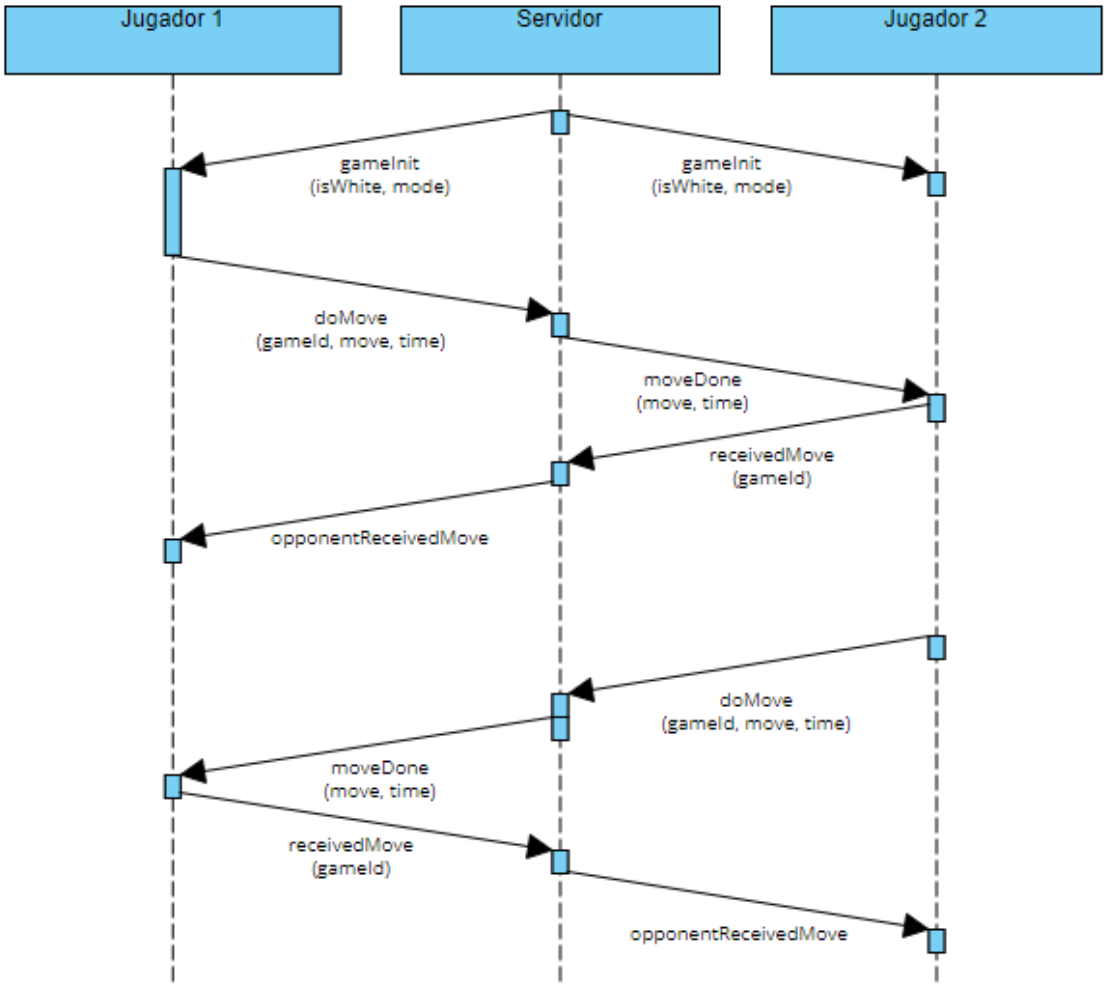

Fig. 6.7.6.2 Diagrama de moviments en partida. Font: Elaboració pròpia.

6.7.7 Amics

Els jugadors han de poder enviar i acceptar peticions d'amistat a altres usuaris.

Quan l'altre usuari accepta la petició d'amistat, aquest jugador apareix a la seva llista d'amics des de la qual el podrà desafiar a una partida.

El servidor manté una llista dels jugadors connectats. Quan un usuari fa una petició d'amistat, s'envia el missatge mitjançant sockets. Si l'altre jugador està connectat li arriba el missatge i apareix a la zona de notificacions com es veu a la Fig. 6.7.7.1.

L'usuari pot triar entre acceptar o cancel·lar la petició. Si l'accepta, s'envia una petició POST a la API per assignar cada usuari a la llista d'amics del altre i guardar-ho a la base de dades.

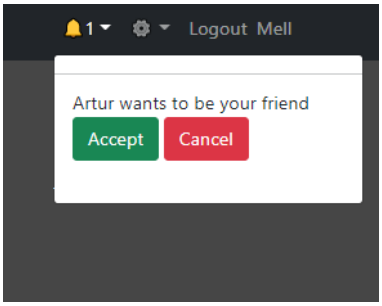

Fig. 6.7.7.1 Notificació de petició d'amistat. Font: Elaboració pròpia.

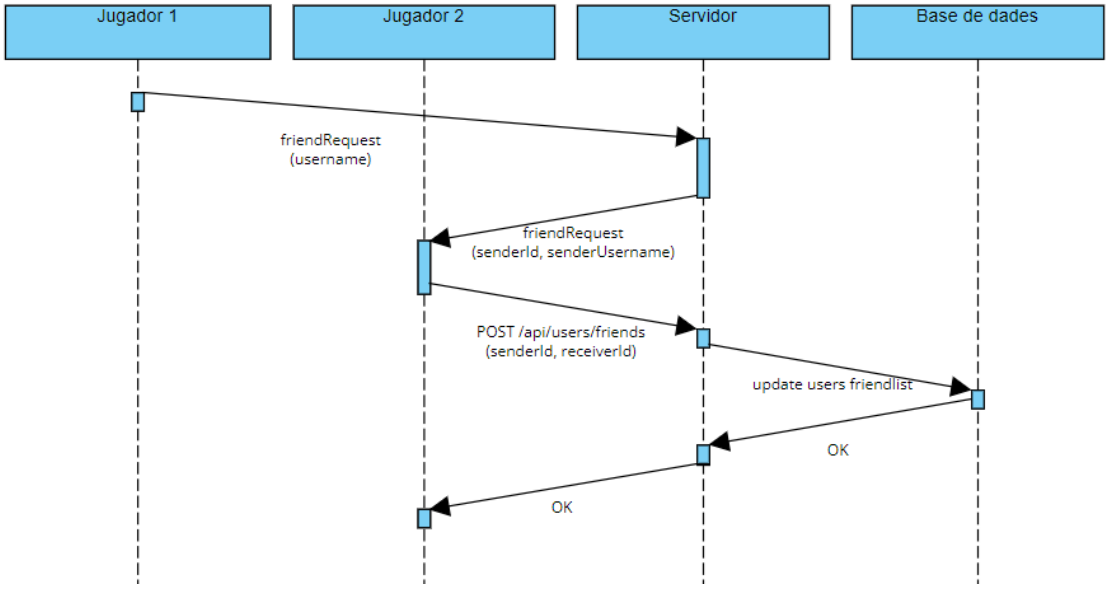

Fig. 6.7.7.2 Diagrama de petició d'amistat exitosa. Font: Elaboració pròpia.

Un cop afegit a la llista d'amics, el jugador pot decidir desafiar-lo. Llavors s'envia la notificació mitjançant sockets i l'altre jugador pot decidir acceptar o cancel·lar.

Fig. 6.7.7.3 Llistat d'amics. Font: Elaboració pròpia.

6.7.8 Botiga d'estils

Aquest apartat serveix com a incentiu i motivació perquè l'usuari vulgui jugar. A part de punts per ascendir en la classificació, l'usuari també rep coins al guanyar una partida classificatòria.

Aquests coins els pot utilitzar a la botiga per comprar diferents estils de les peces i tauler.

No és fàcil trobar conjunts d'estils de peces de lliure ús, per tant s'ha optat per dissenyar les peces de zero per aquest projecte i així evitar conflictes de drets d'ús per determinades imatges.

S'han dissenyat diversos conjunts, amb diferents estils i colors.

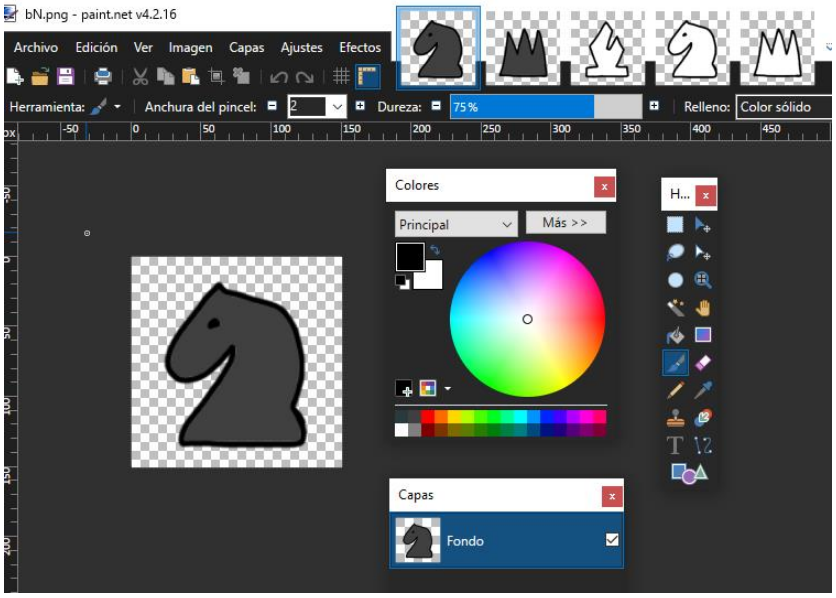

Fig. 6.7.8.1 Disseny d'una de les peces. Font: Elaboració pròpia.

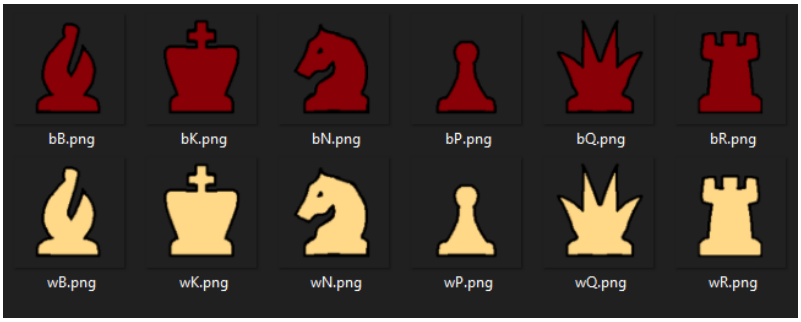

Fig. 6.7.8.2 Exemple de conjunt de peces. Font: Elaboració pròpia.

Quan l'usuari compra un objecte de la botiga, s'envia una petició POST amb l'usuari i item a comprar.

El servidor revisa que l'usuari tingui els coins necessaris i, si es així, els resta i afegeix l'objecte a la llista d'objectes comprats.

Aquest informació s'actualitza a la base de dades.

6.7.9 Ranking

A l'hora de buscar partida el jugador pot decidir si la vol casual o ranked.

Al jugar una partida casual no perd ni guanya res a l'acabar la partida, amb les partides ranked el jugador pot guanyar i perdre punts de la classificació a part de guanyar coins.

Només els jugadors registrats poden jugar partides ranked.

Qualsevol jugador pot consultar la llista de classificació, on es mostren tots els usuaris amb els seus punts corresponents.

Aquesta taula s'ha dissenyat amb un sistema de lligues. En funció dels punts de l'usuari, pertanyerà a una lliga o a una altra.

Les possibles lligues són Pawn, Bishop, Knight, Rook, Queen i King, sent aquesta última la més alta.

Quan es finalitza una partida ranked s'envia la informació al servidor i s'actualitzen els usuaris de la base de dades. El guanyador augmenta els seus coins i el seu elo i el perdedor perd elo.

6.7.10 Sessió d'usuari

Sense estar registrat, l'usuari pot disposar de les funcionalitats bàsiques de la web, pot jugar partides casual, consultar la classificació i jugar contra l'ordinador en el mode d'un jugador.

Però no pot jugar partides classificatòries, no pot tenir amics i no guanya coins i per tant tampoc pot comprar els objectes de la botiga.

Si l'usuari vol gaudir d'aquestes funcionalitats, s'ha de registrar, ha d'introduir el seu nom d'usuari i la seva contrasenya.

Al registrar-se, s'envia un POST a la API /api/users, es consulta que el nom d'usuari estigui disponible i que el format de les credencials és vàlid.

Tot seguit s'encripta la contrasenya i es crea un nou usuari a la base de dades.

Si l'operació ha estat exitosa, es retorna la informació a l'usuari juntament amb el JSON WEB TOKEN necessari per poder realitzar les altres operacions.

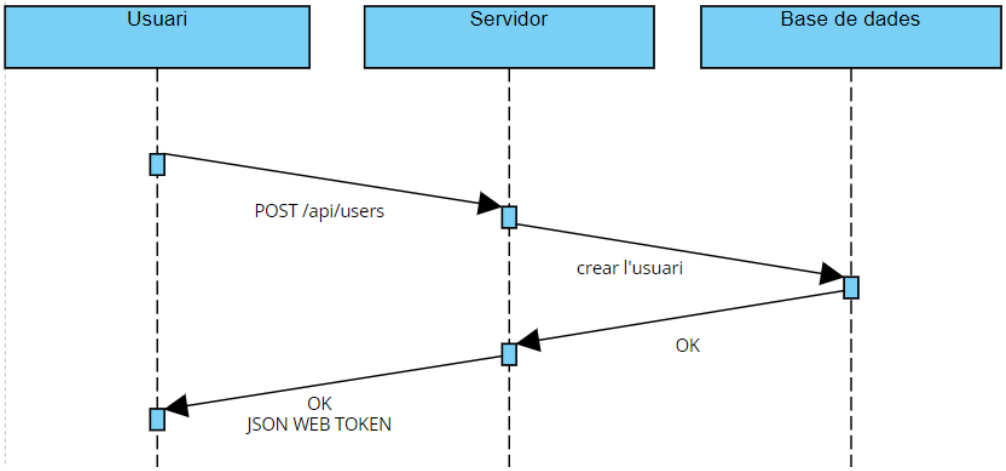

Fig. 6.7.10.1 Diagrama del registre d'un usuari. Font: Elaboració pròpia.

L'usuari pot decidir en qualsevol moment eliminar el seu compte, per fer-ho ha d'obrir el menú de configuracions ubicat a la barra de navegació i clicar sobre l'opció "eliminar compte".

Un cop ha clicat l'usuari ha de confirmar novament que desitja eliminar el compte.

En confirmar-ho s'envia una petició DELETE a l'API /api/users/:id. El servidor valida que el token de l'usuari és vàlid i que correspon a l'usuari que es vol eliminar. D'aquesta manera s'assegura que l'usuari només té permisos per eliminar el seu compte i no el d'altres.

Finalment s'actualitza la base de dades eliminant tota la informació relacionada amb l'usuari en qüestió.

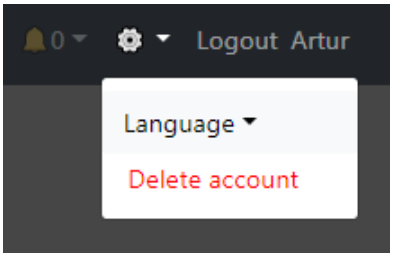

Fig. 6.7.10.2 Opció "eliminar compte" . Font: Elaboració pròpia.

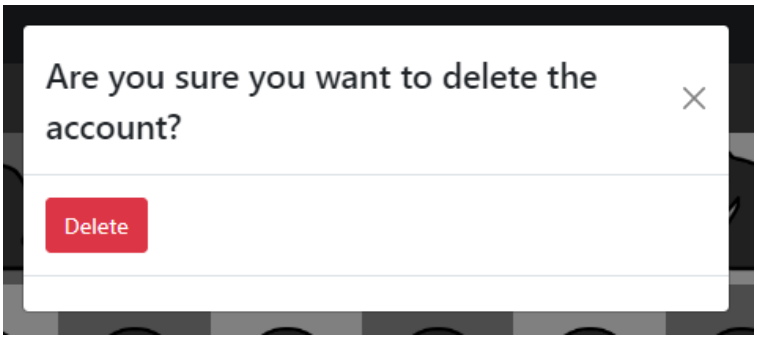

Fig. 6.7.10.3 Modal de confirmació per l'eliminació del compte. Font: Elaboració pròpia.

6.7.11 Gestió i canvi d'idiomes

Per poder gestionar els diferents llenguatges, definir el text per cada un d'ells i poder canviar i escollir l'idioma desitjat, s'utilitza react-i18next [14], un framework per React que facilita la internacionalització de l'aplicació web.

Està basat en i18next i és adequat per utilitzar-se en React.

Tots els texts de la web es defineixen en uns arxius separats i estan identificats per una paraula clau. Des de les parts de la web on es necessita incorporar text es crida amb l'identificador corresponent, d'aquesta manera en funció del llenguatge escollit es mostra un text o un altre.

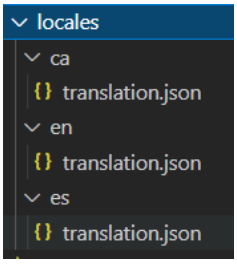

Fig. 6.7.11.1 Arxius de traducció. Font: Elaboració pròpia.

```
"Login":{
  "username":"Nom d'usuari",
  "password":"Contrasenya",
  "login":"Iniciar sessió",
  "register":"Registrar-se",
  "errors":{
    "wrongCredentials":"Usuari o contrasenya incorrectes",
    "wrongFormat":"El nom d'usuari no pot estar buit i la contrasenya ha de tenir més de 5 caràcters",
    "userExists":"Aquest nom d'usuari ja existeix",
    "GeneralError":"ha sorgit un error, intenta-ho de nou més tard"
  }
},
"Shop":{
  "noCoins":"Et falten monedes",
  "try":"Provar",
  "equip": "Equipar",
  "buy":"Comprar"
},
```

Fig. 6.7.11.2 Fragment del arxiu de traducció català. Font: Elaboració pròpia.

Un cop definides les traduccions, es pot utilitzar la funció t() per indicar la clau corresponent al text. En l'exemple de la Fig. 6.7.11.3 es pot veure l'ús de t("Shop.equip"), t("Shop.buy") i t("Shop.try") que correspondria al text "Equipar", "Comprar" i "Provar", tal i com es pot veure a les traduccions de la Fig. 6.7.11.2.

```
return (
  <Card className='mb-3'>
    <Card.Img variant="top" src={require('../assets/' + name + (isPiece ? '/sampleR.png' : '.png'))} />
    <Card.Body>
      {name === 'standard' || user?.info?.itemsPurchased.includes(name)
        ? <Button onClick={equip} size='sm' className='mb-2' variant="success">{t("Shop.equip")}</Button>
        : <>
          <Card.Title>100🏅</Card.Title>
          {user && <Button onClick={purchase} size='sm' className='mb-sm-2' variant="warning">{t("Shop.buy")}</Button>}

          <Button size='sm' onClick={handleTry} variant="secondary">{t("Shop.try")}</Button>
        </>
      }

    </Card.Body>
  </Card>
)
```

Fig. 6.7.11.3 Fragment del component ShopItem.jsx on es pot veure l'ús de la funció t("") per indicar el text a mostrar. Font: Elaboració pròpia.

6.8 Estructura

El codi està dividit en dos repositoris: el client i el servidor.

6.8.1 Client

En el client està el codi de la pròpia web, desenvolupada en React.

A continuació es descriuen els directoris i arxius importants.

- package.json: arxiu on s'indiquen les llibreries utilitzades i la seva versió, també configuracions i scripts del projecte.

- I18n.js: arxiu amb la configuració de i18next necessària per la gestió d'idiomes.

- App.js: component principal, conté totes les rutes per les diferents pagines de la web.

- assets: directori amb les imatges utilitzades, principalment els conjunts de peces.

- chessEngine: directori on es troba el web worker necessari per integrar la intel·ligència artificial.

- hooks: directori on estan els custom Hooks creats per simplificar el codi d'altres components.
  Els custom hooks desenvolupats són:
  useInterval: serveix per iniciar un interval, s'ha d'indicar el delay i el callback.
  useClock: serveix per compactar la funcionalitat dels rellotges dels jugadors en una partida online. Ofereix funcions per iniciar, aturar i obtenir el temps.
  usePromotion: serveix per separar la complexitat de la promoció d'un peó. Aporta totes les funcions necessàries.
  useCustomBoard: gestiona el canvi d'estils de peces i taulells.
  useResponsiveBoard: permet adaptar la mida del taulell segons la mida de la pantalla.

- pages: directori on es troben totes les pagines de la web
  Les pàgines creades són:
  Home.jsx
  Login.jsx
  Register.jsx
  Game.jsx
  ComputerGame.jsx
  Ranking.jsx
  Shop.jsx
  Friends.jsx

- components: directori on es troben tots els components React utilitzats en les pagines.
  Aquests components són:
  CustomNav.jsx
  ErrorMessage.jsx
  GameResult.jsx
  HomeBoard.jsx
  Promotion.jsx
  ShopItem.jsx
  VoiceControl.jsx

- services: directori que conté els arxius login.js i users.js, aquests arxius tenen els mètodes necessaris per connectar-se amb el servidor mitjançant peticions a la API.

- socket: directori que conté l'arxiu d'inicialització del socket.

- tests: directori on es troben diversos tests bàsics.

- utils: directori on hi ha arxius que proporcionen funcions generals.

6.8.2 Servidor

En el servidor es troba el codi de la API, el control dels missatges socket i la connexió amb la base de dades.

A continuació es descriuen els directoris i arxius importants.

- package.json: arxiu on s'indiquen les llibreries utilitzades i la seva versió, també configuracions i scripts del projecte.

- mongodb.js: arxiu de configuració i connexió a la base de dades.

- Models: directori on es troben els models i esquemes de la base de dades.

- Controllers: conté els routers de la API.

- middleware: middleware de la API com gestió d'errors i validació d'usuari.

- Socket: estàn els escoltadors de tots els missatges que els clients poden enviar a través dels sockets: buscar partida, fer un moviment, desafiar un amic, etc...

- server.js: arxiu principal del servidor des de on es criden els routers de la API i els escoltadors de missatges socket.

- .env: arxiu per definir les variables d'entorn del servidor com poden ser el text de connexió a la base de dades o el text secret per encriptar els tokens.

6.9 Llistat de llibreries utilitzades

Client

- axios
- bootstrap
- react-boostrap
- i18next
- i18next-browser-languagedetector
- i18next-http-backend
- react-i18next
- js-chess-engine
- react-chessboard
- chess.js
- react-router-dom
- react-speech-recognition
- socket.io-client

Servidor

- bcrypt
- cors
- dotenv
- express
- jsonwebtoken
- mongoose
- uuid
- socket.io

7 Desplegament

Per realitzar el desplegament s'ha escollit fer-ho amb Heroku.

Heroku és una plataforma com a servei (PaaS) de computació en el núvol. Disposa d'uns contenidors virtuals anomenats dynos que s'encarreguen de mantenir i executar les aplicacions.

Aquests contenidors són completament escalables, per tant això no és un problema.

Per tal de desplegar l'aplicació, primer s'ha de crear una aplicació a Heroku i s'ha d'indicar la zona geogràfica i altres configuracions.

Un cop creada, s'ha de configurar el enllaç remot amb l'aplicació a traves de git.

I, per últim, simplement fa falta fer push a aquesta connexió remota i automàticament s'executa la comanda npm install per instal·lar les dependències de l'aplicació. A continuació executarà l'script de run, per tant es necessari haver configurat aquest script anteriorment.

Una opció és desplegar els dos repositoris, tant el client com el servidor, en aplicacions diferents.

Però s'ha decidit fer-ho en una sola aplicació.

Per fer això, s'ha fet build del codi del client, d'aquesta manera s'ha generat el codi de producció estàtic de l'aplicació client.

Aquest codi generat en la carpeta build s'ha mogut dins del repositori del servidor i s'ha configurat el servidor per tal de servir aquests fitxers directament des del mateix servidor.

Això es coneix com a server side rendering i té moltes avantatges:

- És molt més ràpid. Al renderitzar-se en el servidor, l'usuari pot visualitzar la pàgina de forma molt més eficient.
- Al realitzar una partida d'escacs s'estan enviant constantment missatges al servidor. D'aquesta manera, el temps de resposta dels missatges es molt menor i, per tant, la partida és més fluida.

- Les URL de peticions a la API o de connexió amb el socket del servidor poden ser dinàmiques. Al renderitzar-se l'aplicació client en el mateix servidor, no fa falta canviar les URL.
- No fa falta desplegar dues aplicacions, només el servidor.

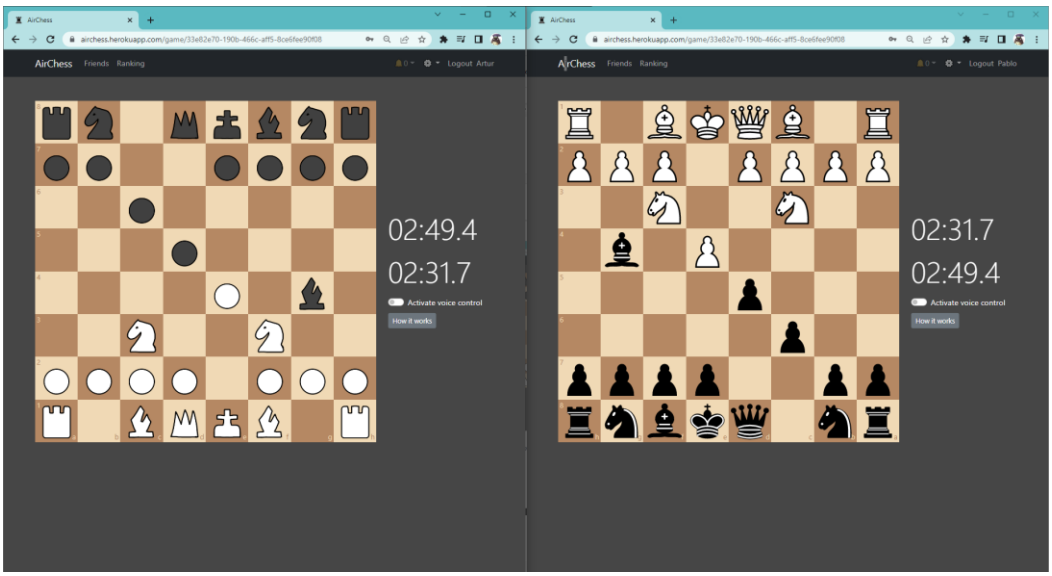

Fig. 7.1 Partida online entre dos usuaris a traves de l'aplicació desplegada a https://airchess.herokuapp.com. Font: Elaboració pròpia.

Un cop desplegada l'aplicació, s'han de configurar les variables d'entorn. En local, aquestes variables estan en el fitxer .env., però aquest fitxer no es publica a git ja que conté informació privada com la clau secreta o la connexió amb la base de dades.

Per tant, s'han d'introduir aquestes variables novament en l'aplicació de Heroku.

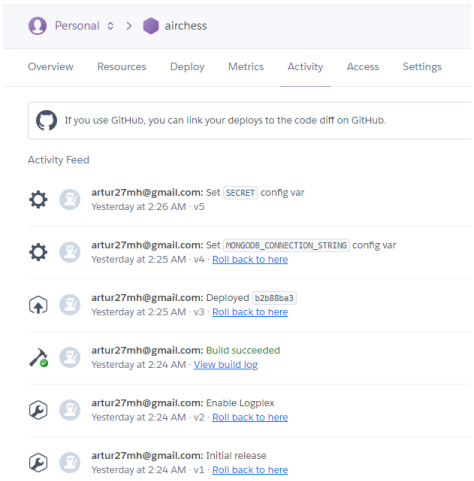

Fig. 7.2 Historial d'activitat del desplegament de la web. Font: heroku.com

8 Dificultats

A continuació es comenten els principals problemes, imprevistos o situacions complicades que s'han trobat en la realització del projecte.

Aquestes dificultats finalment s'han pogut solucionar sense complicacions gràcies al temps de marge que es va calcular a l'hora de la planificació.

8.1 Reconeixement per veu

S'han realitzat moltes proves amb diferents llibreries o sistemes fins trobar l'elecció final.

Això es deu a que moltes de les opcions requerien prémer un botó per indicar que l'usuari vol parlar i, en aquest cas, l'aplicació havia d'estar escoltant de forma continuada.

A més, s'ha hagut de desenvolupar un sistema de processament del text per obtenir amb el major percentatge d'èxit possible el moviment dit per l'usuari.

Tot això ha suposat un repte i moltes hores de proves per arribar al resultat final.

8.2 Sincronització de rellotges en el mode multijugador

La sincronització dels temps de cada jugador ha estat un problema important que no es va tenir gens en compte a l'hora de realitzar la planificació.

No es va considerar com una possible problemàtica i, en canvi, ha resultat ser un dels punts de bloqueig més importants en el desenvolupament de l'aplicació.

El fet que l'enviament de missatges no sigui instantani genera moltes situacions de perill. Pot passar que en la pantalla d'un usuari el contador del seu oponent ja ha arribat a zero i, per tant, ha guanyat, quan en canvi l'altre usuari ja ha realitzat el moviment i, per tant, ell també ha guanyat.

Si això no es soluciona, a part d'aquestes situacions també es genera una disparitat en el temps que té cada usuari i cada cop es fa més gran aquesta diferència.

Això finalment s'ha solucionat amb un sistema de sincronització. Aquest sistema no es perfecte i pot ser millorat, tal i com s'indica en l'apartat de futures ampliacions.

8.3 Temps de formació

El temps necessari per estudiar, provar i entendre les diferents tecnologies utilitzades ha estat major del esperat en proporció al temps de desenvolupament.

Es va planificar un temps de formació menor al temps de desenvolupament i, a l'hora de la veritat, ha resultat bastant semblant el temps dedicat a cada tasca.

Això ha suposat que, a l'hora de començar a desenvolupar, ja es tenien les idees molt més clares gràcies a haver estat estudiant i provant sobre el tema.

9 Conclusions

En aquest apartat es presenten les conclusions finals del projecte. Es farà una avaluació del compliment dels objectius i requeriments definits així com un resum del resultat obtingut.

S'han complert tots els objectius i requeriments definits en l'apartat de planificació. S'ha desenvolupat una aplicació web completament funcional que permet a l'usuari jugar partides d'escacs tant contra la màquina com contra altres usuaris de forma online.

L'usuari es pot registrar i crear el seu compte, amb el qual obté accés a altres funcionalitats com partides classificatòries o possibilitat d'afegir i desafiar amics. La seva informació es guarda a la base de dades.

Un dels objectius més importants fa referencia a la possibilitat d'indicar els moviments a realitzar mitjançant la veu. Això pot ser molt útil per aquelles persones que per desgràcia tenen una discapacitat que els impedeix jugar de forma convencional.

El sistema de control per veu s'ha desenvolupat amb èxit, s'ha testejat amb diferents usuaris i han pogut jugar de forma eficaç. A més de poder dictar els moviments, l'usuari també pot indicar amb la veu que vol buscar una nova partida, d'aquesta manera pot seguir jugant tantes partides com vulgui.

A més, s'han incorporat funcionalitats com la botiga o el ranking per donar un punt de motivació a l'usuari.

El framework i tecnologies escollides han estat les adequades i han permès desenvolupar totes les funcionalitats necessàries tal i com s'havia planejat. Per tant, en cas d'haver de repetir aquest projecte o un de similar es considera una bona opció fer-ho amb les tecnologies utilitzades.

10 Possibles ampliacions

El proper pas és sens dubte desenvolupar un millor sistema de testing automatitzat del producte actual.

Un bon sistema de testing és indispensable per garantir el correcte manteniment i escalabilitat d'un producte de software.

Altres ampliacions poden consistir en afegir funcionalitats. Alguns exemples de possibles noves funcionalitats són:

- Possibilitat de crear i participar en tornejos.
- Més modes de joc amb variacions.
- Puzzles on l'usuari ha de trobar el millor moviment en una posició concreta.
- Xat

Una altre ampliació important consisteix en afegir seguretat i validació de les partides. Actualment el sistema es fia bastant de la informació provinent de l'usuari. Una bona pràctica pot ser afegir validació a la part del servidor, tant del moviment en si com de la data en què es realitza.

11 Bibliografia

[1] A. A. Macdonell, «Art. XIII.—The Origin and Early History of Chess,» de *Journal of the Royal Asiatic Society of Great Britain & Ireland*, 1898, p. 117–141.

[2] FIDE, «International chess federation,» 30 Decembre 2021. [En línia]. Available: https://handbook.fide.com/chapter/E012018. [Últim accés: Gener 2022].

[3] «Lichess,» 9 Gener 2022. [En línia]. Available: https://lichess.org/. [Últim accés: Gener 2022].

[4] «Chess.com,» 20 Febrer 2017. [En línia]. Available: https://www.chess.com/. [Últim accés: Gener 2022].

[5] F. Gliga i P. I. Flesner, «Cognitive Benefits of Chess Training in Novice Children,» de *Procedia - Social and Behavioral Sciences*, 2014, pp. 962-967.

[6] R. Aciego, L. García i M. Betancort, «The Benefits of Chess for the Intellectual and Social-Emotional Enrichment in Schoolchildren,» de *The Spanish journal of psychology*, 2012, p. 551–559.

[7] «LChess,» 27 Juny 2021. [En línia]. Available: https://gamehands.net/LChess/. [Últim accés: Gener 2022].

[8] «Reactjs,» [En línia]. Available: https://reactjs.org/. [Últim accés: Març 2022].

[9] Stackoverflow , «Insights Survey 2021,» 2021. [En línia]. Available: https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks. [Últim accés: Març 2022].

[10] Github, «chess.js,» [En línia]. Available: https://github.com/jhlywa/chess.js/blob/master/README.md. [Últim accés: Març 2022].

[11] Github, «react_chessboard,» [En línia]. Available: https://github.com/Clariity/react-chessboard. [Últim accés: Març 2022].

[12] Mozilla, «Web Sockets,» [En línia]. Available: https://developer.mozilla.org/es/docs/Web/API/WebSockets_API. [Últim accés: 15 Abril 2022].

[13] «Socket.io,» [En línia]. Available: https://socket.io/. [Últim accés: 16 Abril 2022].

[14] «react-i18next,» [En línia]. Available: https://react.i18next.com/. [Últim accés: 16 Abril 2022].