TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

LEAGUE OF GOOSE:
EL JOC DE L'OCA 2.0

Memòria

ADRIÁN RODRÍGUEZ LETELLIER
TUTORA: CATALINA JUAN NADAL

2022-2023

TecnoCampus
Mataró-Maresme

Dedicatòria

A totes aquelles persones que creuen que ja no tenen un nen dins seu.

Mai és tard per passar-ho bé.

Agraïments

A la meva família i a la meva parella, que són les úniques persones capaces de tolerar un nen com jo a totes hores.

Abstract

New times bring new technologies, and with those, it is possible to reimagine any game.

League of Goose is the reinvention of the classic Goose Game, where, following new rules, new boards, and new challenges, players can have joy by playing this well-known table game with modern standards.

The project is about the realization of a completely functional technological platform where different players can play remotely and simultaneously.

Resum

Els nous temps porten noves tecnologies, i amb aquestes es pot reimaginar qualsevol joc.

League of Goose és la reinvenció del clàssic Joc de l'Oca, on seguint noves normes, nous taulells, i nous reptes, els jugadors poden gaudir d'aquest conegut joc de taula amb estàndards moderns.

El projecte consisteix en la realització d'una plataforma tecnològica completament funcional on diferents jugadors poden jugar entre ells de forma remota i simultània.

Resumen

Nuevos tiempos traen nuevas tecnologías, y con estas se puede concebir cualquier juego.

League of Goose es la reinvención del clásico Juego de la Oca, donde siguiendo nuevas normas, nuevos tableros, y nuevos retos, los jugadores pueden disfrutar de este conocido juego de mesa con estándares modernos.

El proyecto consistí en la realización de una plataforma tecnológica completamente funcional donde diferentes jugadores pueden jugar entre ellos de forma remota y simultánea.

Índex

Índex de figures

Glossari de termes

WEB APP	Aplicació desenvolupada per navegadors web.
API REST	Arquitectura d'intercanvi de dades a través d'internet (protocol HTTP)
BACKEND	Codi que s'executa en el servidor, habitualment gestionant peticions de clients.
WEBSOCKET	Tecnologia que proporciona comunicació bidireccional full-dúplex sobre un socket TCP.
DB	“Data Base” (Base de Dades)
VM	“Virtual Machine” (Màquina Virtual)
DNS	“Domain Name Sistem” (Sistema de Noms de Domini)
superset	Framework o llenguatge que parteix d’un altre afegint funcionalitats
AJAX	“Asynchronous JavaScript and XML”. Comunicació asíncrona entre aplicacions web i servidors
Dockerfile	Arxiu estàndard de configuració per la creació d’imatges basades en Docker
Nginx	Servidor web
Kubernetes	Plataforma de codi obert per administrar càrregues de treball i serveis basats en Docker

1. Introducció

Aquest projecte neix d'un altre de l'any 2021-2022, titulat "Aplicació mòbil per a jugar al joc de l'Oca", on el seu és crear un joc de la "Oca multijugador per torns". La part multijugador del mateix es va reservar com a ampliació, donat que només implementava una versió local.

Així doncs, l'objectiu d'aquest projecte és crear una versió 2.0 del joc clàssic de l'Oca, renovant-lo amb idees del segle XXI, i dotant-lo d'interacció multijugador en línia simultani concurrent.

El joc de l'Oca existeix des de fa cinc-cents anys, i tant el joc original com les seves variants han perdurat gairebé immutades als nostres dies. Tenint en compte això, una actualització al joc seria un producte potencial per a poder incloure en el mercat de videojocs. Cal que es basi en nous efectes, noves formes de plantejar la partida, i sobretot interactivitat entre els jugadors pot dotar de noves capes de complexitat, estratègia, i diversió.

2. Estudi previ

2.1. Context

El Joc de l'Oca "original" fa referència al *Gioco dell'oca*, datat de 1500 dC, consisteix en un taulell on:

- Hi ha 63 caselles.
- Si el jugador cau en la casella 58, "Mort", torna a començar.
- S'ha de caure a la casella 63 de forma exacta, si no es conta enrere.
  Quan es cau en aquesta casella de forma exacta es guanya la partida, però es pot tornar a començar (donant sentit a la forma en espiral). Si no es cau de forma exacte, el nombre restant es compte cap enrere.
- Existeixen caselles "Oca" repartides pel taulell, on si un jugador cau dintre repeteix el moviment amb el número del dau llençat (en altres versions es mou fins a la següent "Oca", i algunes ocasions es fa una tirada extra després de moure's fins a la següent "Oca").
- Hi poden existir algunes caselles especials (poden variar lleugerament entre versions), amb la seva norma corresponent, tals com: el "Laberint" (d'un a dos torns sense jugar) o el "Pont" (caure en un pont et porta a l'altre).
- Els jugadors per avançar han de tirar un dau (en algunes versions es tiren dos) per arribar fins al final.

Es destaca que en cap cas hi ha ni estratègia ni interactivitat entre els jugadors. Tenint en compte això, tècnicament els jugadors poden jugar en diferents taulells de forma simultània. Qui arribi en menys torns seria el guanyador. Seria com si diverses persones juguessin al joc del Solitari i guanyés qui el solucioni abans.

2.2. Antecedents

Qualsevol versió mai creada representa un precedent per aquest projecte. Paradoxalment, encara que hi hagi infinites versions al llarg de la història, la gràcia del Joc de l'Oca és que rarament varia molt de l'original, obviant els canvis visuals/estètics.

Tot i això, existeixen jocs de l'Oca en línia multijugador, com el de juegosjuegos.com [1], però no canvien cap aspecte natiu del joc a part de la plataforma. El fonament d'aquest treball implica fer-ne una variació de la versió original que mantingui l'essència del joc però dotant-lo d'aspectes interactius actuals.

Tenint això en compte, s'entendrà com antecedent qualsevol Joc de l'Oca amb les regles tradicionals, independentment de la plataforma, i obviant el seu apartat visual (que sempre es considerarà el mateix perquè no representa cap rellevància pel que fa a la jugabilitat).

2.2. Necessitats d'informació

Respecte al disseny intrínsec del videojoc, és necessari una cerca d'informació adient per a poder generar unes especificacions funcionals i tecnològiques pel joc que fossin les adients, i no simplement posar un conjunt aleatori de funcionalitats.

Per això, es va fer una reunió amb un professor del grau de Disseny de Videojocs al TecnoCampus, en Xavier Figueres (annex 1, 8.1), per preguntar aspectes claus a tenir en compte a l'hora de crear un videojoc des del punt de vista de disseny.

Algunes aplicacions existents poden ser:

- Joc de l'Oca de juegosjuegos.com.
- Joc de l'Oca de la App Store [2].

La primera està desenvolupada per navegador, mentre que la segona està com aplicació pel mòbil.

Ambdues funcionen amb multijugador en línia, fent servir la lògica i normes del Joc de l'Oca clàssic de taulell.

3. Objectius i abast

3.1. Objectius principals:

El projecte consisteix a crear una plataforma capaç de sustentar tot allò que sigui necessari per fer un "Joc de l'Oca 2.0". El "client" que especifica de quina manera s'ha de fer exactament és l'autor del projecte.

3.1.1. Objectius principals del client:

- Crear un MVP d'una versió actualitzada del "Joc de l'Oca", accessible des d'un navegador web.
- Crear un MVP d'una connexió client/servidor, on el client és el jugador dins del taulell virtual i el servidor és l'API/websocket que gestiona el joc.
- Crear una base de dades on emmagatzemar:
    - Jugadors registrats (usuaris)
    - Registre de partides
    - Classificació general
- Crear una connexió MVP entre els clients, el servidor i la base de dades.

3.1.2. Objectius principals del producte:

En aquest punt apareixen tots els objectius que formen part del producte respecte al portal web, que serà el producte principal:

- Permetre registrar-se i iniciar sessió.
- Permetre crear partides amb altres jugadors.
- Permetre escollir entre aspectes personalitzats per fer servir dins del joc (només en l'àmbit gràfic).
- Crear una tenda on comprar aspectes personalitzats.
- Permetre jugar contra la màquina (no una IA, sinó un algoritme de decisió).

3.2. Abast

3.2.1. Públic potencial

- Donat que es realitza un joc per gaudir d'ell, les premisses a seguir impliquen:
- Laïcisme: la religió no ha de quedar representada de cap manera en el producte.
- Sense distinció de gènere: hi juguen oques, així que no té sentit intentar afegir skins amb temàtiques estereotipades de gènere.
- Temàtica universal: el contingut ha de ser entenedor per tota la població, i la millor manera de representar-ho és fent referència als elements que apareixen al planeta. Es fan servir temes com aire, foc, aigua, terra, llamp, etc. per representar els aspectes del joc.

Forma part de públic potencial qualsevol persona amb accés a un dispositiu intel·ligent (dispositiu mòbil o ordinador) amb capacitats motrius bàsiques.

No es tindran en compte grups socials amb discapacitats severes, com per exemple capacitat visual molt reduïda (a causa de les dimensions del projecte i la forma en la que es juga en aquesta versió del joc).

3.2.2. Públic principal del producte

Tenint en compte els conceptes del punt anterior, el target principal està determinat per aquests perfils:

- demogràfic: persona (sense distinció de gènere) d'entre deu i quaranta-quatre anys arreu del món. Aquest interval és definit per un estudi que va fer l'*Entertainment Software Associaton* [3], on s'indiquen els grups per edats de persones que juguen a videojocs. Ja que es vol fer accessible per un gran públic s'ha escollit aquesta franja d'edat que engloba els tres grups més grans de jugadors/es.
- sociocultural: nivell econòmic de classe treballadora (classe mitjana), sense distinció de cultura ni religió, amb costums associats al consumisme, i amb accés a un dispositiu intel·ligent.

- digital: actiu en el món digital, que sap fer servir xarxes socials (en un grau o un altre), i que juga videojocs de forma habitual/setmanal.

Com s'ha esmentat anteriorment, es realitza un joc sense distinció de gènere, laic, sense estigmes ni prejudicis. S'eviten tots els afers que poden crear disputa entre els jugadors pel contingut del joc.

L'únic requisit que es demana és l'anglès, que és l'idioma vehicular del joc; però es pot jugar sense saber-ne molt. El disseny s'ha concebut per ser el més senzill possible per facilitar l'accés al joc

4. Metodologia

Per a la realització d'aquest treball s'ha optat per realitzar processos en cascada de forma cíclica i que es poden avaluar individualment, anomenada "Metodologia en Espiral".

S'ha escollit aquesta metodologia degut a la seva naturalesa de llista de tasques (on simplement es van seguint passes d'un esquema) i a la seva capacitat d'autoavaluació (on al finalitzar les tasques es pot saber si la seva realització ha aportat valor).

A part, troba que és una forma de treballar òptima quan no se sap amb certesa si el conjunt de tasques tindrà el resultat esperat. Això és degut al fet que donada una idea inicial es pot saber, gràcies a la seva avaluació final, si la seva resolució ha estat profitosa o no, podent directament descartar el procés si no ho ha estat.

Per dur a terme cada procés, s'han seguit aquestes pautes:

- Recerca de tecnologia, plataformes, i entorns de desenvolupament
    - Concreció d'objectius
    - Cerca de tecnologia/plataforma/entorn que satisfaci els objectius
    - Creació d'una taula de pros i contres, valorant l'efectivitat de l'eina
    - Si l'eina ha passat una iteració i després no acaba sent l'escollida, passa a formar part de les alternatives (en el cas de continuar sent viable).
- Cerca d'informació
    - Lectura preliminar de la font
    - Si sembla rellevant, important, i fiable, s'apunta al conjunt de fonts emprades
    - Anotació dels punts destacats que s'empraran al projecte
    - Disseny del model del domini/base de dades
    - Definició de classes necessàries
    - Definició dels seus components
    - Definició de les responsabilitats entre les classes

- Realització de tasques
    - Creació d'una llista amb totes les passes a seguir per dur a terme la tasca, indicant el seu nivell de complexitat (0-10)
    - De totes les passes amb complexitat alta (una valoració igual o superior a set), dedicar un temps afegit per avaluar la seva viabilitat
    - En el cas de trobar massa complexa alguna subtasca: reformar-la, dividir-la, o si escau, repensar la tasca sencera
    - Començar a realitzar totes les passes, anotant el temps emprat per cadascuna d'aquestes

Un cop finalitzat cada procés, s'ha fet una revisió del seu estat i, en el cas d'estar bé, s'ha tornat a començar al primer punt de cadascun d'ells, iterant sobre el ja existent o bé fent un de nou.

Respecte a la coordinació amb la dissenyadora, Roser Butrón Isern [4], s'ha acordat fer servir un Trello on es vagin apuntant les hores fetes de cada tasca de forma precisa.

5. Dissenys i il·lustracions

S'ha considerat que, tot i que el projecte a desenvolupar és de caràcter tecnològic, es propici contractar els dissenys i il·lustracions de la dissenyadora Roser Butrón Isern.

La justificació radica en la necessitat que el producte final és un joc. Encara que tècnicament funcioni, cal que sigui atractiu i clar perquè els jugadors tinguin la millor experiència de joc possible.

Després de revisar el seu dossier [4] d'entre diverses candidates (com Sophie MC Pike [5], Alice Romano [6], Emily Schofield [7], Jessica Elena, [8], o Jessica Smith [9]), s'ha considerat que és la persona indicada per fer el treball artístic. Això ha estat degut al fet que el seu estil encaixava molt amb la idea artística inicial del projecte, és l'única que actualment viu a Catalunya (facilitant molt tant la comunicació presencial), i havia rebut opinions molt bones d'ella en l'àmbit professionalment.

Després d'explicar-li el projecte, la dimensió, i la definició dels temes per totes les il·lustracions a dur a terme, va crear un primer pressupost que es va acceptar (el qual finalment queda com la factura que apareix al pressupost del projecte), va començar el seu treball a partir del dia cinc de febrer de 2023.

Posteriorment, es van recollir les tasques individuals que anava fent i el seu progrés fent servir la plataforma Trello [10]. Aquesta plataforma ofereix un lloc web on poder veure i fer un seguiment de diferents tasques entre un equip de persones, sent una bona eina per mantenir una consistència i coneixement del treball per fer, en procés, finalitzat, o finalment entregat.

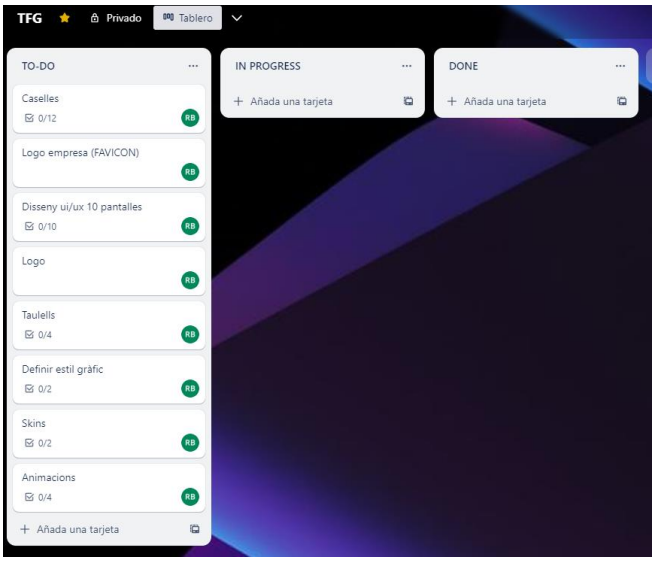

Fig. 5.1. Taulell de Trello inicial. Font: Elaboració pròpia.

Després de tres setmanes, es van mantenir reunions setmanals, demanant ajustos si eren necessaris i afegint les il·lustracions finals al projecte.

Al final, degut a canvis en els terminis i objectius encara plausibles del projecte, es va finalitzar amb la realització d'aquestes il·lustracions i dissenys.

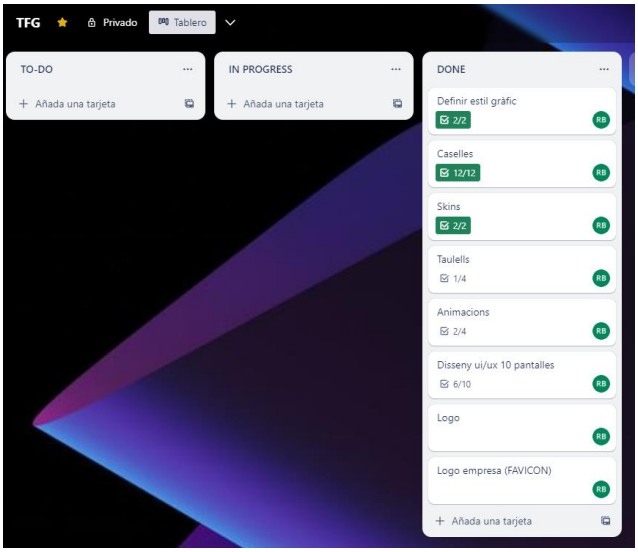

Fig. 5.2. Taulell de Trello final. Font: Elaboració pròpia.

Aquestes reduccions de recursos són:

- Un taulell de quatre: com que no s'ha desenvolupat finalment un sistema d'aleatorització de taulells, només ha estat necessari un d'ells.
- Dues de quatre animacions: només s'han necessitat animacions pels aspectes de les oques i els daus; no han estat necessàries animacions per una suposada finestra de càrrega ni una animació especial per a quan els jugadors cauen en la mateixa casella.

6. Definició de requisits funcionals i tecnològics

6.1. Requisits tecnològics

En primer lloc, s'ha escollit aquests requisits tecnològics pel desenvolupament del projecte i la seva plataforma:

- Fer servir React pel desenvolupament del portal web.
- Fer servir Node 16 com a servidor entre els clients fent servir la llibreria socket.io.
- Fer servir Flask com a REST API backend.
- Fer servir MongoDB com a base de dades no relacional, accedint des de l'API prèviament esmentada en Flask. Aquesta decisió s'ha pres tenint en compte que la magnitud de la mateixa no requereix moltes relacions entre moltes taules.
- Fer servir Docker com a entorn de les aplicacions per poder-les reproduir i desplegar en qualsevol entorn (desenvolupament i producció en aquest cas).
- Fer servir Google Cloud com a servei en el núvol per hostejar el portal web i els serveis pertinents.

6.2. Requisits funcionals

En aquest projecte no hi ha un client concret que hagi donat indicacions concises, ni tampoc hi ha requisits específics indicats en la definició inicial del tema d'aquest TFG. És per això, que per la generació d'aquests s'ha optat per generar pluges d'idees, contactes amb experts en videojocs (Xavier Figueras, professor de la carrera de Disseny i producció de videojocs), enquestes a públic i gustos personals.

Els requeriments del portal web (és a dir, envers l'usuari) són:

- Permetre registrar-se al sistema amb usuari, correu electrònic, i contrasenya.
- Permetre iniciar sessió al sistema amb correu electrònic i contrasenya.
- Permetre comprar aspectes personalitzables en la teva amb moneda virtual.
- Permetre veure un rànquing amb la posició dels jugadors.
- Permetre jugar al Joc de Oca online.

D'altra banda, es defineix quin serà el flux d'execució normal a partir d'entrar al portal web del joc (annex 2, 8.2).

Amb aquest flux es crea una llista de requeriments del joc. L'aportació de valor al joc i la seva dificultat d'implementació són els paràmetres que es tindran en compte per seleccionar-los o descartar-los quan es faci la selecció final. Aquests paràmetres van de l'u al deu, ambdós inclosos; on u és la dificultat més baixa i el valor més petit i el deu és la dificultat més alta i el valor més gran.

- Permetre escollir al jugador tres daus trucats per poder emprar a la partida.

  Valor: 8 Dificultat: 1

- Aleatoritzar les caselles del taulell en la seva inicialització.

  Valor: 8 Dificultat: 1

- Definir que per guanyar el jugador ha de caure de manera exacta a la casella final.

  Valor: 4 Dificultat: 1

- Aleatoritzar el tipus de taulell en la seva inicialització.

  Valor: 7 Dificultat: 4

- Aleatoritzar el tipus de mini joc que tindran els jugadors quan caiguin en una casella ocupada.

  Valor: 7 Dificultat: 4

- Definir els efectes de les caselles.

  Valor: 7 Dificultat: 4

7. Desenvolupament

Durant el procés de desenvolupament d'aquest projecte han esdevingut dues etapes: la planificació, el desenvolupament, i la posada en producció.

7.1. Planificació de l'arquitectura

En primer és realitzar la planificació de l'arquitectura, que seguint aquest diagrama:

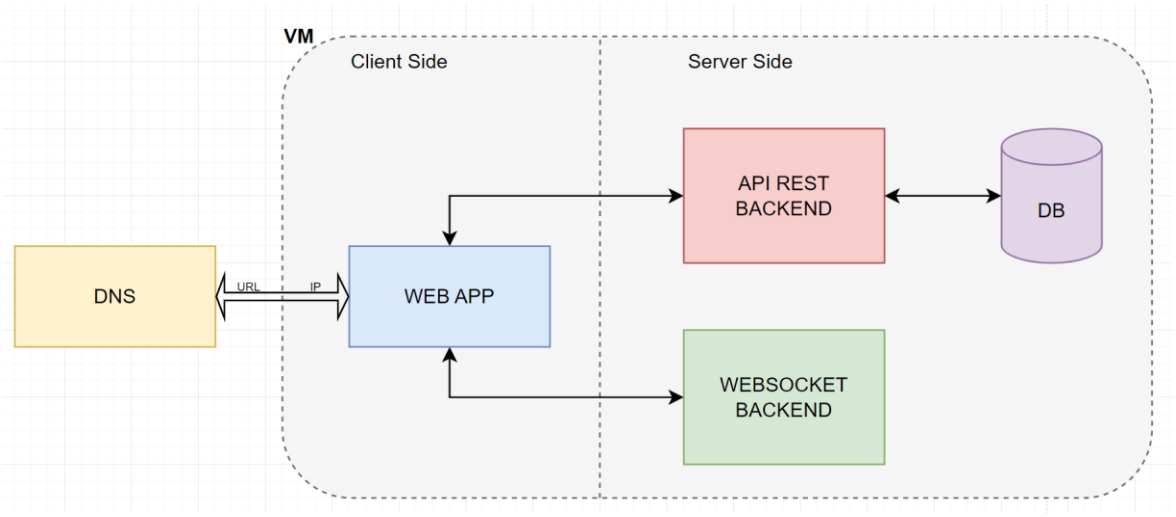

Fig. 7.1.1. Diagrama conceptual de la planificació de l'arquitectura inicial. Font: Elaboració pròpia.

Com es pot veure al diagrama, es busca una màquina virtual (VM) que contingui els diferents serveis que són necessaris per al funcionament del projecte. Aquests són:

WEB APP: aplicació web on els jugadors (clients) jugaran al joc com a tal.

API REST BACKEND: API REST on les peticions que emetrà l'aplicació web seran processades i emmagatzemades en la base de dades, com per exemple el registre, la identificació, o les transaccions d'inventari de cada client.

DB: Base de dades on emmagatzemar la informació persistent de cada jugador, com les seves credencials o inventari.

WEBSOCKET BACKEND: Servidor que manté comunicacions tipus websocket entre els diferents clients i ell mateix. Serveix per coordinar els esdeveniments que transcorrin dins del joc, com tirar daus o moure les peces.

VM: Una màquina virtual capaç d'executar tots aquests serveis dins d'un proveïdor de serveis de hosting.

DNS: Proveïdor DNS que traduirà la IP generada per la instància de la VM a un URL comprensible pels jugadors.

7.2. Docker

Des del primer moment del desenvolupament s'ha fet servir Docker per fer contenidors de tots els serveis que són necessaris per el projecte.

Aquestes reduccions de recursos són:

- Aïllament de les aplicacions: No hi ha entramats de fitxers ni directoris entre els diferents contenidors, propiciant individualitat absoluta dintre de cadascun d'ells.
- Entorns dedicats: gràcies a l'aïllament, es poden definir paràmetres i entorns per cadascun dels contenidors.
- Monitoratge i control: els serveis es poden encendre, apagar, reiniciar, recrear, i veure els registres que generen per veure possibles errors en execució o simplement controlar el flux d'informació que travessa els serveis.

Gràcies a tots aquests avantatges, tots els serveis (aplicació web, API REST, servidor de websockets i la base de dades) han estat implementades fent servir aquesta tecnologia.

De la mateixa manera, s'ha emprat l'ús de Portainer [11]. És una aplicació prefabricada dintre d'un altre contenidor Docker que està preparada per proporcionar un entorn visual on es poden controlar tots els contenidors, imatges, i serveis actius (i no actius) dintre d'una màquina. Ha estat una gran eina durant tot el desenvolupament del projecte.

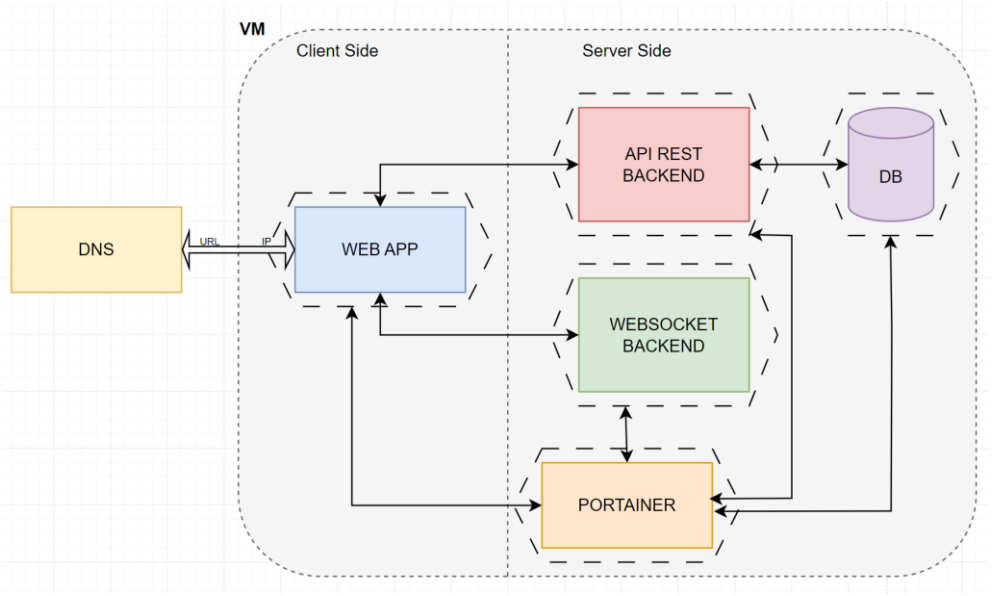

Fig. 6.2.1. Diagrama conceptual de l'arquitectura amb implementació de Docker i Portainer. Font: Elaboració pròpia.

Com es pot veure en la figura 6.2, els serveis s'executen dintre de contenidors, als quals el servei de Portainer (dintre del seu propi contenidor) té accés per monitorar el comportament general de cadascun d'ells i els seus registres.

De la mateixa manera, abans de la implementació de Portainer, també s'ha fet servir l'eina Docker Desktop. En línies generals, ofereix les mateixes funcionalitats que Portainer, però només funciona a escala local i no en una màquina remota. Tot i això, en les proves en local ha estat d'un gran ajut.

7.3. Elecció de frameworks, sistemes, i serveis

Un cop escollits els diferents frameworks i sistemes per emprar, el diagrama esdevé (es manté l'ús de Portainer, però com a diagrama del projecte com a tal, només aquests elements en formen part):

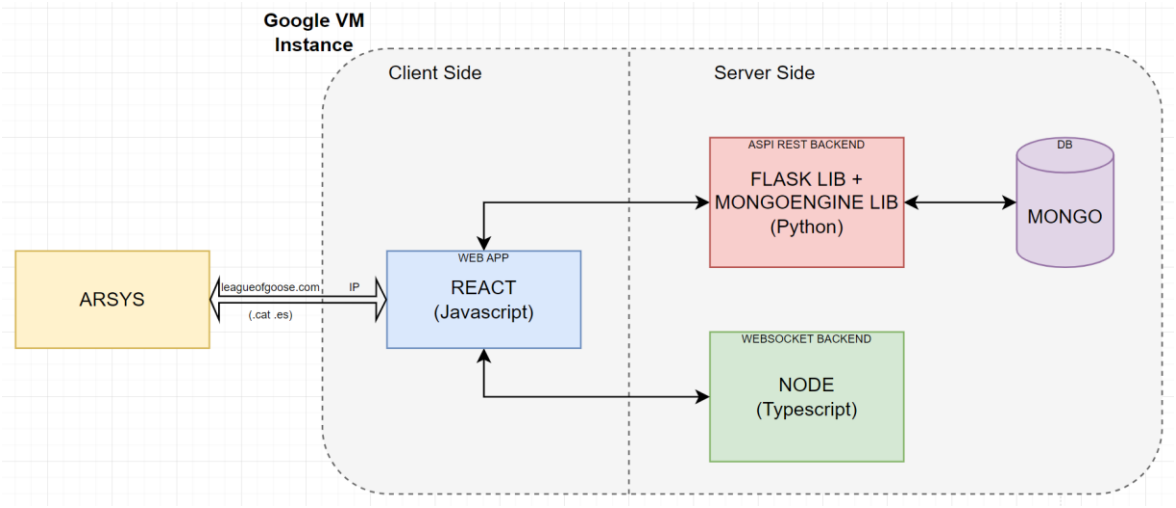

Fig. 7.3.1. Diagrama conceptual dels sistemes, frameworks i entorns aplicats en el projecte. Font: Elaboració pròpia.

WEB APP: framework de React. Fàcil de programar, orientat a web, s'executa en qualsevol dispositiu amb un navegador web (a causa del fet que fa servir Javascript).

API REST BACKEND: Un script de Python que fa servir la llibreria Flask i Mongoengine. Flask opera com a controlador de l'API, on rep les peticions i executa les funcions adients. Aquestes funcions, si escau, fan servir Mongoengine per emmagatzemar, modificar, o esborrar informació de la base de dades, en aquest cas de MongoDB.

DB: Un servei de Mongo. És un sistema de base de dades no relacional, en contrapart de les habitualment emprades bases de dades relacionals.

L'elecció de fer servir la dupla Flask amb Mongo ha estat escollida entre les diferents alternatives (com per exemple Java Springboot [12] amb SQL) per la naturalesa de les dades que s'emmagatzemen a la base de dades. Són taules on merament s'acumulen dades, on no són necessàries relacions entre diferents taules.

De la mateixa manera, el codi necessari per fer servir Flask es pot contenir dintre d'un mateix fitxer no molt gran. És molt simple la seva implementació en aquesta arquitectura, i propicia un estalvi de temps mitjanament significatiu.

WEBSOCKET BACKEND: Està compost per un Node Server, però en comptes de ser codi Javascript com el de React, aquí s'ha fet servir Typescript. Aquest superset de Javascript proporciona tipatge al codi que, de manera nativa, Javascript no té. A nivell funcional el mateix compilador pot conèixer diferents problemes en el codi només mirant el tipatge de les variables, i a nivell humà proporciona una lectura més senzilla i concreta del codi.

VM: El hosting escollit ha estat el de Google Cloud. La seva oferta de tres-cents dòlars en noranta dies i els serveis dels quals disposa han estat el determinat per escollir-lo entre la competència (a la pràctica és completament gratuït, gràcies al fet que s'ha començat a fer servir un parell de mesos abans de la finalització del projecte).

Aquí es poden configurar molts paràmetres, entre els quals estan els de la màquina:

- Sistema operatiu: Ubuntu 22
- Memòria: 4 GB
- Processadors: 2 (Intel)
- Espai en disc: 50 GB

I també els relacionats amb el tallafoc:

- Port 80: port d'entrada predeterminat. Aquí es troba l'aplicació web
- Port 8001: port assignat a l'API REST
- Port 5001: port assignat al servidor de control dels websockets
- Port 9443: port designat per Portainer

7.4. Procés de desenvolupament

Per tal de mantenir un historial de versions i assegurar que el codi desenvolupat es manté fora de la màquina on es crea, s'ha decidit fer servir GitHub. La plataforma proporciona moltes eines a l'abast per tal de gestionar els canvis que es produeixen al codi i facilita la seva descàrrega en qualsevol altra màquina.

De la mateixa manera, també s'ha fet servir l'eina GitHub Desktop per facilitar el control de canvis de versions dels arxius.

7.4.1. Aplicació web

A l'aplicació web s'han emprat diferents llibreries i frameworks:

- Tailwind [13]: és un framework de CSS que enfoca l'elaboració d'estils des d'un altre punt de vista: en comptes de generar una classe amb tots els aspectes que es volen modificar d'un element web, es creen moltes classes on només s'altera un aspecte en concret. D'aquesta manera cal definir en tots els components tots els canvis que facin falta de forma explícita; això encara que no sembli intuïtiu millora molt la seva lectura. També genera automàticament l'arxiu CSS final que es farà servir a l'aplicació web, només amb les classes que s'hagin definit als components (això ho fa fent una lectura dels arxius dintre del codi del projecte i extraient les classes que s'han fet servir).
- Tailwind Material: com s'ha fet servir Tailwind com a framework vehicular dels estils, s'ha fet servir també aquesta llibreria que aporta un conjunt de components per agilitzar el seu desenvolupament i mantenir una consistència entre ells (això poden ser botons, camps de texts, desplegables, etc.).
- Axios [14]: és una llibreria que gestiona i facilita la crida de peticions AJAX. Comunica l'aplicació web amb l'API.
- HammerJS [15]: és una llibreria que facilita la integració d'interacció de l'usuari amb certs components de l'aplicació web, com per exemple galeries d'imatges.
- React Zoom Pan Pinch [16]: aquesta llibreria implementa un control de zoom i arrossegament (tant dispositius mòbils com ordinadors) dintre d'un element HTML. S'ha fet servir per poder-se moure dintre del taulell sense haver de fer zoom a tota la pàgina, permetent que la interfície de l'usuari es mantingui immòbil i preparada per rebre instruccions.
- SocketIO i SocketIO-Client: llibreries necessàries per gestionar en l'aplicació web els missatges que s'envien i es reben a través de les comunicacions websocket.

I es poden destacar aquests aspectes:

- La lògica que es fa servir per generar el taulell i les caselles és completament sensible al dispositiu.
- Per tal de mantenir una distribució ideal de les caselles del taulell sobre el dibuix, primer s'han calculat les proporcions de les caselles sobre el taulell, i després s'han extrapolat aquestes dimensions i posicions sobre la dimensió real en píxels del dispositiu on s'executa l'aplicació.

  Així se soluciona la premissa que es pugui jugar des de qualsevol dispositiu amb un mínim de garanties que es veurà bé. A la pràctica, sempre algun dispositiu amb dimensions no molt estandarditzades pot patir alguna desviació que desajusti l'aparença final de l'aplicació.
- La creació de taulells és completament escalable.

  D'una banda, el taulell es forma de manera semialeatòria (un altre dels requisits). Per fer-ho, el primer jugador que crea una sala dintre del servidor de connexions websocket envia un missatge codificat amb les instruccions. Aquesta llavor té el patró a seguir perquè quan es carreguin els taulells dels diferents dispositius llegeixin la mateixa informació per la generació d'aquests.

  I, d'altra banda, les posicions de les caselles es genera llegint un únic codi amb les instruccions a seguir.

  Amb aquesta combinació, i afegint la lògica específica per cadascuna de les caselles, es poden generar diferents taulells, amb diferents caselles, mesures, normes, etc., de forma molt ràpida i eficient.

7.4.2. REST API Backend

Aquest backend fa servir Flask com a llibreria principal: proporciona la creació d'una aplicació que està preparada per escoltar les peticions que es facin a un port i, depenent de la direcció, executar una funció o altra.

La part interessant és que moltes peticions requereixen identificació, com iniciar sessió o fer transaccions. Per fer-ho, a l'iniciar sessió (un cop registrat) la petició retorna galetes que el navegador guarda i després es fan servir per realitzar les següents peticions sense haver de guardar aquesta informació dintre de l'aplicació web.

A més, proporciona la injecció de variables en fitxers HMTL. Això vol dir que, en aquest cas, es pot demanar a Flask que la petició retorni un HTML, enviant-hi les dades que siguin necessàries per mostrar. Això facilita molt les coses, ja que no cal fer un HTML tradicional on faci encara més peticions per mostrar alguns resultats. És especialment útil per, com per exemple, a nivell d'administració mostrar informació dels usuaris.

Aquí es pot veure un exemple d'algunes funcions que funcionen d'aquesta manera:

```
# ADMIN RELATED ENDPOINTS


@app.route("/admin")
def index():
    return render_template("index.html")


@app.route("/admin/players", methods=["POST"])
@admin_required
def load_players_page():
    return render_template("players.html", players=Players.objects())
```

Fig. 7.4.2.1. Fragment de codi, exemplificant funcions que carreguen codi HTML amb variables. Font: Elaboració pròpia.

Concretament, la de “admin/players” carrega un HTML, on s’envien tots els registres de la col·lecció de “Players” de la base de dades com a variable “players”. Dintre del codi HTML, si se segueixen les indicacions de la documentació [18] es poden, per exemple, iterar aquests objectes i mostrar les seves propietats.

```
<h2>PLAYERS</h2>
<table>
    <tr>
        <th>EMAILS</th>
        <th>USERNAMES</th>
        <th>PASSWORDS</th>
        <th>INVENTORIES</th>
    </tr>
    {% for player in players %}
    <tr>
        <td>{{ player.email }}</td>
        <td>{{ player.username }}</td>
        <td>{{ player.password }}</td>
        <td>{{ player.inventory }}</td>
    </tr>
    {% endfor %}
</table>
```

Fig. 7.4.2.2. Fragment de codi, exemplificant accés a variables de jugadors dintre del codi HTML. Font: Elaboració pròpia.

Finalment, a l’anar a l’URL esmentada, apareixen les dades directament.

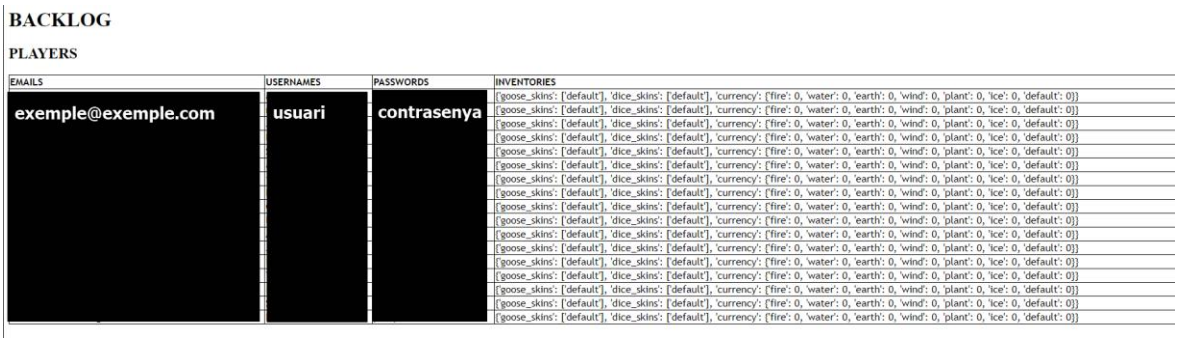

Fig. 7.4.2.3. Captura de pantalla del servei de backend, on es veu el resultat de les dades que es guarden a la base de dades. Font: Elaboració pròpia.

D'aquesta manera es podria elaborar un taulell per administradors molt complet o interactiu on es poguessin fer les operacions que es desitgin.

De la mateixa manera, fa servir Mongoengine, també de Flask. Està preparat per gestionar qualsevol canvi o consulta que es vulgui fer a una base de dades de tipus MongoDB en execució. En aquest s'ha fet servir principalment per emmagatzemar usuaris, i editar el seu inventari.

7.4.3. Servidor de websockets

Aquest servidor creat amb Node és un sistema monolític que principalment empra programació orientada a objectes i interfícies. Degut a aquesta naturalesa, s'ha fet servir Typescript, que aporta tipatge al codi.

Encara que això faciliti el desenvolupament del codi, ha presentat un dels reptes de la realització d'aquest projecte degut al propòsit que ha de servir.

La comunicació Websocket és molt potent i pràcticament instantània, en condicions de laboratori. En la realitat, les desconnexions i pèrdues de missatges representen un percentatge que no es pot ignorar. Inclús en el cas més senzill, com pot ser canviar de pestanya al navegador o canviar de pàgina dintre de l'aplicació web, es pot interrompre la connexió. Això complica molt el codi, ja que es vol una consistència de les dades que reben tots els jugadors.

La solució implementada ha estat la creació dintre del Node d'un procés de manteniment i actualitzacions d'identificadors únics associats als correus electrònics dels jugadors. És a dir, cada cop que es carrega una pàgina de l'aplicació web i es carrega correctament la comunicació websocket, aquesta envia un missatge tipus "login" amb la informació del jugador. Aquest servidor processa la petició, i comprova si aquest jugador existeix en alguna de les sales creades. En el cas que no, es crea una i es guarda dintre la informació de l'usuari i l'identificador de la connexió.

A partir d'aquest punt, cada cop que es necessiti enviar un missatge a aquest jugador (o a una sala que el contingui) s'enviarà directament a aquest identificador.

Si es perd la connexió (no es pot assumir que es mantindrà), aquests missatges no arribaran al jugador desconnectat. El que sí que es pot suposar és que en algun moment es connectarà de nou (probablement), i quan es carrega la pàgina tornarà a enviar un missatge tipus "login". Ara, aquest servidor trobarà que el jugador existeix i només canviarà l'antic identificador amb el nou. Així es manté una persistència de dades (de jugadors en sales, posicions del joc, etc.) sobreposant-se a les possibles múltiples desconnexions que hi puguin haver.

Això a més implica que si un jugador vol accedir des de diferents dispositius només funcionarà l'últim que hagi obert la connexió, la més recent.

Un altre punt que també s'ha tingut en compte ha estat la neteja de les sales dels jugadors. Si no hi ha activitat dintre d'una sala en 10 minuts, es borra automàticament. S'entén per activitat afegir jugadors a una sala o realitzar alguna acció dins del joc.

Paral·lelament, aquestes són les llibreries que fa servir:

- express i http: Són les encarregades de mantenir l'aplicació executant-se esperant peticions. Habitualment es fan servir com una API REST tradicional, però en aquest cas només es fa servir per establir connexions websocket.
- Socket.io: igual que a l'aplicació web, aquesta llibreria facilita el punt d'enllaç de les connexions websocket emeses per l'aplicació web.
- Node-cron: permet realitzar funcions programades. Es fa servir per netejar les sales que estan buides o no han registrat activitat en una estona; en aquest cas s'executa cada minut.

7.4.4. Servei de bases de dades MongoDB

Aquest servei és autònom com a tal i no requereix modificacions a nivell intern. Un cop creat el Docker amb aquest codi i realitzada la seva execució, només s'altera el seu contingut a través del servidor de backend amb la llibreria Mongoengine de Flask.

7.4.5. Alternatives o enfocaments explorats, però descartats

- La utilització d'un repositori d'imatges per accelerar la càrrega d'aquestes dintre de l'aplicació web. En aquest moment les imatges existeixen internament de la web, i en alguns dispositius no gaire potents es carregava de forma lenta. Al posar les imatges en una direcció (en aquest cas s'ha provat amb un altre repositori de GitHub), no s'apreciava cap millora, així que es van descartar els canvis.
- Unificar l'API REST amb el servidor de websockets. En principi, Flask també proveeix de funcions preparades per la comunicació websocket. Tot i això, aquesta unió no s'ha fet per diferents motius:
    - Claredat: no era necessari a nivell funcional unificar dos serveis que compleixen propòsits diferents.
    - Nativitat: React i Node es basen en Javascript, però Flask fa una conversió a Python d'aquest codi; potencialment podria complicar les connexions. Per això s'ha fet servir la llibreria de forma nativa en ambdós punts de la comunicació.
    - Esquema de l'arquitectura: encara que sí que s'hagués pogut unificar al Node aquests dos serveis, no s'ha trobat una forma més senzilla que la de Flask d'operar amb una base de dades no relacional a Node. Altres tipus de bases de dades o implementacions sí que són possibles, però s'ha decidit fer servir Flask entre les alternatives.

7.4.6. Prototips

Aquest són alguns dels prototips inicials a seguir que s'han pactat amb la dissenyadora després de diverses reunions:

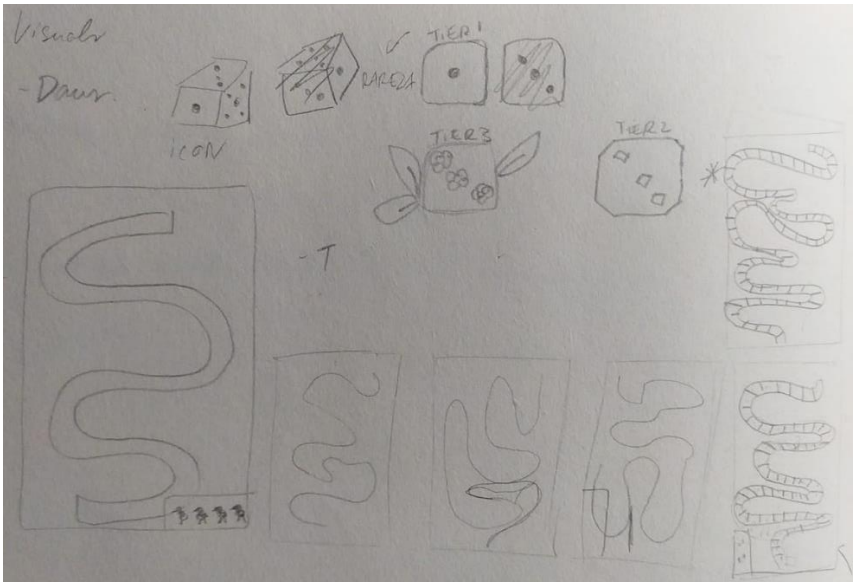

Fig. 7.4.6.1. Il·lustració conceptual, pluja d'idees del concepte del joc. Font: Proporcionada per la il·lustradora Roser Butrón Isern.

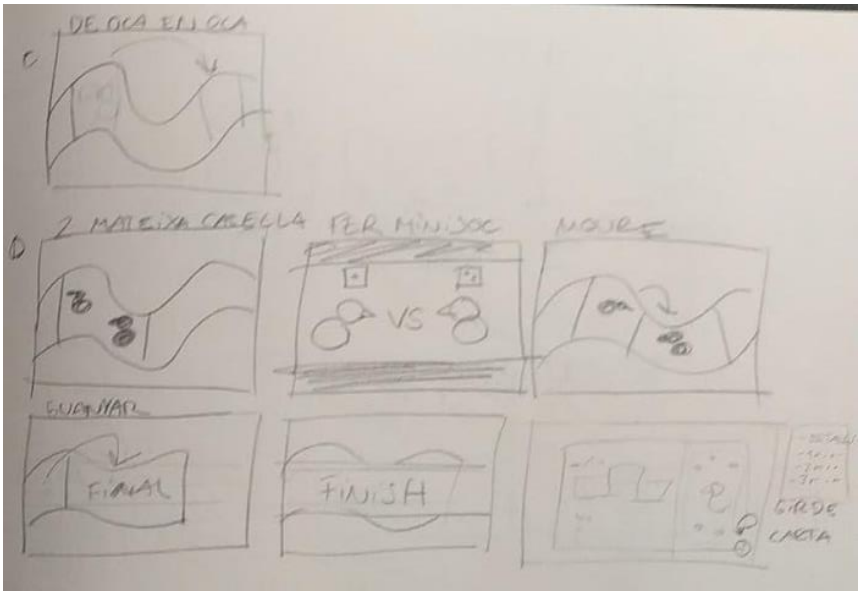

Fig. 7.4.6.2. Il·lustració conceptual, seqüències de tirada de dau, mini joc dels jugadors, i final de partida. Font: Proporcionada per la il·lustradora Roser Butrón Isern.

Fig. 7.4.6.3. Il·lustració conceptual, disseny de la pantalla principal, tenda, detall d'objecte de tenda, i selecció d'aspectes. Font: Proporcionada per la il·lustradora Roser Butrón Isern.

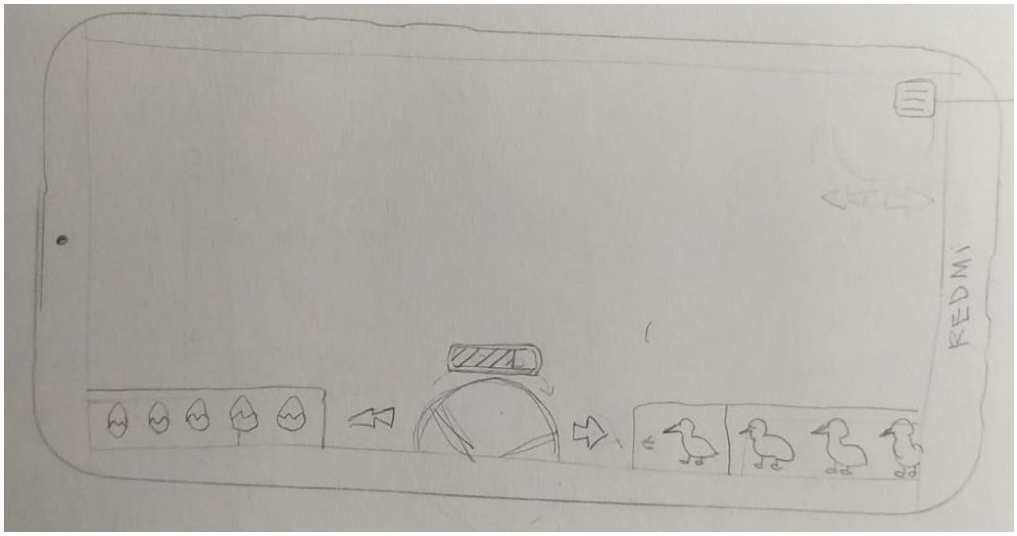

Fig. 7.4.6.4. Il·lustració conceptual, disseny d'interfície del jugador. Font: Proporcionada per la il·lustradora Roser Butrón Isern.

Aquestes il·lustracions han estat la base on realitzar la major part del disseny de l'aplicació web, com per exemple:

- Taulell lineal, generat de forma aleatòria
- 3 nivells de raresa de daus, col·leccionables i utilitzables dintre de joc
- Pantalla principal: a aquest disseny més tard s'hi van afegir els botons de registre i d'inici de sessió.
- Tenda: tot i al final no haver estat desenvolupada, el disseny inicial la concebia com diverses col·leccions d'aspectes (que al clicar a sobre obren una pantalla amb la informació i el preu) organitzades per temàtiques (com per exemple aspectes de temporada, que estan disponibles per un temps limitat, o permanents).
- Pantalla de selecció d'aspectes
- Interfície d'usuari: finalment, al no implementar una lògica de torns, la part baixa a la dreta de la pantalla s'ha substituït amb la descripció del dau a emprar.

7.5. Procés de desplegament a producció

Un cop finalitzat tot el codi de desenvolupament, s'ha realitzat la seva migració d'entorns de producció.

Habitualment, els mateixos entorns de desenvolupament proveeixen d'eines per transformar tot el projecte en codis molt simplificats pel seu ús en entorns reals. De tota manera, aquests entorns també s'han desenvolupat per existir dintre d'una arquitectura de Docker, com s'ha fet durant tota l'etapa de desenvolupament.

7.5.1. Aplicació web

El primer pas ha estat crear un nou codi de tipus Dockerfile (en aquest cas, dintre del repositori s'anomena "Dockerfile-pro") que prepari un entorn on, copiant el codi de desenvolupament, executi les comandes per generar el codi de producció.

Al mateix codi, també es prepara un entorn que faci servir Nginx com a servidor. Aquesta implementació (o qualsevol altra que sigui capaç d'executar un servei de servidor) és necessària perquè ara el codi no s'està executant en cap port de la màquina per si mateix. Llavors, es copia el codi de producció dintre del directori apropiat de Nginx, i és aquest que es programa per tal que sí s'executi en el port que desitgem (en aquest cas, el 80).

Durant aquest procés s'ha hagut de modificar l'arxiu generat automàticament per Nginx de configuració del servidor (anomenat "nginx.conf" al repositori) per tal que al canviar de pàgina dintre de la mateixa aplicació continuï executant el codi de React.

7.5.2. Backend REST API i servidor de websockets

Degut a una mancança de temps no s'han realitzat entorns de producció optimitzats per aquests dos serveis.

Això ha estat degut a la priorització d'optimitzar l'aplicació web per sobre de la resta, i per tant consumir més hores.

Tot i això, com que la quantitat de peticions que es fan aquests dos serveis no és gaire elevada (durant tot el procés de desenvolupament han participat no més de quatre persones a la vegada), aquests serveis no necessiten una versió de producció per funcionar amb normalitat.

De tota manera, aquesta part consisteix una part fonamental per la comercialització del projecte final.

7.5. Valoració d'objectius aconseguits

Al moment de l'entrega del projecte, es donen per aconseguits els següents objectius:

- Crear un MVP d'una versió actualitzada del "Joc de l'Oca", accessible des d'un navegador web.
- Crear un MVP d'una connexió client/servidor, on el client és el jugador dintre del taulell virtual i el servidor és l'API/websocket que gestiona el joc.
- Crear una base de dades on emmagatzemar:
    - Jugadors registrats (usuaris)
- Crear una connexió MVP entre els clients, el servidor i la base de dades.
- Permetre registrar-se i iniciar sessió.
- Permetre crear partides amb altres jugadors.
- Permetre escollir entre aspectes personalitzats per fer servir dins del joc (només a nivell gràfic).

I concretament del joc:

- Creació dinàmica de taulells
- Possibilitat d'escollir entre diferents daus trucats per fer servir.
- Programació dels efectes que tenen les caselles quan els jugadors cauen sobre elles.
- Animació de tirada de daus.
- Animació de mostra de descripció d'efecte de casella per tots els jugadors.
- Aconseguir "minigeese" durant el transcurs del joc. Al caure en una casella d'oca a oca s'obté la seva forma bàsica, i quan es cau en una casella elemental es transforma en la seva versió elemental.
- Desenvolupament de finestra flotant amb la descripció de casella quan es pitja sobre ella.
- Mostra de pantalla resum de la partida quan es finalitza.

I no es donen per aconseguits aquests objectius:

- Registre de partides.
- Classificació general.
- Crear una tenda on comprar aspectes personalitzats.
- Permetre jugar contra la màquina (no una IA, sinó un algoritme de decisió).
- Creació de diferents taulells, que s'escullen de forma aleatòria al iniciar la partida.
- Implementació de fonts d'àudio per millorar l'experiència d'usuari.

I respecte al joc:

- Desenvolupament de minijocs entre els jugadors quan es cau en una casella ocupada per un d'ells.
- Implementació de lògica de torns.

7.6. Repositoris emprats

Els repositoris que s'han fet servir durant tot el procés són:

- Aplicació web:

Inicialment es va fer servir aquest repositori:

https://github.com/arodriguezle/league_of_goose

Però en el moment de generar una versió de producció van aparèixer errors de compilació i es va haver de recrear el projecte, esdevenint:

https://github.com/arodriguezle/league_of_goose_webapp

- API REST Backend:

https://github.com/arodriguezle/league_of_goose_backend

- Servidor websocket

https://github.com/arodriguezle/league_of_goose_node_server

8. Conclusions

L'objectiu principal del projecte ha estat la realització d'una versió nova del clàssic Joc de l'Oca online multijugador simultani. I es considera aconseguit.

Tot i això, cal perfilar alguns aspectes d'aquesta afirmació: al moment de l'entrega d'aquest projecte es considera al producte (és a dir, l'aplicació web i les seves funcionalitats) un **prototip**. Els motius són:

- Un dels objectius principals del projecte era la implantació de la tenda, que no ha estat finalment desenvolupada.
- No presenta cap control de l'ordre dels jugadors. Cada jugador pot llençar el dau en qualsevol moment, propiciant possibles errors en quant al càlcul de les posicions dels jugadors. A més, la casella que té l'efecte de fer que un jugador saltar un torn el perd completament.
- No implementa fonts d'àudio. Tot i valorar plataformes on poder descarregar aquests recursos de forma lliure, degut a falta d'hores per poder dedicar no han afegits en aquesta versió.
- Errors generals: tot i que la resta sí que ha estat desenvolupat, no s'ha dedicat el temps suficient a comprovar els casos compromesos del codi. Això propicia errades com:
    - Si un jugador cau en l'última casella amb una Oca, torna al principi.
    - Si un jugador cau en una casella que mou diversos jugadors, visualment només es mou un (internament sí que fa bé el càlcul, i quan es torna a moure ho fa des de la posició correcta, però és confós pels jugadors).
    - El selector de daus a la pantalla de selecció d'aspectes no es comporta correctament, així que carrega tres per defecte.
    - La primera càrrega del taulell, depenent de l'ordre d'arribada dels jugadors al taulell, pot aparèixer que tenen dos taulells diferents. Tot i això, els moviments seran els mateixos. Si es recarrega la pàgina aquest error es soluciona.
    - No es pot veure la descripció d'una casella si hi ha un jugador a sobre al fer clic.

  - No apareix la pantalla de finalització de la partida: tot i estar desenvolupada, si un jugador matemàticament cau en l'última casella (inclòs abans de veure el moviment) el joc finalitza, talla la connexió, i per tant no apareix el resum.
  - En finalitzar la partida no es guarden els "minigeese" de forma correcta a l'inventari del jugador.

Respecte a la valoració de l'arquitectura realitzada i les tecnologies i sistemes emprats, el resultat és bastant satisfactori.

Han servit el seu propòsit, tant per l'etapa desenvolupament com la de producció, d'una forma propícia i òptima, facilitant la inversió d'hores i propiciant serveis que funcionen eficientment.

Tot i això, hi ha dos aspectes del desenvolupament que en un suposat cas haurien facilitat encara més aquest procés:

- Implementació de Redux a l'aplicació web: Redux és una llibreria que permet que tots els components de l'aplicació puguin accedir a tots els estats dels components escollits i mantenir un control exhaustiu. Com que el joc es basa en la comparació d'estats entre el client i el servidor de websockets, mantenir un control d'aquests i que els components que no estan dintre del taulell (com el d'interfície d'usuari i el de contenidor d'animacions) poguessin accedir de forma més senzilla a aquesta informació.
- Implementació de la lògica al servidor de websocket: conceptualment, sembla que té més sentit que l'aplicació sigui qui tingui la lògica de com controlar el joc i els seus components. La realitat és que com altres clients han de tenir aquesta informació i carregar-la de la mateixa manera, la implementació òptima ha de ser la que contempla el servidor com a procés general i els clients com a carregadors d'informació.

D'altra banda, pel que fa al procés de la realització del projecte, s'han subestimat les hores i complexitat de desenvolupament.

Això ha provocat una reducció d'hores significatives a invertir en altres objectius, requisits, i tasques. Tot i això, es considera que el resultat final de l'aplicació de l'arquitectura i el joc final és satisfactori, amb els defectes esmentats al moment d'entrega del projecte.

Per donar un context, es va planificar la tasca de proves generals amb jugadors reals durant la primera setmana de maig. En la realitat, la primera prova funcional que tots els serveis i connexions estaven en ordre es va obtenir al disset de maig de 2023; amb la primera prova de connectivitat entre quatre jugadors:

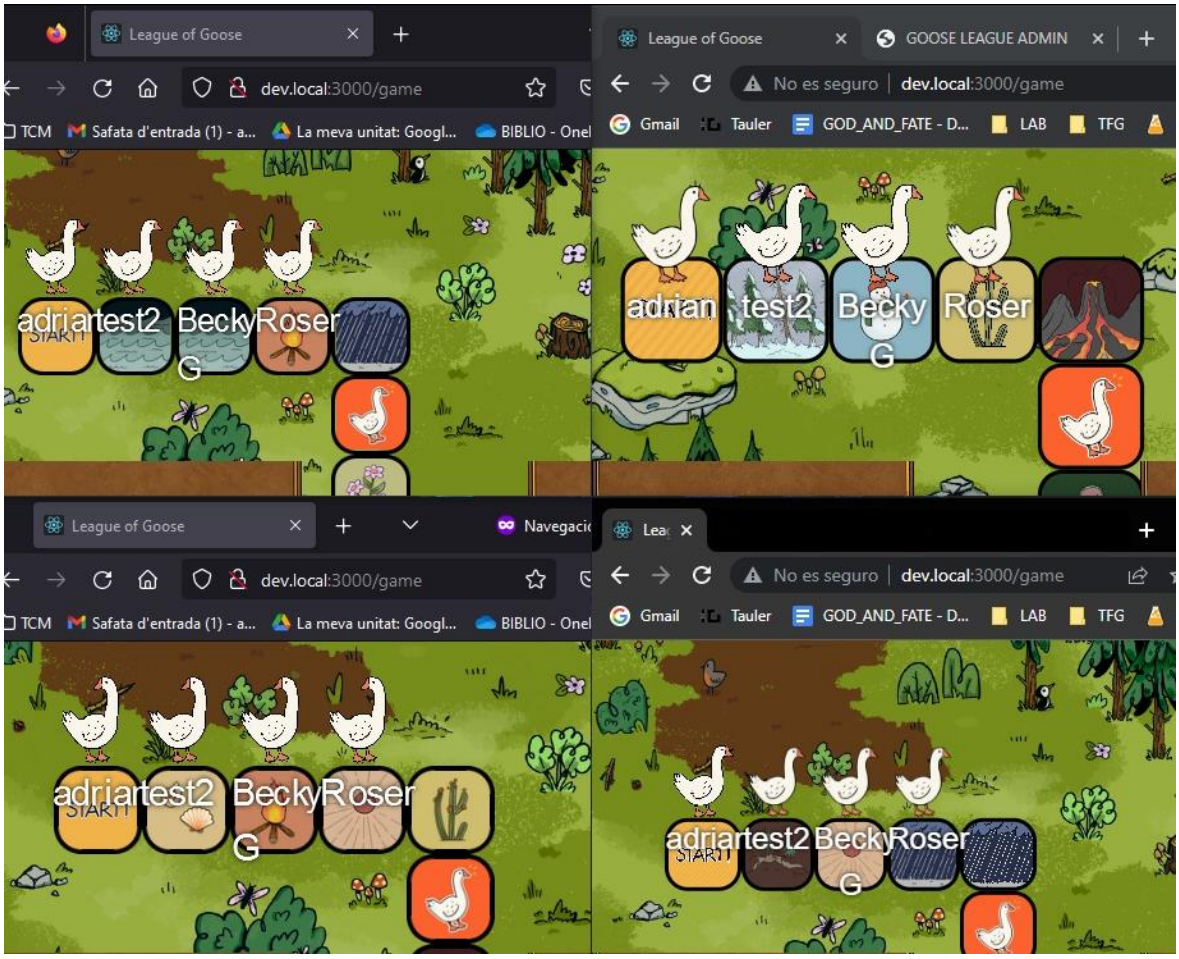

Fig. 8.1. Captura de pantalla de la primera prova de connectivitat funcional. Font: Elaboració pròpia.

En general hi ha hagut un arrossegament de moltes hores invertides en la recerca de la implementació de les tecnologies que s'han fet servir ja que no es tenia cap coneixement previ de moltes d'aquestes. Tot i això, l'aprenentatge sobre aquestes aporta una bona base per futurs projectes amb objectius o funcionalitats semblants a les posades a prova en aquest projecte.

En el futur pròxim, la intenció és fer el primer punt de les possibles millores: la millora general del joc. Es considera la millora més important de totes, perquè encara que no es realitzin cap de les ampliacions proposades el producte es podria considerar finalitzat i completament operatiu.

9. Possibles ampliacions

9.1 Millores generals del joc

Si se solucionen els errors que presenta el joc, esmentats a les conclusions, es pot considerar un producte complet i amb potencial.

A més, el trasllat de lògica de l'aplicació web al servidor de websockets facilitaria encara més desenvolupament a futur.

9.2 Securització

Tant els missatges entre els diferents serveis com les contrasenyes no apliquen cap mesura de seguretat.

Tanmateix, l'intercanvi d'informació entre l'aplicació web amb el backend i el servidor de websocket, tampoc aplica cap tractament de seguretat.

Finalment, l'aplicació web no té implementat un certificat SSL per tal d'encriptar la informació. Per fer-ho s'hauria de comprar un certificat SSL i afegir-lo al servei Nginx que allotja l'aplicació web.

9.3 Manteniment d'informació entre partides

En l'entrar en una nova sala després de cada partida, mantenir la selecció de daus i aspectes seleccionats prèviament.

Tot i semblar un canvi relativament senzill, implica moltes millores envers els jugadors i el seu temps en pantalla (que potencialment es tradueix en ingressos). Entre els principals jocs multijugador del moment (*Marvel Snap, Leauge of Legends, Valorant, Teamfight Tactics*, etc.) es pot trobar que en el moment que finalitza una partida reben una recompensa (en forma de punts, moneda interna, aspectes, etc.) i després apareix la pantalla prèvia a començar al joc amb tota la configuració inicial intacta i llesta per tornar a començar.

9.4 Sistema de repeticions

Si tots els estats dels jocs es guardessin en una base de dades, es podria elaborar un codi que llegís aquests estats i pogués veure repeticions de les partides.

9.5 Accessibilitat

Qualsevol mena de millora d'accessibilitat (audiodescripció, interfície encara més gran, dictat per veu, etc.) pot ser una gran millora del projecte.

9.6 Desplegament en Kubernetes

Com que els serveis estan desenvolupats en Docker, la seva migració a un sistema de Kubernetes on es poden crear i eliminar instàncies de serveis seria relativament senzilla. D'aquesta manera es podria tenir un control de cada servei del projecte controlat sobre la base de la demanda.

Tot i aquesta potencial millora, només seria necessari en el cas d'una massificació d'usuaris i trànsit d'informació.

9.7 Passarel·la de pagament

Implementar una passarel·la de pagament on, per exemple, poder comprar moneda del joc amb diners reals. Aquesta seria la principal forma de sustentació financera del projecte, tenint en compte els criteris esmentats l'anàlisi de viabilitat.

10. Bibliografia

[1] Joc de l'oca de "juegosjuegos": [en línia] [consulta: 8 de febrer de 2023].

https://www.juegosjuegos.com/jugar-juego/goose-game.html

[2] Joc de l'oca d'App Store: [en línia] [consulta: 10 de febrer de 2023].

https://apps.apple.com/us/app/goose-game-multiplayer/id1446722303

[3] Estudi "2022 Essential Facts About the Video Game Industry": [en línia] [consulta: 15 d'abril de 2023].

https://www.theesa.com/resource/2022-essential-facts-about-the-video-game-industry/

[4] Web de la il·lustradora Roser Butrón Isern: [en línia] [consulta: 15 d'abril de 2023].

https://roserbutron.github.io/

[5] Dossier de Sophie McPike: [en línia] [consulta: 16 de gener de 2023]

https://www.sophiemcpike.com

[6] Dossier d'Alice Romano: [en línia] [consulta: 16 de gener de 2023]

https://alicenoah.wixsite.com/alicenoah

[7] Dossier d'Emily Schofield: [en línia] [consulta: 17 de gener de 2023]

https://www.emilyschofield.net

[8] Dossier de Jessica Elena: [en línia] [consulta: 18 de gener de 2023]

https://byjessicaelena.com

[9] Dossier de Jessica Smith: [en línia] [consulta: 18 de gener de 2023]

https://www.jessicasmithillustration.co.uk

[10] Web Trello: [en línia] [consulta: 29 de maig de 2023]

https://trello.com

[11] Web Portainer: [en línia] [consulta: 20 de març de 2023]

https://www.portainer.io

[12] Web Spring.io (Java Spring Boot): [en línia] [consulta: 18 de gener de 2023]

https://spring.io/projects/spring-boot

[13] Documentació Tailwind: [en línia] [consulta: 04 d'abril de 2023]

https://tailwindcss.com

[14] Documentació Axios: [en línia] [consulta: 03 d'abril de 2023]

https://axios-http.com/docs/intro

[15] Web de HammerJs: [en línia] [consulta: 03 d'abril de 2023]

https://hammerjs.github.io/

[16] Repositori llibreria react-zoom-pan-pinch: [en línia] [consulta: 20 de gener de 2023]

https://github.com/prc5/react-zoom-pan-pinch#readme