laSalle

UNIVERSITAT RAMON LLULL

Escola Tècnica Superior d'Enginyeria La Salle

Treball Final de Grau

Grau en Enginyeria Informàtica

Age of Ra: Desenvolupament d'un videojoc d'estratègia en temps real escalable

Alumne

Pol Piñol Castuera

Professor Ponent

Pol Muñoz Pastor

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Pol Piñol Castuera

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Age of Ra: Desenvolupament d'un videojoc d'estratègia en temps real escalable

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Abstract

The main objective of this project is to design and implement an MMORTS video game with its own theme and different from the conventional ones on the market. It also seeks to address performance issues, particularly system loading and response time, that affect these types of games.

This project analyzes massively multiplayer online real-time strategy video games (MMORTS), which allow players to interact in real time with a virtual world. This genre of games is based on the concept of a player who assumes the role of a village leader, managing resources, diplomacy and training armies.

An analysis of MMORTS is performed in web browsers during their heyday, when modern tools such as *game engines* or *frameworks* were not used. It also explores the constraints and difficulties in developing these games using the technology that have emerged since then.

Keywords: web application, performance, scalability, video game, MMORTS

Resumen

El objetivo principal de este proyecto es diseñar e implementar un videojuego MMORTS con una temática propia y distinta a las convencionales del mercado. Además, se busca abordar los problemas de rendimiento, especialmente el tiempo de carga y respuesta del sistema, que afectan a este tipo de juegos.

Este trabajo analiza los videojuegos de estrategia en tiempo real multijugador masivos online (MMORTS), que permiten a los jugadores interactuar a tiempo real con un mundo virtual. Este género de juegos se basa en el concepto de un jugador que asume el papel de un líder de poblado, gestionando recursos, diplomacia y formación de ejércitos.

Se realiza un análisis de los MMORTS en navegadores web durante su época de mayor popularidad, cuando no se utilizaban herramientas modernas tales como *game engines* o *frameworks*. También se exploran las restricciones y dificultades en el desarrollo de estos juegos utilizando las tecnologías que han surgido desde entonces.

Palabras clave: aplicación web, rendimiento, escalabilidad, videojuego, MMORTS

Resum

L'objectiu principal d'aquest projecte és dissenyar i implementar un videojoc MMORTS amb una temàtica pròpia i diferent de les convencionals del mercat. A més, es busca abordar els problemes de rendiment, especialment el temps de càrrega i de resposta del sistema, que afecten aquest tipus de jocs.

Aquest treball analitza els videojocs d'estratègia en temps real multijugador massius en línia (MMORTS), que permeten als jugadors interactuar a temps real amb un món virtual. Aquest gènere de jocs es basa en el concepte d'un jugador que assumeix el paper d'un líder de poblat, gestionant recursos, la diplomàcia i formació d'exèrcits.

Es realitza una anàlisi dels MMORTS en navegadors web durant la seva època de major popularitat, quan no es feien servir eines modernes com ara *game engines* o *frameworks*. També s'exploren les restriccions i dificultats en el desenvolupament d'aquests jocs utilitzant la tecnologia que han sorgit des de llavors.

Paraules clau: aplicació web, rendiment, escalabilitat, videojoc, MMORTS

Agraïments

Agraïments al Pol Muñoz, tutor del treball de fi de grau des de principis d'estiu de l'any 2022, per les seves correccions, per l'orientació i la confiança d'acompanyar-me durant aquest últim projecte d'informàtica.

A la meva parella, Mireia, vull expressar-li el meu profund agraïment per tota la comprensió del món i el suport que m'ha brindat al llarg d'aquests últims anys.

Finalment, també vull agrair a la meva família, especialment als meus pares, Olga i Joan, i al meu germà, Jan, per haver estat sempre al meu costat, donant-me suport i, sobretot, confiança en cada pas que he pres durant aquest camí.

Índex

1 Introducció **1**
- 1.1 Context . . . 1
- 1.2 Objectius del treball . . . 2
- 1.3 Metodologia . . . 2
- 1.4 Estructura del document . . . 3

2 Estudi de mercat **5**
- 2.1 Travian: Legends . . . 5
- 2.2 Guerras tribales . . . 7
- 2.3 OGame . . . 9
- 2.4 Ikariam . . . 11
- 2.5 Conclusions de l'estudi de mercat . . . 13

3 Especificacions de requisits de *software* **15**
- 3.1 Introducció . . . 15
  - 3.1.1 Propòsit . . . 15
  - 3.1.2 Abast . . . 16
  - 3.1.3 Definicions, sigles i acrònims . . . 16
  - 3.1.4 Referències . . . 16
  - 3.1.5 Resum . . . 16
- 3.2 Descripció general . . . 16
  - 3.2.1 Perspectiva del producte . . . 16
    - 3.2.1.1 Interfícies d'usuari . . . 17
  - 3.2.2 Funcionalitats del producte . . . 17
  - 3.2.3 Característiques de l'usuari . . . 18
  - 3.2.4 Suposicions i dependències . . . 18
  - 3.2.5 Requisits a futur . . . 18
- 3.3 Requisits específics . . . 18
  - 3.3.1 Requisits funcionals del producte . . . 19
    - 3.3.1.1 Gestió usuaris . . . 19
    - 3.3.1.2 Gestió terra . . . 20
    - 3.3.1.3 Estadístiques . . . 22
    - 3.3.1.4 Gestió joc . . . 25
    - 3.3.1.5 Gestió admin . . . 31
  - 3.3.2 Requisits no funcionals del producte . . . 33
    - 3.3.2.1 Rendiment . . . 34
    - 3.3.2.2 Fiabilitat . . . 34
    - 3.3.2.3 Disponibilitat . . . 34
    - 3.3.2.4 Seguretat . . . 34
    - 3.3.2.5 Manteniment . . . 34

4 Model de dades **35**
- 4.1 Diagrama de classes estàtic . . . 35
- 4.2 Diagrames de seqüència . . . 40

5 Disseny tecnològic **42**
5.1 Arquitectura del sistema . . . 42
5.2 Comunicació . . . 43
5.3 Decisions tecnològiques . . . 44
5.3.1 *Frontend* . . . 45
5.3.2 *Backend* . . . 46
5.3.3 Persistència de dades . . . 48
5.4 Diagrama de desplegament . . . 49
5.5 Interfície d'usuari . . . 50
5.6 Diagrama de classes . . . 58
5.7 Diagrama de paquets . . . 59

6 Implementació **60**
6.1 Implementació tecnològica . . . 60
6.1.1 Arquitectura de l'API REST . . . 60
6.1.2 Càlcul en temps real dels esdeveniments . . . 61
6.1.3 Persistència de les regles del videojoc . . . 66
6.1.4 Components de Vue.js . . . 67
6.1.5 SSOT i *stores* de Pinia . . . 71
6.1.6 Versions del joc . . . 72
6.2 Desenvolupament del disseny gràfic . . . 73
6.2.1 Temàtica del joc . . . 73
6.2.2 Disseny del logotip i icona . . . 73
6.2.3 Disseny dels edificis . . . 75
6.2.4 Disseny dels recursos . . . 78
6.2.5 Disseny de les tropes . . . 79
6.2.6 Disseny final del joc . . . 80

7 Experimentació i resultats **93**
7.1 Locust com a eina de testeig . . . 93
7.2 Proves de funcionament . . . 94
7.3 Proves de rendiment . . . 95
7.4 Afegir nous elements a les regles del joc . . . 101

8 Cost del projecte **103**
8.1 Cost en hores . . . 103

9 Conclusions i futur **105**
9.1 Conclusions tècniques . . . 105
9.2 Conclusions personals . . . 106
9.3 Línies de futur . . . 107

10 Bibliografia **108**

11 Annex **110**

Figures

2.1 Mapa on es situa la terra del jugador 6
2.2 Interior i exterior d'una terra 6
2.3 Mapa on es situa la terra d'un jugador 8
2.4 Interior de la terra d'un jugador 8
2.5 Visió general del planta principal 10
2.6 Mapa de totes les illes veïnes 11
2.7 Interior d'una illa del jugador 12

3.1 Diagrama conceptual de l'arquitectura del sistema *Age of Ra* . . 17
3.2 Diagrama general de cas d'ús de la interfície AOR (*Age of Ra*) . 19
3.3 Diagrama de cas d'ús del mòdul de gestió d'usuaris 20
3.4 Diagrama d'activitats del mòdul de gestió d'usuaris 20
3.5 Diagrama d'activitats del cas d'ús AOR_UC2.2._Crear nova terra 21
3.6 Diagrama de cas d'ús del mòdul de gestió de terres 21
3.7 Diagrama d'activitats del mòdul de gestió de terres 22
3.8 Diagrama d'activitats del cas d'ús AOR_UC3.1._Mostrar *timeline* 23
3.9 Diagrama d'activitats del cas d'ús AOR_UC3.2_Mostrar Rànquing 24
3.10 Diagrama de cas d'ús del mòdul d'estadístiques 24
3.11 Diagrama d'activitats del mòdul d'estadístiques 25
3.12 Diagrama d'activitats del cas d'ús AOR_UC4.1_Fer aventura . . . 26
3.13 Diagrama d'activitats del cas d'ús AOR_UC4.2_Mostrar exterior terra 27
3.14 Diagrama d'activitats del cas d'ús AOR_UC4.12_Refresh terra . . 29
3.15 Diagrama de cas d'ús del mòdul de gestió del joc 30
3.16 Diagrama d'activitats del mòdul de gestió del joc 30
3.17 Diagrama d'activitats del cas d'us AOR_UC5.1_Mostrar recurs . 31
3.18 Diagrama d'activitats del cas d'us AOR_UC5.5_Afegir recurs . . 32
3.19 Diagrama de cas d'ús del mòdul de gestió administrador 33
3.20 Diagrama d'activitats del mòdul de gestió administrador 33

4.1 Diagrama de classes estàtic del sistema AOR 40

5.1 Diagrama no formal de l'arquitectura del sistema 43
5.2 *Frameworks* més utilitzats de JavaScript pel *frontend* segons *State of JavaScript* 45
5.3 *Frameworks* amb més interès per part de la comunitat de JavaScript pel *frontend* segons *State of JavaScript* 46
5.4 Comunicació utilitzant una API REST 47
5.5 Comunicació utilitzant WebSockets 47
5.6 *Frameworks* més utilitzats de JavaScript pel *backend* segons *State of JavaScript* 48
5.7 Diagrama de desplegament del sistema AOR 49
5.8 *Wireframe* de la pantalla d'inici 50
5.9 *Wireframe* de la pantalla de registre d'usuari 51
5.10 *Wireframe* de la pantalla d'inici de sessió 51
5.11 *Wireframe* de la pantalla de joc exterior 52

5.12 *Wireframe* de la pantalla de joc interior . . . 52
5.13 *Wireframe* de la pantalla de *timeline* . . . 53
5.14 *Wireframe* de la pantalla del rànquing . . . 53
5.15 *Wireframe* de la pantalla per crear una terra . . . 54
5.16 *Wireframe* de la pantalla per tancar la sessió . . . 54
5.17 *Wireframe* de la pantalla principal dels administradors . . . 55
5.18 *Wireframe* de la pantalla per editar aventures dels administradors 55
5.19 *Wireframe* de la pantalla per editar edificis dels administradors . 56
5.20 *Wireframe* de la pantalla per editar recursos dels administradors 56
5.21 *Wireframe* de la pantalla per editar tropes dels administradors . 57
5.22 Diagrama de classes del sistema AOR . . . 58
5.23 Diagrama de paquets del sistema AOR . . . 59

6.1 Arquitectura en capes Controller-Service-Repository . . . 61
6.2 Simulació del cas base de producció de recursos . . . 63
6.3 Simulació del cas del càlcul de recursos finals amb una millora de recurs . . . 64
6.4 Simulació del cas del càlcul de recursos finals amb tres millores de recursos . . . 64
6.5 Simulació del cas del càlcul de recursos finals amb una millora de piràmide . . . 65
6.6 Simulació del cas del càlcul de recursos finals amb millores de recursos i piràmide . . . 65
6.7 Simulació del cas del càlcul de recursos finals amb una aventura . 66
6.8 Arbre dels components Vue.js utilitzats en l'aplicació . . . 70
6.9 Logotip *Age of Ra* amb un fons blanc . . . 74
6.10 Logotip *Age of Ra* amb un fons de color blau . . . 74
6.11 Icona *Age of Ra* . . . 74
6.12 Icona interaccionat amb diferents navegadors web . . . 75
6.13 *Assets* utilitzats per dissenyar el videojoc . . . 76
6.14 Disseny de la part central de la terra on s'ubiquen edificis . . . 77
6.15 Diferents estils de la piràmide depenent del nivell . . . 77
6.16 Art conceptual del conjunt d'edificis del *Age of Ra* . . . 78
6.17 Disseny dels recursos que formen *Age of Ra* . . . 79
6.18 Disseny final de les caselles de recursos d'una terra . . . 79
6.19 Disseny final de les tropes de *Age of Ra* . . . 80
6.20 Disseny final de la pantalla d'inici . . . 80
6.21 Disseny final de la pantalla d'inici de sessió . . . 81
6.22 Disseny final de la pantalla d'inici de sessió amb errors . . . 81
6.23 Disseny final de la pantalla de registre d'usuari . . . 82
6.24 Disseny final de la pantalla de registre d'usuari amb errors . . . 82
6.25 Disseny final de la pantalla de joc exterior . . . 83
6.26 Disseny final de la pantalla per millorar un recurs . . . 84
6.27 Disseny final de la pantalla per crear una terra . . . 84
6.28 Disseny final de la pantalla de joc interior . . . 85
6.29 Disseny final de la pantalla de millora de la piràmide . . . 86

6.30 Disseny final de la pantalla de la millora d'un edifici 86
6.31 Disseny final de la pantalla de construcció d'un edifici 87
6.32 Disseny final de la cua de construccions 88
6.33 Disseny final de la cua d'aventures 88
6.34 Disseny final de la pantalla per reclutar tropes 89
6.35 Disseny final de la pantalla del rànquing 89
6.36 Disseny final de la pantalla de *timeline* 90
6.37 Disseny final de la pantalla per tancar la sessió 90
6.38 Disseny final de la pantalla principal dels administradors 91
6.39 Disseny final de la pantalla per editar aventures dels administradors 91
6.40 Disseny final de la pantalla per editar edificis dels administradors 92
6.41 Disseny final de la pantalla per editar recursos dels administradors 92

7.1 Codi del test de rendiment 96
7.2 Resultats del test de rendiment amb un usuari 97
7.3 Resultats del test de rendiment amb 10 usuaris 98
7.4 Resultats del test de rendiment amb 100 usuaris 99
7.5 Resultats del test de rendiment amb 500 usuaris 100
7.6 Sistema *Age of Ra* amb un nou recurs 101

8.1 Distribució en percentatge del cost temporal del projecte 104

11.1 Diagrama de seqüència D_SEC_UC1.1_Inici de sessió usuari . . . 111
11.2 Diagrama de seqüència D_SEC_UC1.2_Registre usuari 112
11.3 Diagrama de seqüència D_SEC_UC2.1_Canviar de terra 113
11.4 Diagrama de seqüència D_SEC_UC2.2_Crear nova terra 114
11.5 Diagrama de seqüència D_SEC_UC3.1_Mostrar *timeline* 115
11.6 Diagrama de seqüència D_SEC_UC3.2_Mostrar rànquing 116
11.7 Diagrama de seqüència D_SEC_UC4.1_Fer aventura 117
11.8 Diagrama de seqüència D_SEC_UC4.4_Millorar recurs 118
11.9 Diagrama de seqüència D_SEC_UC4.9_Construir edifici 119

Taules

8.1 Distribució de les hores per categoria 104

Acrònims

AOR: Age Of Ra

API: Application Programming Interface

REST: Representational State Transfer

CPU: Central Processing Unit

ERS: Especificacions de requisits de software

HTML: HyperText Markup Language

IEEE: Institute of Electrical and Electronics Engineers

JSON: JavaScript Object Notation

MMORTS: Massively Multiplayer Online Real-Time Strategy

PHP: PHP: Hypertext Preprocessor

SQL: Structured Query Language

SPA: Single-Page Application

SSOT: Single Source of Truth

UML: Unified Modeling Language

Capítol 1

Introducció

Aquest capítol ofereix context i motivació per al desenvolupament d'un joc digital d'estratègia en temps real, definint clarament els termes clau. Esbossa els objectius, la metodologia de treball i l'abast del projecte per proporcionar una visió completa del mateix.

1.1 Context

Els videojocs d'estratègia en temps real multijugador massius en línia o MMORTS són videojocs d'estratègia que permeten als jugadors interactuar a temps real amb un món virtual, marcat per una temàtica concreta.

Aquests videojocs tenen diferents temàtiques, però solen basar-se en el mateix concepte: el jugador és un líder d'un poblat que haurà de gestionar els recursos econòmics, la diplomàcia i la formació d'un exèrcit per defensar-se o atacar als altres jugadors.

A més, el joc avança en un món virtual que es desenvolupa i evoluciona independentment que els jugadors estiguin connectats o no.

Al voltant de l'any 2008, una època on es començaven a comercialitzar els smartphones i on el públic objectiu d'aquests videojocs -els adolescents- cada cop tenia més accés a ordinadors personals, els MMORTS van començar a prendre força i popularitat. Això és degut a una de les seves principals avantatges a temps anteriors respecte a videojocs d'altres gèneres: no requerien tants requisits de CPU ni memòria en els ordinadors, al estar dissenyats per executar-se en navegadors web sense gràfics espectaculars.

D'igual forma, amb el transcurs del temps i l'evolució dels videojocs, es fa èmfasi en els gràfics i la resolució d'aquests, els videojocs MMORTS han perdut la seva

popularitat entre el públic general, passant a ocupar un nínxol de mercat amb una audiència reduïda però fidel.

En aquest treball es vol analitzar com funcionaven en els anys de major popularitat els MMORTS per navegadors web sense eines modernes com game engines ni *frameworks*. També es vol comprendre les restriccions i dificultats que es podien trobar en el desenvolupament dels MMORTS mitjançant l'ús de les eines que han aparegut des de la seva època daurada.

1.2 Objectius del treball

El principal objectiu d'aquest projecte és dissenyar i implementar un videojoc multijugador d'estratègia en temps real amb una temàtica pròpia i diferent a les populars del mercat. A més, es busca analitzar i solucionar els possibles problemes de rendiment que afecten a aquests tipus de jocs, especialment respecte el temps de càrrega i de resposta del sistema.

Durant el desenvolupament del videojoc s'aplicaran tècniques i patrons pròpies del Clean Code [1] utilitzant des d'un inici un disseny eficient, patrons de *software* i una completa documentació del codi i eines utilitzades.

Per últim, es vol que el videojoc sigui totalment funcional des del dia que s'obri al públic. L'objectiu seria que encara que una persona estigués jugant completament sol, l'experiència fos semblant a jugar amb gent.

Per a aconseguir una millor evolució del projecte, s'ha procedit a dividir els objectius en metes individuals:

- Fer un estudi de mercat per veure la competència activa i proposar una solució innovadora i amb potencial de competir.
- Tenir un joc el més segur possible: aplicar tècniques de seguretat tant per la part de *frontend* com per la part de *backend*.
- Fer un disseny escalable de forma que, en el pitjor cas, l'aplicació respongui de forma lineal a mesura que augmenten els usuaris connectats al videojoc.
- Fer un disseny flexible de forma que sigui fàcil afegir-hi nou contingut sense implicar canvis al codi existent.
- Publicar el videojoc i fer proves amb públic real.

1.3 Metodologia

La metodologia emprada durant el transcurs d'aquest projecte ha estat el desenvolupament en espiral [2]. Aquesta metodologia combina el model tradicional *waterfall* i el model per iteracions. El desenvolupament amb aquesta metodologia

passa per les següents etapes: conceptualització, desenvolupament, millores i manteniment.

Així, aquest model permet començar amb un petit conjunt de tasques a implementar que sempre han de passar per totes les fases i revisions pertinents. Conforme es van afegint més funcionalitats, el projecte va creixent (en forma d'espiral) fins a tenir realitzada una primera versió final del producte.

S'ha escollit aquesta metodologia per adaptar-se a les necessitats del projecte, al ser un desenvolupament d'un joc amb diverses funcionalitats a incorporar amb unes dates d'entrega limitades i ser un projecte individual on per motius de calendari no es corresponia utilitzar metodologies àgils.

1.4 Estructura del document

Els continguts d'aquest document estan estructurats en capítols que resumeixen el desenvolupament del joc *Age of Ra*. A continuació, es detallen tots els capítols amb una breu explicació:

- **Capítol 1 - Introducció**: S'explica el context, motivació i abast d'aquest document. També es declaren els objectius principals del projecte i la metodologia que s'ha seguit.
- **Capítol 2 - Estudi de mercat**: S'estudien i comparen tots els jocs actualment funcionals i rendibles del mercat.
- **Capítol 3 - Especificacions de requisits de *software***: Es descriuen les especificacions dels requisits de *software* seguint l'estàndard IEEE Std 830-1998.
- **Capítol 4 - Model de dades**: Es descriu el model de dades a partir dels requisits de *software* a través d'un model estàtic i un model dinàmic.
- **Capítol 5 - Disseny tecnològic**: Es descriu el disseny tecnològic del sistema a partir del capítol anterior, tot actualitzant i acabant de definir el model.
- **Capítol 6 - Implementació**: Es descriuen els detalls de més importància de la implementació dels requisits de *software*, tant implementació tecnològica com de disseny.
- **Capítol 7 - Experimentació i resultats**: Es descriu tot el procés de testing o experimentació portat a terme per assegurar que el *software* compleix amb els requisits descrits al seu apartat.
- **Capítol 8 - Cost del projecte**: Es detallen les hores invertides en el projecte.

- **Capítol 9 - Conclusions**: Es detallen les conclusions tècniques i personals del projecte. També es llistaran totes aquelles funcionalitats que no ha donat temps de desenvolupar i es deixen per una segona versió del joc.
- **Capítol 10 - Referències**: Es llisten totes les referències utilitzades en aquest document i durant el desenvolupament del projecte.
- **Capítol 11 - Annex**: Finalment, s'adjunten en aquest capítol totes aquelles figures, elements o taules que formen part de l'annex.

Capítol 2

Estudi de mercat

En aquest capítol es fa una comparativa entre quatre dels videojocs més populars pertanyents al gènere MMORTS, per arribar a una proposta innovadora i de temàtica diferent per competir en el mercat.

Per seleccionar els videojocs, s'ha tingut en compte una anàlisi exhaustiva del mercat dels MMORTS. Es va considerar la popularitat dels diferents videojocs, així com el seu nombre de jugadors concurrents durant l'any 2008-2022.

Per fer la comparativa entre els diferents videojocs s'assenyalen els punts forts i els febles de cadascun d'ells, així com les diferents eines que utilitzen. Finalment, es dedica un breu apartat per extreure conclusions dels punts forts dels videojocs del mercat i marcar uns punts per la proposta de videojoc.

2.1 Travian: Legends

Travian [3] és un videojoc del gènere MMORTS ambientat en la època de la Pax Romana.

En aquest videojoc el jugador comença escollint un dels tres pobles principals: romans, germans o gals. Cada poble té unes característiques diferents que aporten unes avantatges o inconvenients a l'hora de jugar en el servidor. Això també millora la rejugabilitat, fent que sigui més difícil caure en la monotonia i facilitant que l'usuari pugui jugar en diferents servidors.

Quan es comença en un servidor, es crea un poble dins d'un plànol de coordenades cartesianes. Tots els espais veïns poden estar controlats per pobles sense ocupar, enemics o bots. Es pot observar el mapa dels poblats veïns a la Figura 2.1.

Figura 2.1: Mapa on es situa la terra del jugador

Per anar avançant en el joc, el jugador ha de construir edificis al seu poblat (Figura 2.2). Per construir edificis es necessiten recursos que es generen cada hora. Un cop edificats certs edificis es podrà començar a formar un exèrcit.

Figura 2.2: Interior i exterior d'una terra

L'objectiu principal del joc és sobreviure el temps suficient (varia depenent del servidor) per arribar a construir un tipus especial d'estructura anomenada "Meravella" a través d'una aliança de poblats.

En termes tecnològics, s'ha pogut esbrinar que els servidors corren en Ubuntu 20 utilitzant Apache, MySQL Server i PHP 7.4. No utilitzen cap game engine pel que fa al *frontend*, sinó que superposen totes les imatges amb diferents nivells de profunditat amb CSS.

Finalment, es farà una breu menció als punts forts i febles del videojoc tractat en aquest apartat.

Punts forts:

- Videojoc de navegador més popular actualment a Turquia.
- Joc de pantalla completa: els gràfics s'integren en qualsevol pantalla indiferentment de la mida del dispositiu (tant mòbil com ordinador).
- Existeix un tutorial complet i seguit d'ajudes per entendre el joc.
- Existeixen molts servidors actius de diferents nacionalitats.
- El joc té un objectiu clar i comú per tots els jugadors.

Punts febles:

- Cada cop que es consulta a qualsevol informació del joc, aquest tarda força a carregar al fer moltes consultes a la base de dades.
- Existeix un temps de resposta superior a 1 segon en la majoria de les peticions. Segons diversos estudis ([4]), un temps de resposta de més de 1 segon és considerat generalment com a lent i pot generar una experiència d'usuari insatisfactòria.
- Els gràfics estan dissenyats sobre imatges per superposició sense utilitzar cap eina, fet que sigui difícil afegir noves funcionalitats.
- Massa informació de cop a través de molts pop-ups o menús de navegació.
- Cal fer clics a tot arreu per consultar edificis, recursos, classificacions o mapa, fet que requereix una recàrrega de pàgina.

2.2 Guerras tribales

Guerras Tribales [5] és un videojoc del gènere MMORTS ambientat en l'edat mitjana. Aquest videojoc té la mateixa jugabilitat que Travian, per tant, en aquest estudi de mercat no es n'explicaran tant els mecanismes. La diferència principal és que l'objectiu del jugador en aquest joc és expandir al màxim el seu imperi, construint edificis, recursos i tropes per augmentar les conquestes i poblats en el temps que dura el servidor (Figura 2.3).

Figura 2.3: Mapa on es situa la terra d'un jugador

Existeixen menys tipus d'edificis i recursos que el Travian i gairebé les mateixes tropes. Tot i això, no existeixen poblats diferents i, per tant, tots tenen el mateix estil i jugabilitat (Figura 2.4).

Figura 2.4: Interior de la terra d'un jugador

La tecnologia utilitzada en aquest cas no es coneix amb certesa, per tant, no és possible afirmar amb precisió quina plataforma o llenguatge específic s'ha utilitzat en el desenvolupament del videojoc. Tot i això, es poden trobar indicis, com la presència d'enllaços que acaben en ".php" i l'ús de *cookies* com *PHPSESSID*. Aquests indicis suggereixen que podria utilitzar-se una tecnologia com PHP

juntament amb MySQL Server o similars per al seu desenvolupament.

Finalment, es farà una breu menció als punts forts i els punts febles del videojoc tractat en aquest apartat.

Punts forts:

- Videojoc popular en Alemanya (país de creació del videojoc).
- Existeixen força jugadors en actius separats per servidors nacionals.
- Existeix una versió per Android i iOS del videojoc.
- Temps de càrrega de les pantalles més fluid.
- El videojoc no és estàtic, cada cop que es recarrega la pàgina té un efecte aleatori apareixent diferents gràfics animats al poblat.

Punts febles:

- Tot i ser més fluid, cada cop que es consulta qualsevol informació del joc, la càrrega no és immediata i es pot experimentar un temps de resposta superior a 1 segon.
- Els gràfics del videojoc no han estat actualitzats amb el temps.
- Els gràfics estan dissenyats sobre imatges per superposició sense utilitzar cap eina, fet que sigui difícil afegir noves funcionalitats.
- Molta informació a la pantalla i molts menús de navegació.
- No existeix un tutorial per guiar al jugador principiant.
- D'igual forma que el Travian, cal fer clics a tot arreu per consultar edificis, recursos, classificacions o mapa, fet que requereix una recàrrega de pàgina.

2.3 OGame

OGame [6] és un videojoc del gènere MMORTS ambientat en l'espai i recerca de nous planetes.

De nou, aquest videojoc implementa les mateixes mecàniques que la resta, en tant que pertany al mateix gènere (Figura 2.5). En aquest cas, els jugadors prenen control d'un planeta en comptes d'un poble, gastant els seus recursos en edificis futuristes i un exèrcit en forma de flota espacial.

Així, el joc es divideix en edificis (estructures que aporten beneficis diferents), investigacions (millores per permeten construir noves naus, edificis o defenses), naus (exèrcit atacant), defenses (exèrcit defensiu) i finalment aliances (agrupacions de jugadors aliats). En aquest cas, cada servidor o "Univers" consta d'una

població de 16.500 jugadors.

Figura 2.5: Visió general del planta principal

La tecnologia utilitzada en aquest cas no és tan transparent com la de Travian. Tot i això, pel tipus de gràfics i època de creació del videojoc, es pot arribar a imaginar que s'empri algun tipus de bases de dades relacional juntament amb PHP, o tecnologies similars.

Finalment, es farà una breu menció als punts forts i els punts febles del videojoc tractat en aquest apartat.

Punts forts:

- Videojoc més popular al 2007, encara té una base de jugadors en línia molt forta.
- Com que no existeixen gràfics animats, el joc se sent més responsiu.
- La majoria de les accions i funcionalitats del videojoc estan detalladament explicades amb extenses descripcions.
- Existeix una versió per Android i iOS.

Punts febles:

- No existeixen gràfics animats en el videojoc, només imatges.
- No existeix cap tutorial per guiar al jugador principiant.

- D'igual forma que els altres videojocs, cal fer clics a tot arreu per consultar edificis, recursos, classificacions o mapa, fet que requereix una recàrrega de pàgina.
- Existeix un excés de menús de navegació.

2.4 Ikariam

Desenvolupat pel mateix equip que OGame, Ikariam [7] és un videojoc del gènere MMORTS ambientat en l'antiga Grècia espai i amb l'objectiu de gestionar una polis i expandir les seves colònies.

Aquest videojoc és molt similar al Travian o Guerras Tribales. El jugador comença amb una ciutat en una illa grega on haurà d'invertir recursos per construir edificis, expandir el seu exèrcit i fer investigacions en tecnologia per millorar-les.

Aquest món virtual està format per molts arxipèlags (Figura 2.6). Cada un d'aquests arxipèlags té illes que el formen amb vuit meravelles diferents.

Figura 2.6: Mapa de totes les illes veïnes

Aquestes illes estan formades per alguns recursos, font primària per construir els edificis de la ciutat (Figura 2.7).

Figura 2.7: Interior d'una illa del jugador

Les diferents meravelles que existeixen actuen com un edifici que bonifica al jugador o bé amb recursos o bé altres avantatges.

A diferència dels seus competidors, els servidors no acaben en una data determinada ni existeix un final del joc o objectiu a complir. També, el joc es centra més en el comerç i intercanvi de recursos amb les illes veïnes que en atacar.

D'igual forma que la majoria dels seus competidors, la tecnologia utilitzada no és tan transparent. Tot i això, pel tipus de gràfics i època de creació del videojoc, es pot arribar a imaginar que s'empri algun tipus de bases de dades relacional i algun *game engine*.

Finalment, es farà una breu menció als punts forts i els punts febles del videojoc tractat en aquest apartat.

Punts forts:

- Joc de pantalla completa amb gràfics interactius.
- La majoria d'interaccions amb el videojoc obren petits menús per superposició sense recarregar la pàgina.
- Joc molt fluid i gens lent. Existeix un temps de resposta entre 100 mil·lisegons i 1 segon, que es considera òptim per a una interacció fluida i sense retard en un pàgina web ([4]).

- Existeix un tutorial complet i ajudes per entendre el joc.
- El joc és infinit i no existeix tanta competitivitat, un jugador no pot ser eliminat del servidor.
- Existeix una versió per Android i iOS del videojoc.

Punts febles:

- Al ser un joc infinit, per un jugador que acaba de començar, pot ser més difícil introduir-se que els videojocs competidors, a l'existir jugadors que van més avançats.
- Tot i no existir tants menús, cal fer clics a tot arreu per consultar illes, edificis, recursos o classificacions.

2.5 Conclusions de l'estudi de mercat

Un cop analitzat els quatre videojocs de gènere MMORTS més populars i rendibles del mercat, cal extreure conclusions dels seus punts forts que els han fet dominar el sector i els seus punts febles més importants que els hi ha faltat millorar.

Per detallar aquestes conclusions, es proposa un nou videojoc que incorpori els punts forts trobats i millori els punts febles més característics.

Per tant, una possible proposta de videojoc de gènere MMORTS que podria encaixar en el mercat i trobar un lloc amb el qual competir podria tenir les següents característiques:

- Nom del videojoc: *Age of Ra*
- Videojoc enfocat a la construcció i gestió d'un imperi egipci.
- Videojoc de pantalla completa: els gràfics s'integren en qualsevol pantalla indiferentment de la mida del dispositiu (tant mòbil com ordinador).
- Existència d'un tutorial complet o bé d'ajudes al jugador principiant.
- Objectiu clar i comú per tots els jugadors, indiferentment del nivell d'experiència.
- Joc fluid a través de gràfics senzills. Preferència per la fluïdesa al visual.
- Accions i funcionalitats suficientment detallades amb extenses descripcions.
- Videojoc de caràcter infinit, és a dir, existeix la possibilitat d'incorporar noves funcionalitats pels jugadors amb més experiència.
- Reduir l'ús de menús de navegació.

- Reduir els clics per arribar a realitzar accions i funcionalitats.
- Evitar o minimitzar la recàrrega completa de pàgines.

Capítol 3

Especificacions de requisits de *software*

En aquesta capítol es descriuen les especificacions dels requisits de *software* per llançar la primera versió del videojoc *Age of Ra*.

3.1 Introducció

Aquest document d'Especificació de Requisits de *Software* (ERS) descriu un model basat en especificacions inequívoques, funcionals i completes per al desenvolupament del *software* del videojoc *Age of Ra*.

L'objectiu principal d'aquest ERS és proporcionar una base clara i concisa per al desenvolupament del *software*, amb la intenció de reduir l'esforç d'implementació, estimar els costos i horaris necessaris, validar el *software* i establir una línia de base per a futures millores del model.

3.1.1 Propòsit

El videojoc *Age of Ra* és un videojoc d'estratègia en línia multijugador massiu amb la finalitat de construir edificis i recursos per ser l'emperador màxim i conquerir la primera posició del rànquing de recursos del joc.

Està enfocat a un públic específic i reduït, amant dels jocs d'estratègia, on en comptes de dominar els gràfics i animacions, prefereixen un disseny senzill i usable a l'hora de jugar en el navegador web.

3.1.2 Abast

Aquest ERS s'enfoca en el desenvolupament del *software* del videojoc *Age of Ra* per navegadors web.

El videojoc té una temàtica egípcia i gira entorn a la construcció i les aventures. Construir edificis o entrenar noves tropes s'efectua mitjançant l'ús de recursos que s'aconsegueixen a través del que generen les caselles de recursos.

Aquest *software* inclourà totes les funcionalitats per crear un usuari, iniciar sessió, interactuar amb el joc, mostrar estadístiques i crear noves terres.

Destacar que actualment no es contemplen possibles funcionalitats socials del *software* (enviament i recepció de missatges entre jugadors o un xat global), ni tampoc un mode de batalles entre jugadors.

3.1.3 Definicions, sigles i acrònims

L'ús de sigles, acrònims i definicions són els mateixos explicats a la seva secció corresponent.

3.1.4 Referències

IEEE Recommended Practice for *software* Requirements Specifications. (IEEE Std nº 830-1998) [8].

3.1.5 Resum

Aquest ERS es basa en tres grans seccions. La primera secció és una introducció on s'explica el propòsit, abast, acrònims i referències utilitzades. En la segona secció es tracta d'una descripció general definint les funcionalitats del producte i les restriccions. En la tercera secció s'estableix amb més detalls els requisits tècnics i les funcionalitats del desenvolupament del *software*.

3.2 Descripció general

La descripció general és una secció que proporciona una visió general del producte que s'està desenvolupant. En aquesta secció s'inclou informació sobre la perspectiva del producte, les funcionalitats del producte, les característiques de l'usuari, suposicions i dependències, i els requisits de futur.

3.2.1 Perspectiva del producte

El joc és un producte totalment independent. No està relacionat amb altres productes que no siguin el sistema principal encarregat de córrer el joc. A la Figura 3.1 es mostra un possible desplegament del sistema a nivell informal.

Més endavant es farà un diagrama de desplegament normatiu d'UML un cop coneguts totes les funcionalitats i detalls del disseny que es vol implementar.

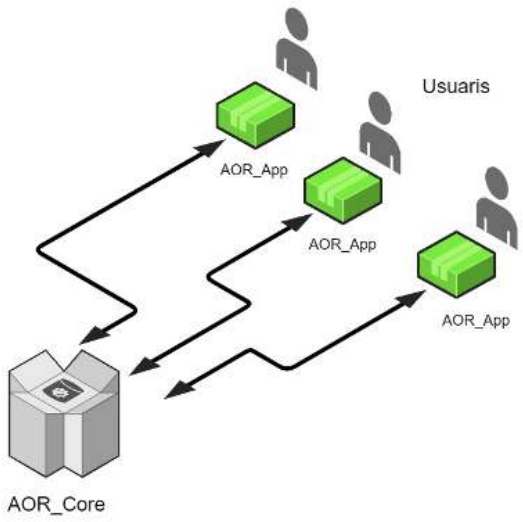

Figura 3.1: Diagrama conceptual de l'arquitectura del sistema *Age of Ra*

Els usuaris accediran al sistema mitjançant una aplicació (AOR_App) de navegador web, el qual permetrà a l'usuari realitzar totes les funcionalitats que definirem més endavant. Aquest sistema es connectarà a través d'internet amb un servidor central.

El servidor central (AOR_Core) disposarà d'un entorn d'execució, que s'encarregarà de facilitar els serveis necessaris a les aplicacions client.

3.2.1.1 Interfícies d'usuari

El sistema només proveirà d'una sola interfície d'usuari AOR_App. El sistema proveirà una interfície a través de la pantalla de l'usuari en la que es podran realitzar totes les funcionalitats que el sistema ofereix. És important destacar que l'usuari podrà interactuar amb el sistema a través del navegador web de qualsevol dispositiu electrònic amb accés a internet. Cal destacar que existeix un usuari amb drets administratius que tindrà més funcionalitats per gestionar el videojoc.

3.2.2 Funcionalitats del producte

En aquesta secció de l'ERS definirem el comportament que ha de tenir el sistema, els seus serveis, funcions i tasques que haurà de ser capaç de fer:

- **Gestió d'usuaris**: El sistema AOR haurà de permetre a l'usuari fer un

registre de nous usuaris per accedir al sistema. També s'haurà de permetre fer un inici de sessió per habilitar totes les altres funcionalitats.

- **Gestió de terres**: El sistema ha de permetre per una gestió de les terres que controla cada jugador. En aquestes gestions estan incloses les funcionalitats com canviar la terra que es mostrà per pantalla i crear una nova terra.
- **Estadístiques**: El sistema ha de permetre visualitzar un seguit d'estadístiques o del jugador o de la terra que s'està mostrant. Es permetrà mostrar estadístiques amb una línia de temps dels esdeveniments passats de les terres i un rànquing global dels jugadors.
- **Gestió del joc**: El sistema haurà de permetre a l'usuari tot un seguit de funcionalitats per poder interactuar, visualitzar, construir i millorar recursos, edificis i piràmide, reclutar tropes i realitzar aventures.
- **Gestió de l'administrador**: El sistema haurà de permetre a l'usuari de caràcter administrador poder visualitzar els arxius interns dels recursos, edificis, tropes i aventures, així com afegir-ne de nous.

3.2.3 Característiques de l'usuari

El joc està principalment enfocat a un públic nínxol i reduït. L'usuari perfecte ha de tenir temps lliure per jugar i li ha d'agradar passar el temps davant l'ordinador per estar constantment construint i fent aventures. L'usuari normalment tindrà experiència d'altres jocs multijugador similars.

3.2.4 Suposicions i dependències

En tot l'ERS s'ha parlat sobre com efectuar el videojoc enfocat a navegadors webs. En els requeriments està contemplat que sigui funcional en tots els navegadors web amb més popularitat, independentment de futures versions de navegadors web.

3.2.5 Requisits a futur

Els requisits a futur més destacables són tota la part social que el joc podria oferir: missatges entre jugadors, possibilitat d'un xat global, aliances entre jugadors, batalles entre jugadors i intercanvi de recursos.

3.3 Requisits específics

En un ERS, els requisits específics són aquells que descriuen detalladament les característiques i funcionalitats del producte i es poden dividir en dues categories: requisits funcionals i requisits no funcionals.

3.3.1 Requisits funcionals del producte

En aquest apartat descriurem els requisits funcionals que ha d'oferir el sistema de forma més tècnica i detallada. En el nostre sistema existeixen dos possibles actors: l'usuari del joc i l'administrador. L'usuari del joc haurà de ser capaç d'interactuar amb 4 mòduls diferents: AOR_M1_Gestió d'usuaris, AOR_M2_Gestió Terra, AOR_M3_Estadístiques i AOR_M4_Gestió Joc. L'administrador haurà de ser capaç d'interactuar amb 2 mòduls diferents: AOR_M1_Gestió d'usuaris i AOR_M5_Gestió Admin. La interacció dels actors amb aquests quatre mòduls es poden observar a la Figura 3.2.

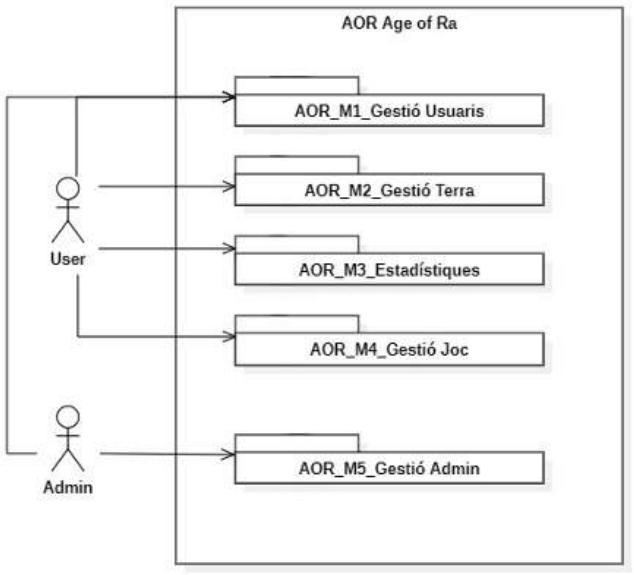

Figura 3.2: Diagrama general de cas d'ús de la interfície AOR (*Age of Ra*)

3.3.1.1 Gestió usuaris

En aquest mòdul l'usuari podrà entrar al sistema fent un inici de sessió o, si no té un usuari, registrar-se per després fer l'inici de sessió. Cal destacar que un usuari de caràcter administrador no podrà registrar-se al sistema. Els seus casos d'ús es poden observar a la Figura 3.2 amb un diagrama d'activitats general a la Figura 3.4.

- **AOR_UC1.1_Inici de sessió usuari**: El sistema permetrà a l'usuari fer un inici de sessió amb un nom d'usuari i una contrasenya.
- **AOR_UC2.1_Registre usuari**: El sistema permetrà a l'usuari registrar un nou usuari amb una contrasenya.

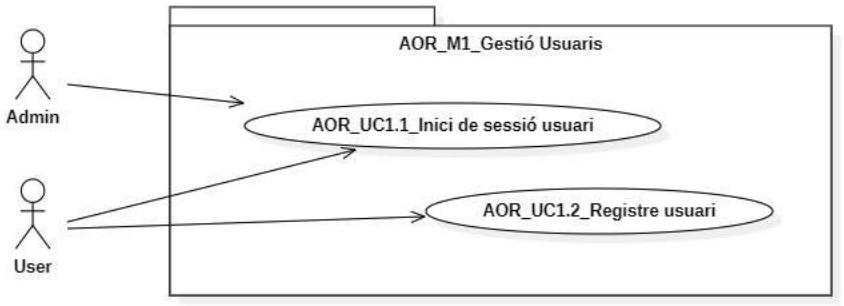

Figura 3.3: Diagrama de cas d'ús del mòdul de gestió d'usuaris

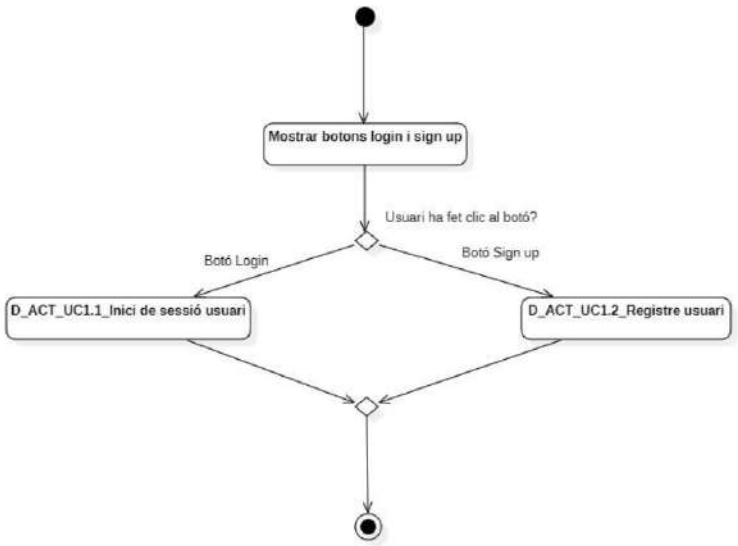

Figura 3.4: Diagrama d'activitats del mòdul de gestió d'usuaris

Per accedir als següents mòduls, l'usuari ha d'haver iniciat sessió al sistema a través de AOR_UC1.1.

3.3.1.2 Gestió terra

En aquest mòdul l'usuari podrà canviar la terra que s'està mostrant per pantalla o crear una nova terra. Els seus casos d'ús es poden observar a la Figura 3.6 amb un diagrama d'activitats general a la Figura 3.7.

- **AOR_UC2.1_Canviar de terra**: El sistema permetrà a l'usuari canviar

entre les diferents terres que són de la seva propietat.

- **AOR_UC2.2_Crear nova terra**: El sistema permetrà a l'usuari crear una nova terra pagant els recursos necessaris. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.5.

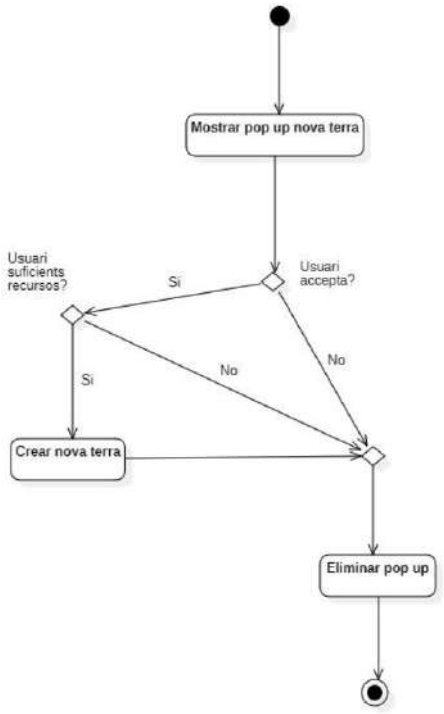

Figura 3.5: Diagrama d'activitats del cas d'ús AOR_UC2.2._Crear nova terra

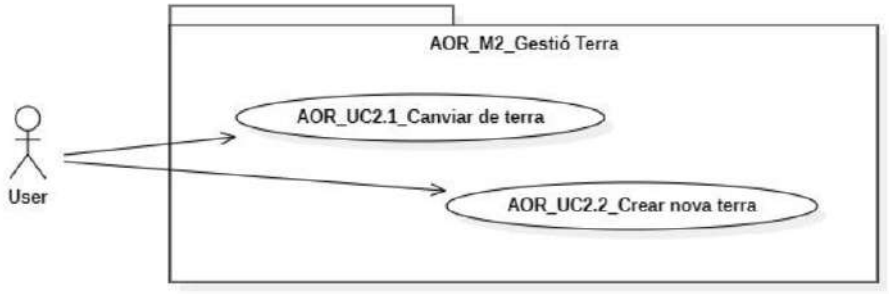

Figura 3.6: Diagrama de cas d'ús del mòdul de gestió de terres

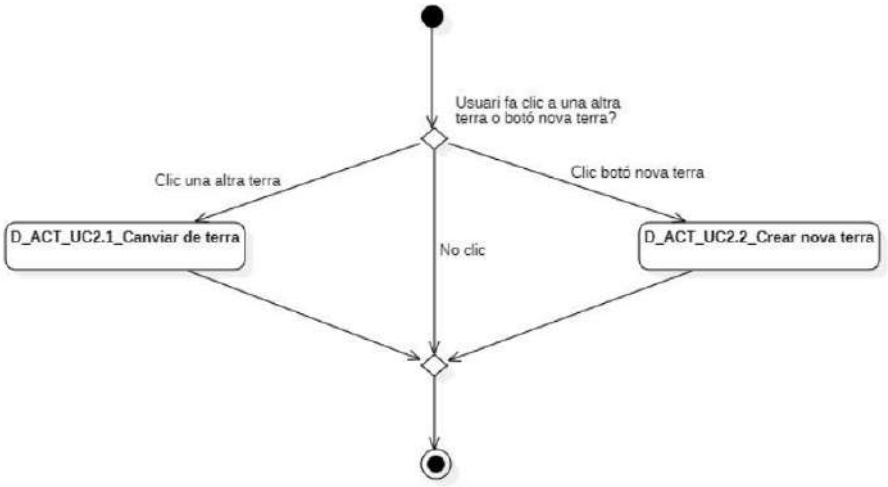

Figura 3.7: Diagrama d'activitats del mòdul de gestió de terres

3.3.1.3 Estadístiques

En aquest mòdul l'usuari podrà mostrar informació relativa a estadístiques de la terra mostrant una línia de temps o un rànquing amb els millors jugadors del joc. Els seus casos d'ús es poden observar a la Figura 3.10 amb un diagrama d'activitats general a la Figura 3.11.

- **AOR_UC3.1_Mostrar *timeline***: El sistema mostrarà una nova pantalla amb tota la informació de les accions que han ocorregut en la terra actual. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.8.

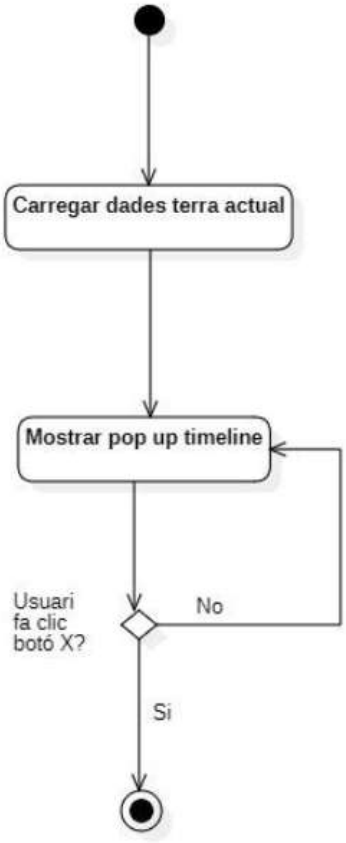

Figura 3.8: Diagrama d'activitats del cas d'ús AOR_UC3.1._Mostrar *timeline*

- **AOR_UC3.2_Mostrar rànquing**: El sistema mostrarà una nova pantalla amb un rànquing de posicionament global entre tots els jugadors on importarà l'ordre de la suma dels recursos actuals de totes les terres. En aquest rànquing només es mostraran els millors 10 jugadors actuals amb els seus recursos totals i la posició global en la qual l'usuari es troba. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.9.

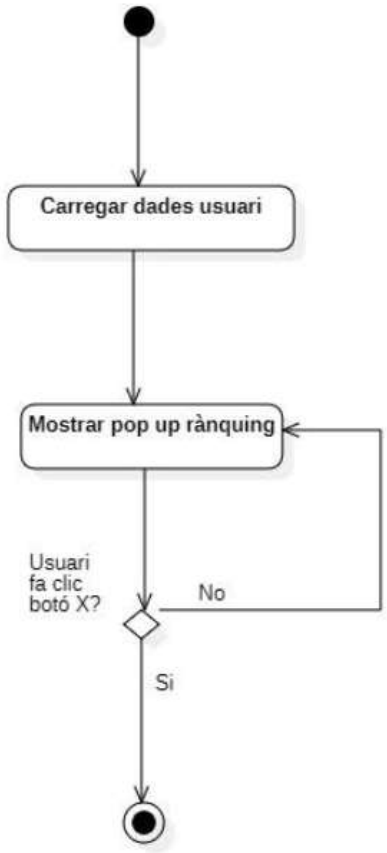

Figura 3.9: Diagrama d'activitats del cas d'ús AOR_UC3.2_Mostrar Rànquing

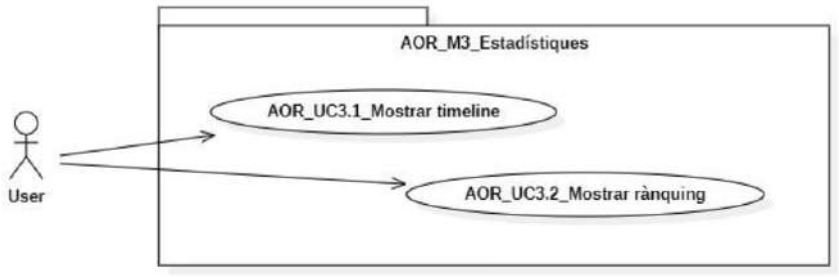

Figura 3.10: Diagrama de cas d'ús del mòdul d'estadístiques

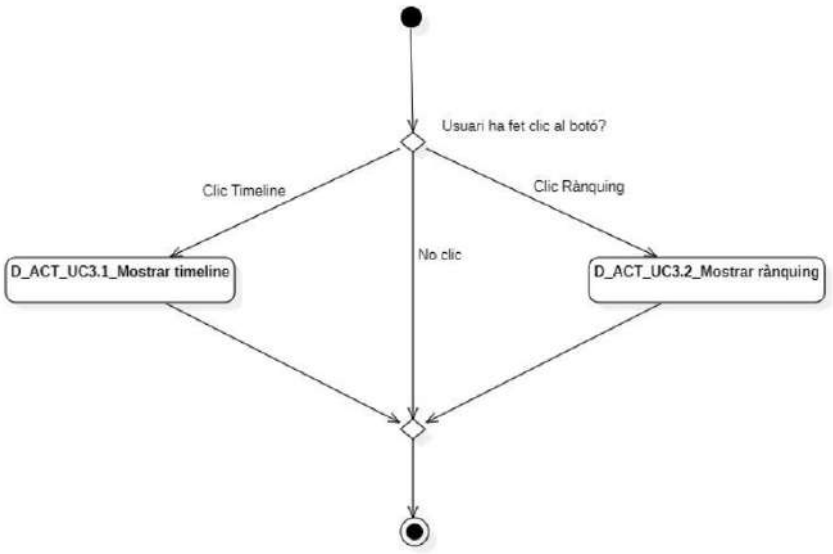

Figura 3.11: Diagrama d'activitats del mòdul d'estadístiques

3.3.1.4 Gestió joc

Aquest mòdul gestionarà tota la lògica i accions que l'usuari pot fer en el joc. Els seus casos d'ús es poden observar a la Figura 3.15 amb un diagrama d'activitats general a la Figura 3.16.

- **AOR_UC4.1_Fer aventura**: El sistema permetrà a l'usuari començar una nova aventura llançant totes les tropes d'aquella terra actual. Si aquella terra no té tropes per començar l'aventura, s'informarà a l'usuari per pantalla. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.12.

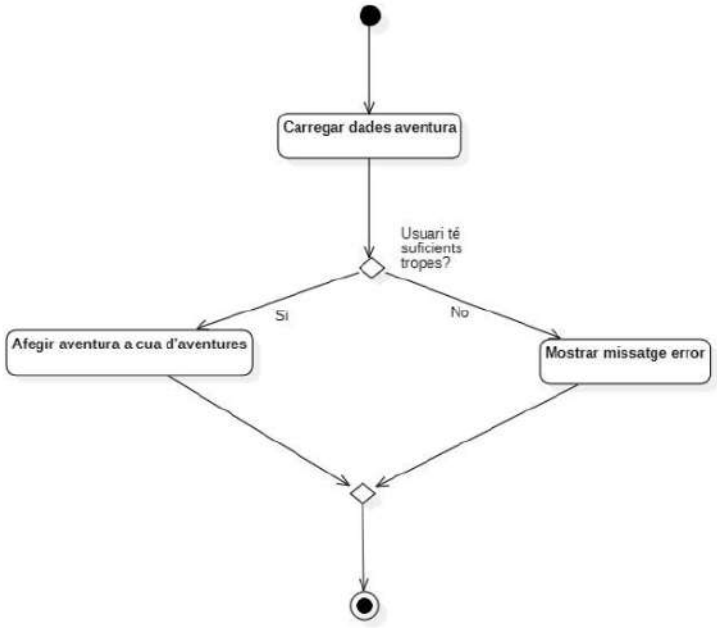

Figura 3.12: Diagrama d'activitats del cas d'ús AOR_UC4.1_Fer aventura

- **AOR_UC4.2_Mostrar exterior terra**: El sistema mostrarà per pantalla l'exterior de la terra actual que es conforma per totes les caselles de recursos. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.13.

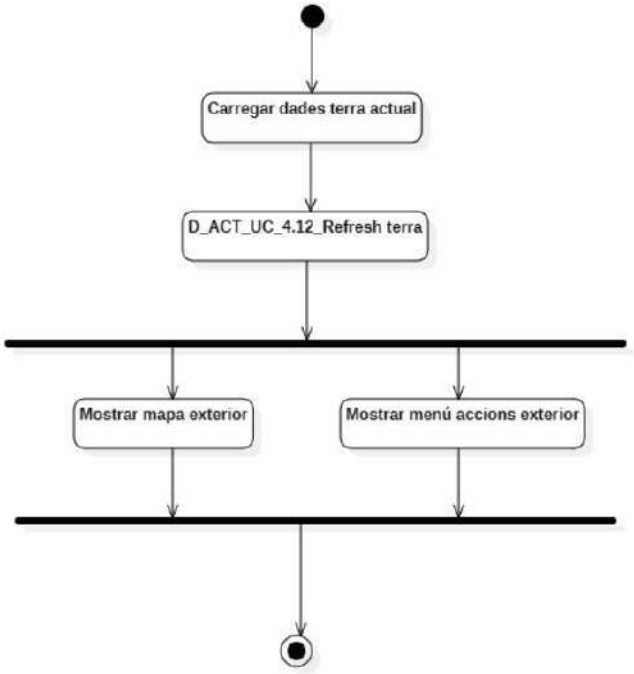

Figura 3.13: Diagrama d'activitats del cas d'ús AOR_UC4.2_Mostrar exterior terra

- **AOR_UC4.3_Seleccionar recurs**: Primer el sistema haurà d'estar mostrant l'exterior de la terra. El sistema permetrà a l'usuari seleccionar una casella de recurs. Un cop seleccionat el recurs es mostrarà informació de millora d'aquest.
- **AOR_UC4.4_Millorar recurs**: Primer el sistema haurà d'estar mostrant l'exterior de la terra i haver seleccionat un recurs. El sistema permetrà a l'usuari millorar el recurs prèviament seleccionat.
- **AOR_UC4.5_Mostrar interior terra**: El sistema mostrarà per pantalla l'interior de la terra actual que es conforma per totes les caselles de construcció i una casella de piràmide.
- **AOR_UC4.6_Seleccionar piràmide**: Primer el sistema haurà d'estar mostrant l'interior de la terra. El sistema permetrà a l'usuari seleccionar la casella de la piràmide. Un cop seleccionada la piràmide es mostrarà informació de millora.
- **AOR_UC4.7_Millorar piràmide**: Primer el sistema haurà d'estar mostrant l'interior de la terra i haver seleccionat la piràmide. El sistema permetrà a l'usuari millorar la piràmide.

- **AOR_UC4.8_Seleccionar edifici**: Primer el sistema haurà d'estar mostrant l'interior de la terra. El sistema permetrà a l'usuari seleccionar una casella de construcció. Un cop seleccionada la casella es mostrarà informació de l'estat actual de la casella o edifici.
- **AOR_UC4.9_Construir edifici**: Primer el sistema haurà d'estar mostrant l'interior de la terra i haver seleccionat una casella de construcció buida. El sistema permetrà a l'usuari construir un edifici a la casella prèviament seleccionada.
- **AOR_UC4.10_Millorar edifici**: Primer el sistema haurà d'estar mostrant l'interior de la terra i haver seleccionat una casella de construcció ocupada. El sistema permetrà a l'usuari millorar l'edifici.
- **AOR_UC4.11_Reclutar tropes**: Primer el sistema haurà d'estar mostrant l'interior de la terra i haver seleccionat una casella de construcció ocupada amb un edifici capaç de reclutar tropes. El sistema permetrà a l'usuari reclutar tropes.
- **AOR_UC4.12_Refresh terra**: El sistema efectuarà una recàrrega de la terra, actualitzant els esdeveniments, aventures, tropes, recursos i edificis. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.14.

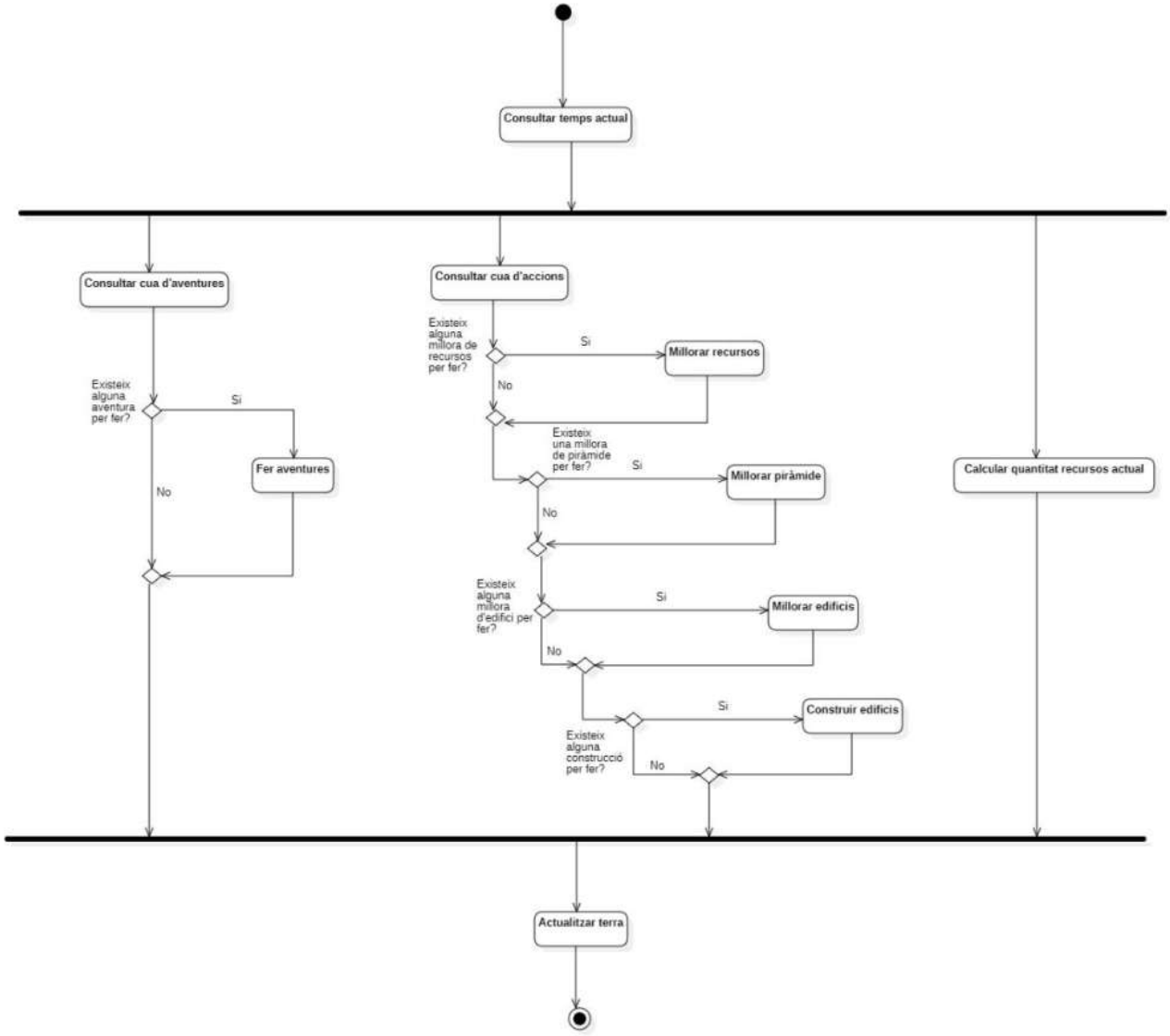

Figura 3.14: Diagrama d'activitats del cas d'ús AOR_UC4.12_Refresh terra

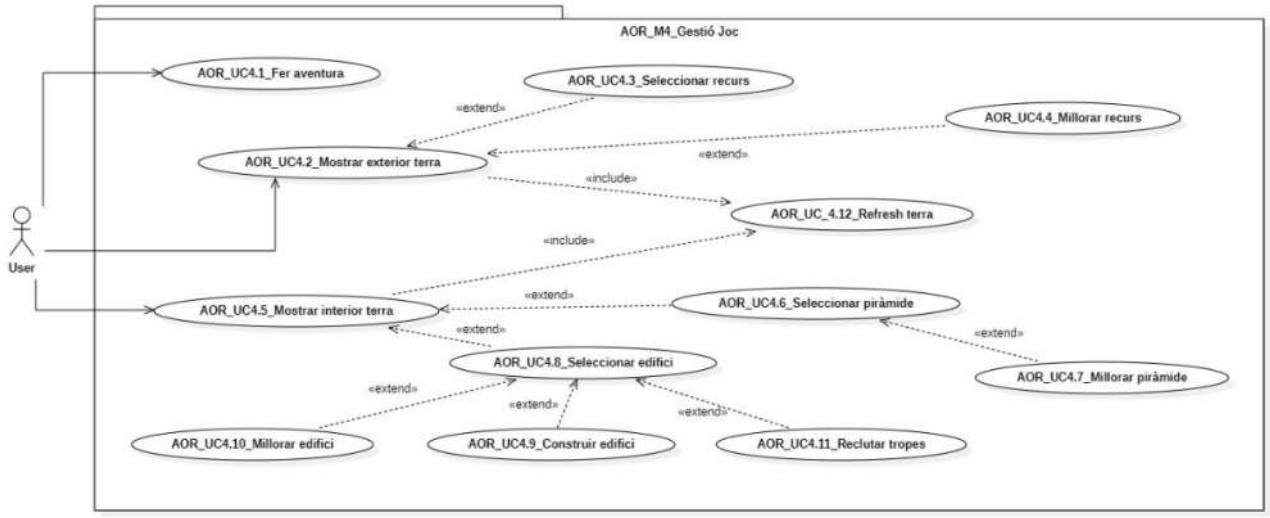

Figura 3.15: Diagrama de cas d'ús del mòdul de gestió del joc

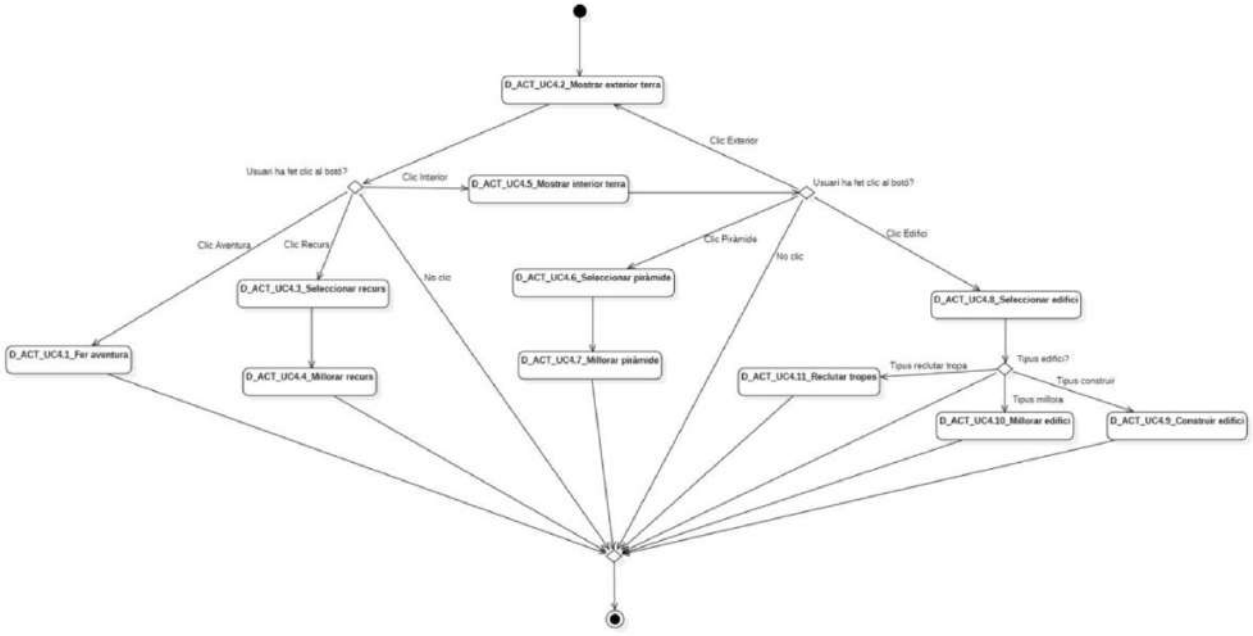

Figura 3.16: Diagrama d'activitats del mòdul de gestió del joc

3.3.1.5 Gestió admin

L'últim mòdul gestionarà tota la lògica i accions que l'usuari pot fer en el joc. Els seus casos d'ús es poden observar a la Figura 3.19 amb un diagrama d'activitats general a la Figura 3.20.

- **AOR_UC5.1_Mostrar recursos**: El sistema permetrà a l'usuari administrador visualitzar tots els recursos del sistema. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.17.

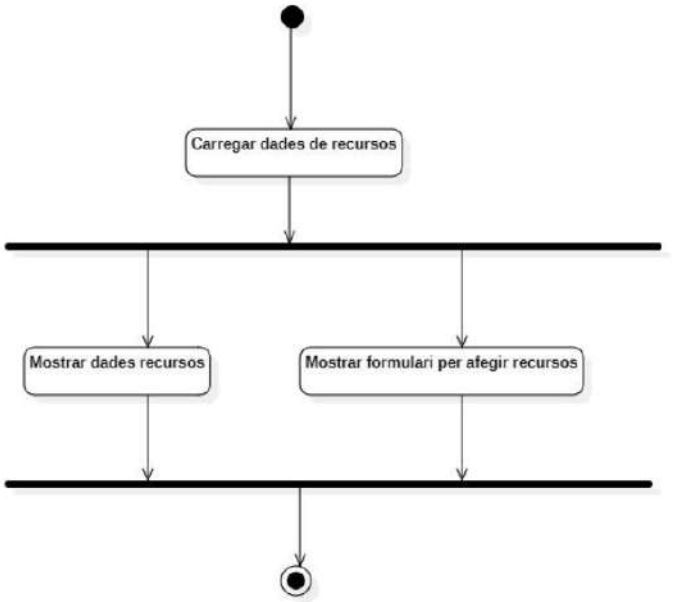

Figura 3.17: Diagrama d'activitats del cas d'us AOR_UC5.1_Mostrar recurs

- **AOR_UC5.2_Mostrar aventures**: El sistema permetrà a l'usuari administrador visualitzar tots les aventures del sistema.
- **AOR_UC5.3_Mostrar tropes**: El sistema permetrà a l'usuari administrador visualitzar tots les tropes del sistema.
- **AOR_UC5.4_Mostrar edificis**: El sistema permetrà a l'usuari administrador visualitzar tots els edificis del sistema.
- **AOR_UC5.5_Afegir recurs**: El sistema permetrà a l'usuari administrador afegir un nou tipus de recurs. Es pot observar un diagrama d'activitats d'aquest cas d'ús més detallat a la Figura 3.18.

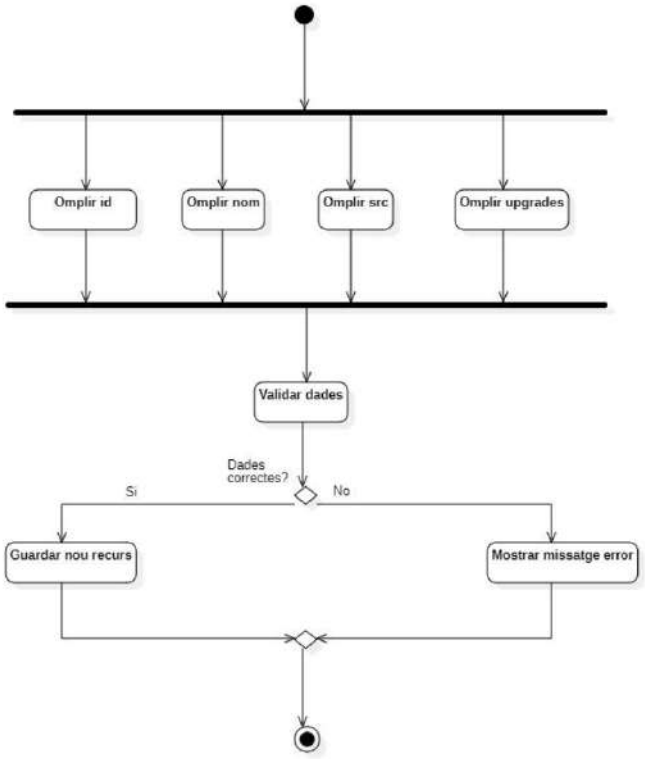

Figura 3.18: Diagrama d'activitats del cas d'us AOR_UC5.5_Afegir recurs

- **AOR_UC5.6_Afegir aventura**: El sistema permetrà a l'usuari administrador afegir un nou tipus d'aventura.
- **AOR_UC5.7_Afegir tropa**: El sistema permetrà a l'usuari administrador afegir un nou tipus de tropa.
- **AOR_UC5.8_Afegir edifici**: El sistema permetrà a l'usuari administrador afegir un nou tipus d'edifici.

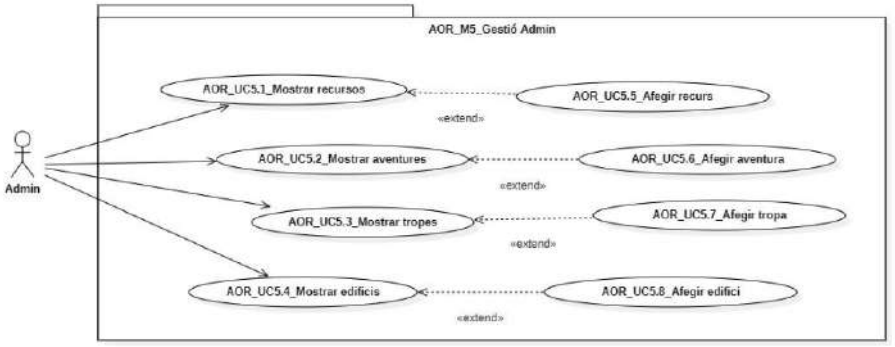

Figura 3.19: Diagrama de cas d'ús del mòdul de gestió administrador

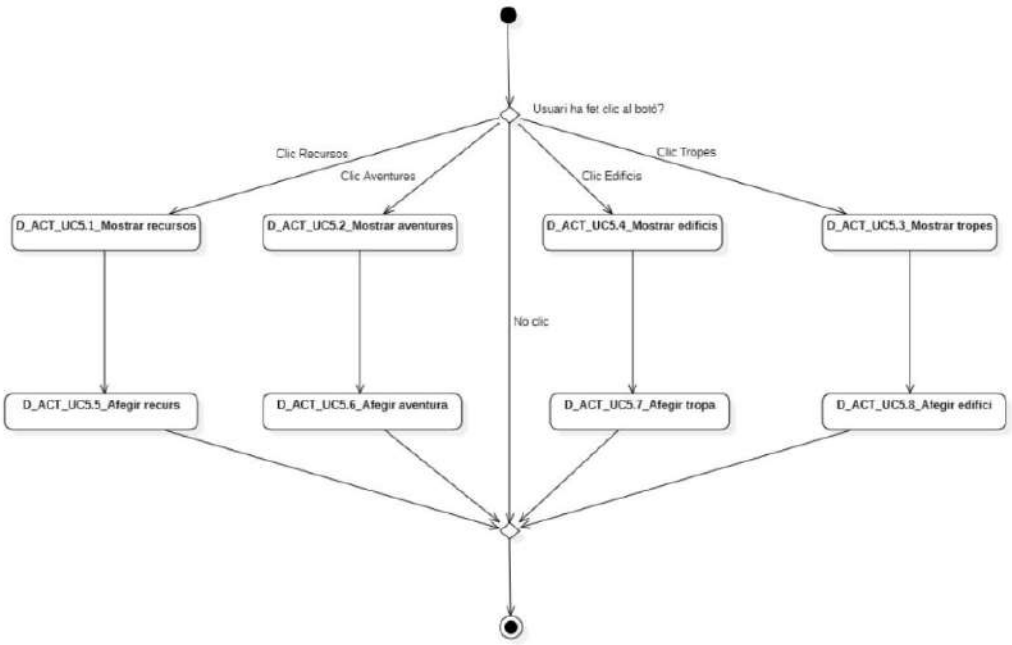

Figura 3.20: Diagrama d'activitats del mòdul de gestió administrador

3.3.2 Requisits no funcionals del producte

Els requeriments no funcionals del producte es defineixen pel rendiment, fiabilitat, disponibilitat, seguretat i manteniment.

3.3.2.1 Rendiment

L'usuari ha de poder navegar i executar totes les funcionalitats del videojoc sense notar cap parpelleig o *flickering* de les imatges a mesura que carreguen i cap retard a l'hora de mostrar la informació. Tot i que l'aplicació web i videojoc està enfocat a navegadors web, també s'ha de mostrar correctament en altres dispositius com tauletes o mòbils.

3.3.2.2 Fiabilitat

El *software* ha de funcionar correctament per totes les funcionalitats descrites en aquest ERS. En el cas que existís algun tipus d'error, el *software* ha de ser capaç de recuperar-se i gestionar-ho adequadament.

3.3.2.3 Disponibilitat

Durant el desenvolupament del *software* es faran diferents *checkpoints* abans de la publicació de la primera versió que serviran de punt de recuperació. A partir de la primera publicació, es publicaran noves versions del *software* a mesura que es modifiquin funcionalitats.

3.3.2.4 Seguretat

El *software* ha de ser capaç de suportar atacs comuns com la injecció. També s'ha de protegir tota la informació crítica de l'usuari com és la contrasenya. Finalment, es voldrà monitoritzar les interaccions de tots els usuaris a través d'algun sistema de *logs*.

3.3.2.5 Manteniment

El *software* es desenvoluparà amb l'objectiu que sigui fàcilment escalable i ben documentat en totes les parts. Així es podrà garantir un manteniment eficient en un futur per llançar futures versions.

A més, el *software* ha de poder ser escalable en el sentit que els administradors han de poder afegir dades noves en el sistema sense haver de fer canvis en el codi per part dels desenvolupadors.

Capítol 4

Model de dades

En aquest capítol s'ha desenvolupat el model de dades a partir del document ERS elaborat a la secció anterior.

S'ha dividit el model de dades en les següents fases:

- Modelat estàtic: En aquesta fase s'ha realitzat un diagrama de classes estàtic del sistema AOR. Per aquest diagrama de classes s'han descrit els tipus d'objectes del sistema AOR, així com els diversos tipus de relacions estàtiques entre aquests mateixos objectes. A més, s'observen els atributs de cada classe i les restriccions respectives entre els objectes que s'interconnecten.
- Modelat dinàmic: En aquesta fase s'han realitzat diferents diagrames de seqüència. Per als diagrames de seqüència s'ha representat el comportament del sistema AOR. Exactament, cada diagrama de seqüència correspon a un cas d'ús. En aquesta fase s'han efectuat un total de 9 diagrames de seqüència per als casos d'ús més representatius.

4.1 Diagrama de classes estàtic

Abans de passar a la fase d'implementació del *software* d'acord amb l'ERS ja definit, és important prendre un moment per decidir i establir un bon model de dades.

En primer lloc, es definiran les classes que formen el sistema, seguidament de les relacions entre aquestes classes i finalment es mostrarà un diagrama de classes.

La següent llista conté les classes principals del model amb els seus atributs més característics, sense tenir en compte les relacions entre elles.

- **Player**: Classe que descriu un jugador.
  - **Username**: Nom d'usuari del jugador.
  - **Password**: Contrasenya del jugador.
- **Land**: Classe que conté tota la informació de la terra.
  - **Name**: Nom de la terra.
- **LandType**: Tipus de terres diferents.
  - **Src**: Representa la direcció de la imatge de la plantilla utilitzada per representar visualment una terra en el joc.
  - **Resources**: Llista de recursos existents en la terra.
- **Event**: Esdeveniment de diferents accions d'una terra. Aquests esdeveniments poden produir millores, accions o construccions.
  - **Type**: Tipus d'esdeveniment.
  - **Time**: Temps programat per efectuar-se.
  - **LogsMessage**: Missatge per mostrar per pantalla.
- **Map**: Classe amb coordenades de terres.
  - **Coordinates**: Atribut que marca les coordenades d'un punt en el mapa.
- **Adventure**: Classe que guarda informació de les aventures en el sistema. Les aventures són batalles simulades jugador contra la màquina.
  - **Name**: Nom de la batalla.
  - **Difficulty**: Dificultat de guanyar l'aventura. Aquest atribut és un número superior a 0, que ajuda a l'usuari a identificar la dificultat de la batalla.
  - **Attack**: Atac total que té l'aventura. Aquest atribut és un número intern per efectuar els càlculs necessaris per identificar si l'usuari pot completar una aventura.
  - **Life**: Vida de l'aventura per completar-la. D'igual forma que l'atac d'una aventura, la vida és un número intern per efectuar els càlculs necessaris per identificar si es pot completar una aventura per part de l'usuari.
  - **Time**: Temps per completar l'aventura.

- **Resource**: Classe que guarda la quantitat d'un recurs.
  - **Quantity**: Quantitat del recurs.
- **ResouceData**: Classe que guarda informació dels recursos necessaris per totes les funcionalitats.
  - **Name**: Nom del recurs.
  - **Src**: Representa la direcció de la imatge de la plantilla utilitzada per representar visualment un recurs.
- **ResourceField**: Classe que guarda la informació d'una casella de recurs.
  - **NField**: Número de la casella del recurs.
  - **Level**: Nivell de la casella del recurs.
- **UpgradeResource**: Classe que guardà la informació de millora d'una casella de recurs.
  - **NewProduction**: Producció per segon d'una casella de recurs.
- **Upgrade**: Classe que guarda informació de les millores.
  - **Level**: Nivell de la millora.
  - **Time**: Temps necessari per completar la millora.
- **BuildingData**: Classe que guarda informació dels edificis que es poden construir en caselles de construcció.
  - **Name**: Nom del edifici.
  - **Description**: Descripció del edifici.
  - **Src**: Representa la direcció de la imatge de la plantilla utilitzada per representar visualment un edifici.
  - **Action**: Accions que es poden efectuar en l'edifici.
- **BuildingField**: Classe que guarda la informació d'una casella de construcció.
  - **NField**: Número de la casella de construcció.
  - **Level**: Nivell de la casella de construcció.
- **UpgradeBuilding**: Classe que guardà la informació de millora d'una casella d'edifici.
  - **NewAction**: Acció nova de la millora del edifici.

- **TroopData**: Classe que guarda informació de les tropes que es poden reclutar en el joc.
  - **Name**: Nom de la tropa.
  - **Src**: Representa la direcció de la imatge de la plantilla utilitzada per representar visualment una tropa.
  - **AttackDamage**: Atribut d'atac de la tropa.
  - **DefenderDamage**: Atribut de defensa de la tropa.
  - **Life**: Atribut de vida de la tropa.
  - **Mobility**: Atribut de mobilitat (caselles per segon) de la tropa.
  - **Level**: Nivell de la tropa d'aquestes característiques.
- **Troop**: Classe que guarda la quantitat d'una tropa.
  - **Quantity**: Nombre de tropes.
  - **Level**: Nivell de les tropes.
- **UpgradeTroop**: Classe que guardà la informació de millora d'una tropa.
  - **NewAttack**: Nou atac de la tropa aconseguit per la millora.
  - **NewDefense**: Nova defensa de la tropa aconseguida per la millora.
  - **NewLife**: Nova vida de la tropa aconseguida per la millora.
  - **NewMobility**: Nova mobilitat de la tropa aconseguida per la millora.
- **Pyramid**: Classe que guarda el nivell de les piràmides que es poden obtenir. La piràmide és l'edifici principal que bonifica en forma de recursos a la terra.
  - **Level**: Nivell de la piràmide.
- **UpgradePyramid**: Classe que guarda la informació de millora d'una piràmide.
  - **BonusProduction**: Bonus de millora en la producció de recursos.
  - **Src**: Representa la direcció de la imatge de la plantilla utilitzada per representar visualment una piràmide amb un nivell determinat.

Abans d'entrar a ensenyar el diagrama de classes, és important descriure correctament les relacions entre les classes. En la següent llista s'observen com estan relacionades les classes prèviament definides:

- Un **usuari** té una o més **terres**.
- Una **terra** pertany a un punt d'un **mapa**.
- Una **terra** té un **tipus de terra** predefinida.
- Les **terres predefinides** estan compostes per un conjunt de **caselles de recursos** fixats a nivell 1.
- Una **terra** té **recursos** de diferents tipus.
- Una **terra** té un conjunt de **caselles de recursos** que generen recursos per segon.
- Una **terra** té un conjunt de **caselles d'edificis**.
- Una **terra** té una **piràmide** amb un nivell determinat.
- Una **terra** pot tenir **tropes**. Les tropes es poden reclutar a través de les accions de certs edificis.
- Una **terra** pot realitzar **aventures**.
- Un **edifici** té **millores**. Una millora d'edifici puja de nivell la casella de l'edifici i produeix millores depenent de l'acció d'aquell edifici.
- Una **casella de recurs** té **millores**. Una millora puja de nivell la casella de recurs i millora la producció d'aquell recurs per segon.
- Una **piràmide** té **millores**. Aquestes millores canvien la imatge de la piràmide i dona una millor bonificació de producció.
- Per efectuar qualsevol **millora** o **construcció** a través d'una terra, s'executa un **esdeveniment**.
- Cada **esdeveniment** té associat una **acció** que s'efectuarà en un futur: millora d'edifici, millora de piràmide, construcció d'edifici, reclutament de tropa, etc.
- Els **esdeveniments** van associats a una **terra**.

Finalment, en la Figura 4.1 es mostra el diagrama de classes estàtic definit per les classes principals i les relacions entre elles prèviament definides.

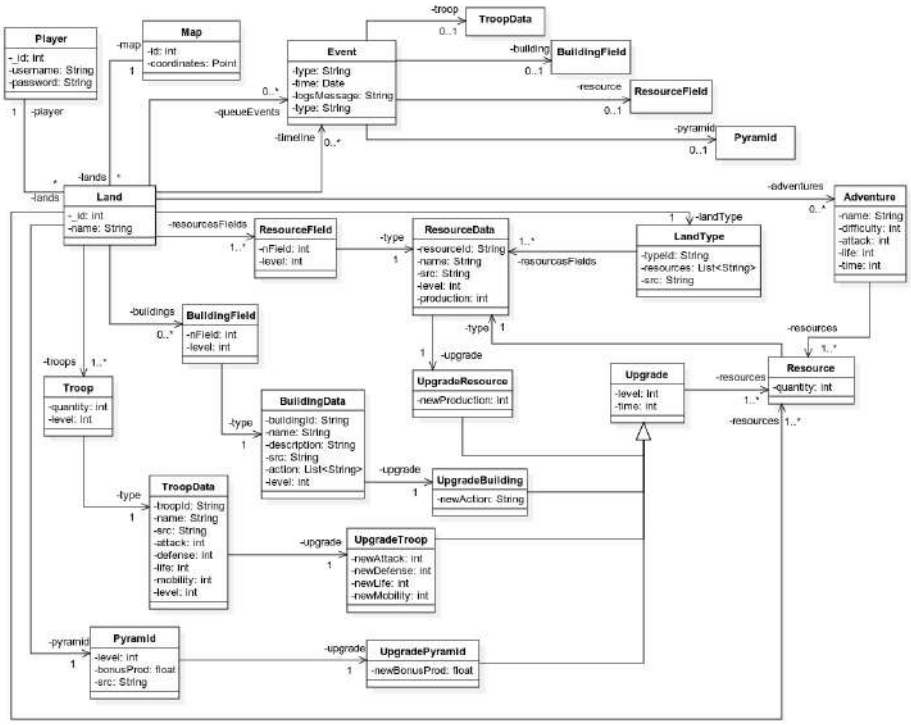

Figura 4.1: Diagrama de classes estàtic del sistema AOR

4.2 Diagrames de seqüència

Per aquesta secció cal destacar un seguit de diagrames de seqüència que faciliten al lector la comprensió del projecte, així com documentar i validar el modelatge de les dades. Els digrames de seqüència escollits són deguts a la importància de la funcionalitat que defineixen i per guiar a la futura implementació del sistema AOR.

- **D_SEC_UC1.1_Inici de sessió usuari**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.1. En aquest diagrama de seqüència es pot observar com un usuari executa un inici de sessió i interactua amb les classes més representatives de totes les capes.

- **D_SEC_UC1.2_Registre usuari**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.2. En aquest diagrama de seqüència es pot observar com es crea un usuari en el sistema AOR, i a més, com es crea per defecte una terra associada al nou usuari.

- **D_SEC_UC2.1_Canviar de terra**: Es pot observar el seu diagrama de

seqüència corresponent en la secció Annex a la Figura 11.3. En aquest diagrama es pot observar com el sistema intercanvia informació per tal de mostrar per pantalla la nova terra seleccionada.

- **D_SEC_UC2.2_Crear nova terra**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.4. En aquest diagrama de seqüència es poden observar totes les validacions que s'efectuen per tal de crear una nova terra, i en cas positiu, crear-la i associar-la a l'usuari que ha fet la petició.

- **D_SEC_UC3.1_Mostrar *timeline***: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.5. En aquest diagrama de seqüència es pot observar com es mostra per pantalla i s'aconsegueix la informació del seguit d'esdeveniments passats associats a una terra.

- **D_SEC_UC3.2_Mostrar rànquing**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.6. En aquest diagrama es pot observar com el sistema efectua peticions per aconseguir els usuaris amb puntuacions més altes (suma de recursos totals actuals) i mostrar-los per pantalla.

- **D_SEC_UC4.1_Fer aventura**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.7. En aquest diagrama es pot observar el transcurs d'efectuar una aventura, on primer es validarà que es pugui efectuar per després registrar aquest nou esdeveniment i enviar les tropes.

- **D_SEC_UC4.4_Millorar recurs**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.8. En aquest diagrama es pot observar com s'efectua una millora d'una casella de recurs, on primer es validarà que es pugui efectuar per després registrar un esdeveniment i restar els recursos consumits de la millora.

- **D_SEC_UC4.9_Construir edifici**: Es pot observar el seu diagrama de seqüència corresponent en la secció Annex a la Figura 11.9. En aquest diagrama es pot observar com s'efectua una construcció d'un edifici, on primer es validarà que es pugui efectuar per després registrar un esdeveniment i restar els recursos consumits.

Capítol 5

Disseny tecnològic

Un cop completats els capítols corresponents a l'ERS i el model de dades, aquest document es vol centrar en una tercera fase dedicada a desenvolupar el disseny tecnològic del sistema *Age of Ra*.

En aquest capítol es fa una proposta de les tecnologies a escollir de cara a la implementació de la nostra aplicació. També, es determina l'arquitectura física a través d'un diagrama de desplegament. Després, es dissenya la interfície d'usuari de l'aplicació a través d'un seguit de *wireframes* per cada pantalla. I finalment, millorant el diagrama de classes estàtic, es desenvolupa l'arquitectura lògica del sistema amb un diagrama de classes actualitzat i un diagrama de paquets per capes.

5.1 Arquitectura del sistema

L'arquitectura del sistema *Age of Ra* es basa en una arquitectura web *Single-Page Application* (SPA), amb dues parts clarament desacoblades: el *frontend* i el *backend*.

Una SPA és un tipus d'aplicació web que carrega una sola pàgina inicial en el navegador del client i actualitza dinàmicament el contingut d'aquesta pàgina a mesura que els usuaris interactuen amb l'aplicació, sense necessitat de recarregar tota la pàgina. Això proporciona una experiència de navegació fluida i àgil.

El *frontend* es tracta d'una SPA que es carrega en el navegador web del client. Els usuaris realitzen peticions a través d'aquesta interfície cap al servidor per obtenir dades o realitzar comandes. Aquesta part es responsabilitza de gestionar les interaccions de l'usuari i mostrar els continguts de manera dinàmica.

D'altra banda, el *backend* és la part del servidor que gestiona les peticions dels clients i proporciona les dades i els serveis necessaris. Quan un client fa una petició al servidor, aquest processa la sol·licitud, accedeix a la base de dades per obtenir o actualitzar la informació rellevant del joc i retorna la resposta al client. El *backend* és el motor del sistema *Age of Ra* i s'encarrega de la lògica de negoci i del processament de les operacions.

A més del *frontend* i del *backend*, el sistema inclou una base de dades per emmagatzemar la informació rellevant del joc, com ara les dades dels usuaris, els estats dels recursos i l'estat general del joc. Això permet mantenir una persistència de la informació i assegurar una correcta gestió de l'estat del joc.

En resum, l'arquitectura del sistema *Age of Ra* es compon del *frontend* com a interfície d'usuari, el *backend* com a part del servidor que gestiona les peticions i la base de dades que emmagatzema les dades del joc. Es pot observar un diagrama d'aquesta arquitectura a la Figura 5.1.

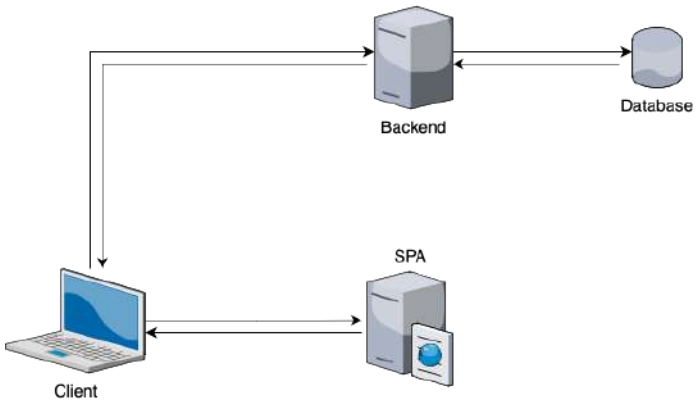

Figura 5.1: Diagrama no formal de l'arquitectura del sistema

5.2 Comunicació

Al principi del document es plantejava com a objectiu fer un disseny escalable tal que l'aplicació hagi de respondre de forma lineal a mesura que augmenten els usuaris efectuant peticions. Per tant, es creu important discutir com s'han d'efectuar les accions en temps real del sistema abans de prendre les decisions de disseny tant del *backend* com del *frontend*.

Per a una correcta comunicació entre el *frontend* i el *backend*, s'ha de decidir si efectuar el càlcul de les accions en temps real per a tots els usuaris o bé de forma diferida. Un càlcul de les accions en temps real implica que quan s'efectuen

actualitzacions del videojoc, els clients han de rebre les actualitzacions a l'instant, és a dir, han d'estar sincronitzats amb el servidor. Un càlcul diferit es defineix per no efectuar les actualitzacions fins a un cert interval de temps, per exemple actualitzar cada hora, o fins que un client ho requereixi.

La primera solució que es planteja és la més senzilla i trivial d'efectuar: programar un *daemon* o programa que s'executa en segon pla del servidor per actualitzar tots els esdeveniments. En aquest cas, el *backend* seria una font única de veritat i sempre estaria actualitzat. Aquesta solució és eficaç, però comporta una gran càrrega de treball ineficient quan no existeixen usuaris actius en el sistema. El sistema hauria d'estar actiu les 24 hores del dia per actualitzar tots els esdeveniments dels usuaris. Per tant, aquest fet és el principal motiu que ha portat a descartar aquesta solució, ja que s'està dissenyant un videojoc d'accions esporàdiques i molt temps mort.

La segona solució que es planteja és la d'establir una relació bidireccional entre client i servidor, així el client està sempre actualitzat de tots els canvis que es produïssin al sistema. Aquesta comunicació permet al client rebre immediatament les actualitzacions gràcies a la relació bidireccional, però es requereix que el servidor estigui permanentment actualitzat i enviant dades al client. D'igual forma que la primera solució, comporta una gran càrrega de treball ineficient quan el client no està efectuant peticions al servidor. També, com pot existir una comunicació entre usuaris, els usuaris no connectats del sistema haurien d'estar actualitzats en tot moment. Per tant, també s'ha descartat aquesta solució.

Com a tercera proposta, s'ha plantejat un sistema que no tingués cap programa en paral·lel per actualitzar valors ni esdeveniments. Es basa a actualitzar les accions i camps necessaris només quan el servidor rep una petició. Tot i això, per l'usuari connectat és important que rebi una percepció fluida del videojoc, per tant, ha de ser el *frontend* l'encarregat de simular les accions en temps real. Amb aquesta proposta es millora l'escalabilitat de la primera i de la segona, en termes d'usuaris, ja que mentre el servidor no rebi peticions, no s'efectuaran càlculs.

D'acord amb aquestes tres propostes, s'ha acabat decidint per la tercera, ja que gràcies al càlcul diferit en les actualitzacions del sistema, permet una millora d'escalabilitat en termes de jugadors totals del servidor. Així, en els següents apartats es farà una tria de la tecnologia a implementar d'acord amb el càlcul diferit de les accions en temps real i la simulació d'aquests per part del *frontend*.

5.3 Decisions tecnològiques

En aquest apartat, es discutiran les decisions realitzades per a les tecnologies utilitzades en el *frontend*, *backend* i la persistència de dades del sistema AOR.

5.3.1 *Frontend*

En el moment de desenvolupar el projecte, les tecnologies estàndard pel desenvolupament web són HTML per determinar l'estructura de la pàgina mitjançant etiquetes, CSS per definir l'estil i JavaScript per gestionar la interacció amb l'usuari.

En el mercat actual existeixen diferents *frameworks* de JavaScript que ajuden a l'usuari a crear una pàgina web interactiva de forma senzilla i modular. En aquest treball s'ha decidit desenvolupar el codi del *frontend* amb un *framework* de JavaScript popular i amb tendència positiva de creixement d'ús en els últims anys.

Per comparar les diferents opcions de *framework* per JavaScript, s'ha consultat la web "*State of JavaScript*" [9]. Aquesta pàgina ofereix una recopilació d'enquestes anuals a desenvolupadors web de diferents àmbits, amb l'objectiu tendències actuals i futures de diferents *frameworks* de JavaScript.

D'acord amb la Figura 5.2, els *frameworks* més utilitzats actualment són React, Angular i Vue.js. A més, en la Figura 5.3 es pot concloure que dins dels tres *frameworks* més usats, Vue.js i React són els que tenen més interès.

Figura 5.2: *Frameworks* més utilitzats de JavaScript pel *frontend* segons *State of JavaScript*

Figura 5.3: *Frameworks* amb més interès per part de la comunitat de JavaScript pel *frontend* segons *State of JavaScript*

Per tant, dins de les estadístiques observades i buscant un *framework* popular i amb interès de la comunitat, però alhora suficientment madur, s'ha decidit implementar el *frontend* amb Vue.js.

5.3.2 *Backend*

La decisió de la tecnologia per implementar el *backend* s'ha pres d'acord amb el discutit a la secció anterior sobre la comunicació i l'actualització de les accions amb un càlcul diferit.

Per complementar la decisió, cal explicar que existeixen dos mecanismes de comunicació disponibles del *backend* amb el *frontend*: API REST o WebSockets.

De forma senzilla, una API REST és un estàndard de disseny d'interfícies per a la comunicació entre dos sistemes de computació a través d'Internet, que posa èmfasi en l'ús d'una arquitectura basada en el protocol HTTP. Aquesta interfície permet intercanviar informació de manera segura i eficient mitjançant diferents tipus de peticions, com ara GET, POST, PUT i DELETE [10]. És important destacar que aquestes peticions són sempre iniciades per part del client.

Alguns avantatges de l'API REST són l'escalabilitat, ja que s'optimitza les interaccions entre el client i servidor, la flexibilitat, perquè admeten una divisió entre el client i el servidor, i independència, a causa del fet que no importa la

tecnologia que es fa servir per interactuar.

També, una API REST està enfocada a donar una comunicació *stateless*. Això vol dir que és una comunicació que no demana un continuat tràfic d'intercanvi de dades.

Es pot observar un exemple d'aquesta comunicació a la Figura 5.4.

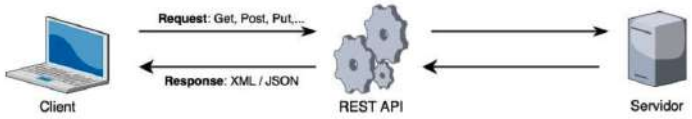

Figura 5.4: Comunicació utilitzant una API REST

En canvi, els WebSockets és una tecnologia que fa possible obrir un canal bidireccional de comunicació entre el client i el servidor.

Un tret característic dels WebSockets és que el servidor pot enviar missatges al client de forma proactiva [11]. Els WebSockets estan orientats a una comunicació *stateful*, és a dir, una comunicació que demana un continu ininterromput tràfic de dades.

Es pot observar un exemple d'aquesta comunicació a la Figura 5.5.

Figura 5.5: Comunicació utilitzant WebSockets

D'acord amb aquestes explicacions i la decisió d'efectuar les actualitzacions del sistema amb un càlcul en diferit, s'ha decidit utilitzar una API REST per resoldre la totalitat del *backend*.

Un cop decidida l'arquitectura del sistema *Age of Ra*, cal prendre una decisió de com desenvolupar aquest *backend*. D'una forma més conservadora pel *backend* respecte al *frontend*, s'ha decidit utilitzar Express, el *framework* més utilitzat de JavaScript pel *backend*. Es pot apreciar a la Figura 5.6 com la tendència d'ús d'Express continua sent de les més altes amb el transcurs dels anys.

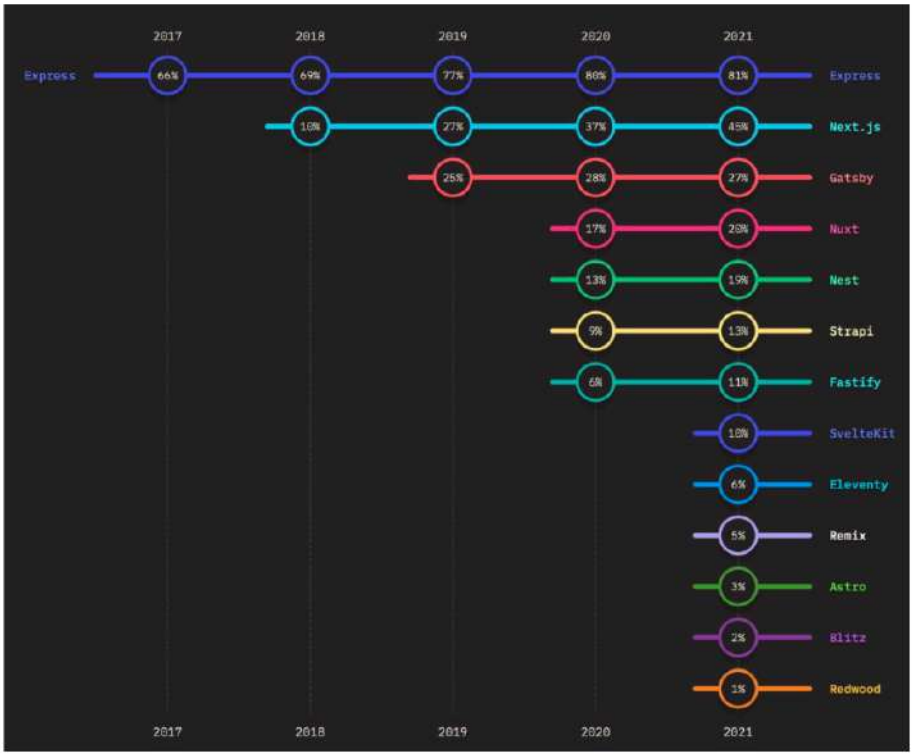

Figura 5.6: *Frameworks* més utilitzats de JavaScript pel *backend* segons *State of JavaScript*

5.3.3 Persistència de dades

Un cop establerta l'arquitectura del *frontend* i del *backend*, falta parlar sobre les decisions de disseny de la persistència de dades. En aquest apartat es fa una anàlisi tecnològica sobre les bases de dades relacionals i de les no relacionals, això com la decisió final pel sistema *Age of Ra*.

Principalment, les bases de dades relacionals SQL emmagatzemen les dades de forma estructurada, mentre que les bases de dades no relacionals NoSQL ho fan en el seu format original. També, en les bases de dades SQL proporcionen una baixa escalabilitat a canvi de robustesa en els seus esquemes, en comparació a les NoSQL, que ofereixen més flexibilitat.

Atesos als objectius d'aquest treball, s'ha escollit les bases de dades NoSQL, fet que ens oferirà ampliar el catàleg de funcionalitats i característiques del joc sense tenir tantes dependències entre taules i relacions. A més, s'ha escollit utilitzar MongoDB com a bases de dades no relacionals en ser una base de dades orientada a documents i facilitar una bona documentació per utilitzar-la.

5.4 Diagrama de desplegament

En aquesta secció, s'observa en la Figura 5.7 el diagrama de desplegament del sistema AOR, que es construeix utilitzant tres blocs principals: AOR_Frontend, AOR_Backend i DatabaseServer. Com es pot veure, aquest diagrama implementa en diferents blocs les tecnologies justificades de la secció anterior.

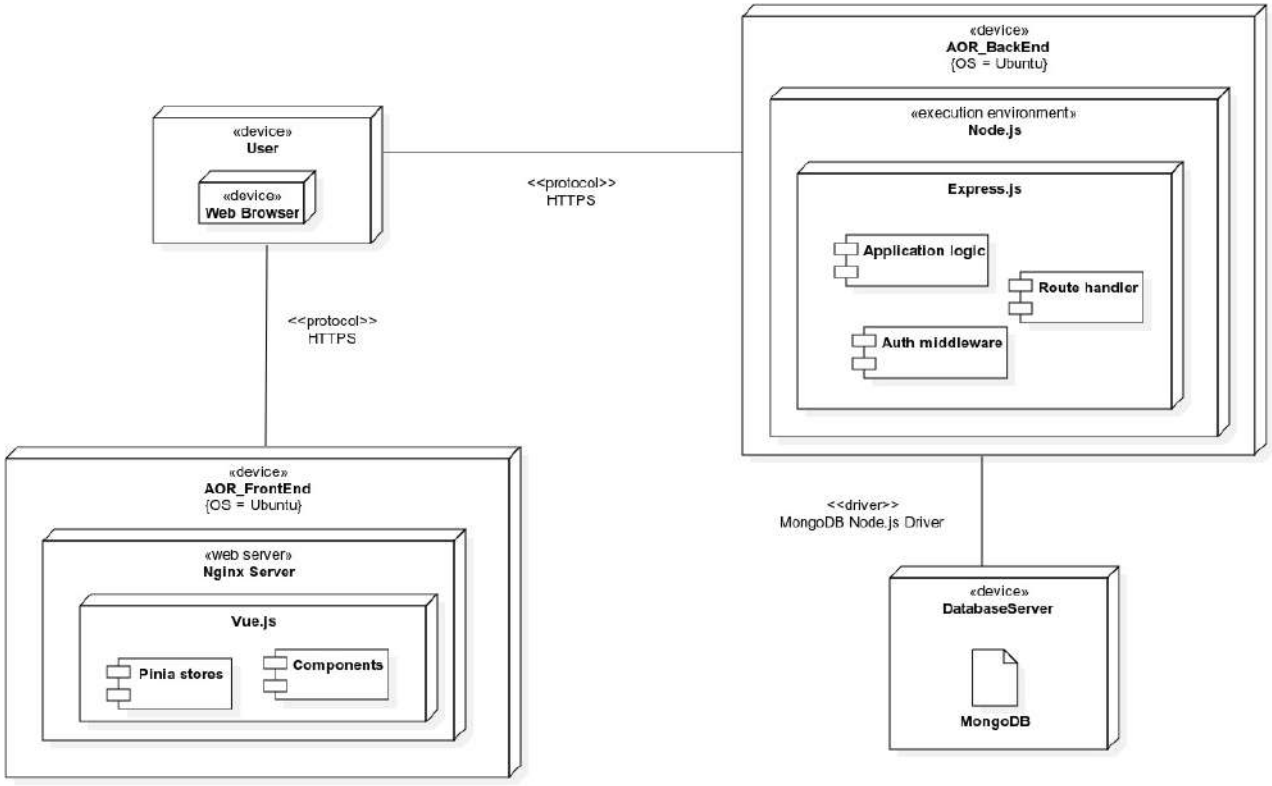

Figura 5.7: Diagrama de desplegament del sistema AOR

5.5 Interfície d'usuari

En aquesta secció es mostren els diferents *wireframes* de la interfície de l'usuari per interactuar amb l'aplicació a través d'un navegador web. Aquests *wireframes* s'han dissenyat acuradament per garantir una experiència d'usuari fluida i intuïtiva, tenint en compte el "*look and feel*" de l'aplicació.

En la Figura 5.8 s'ha dissenyat la pantalla principal amb la qual qualsevol usuari ha d'interaccionar, tant per iniciar sessió o per registrar-se.

Figura 5.8: *Wireframe* de la pantalla d'inici

En la Figura 5.9 s'ha dissenyat la pantalla de registre d'un usuari, on només es demana un nom d'usuari, una contrasenya i la confirmació d'aquesta.

Figura 5.9: *Wireframe* de la pantalla de registre d'usuari

En la Figura 5.10 s'ha dissenyat la pantalla d'inici de sessió amb els camps usuari i contrasenya.

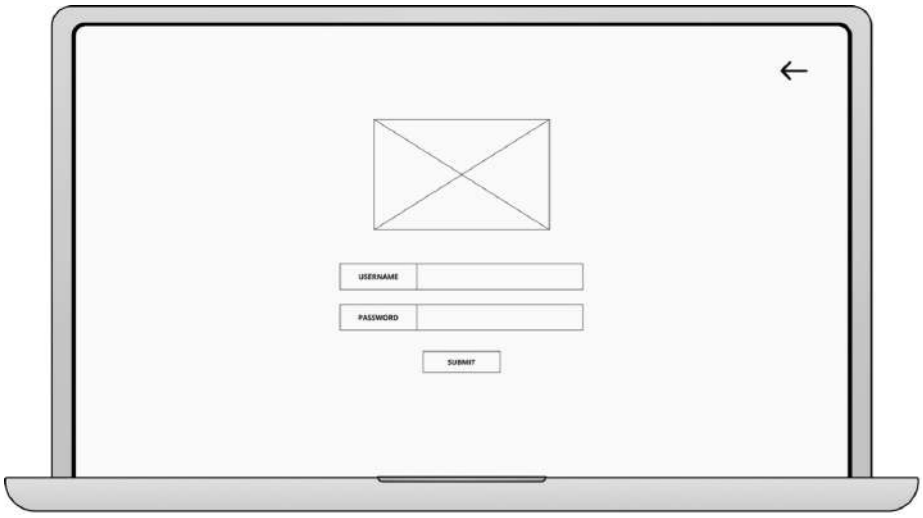

Figura 5.10: *Wireframe* de la pantalla d'inici de sessió

En la Figura 5.11 s'ha dissenyat la pantalla principal del joc, on es caracteritza per la consulta de caselles de recursos al bloc del mig. Aquesta pantalla és la pantalla per defecte després d'haver iniciat sessió. Es pot observar com en

aquesta pantalla es separa diferents accions o informació en blocs. Entre els més destacats, estan el bloc de consultar recursos, consultar tropes, canviar de terra, consultar cues, efectuar aventures, entre d'altres.

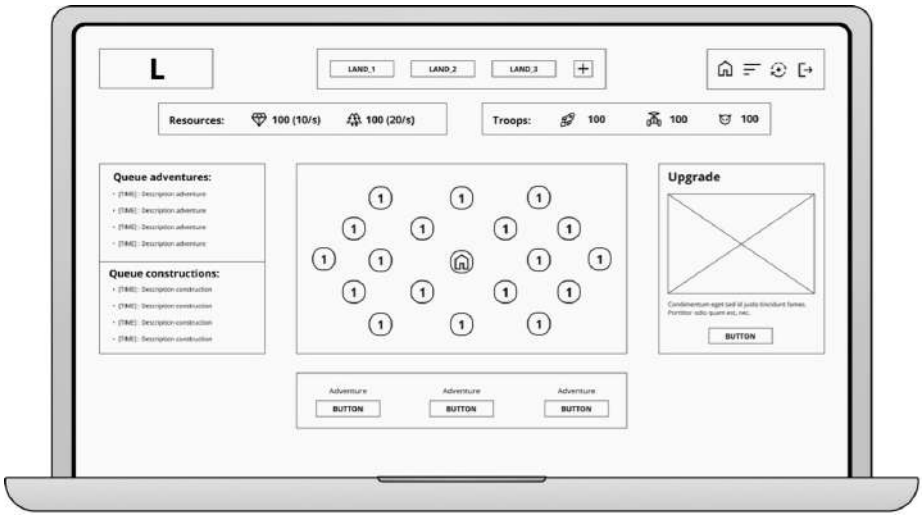

Figura 5.11: *Wireframe* de la pantalla de joc exterior

En la Figura 5.12 es pot observar el disseny del joc, enfocat a edificis. És el mateix disseny que l'anterior, però al bloc del mig es mostraran tots els edificis corresponents a la terra actual.

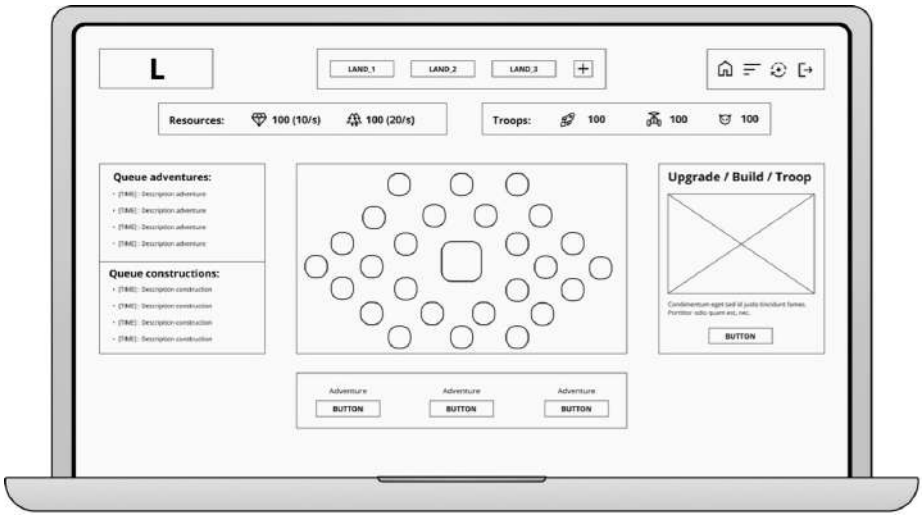

Figura 5.12: *Wireframe* de la pantalla de joc interior

En la Figura 5.13 s'ha dissenyat la pantalla de quan es consulta la *timeline* de les accions d'una terra.

Figura 5.13: *Wireframe* de la pantalla de *timeline*

En la Figura 5.14 s'ha dissenyat la pantalla de quan es consulta el rànquing de jugadors del sistema.

Figura 5.14: *Wireframe* de la pantalla del rànquing

En la Figura 5.15 s'ha dissenyat la pantalla de quan es demana la confirmació a l'usuari per crear una nova terra.

Figura 5.15: *Wireframe* de la pantalla per crear una terra

En la Figura 5.16 s'ha dissenyat la pantalla de quan es demana la confirmació a l'usuari per tancar la sessió.

Figura 5.16: *Wireframe* de la pantalla per tancar la sessió

En la Figura 5.17 s'ha dissenyat la pantalla principal d'un usuari del tipus administrador. En aquesta pantalla es presenten les diferents accions que aquest usuari pot realitzar.

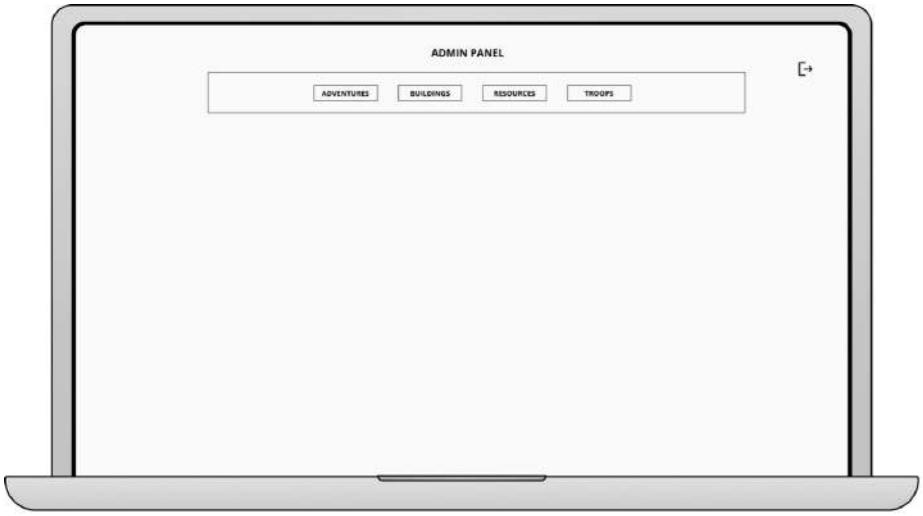

Figura 5.17: *Wireframe* de la pantalla principal dels administradors

En la Figura 5.18 s'ha dissenyat la pantalla principal d'un usuari del tipus administrador quan està consultant les aventures. En aquesta pantalla es podrà consultar les aventures actuals o crear de noves.

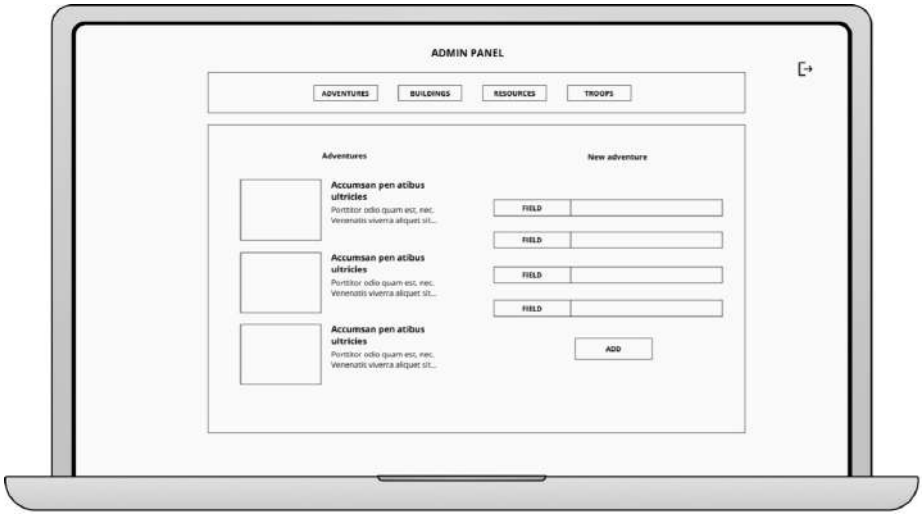

Figura 5.18: *Wireframe* de la pantalla per editar aventures dels administradors

En la Figura 5.19 s'ha dissenyat la pantalla principal d'un usuari del tipus administrador quan està consultant els edificis. En aquesta pantalla es podrà consultar els edificis o crear de nous.

Figura 5.19: *Wireframe* de la pantalla per editar edificis dels administradors

En la Figura 5.20 s'ha dissenyat la pantalla principal d'un usuari del tipus administrador quan està consultant els recursos. En aquesta pantalla es podrà consultar els recursos o crear de nous.

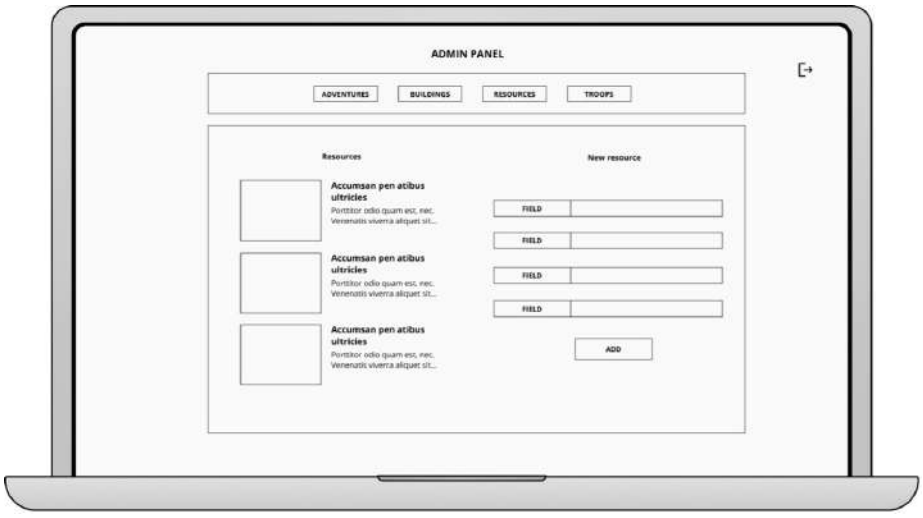

Figura 5.20: *Wireframe* de la pantalla per editar recursos dels administradors

En la Figura 5.21 s'ha dissenyat la pantalla principal d'un usuari del tipus administrador quan està consultant les tropes. En aquesta pantalla es podrà consultar les tropes o crear de noves.

Figura 5.21: *Wireframe* de la pantalla per editar tropes dels administradors

5.6 Diagrama de classes

En aquesta secció s'introdueix el diagrama de classes actualitzat (Figura 5.22) respecte de la primera versió. En aquesta actualització es pot observar com s'han afegit al sistema les classes, relacions i atributs, fruit dels diagrames de seqüència realitzats a posterior. Tot i que aquest diagrama es centra en les relacions entre les diferents classes i atributs, s'han afegit les capes a les quals pertanyen a cada classe per relacionar el diagrama de classes amb el diagrama de paquets.

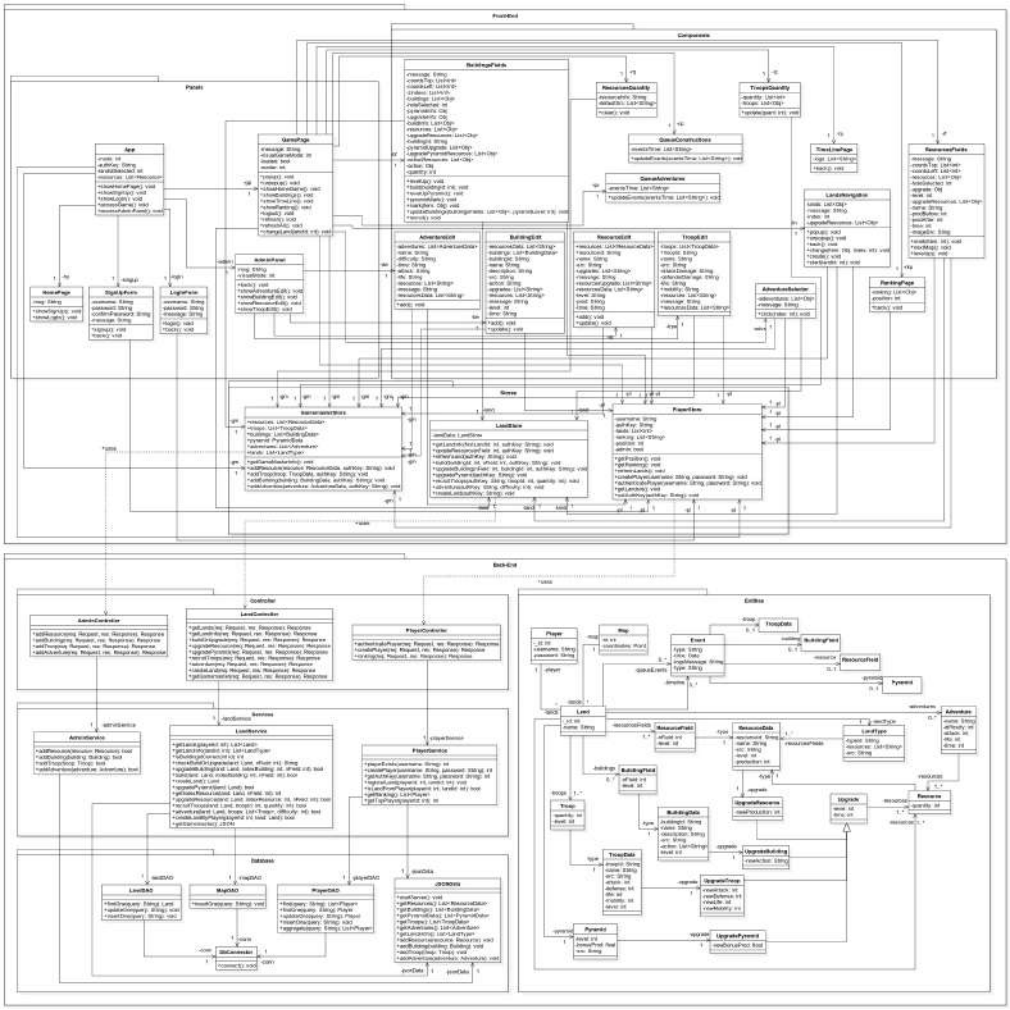

Figura 5.22: Diagrama de classes del sistema AOR

5.7 Diagrama de paquets

En aquesta secció s'introdueix el diagrama de paquets (Figura 5.23), que és un resum del diagrama de classes sense tenir en compte les relacions i els atributs. Així, es pot observar com el sistema AOR s'ha dividit en dues grans capes: *frontend* i *backend*. La capa de *frontend* conté la capa de *Panels*, la capa de *Components* i les *Stores*. En canvi, la capa del *backend* conté les capes de *Controller*, *Services*, *Database* i *Entities*.

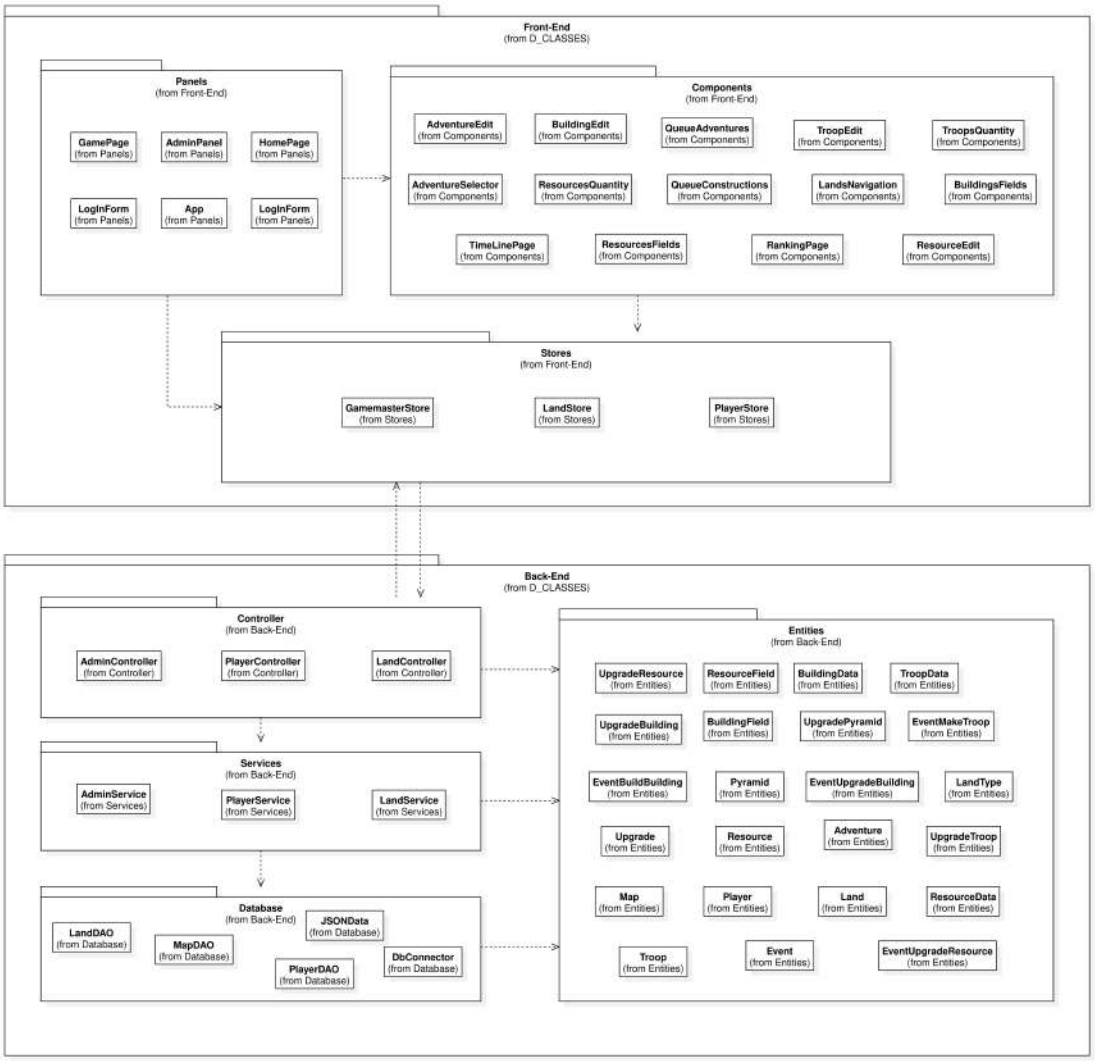

Figura 5.23: Diagrama de paquets del sistema AOR

Capítol 6

Implementació

En aquest capítol, es descriuen els detalls de més importància de la implementació tècnica de l'arquitectura proposada en els capítols anteriors i de la implementació del disseny final de l'aplicació.

6.1 Implementació tecnològica

Durant el transcurs del desenvolupament del codi, hi ha hagut certs detalls tècnics que és important destacar. En aquest apartat es vol fer èmfasi en l'arquitectura API REST implementada, com s'ha gestionat el càlcul en temps real dels esdeveniments, la persistència de les regles del videojoc, els components de Vue.js i el concepte de disseny d'una única font de veritat.

6.1.1 Arquitectura de l'API REST

Per desenvolupar el codi que gestiona l'API REST del *backend*, s'ha efectuat mitjançant una arquitectura en capes.

Una arquitectura en capes és un patró habitual de disseny de codi que organitza els components d'un sistema en capes diferents. Cada capa té una responsabilitat en específic i es pot comunicar amb les capes superiors o inferiors. L'objectiu d'aplicar aquesta arquitectura és dividir la complexitat d'un sistema en peces més petites i més fàcils de mantenir en el temps de forma independent.

Més en concret, s'ha aplicat el patró de disseny Controller-Service-Repository per gestionar les peticions de l'API REST. Aquest patró de disseny és característic per dividir el codi en tres capes:

- **Controller**: Gestiona les peticions i respostes de l'API REST.

- **Service**: Gestiona la lògica del model.
- **Repository**: Accedeix i manipula les dades emmagatzemades en una base de dades o un altre emmagatzematge persistent.

Es pot observar un exemple d'aquest patró de disseny a la Figura 6.1.

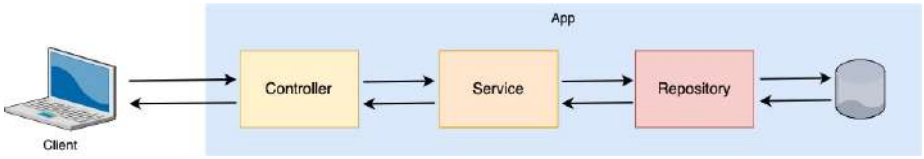

Figura 6.1: Arquitectura en capes Controller-Service-Repository

6.1.2 Càlcul en temps real dels esdeveniments

Anteriorment, s'ha decidit que per permetre una millor escalabilitat respecte als jugadors totals del servidor s'utilitzarà una estratègia d'actualització dels esdeveniments basada en un càlcul en diferit. En aquest apartat, s'explica aquesta implementació i el procés que s'ha seguit per arribar al resultat final.

És important recordar que quan es parla d'esdeveniments, es fa referència a totes aquelles accions que un usuari pot efectuar a una terra del servidor i que no tenen un efecte immediat. Aquests esdeveniments s'efectuen a partir d'un retard determinat a l'hora que l'usuari el posa en cua. A més, cal sumar la generació passiva de recursos, que s'efectua mitjançant la suma de la producció de cada una de les caselles de recursos de cada terra. Sumant tots aquests factors, és clar que fa falta modelar una solució adient i adequada per fer un càlcul en diferit dels esdeveniments.

A la primera versió i presentació d'aquest document, els esdeveniments presents són:

- Construir un edifici
- Millorar un edifici
- Millorar una casella de recurs
- Efectuar una aventura
- Millorar la piràmide
- Generació de recursos per segon (passiu)

Per la part del *frontend*, l'estratègia que s'ha seguit és només recalcular els esdeveniments quan es produeixi una recàrrega de la pàgina o quan hagi passat suficient estona perquè un esdeveniment de la cua s'hagi completat. En l'últim

cas, s'efectuarà una petita recàrrega del component Vue.js afectat i no comportarà una recàrrega total de la pàgina. Això també inclou la generació de recursos per segon, la qual el *frontend* és l'encarregat de calcular la producció de recursos, a través de la informació que obté de la terra, i simular a temps real sumant al comptador que es mostra en l'usuari.

Efectuant aquesta estratègia, no es sobrecarrega el servidor de peticions i presenta una fluïdesa a l'usuari, ja que de cara a l'espectador sembla com si tot s'estigués calculant cada segon. Tot i això, és important destacar que tot és una simulació per part del *frontend*, i sempre que es necessiti efectuar alguna acció, s'haurà d'involucrar el servidor. A més, el servidor està dissenyat per mai agafar dades del *frontend*, així evitar possibles asincronismes de recursos (*frontend* – *backend*) o manipulació de les dades per part dels usuaris que intenten realitzar accions fraudulentes.

Per la part del *backend*, s'han de crear un seguit de regles de càlcul en diferit d'esdeveniments i guardar un historial d'esdeveniments associat a cada terra, així com l'última petició al servidor per part del client.

En resum, l'ordre de la lògica que segueix el servidor és:

1. Actualitzar recursos de la terra respecta l'última petició.
2. Actualitzar esdeveniments que hagin passat des de l'última petició.
3. Efectuar petició del client segons dades actualitzades del servidor.
4. Respondre client de la petició.
5. Refrescar client amb nous recursos actualitzats i possibles esdeveniments passats.

És important destacar que aquest procés de 5 passos seria senzill si els esdeveniments i la producció de recursos fossin càlculs excloents. A la versió del videojoc presentada en aquest document existeixen diferents interseccions en la producció de recursos amb millores de caselles de recursos, piràmide i/o realització d'aventures.

Però, abans d'entrar en més detall, cal recordar quines accions tenen els elements descrits anteriorment sobre la producció de recursos:

- Cada casella de recursos genera una producció d'un tipus de recurs per segon. Una casella de recurs té diferents nivells amb produccions diferents. Per calcular la producció total per segon d'un recurs cal fer la suma de totes les produccions de les caselles de recursos del mateix tipus.
- La piràmide de la terra bonifica en percentatge a la producció per segon de tots els tipus de recursos. La piràmide té diferents nivells que fan variar el percentatge de la bonificació.

- Una aventura, si és completada amb èxit, suma una quantitat definida de recursos.

Per veure les interseccions comentades, a continuació s'analitzen tots els casos de càlcul d'esdeveniments, començant pels més trivials fins als més complicats.

El primer cas a investigar és el més senzill. En aquest cas l'usuari efectua una petició en un instant determinat "*New Connection*". Durant "*NW: New Connection*" i "*LT: Last Connection*" no hi ha hagut esdeveniments. Per tant, tal com es mostra en la Figura 6.2, només cal sumar als recursos totals, els recursos que s'han produït entre LT i NW.

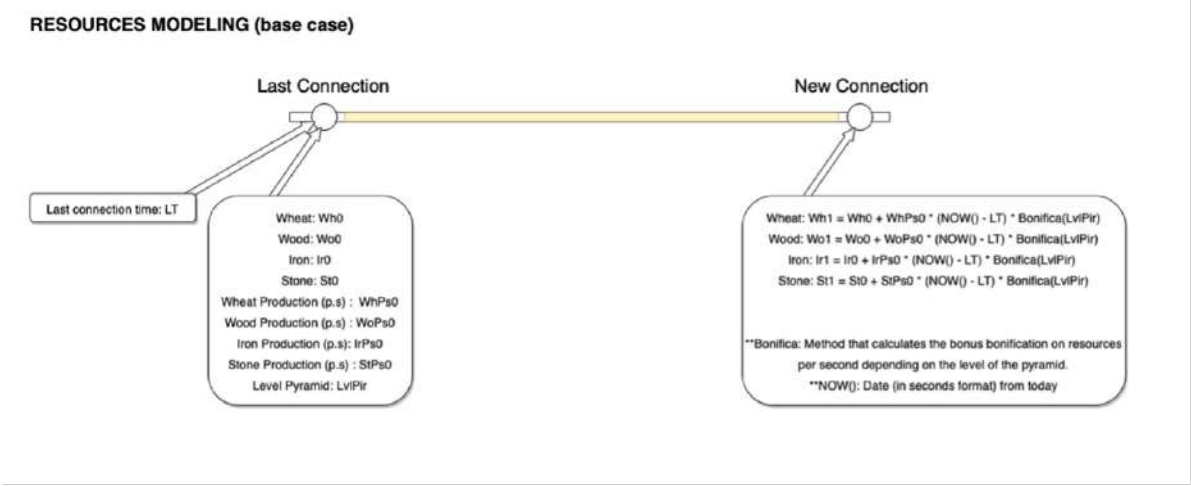

Figura 6.2: Simulació del cas base de producció de recursos

En aquest cas és semblant a l'anterior, però es completa una millora d'una casella de recurs tal com es mostra a la Figura 6.3 entre l'última connexió i la nova connexió. Això vol dir que existeixen dos segments de temps, un segment que s'estaven produint uns recursos determinats i un altre segment que es produeix una quantitat major. La forma correcta de calcular els recursos actualitzats a la nova connexió del client és calcular en el primer segment la quantitat de recursos produïts, la quantitat de recursos produïts en el segon segment, i sumar-ho als recursos inicials.

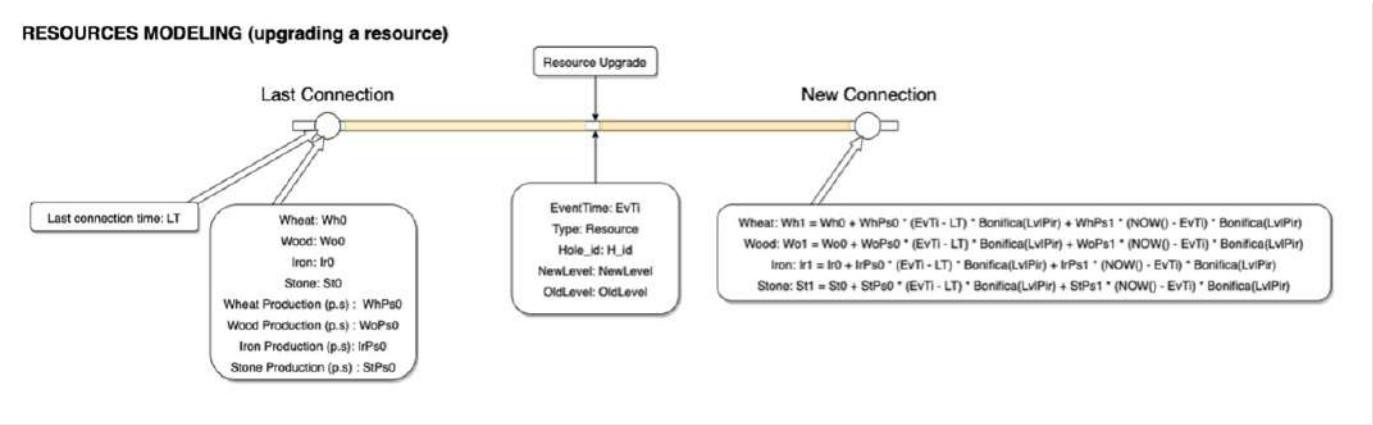

Figura 6.3: Simulació del cas del càlcul de recursos finals amb una millora de recurs

Per tant, generalitzant el cas anterior, es pot obtenir la quantitat actualitzada d'un recurs tal com es mostra en la Figura 6.4.

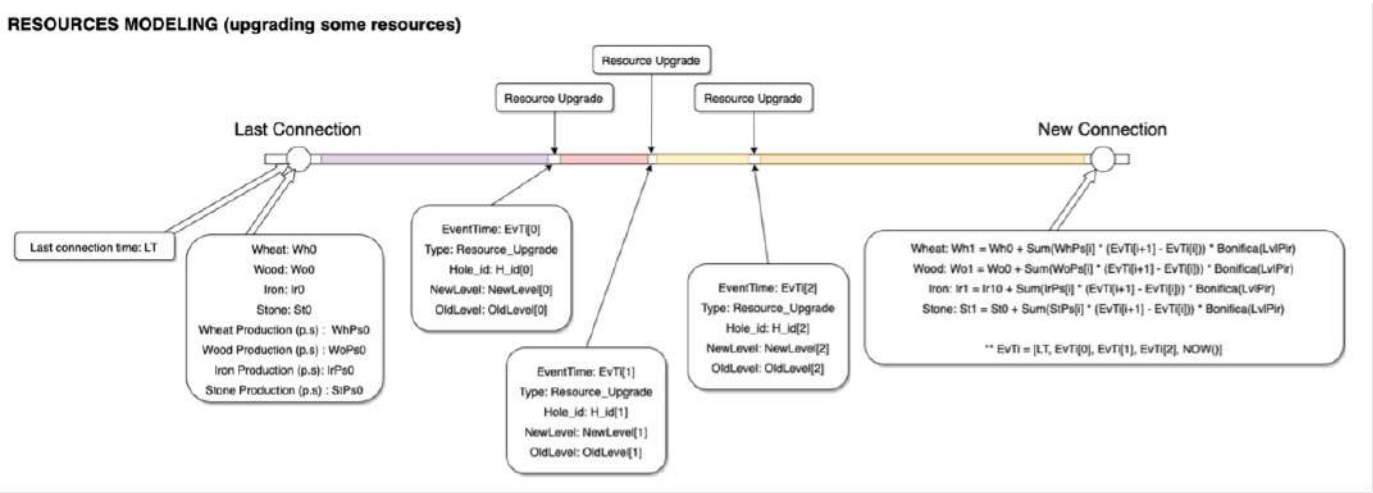

Figura 6.4: Simulació del cas del càlcul de recursos finals amb tres millores de recursos

D'igual forma, pot existir una millora completada de la piràmide entra l'última connexió i la nova connexió. Per calcular els recursos totals actualitzats, s'han de calcular primer els recursos del primer segment amb un percentatge, i després calcular els recursos del segon segment en un altre percentatge. Es pot observar en l'exemple de la Figura 6.5.

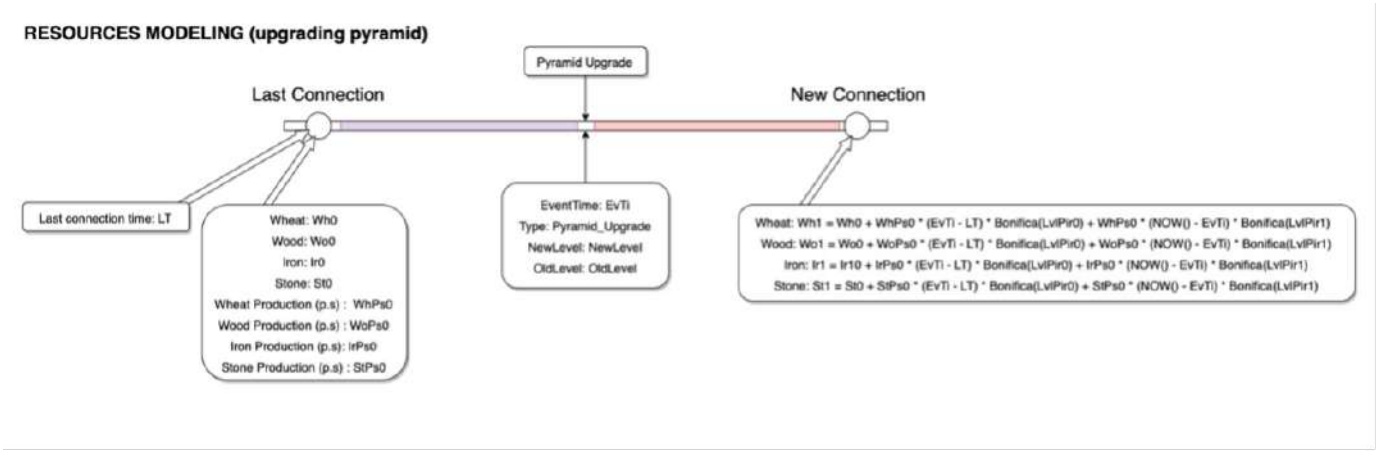

Figura 6.5: Simulació del cas del càlcul de recursos finals amb una millora de piràmide

Combinant els casos vistos anteriorment, es pot donar el cas on es completin esdeveniments, tant de millores de caselles de recursos com de piràmide. Per tant, es planteja una solució que combina a les dues ja plantejades. Aquest cas es mostra a la Figura 6.6.

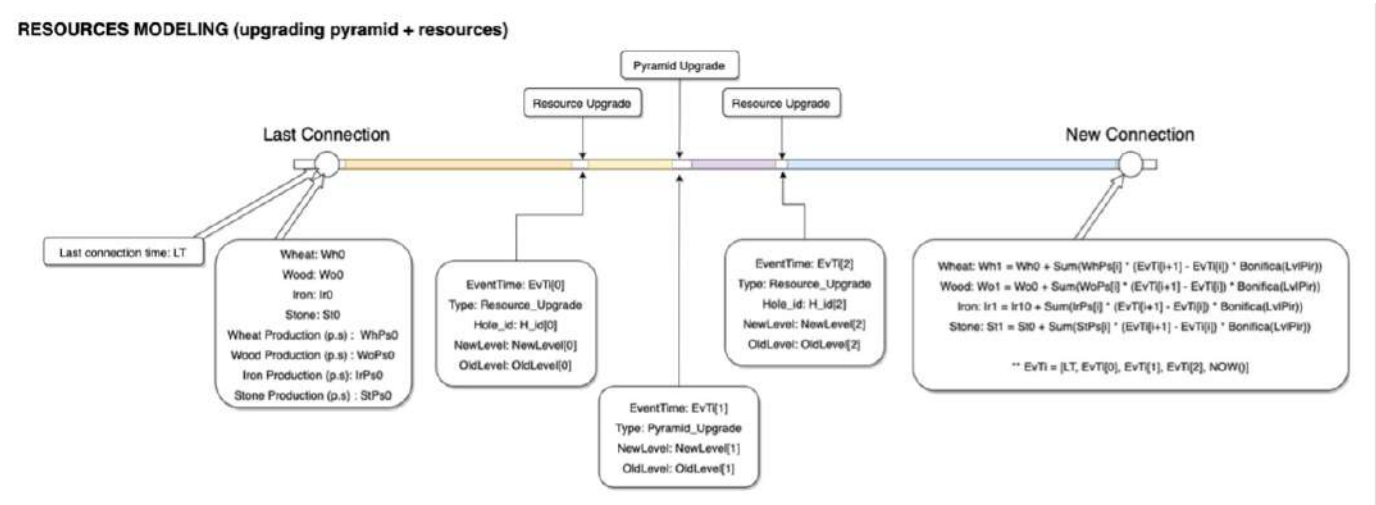

Figura 6.6: Simulació del cas del càlcul de recursos finals amb millores de recursos i piràmide

Arribant al final, pot existir un cas on s'hagi enviat una aventura i aquesta es

completi al mig de les dues connexions. En aquest cas és senzill calcular els recursos actualitzats, ja que és com el primer cas tot sumant els recursos que s'ha obtingut de l'aventura. A la Figura 6.7 es pot observar aquest cas.

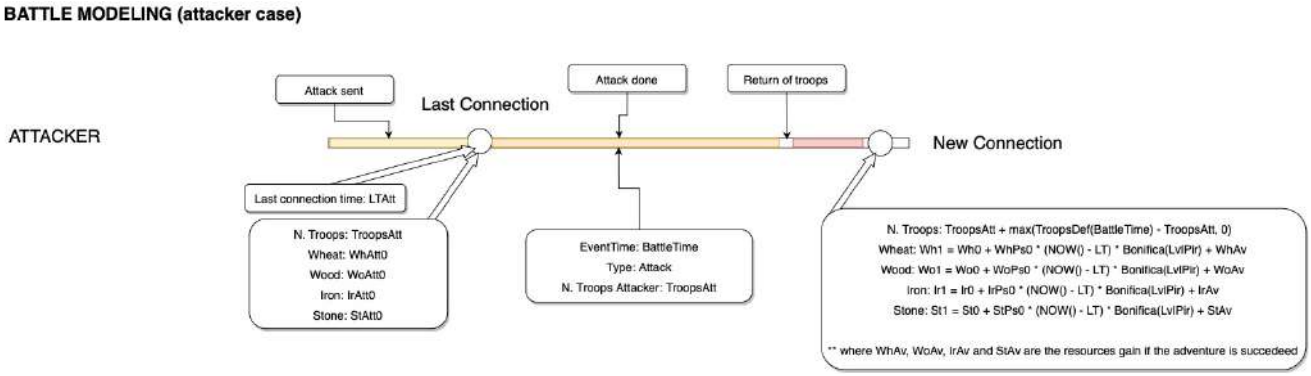

Figura 6.7: Simulació del cas del càlcul de recursos finals amb una aventura

Generalitzant tots els casos que existeixen a la primera versió del videojoc, i pensant en pròximes versions a futur com edificis bonificant produccions o intercanvis de recursos, la fórmula final per calcular els recursos en una nova connexió és donada per:

$$Res_1 = Res_0 + \sum_{i=0}^{n}(Res_{prod}[i] * (EvT[i+1] - EvT[i]) * Pir[i]) + \sum_{i=0}^{k} AvT[i]$$

on:

Res_0 = recursos inicials
Res_1 = recursos finals
$Res_{prod}[i]$ = producció del recurs en l'interval de temps i
$AvT[i]$ = recursos de l'aventura completada en l'interval de temps i
$Pir[i]$ = porcentantge de bonificació de la piràmide en l'interval de temps i
EvT = $[LT,\ EvT[1],\ EvT[2],\ \ldots,\ EvT[n],\ NOW]$ =
= vector dels esdeveniments que modifiquen la producció de recursos
$EvT[i]$ = esdeveniment finalitzat a l'instant de temps i
LT = temps última petició
NOW = temps actual

6.1.3 Persistència de les regles del videojoc

Com ja s'ha parlat en la seva secció pertinent, s'ha implementat un disseny desacoblant el *frontend* del *backend*. A més, el *frontend* simularà esdeveniments i accions a l'usuari per evitar tenir una comunicació constant amb el servidor.

Per facilitar aquest treball, s'ha dissenyat un sistema perquè el *frontend* i el *backend* funcionin amb les mateixes regles i siguin escalables en tot moment. Aquest sistema es basa en un fitxer de tipus JSON anomenat *Gamemaster* que inclourà totes les directrius, directoris, identificadors i noms necessaris per al correcte funcionament del joc *Age of Ra*. La idea d'aquest fitxer ha estat inspirada del videojoc de mòbil Pokémon Go [12].

Aquest fitxer estarà situat en el *backend*, però quan un nou client es connecti a l'aplicació a través del *frontend*, es farà una primera crida al servidor per a descarregar-se aquest fitxer i tenir totes la informació necessària evitant les excessives consultes al servidor.

Actualment, a l'última versió del *Gamemaster*, el fitxer conté les següents entrades:

- **Lands**: Conjunt d'informació de tots els tipus de terres que existeixen en el sistema amb les seves caselles de recursos corresponents.
- **Adventures**: Conjunt d'informació de totes les aventures, així com els seus costos, recompenses, dificultats, etc.
- **Resources**: Conjunt d'informació de tots els recursos del sistema i informació de les millores de les caselles de recursos.
- **Troops**: Conjunt d'informació de totes les tropes del sistema.
- **Buildings**: Conjunt d'informació de tots els edificis existents, així com les seves millores i accions.
- **Pyramid**: Conjunt d'informació de les millores de la piràmide i les bonificacions.

6.1.4 Components de Vue.js

Els components són un concepte clau a l'hora de dissenyar una aplicació amb Vue.js. Aquests s'utilitzen per dividir la interfície de l'aplicació en peces independents i reutilitzables. Cada component consta d'una plantilla de codi HTML, una secció *script* per definir la lògica, i la possibilitat de definir un estil propi del component [13].

S'ha escollit fer servir els components de Vue.js perquè ajuden a millorar el manteniment i l'escalabilitat d'una aplicació, ja que permeten reutilitzar i modular fàcilment el codi. Aquest fet ha facilitat durant el transcurs de les diferents iteracions del videojoc, realitzar els canvis necessaris sense afectar a la resta de l'aplicació.

Un tret característic dels components de Vue.js, és que es poden imbricar dins d'altres components, cosa que ha permès crear jerarquies entre ells. En general, és comú pensar l'aplicació com un arbre organitzat de components Vue.js.

Els components de Vue.js que defineixen l'aplicació *Age of Ra* són els següents:

- **App**: Component pare que gestiona el canvi de pantalles principals entre l'inici de sessió, crear un usuari, pantalla d'administrador i pantalla de joc.
- **LogInForm**: Component que mostra i gestiona tota la lògica de l'inici de sessió.
- **SignUpForm**: Component que mostra i gestiona tota la lògica per registrar un usuari.
- **HomeButtons**: Component que mostra la pantalla principal on es deixa a l'usuari escollir si es vol iniciar sessió o registar en l'aplicació.
- **GamePage**: Component pare del videojoc. Aquest component es responsabilitza de mostrar o amagar els components fills, depenent de la pantalla o accions que l'usuari està efectuant en el videojoc. No gestiona la lògica del videojoc, sinó que responsabilitza a cada component fill.
- **AdventureSelector**: Component que mostra les aventures i gestiona la lògica d'efectuar aventures.
- **ResourcesQuantity**: Component que gestiona i mostra la quantitat de recursos d'una terra.
- **QueueConstructions**: Component que mostra la cua de construccions d'una terra.
- **LandsNavigation**: Component que gestiona la navegació entre terres d'un usuari, així com la creació de noves.
- **ResourcesFields**: Component que mostra totes les caselles de recursos de la terra, així com gestionar les possibles interaccions de les millores.
- **BuildingsFields**: Component que mostra totes les caselles d'edificis de la terra, així com gestionar les possibles interaccions de les millores, construccions i accions d'edificis.
- **TroopsQuantity**: Component que mostra les tropes actuals de la terra.
- **QueueAdventures**: Component que mostra la cua d'aventures d'una terra.
- **RankingPage**: Component que gestiona i mostra la pàgina del rànquing.
- **TimeLinePage**: Component que gestiona i mostra la pàgina de la *timeline*.
- **AdminPanel**: Component pare de la pantalla dels usuaris administradors. No gestiona cap lògica, sinó que s'encarrega de mostrar o amagar els seus components fills.

- **AdventureEdit**: Component que mostra les aventures del sistema i gestiona la interacció per agregar de noves.
- **BuildingEdit**: Component que mostra els edificis del sistema i gestiona la interacció per agregar de nous.
- **TroopEdit**: Component que mostra les tropes del sistema i gestiona la interacció per agregar de noves.
- **ResourceEdit**: Component que mostra els recursos del sistema i gestiona la interacció per agregar de nous.

Aquests components segueixen la jerarquia mostrada en la Figura 6.8.

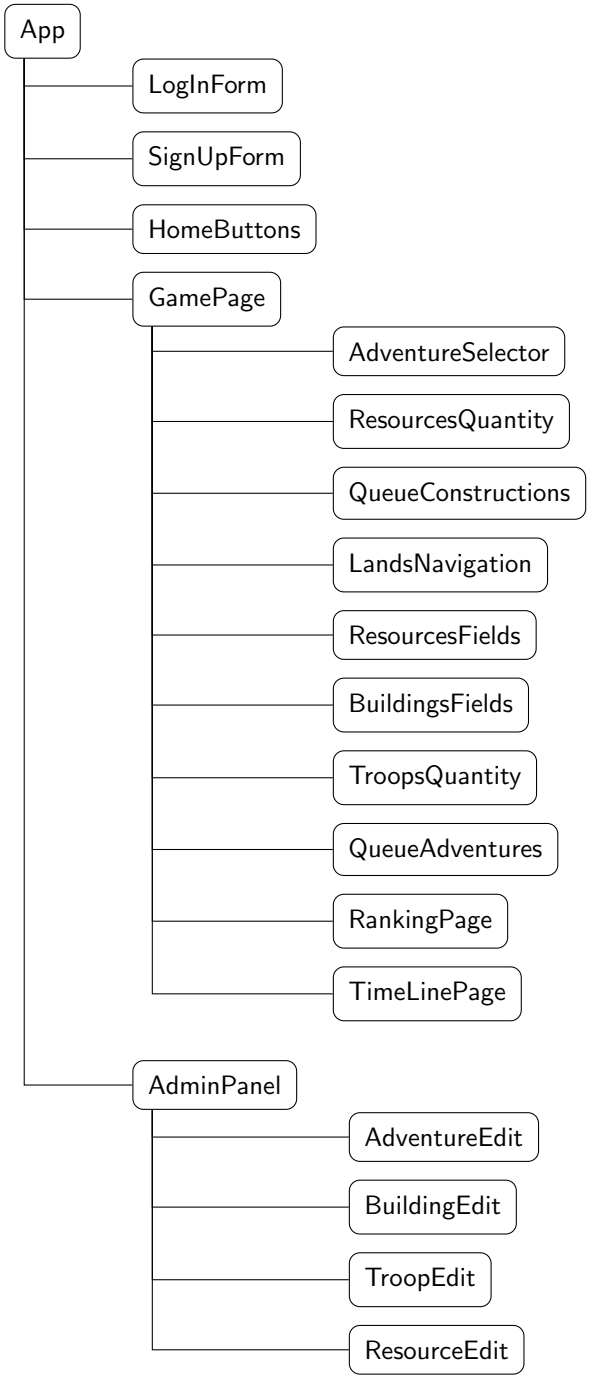

Figura 6.8: Arbre dels components Vue.js utilitzats en l'aplicació

En general, l'ús de components és un aspecte clau de Vue.js que permet als programadors crear aplicacions escalables i fàcils de mantenir.

6.1.5 SSOT i *stores* de Pinia

El concepte *Single Source of Truth* (SSOT) o única font de veritat fa referència a la pràctica d'emmagatzemar i gestionar dades en una ubicació o sistema centralitzat. Aquesta pràctica garanteix que totes les parts implicades en una aplicació utilitzin la mateixa informació actualitzada [14].

Utilitzar aquest concepte en el desenvolupament d'una aplicació garanteix un seguit de característiques:

- Coherència: una única font de veritat garanteix que tots els components treballin amb les mateixes dades, evitant així incoherències i errors.
- Eficàcia: l'ús del concepte SSOT racionalitza els processos i redueix la quantitat de temps dedicat a la conciliació de dades.
- Integritat de les dades: emmagatzemant les dades en una ubicació centralitzada, fa que sigui més fàcil mantenir la integritat d'aquestes.

Per tant, la implementació de la part de *frontend* de l'aplicació ha sigut programada aplicant el concepte d'una única font de veritat. Aquesta decisió es va prendre a causa de la dificultat que s'estava trobant en treballar amb diferents components de Vue.js que interaccionaven amb la mateixa informació.

Per implementar aquest concepte a Vue.js, cal introduir el concepte de *stores* i la llibreria més popular que existeix en el mercat per aplicar el SSOT: Pinia.

Pinia [30] és una biblioteca per crear aplicacions del costat del client altament dinàmiques utilitzant Vue.js i els seus components. Pinia proporciona un conjunt d'eines per gestionar l'estat i emmagatzemar dades de l'aplicació en una ubicació centralitzada. Aquestes eines s'implementen sobre un component modular que gestiona l'estat de l'aplicació anomenat *stores*.

Per implementar el concepte d'una única font de veritat al codi de l'*Age of Ra*, s'han hagut de crear 3 *stores* diferents:

- ***Gamemaster***: En aquesta *store* es guardarà tota la informació relativa a les regles del videjoc. Quan els components de Vue.js vulguin consultar les regles de joc, només hauran de consultar aquesta *store* que tindrà les dades actualitzades. Aquesta *store* només s'actualitza una vegada iniciada l'aplicació, fent que les consultes estiguin en memòria i no s'hagin de fer consultes al servidor remot.
- **Land**: En aquesta *store* es guardarà l'estat de la terra actual que s'està mostrant al jugador per pantalla. Qualsevol component de Vue.js que necessiti informació sobre la terra, haurà de consultar a aquesta *store*.

- **Player**: En aquesta *store* es guardarà l'estat de l'usuari que ha iniciat sessió en l'aplicació.

6.1.6 Versions del joc

Com ja s'ha explicat a la secció de Metodologia, s'ha desenvolupat el joc en diferents versions amb funcionalitats diferents en cada una. De les versions desenvolupades s'etiqueten en una versió major i una versió menor. Les implementacions de petites funcionalitats es presenten en les versions menors i quan existeix un producte prou fort es presenta una versió amb canvi de versió major. A més, la versió major v0 representa la fase beta del joc i la versió v1 representa la primera sortida oficial del joc.

Les versions que s'han desenvolupat durant el transcurs del projecte han sigut les següents:

v0.1

- S'implementa d'un sistema de registre i inici de sessió
- S'implementa un fitxer bàsic *Gamemaster*
- S'implementa un *frontend* bàsic amb Vue.js
- S'implementa MongoDB com a motor de base de dades
- Es pot crear una terra amb una piràmide bàsica
- Existeixen recursos a la terra.
- Es poden construir edificis
- Recursos generats automàticament

v0.2

- S'implementa un testing pel registre i inici de sessió
- S'implementa un testing per la creació de les terres
- Es poden millor edificis, piràmide i recursos.

v0.3

- S'implementa un testing per a les millores dels edificis, piràmide i recursos
- Actualització del fitxer *Gamemaster*

v0.4

- S'implementa la funcionalitat de reclutar tropes

- S'implementa la funcionalitat d'aventures
- S'implementa la compra de noves terres
- En fer un inici de sessió es guarden les cookies de l'usuari

v1.0

- S'afegeix un rànquing global
- S'afegeix la pàgina de *timeline* d'esdeveniments
- Es millora la interfície gràfica de l'usuari
- Desplegament de l'aplicació al públic

6.2 Desenvolupament del disseny gràfic

En aquesta secció s'explicarà en detall les decisions i la implementació del disseny gràfic final per l'aplicació, després de dissenyar la interfície d'usuari amb els *wireframes* ja presentats.

L'objectiu és presentar al lector la temàtica del videojoc, com s'ha dissenyat el logotip, com s'ha portat a terme el disseny dels recursos de l'aplicació, per finalment lligar-ho amb la implementació real pantalla per pantalla de l'aplicació *Age of Ra*.

6.2.1 Temàtica del joc

La temàtica escollida pel videojoc *Age of Ra* s'ha inspirat en l'estètica i cultura de l'antic Egipte. L'aplicació presentarà un esquema de colors daurat i ataronjat, tot fent servir imatges de piràmides, faraons, deus egipcis i símbols emblemàtics d'Egipte. A més, es presentarà un disseny net i modern, enfocat a la jugabilitat i immersió de la temàtica.

6.2.2 Disseny del logotip i icona

El disseny del logotip i la icona corresponent està inspirat en els jeroglífics egipcis. Per crear el logotip s'ha fet ús de la font "Mythology of Egypt" [15], a més d'una ombra per darrere les lletres, tot respectant els colors prèviament definits a la temàtica del joc.

Es mostra el disseny final del logotip en un fons blanc (Figura 6.9), tant com en el fons de l'aplicació de navegador del joc (Figura 6.10).

Figura 6.9: Logotip *Age of Ra* amb un fons blanc

Figura 6.10: Logotip *Age of Ra* amb un fons de color blau

Fer un bon disseny de la icona de la nostra aplicació és important, ja que es vol que l'usuari sigui capaç d'identificar el videojoc respecte a altres aplicacions. A més, la icona acostuma a ocupar poc espai, fet que fa hagi de ser reconeixible amb poca informació i un ús de colors particulars. Per això, s'ha aprofitat les inicials de *Age of Ra*, per crear la icona amb un fons de color blau clar, en contrast del daurat. Es pot observar aquesta icona en la Figura 6.11. A més, per comprovar que s'ha fet un disseny correcte, s'ha comprovat el correcte visualitzar en diferents navegadors web (Figura 6.12) [16].

Figura 6.11: Icona *Age of Ra*

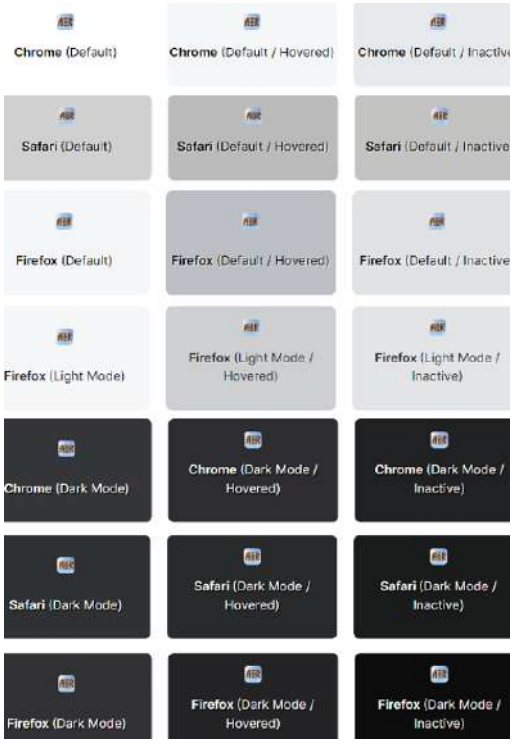

Figura 6.12: Icona interaccionat amb diferents navegadors web

6.2.3 Disseny dels edificis

Un cop ja escollit el logotip, icona i dissenyat els *wireframes* del videojoc es va escollir l'estil d'aquest. Es va procedir a fer recerca i investigació sobre altres videojocs similars, i es va acabar escollint fer un estil simplista i minimalista. A partir d'això es van dissenyar tots els elements que formarien els edificis, tropes i recursos. Tots els elements han sigut creats a partir d'*assets* de lliure ús proporcionats per MacroVector [17]. Es pot observar un conjunt de tots els *assets* que s'usaran per dissenyar els recursos del videojoc a la Figura 6.13.

Figura 6.13: *Assets* utilitzats per dissenyar el videojoc

A partir de tots els elements es va procedir a crear la ciutat (part central on existeixen edificis) que tindria cada terra. Aquest disseny es pot observar a la Figura 6.14.

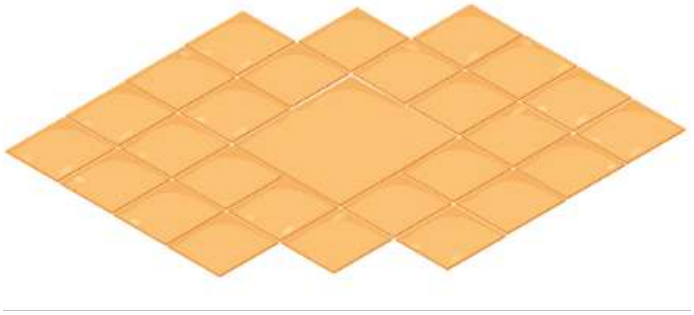

Figura 6.14: Disseny de la part central de la terra on s'ubiquen edificis

Un cop dissenyada la part central, falta mostrar el desenvolupament de cada edifici que es pot situar en les caselles observables. En la part central està reservada l'edifici principal (piràmide). En aquest cas es va procedir a fer un disseny que variés depenent del nivell. Aquests dissenys es poden observar a la Figura 6.15.

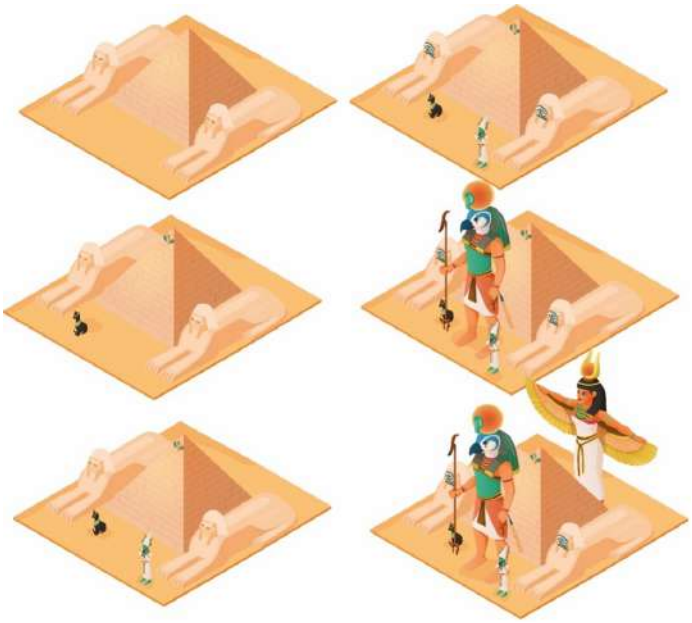

Figura 6.15: Diferents estils de la piràmide depenent del nivell

Per la part dels edificis, s'han generat un total de 15 edificis que habiliten certes accions al jugador (entrenar tropes noves, millorar tropes, amagar recursos, etc.).

Es pot veure un art conceptual de tots els edificis amb una breu descripció de les accions que permet a la Figura 6.16.

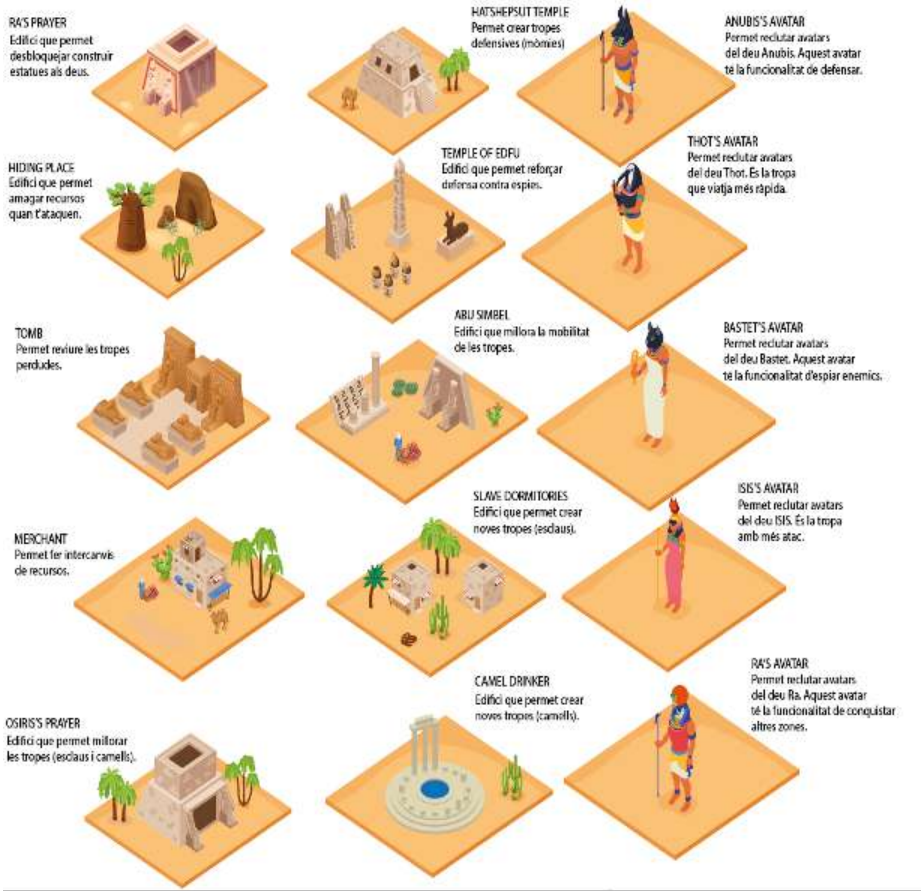

Figura 6.16: Art conceptual del conjunt d'edificis del *Age of Ra*

Es deixa per una millora del futur fer els dissenys de totes les variants dels edificis depenent del seu nivell.

6.2.4 Disseny dels recursos

Els recursos són el principal element de la jugabilitat del *Age of Ra*. Seguint l'estètica i l'estil, es volia dissenyar uns recursos diferenciables entre ells i minimalistes. Per tant, es va decidir inicialment comptar amb 4 recursos: cereal (color groc), ferro (color gris), fusta (color verd) i pedra (color marró). En la Figura 6.17. es poden observar els dissenys finals de cada recurs.

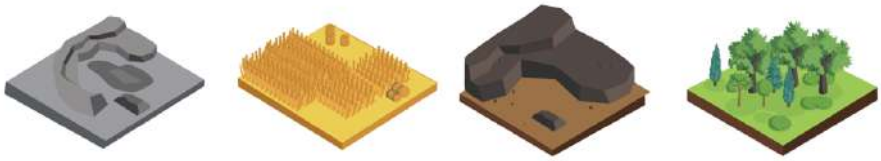

Figura 6.17: Disseny dels recursos que formen *Age of Ra*

La unió de caselles de recursos, formen un mapa o terra associada a cada jugador. Aquesta terra té caselles de recursos de tipus aleatori. Un exemple de disseny final de mapa d'una terra seria la Figura 6.18.

Figura 6.18: Disseny final de les caselles de recursos d'una terra

6.2.5 Disseny de les tropes

S'han dissenyat un conjunt de tropes, utilitzant simbologies i deus egipcis. Aquestes tropes són: esclaus, mòmies, camells, déu Anubis, déu Thot, entre d'altres. Els dissenys finals de les tropes es troben a la Figura 6.19.

Figura 6.19: Disseny final de les tropes de *Age of Ra*

6.2.6 Disseny final del joc

En aquesta secció es mostrarà la implementació dels *wireframes* presentats a la secció "Interfície de l'usuari".

En la Figura 6.20 es mostra el disseny final de la pantalla d'inici de l'aplicació. Aquest disseny es basa en el logotip en gran i dos botons per iniciar sessió o registrar un nou usuari.

Figura 6.20: Disseny final de la pantalla d'inici

En la Figura 6.21 es mostra el disseny final de la pantalla d'inici de sessió. En

aquesta pantalla es mostren dos camps de texts per iniciar sessió amb l'usuari i contrasenya.

Figura 6.21: Disseny final de la pantalla d'inici de sessió

En la Figura 6.22 es mostra el disseny final d'un inici de sessió incorrecte.

Figura 6.22: Disseny final de la pantalla d'inici de sessió amb errors

En la Figura 6.23 es mostra el disseny d'un registre d'usuari. Es mostren tres camps de text: usuari, contrasenya i confirmar contrasenya.

Figura 6.23: Disseny final de la pantalla de registre d'usuari

En la Figura 6.24 es mostra el disseny final d'un registre d'usuari incorrecte.

Figura 6.24: Disseny final de la pantalla de registre d'usuari amb errors

En la Figura 6.25 es mostrà el disseny final de la pantalla principal i per defecte del videojoc *Age of Ra*. En aquesta pantalla es poden veure la majoria d'accions disponibles en l'aplicació: canviar de terra, crear nova terra, consultar recursos o tropes, millora casella de recursos, consultar cues d'esdeveniments i efectuar aventures.

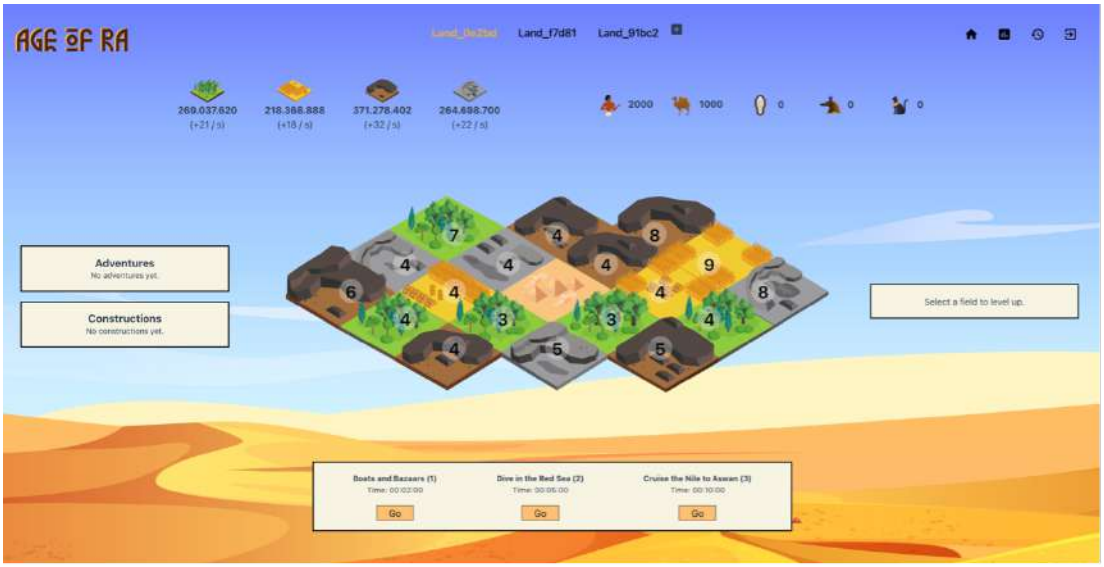

Figura 6.25: Disseny final de la pantalla de joc exterior

En la Figura 6.26 es mostra el disseny quan l'usuari intenta millorar una casella de recurs.

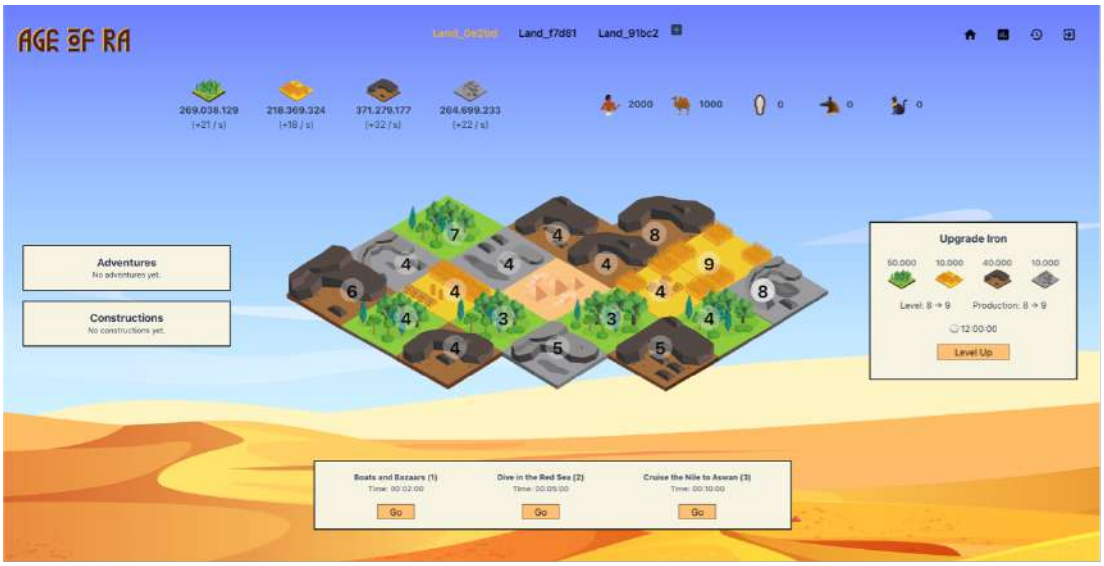

Figura 6.26: Disseny final de la pantalla per millorar un recurs

En la Figura 6.27 es mostra el disseny de quan un usuari intenta crear una nova terra.

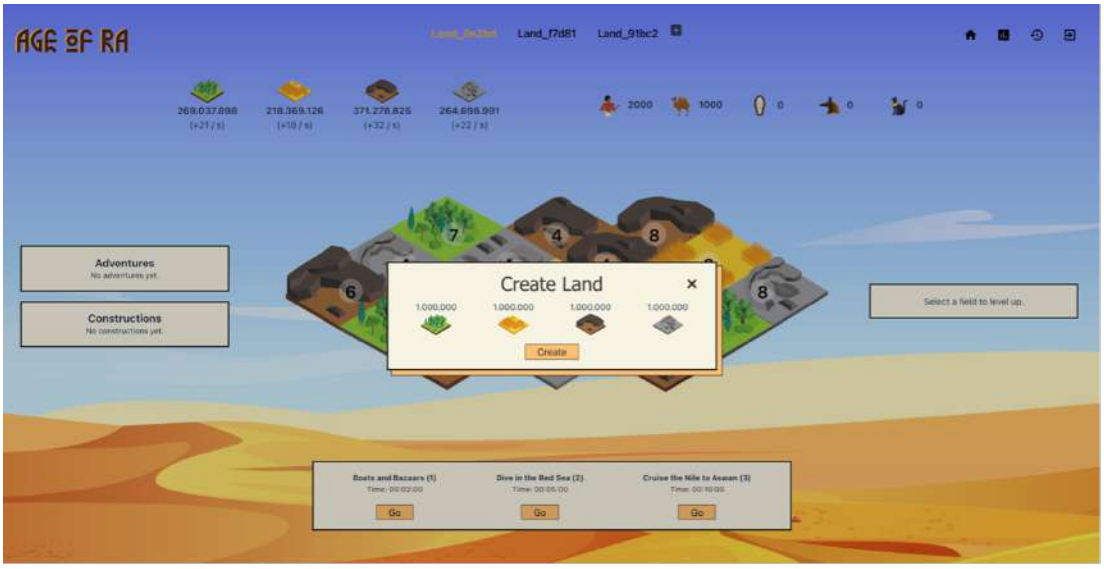

Figura 6.27: Disseny final de la pantalla per crear una terra

En la Figura 6.28 es mostra la pantalla de l'interior de la terra. En aquesta pantalla es poden efectuar la majoria d'accions que la pantalla anterior, excepte el bloc del centre que ha canviat al mapa d'edificis.

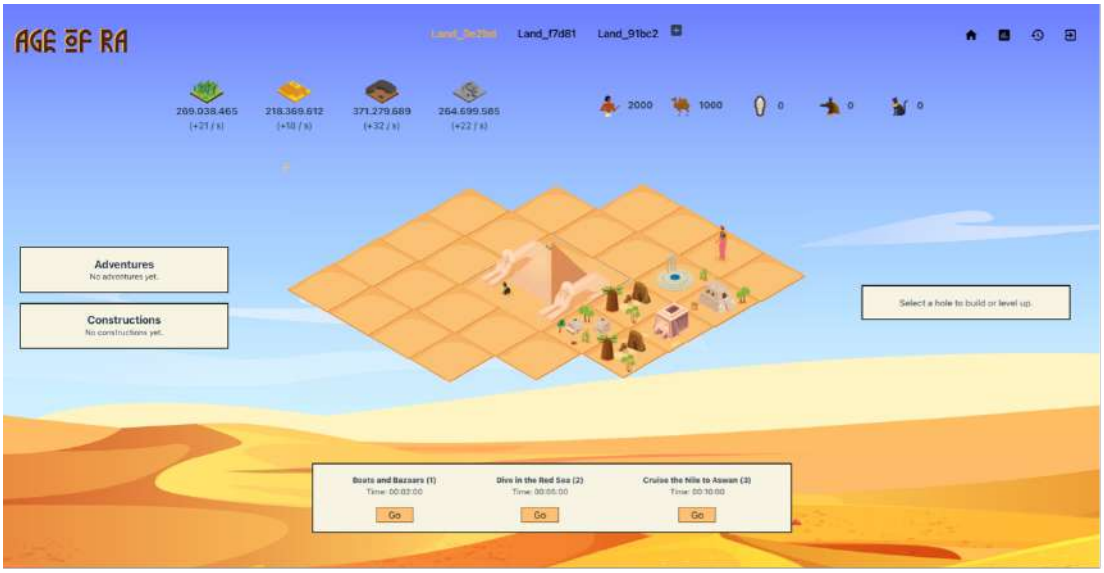

Figura 6.28: Disseny final de la pantalla de joc interior

En la Figura 6.29 es mostra el disseny quan un usuari intenta millorar la piràmide (edifici central).

Figura 6.29: Disseny final de la pantalla de millora de la piràmide

En la Figura 6.30 es mostra el disseny quan un usuari intenta millorar una casella d'edifici.

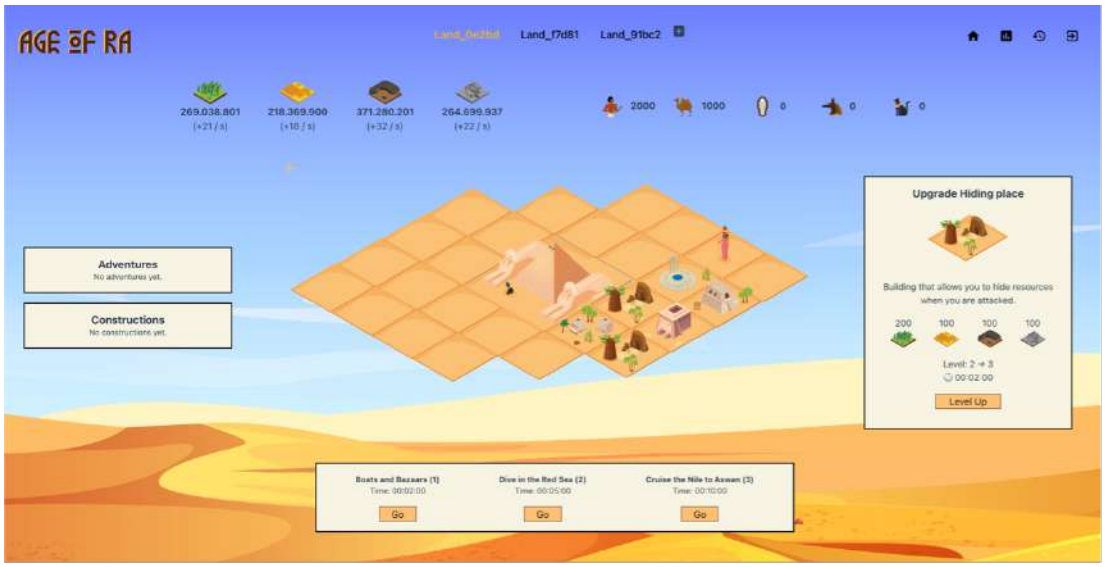

Figura 6.30: Disseny final de la pantalla de la millora d'un edifici

En la Figura 6.31 es mostra el disseny quan un usuari intenta construir en una casella d'edifici buida.

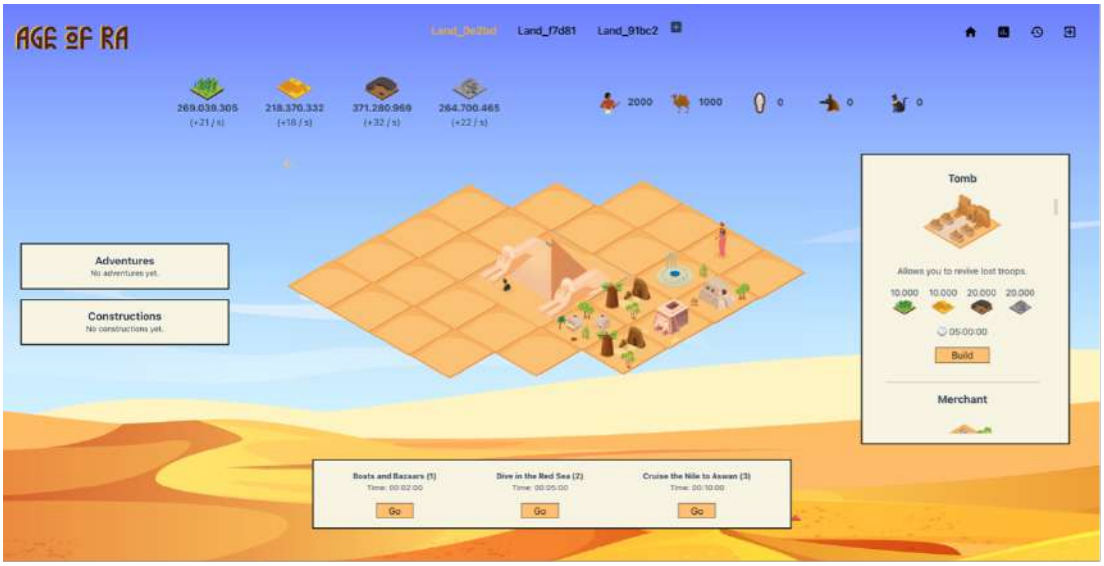

Figura 6.31: Disseny final de la pantalla de construcció d'un edifici

En la Figura 6.32 es mostra el disseny de la cua de construccions quan existeix alguna construcció en procés.

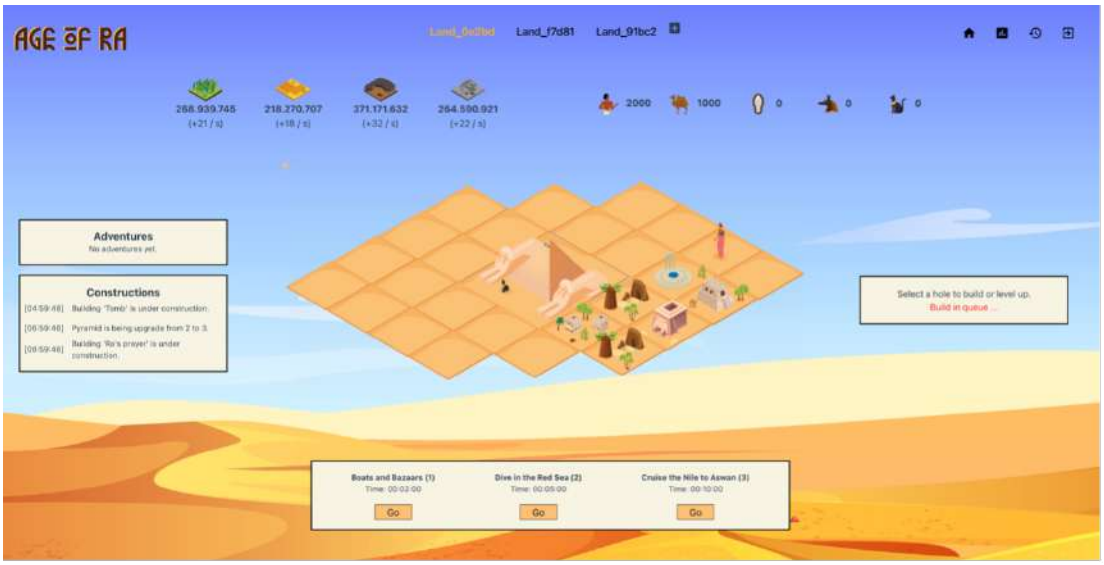

Figura 6.32: Disseny final de la cua de construccions

En la Figura 6.33 es mostra el disseny de la cua d'aventures quan existeix alguna aventura en procés.

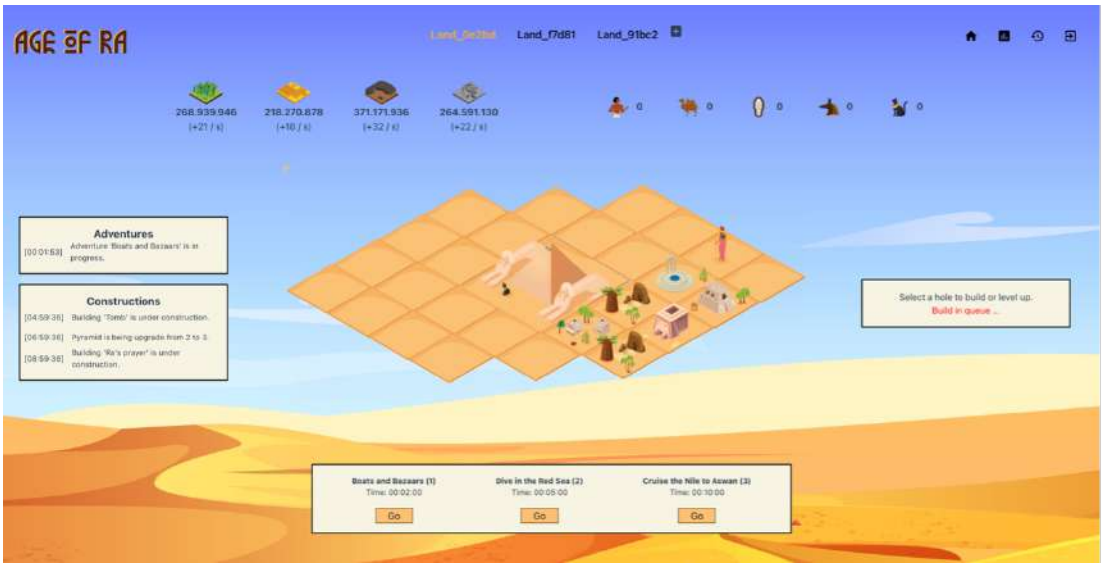

Figura 6.33: Disseny final de la cua d'aventures

En la Figura 6.34 es mostra el disseny quan un usuari intenta reclutar tropes a través d'un edifici.

Figura 6.34: Disseny final de la pantalla per reclutar tropes

En la Figura 6.35 es mostra el disseny de la pantalla del rànquing. En aquest rànquing apareixen els millors 10 usuaris del servidor, i a més, la posició actual del jugador.

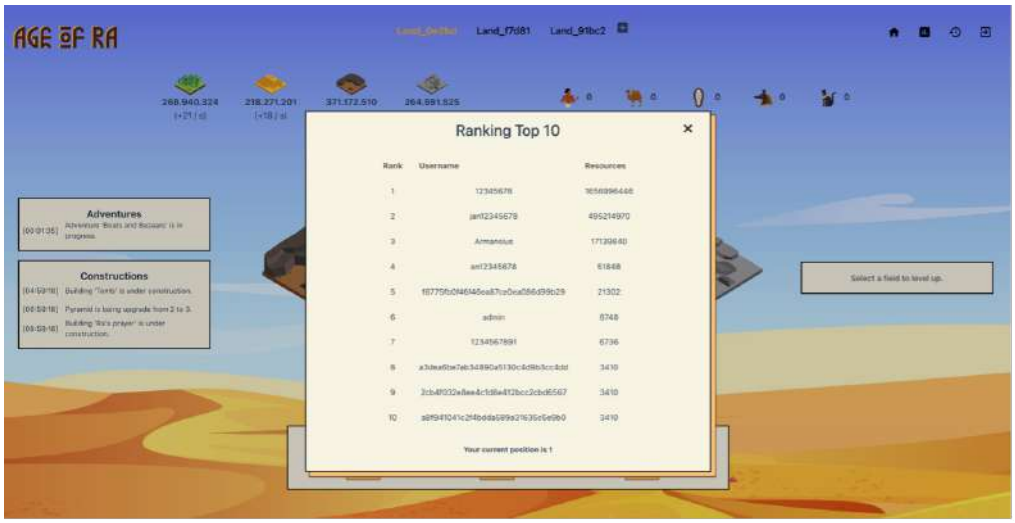

Figura 6.35: Disseny final de la pantalla del rànquing

En la Figura 6.36 es mostra el disseny de la pantalla de la *timeline*. En aquesta pantalla es pot desplaçar verticalment per observar tots els esdeveniments passats.

Figura 6.36: Disseny final de la pantalla de *timeline*

En la Figura 6.37 es mostra el disseny quan un usuari intenta tancar la sessió.

Figura 6.37: Disseny final de la pantalla per tancar la sessió

En la Figura 6.38 es mostra el disseny final de la pantalla principal dels usuaris tipus administrador. En aquesta pantalla es poden efectuar 4 opcions: editar aventures, editar edificis, editar recursos i editar tropes.

Figura 6.38: Disseny final de la pantalla principal dels administradors

En la Figura 6.39 es mostra el disseny de la pantalla per consultar (bloc esquerre) i afegir noves aventures (bloc dret).

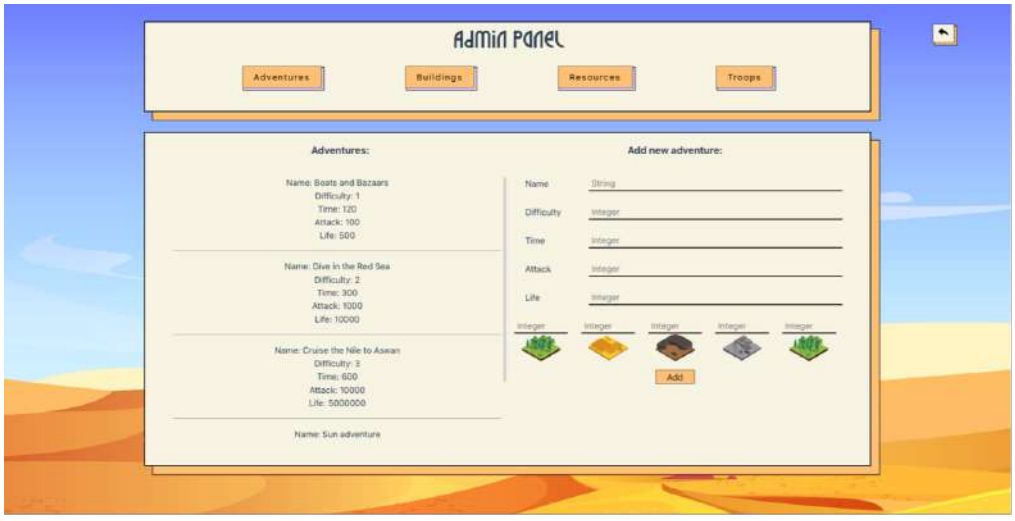

Figura 6.39: Disseny final de la pantalla per editar aventures dels administradors

En la Figura 6.40 es mostra el disseny de la pantalla per consultar (bloc esquerre) i afegir nous edificis (bloc dret).

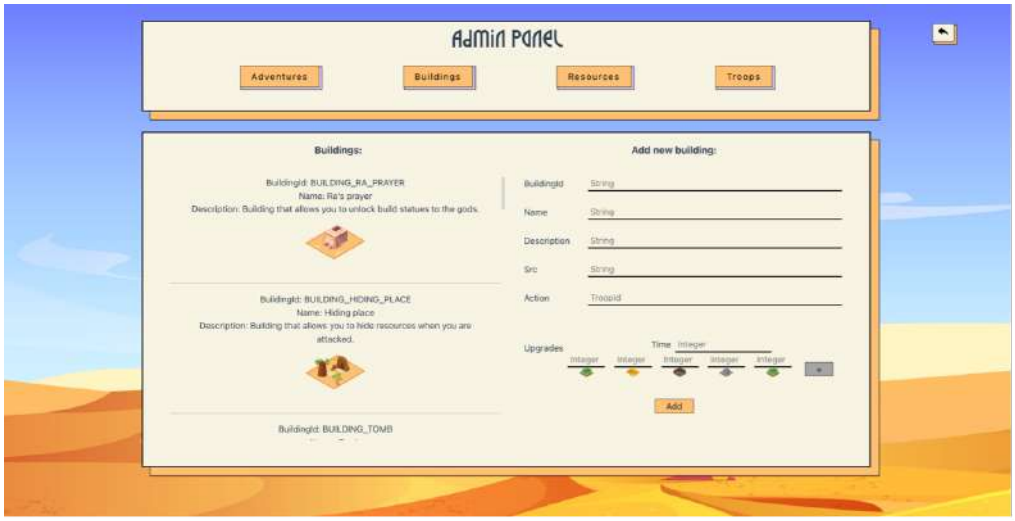

Figura 6.40: Disseny final de la pantalla per editar edificis dels administradors

En la Figura 6.41 es mostra el disseny de la pantalla per consultar (bloc esquerre) i afegir nous recursos (bloc dret).

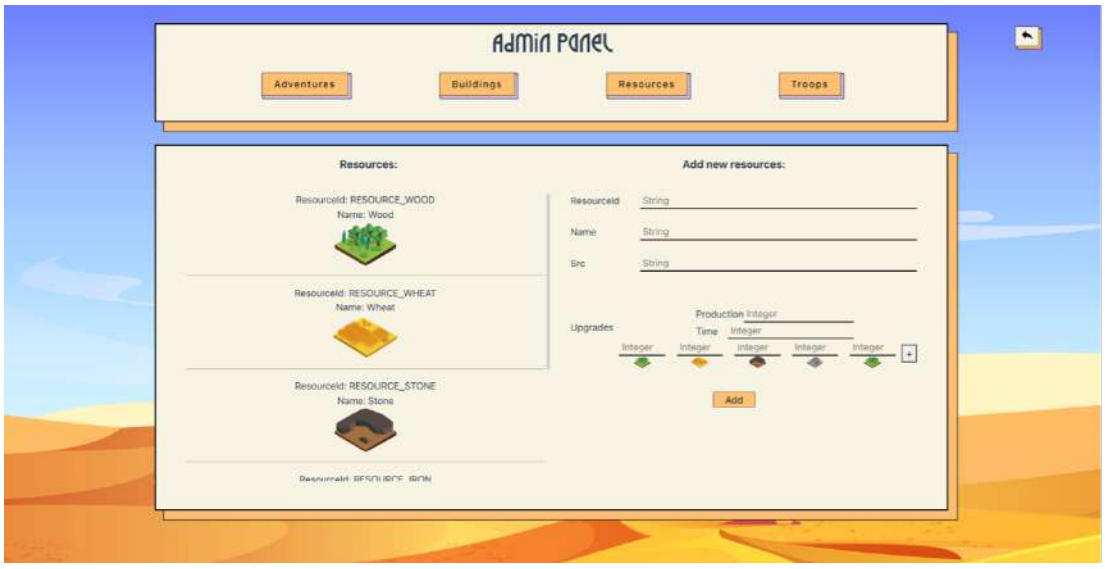

Figura 6.41: Disseny final de la pantalla per editar recursos dels administradors

Capítol 7

Experimentació i resultats

En aquest capítol es defineixen les eines utilitzades per experimentar i obtenir els resultats finals del sistema *Age of Ra*. Aquest capítol té una importància rellevant ja que respon als objectius establerts al principi del treball.

Per arribar a les conclusions desitjades, s'han realitzat proves de rendiment i funcionalitat de les característiques implementades. A més, aquestes proves han d'evolucionar amb les futures versions del videojoc. És essencial disposar d'una interfície visual per analitzar els resultats de manera eficient, especialment en la comparació del nombre de peticions i el temps de resposta.

Tenint en compte aquestes consideracions, s'ha determinat que l'ús de Locust és adequat per a aquest propòsit. Aquesta eina permetrà realitzar proves escalables i brindarà una visió clara del rendiment del sistema, contribuint a l'acompliment dels objectius establerts.

7.1 Locust com a eina de testeig

Locust [18] és una eina de *testing*, tant de rendiment com funcional, de càrrega distribuïda opensource, basada en Python.

Locust permet als desenvolupadors programar diferents unitats de proves automatitzades enfocades a fer peticions a API REST o aplicació web. S'ofereix simular els comportaments dels usuaris al lloc web testejat, per veure com respon a un gran volum de peticions.

Després de definir els tests com a codi Python i executar-los, Locust proporciona un seguiment en temps real dels resultats de les proves, com ara els temps de resposta, les taxes d'error i altres estadístiques, que permeten identificar testejar l'aplicació i detectar problemes de rendiment. Totes aquestes estadístiques es

mostren a través de la terminal o a través d'una interfície web. Per tant, es va creure necessari utilitzar aquesta llibreria per fer les proves funcionals del *backend* i les proves de rendiment de la nostra aplicació simulant un gran nombre de peticions.

7.2 Proves de funcionament

Pel que fa a l'estudi del correcte funcionament, s'ha proposat fer un *testing* funcional dels punts més crítics de l'aplicació i del sistema *backend* de simulació a temps real.

Locust ofereix un conjunt d'eines per separar i no haver d'executar totes les unitats de test a la vegada. Amb això, hem dividit les nostres proves de funcionalitats en 5 tags, i cada tag conté un seguit de tests que testejant una funcionalitat en particular.

El conjunt de tags i els seus tests són:

- **@tag('test_signup')**: Conjunt de tests que proven el correcte funcionament del registre d'un usuari.
    - Test1: Comprova un registre correcte.
    - Test2: Comprova que es dona un error quan es fa registre sense el nom d'usuari.
    - Test3: Comprova que es dona un error quan es fa registre sense la contrasenya.
    - Test4: Comprova que es dona un error si es fa registre amb un nom d'usuari existent en el sistema.
    - Test5: Comprova que es dona un error si el nom d'usuari té menys de 8 dígits.
    - Test6: Comprova que es dona un error si la contrasenya té menys de 8 dígits.
- **@tag('test_login')**: Conjunt de tests que proven el correcte funcionament de l'inici de sessió.
    - Test1: Comprova l'error d'inici de sessió si falta la contrasenya.
    - Test2: Comprova l'error d'inici de sessió si el nom d'usuari té menys de 8 caràcters.
    - Test3: Comprova l'error d'inici de sessió si la contrasenya té menys de 8 caràcters.

- **@tag('test_lands')**: Conjunt de tests que proven el correcte funcionament de les funcionalitats de la terra.
    - Test1: Comprova el correcte funcionament de la creació d'una terra i obtenció de totes les dades d'aquesta quan es demana pel seu identificador.
    - Test2: Comprova l'error amb un identificador de terra incorrecte, que no és propietat del jugador.
    - Test3: Comprova el correcte funcionament de la gestió de la construcció d'edificis.
- **@tag('test_upgrades')**: Conjunt de tests que proven el correcte funcionament de les millores de qualsevol edifici, recurs o pirámide.
    - Test1: Comprova el correcte funcionament de la gestió de les millores dels edificis.
    - Test2: Comprova el correcte funcionament de la gestió de les construccions d'edifici i posterior millora.
    - Test3: Comprova el correcte funcionament de la gestió de les millores dels recursos.
    - Test4: Comprova el correcte funcionament de la gestió de les millores de la piràmide.
- **@tag('test_troops')**: Conjunt de tests que proven el correcte funcionament de reclutar tropes i efectuar aventures.
    - Test1: Comprova el correcte funcionament de la gestió de les accions dels edificis (reclutar tropes) i efectuar aventures.

És important destacar que aquests tests s'han dissenyat durant el transcurs de les diferents versions del videojoc. S'han ampliat les unitats de test per anar cobrint les funcionalitats implementades. Ampliar la cobertura de casos d'aquests tests és una possible línia de futur d'aquest projecte.

7.3 Proves de rendiment

Un dels objectius d'aquest treball es basa a implementar un videojoc innovador tal que el seu disseny sigui escalable. L'aplicació ha de respondre de forma lineal a mesura que augmenten els usuaris connectats al videojoc.

Durant els capítols anteriors, ja s'ha debatut el perquè de les decisions de disseny i arquitectura el qual la implementació de l'aplicació estarà enfocada en assolir aquest objectiu. Ara, en aquesta secció i gràcies a l'ajuda de l'eina Locust, es

comprovaran els resultats obtinguts quan l'aplicació creix en nombre de clients i peticions.

Per fer aquestes comprovacions s'ha definit un test de Locust com el de la Figura 7.1. Cada cop que s'executa aquest test es genera un registre d'usuari en el sistema, que inicia sessió i comença a executar recàrregues de la terra cada segon. Això ha sigut possible gràcies a fer les peticions directament contra els endpoints de l'API REST de l'aplicació.

```
1  @tag('test_performance')
2      @task
3      def task(self):
4          username = uuid.uuid4().hex
5          password = uuid.uuid4().hex
6
7          self.client.post("/players", json={"username":
               username, "password": password})
8
9          with self.client.post("/players/login", json={"
               username": username, "password": password},
10                                catch_response=True) as resp:
11             auth = resp.json()["auth_key"]
12
13         auth_key = 'Bearer ' + auth
14
15         with self.client.get("/lands", headers={'
               Authorization': auth_key}, catch_response=True)
               as resp:
16             land_id = resp.json()['lands'][0]
17
18         while True:
19             with self.client.get("/lands/" + land_id,
                   headers={'Authorization': auth_key},
                   catch_response=True) as resp:
20                 if resp.status_code == 200:
21                     resp.success()
22                 else:
23                     resp.failure()
```

Figura 7.1: Codi del test de rendiment

A partir d'aquesta secció es comproven diferents casos d'usuaris fent peticions cada cert interval de temps per provar el rendiment del sistema i extreure conclusions. S'ha escollit fer proves sobre l'endpoint de recàrrega de terra, ja que és el bloc més important i més habitual en utilitzar en el videojoc.

En les figures següents es poden observar dos gràfics per cada cas:

- Gràfic 1: Nombre de peticions per segon al sistema.
- Gràfic 2: Temps mitjà de resposta en mil·lisegons per petició.
- Gràfic 3: Nombre total d'usuaris

Al principi de cada cas, es podrà observar com existeix un nombre de peticions major. Aquest fet és perquè s'estan comptabilitzant les peticions per crear usuaris i iniciar sessió. Un cop tots els usuaris han estat creats, cada cert interval de temps faran una petició de recàrrega de terra.

És important destacar uns punts, abans d'entrar en detall dels resultats. MongoDB restringeix el nombre de peticions per segon, per tant, és normal veure que existeix un màxim que el nostre sistema mai podrà assolir (50 peticions per segon) com que s'està utilitzant una base de dades al núvol gratuïta. També, en aquestes simulacions es pretén simular als usuaris i no a les peticions, en conseqüència, si un usuari està esperant una petició, aquest esperarà fins a obtenir resultats i no efectuarà cap petició més.

El primer test de rendiment està definit per un usuari que efectua una petició cada segon. Els resultats d'aquest test es mostren a la Figura 7.2.

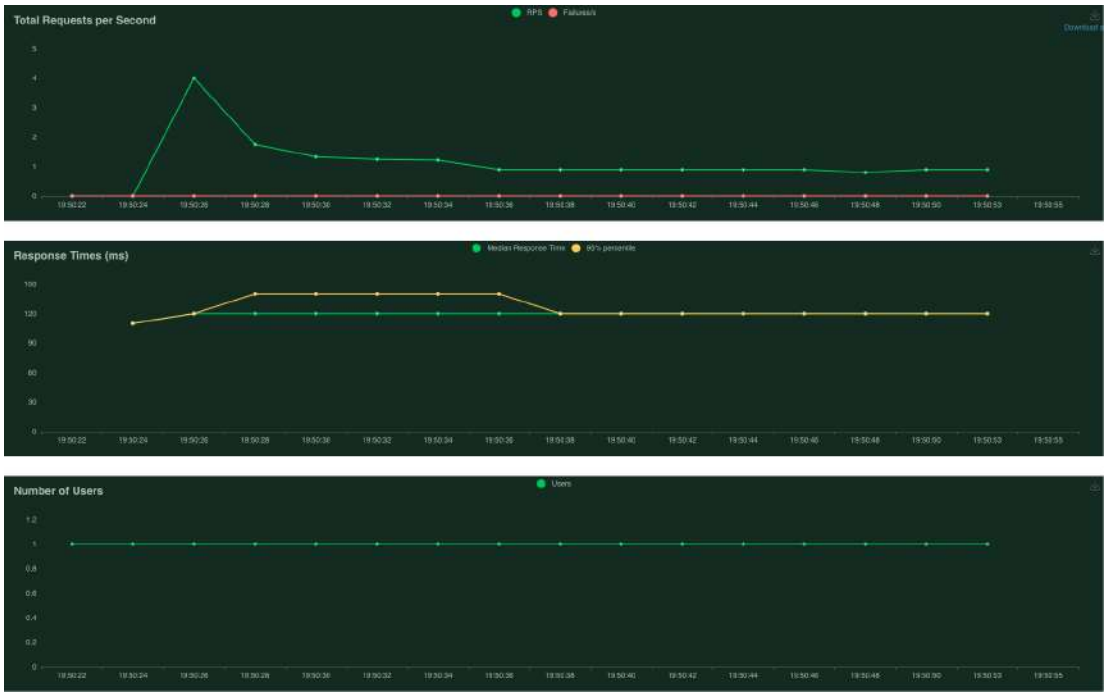

Figura 7.2: Resultats del test de rendiment amb un usuari

El segon test de rendiment està definit per 10 usuaris que efectua peticions cada segon. Els resultats d'aquest test es mostren a la Figura 7.3.

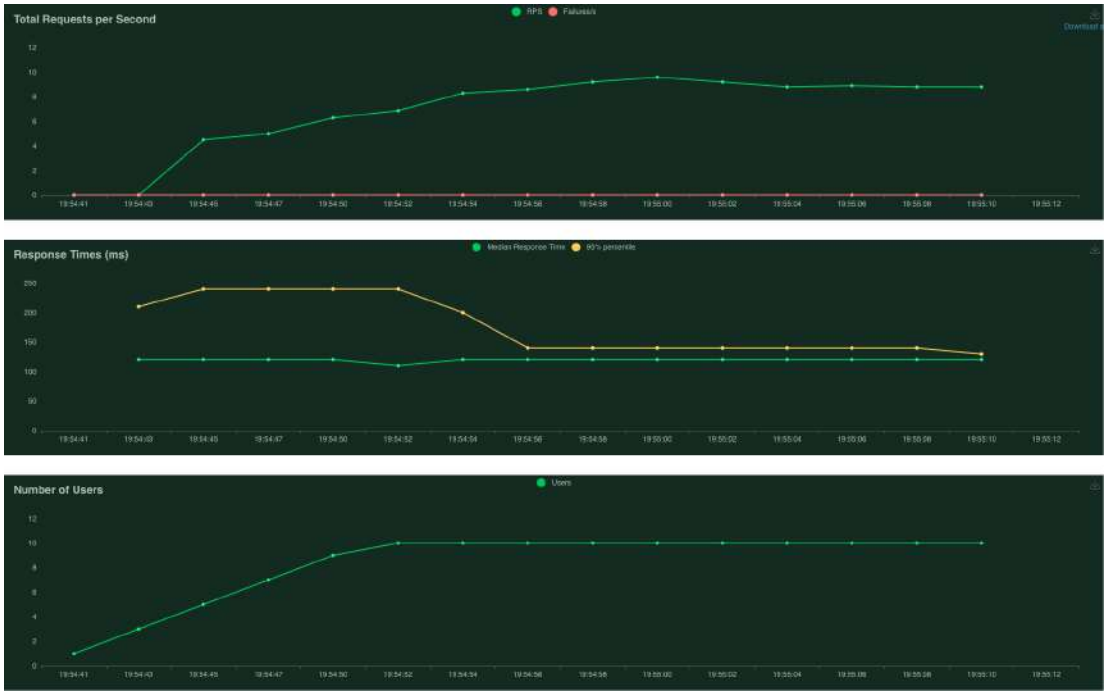

Figura 7.3: Resultats del test de rendiment amb 10 usuaris

El tercer test de rendiment està definit per 100 usuaris que efectuen peticions per segon. Els resultats d'aquest test es mostren a la Figura 7.4.

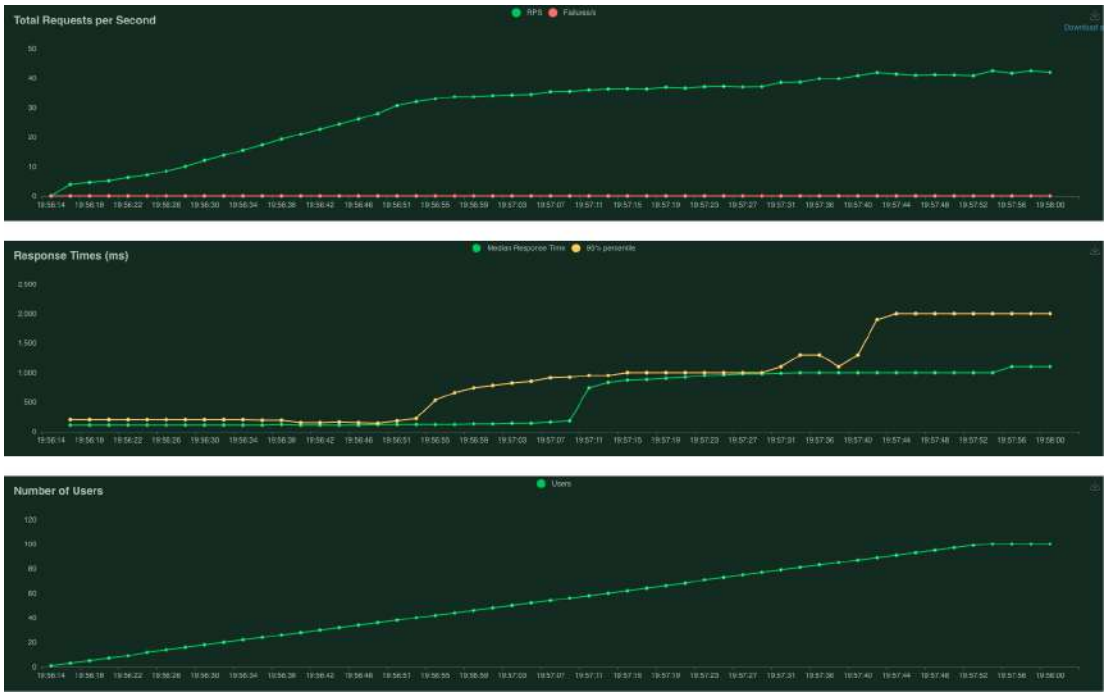

Figura 7.4: Resultats del test de rendiment amb 100 usuaris

L'últim test de rendiment està definit per 500 usuaris que efectuen el màxim de peticions per segon de forma incremental. Els resultats d'aquest test es mostren a la Figura 7.5.

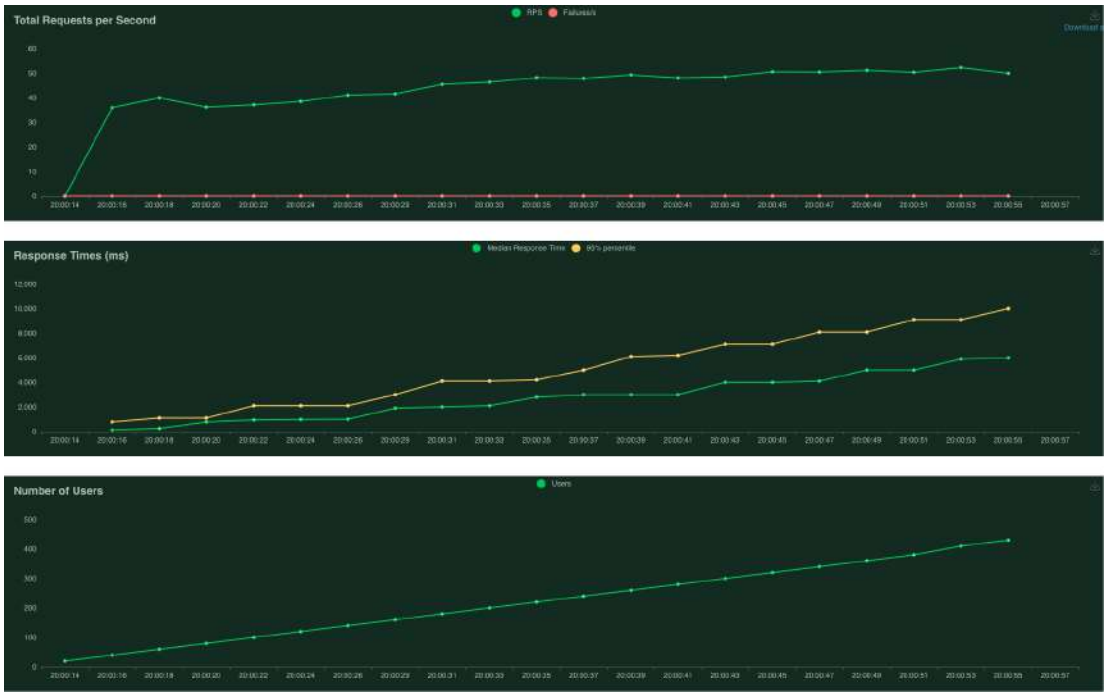

Figura 7.5: Resultats del test de rendiment amb 500 usuaris

Analitzant els resultats obtinguts, es pot extreure conclusions tals que:

- Per pocs usuaris i/o peticions s'obté un temps constant de resposta de les peticions. Aquest fet és degut al fet que la infraestructura limita més la resposta mínima, que no la lògica a executar per la petició. Aquest fet és el buscat inicialment, i es compleix l'objectiu de gestionar un gran nombre de peticions de forma lineal.
- Incrementant el nombre de peticions, es pot observar com clarament el temps de resposta de les peticions creix de forma lineal. Amb aquest resultat, es continua complit que el sistema creix linealment amb el nombre de peticions.
- A partir d'un cert nombre d'usuaris, aproximadament uns 500 usuaris, executant peticions simulataniament per segon, el sistema queda limitat a acceptar 50 peticions per segon pel fet que la base de dades, com s'ha comentat anteriorment, ens restringeix.

A més dels resultats anteriors, pel disseny implementat, el sistema no executa en paral·lel cap programa per actualitzar els recursos ni esdeveniments, i només ho fa quan existeixen peticions. Per tant, es pot concloure que s'ha implementat

un disseny escalable tal que l'aplicació respon de forma lineal a mesura que augmenten les peticions al servidor indiferentment dels clients connectats.

7.4 Afegir nous elements a les regles del joc

Un altre gran objectiu del treball és que el disseny implementat sigui eficient tal que tal que sigui fàcil afegir-hi nou contingut sense implicar canvis al codi existent.

Aquest objectiu es compleix des del moment que es programa el codi utilitzant un format JSON en les regles del videojoc. Així, en comptes de tenir escrites al *frontend* o al *backend* totes les regles, s'ha d'anar comprovant aquest fitxer per veure si existeixen noves versions.

Això fa que es tingui un videojoc modul·lar i obert a afegir nous canvis. A més, tots els canvis que s'introdueixin en el fitxer JSON han de ser compatibles amb les versions antigues d'aquest. És a dir, afegir una nova entrada al fitxer, per exemple un edifici, no ha de trencar les terres ja existents del sistema.

A més, com ja s'ha explicat anteriorment, per facilitar afegir noves entrades, s'ha habilitat una pantalla pels usuaris administradors perquè sigui més real l'objectiu de no haver de canvi codi per introduir noves variables al sistema.

En la Figura 7.6 es mostra un exemple d'aquests resultats, com un administrador ha afegit un nou recurs dels habituals.

Figura 7.6: Sistema *Age of Ra* amb un nou recurs

Finalment, la introducció de noves entrades al fitxer de regles del videojoc sense haver de tocar codi, ha resultat la incorporació de les mecàniques emergents en el projecte. Les mecàniques emergents són noves funcionalitats que es produeixen sense que els desenvolupadors del videojoc ho tinguessin previst.

Per posar un exemple, una nova mecànica emergent ha sigut la creació de les monedes. Un administrador pot crear un nou tipus de recurs que sigui una moneda. Com per defecte, encara no existeix cap terra que produeixi aquest

tipus de recurs, no es pot generar passivament. Per tant, l'administrador pot habilitar una aventura de dificultat elevada que produeixi generi aquesta moneda al jugador. I finalment, pot crear una tropa especial que només es pugui pagar amb aquesta moneda. Fins i tot, l'administrador podria habilitar que les monedes només es poguessin pagar a través de transaccions amb diners reals, tot i que ja canvia, en petita part, el codi del projecte.

Capítol 8

Cost del projecte

En aquesta secció es fa una anàlisi del cost final del projecte. Aquest cost està compost únicament pel cost temporal del projecte, que corresponen al volum d'hores dedicades des de la primera entrevista amb el tutor, fins a la realització d'aquest document.

8.1 Cost en hores

El volum de treball i esforç de la realització del projecte s'ha dividit en diferents categories.

- **Recerca i investigació**: Aquesta categoria inclou els treballs relacionats amb la investigació i estudi de les tecnologies utilitzades durant el projecte, així com diferents perspectives per assolir els objectius presentats.
- **Disseny**: Aquesta categoria inclou el temps dedicat a dissenyar i implementar els *wireframes*, logotip, icona, i altres elements de l'aplicació. La major part d'aquest treball ha estat efectuar amb Illustrator.
- **Implementació**: Aquesta categoria inclou el temps dedicat a implementar el sistema *Age of Ra*. S'inclouen tots els treballs dedicats a la programació: *frontend*, *backend* i persistència de dades.
- **Testing**: Aquesta categoria inclou el temps dedicat a testejar, tant de rendiment com de funcional, l'API del *backend*.
- **Documentació**: Aquesta categoria inclou el temps dedicat a documentar el codi, tant per la realització de tots els diagrames, com l'escriptura d'aquest document.
- **Entrevistes**: Aquesta categoria inclou el temps dedicat a les reunions de

seguiment i el temps dedicat a preparar-les.

En la Taula 8.1 es pot observar el total d'hores dedicat a cada categoria. Amb la finalitat de facilitar-ne la comprensió per al lector, aquestes hores s'han arrodonit.

Categoria	Hores	%
Recerca i investigació	50	11.1
Disseny	50	11.1
Implementació	200	44.5
Testing	20	4.4
Documentació	120	26.7
Entrevistes	10	2.2
Total	450	100

Table 8.1: Distribució de les hores per categoria

Finalment, en la Figura 8.1 es mostra un diagrama del total d'hores, en percentatge, dedicades a cada categoria.

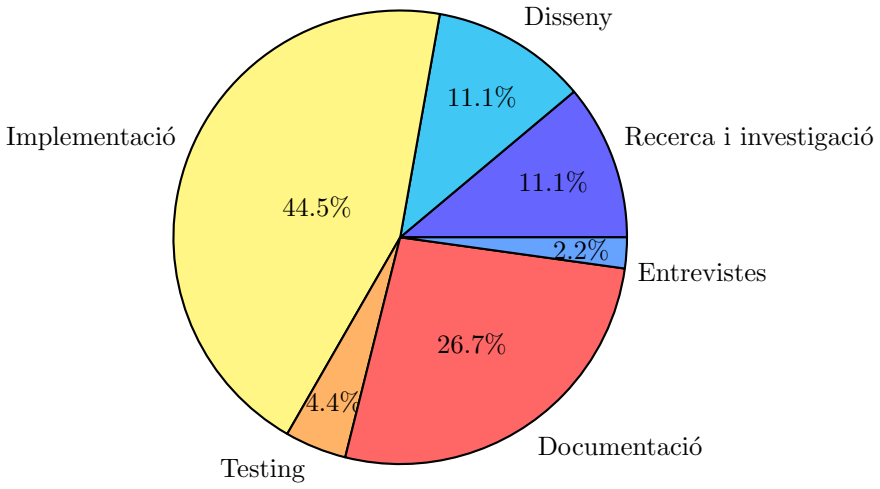

Figura 8.1: Distribució en percentatge del cost temporal del projecte

Capítol 9

Conclusions i futur

En aquest capítol es presenten les conclusions finals del projecte, tant a nivell tècnic com personal. També s'analitza en quina mesura s'han complert els objectius establerts a la introducció i es presenten algunes línies de futur per continuar treballant en futures versions del videojoc.

9.1 Conclusions tècniques

En aquest projecte s'ha dissenyat i implementat un videojoc multijugador d'estratègia en temps real proposant una temàtica pròpia i diferent de les analitzades a l'estudi de mercat, solucionant els possibles problemes de rendiment a causa del temps de càrrega i de resposta del sistema.

En la primera fase del projecte, s'han analitzat els quatre videojocs de gènere MMORTS més populars i rendibles del mercat. A partir d'aquí, s'ha proposat un nou videojoc que incorpori els punts forts trobats i millori els punts febles més característics. Es pot destacar com el model de videojoc proposat s'enfocava a la construcció i en la gestió d'un imperi egipci, havia de ser fluid a través de gràfics senzills i havia de minimitzar la recàrrega completa de les pàgines.

Durant l'apartat de disseny, s'ha dedicat una gran feina d'investigació sobre com efectuar el càlcul d'esdeveniments en temps real de forma eficient sense generar excessives peticions al servidor. S'ha conclòs que la millor forma d'implementar el càlcul en temps real dels esdeveniments és simulant les accions a la part de *frontend* i crear un seguit de regles per fer els càlculs en diferits per la part del *backend*.

En aquest treball, s'ha demostrat la importància de les decisions tecnològiques prèvies a la implementació. En el transcurs d'aquestes decisions es va decidir per implementar un càlcul en diferit per simular els esdeveniments en temps

real. Com s'han observat a l'apartat de resultats, això ha fet que el programi respongui de forma lineal quan s'augmenten el nombre de peticions per segon. A més, de la importància de no perdre recursos calculant esdeveniments per clients desconnectats. També, aquests resultats van acompanyats d'haver separat el *backend* del *frontend* amb una SPA. Això ha permès tenir una modulabilitat i fluïdes per la part de client, sense perdre l'única font de veritat del *backend*.

Tot i això, és important destacar que es podia haver decidit per una solució totalment diferent com haver fet un càlcul en temps real creant, fent que el servidor estigués permanentment calculant totes les accions per tots els clients, i mantingués sempre actualitzat el client. Aquesta solució, també és viable, i per extreure més conclusions s'hauria de fer un estudi de rendiment i poder-la comparar de veritat amb la proposada al treball. A més, amb aquesta opció, tot i possiblement no ser tan bona en rendiment, hauria solucionat el gran treball de disseny del càlcul en diferit i facilitat les seves fórmules. A més, depenent de la tipografia de videojoc i del nombre d'usuaris i peticions per segon, podria ser més vàlida que el càlcul en diferit.

Finalment, un altre punt destacable d'aquest projecte ha sigut fer un disseny fàcilment ampliable. A més, aquest objectiu ha permès obtenir mecàniques emergents sense fer canvis en el codi, un resultat que no s'esperava al principi del treball. Amb aquest objectiu complert, fa que el treball es quedi obert a ser continuat en un futur per pròximes versions sense implicar una gran càrrega de feina.

Per altra banda, existeix una limitació al nombre de peticions que es poden efectuar per segon, fent que encara que l'aplicació creixi de forma lineal pel temps de resposta d'acord amb el nombre de peticions, queda limitada pel temps de resposta de les consultes a la base de dades de MongoDB.

Pel que fa als altres objectius, no s'ha pogut complir de provar el videojoc en públic i ha quedat pendent implementar una acció real d'interacció entre jugadors de diferents terres. Aquestes dues metes han quedat finalment fora del projecte, a l'haver prioritzat el disseny i implementació de l'estructura bàsica jugable de l'aplicació.

9.2 Conclusions personals

El desenvolupament d'aquest projecte m'ha suposat un repte interessant i gratificant des del punt de vista tècnic i personal. A través de l'anàlisi, disseny i implementació del videojoc *Age of Ra*, he pogut aplicar els coneixements adquirits durant aquests estudis i aprofundir en l'àmbit del desenvolupament web i les seves arquitectures.

Aquest projecte va començar com un petit treball amb uns llenguatges i uns *frameworks* i ha acabat sent un altre completament diferent. Ha sigut un camí de

molts canvis, proves i reflexionar. Al final, el projecte finalitzat i el seu resultat visual ha superat les expectatives que tenia al principi del curs.

Una de les principals lliçons apreses durant aquest projecte és la importància de dissenyar abans d'implementar. Els objectius principals han sigut molt condicionants a l'hora d'implementar l'arquitectura i d'escollir tecnologies. Aquesta importància de dissenyar abans d'implementar és una bona metodologia que ja s'havia treballat durant el grau en diverses assignatures, però no és fins que es realitza un projecte d'aquestes característiques que no s'observa l'impacte que pot tenir.

A més, el fet de lligar en certa forma el treball final de grau amb les pràctiques dutes a terme a MongoDB, ha fet que fos més completa l'experiència, tant per la part de les pràctiques com aquest projecte.

En conclusió, aquest últim any treballant en l'*Age of Ra* ha suposat una experiència enriquidora que m'ha permès aplicar els coneixements teòrics adquirits en el grau i aprofundir en l'àmbit del desenvolupament web. Ha posat de manifest la importància de l'escalabilitat i rendiment, així com m'ha proporcionat habilitats valuoses en el disseny i implementació d'aplicacions web. Estic satisfet amb els resultats obtinguts i motivat per continuar explorant i treure futures versions del videojoc *Age of Ra*.

9.3 Línies de futur

A continuació es mostren un seguit de millores que s'ha considerat per seguir continuant i treballant amb el projecte i així, millorar els resultats obtinguts. Aquestes millores es basen en possibles versions del videojoc que poden ser llençades en un futur:

- Desplegar el videojoc en un entorn real i provar-lo en públic. El fet de provar-lo en el públic real, habilitaria a fer un seguiment de les estadístiques de joc, per així acabar de balancejar les regles, recursos, preus, etc.
- Implementar accions que requereixin una acció multijugador real. Algunes idees que es van deixar fora del projecte han sigut una transmissió de recursos entre terres de diferents jugadors o poder atacar a terres controlades per altres jugadors.
- Habilitar aliances entre jugadors. Si es vol anar pel camí de permetre atacs entre jugadors, s'ha de treballar en una forma que els jugadors d'interessos similars s'ajudin enviant tropes o recursos.

Capítol 10

Bibliografia

[1] Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson.

[2] Boehm, B. O. (1986). A spiral model of software development and enhancement. *ACM Sigsoft Software Engineering Notes, 11*(4), 14–24. https://doi:10.1145/12944.12948

[3] Travian Games GmbH. (2023). *Travian: Legends – International — The Online Multiplayer Strategy Game*. https://www.travian.com/

[4] Nielsen, J. (2009). *Usability engineering*. Morgan Kaufmann.

[5] InnoGames GmbH. (2023). *El clásico juego de navegador Guerras Tribales*. https://www.guerrastribales.es/

[6] Gameforge AG. (2023). *OGame - ¡El juego de navegador con más éxito del universo!*. https://lobby.ogame.gameforge.com/es_ES/

[7] Gameforge AG. (2023). *Ikariam - ¡Construye, conquista y gobierna!*. https://lobby.ikariam.gameforge.com/es_ES/

[8] IEEE Std. (1998). *IEEE Recommended Practice for Software Requirements Specifications*. IEEE Std 830-1998. https://doi.org/10.1109/ieeestd.1998.88286

[9] Sacha Greif. (2023). *State of JavaScript*. https://stateofjs.com/en-us/

[10] Amazon Web Services, Inc. (2023). *¿Qué es una API de RESTful? - Explicación de API de RESTful - AWS*. https://aws.amazon.com/es/what-is/restful-api/

[11] Mozilla. (2023). *WebSockets - Referencia de la API Web — MDN*. https://developer.mozilla.org/es/docs/Web/API/WebSockets_API

[12] PokeMiners. (2023). *GitHub - PokeMiners/game_masters: Repository for latest JSON and txt versions of the Game Master for Pokemon GO*. https://github.com/PokeMiners/game_masters

[13] Vue.js. (2023). *Components Basics — Vue.js*. https://vuejs.org/guide/essentials/component-basics.html

[14] Eduardo San Martin Morote. (2023). *Pinia — The intuitive store for Vue.js*. https://pinia.vuejs.org/

[15] Dafont. (2023). *Mythology Of Egypt Font — dafont.com*. https://www.dafont.com/mythology-of-egypt.font

[16] Colin Keany. (2021). *Favicon Checker*. http://www.colinkeany.com/favicon-checker/

[17] Freepik Company S.L. (2022). *Macrovector — Freepik*. https://www.freepik.es/autor/macrovector

[18] Locust.io (2023). *Locust - A modern load testing framework*. https://locust.io/

Capítol 11

Annex

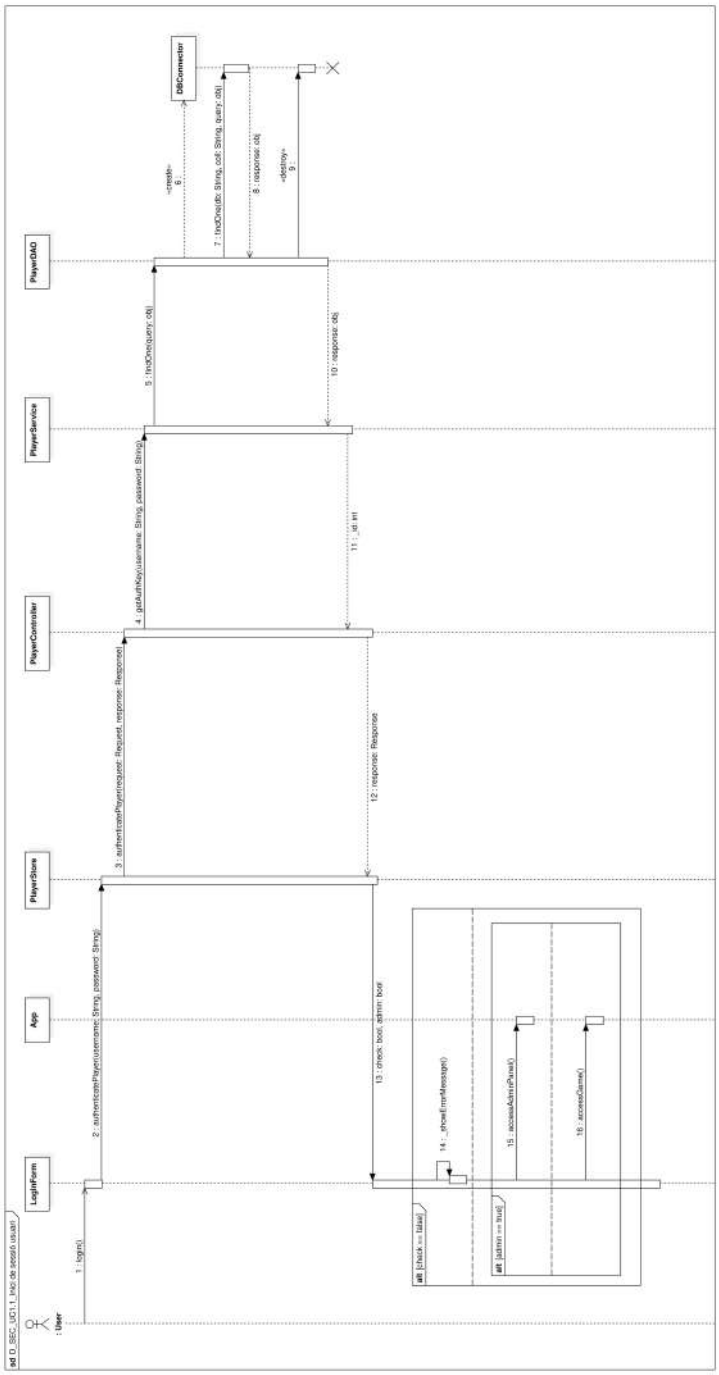

Figura 11.1: Diagrama de seqüència D_SEC_UC1.1_Inici de sessió usuari

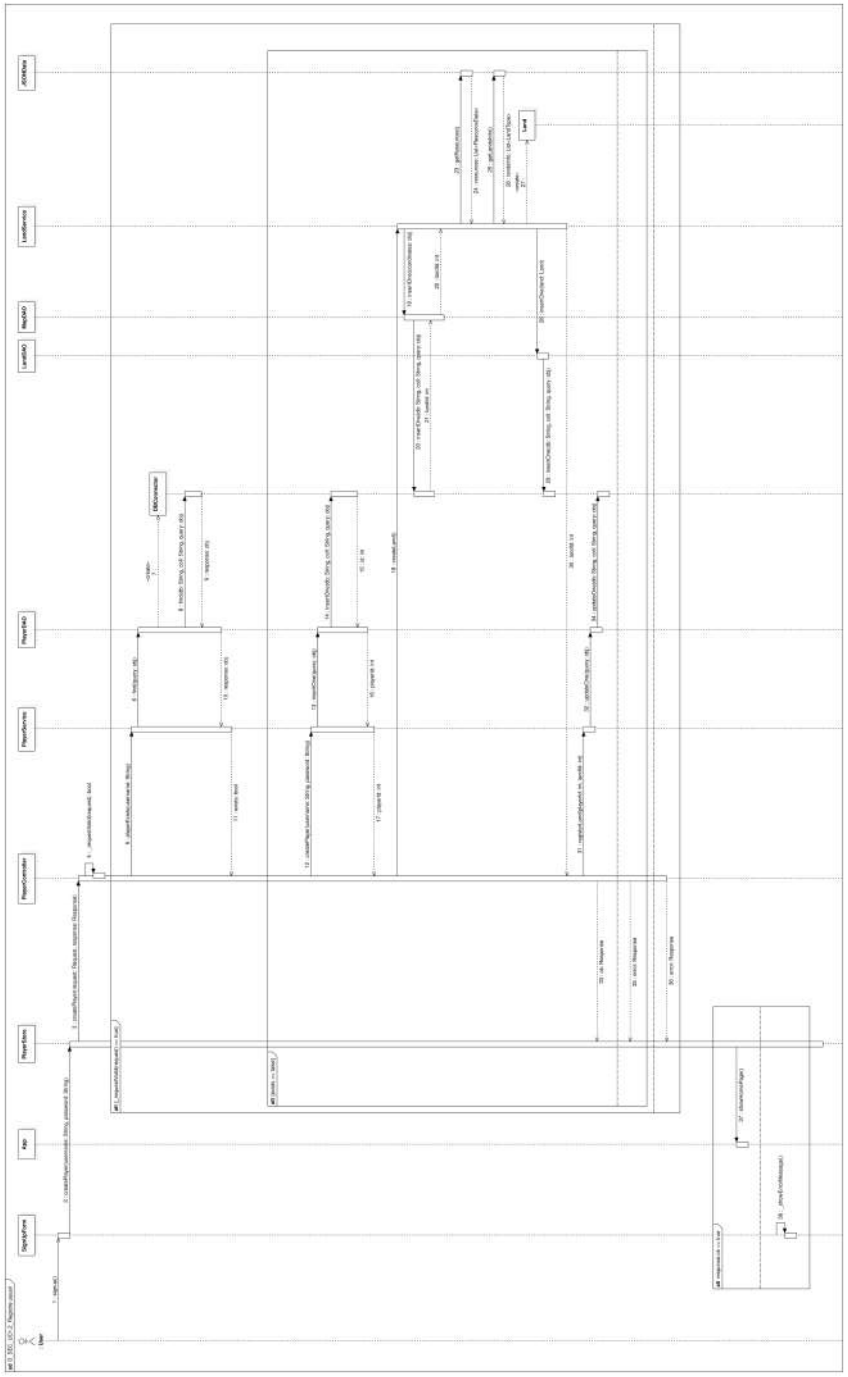

Figura 11.2: Diagrama de seqüència D_SEC_UC1.2_Registre usuari

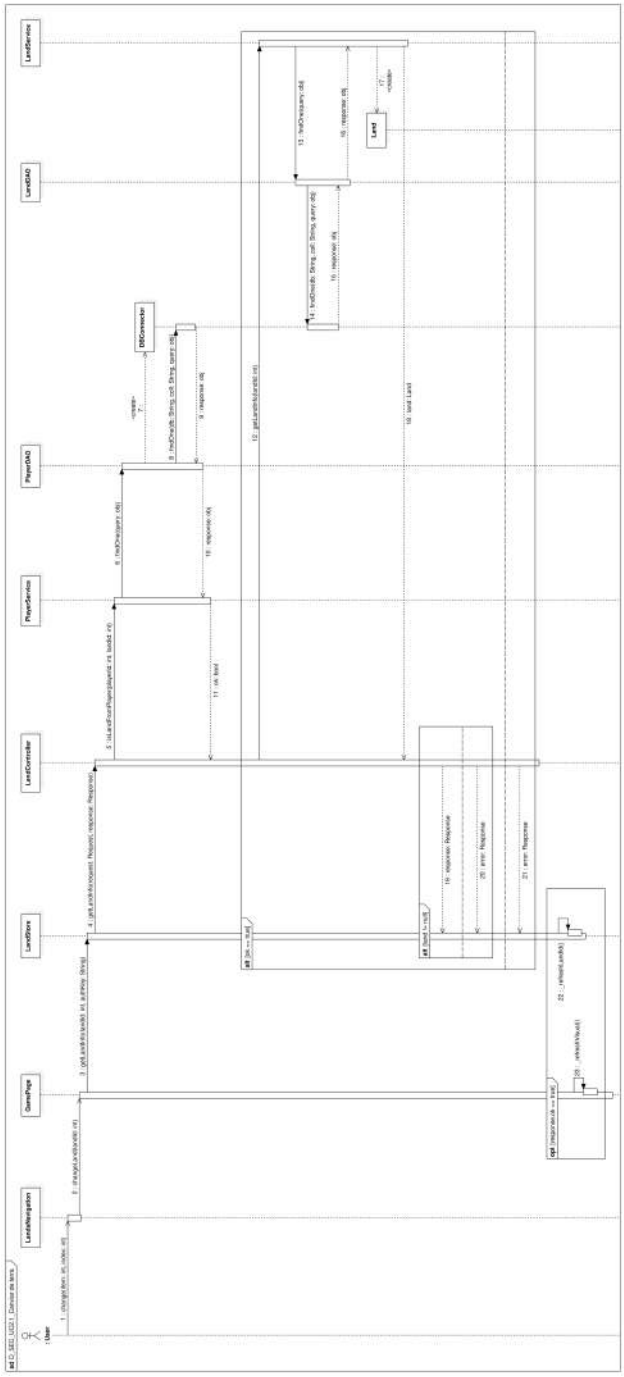

Figura 11.3: Diagrama de seqüència D_SEC_UC2.1_Canviar de terra

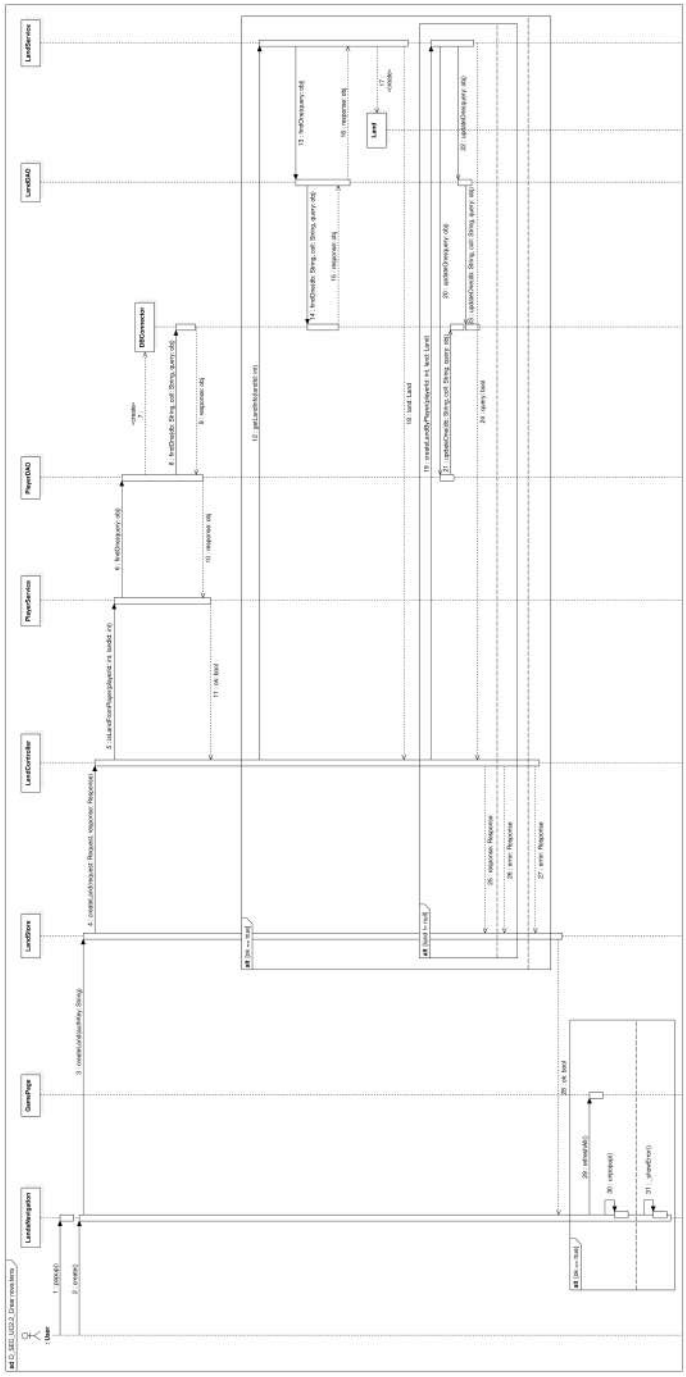

Figura 11.4: Diagrama de seqüència D_SEC_UC2.2_Crear nova terra

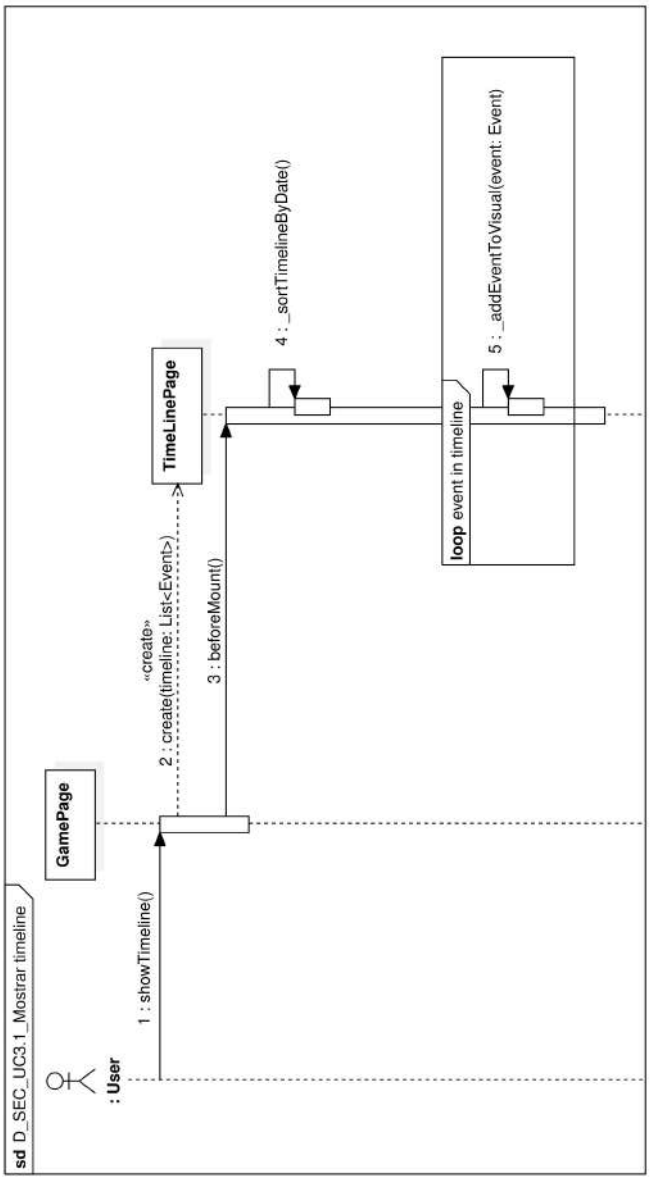

Figura 11.5: Diagrama de seqüència D_SEC_UC3.1_Mostrar *timeline*

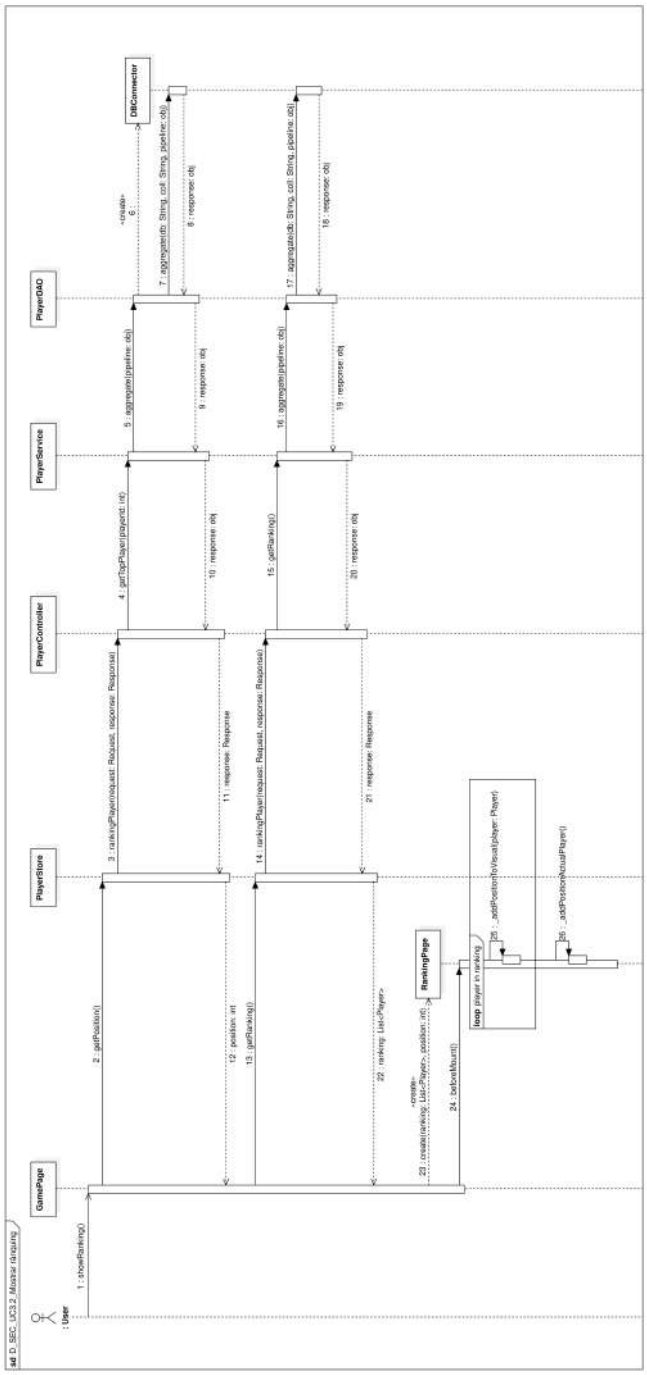

Figura 11.6: Diagrama de seqüència D_SEC_UC3.2_Mostrar rànquing

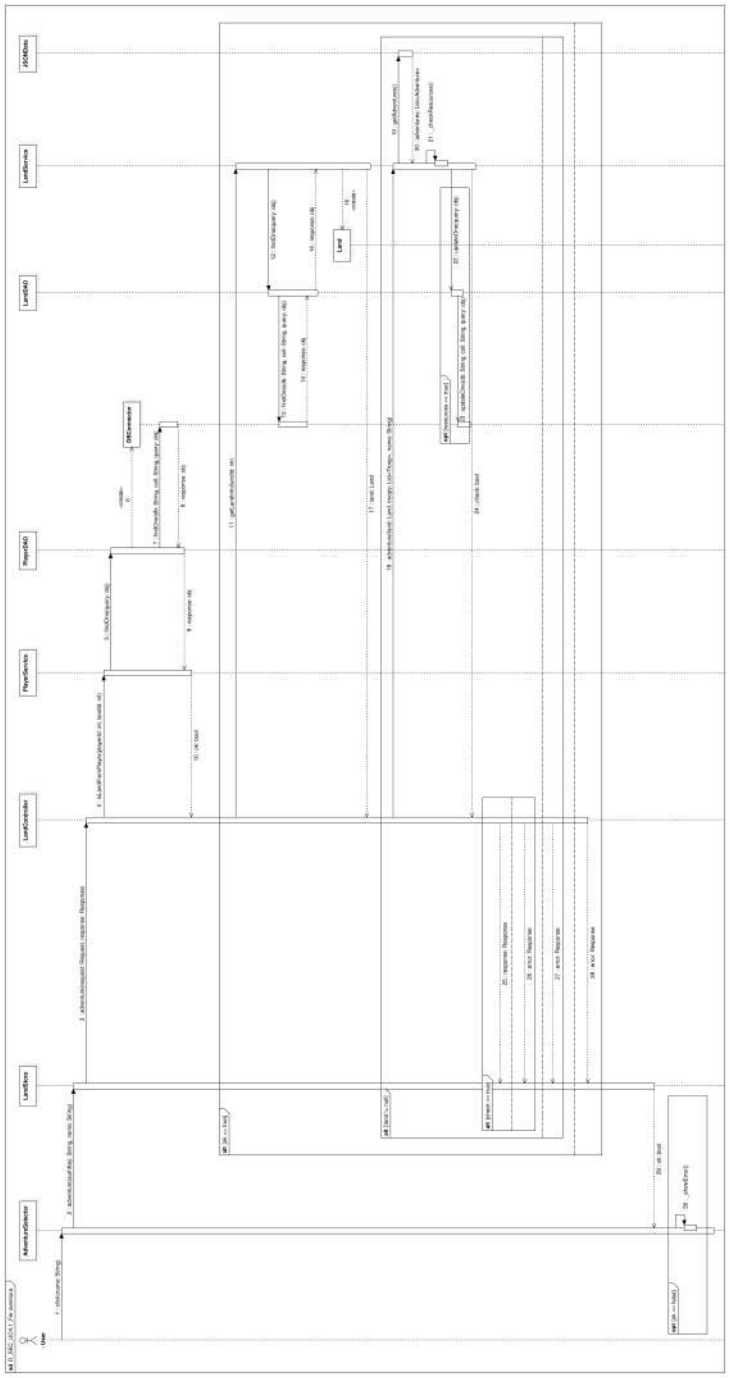

Figura 11.7: Diagrama de seqüència D_SEC_UC4.1_Fer aventura

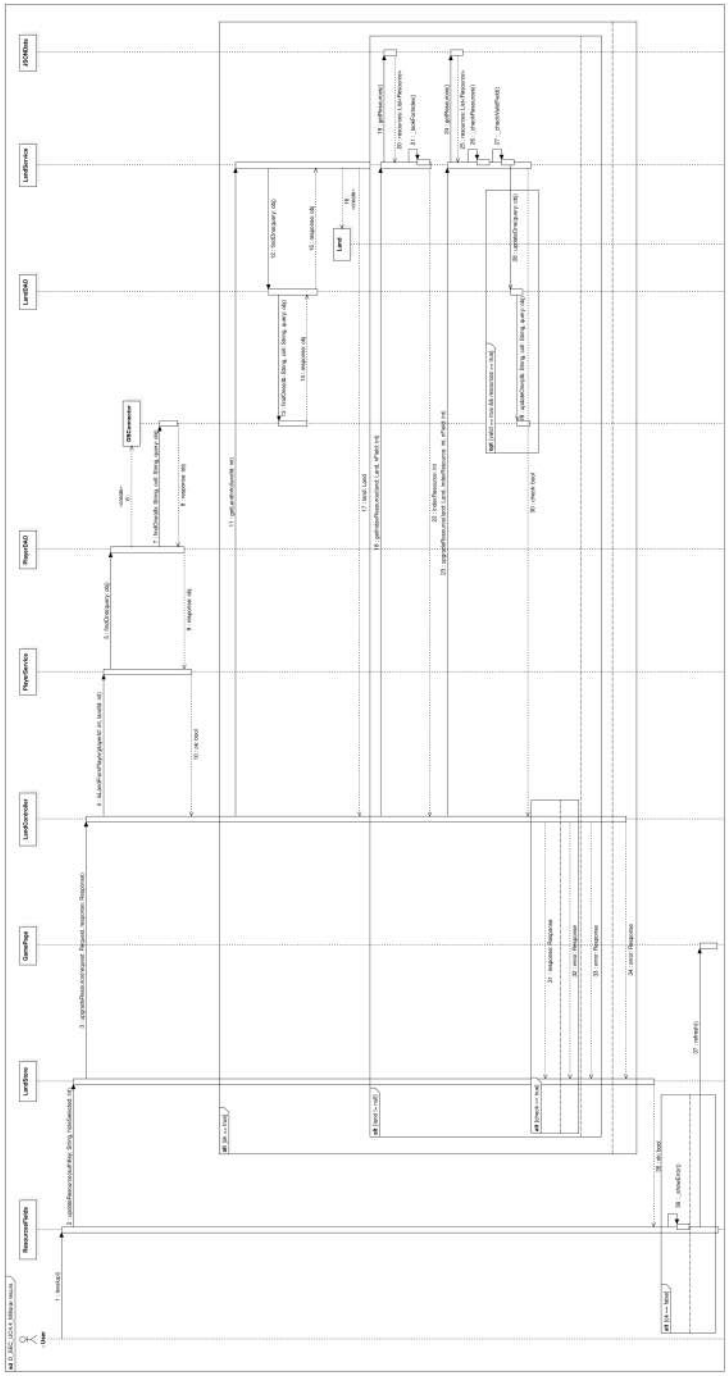

Figura 11.8: Diagrama de seqüència D_SEC_UC4.4_Millorar recurs

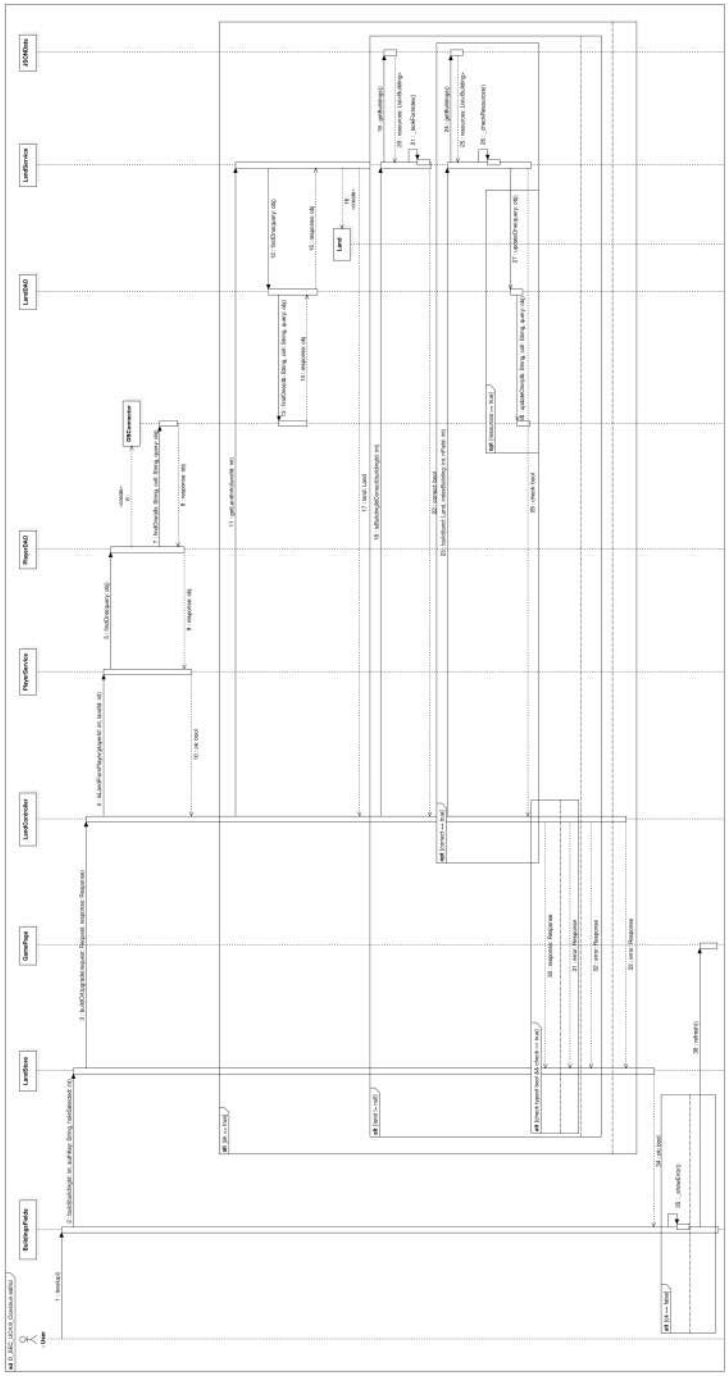

Figura 11.9: Diagrama de seqüència D_SEC_UC4.9_Construir edifici