TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

Joc en línia dels Colonitzadors de l'Illa

MEMÒRIA

JOFRE SANT MUNIESA

TUTOR: DAVID RÓDENAS PICÓ

CURS 2021/2022

TecnoCampus
Mataró-Maresme

Agraïments

En primer lloc, vull agrair el meu tutor David Ródenas qui m'ha ajudat amb els seus coneixements durant el transcurs del projecte.

També m'agradaria agrair a la meva mare i als meus avis el seu suport, ja que sempre han estat al meu costat durant tota la carrera.

Abstract

The goal of this project is to develop an online multiplayer board game, in web format, like Catan. The development has been through TDD. The backend was developed using the Spring framework and the ECS pattern to maximize game performance. The frontend has been made with React and Redux to be able to manage and centralize the state of the application and finally the graphics have been generated using SVG.

Resumen

El objetivo de este proyecto es desarrollar un juego de mesa multijugador en línea, en formato web, similar al Catan. El desarrollo ha sido mediante TDD. El “backend” se ha desarrollado utilizando el “framework” de “Spring” y el patrón ECS a fin de optimizar al máximo el rendimiento del juego. El “frontend” se ha realizado con “React” y “Redux” para poder gestionar y centralizar el estado de la aplicación y, finalmente, los gráficos se han generado mediante SVG.

Resum

L’objectiu d’aquest projecte és desenvolupar un joc de taula multijugador en línia, en format web, semblant al Catan. El desenvolupament ha estat mitjançant TDD. El “backend” s’ha desenvolupat utilitzant el “framework” d’”Spring” i el patró ECS per tal d’optimitzar al màxim el rendiment del joc. El “frontend” s’ha realitzat amb “React” i “Redux” per a poder gestionar i centralitzar l’estat de l’aplicació i, finalment, els gràfics s’han generat mitjançant SVG.

Índex

Índex de Figures

Glossari de Termes

Frontend: part del software que interactua amb l'usuari.

Backend: part del software que s'encarrega de processar i guardar les dades.

Maven: eina utilitzada per a la gestió i la construcció del software.

Crossplay: Funcionalitat que permet jugar al joc entre diferents plataformes.

Refactorització: Canvis en l'estructura interna del software per optimitzar-lo sense canviar el seu comportament.

Coverage: mètode d'anàlisi que determina quines parts han estat cobertes pels tests.

Framework: És una estructura conceptual i tecnològica amb artefactes i mòduls que serveixen de base per a l'organització i desenvolupament de software.

Boilerplate: és una secció de codi que es repeteix en diversos llocs sense pràcticament cap variació.

1 Introducció

L'ús de les tecnologies de la informació i la comunicació, més concretament internet s'ha incrementat de manera excepcional degut a la de pandèmia del Covid-19. Les organitzacions i els individus han accelerat el procés de digitalització, introduint nous canvis de comportament, com per exemple el teletreball, que probablement ja s'establiran permanentment en la societat.

Un dels principals afavorits durant la pandèmia han estat els jocs de taula, que durant el primer any de pandèmia van augmentar les vendes un 18,3%. A més, el CEO de Zacatrus (empresa que es dedica a vendre jocs de taula), Sergio Viteri, recalca que normalment el públic de jocs de taula són majoritàriament adults i famílies amb nens. Però assegura que el tipus de públic interessat s'ha obert en un mercat més ampli. Per tant, aquest pot ser un bon moment per digitalitzar un joc de taula. [1]

Un altre motiu és una possible eina per combatre l'estrès. Una de les millors maneres per rebaixar l'estrès i poder desconnectar una estona és poder fer una partida de qualsevol joc de taula amb amics o familiars. Existeixen moments i situacions en les que els usuaris no els és possible jugar a un joc de taula amb els seus propers per qüestions d'horaris o de distància.

Aquest TFG no pretén ser un substitut dels jocs de taula sinó una alternativa, el que es busca és que els usuaris, es trobin en la situació que es trobin, no es quedin sense aquest moment de plaer del dia i disposin de l'oportunitat de poder gaudir igual que si estiguessin jugant físicament.

El projecte s'ha dut a terme amb la premissa de que ha de ser un joc web i fent ús de les eines i tècniques apreses durant el grau.

El resultat final que s'ha assolit és una "demo" del joc en línia dels colonitzadors de l'illa. On els usuaris poden registrar-se i iniciar sessió, consultar el blog amb l'historial de totes les funcionalitats del projecte, crear una partida o unir-se a partides d'altres usuaris i jugar les partides que desitgin.

El desenvolupament ha estat cíclic i dividit per les funcionalitats del mateix. Al haver-se aplicat TDD, primer s'ha implementat el test de la nova funcionalitat en forma de blog, en segon lloc el "backend" i finalment el "frontend". Seguint aquest mateix procés per cadascuna de les noves funcionalitats que s'han afegit.

Durant el desenvolupament del projecte han anat sorgint problemes i imprevistos que han afectat al temps de desenvolupament i conseqüentment lleugerament la qualitat final del producte. Alguns d'aquets errors han estat provocats per Maven, altres per problemes de compatibilitat entre CSS i "React". També la creació del mapa ha estat més complexa del que s'esperava, i degut al patró ECS, han aparegut dificultats en realitzar accions trivials (com ara recuperar un atribut d'un objecte). A més, els tests i el "coverage" han alentit considerablement el desenvolupament del project

2 Marc Teòric i Anàlisi de Referents

A continuació es realitza l'estudi previ del TFG per entendre com sorgeix i com es durà a terme.

2.1 Context

Klauss Teuber, nascut a Alemanya, és el creador de Catan. Els jocs de taula no van jugar un paper important durant la seva infància fins que li van regalar el joc "Romans vs. Carthaginians" quan tenia onze anys. El joc el fascinava tant que ideava noves normes i regles del joc i les posava en pràctica. Però aquesta afició va quedar apartada durant anys.

Després de fer el servei militar va estudiar química i es va incorporar al laboratori dental del seu pare per raons familiars. Klauss assegura que aquesta època va ser molt estressant i, per aquest motiu, va començar a idear jocs de taula, per evadir-se una estona de la realitat.

Al 1988, va crear el seu primer joc de taula "Barbarossa und die Rätselmeister" i va ser anomenat joc del any. Al 1990 va crear el joc anomenat Hoity Toity i al 1991 el Drunter and Drüber. Finalment, al 1995 va crear el Catan, que és el seu joc més exitós amb milions de vendes arreu del món i traduït a més de trenta idiomes. Degut aquest èxit ha creat varies expansions del joc que mantenen l'essència del joc base però incorporen noves funcionalitats i objectius. També, ha seguit creant altres jocs de taula però s'han vist eclipsats pel seu joc estrella. [2]

2.1.1 Catan: El Joc

El joc base de Catan es juga amb tres o quatre jugadors que arriben a una illa despoblada plena de matèries primes per explotar. Aquests jugadors han d'intentar colonitzar-la mitjançant construccions com ara pobles, ciutats o carreteres i explotar els recursos que posseeixin les terres del voltant dels seus pobles o ciutats. L'illa de Catan consta de cinc recursos diferents: l'argila, la fusta, l'ovella, el cereal i el mineral, que són representats per caselles hexagonals. A cada casella se li assigna un número que posteriorment, quan es llencin els daus, produiran els recursos que coincideixin amb el número.

Figura 1. Recursos. Font: Elaboració pròpia.

Els jugadors construeixen poblats o ciutats als vèrtex d'aquests hexàgons guanyant accés als recursos. Les construccions costen matèries primes, i al començament de la partida, és molt poc probable que un jugador disposi de les matèries primes de cadascun dels tipus per poder ser autosuficient. Per tant, es veurà obligat a comerciar i arribar a acords amb altres jugadors o la banca per intercanviar matèries.

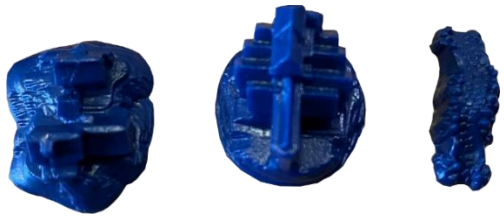


Figura 2. Estructures. Font: Elaboració pròpia.

També, hi ha la figura del lladre. Aquest comença al desert (és l'únic terreny no fèrtil del tot el taulell). Quan es llencen els daus, si toca el número set, tots els jugadors han de comprovar que no posseeixin més de set cartes a la mà i, si es dóna el cas, el jugador en qüestió ha de descartar la meitat de les seves cartes. En cas que el número de cartes sigui senar, s'arrodoneix a favor del jugador afectat. A més, és obligatori moure el lladre. El jugador que hagi tirat els daus mou el lladre a una casella diferent de la que es troba, deixant estèril el terreny del nou hexàgon. També, roba una carta aleatòria a un jugador posseïdor del terreny.

Figura 3. Lladre. Font: Elaboració pròpia.

L'objectiu del joc és aconseguir deu punts de victòria. Existeixen vàries maneres de guanyar aquests punts. Cada poblat, per exemple, atorga un punt de victòria. Cada ciutat atorga dos punts de victòria. El camí més llarg proporciona dos punts de victòria addicionals. El jugador també pot provar sort i invertir en cartes de desenvolupament on s'obtenen avantatges aleatòries. Una d'aquestes avantatges són els cavallers. El jugador amb l'exèrcit de cavallers més gran, també disposarà de dos punts de victòria addicionals. En les cartes de desenvolupament també es poden trobar cartes de punts de victòria addicionals. La resta d'avantatges de les cartes de desenvolupament ajuden al jugador a desenvolupar-se però no atorguen punts directament.

En general, el joc es basa en economia i probabilitats. La aleatorietat dels daus determinen l'ingrés de recursos, mentre que els subministraments i la demanda determinen el valor dels recursos.

Hi ha múltiples factors que converteixen Catan en un joc més estratègic que de sort, com ara les múltiples diferents maneres de guanyar punts. El fet que el taulell sigui construït de manera aleatòria fa que canviï cada partida, així com la competència per la colonització dels terrenys envers la resta de jugadors. [3]

Figura 4. Taulell. Font: Imatge Extreta de [4]

2.1.2 Catan: Ciutats i Cavallers

Una de les expansions més exitoses és el Catan: Ciutats i Cavallers a causa de la gran varietat de funcionalitats i mecàniques noves que afegeix. En aquesta expansió, trobem uns bàrbars que es van acostant a l'illa i quan arriben l'ataquen. Els jugadors, per poder-se defensar, disposen de cavallers.

Existeixen tres nivells de cavallers. Els cavallers poden estar actius o no, si estan actius poden realitzar accions com ara moure's, expulsar el lladre o expulsar un altre cavaller enemic, però la seva principal funció és participar i ajudar a guanyar la guerra contra els bàrbars.

Figura 5. Cavaller. Font: Elaboració pròpia.

Durant l'atac dels bàrbars, començant pel jugador que té el torn i en sentit de les agulles del rellotge, es decidirà qui participa o no a la guerra i amb quants cavallers hi contribuirà. El resultat de la guerra es calcula de la següent manera: els bàrbars posseeixen una força igual al número de ciutats que hi ha al taulell independentment de qui en sigui el propietari. I la força dels cavallers que defensen l'illa es calcula sumant totes les puntes dels estendards de cada cavaller. Si els bàrbars guanyen la guerra, el jugador que hagi contribuït menys a la guerra perdrà una ciutat, per contra, el jugador que hi hagi contribuït més guanyarà un punt de victòria. En cas de que dos jugadors hagin aportat el mateix nombre de puntes, en comptes de guanyar un punt de victòria aconseguiran ambdós una carta de desenvolupament.

Una altra gran diferència respecte del joc bàsic, són les mercaderies. Aquestes serveixen per a comprar millores de ciutat. En trobem de tres tipus: la llana, les monedes i el paper, i cada una va associada al tipus de millora de comerç, política i ciència respectivament. Quan es compra una millora d'un tipus és més probable aconseguir cartes de desenvolupament d'aquell tipus. En el moment en què s'arriba a la millora cinc d'un mateix tipus es converteix una de les ciutats del jugador en una metròpolis (i ja no es pot perdre encara que un altre jugador hi arribi després) atorgant dos punts de victòria extra.

Al haver-hi tres tipus de cartes més que en el joc bàsic és senzill acumular cartes fent que el lladre sigui molt perillós. Perquè els jugadors puguin fer front al lladre se'ls facilita la possibilitat de poder construir muralles sota les ciutats augmentant en dos la capacitat màxima de cartes, que pot posseir un jugador a la mà abans de ser robat pel lladre. No obstant, no es poden construir muralles als pobles. [5]

Figura 6. Matèries Primeres. Font: Elaboració pròpia.

2.1.3 Catan: Pirates i Exploradors

En aquesta expansió s'afegeix la mecànica d'exploració, que no existia amb els altres. Existeix, també, una illa inicial on els jugadors comencen a desenvolupar-se , però, el taulell és més gran introduint mar i noves illes per explorar.

L'expansió disposa de peces noves que compliran funcionalitats, també, noves. Alguns d'aquestes són les ciutats portuàries, que són les ciutats del joc bàsic amb les particularitats de que només es poden construir a la costa i poden emmagatzemar una unitat de colons. Els colons és la unitat que servirà per poder fundar nous poblats a les illes inexplorades. Per desplaçar-se hauran d'embarcar a un vaixell.

Figura 8. Port. Font: Elaboració pròpia.

Figura 7. Colon. Font: Elaboració pròpia.

El funcionament de l'exploració és el següent: cada torn els vaixells es poden desplaçar quatre caselles (els vaixells poden anar sense colons però no podran fundar cap poblat nou). En el moment en que un vaixell topa amb una terra inexplorada, es girarà exhibint el no tipus de terreny (pots ser qualsevol tipus de recurs o mar), aquest ja no es podrà desplaçar més fins el següent torn. A més el jugador cobrarà un de recurs del que hagi descobert, si és mar cobrarà dues monedes.

Figura 9. Vaixell. Font: Elaboració pròpia.

Existeixen dues maneres de fundar nous poblats. El clàssic, que funciona igual que la resta de expansions, servirà per expandir-se per la illa inicial. La nova manera que incorpora és la següent: primer s'embarca el colon a una barca, seguidament es desplaça la barca, quan arriba a una de les illes inexplorades pot decidir si seguir explorant la illa o intentar fundar un poblat. En el moment que vol fundar el poblat la barca ha d'estar aparcada a la costa i podrà col·locar un poblat o a la cruïlla de davant de la barca o la del darrere. Un cop establert el poblat el jugador perdrà la barca i el colon però, podrà començar-se expandir de la manera clàssica si ho desitja.

A més l'expansió de pirates i exploradors incorpora missions com ara conquistar les guarides pirates, aconseguir peix per entregar-lo al Consell de Catan o comerciar amb misteriosos habitants de les illes de les espècies. És poden implementar a la mateixa partida o per separat creant una gran varietat d'estratègies possibles. [6]

2.1.4 Catan: Duels

El Catan Duels és una versió per dues persones. Manté la filosofia del joc bàsic però canvia dràsticament la manera en què es juga. No hi ha taulell. El territori s'ha de construir mitjançant cartes. En el joc bàsic (Catan "El Joc") existeixen pobles, ciutats i carreteres, en aquest cas també existiran, però en forma de cartes. També ho seran tots els territoris de recursos que conformaven el taulell de l'original.

Figura 10. Distribució de les cartes Font: Imatge extreta de [7]

En aquesta imatge es pot observar com seria la distribució inicial de la partida. Els dos pobles es troben separats per una carretera i a cada cantonada dels pobles es col·loquen els recursos. Per saber la quantitat de recursos de que disposa el jugador, cal observar la part inferior de cada carta de recurs. Inicialment es comença amb un recurs de cada, menys l'or que comença a zero. Per poder emmagatzemar recursos, es llença el dau, la carta de recurs amb el número coincident es girarà en sentit contrari a les agulles del rellotge, deixant dos símbols d'aquell recurs. Per tant, com es pot observar, no es poden emmagatzemar més de tres recursos per carta. Si se n'obtenen més de 3 es perden.

Els espais buits, que queden a sota i a dalt de cada poble, serviran per efectuar-hi millores i construccions. Quan el poble evoluciona a ciutat desbloqueja dos espais més per seguir-hi construint.

Aquest Joc també disposa d'un dau d'esdeveniments que no existia en l'original. Cada esdeveniment planteja una situació si s'esdevenen certes condicions, si un jugador no les acompleix normalment surt perjudicat si l'altre sí les compleix. [8]

2.2 Antecedents

2.2.1 Catan Universe

És la competència directe més forta i treballada ja que planteja el joc en diferents plataformes, com ara android, iOs, mac i Windows. Amb el sistema de "crossplay" permetent jugar als jugadors d'una plataforma amb la resta, per exemple, els d'Android amb els de Windows. El joc té una versió "Free to Play" però és molt limitada ja que només deixa jugar una partida online gratuïta al dia. Fet, que pots ser contraproduent a l'hora d'aconseguir que l'usuari acabi comprant la totalitat del joc base o les expansions ja que amb una partida al dia és poc probable que l'usuari s'acabi enganxant al joc i, conseqüentment, difícilment comprarà alguna versió ampliada.

Els gràfics del joc són correctes i les mecàniques es troben ben implementades. No obstant, li manquen expansions que ja porten anys creades al joc de taula original. Tampoc ofereixen la possibilitat de personalitzar elements del joc (és l'aspecte que més diners aporta als jocs "Free to play").

Posseeixin, també un perfil a les xarxes socials més utilitzades (Facebook, Instagram i Twitter) però amb manca de contingut només s'utilitza per notificar actualitzacions. Entre "Post" i "Post" poden passar setmanes o mesos. [9]

2.2.2 Catan Classic

És un joc per Android i per iOs, no és "Free to Play", el joc té un cost de 4.99€ tant a Android com a iOS. Els gràfics són correctes i les mecàniques no s'han pogut testejar. Igual que en l'anterior, consta de vàries expansions (però no totes) que es poden comprar dins del joc, també, amb diners reals que varien entre els 1.99€ als 9.99€.

És necessari remarcar, que aquest joc té moltes ressenyes negatives degut el mal funcionament del joc online entre d'altres. Tot i així, acumula més de cinc-centes mil descàrregues només a la tenda de Google Play. [10]

2.2.3 Colonists

És un joc web que es troba en diverses plataformes com ara la de “minijuegos” o la de “silvergames.com”, el joc base és “Free to Play”, no té cap tipus de limitació de partides gratuïtes al dia. Els gràfics i les mecàniques són correctes.

Aquest, també t’ofereix les mateixes expansions que els altres però n'afegeix de noves per augmentar la capacitat màxima de jugadors per partida. El preu de cada expansió és aproximadament dels $20. A més, de la competència, aquest és el que treu més profit a les microtransaccions, a part de les expansions també venen noves distribucions de mapes (noves formes) per aproximadament quatre dolars, nous colors per les fitxes de joc que varien d’entre els zero dolars als trenta-cinc dolars i també ofereixen avatars que oscil·len entre els zero i aproximadament els dos dolars.

Una altre punt que els diferencia de la resta és que el sistema del joc és capaç de emparellar els jugadors per habilitat fent que l’experiència de joc sigui més agradable. [11]

2.3 Necessitats d'informació

2.3.1 TDD

El TDD (Test Driven Development) és un procés de software que consisteix en realitzar petites iteracions d'un cicle. Aquest cicle està compost per diferents etapes.

La primera, és la creació del test. En aquesta etapa s'expressa la funcionalitat que es vol implementar en forma de test. Un cop acabat, s'executa el test per comprovar que falla.

La segona, és la implementació del test. En aquesta etapa es programa el codi mínim perquè passi el test.

La tercera, és la "refactorització" del codi. En aquesta etapa es netejarà el codi efectuat en l'etapa anterior. Alguns exemples de refactorització poden ser: eliminar codi duplicat, canviar noms de variables perquè siguin més entenedores, dividir mètodes en trossos més petits per tal de simplificar-los, etc. En aquesta etapa, també s'ha d'aconseguir el 100% del "coverage".

Si el TDD s'aplica correctament s'aconsegueix un codi testejat, net i mantenidor a llarg termini. [12, 13, 14]

2.3.2 Spring

S'usa el "framework" "Spring" per desenvolupar tot el "backend" del projecte. Aquest "framework" serveix per a facilitar i simplificar la construcció d'aplicacions en Java. El gran avantatge que et proporciona l'"Spring" és la injecció de dependències, afavorint el baix acoblament entre classes i proporcionant modularitat i escalabilitat sense necessitat de modificar les classes. També minimitza el codi "boilerplate" (codi repetitiu), facilita l'accés a les dades i permet integrar moltes altres tecnologies de manera senzilla. [15, 16]

2.3.3 React

S'usa el "framework" de Javascript anomenat "React" per desenvolupar tot el "frontend" del projecte. "React" ens ajuda a construir interfícies d'usuari amb dades que canvien amb el pas del temps, fent que aquestes siguin escalables i mantenidores a llarg termini. [17]

2.3.4 Redux

S'usa la llibreria de "Redux" per controlar els estats de l'aplicació al "frontend". Va ser creada degut a l'increment de la complexitat de les aplicacions desenvolupades amb JavaScript. "Redux" es regeix per tres principis fonamentals:

- **Única font de la veritat**: L'estat global de l'aplicació s'emmagatzema en un arbre d'objectes dins d'un sol magatzem.
- **L'estat és només de lectura**: L'única manera possible de canviar l'estat és realitzant una acció (un objecte que descriurà què ha passat).
- **Els canvis són efectuats per funcions pures**: per especificar com és transformat l'estat de l'arbre amb accions, és necessari escriure "reducers" (o reductors) purs.

Per entendre "Redux" s'han de conèixer els següent conceptes:

- **"Actions" (o accions)**: són esdeveniments normalment generats degut a la interacció de l'usuari amb l'aplicació, enviats al magatzem.
- **"Reducers" (o reductors)**: són funcions que agafen un estat i una acció i retornen una acció resultant.
- **"Store" (o magatzem)**: és l'objecte que emmagatzema els estats de l'aplicació i proporciona alguns mètodes d'ajuda per accedir a l'estat, enviar accions i registrar escoltadors. [18, 19]

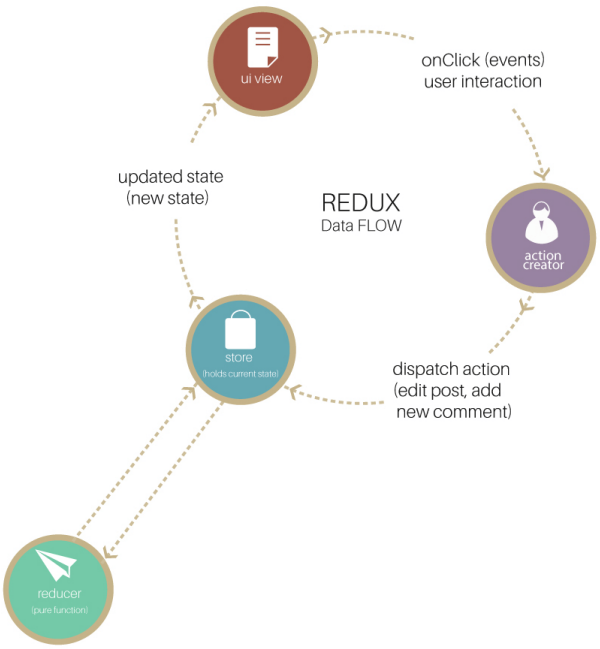


Figura 11. Esquema del funcionament de "Redux". Font: Imatge extreta de [20]

Aquesta imatge ajuda a entendre quin és el procés d'execució que segueix "Redux".

2.3.5 Selectors

És molt comú que les aplicacions que fan servir "Redux" recuperin les dades a través de funcions que s'anomenen selectors. Un selector no és res més que una funció que accepta com a paràmetre l'estat del magatzem de "Redux" i retorna dades basades amb l'estat.

Els selectors, normalment, es defineixen en un fitxer apart però dins de la lògica de "Redux" perquè puguin accedir a les dades. També, es poden definir dins els fitxers de components. [21]

2.3.6 ECS

Pel desenvolupament del projecte també s'usa el patró de ECS (Entity Component System), ja que afavoreix la reutilització de codi separant les dades del comportament. Aquest patró conté entitats que són identificadors únics, components que són tipus de dades simples, les entitats poden contenir zero o més components i els components de les entitats poden canviar dinàmicament.

A més, ECS facilita la incorporació de noves funcionalitats i és més eficient que la programació orientada a objectes convencional.

És important remarcar que, tot i que aquest patró és majoritàriament usat en el món dels videojocs, també pot ser utilitzat en altres àmbits. [22]

2.3.7 Middleware

"Redux", per si sol, no contempla l'asincronisme. En aquest projecte, s'usa el patró Middleware, principalment per facilitar un punt entre l'enviament d'una acció i el moment que arriba al "reducer". És en aquest punt on s'introduirà l'asincronisme, d'aquesta manera es podran fer peticions a l'Api del "backend" en cas que sigui necessari. També, es podria utilitzar per portar un registre de les accions que es van realitzant, informes d'errors, "routing", etc. [23]

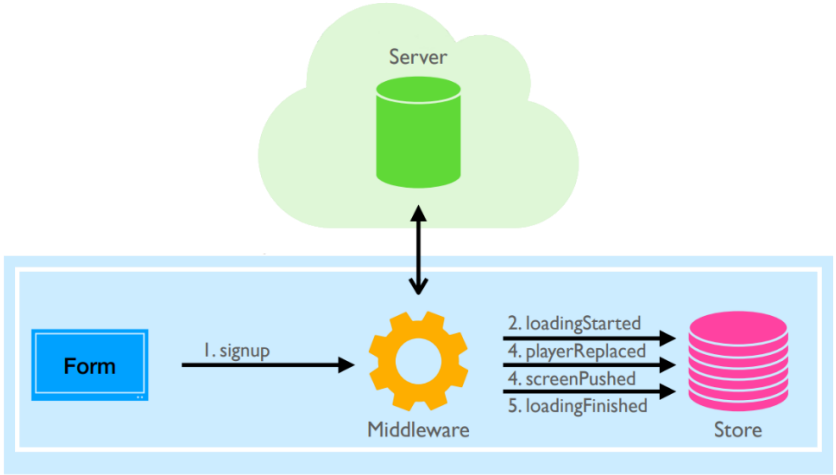


Figura 12. Exemple Middleware. Font: Imatge extreta de [24]

3 Objectius i Abast

- Crear una “demo” de partida ràpida que sigui jugable per mitjans de juny.
- Disposar d’un blog a l’aplicació per poder fer un seguiment de les funcionalitats que es van implementant abans d’acabar abril.
- Implementar la creació de partida amb els jugadors abans d’acabar abril.

En aquest projecte es desenvoluparà una “demo” del joc dels colonitzadors de l’illa. És indispensable dividir l’objectiu principal en diferents tasques per tal de poder-lo gestionar més efectiva i eficientment. Es proposen les següents tasques que es duran terme a partir de mitjans de febrer fins a mitjans de juny:

Objectius primaris:

- Creació de projecte (inclou tot el software i totes les dependències perquè el projecte funcioni).
- Creació de blog.
- Creació de partida.
- Implementació dels elements bàsics del joc.
- Inicialització dels jugadors.
- Construcció del terreny.
- Implementació del flux de rondes.
- Implementació de la compra dels elements.
- Gestió i assignació de punts.

Objectius secundaris:

- Implementació cartes de desenvolupament.
- Implementació de comerç entre la màquina i entre els jugadors.

És necessari mencionar que queda fora de l’abast del projecte la implementació de les expansions, la personalització d’elements del joc, les estadístiques i classificació dels jugadors.

En aquest projecte coincideix que el client potencial també és l'usuari final. Les funcionalitats que disposarà l'usuari en acabar el desenvolupament al mes de juny seran:

- Registrar-se i iniciar sessió.
- Crear partida.
- Buscar partides d'amics/familiars i poder-s'hi unir.
- Jugar la partida.
- Consultar el blog de les funcionalitats del joc.

4 Metodologia

Per a dur a terme aquest projecte es decideix aplicar la metodologia en espiral. Consta de quatre grans etapes:

A. Determinar objectius i requeriments:

Es plantegen els objectius i requeriments que es desenvoluparan durant la imminent iteració. Per a què aquesta metodologia sigui efectiva s'ha de complir que els requeriments es coneguin abans de començar l'etapa de desenvolupament. Tots els requeriments han d'estar reflectits en el pla de riscos. La natura dels requeriments no canviarà durant l'etapa de desenvolupament. Els requeriments han de ser compatibles amb tots els actors implicats amb el projecte, com poden ser: clients, usuaris, sponsors, inversors, desenvolupadors, etc. L'arquitectura adequada per implementar els requeriments ha d'estar ben entesa. S'ha de disposar del temps necessari per procedir seqüencialment.

B. Planificació i Gestió de riscos:

Els riscos són possibles esdeveniments que poden provocar que no s'arribi als objectius plantejats. En aquesta etapa s'enumeren els riscos i es prioritzen segons l'impacte i la probabilitat de que apareguin. Per cada risc, es planeja un pla de mitigació per reduir el cost o l'impacte del mateix.

C. Desenvolupament i implementació:

Es desenvolupen els requeriments plantejats en aquesta iteració i es sotmeten a un control de qualitat. Un cop validats, es realitza la implementació.

D. Planificació:

Es realitza la planificació de la següent iteració, sempre tenint en compte com ha anat l'anterior.

Les etapes explicades es realitzen de manera iterativa. Es comença mirant les possibles alternatives de desenvolupament i s'opta per la de risc més assumible o de menor risc i s'esdevé un cicle d'espiral. [25, 26]

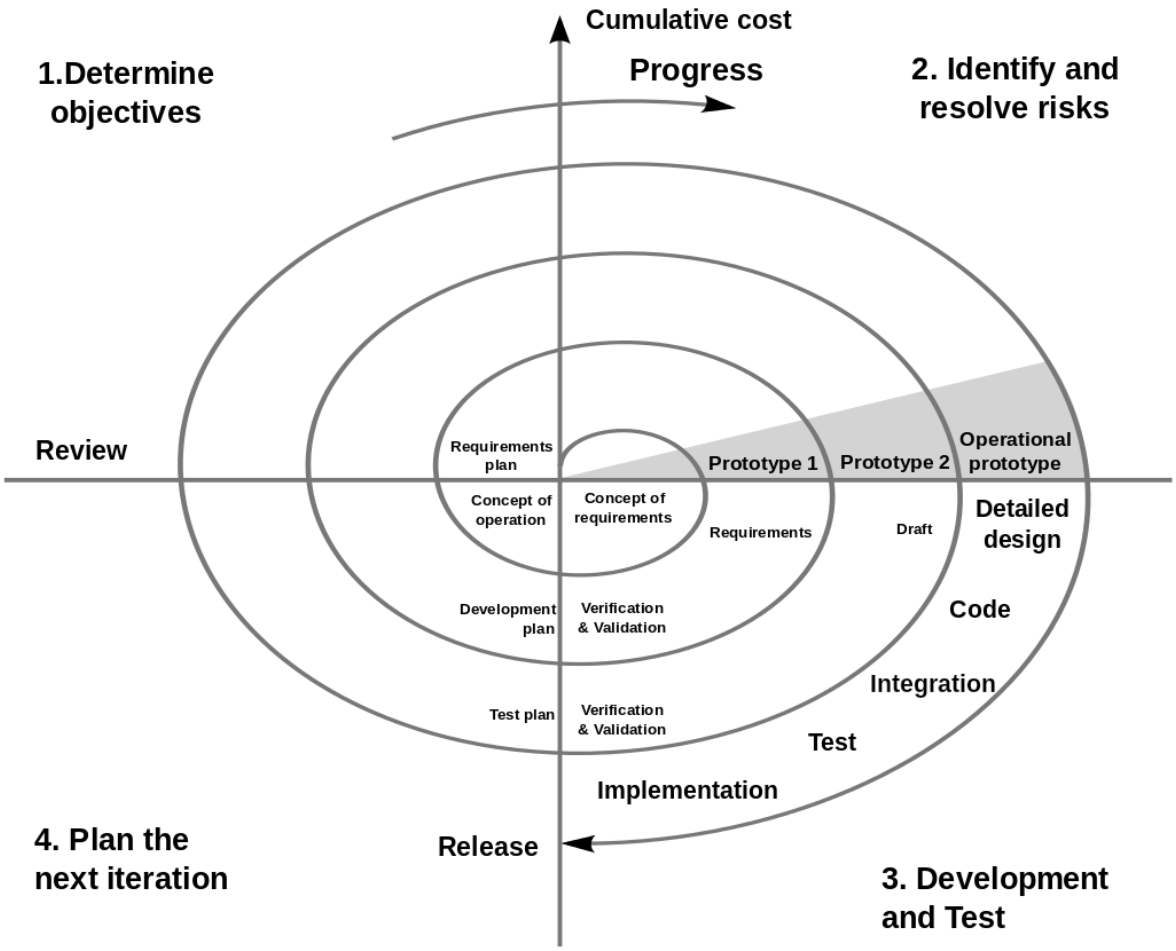


Figura 13. Metodologia en espiral. Font: Imatge extreta de [27]

5 Desenvolupament

5.1 Requeriments

Definició de requeriments funcionals i tecnològics

Es plantegen els següents requeriments funcionals:

- Crear una partida ràpida.
- Jugar de manera cooperativa en línia.
- Entrar en una partida amb un nom personalitzat.
- Mostrar, en forma de blog, les funcionalitats que es van implementant.

Un cop dins la partida:

Elements indispensables

- Proveir els recursos de tipus : fusta, argila, cereals i ferro.
- Proveir poblats, ciutats i carreteres.
- Proveir cartes de desenvolupament proporcionant un avantatge aleatori.

Punts de victòria

- Atorgar un punt de victòria per cada poblat col·locat pel jugador.
- Atorgar dos punts de victòria per cada ciutat col·locada pel jugador.
- Atorgar dos punts de victòria pel jugador amb la carretera més llarga.
- Atorgar dos punts de victòria pel jugador que posseeixi l'exèrcit més gran.
- Atorgar la victòria al primer jugador que arribi als deu punts de victòria.

Creació del taulell

- Contenir caselles dins d'un taulell.
    - Posseir un pes assignat.
    - Posseir un recurs assignat.

Estat inicial dels jugadors

- Posseir quatre ciutats.
- Posseir cinc poblats.
- Posseir quinze carreteres.

Flux i elements del joc

- Administrar torns.
- Poder llençar dos daus.
- Poder col·locar poblats o ciutats als vèrtex de les caselles.
- Poder col·locar carreteres a les arestes de les caselles.
- Poder emmagatzemar recursos de cada tipus.
- Poder comprar poblats, ciutats, carreteres i cartes de desenvolupament.
- Poder intercanviar recursos amb la banca.
- Poder intercanviar recursos amb altres jugadors.

Es plantegen els següents requeriments tecnològics:

- Mostrar la interfície d'usuari a un navegador.
- Comunicar jugadors mitjançant un servidor que gestioni diferents jugadors.
- Ha d'existir un conjunt de tests segurs que ajudin a crear el codi i refactoritzar-lo.
- Usar un gestor de control de versions.

5.2 Funcionalitats implementades

Figura 14. Funcionalitats. Font: Elaboració pròpia.

Aquestes són les funcionalitats que s'han implementat durant el desenvolupament del projecte.

5.3 Eines / Tecnologies

5.3.1 API REST

Per a la comunicació client / servidor s'ha utilitzat l'arquitectura d'API REST. El "frontend" disposa d'una capa "Middleware" on pot fer peticions HTTP de tipus GET, PUT, POST i DELETE. Totes i cadascuna de les peticions les rep la capa de l' API del "backend", on són processades i se'ls hi retorna una resposta. Tot seguit, el client s'encarrega de formatar la resposta i col·locar-la a la pantalla.

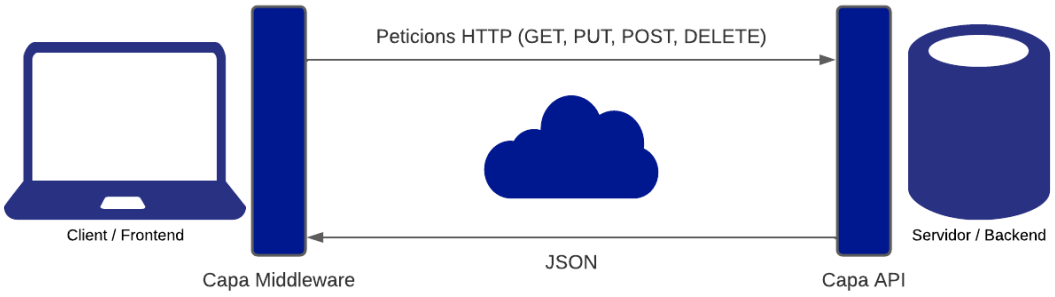


Figura 15. Api Rest. Font: Elaboració pròpia.

5.3.2 Desplegament

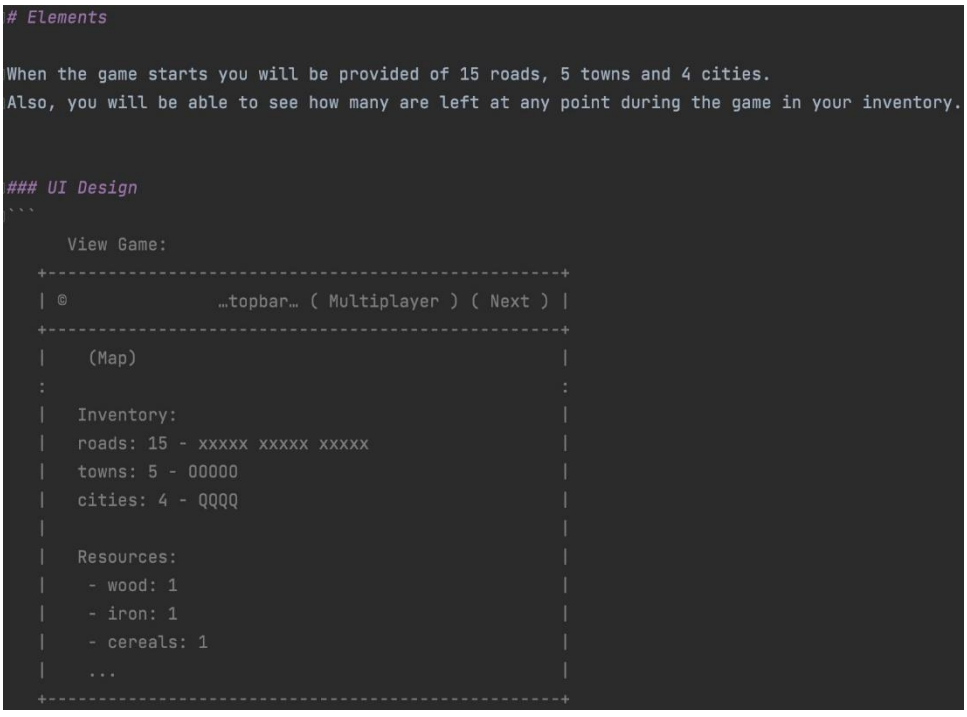
S'ha realitzat el desplegament de l'aplicació amb Heroku amb l'ajuda d'un "script", anomenat deploy.sh, que s'ubica a l'arrel del projecte. Heroku és una PaaS que et permet, de manera molt senzilla, i en la majoria dels casos, desplegar gratuïtament una aplicació web.

L'enllaç de l'aplicació és: https://colonitzadors-illa.herokuapp.com/

5.3.3 TDD

Durant el desenvolupament d'aquest projecte s'han realitzat tests en forma de blog que pot llegir l'usuari final. Es genera un test per cada funcionalitat nova que es vol implementar.

Per crear un nou test es genera un fitxer amb l'extensió ".md". En aquest fitxer, primer, es realitza una breu descripció de la funcionalitat acompanyada d'un esquema de com, aquesta, es veuria a la pantalla un cop implementada. Tot seguit, es realitza una seqüència d'instruccions explicant què és el que hauria de fer l'usuari i el que hauria de veure en tot moment. Al mateix temps, aquestes instruccions són els tests que s'executen tant al "frontend" com al "backend". Si totes les instruccions passen els tests associats a cadascuna d'elles, i suposant que la seqüència d'instruccions sigui correcte, la funcionalitat es podrà donar per acabada. És necessari recalcar que la descripció de la funcionalitat ha de ser divertida, amb la intenció de vendre-la a l'usuari final. A més, les instruccions s'escriuen sense cap tecnicisme, ja que l'usuari final ha de poder-les entendre i no ha de perquè ser un expert.



```
# Elements

When the game starts you will be provided of 15 roads, 5 towns and 4 cities.
Also, you will be able to see how many are left at any point during the game in your inventory.


### UI Design
```
      View Game:
   +--+
   | ©   …topbar… (Multiplayer) (Next) |
   +--+
   | (Map) |
   :   :
   | Inventory: |
   | roads: 15 - xxxxx xxxxx xxxxx |
   | towns: 5 - OOOOO |
   | cities: 4 - QQQQ |
   | |
   | Resources: |
   | - wood: 1 |
   | - iron: 1 |
   | - cereals: 1 |
   | ... |
   +--+
```

Figura 16. Tdd: fitxer .md 1. Font: Elaboració pròpia.
```

```
## See elements

When the game starts you will be provided of 15 roads, 5 towns and 4 cities.

### See roads

When the game starts, each player gets some resources in their planets.

 * Given there is "leonard" playing their game "together".
 <!-- SNAPSHOT status=200 -->
 * "leonard" should be the current player.
 * Game round should be 1.
 * "leonard" should have 1 inventory.
 * There should be 15 roads in the inventory.

### See towns
 * There should be 5 towns in the inventory.

### See cities
 * There should be 4 cities in the inventory.
```

Figura 17. Tdd: fitxer .md 2. Font: Elaboració pròpia.

A les figures 16 i 17 es mostra un exemple de com s'ha desenvolupat el blog/test. Concretament és l'exemple dels elements de què disposa un jugador al seu inventari.

Com es pot observar a la figura 16, es realitza una breu descripció o explicació de la nova funcionalitat, acompanyada d'un petit esquema o dibuix de com s'hauria de veure aproximadament el resultat final.

Tot seguit, com es mostra a la figura 17, s'escriuen un conjunt d'instruccions, en forma de llista sense tecnicismes, de com interactuaria l'usuari amb aquesta nova funcionalitat.

Podem considerar que cada apartat del blog és un cas d'ús que ajuda a entendre, tant a l'usuari com al desenvolupador, què i com se suposa que l'usuari ha d'interactuar amb aquella nova funcionalitat.

```java
@Override
protected String getRegex() { return "There should be (\\d+) ([a-z ]+) in the inventory"; }

@Override
protected void run(PostLine line, String[] match) {
    var entityPropertyKey : String = PrettyKey.getKey(match[2]);

    mayRunWithNumber(entityPropertyKey, match[1]);
}

private void mayRunWithNumber(String key, String match) {
    if (match == null) return;
    var expectedValue : int = Integer.parseInt(match);

    String inventoryId = inventoryTestView.getInventoryId();

    var value : int = entityTestView.getEntityPropertyInt(inventoryId, key);
    assertThat(value).isEqualTo(expectedValue);
}
```

Figura 18. Tdd: Test. Font: Elaboració pròpia.

En aquest cas, a la figura 18, s'ha escollit una instrucció de la funcionalitat dels elements de joc, mostrant què passa quan s'executa la instrucció. Primer de tot, es fan servir expressions regulars perquè el test admeti el rang de números i qualsevol element de joc que ens interessi. Un cop executada la instrucció, es guarda l'element de joc a una variable i es deixa en un format amb el qual es pot treballar. Tot seguit es comprova que la cadena de resultats no sigui nul·la, es transforma a nombre sencer la cadena de text que conté el número d'elements que es necessiten. Es recupera la "id" de l'inventari del jugador. I Amb aquesta "id" i el nom de l'element es busca que realment aquell inventari posseeixi el que el test assegura que hauria de posseir.

És necessari assenyalar que perquè el codi de tests quedi net es treballa amb "TestViews", que el que fan és encapsular un conjunt de funcions que ataquen un tema similar. Com per exemple, l' "inventoryTestView", que conté funcions relacionades o que treballen amb l'inventari.

```java
public String getInventoryId(){
    return gameTestView.findEntity(byType("inventory")).get().getId();
}
```

Figura 19. Tdd: Get inventory Id. Font: Elaboració pròpia.

```
public Optional<EntityResponse> findEntity(Predicate<EntityResponse> predicate) {
    return streamEntities(predicate).findFirst();
}
```

Figura 20. Tdd: Find entity. Font: Elaboració pròpia.

A les figures 19 i 20 es mostra com es recupera la "id" de l'inventari. Com es pot observar, es recupera l'inventari especificant el tipus d'entitat que es vol buscar.

```
public int getEntityPropertyInt(String entityId, String key) { return getEntity(entityId).getInt(key); }
```

Figura 21. Tdd: Get entity property int. Font: Elaboració pròpia.

A la figura 21 es mostra com es recupera el valor d'un atribut de tipus "integer" d'una entitat. S'ha d'especificar la "id" de l'entitat i el nom d'aquell atribut.

5.3.4 Disseny de ECS

Tot el "backend" està basat en el patró ECS (Entity Component System), el qual redueix l'acoblament de classes i presenta una estructura modular. Aquest afavoreix i facilita la implementació de noves funcionalitats de les classes o entitats i la reutilització de codi. També, millora el rendiment en comparació amb un projecte amb una estructura de programació orientada a objectes convencional.

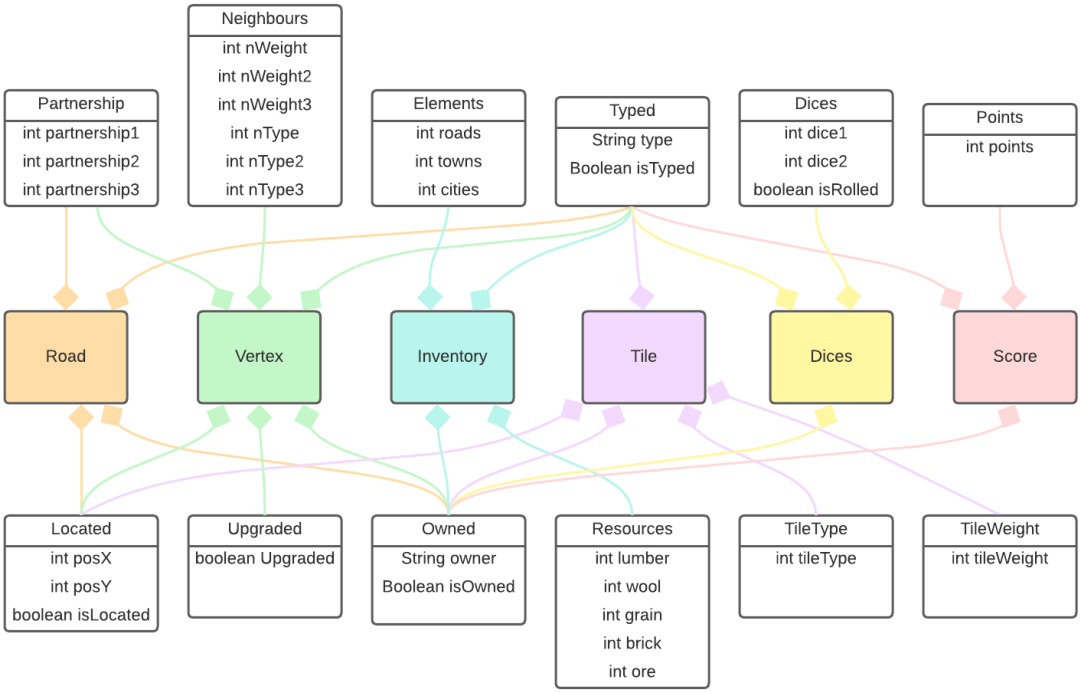


Figura 22. ECS: disseny d'entitats i components. Font: Elaboració pròpia

En aquesta imatge es mostra el diagrama d'entitats i components on s'ha aplicat ECS.

```java
@Component
public class InventoryFactory {

    private final EntityIdGenerator entityIdGenerator;
    private final OwnedsController ownedsController;
    private final TypedsController typedsController;
    private final ElementsController elementsController;
    private final ResourcesController resourcesController;

    public InventoryFactory(EntityIdGenerator entityIdGenerator,  OwnedsControl
        this.entityIdGenerator = entityIdGenerator;
        this.ownedsController = ownedsController;
        this.typedsController = typedsController;
        this.elementsController = elementsController;
        this.resourcesController = resourcesController;
    }

    public void buildInventory(Game game, Player owner) {
        var entityId : String  = entityIdGenerator.nextEntityId( hint: "inventory");
        ownedsController.create(entityId, game, owner);
        typedsController.create(entityId, game,  entityType: "inventory");
        elementsController.create(entityId,game);
        resourcesController.create(entityId,game);
    }
}
```

Figura 23. ECS: Inventory factory. Font: Elaboració pròpia.

A la figura 23 es pot observar la creació d'una entitat, concretament la de l'inventari. Com es pot apreciar, l'únic que pertany exclusivament a l'entitat inventari és la "id". La resta són els controladors de funcionalitats que ha de tenir l'inventari. En aquest cas es disposa del controlador de "owneds", "typeds", "elements" i "resources". "Owneds" permet guardar el propietari d'aquella entitat, "typeds" permet marcar el tipus d'entitat que és, "elements" i "resources" són les peces i els recursos de què disposa el jugador durant la partida.

```java
@Component
public class InventoryGameJoiner implements GameJoiner {

    private final com.tfg.game.entities.inventory.InventoryFactory inventoryFactory;

    public InventoryGameJoiner(com.tfg.game.entities.inventory.InventoryFactory inventoryFactory) {
        this.inventoryFactory = inventoryFactory;
    }

    @Override
    public void joinGame(Player owner, Game game) {
        inventoryFactory.buildInventory(game, owner);
    }
}
```

Figura 24. ECS: Inventory game joiner. Font: Elaboració pròpia.

Tota entitat disposa d'una classe que implementa el "GameJoiner" (figura 24). L'únic propòsit de la classe "GameJoiner" és permetre cridar totes les classes que implementin la interfície "GameJoiner" des d'un lloc comú. I cada classe, al mateix temps, fa una crida a la factoria perquè construeixi l'entitat, amb els seus components associats en el moment de generar la partida.

```java
@Entity
public class Typed extends EcsComponent {
    private String type;

    public Typed(String entityId, Game game, String type) {
        super(entityId, game);
        this.type = type;
    }

    protected Typed() {}

    public String getType() { return type; }

    public boolean isType(String type) { return this.type.equals(type); }
}
```

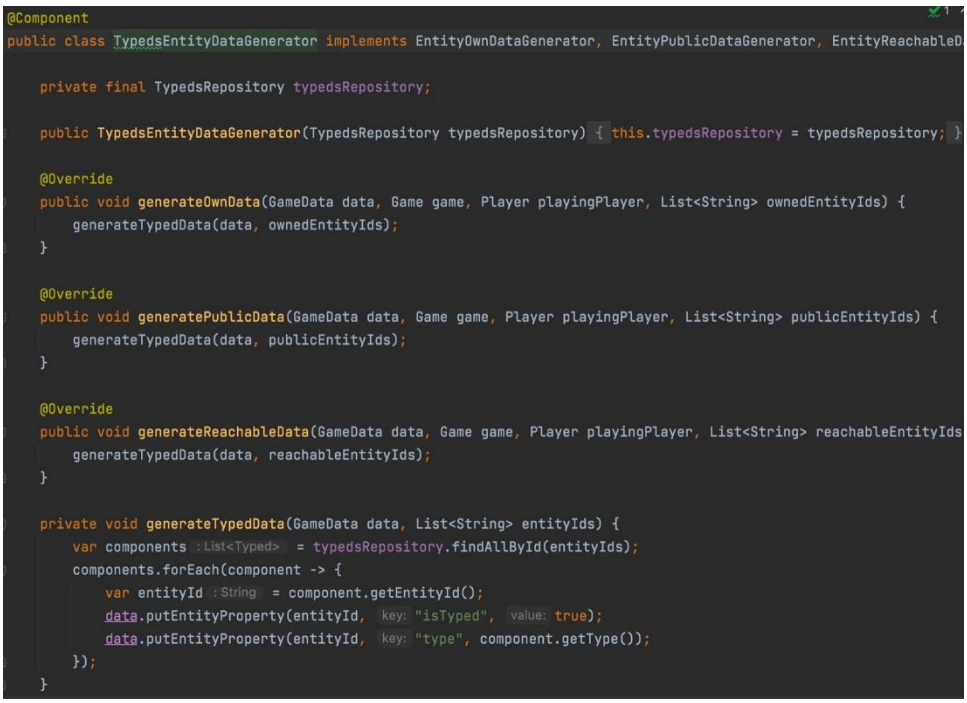
Figura 25. ECS: Typed. Font: Elaboració pròpia.

Per tal de mostrar com es troben estructurats els components de qualsevol entitat s'ha utilitzat el "typed" (figura 25) com a exemple. Tots els components disposen de la seva classe amb els atributs i tots els mètodes necessaris per a interaccionar amb ells.

```java
@Component
public class TypedsController {

    private final TypedsRepository typedsRepository;

    public TypedsController(TypedsRepository typedsRepository) { this.typedsRepository = typedsRepository; }

    public void create(String entityId, Game game, String entityType) {
        var component = new Typed(entityId, game, entityType);
        typedsRepository.save(component);
    }

    public boolean isTyped(String entityId, String entityType) {
        return typedsRepository.findById(entityId).map(c -> c.isType(entityType)).orElse( other: false);
    }

    public List<Typed> findAllByGameAndType(Game game, String type) {
        return typedsRepository.findAllByGameAndType(game, type);
    }

}
```

Figura 26. ECS: Typeds controller. Font: Elaboració pròpia.

Cada component també disposa del seu controlador (figura 26), que és qui s'encarrega de gestionar la persistència d'aquell component.



```java
@Component
public class TypedsEntityDataGenerator implements EntityOwnDataGenerator, EntityPublicDataGenerator, EntityReachableD

    private final TypedsRepository typedsRepository;

    public TypedsEntityDataGenerator(TypedsRepository typedsRepository) { this.typedsRepository = typedsRepository; }

    @Override
    public void generateOwnData(GameData data, Game game, Player playingPlayer, List<String> ownedEntityIds) {
        generateTypedData(data, ownedEntityIds);
    }

    @Override
    public void generatePublicData(GameData data, Game game, Player playingPlayer, List<String> publicEntityIds) {
        generateTypedData(data, publicEntityIds);
    }

    @Override
    public void generateReachableData(GameData data, Game game, Player playingPlayer, List<String> reachableEntityIds
        generateTypedData(data, reachableEntityIds);
    }

    private void generateTypedData(GameData data, List<String> entityIds) {
        var components :List<Typed>  = typedsRepository.findAllById(entityIds);
        components.forEach(component -> {
            var entityId :String  = component.getEntityId();
            data.putEntityProperty(entityId,  key: "isTyped",  value: true);
            data.putEntityProperty(entityId,  key: "type", component.getType());
        });
    }
```

Figura 27. ECS Entity data generator. Font: Elaboració pròpia.

Tots els components disposen del seu "EntityDataGenerator" (figura 27). Aquesta classe s'encarrega de adaptar les dades del "backend" i enviar-les al "frontend" de manera senzilla.

```java
@Component
public class TypedsGameDataGenerator implements GamePublicDataGenerator {

    private final TypedsRepository typedsRepository;
    private final List<EntityPublicDataGenerator> entityPublicDataGenerators;

    public TypedsGameDataGenerator(TypedsRepository typedsRepository, List<EntityPublicDataGenerator> entityPublicData
        this.typedsRepository = typedsRepository;
        this.entityPublicDataGenerators = entityPublicDataGenerators;
    }

    @Override
    public void generatePublicData(GameData data, Game game, Player playingPlayer) {
        var components : List<Typed> = typedsRepository.findAllByType("inventory");
        var publicEntityIds : List<String> = components.stream().map(c -> c.getEntityId()).collect(Collectors.toList());
        entityPublicDataGenerators.forEach(g -> g.generatePublicData(data, game, playingPlayer, publicEntityIds));
    }
}
```

Figura 28. ECS: Game Public Data Generator. Font: Elaboració pròpia.

Els components que necessiten compartir les dades utilitzen el "GamePublicDataGenerator" (figura 28), per convertir les entitats en públiques i que la resta de jugadors puguin veure-les.

```java
interface TypedsRepository extends JpaRepository<Typed, String> {
    List<Typed> findAllByGame(Game game);
    List<Typed> findAllByType(String type);
    List<Typed> findAllByGameAndType(Game game, String type);
}
```

Figura 29. ECS: Typeds Repository. Font: Elaboració pròpia.

Amb aquesta interfície que estén de "JpaRespository" es genera la persistència acompanyada d'un "CRUD" bàsic (figura 29).

5.3.5 SVG

Per dibuixar els gràfics del joc s'ha utilitzat "SVG" (Scalable Vector Graphics). Com diu el seu nom, són gràfics generats a partir de vectors escalables. S'ha decidit utilitzar aquesta tecnologia ja que és totalment compatible amb "React". A més, aporta una sèrie d'avantatges respecte al "canvas", com ara que els gràfics són escalables, que aquests es poden imprimir amb una alta qualitat de resolució, que es pot canviar la mida en qualsevol moment mantenint la mateixa qualitat de resolució. [28]

Aquest apartat ha requerit d'un estudi previ, ja que es desconeixia l'existència d'aquesta tecnologia.

A continuació es mostra un exemple de com s'han realitzat els gràfics. Concretament la construcció dels vèrtex de cada hexàgon. Per vèrtex s'entén el cercle o quadrat que simula un poble o una ciutat respectivament (d'ara endavant es farà referència només a vèrtex):

```
export function VertexComponent({ entity }: any) {
  const currentPlayer = useAppSelector(getPlayerName);
  const owned = useDispatchFormBig(own, entity.id, currentPlayer);

  if(entity.type !== "vertex") return null;

    var yOffset = entity.column*32;
    var xOffset = entity.row*65 +66;

    if(entity.column > 1)
        yOffset = yOffset + 45;
    if(entity.column > 3)
        yOffset = yOffset + 50;
    if(entity.column > 5)
        yOffset = yOffset + 50;
    if(entity.column > 7)
        yOffset = yOffset + 50;
    if(entity.column > 9)
        yOffset = yOffset + 50;

    var xPos = xOffset;
    var yPos = yOffset -70;
    var entityId = entity.id;

  return (
      <g id="vertex">
        <Vertex id={entityId} cx={xPos} cy={yPos} r="20" stroke="black" stroke-width="3" fill="white"  onClick={() => {
            owned();
  }}></Vertex>
      </g>
  );
}
```

Figura 30. SVG: Vertex component. Font: Elaboració pròpia.

Com es pot observar a la figura 30, s'ha creat un component per dibuixar el vèrtex. Se li passa per paràmetre l'entitat que s'ha de dibuixar. Per poder-ho realitzar es necessita assignar-li una posició, es consulta la seva posició tan de fila com de columna facilitada pel "backend" i passada per paràmetre, mitjançant la variable entitat, i depenent de la posició se

li aplica un “offset” o un altre. Tot seguit se li assigna la “id” de l’entitat, també recuperada del “backend”.

```
const Vertex = styled.circle`
cursor: pointer;
fill-opacity: 0.4;
stroke: #000;
stroke-width: 1;
transition: transform, fill-opacity, stroke-width;
-webkit-transition: transform, fill-opacity, stroke-width;
transition-duration: 1s;
-webkit-transition-duration: 1s;;
&:hover{
  fill-opacity: 1;
  stroke: #ff0000;
  stroke-width: 5;
}
`
```

Figura 31. SVG: Constant vèrtex. Font: Elaboració pròpia.

Al no poder aplicar-se “CSS” directament a un component de “React”, és necessari importar una llibreria, que permet assignar-li estil a qualsevol element d’ html mitjançant una variable de tipus constant (figura 31).

5.3.6 Redux

Com ja s'ha explicat anteriorment, "Redux" és una eina de control d'estats, normalment utilitzada amb "React", que facilita la comunicació i la compartició de dades entre components. Crea una estructura de dades tangible que representa l'estat de l'aplicació, des d'on es poden llegir i escriure dades.

A continuació es mostra un exemple de "Redux" de com està implementat el component "own":

```
export function componentOwnReducer(
  state: ComponentOwnState = null,
  action: ComponentOwnActionTypes
) {
  switch (action.type) {
    default:
      return state;
  }
}
```

Figura 32. Redux: Component own reducer. Font: Elaboració pròpia.

Es crea el "reducer" del component passant-li per paràmetre un estat i una acció. Com a resultat retorna un nou estat (figura 32).

```
export const OWN = "componentOwn/OWN";
export interface OwnAction {
  type: typeof OWN;
  entityId: string;
  playerName: string;
}

export type ComponentOwnActionTypes = OwnAction;
```

Figura 33. Redux: Own Interface. Font: Elaboració pròpia.

Per crear l'acció, primer de tot es crea el tipus d'acció i una interfície del que haurà de contenir l'acció (figura 33).

```
export function own(
  entityId: string,
  playerName: string
): OwnAction {
  return {
    type: OWN,
    entityId,
    playerName,
  };
}
```

Figura 34. Redux: Own action. Font: Elaboració pròpia.

Finalment, es crea l'acció amb la interfície i el tipus creats anteriorment (figura 34).

```
export const componentOwnMiddleware: Middleware<{}, AppState> =
  (store: any) => (next) => async (action: ComponentOwnActionTypes) => {
    next(action);

    if (action.type === OWN) {
      await gamePost(
        store,
        `/api/v1/owned/${action.entityId}/${action.playerName}/own`,
        {}
      );
    }
  };
```

Figura 35. Redux: Own middleware. Font: Elaboració pròpia.

En aquest cas ha estat necessari afegir un "middleware" per tal de fer possible la comunicació amb el "backend" (figura 35).

5.3.7 Taulell

Per construir el taulell amb els hexàgons i els vèrtex s'ha tractat com si fos una "array" de dues dimensions, tal i com es mostra a les següents figures 36 i 37.

	q = 0	q = 1	q = 2	q = 3	q = 4	q = 5	q = 6
r = 0	(null)	(null)	(null)	3, 0	4, 0	5, 0	6, 0
r = 1	(null)	(null)	2, 1	3, 1	4, 1	5, 1	6, 1
r = 2	(null)	1, 2	2, 2	3, 2	4, 2	5, 2	6, 2
r = 3	0, 3	1, 3	2, 3	3, 3	4, 3	5, 3	6, 3
r = 4	0, 4	1, 4	2, 4	3, 4	4, 4	5, 4	(null)
r = 5	0, 5	1, 5	2, 5	3, 5	4, 5	(null)	(null)
r = 6	0, 6	1, 6	2, 6	3, 6	(null)	(null)	(null)

Figura 36. Taulell: Array dues dimensions. Font: Imatge extreta de [29].

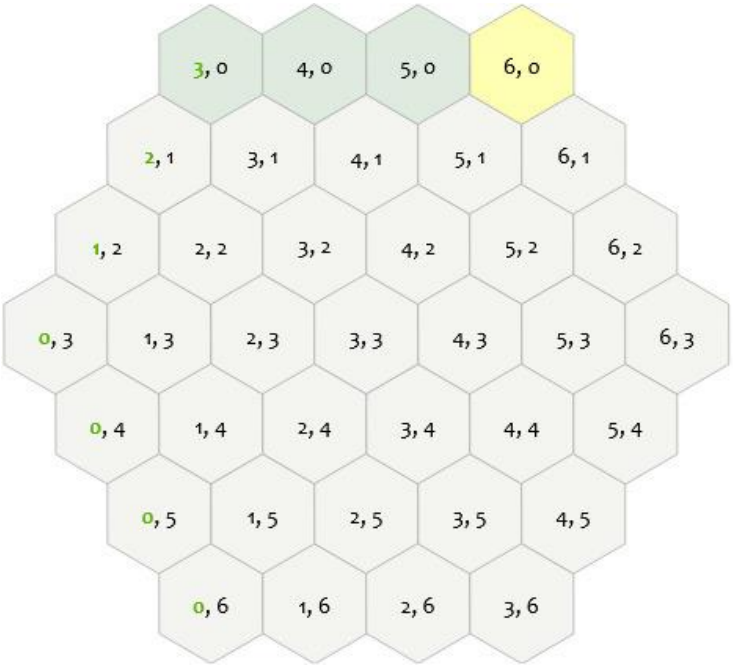


Figura 37. Taulell: Distribució hexàgons. Font: Imatge extreta de [29].

Si es respecten les posicions que es mostren a la graella, després resulta més senzill col·locar cada peça al lloc que li pertoca, ja que es pot multiplicar la posició a l'eix de les "x" i de les "y" per un valor de recol·locament. A continuació s'explica més detalladament.

Per fer-ho més entenedor, i explicar el procediment que s'ha seguit, es fa servir l'exemple dels vèrtex ja que els hexàgons tenen l'eix girat 30 graus i el seu recol·locament és més complex i difícil d'entendre.

Figura 38. Taulell: Vertexs sol·lapats. Font: Elaboració pròpia.

Suposant que ja es dibuixa el cercle de cada vèrtex, el primer pas es col·locar-los tots al mateix punt, per tant tots els vèrtexs queden solapats (figura 38).

```
var xOffset = entity.row*65 +66;
```

Figura 39. Taulell: Offset eix de les x. Font: Elaboració pròpia.

Figura 40. Taulell: Offset al eix de les x aplicat. Font: Elaboració pròpia.

Amb aquest valor de correcció, es pretén moure el vèrtex en la posició de l'eix horitzontal que li toca a cada un depenent de la posició establerta donada pel "backend". Al no tocar encara l'eix de les "y", els vèrtex segueixen solapats (figures 39 i 40).

```
var yOffset = entity.column*32;


if(entity.column > 1)
    yOffset = yOffset + 45;
if(entity.column > 3)
    yOffset = yOffset + 50;
if(entity.column > 5)
    yOffset = yOffset + 50;
if(entity.column > 7)
    yOffset = yOffset + 50;
if(entity.column > 9)
    yOffset = yOffset + 50;
```

Figura 41. Taulell: Offest eix de les y. Font: Elaboració pròpia.

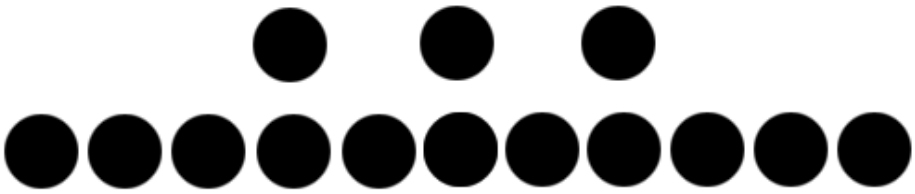


Figura 42. Taulell: Resultat visual de la primera condició. Font: Elaboració pròpia.

Exemple de quan s'executa la primera línia de codi de la imatge de cada vèrtex (figures 41 i 42).

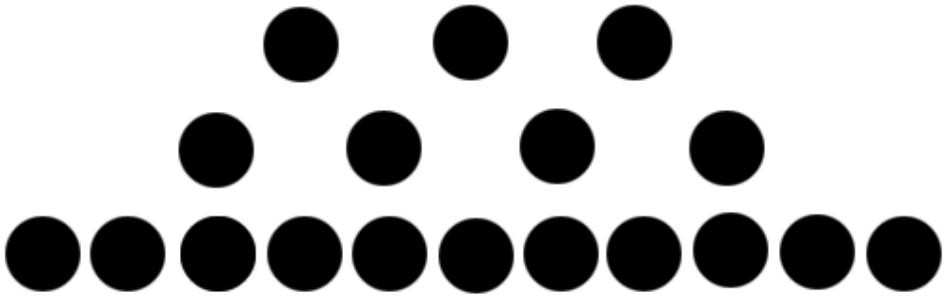


Figura 43. Taulell: Resultat visual de la segona condició. Font: Elaboració pròpia.

Exemple de quan s'executa la primera condició de la imatge (figura 43) .

Per aplicar el valor de correcció de l'eix de les "y", s'assigna un valor inicial i, mitjançant un conjunt de condicions que comproven si la posició vertical és més gran que un altre valor, s'incrementa la posició de l'eix vertical en cas de complir-se. En cas de no complir-se, el vèrtex es queda allà on es troba.

5.4 Disseny de pantalles

A continuació es presenta un possible flux de pantalles. Existeixen alternatives però aquest seria el més freqüent (figura 44).

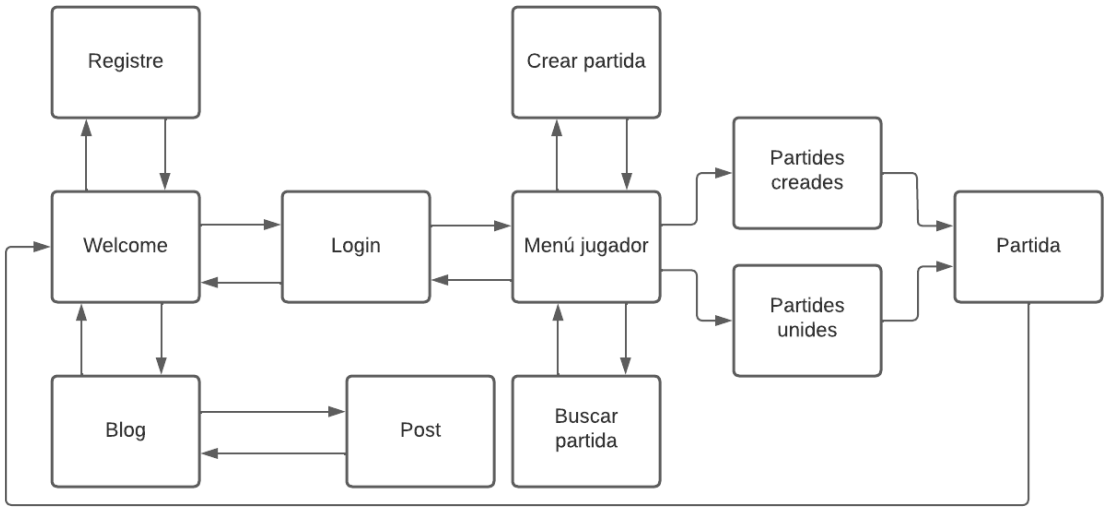


Figura 44. Disseny: Flux. Font: Elaboració pròpia.

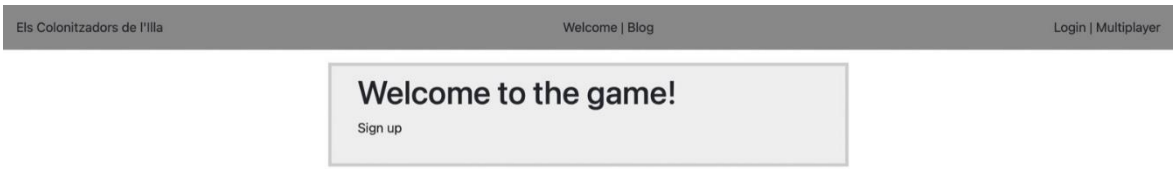


Figura 45. Disseny: Welcome. Font: Elaboració pròpia.

Aquesta és la primera pàgina que veu l'usuari quan accedeix a la web, la pàgina de benvinguda (figura 45).

Figura 46. Disseny: Sign up. Font: Elaboració pròpia.

Aquest és el formulari per a registrar-se a la web, només es demana un nom d'usuari i contrasenya (figura 46).

Figura 47. Disseny: Log in. Font: Elaboració pròpia.

Formulari d'accés a la web (figura 47).

Figura 48. Disseny: Blog. Font: Elaboració pròpia.

Quan l'usuari fa clic al botó de blog, apareix el llistat amb totes les funcionalitats del joc i la data en què van ser creades (figura 48).

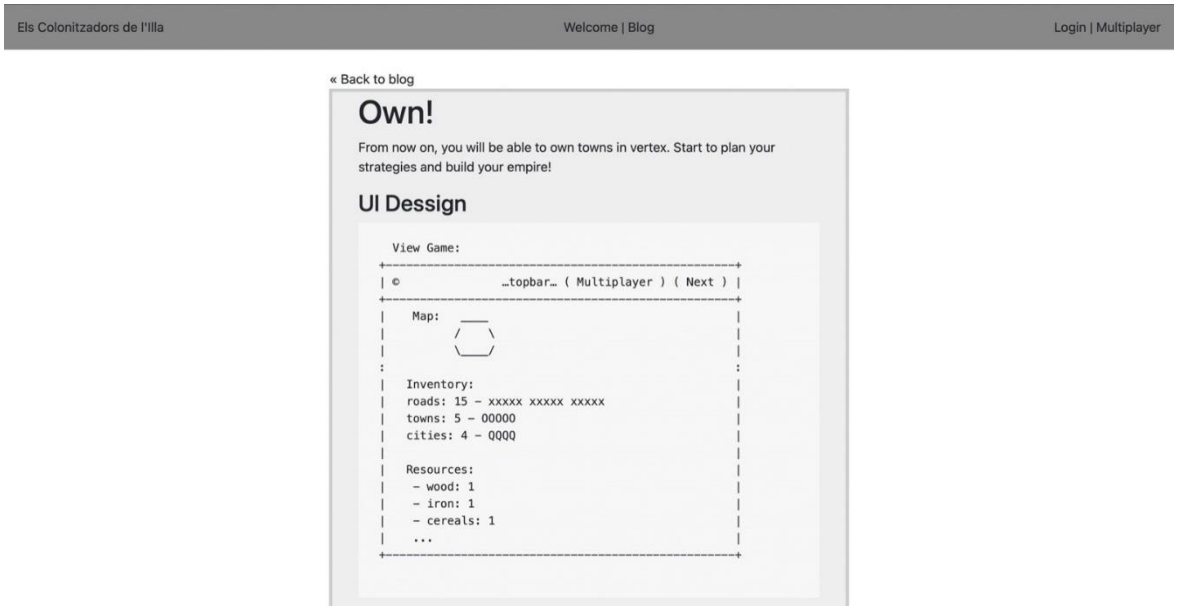


Figura 49. Disseny: Post 1. Font: Elaboració pròpia.

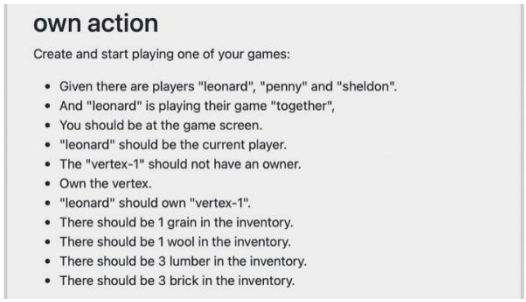


Figura 50. Disseny: Post 2. Font: Elaboració pròpia.

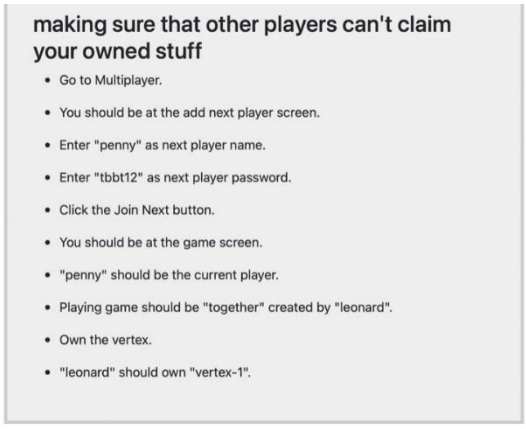


Figura 51. Disseny: Post 3. Font: Elaboració pròpia.

Quan l'usuari fa clic a un element de la llista pot observar tota la informació sobre la funcionalitat (figures 49, 50 i 51).

Figura 52. Disseny: Menú de Joc. Font: Elaboració pròpia.

Aquest és el menú de joc un cop l'usuari s'ha autenticat, on té l'opció de crear una partida, consultar el llistat de partides que està jugant, consultar el llistat de partides que ha creat i buscar la partida d'algun amic (figura 52).

Figura 53. Disseny: Creació partida. Font: Elaboració pròpia.

Aquest seria el formulari per a crear una partida, on l'únic que es demana és el nom de la partida (figura 53).

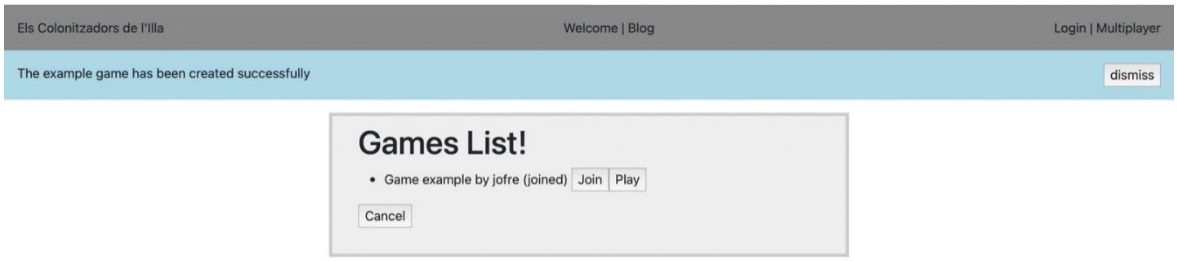


Figura 54. Disseny: Llistat de partides. Font: Elaboració pròpia.

Aquesta seria la llista de partides que està jugant l'usuari, amb un botó d'unir-se (si encara no s'hi ha unit) i un botó de jugar (figura 54).

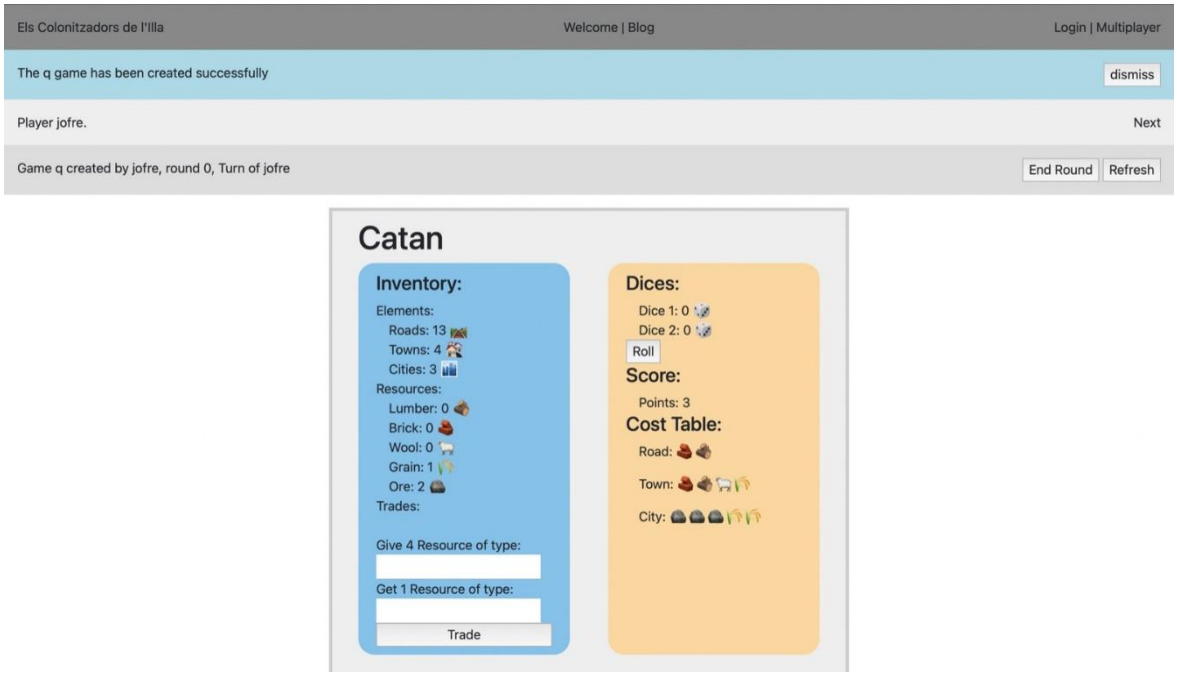


Figura 55. Disseny: partida 1. Font: Elaboració pròpia.

Quan l'usuari accedeix a la partida pot consultar a la barra de dalt el nom de la partida, qui la ha creat, el número de ronda i de quin usuari és el torn. En aquesta mateixa barra, a la dreta, pot passar el torn o refrescar la pantalla. Disposa d'un requadre en blau que és el seu inventari i l'opció de intercanviar els seus recursos amb la màquina a raó de 4:1. També, disposa del requadre taronja, on podrà llençar els daus una vegada per torn, podrà consultar la seva puntuació i podrà consultar quan costa comprar cada element del joc (figura 55).

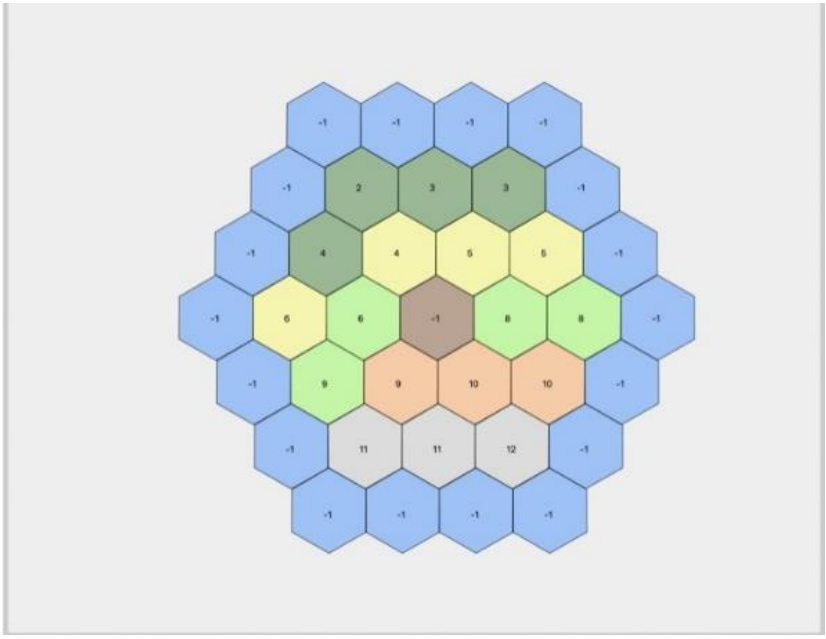


Figura 56. Disseny: Partida 2. Font: Elaboració pròpia.

Sota els dos requadres de colors es troba el mapa de la partida. On els usuaris podran realitzar les seves accions estratègiques per intentar guanyar la partida. Aquestes accions poden ser: comprar poblats, comprar ciutats i comprar carreteres (figura 56).

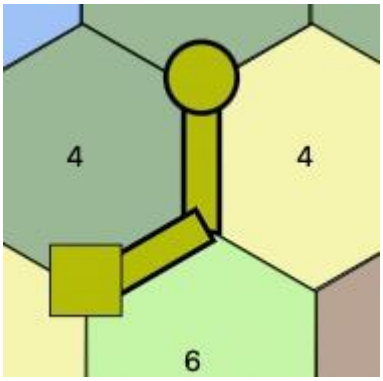


Figura 57. Disseny: Partida 3. Font: Elaboració pròpia.

En aquesta imatge es pot observar com es representa el poblat en forma de cercle, la ciutat en forma de quadrat i les carreteres en forma de rectangle (figura 57).

5.5 Problemes durant el desenvolupament

Durant el desenvolupament del projecte han anat sorgint problemes, inconvenients i petits matisos que es desconeixien quan es va plantejar el mateix. Això ha implicat una pèrdua de temps, que ha afectat i s'ha vist reflectit a la qualitat final del projecte.

5.5.1 Error de maven

Per motius desconeguts, d'un dia per l'altre, el projecte va deixar de funcionar. Quan s'executava el "backend" petava amb un cadena d'errors extensa. Primer es sospitava que era un error del codi però, després de revisar exhaustivament tot el codi (des de que no hi havia error fins que va aparèixer aquest), es va arribar a la conclusió que no ho era.

A continuació es va fer una lectura, també exhaustiva, de la cadena d'errors. Alguns dels errors de l'origen de la cadena eren que no es podia iniciar l' "Application Context" (encarregat d'inicialitzar el "backend" amb tot el funcionament intern d' "Spring"), i que l'"Spring" no podia crear una "Bean" (objecte que el contenidor d' "Spring" necessita inicialitzar i gestionar-lo per poder funcionar). Per tant, es va procedir a revisar fitxer a fitxer per si faltava algun import, alguna etiqueta o alguna herència d' "Spring". Però tampoc es va trobar res sospitós.

Després d'hores buscant es va trobar que, respecte a versions anteriors del projecte, havien desaparegut directoris amb dependències de Maven. Es va buscar com reinstal·lar Maven i reconfigurar-lo amb "Spring". Finalment, es va aconseguir executar el projecte sense errors.

L'error de Maven va implicar una pèrdua de dos o tres dies de treball.

5.5.2 "CSS" i components "react"

Es desconeixia que no es podien importar fitxers de tipus "CSS" a components de "React". Es va estar revisant perquè no deixava importar el fitxer però no es va trobar el perquè. Després de vàries hores d'investigació i de provar vàries alternatives, es va trobar una llibreria que permet crear una variable constant, que assigna el tipus d'element al qual anirà aplicat aquell "CSS" i els propis atributs del "CSS" dins d'una "String".

Aquest inconvenient va suposar aproximadament dos dies de treball.

5.5.3 Mapa Complex

Quan es va fer el diagrama de Gantt es va calcular que el mapa i els elements del mapa estarien desenvolupats aproximadament en dues setmanes, però el desenvolupament ha durat un mes i mig.

5.5.4 Dificultat en realitzar accions que haurien de ser trivials

A causa de que es fan servir tecnologies i patrons dels quals es té un coneixement baix o nul, han sorgit inconvenients alhora de realitzar accions trivials o rutinàries, com pot ser el simple fet d'intentar recuperar informació d'un objecte. Com s'explica més endavant, el patró "ECS", a grans trets, consisteix en generar una "id" i a aquella "id" se li associen les funcionalitats que pugui necessitar. El problema és que al no disposar d'un objecte que unifiqui totes les funcionalitats i que cada funcionalitat queda guardada en repositoris diferents, la complexitat alhora de treballar amb aquell objecte s'incrementa. A aquesta complexitat, també se li ha de sumar que no es disposa d'una base de dades convencional, sinó que es fan servir els repositoris de "jpa" (dels quals no disposava de cap coneixement previ). Pot ser còmode quan es coneix la tecnologia però al principi costa interactuar amb ells.

5.5.5 Tests i "Coverage"

Al no disposar de soltesa en la realització de tests es va arribar a un punt en el qual es trigava més en fer el test i, posteriorment al desenvolupament, comprovar i fer que el "coverage" fos del cent per cent, que en el propi desenvolupament. Això feia inviable acabar el projecte a temps. Per aquest motiu el que s'ha fet és reduir l'exhaustivitat dels tests i deixar aparcat el "coverage". En cas d'acabar a temps les funcionalitats del joc s'invertirà temps en arreglar el "coverage".

5.5.6 Valoració de la Planificació Inicial i el desenvolupament

Durant la redacció de l'avantprojecte es va realitzar la planificació inicial, aquesta planificació inicial comportava definir les fases del projecte, la inversió temporal, la inversió econòmica i la inversió de recursos que es necessitarien.

Respecte les fases del projecte i el temps invertit en cada una d'elles no ha sorgit cap imprevist que hagi provocat un desajust de fases o temps. Sí és cert que han sorgit imprevistos amb alguna implementació de funcionalitats però s'han solucionat amb fluïdesa. A més, durant la planificació inicial, en el moment d'assignar el temps a cada tasca, ja es va tenir en compte que podrien sorgir problemes i cada tasca disposa d'un percentatge de temps extra segons la seva complexitat que s'ha fet servir per mitigar les emergències tècniques.

La inversió econòmica i de recursos també han estat les planejades a l'estudi de viabilitat. S'ha afegit alguna tecnologia no planejada inicialment, com per exemple SVG, però és de caràcter gratuït i fàcil d'integrar amb "React".

6 Conclusions

S'ha aconseguit crear amb èxit el Joc del Colonitzadors de l'illa inspirat en el joc de Catan. S'han completat tots els objectius i funcionalitats del joc principals. No obstant i això, com que el temps i els recursos són limitats, l'abast del projecte també ha estat molt reduït i els objectius secundaris, com la implementació de cartes de desenvolupament i el comerç entre jugadors, han quedat pendents de desenvolupar-se.

L'ús del TDD és útil per projectes de llarg termini, que requeriran manteniment, a més d'un equip de desenvolupadors, perquè l'aplicació de TDD asseguri que tothom vagi en la mateixa direcció. No obstant, l'aplicació de TDD en aquest projecte s'ha realitzat perquè era un dels requeriments inicials. Tanmateix, el TDD és una metodologia que no era necessària per l'elaboració d'aquest projecte, ja que potser alenteix més el desenvolupament d'aquest que no pas li aporta major eficiència.

Per la part del "backend", l'elecció del "framework" "Spring", ha estat encertada. Es tracta d'un "framework" còmode d'iniciar i fàcil de treballar-hi. Per tant "l'Spring" és un "framework" que gairebé no dóna problemes.

L'elecció del patró ECS juntament amb la persistència en forma de repositoris "jpa", són adients per a aquest tipus de projecte. Es requereix un temps d'aprenentatge per a conèixer i habituar-se a aquest tipus de patró però, un cop es domina, es necessita menys temps per a desenvolupar les funcionalitats del joc i és més eficient que el que s'obté si es compara amb la programació orientada a objectes convencionals. A més, l' ECS és útil, ja que permet mantenir un ordre i una estructura del projecte que fa que sigui intuïtiu navegar-hi. Altrament, permet reduir l'acoblament i, consegüentment, garanteix que el projecte sigui fàcilment escalable.

Per la part del "frontend", s'ha utilitzat "React" i "Redux" ja que eren una de les premisses del projecte. L'ús d'aquestes tecnologies ha estat satisfactori. En projectes grans, com és el cas del TFG, "React" millora el rendiment de l'aplicació i, en ser de codi obert i disposar d'una estructura modular, és compatible amb llibreries i tecnologies de tot tipus permetent

així una integració senzilla. Com ja s'ha explicat durant el projecte, s'ha integrat amb "redux", que és útil per projectes també grans, perquè obliga a mantenir una estructura del projecte i una organització coherent. Tanmateix, també s'aconsegueix optimitzar el rendiment de l'aplicació i permet que l'experiència de l'usuari sigui més fluida.

Per pintar els gràfics del joc s'ha utilitzat SVG ja que és compatible amb "React". Es requereix d'una mínima formació prèvia abans d'utilitzar aquesta tecnologia, però l'ús d'aquesta eina ha estat una decisió apropiada perquè ha estat fàcil d'utilitzar i d'implementar amb "React".

7 Possibles Ampliacions

Existeixen varies ampliacions que es podrien fer al joc en cas de disposar de més temps. A continuació s'exposen possibles ampliacions.

En primer lloc, acabar els objectius secundaris plantejats inicialment, com les cartes de desenvolupament, on el cost unitari de les cartes seria d'una ovella, un cereal i un mineral. Cadascuna d'elles et proporcionaria una avantatge, com per exemple: El propietari de la carta escull un recurs i tots els jugadors li hauran de donar tots els recursos d'aquell tipus. Un altre avantatge és la carta que atorga un punt de victòria. O la que et permet construir dues carreteres gratuïtament, etc. L'altre objectiu secundari que faltaria implementar és l'intercanvi de recursos entre jugadors amb les quantitats variables.

També es podria afegir la figura del lladre, que com ja es va explicar al marc teòric, la seva funció és denegar l'obtenció de recursos d'una zona. Si toca el número set, el jugador ha de moure el lladre a una altre zona que consideri estratègica i robar un recurs aleatori del jugador que posseeixi una propietat a la zona en qüestió.

També, en el moment de crear la partida, es podria donar l'oportunitat d'escollir la mida del mapa i canviar la forma del mateix. També donar l'opció de poder elegir el color i la forma de les cases, ciutats i pobles. A més donar l'opció de poder escollir un avatar de perfil.

Finalment, també intentaria realitzar un apartat on el jugador pogués consultar la seva trajectòria i les estadístiques de les seves partides i una classificació dels jugadors amb millor estadístiques.

8 Bibliografía

[1] «20 minutos,» 28 Abril 2021. [En línea]. Available: https://www.20minutos.es/noticia/4674789/0/la-venta-de-juegos-de-mesa-se-disparo-un-18-el-ano-de-la-pandemia-virus-y-monopoly-lideraron-la-demanda-nacional/?autoref=true. [Último acceso: 31 Gener 2022].

[2] «Catan, "About Us",» [En línea]. Available: https://www.catan.com/about-us. [Último acceso: 4 Gener 2022].

[3] «Catan, "Base Rules",» [En línea]. Available: https://www.catan.com/sites/prod/files/2021-06/catan_base_rules_2020_200707.pdf. [Último acceso: 5 Enero 2022].

[4] «Devir,» [En línea]. Available: http://devir.es/producto/catan/. [Último acceso: 5 Gener 2022].

[5] «1jour-1jeu, "Catan: Cities & Knights Rulebook",» [En línea]. Available: https://cdn.1j1ju.com/medias/93/08/2a-catan-cities-knights-rulebook.pdf. [Último acceso: 6 Enero 2022].

[6] «1jour-1jeu, "Catan: Explorers & Pirates",» [En línea]. Available: https://cdn.1j1ju.com/medias/8d/db/3c-catan-explorers-pirates-rulebook.pdf. [Último acceso: 8 Gener 2022].

[7] «Catan, "Rivals for Catan",» [En línea]. Available: https://www.catan.com/rivals-catan. [Último acceso: 10 Gener 2022].

[8] «f.g.bradleys, "Rivals for catan Rules",» [En línea]. Available: https://www.fgbradleys.com/rules/rules3/RivalsforCatanRules.pdf. [Último acceso: 15 Gener 2022].

[9] «Catan Universe,» [En línea]. Available: https://catanuniverse.com/es/. [Último acceso: 24 Gener 2022].

[10] «Google Play, "Catan Classic",» [En línea]. Available: https://play.google.com/store/apps/details?id=com.exozet.android.catan&hl=es&gl=US. [Último acceso: 25 Gener 2022].

[11] «Colonist,» [En línea]. Available: https://colonist.io. [Último acceso: 25 Gener 2022].

[12] R. C. Martin, «TheClean Code Blog,» [En línea]. Available: https://blog.cleancoder.com/uncle-bob/2016/11/10/TDD-Doesnt-work.html. [Último acceso: 29 Desembre 2021].

[13] A. Moratilla, «Adictos al Trabajo, "Revisión de TDD by Example",» [En línea]. Available: https://www.adictosaltrabajo.com/2015/12/21/revision-de-test-driven-development-by-example-de-kent-beck/. [Último acceso: 29 Desembre 2021].

[14] K. Beck, «Test- Driven Development by Example,» Addison Wesley, Novembre de 2002.

[15] R. Pahino, «campusMVP, ¿Qué son Spring framework y Spring Boot?,» [En línea]. Available: https://www.campusmvp.es/recursos/post/que-son-spring-framework-y-spring-boot-tu-primer-programa-java-con-este-framework.aspx. [Último acceso: 30 Desembre 2021].

[16] J. Cuervas, «atSistemas, "¿Qué es Spring Framework? Características",» [En línea]. Available: https://www.atsistemas.com/blog/qu-es-spring-framework-caractersticas-i. [Último acceso: 30 Desembre 2021].

[17] C. Gackenheimer, «Introduction to React,» Apress, September 2015, pp. 1-16.

[18] «Redux, "Motivation",» [En línea]. Available: https://redux.js.org/understanding/thinking-in-redux/motivation. [Último acceso: 31 Desembre 2021].

[19] «Redux, "Three Principles,» [En línea]. Available: https://redux.js.org/understanding/thinking-in-redux/three-principles. [Último acceso: 31 Desembre 2021].

[20] A. Bachuk, «Smashing Magazine, "Redux · An Introduction",» [En línea]. Available: https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/. [Último acceso: 31 Desembre 2021].

[21] «Redux, "Deriving Data with Selectors,» [En línea]. Available: https://redux.js.org/usage/deriving-data-selectors. [Último acceso: 2 Gener 2022].

[22] S. Mertens, «GitHub, "Entity Component System FAQ",» [En línea]. Available: https://github.com/SanderMertens/ecs-faq#what-is-ecs. [Último acceso: 3 Gener 2022].

[23] «Redux, "Middleware",» [En línea]. Available: https://redux.js.org/understanding/history-and-design/middleware. [Último acceso: 3 Gener 2022].

[24] D. Ródenas, «Ecampus, "LS2-Architecture",» [En línea]. Available: https://aulavirtual.tecnocampus.cat/pluginfile.php/127257/mod_resource/content/1/LS2-2021-04-Architecture-frontend.pdf. [Último acceso: 4 Gener 2022].

[25] B. Boehm, «A Spiral Model of Software Development and Enhancement,» *Computer,* vol. 21, nº 5, pp. 61 - 72, May 1988.

[26] B. Boehm, «Spiral Development: Experience, Principles, and Refinements,» *SPECIAL REPORT, CMU/SEI-2000-SR-008,* pp. 1-5, July 2000.

[27] «Spiral Model, wikipedia,» [En línea]. Available: Disponible a https://en.wikipedia.org/wiki/Spiral_model. [Último acceso: 5 12 2021].

[28] «SVG Tutorial,» [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Introduction. [Último acceso: 15 3 2022].

[29] R. B. Games, «Hexagonal Grids,» [En línea]. Available: https://www.redblobgames.com/grids/hexagons/. [Último acceso: 1 3 2022].