Universitat de Girona
Escola Politècnica Superior

Projecte - Treball final de carrera

Estudi: EINF

Títol: Disseny i implementació d'un joc d'estratègia en xarxa

Document: Memòria

Alumne: Pau Vidal Escudero

Tutor: Gustavo Patow
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria (mes/any) Juny 2015

ÍNDEX

ÍNDEX DE FIGURES

1 INTRODUCCIÓ

El sector dels videojocs va aparèixer als anys 80, però no es va donar a conèixer fins l'any 81, on els fundadors de la coneguda Atari van treure el primer videojoc comercial. A partir d'aquest punt, la indústria es va desenvolupar, al llarg del temps, màquines recreatives, videoconsoles (de taula i portàtils) i a la vegada establint-se en altres sistemes com ordinadors i, ja recentment, dispositius mòbils.

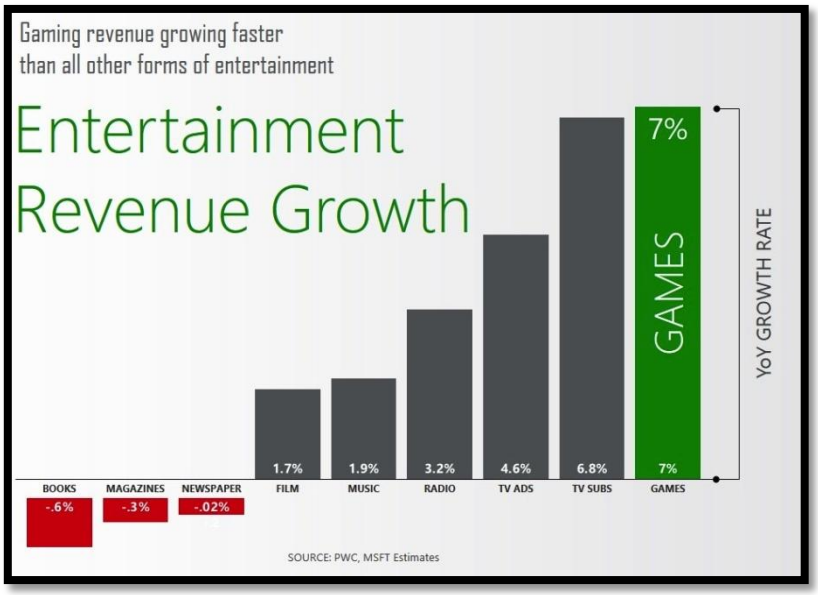

Figura 1: Creixement percentual del mercat dels videojocs l'últim any

Avui dia, gràcies a la inclusió del sector mòbil i dels motors de desenvolupament, la indústria dels videojocs ha crescut tant que ja genera un benefici major (101.62 bilions de dòlars) que el sector cinematogràfic (88.3 bilions de dòlars) i de música (5.9 bilions de dòlars) junts.

1.1 MOTIVACIÓ PERSONAL

La meva motivació per a realitzar aquest projecte ha estat la de poder compaginar una experiència en el món del desenvolupament de videojocs amb la meva experiència pròpia en el món del desenvolupament web. El món dels videojocs és una part de la informàtica que m'agrada molt i a la que aspiro arribar-hi en la meva etapa professional. El projecte final de carrera és una oportunitat per iniciar-se, ja que és possible treballar juntament amb un tutor experimentat en la matèria, disposat a supervisar, ajudar i a transmetre entusiasme. Per tant, podem dir que el meu objectiu és tenir la base d'un producte interessant, per poder més endavant complementar-lo amb funcionalitats noves.

1.2 PROPÒSITS I OBJECTIUS

L'objectiu d'aquest Projecte de Final de Carrera és crear un videojoc de tipus MMORTS (Massively Multiplayer Online Real-Time Strategy) on l'usuari té a la seva disposició una o vàries ciutats i les va millorant amb l'obtenció de recursos a través del temps, o interactuant amb altres usuaris.

Per augmentar el nombre de recursos que s'obtenen regularment, l'usuari ha de millorar els diversos edificis que hi ha a cada civilització. També s'haurà d'emprar el nombre de recursos/hora obtingut per poder-ne fabricar de més complexos.

Aquest projecte apunta a crear un entorn on l'usuari podrà connectar-se per web a la seva ciutat i millorar-la usant els recursos obtinguts.

També es configurarà un servidor de proves a través d'una RaspberryPi, i es prepararan tots els processos per mantenir el servei en línia i dur a terme els càlculs d'increment de recursos en temps real.

1.3 BREU HISTÒRIA DELS RTS I MMORTS

Els videojocs d'estratègia en temps real (RTS) són videojocs d'estratègia en els que no hi ha torns, sinó que el temps avança de forma contínua per als jugadors.

Els videojocs en temps real són un dels subgèneres dels jocs d'estratègia més dinàmics que hi ha. Els RTS poden estar pensats en ser jugats de manera molt dinàmica i molt ràpida en alguns subgèneres, o de manera dinàmica però al llarg d'un temps elevat en alguns altres, però en tots els casos es centren en la recol·lecció de recursos i l'acció militar.

El gènere RTS va néixer a l'any 1982 amb el joc *Cytron Masters,* on cada jugador assumia el rol de comandant, representats a l'esquerra i dreta de la pantalla. Tenint possessió d'uns generadors, podien generar un robots anomenats Cytrons i moure'ls per destruir la base enemiga.

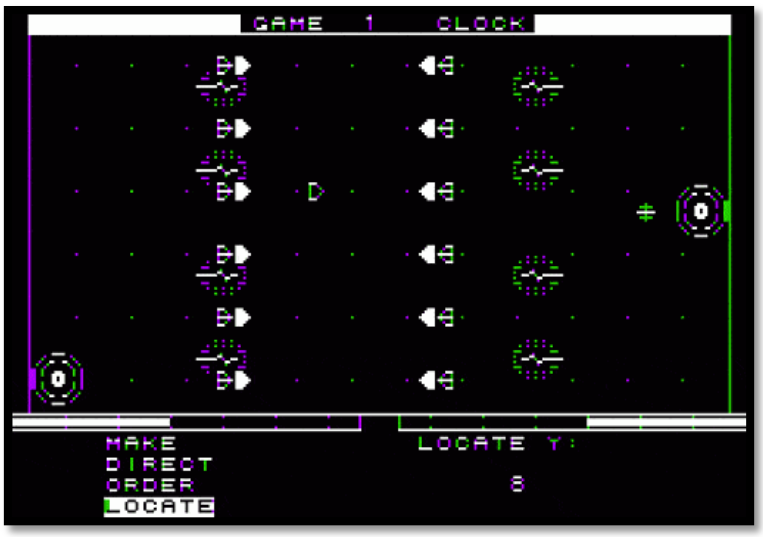

Figura 2: Cytron Masters (1982)

L'1 de Gener de 1992 va sortir el joc *Dune II: The Building of a Dynasty,* que va ser el primer en fer aparèixer tots els aspectes que més endavant tindrien tots els successors, com:

- Recopilació de recursos per reclutar exèrcit i crear edificis.
- Dependència de construcció (arbre de tecnologia).
- Unitats mòbils que apareixen dels edificis.
- Diferents faccions/civilitzacions amb unitats, habilitats i armes úniques.
- Cursor contextual per introduir ordres.

Figura 3: Dune II: The Building of a Dynasty (1992)

Uns anys després, i aprofitant les bases que va deixar el Dune 2, va aparèixer un dels jocs més coneguts del gènere RTS, *Warcraft* (1994). Va incloure novetats en nous modes com el de crear missions personalitzades i va consolidar per primer cop la possibilitat de poder jugar a través d'internet.

Figura 4: Warcraft (1994)

L'any 1997 va sortir un altre dels jocs més coneguts del gènere RTS, *Age of Empires,* que va incloure l'ús d'*sprites* isomètrics per donar una sensació de profunditat al joc.

Figura 5: Age of Empires (1997)

A partir d'aquí, tot i que els RTS han continuat evolucionant aplicant les noves tecnologia en 3D, l'any 1999 va aparèixer un nou subgènere que aprofitava al màxim les característiques d'internet a través de la web, els MMORTS.

En el subgènere basat en la interacció online trobem els MMORTS, on els jugadors construeixen i milloren poc a poc una (o vàries) civilitzacions a base de recol·lectar recursos o robar-los d'altres jugadors.

Tot i que llavors la web tenia moltes limitacions envers avui dia, es van deixar de banda els gràfics per centrar tota la jugabilitat del joc en la estratègia pura i la interacció entre milers d'usuaris. Es van canviar les partides curtes però intenses per la evolució a llarg termini de tota la civilització.

Un dels primers MMORTS va ser l'Ogame, centrat en desenvolupar planetes per l'espai per crear exèrcits i conquerir nous astres.

Figura 6: Ogame (1999)

Avui dia, amb l'entrada amb força dels jocs mòbils, podem dir que els MMORTS estan evolucionant cap a un subgènere nou més interactiu, que obliga a l'usuari a interactuar amb la civilització creant-la de forma visual i estratègica, afegint combats en temps real depenent de l'habilitat dels usuaris per crear la defensa o saber com i amb què atacar, a més de fer que es connecti cada cert temps per recollir els recursos que es generen automàticament. D'aquesta manera aconsegueixen mantenir a l'usuari molt més pendent del joc (a part de controlar l'enviament de notificacions). El més famós que es coneix és el *Clash of Clans*, que va aparèixer en 2013.

Figura 7: Clash of Clans (2013)

1.4 ESTRUCTURA DEL DOCUMENT

Aquest document s'ha organitzat en 15 capítols, que són els següents.

1. Introducció, motivació, propòsit i objectius del projecte.

En aquest capítol s'explica el perquè del desenvolupament d'aquest projecte, quins són els objectius proposats i com s'ha organitzat el desenvolupament.

2. Estudi de viabilitat.

En aquest capítol es justifica els paràmetres que fan possible el desenvolupament del projecte.

3. Metodologia.

Aquest capítol conté una explicació de la metodologia utilitzada.

4. Planificació.

En aquesta etapa es defineix la estratègia seguida per arribar als objectius plantejats.

5. Entorn de treball i conceptes previs.

En aquest capítol es descriuen els diferents aspectes relacionats amb el desenvolupament general del projecte, que ajudaran a entendre millor els següents capítols.

També es tractaran les principals accions desenvolupades durant les primeres etapes de la realització del videojoc. S'inclouran els passos d'estudi i aprenentatge de conceptes que s'han usat en el desenvolupament.

6. Requisits del sistema.

Aquest capítol es defineixen els requeriments del software, els quals recullen, a grans trets, els objectius de la aplicació juntament amb les funcionalitats que es volen obtenir. Aquest document permet entendre els elements que envolten el sistema informàtic que s'intenta construir.

7. Estudis i decisions.

Aquesta secció conté una descripció de les eines usades, amb les característiques i l'ús que se'ls ha donat, tant de llibreries i motor com de software.

8. Anàlisis i disseny del sistema.

Aquest apartat proporciona una comprensió precisa de les necessitats del sistema. Es a dir, s'encarrega de la investigació del problema a resoldre, però no tracta el trobar una solució. Usant Enginyeria del Sofware, en aquesta secció es tradueixen els requeriments esmentats en capítols anteriors a un llenguatge més formal. La part de disseny permet augmentar el nivell de especificació, i realitzar un esquema de implementació del sistema mitjançant vàries eines de programació orientades a objectes.

9. Implementació i proves.

En aquest capítol es donen a conèixer com s'ha construir l'aplicació, les classes i els mètodes implementats que resulten més significatius per la comprensió del funcionament del videojoc.

10. Resultats.

En aquest capítol es mostren proves d'execució de l'aplicació, es mostren imatges del videojoc, incloent interfícies, imatges de la partida, i tot el que es pot visualitzar del conjunt que ha estat implementat.

11. Conclusions.

En aquest apartat s'exposen les conclusions extretes una vegada finalitzat el projecte.

12. Treball futur.

En aquesta secció s'exposa tot allò que es pot millorar en el videojoc, o ampliar de forma interessant.

13. Bibliografia.

Aquest capítol conté les referències usades pel desenvolupament del projecte.

14. Annexes.

Aquesta secció conté les definicions dels tecnicismes més freqüents usats.

2 ESTUDI DE VIABILITAT

Per a desenvolupar el projecte no es requereix d'una gran infraestructura i els costos d'estructura són inexistents. El material amb el que s'ha treballat és:

- Ordinador amb un sistema operatiu de distribució Linux.
- Una RaspberryPi on apunta el domini configurada per tenir un sistema de proves real.

2.1 RECURSOS HUMANS

Al ser un videojoc per web, es requereixen com a mínim tres elements essencials: disseny, programació i rentabilitat. Si qualsevol dels tres components falla, el videojoc no acabarà contentant al públic o no generarà suficient benefici per mantenir-lo en línia.

Com es pot comprovar, per molt ben programat que estigui el videojoc o molt ben pensat el model de negoci, si no es treballa amb la interfície i no s'ajuda a l'usuari en l'entendiment del sistema, el joc no tindrà cap possibilitat de tenir èxit.

Figura 8: Exemple d'una interfície mal dissenyada

Una solució a aquesta problemàtica és demanar ajuda a un dissenyador gràfic. En el procés de crear un videojoc, un dissenyador gràfic té la responsabilitat de realitzar els dibuixos i temes necessaris de la interfície gràfica del joc (fons, imatges d'edificis, recursos...) que interaccionen amb el comportament del programa.

En aquest projecte no s'ha comptat amb un dissenyador gràfic, però s'han tractat prou els elements gràfics per tenir un resultat final més apurat.

Per altra banda, el joc té tots els elements necessaris en la seva funcionalitat per poder incloure elements que augmentin la rentabilitat. Per exemple, en un futur es podrien comprar recursos o reduir temps de millores a partir de diners reals.

En la part de programació es requereix la participació d'un desenvolupador de software capaç de plasmar les idees inicials en el codi que farà que tot funcioni. Aquest és el rol principal assumit pel projecte.

2.2 AVALUACIÓ PRÈVIA DE COSTOS I MITJANS

Pel desenvolupament del sistema és necessari dur a terme un estudi de viabilitat tenint en compte la viabilitat econòmica, la viabilitat tècnica i la viabilitat legal. Com que un videojoc d'aquest tipus presenta pocs problemes legals, només es consideren les següents àrees:

- Estudi de la viabilitat tecnològica.
- Estudi de la viabilitat econòmica.

2.2.1 Estudi de la viabilitat tecnològica

L'estudi de la viabilitat tecnològica comença amb una definició tècnica del videojoc proposat, donant resposta a les següents preguntes:

- Quines tecnologies es requereixen per aconseguir la funcionalitat i el rendiment del joc?
- Quins nous materials, mètodes o processos son necessaris?
- Com afectaran els costos aquests elements de tecnologia?

Afortunadament, es disposa inicialment del hardware necessari per a poder realitzar el projecte, és a dir, un ordinador capaç de realitzar tota la programació, compilació i que fa possible la execució de tot el programari que d'alguna manera ha participat en el desenvolupament del videojoc en major o menor part.

També es disposa d'una RaspberryPi per tenir el sistema en un entorn estable per poder fer proves de la degradació produïda durant llargs períodes de temps.

2.2.2 Estudi de la viabilitat econòmica

La valoració de l'anàlisi econòmic és separada en dos parts: els costos de recursos humans i els costos de maquinària.

2.2.2.1 Costos de recursos humans

En aquest estudi, es planteja un cas hipotètic més pròxim a la realitat, on existeix personal específic en cada una de les tasques del desenvolupament. D'aquesta manera tenim quatre perfils diferents, cada un amb els costos per hora, següents:

- Costos analista: 20 € l'hora.
- Costos dissenyador: 20 € / hora.
- Costos programador: 15 € / hora.
- Costos il·lustrador: 15 € / hora.

TASCA	PERFIL	HORES	COSTOS
LECTURA I INVESTIGACIÓ	Analista	40	800€
	Dissenyador	40	800€
ESTUDI DE TECNOLOGIES WEB	Analista	40	800€
	Dissenyador	40	800€
DISSENY DELS ALGORITMES	Analista	90	900€
	Dissenyador		900€
IMPLEMENTACIÓ DELS ALGORITMES	Programador	90	1350 €
PROVES I OPTIMITZACIONS	Programador	50	750 €
CREACIÓ DE RECURSOS GRÀFICS	Il·lustrador	80	1200 €
DISSENY I USABILITAT DE LA PLANA WEB	Il·lustrador	80	1200 €
DOCUMENTACIÓ	Analista	90	1800 €
TOTAL		**640**	**11300 €**

2.2.2.2 Costos de maquinària

Pel que fa als costos de maquinària, s'hauria de tenir en compte els costos dels ordinadors, i del material necessari per realitzar tot els passos. S'inclou el material que haurien d'usar tots els membres de l'equip per a la realització correcte del projecte. S'ha considerat que el preu mitjà dels ordinadors és de 1200 €.

COMPONENT	UNITATS	PREU UNITARI	PREU
ORDINADOR	3	1200 €	3200 €
RASPBERRYPI+ MICROSD	1	46 €	46 €
TAULETA DIGITALITZADORA	1	150 €	150 €
TOTAL			**3396 €**

No s'han inclòs costos de llicència, ja que o bé es tractaven de llicències gratuïtes, o bé de versions de proves per a realitzar tasques concretes del projecte.

D'aquesta manera, el cost total del projecte està estimat a uns 14696 €.

3 METODOLOGIA

Per a la realització del projecte no s'ha seguit una metodologia de treball estàndard, sinó que s'ha triat i usat una metodologia personalitzada plantejada amb el tutor.

Els passos d'aquesta metodologia són els següents:

0. Triar el treball a realitzar.

1. Decidir el llenguatge de programació i eines a utilitzar.

2. Aprendre el llenguatge de programació i les eines escollides.

3. Estructurar el treball en parts segons les funcions que s'han de realitzar.

4. Desenvolupar la part corresponent seguint l'ordre de l'estructura del treball.

5. Fer les comprovacions per confirmar que el funcionament és correcte al acabar cada part.

- Si al fer les comprovacions el resultats no és el desitjat, es tornarà al punt 4 per a realitzar els canvis oportuns a la última part desenvolupada o a les anteriors, si així és necessari.
- Si al fer les comprovacions el resultat és el desitjat, es desenvoluparà la següent part tornat al punt 4. Una vegada que s'hagin finalitzar totes les parts amb les respectives comprovacions, s'iniciarà el punt 6.

6. Unir totes les parts desenvolupades i comprovar que el funcionament és correcte.

- Si al fer les comprovacions el resultat no és el desitjat, es tornarà al punt 4 per realitzar els canvis oportuns en la última part desenvolupada o en les anterior, si així és necessari.
- Si al realitzar les comprovacions el resultat és l'esperat, s'iniciarà el punt 7.

7. Generar diferents models d'exemple per a comprovar que el funcionament és el correcte.

- Si al fer les comprovacions el resultat no és el desitjat, es tornarà al punt 4 per a realitzar els canvis oportuns a la última part desenvolupada o en les anteriors si així és convenient.
- Si al realitzar les comprovacions el resultat és l'esperat, s'iniciarà el punt 8.

8. Arrodonir la documentació desenvolupada al llarg del projecte.

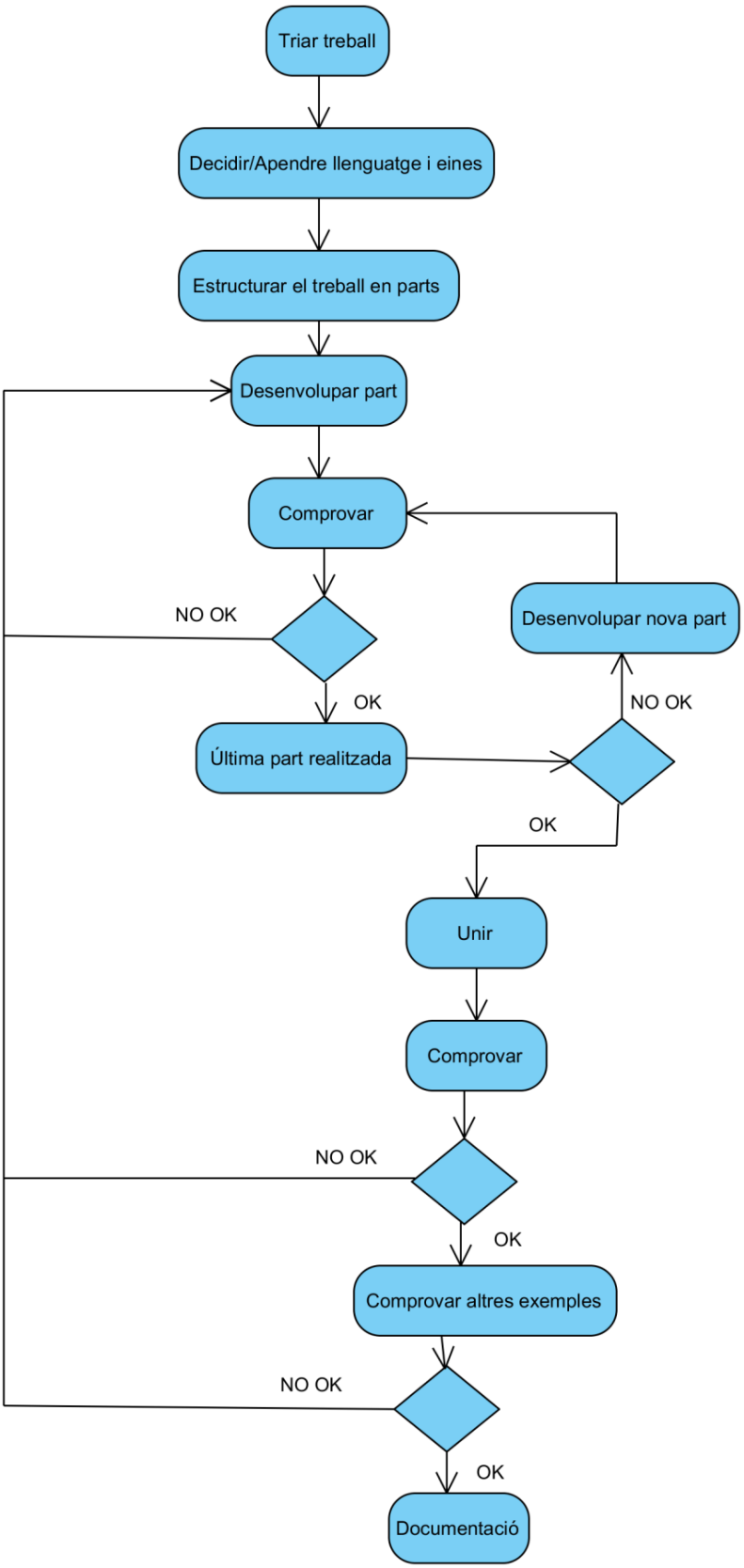

Figura 9: Diagrama d'activitat de la metodologia

4 PLANIFICACIÓ

Per a qualsevol projecte, sempre és important fer una valoració del temps dedicat. En aquest capítol es descriu una planificació que s'ha seguit durant la elaboració del projecte.

4.1 PLA DE TREBALL

Es van planificar les tasques que s'havien de realitzar, que arriben a un total de 11 tasques.

4.2 TASQUES PLANIFICADES:

4.2.1 Planificació del joc

En aquesta primera tasca es va definir el guionatge i els diferents elements d'interacció que intervenen en el joc. També es van definir les regles del joc.

4.2.2 Estudi de la competència i de les noves tecnologies

En aquesta fase es va estudiar el funcionament d'altres jocs MMORTS de l'actualitat i transportar-los a les noves tecnologies d'avui dia com nodejs i socket.io.

4.2.3 Estudi de servidor

En aquesta etapa es va estudiar el funcionament de nodejs, juntament amb les llibreries escollides per facilitar el funcionament.

4.2.4 Estudi de client

En aquesta tasca es va estudiar el funcionament de les diverses tecnologies escollides (bootstrap, backbone, ...) per mostrar la interfície a l'usuari.

4.2.5 Disseny i implementació de la base de dades

En aquesta etapa es va dissenyar i implementar l'estructura final de totes les dades del sistema.

4.2.6 Disseny i implementació del sistema de civilitzacions

En aquesta etapa es va dissenyar i implementar tot el sistema de civilitzacions del joc, que està conformat per totes les funcions del servidor (incloent també les funcions de l'APIRest) així com les vistes del client.

4.2.7 Disseny i implementació del sistema de recursos

En aquesta etapa es va dissenyar i implementar tot el sistema de recursos del joc, que està conformat per totes les funcions del servidor (incloent també les funcions de l'API Rest), on es calcula periòdicament l'augment de recursos en una ciutat així com les vistes del client.

4.2.8 Disseny i implementació del sistema d'edificis

En aquesta etapa es va dissenyar i implementar tot el sistema d'edificis del joc, que està conformat per totes les funcions del servidor per verificar si un usuari pot millorar o no els edificis i gestionar quan es duu a terme aquesta millora amb un sistema d'events. S'inclouen també totes les funcions de l'API Rest al respecte. Per la part del client es gestionen totes les vistes i els comptes enrere de les millores que s'estan efectuant.

4.2.9 Disseny i implementació del sistema de batalles

En aquesta etapa es va dissenyar i implementar la primera versió del sistema de batalles del joc, que està conformat per les funcions que decideixen el guanyador, les funcions API Rest, i les vistes del client.

4.2.10 Verificació i proves dels algoritmes desenvolupats

Aquesta tasca va consistir en comprovar que els algoritmes implementats funcionaven de la manera esperada. Per arribar a una valoració al respecte, es van aplicar casos concrets a la base de dades per comprovar que els resultats eren els esperarts en diferents casos.

4.2.11 Documentació

La documentació és una tasca constant que s'ha de portar a terme durant tot el projecte. Els diferents mètodes, funcions i idees que van apareixent durant el desenvolupament s'han de plasmar a la documentació.

4.3 TEMPS ESTIMAT

Es va plantejar que el projecte tingués una duració d'uns 8 mesos i la distribució que es va triar en un primer moment va ser la següent:

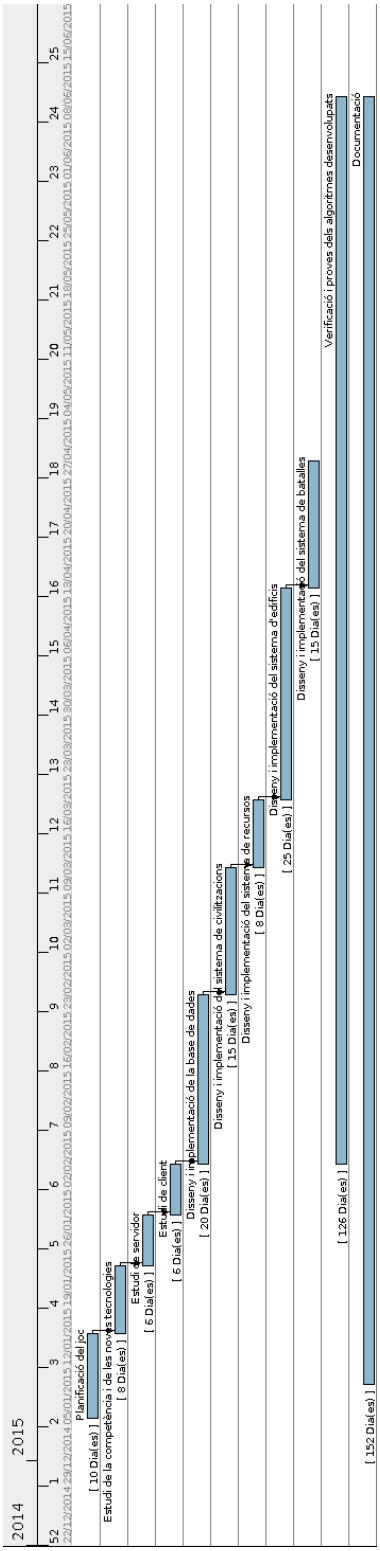

Figura 10: Diagrama de Gantt de la planificació inicial

4.4 RESULTATS ESTIMATS

S’espera crear un videojoc totalment funcional i mínimament divertit, al mateix temps que compleixi totes les expectatives i aspectes esmentats en els anteriors apartats.

5. Marc de treball i conceptes previs

En el desenvolupament de videojocs web normalment s'utilitza un servidor HTTP que gestioni les peticions de l'usuari i les redireccioni al codi de servidor, de manera que l'usuari pugui rebre la informació que necessita quan fa la petició.

A la vegada també es solen utilitzar llibreries, tant en el servidor com en el client, per facilitar i organitzar el codi, així com afegir funcionalitats que serien molt complexes fer des de zero.

S'ha de valorar en cada desenvolupament (client i servidor) quina és la millor manera de treballar i quines eines seran necessàries per assolir tots els objectius.

Per a aquest projecte la primera decisió recau sobre el llenguatge a usar en el servidor:

- PHP.
- Node.js (Javascript).

5.1 PHP

Va ser un dels primers llenguatges de programació de servidor on es podia incorporar directament en el document HTML en lloc de cridar a un arxiu extern que processi les dades. El codi és interpretat per un servidor web amb un mòdul de processador de PHP que genera la pàgina web resultant.

Pot ser usat en la majoria dels servidors web de la mateixa manera que en gairebé tots els sistemes operatius i plataformes sense cap cost. PHP es considera un dels llenguatges més flexibles , potents i d'alt rendiment coneguts fins al dia d' avui, cosa que ha atret l'interès de múltiples llocs amb gran demanda de trànsit, com Facebook, que l'utilitzen amb un framework personalitzat com a tecnologia de servidor.

Actualment és el llenguatge de servidor més usat del món (81.9% dels servidors l'utilitzen).

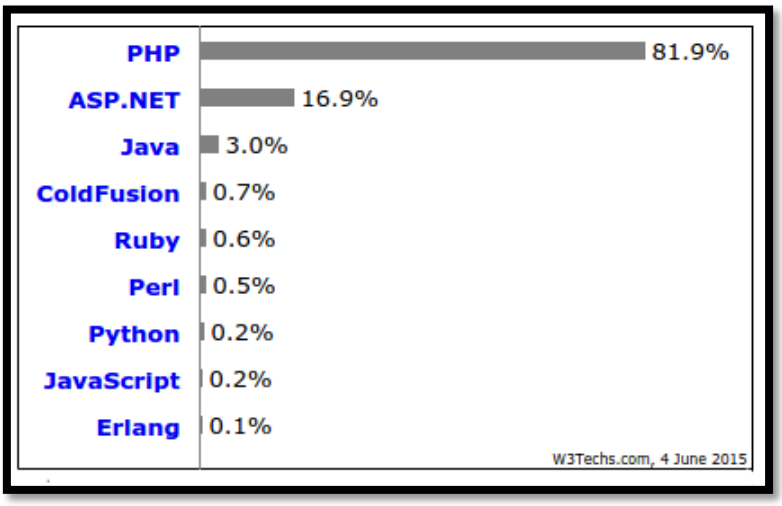

Figura 11: % d'ús dels diversos codis de servidors (4 Juny,2015)

5.2 NODE.JS (JAVASCRIPT)

Node.js és un entorn de programació en la capa del servidor (tot i que no es limita solament a servidor) basat en el llenguatge de programació ECMAScript (també anomenat JavaScript). Funciona de forma asíncrona , amb I/O de dades en una arquitectura orientada a esdeveniments i basat en el motor V8 de Google.

Va ser creat amb l'enfocament de ser útil en la creació de programes de xarxa altament escalables, com per exemple servidors web.

Juntament amb un petit conjunt de llibreries es poden crear de manera molt còmoda API Rests, connexions amb sockets, etc.

Al ser molt recent hi ha pocs servidors a nivell mundial que l'utilitzen (0.2%), però l' ús cada cop és més freqüent i, a causa de la seva potència i expressivitat, les possibilitats d'acabar sobrepassant PHP en un futur són molt elevades.

5.3 ELECCIÓ DE NODE.JS

Node.js està més enfocat al concepte d'aplicacions web, és a dir, contingut completament dinàmic, oferint una potent APIRest que sigui independent del client utilitzat.

A més a més, com a llenguatge de programació, JavaScript és més ràpid que PHP, i està completament enfocat a l'escalabilitat.

Per aquestes raons s'ha escollit utilitzar Node.js en el servidor.

6 REQUERIMENTS DEL SISTEMA

En aquest capítol seran descrits els requeriments, que expliquen a grans trets els objectius de l'aplicació, juntament amb les funcionalitats desitjades.

Dins de qualsevol aplicació apareixen dos tipus de requeriments:

- Requeriments funcionals: Descriuen quins són els serveis que oferirà la aplicació, independentment de la implementació.
- Requeriments no funcionals: Informen sobre les restriccions que venen imposades pel client o pel propi problema.

A continuació es descriuen en detall els dos tipus de requeriments comentats.

6.1 REQUERIMENTS FUNCIONALS

Els principals requeriments que ha de satisfer l'aplicació són:

- Registre i Login d'usuari.
- Creació d'una ciutat.
- Consultar i evolucionar les dues civilitzacions de la ciutat.
- Consultar els recursos d'una ciutat.
- Consultar i evolucionar els edificis d'una ciutat.
- Atacar a una altra ciutat.

Per la resta de requeriments, veure el capítol 7.

6.2 REQUERIMENTS NO FUNCIONALS

En tot projecte s'ha de prestar especial atenció a tots els aspectes que han de tenir-se en compte quan s'ha de dissenyar un sistema, més allà de la explicació funcional presentada anteriorment. Els requeriments no funcionals del sistema són aquells que fan referència a restriccions del tipus de disponibilitat dels recursos, seguretat o interfícies externes (hardware i software), entre altres. Aquestes condicions permeten executar l'aplicació sense problema.

En quan a l'aplicació implementada, s'ha de dir que, des del punt de vista de seguretat, no hi ha cap requisit de control d'accés al programa, ja que es tracta d'una aplicació on els usuaris que hi accedeixen només tenen un rol. Tampoc disposa de dades confidencials o d'alt risc, pel que no es tindrà en compte un sistema de protecció de dades.

Cal afegir que tot i que l'aplicació ha estat implementada sobre la plataforma ElementaryOs, amb qualsevol altre sistema operatiu el resultat hagués estat el mateix.

La implementació i una part de les proves de l'aplicació s'han portat a terme sobre un equip amb les següents característiques:

- Intel® Core™ i7-3630QM @2.40GHz
- ATI Radeon HD7670M
- 4 GB de memòria RAM
- ElementaryOs 0.2.1

Les proves de llarga duració s'han portat a terme sobre una RaspberryPi de les següents característiques:

- ARM1176JZF-S 700 MHz
- Broadcom VideoCore IV
- 512 de memòria RAM (compartits amb GPU)
- Raspbian 3.18

7 ESTUDIS I DECISIONS

En aquest capítol es nombren els programes i llibreries usats per realitzar aquest projecte. Es mostren les eines que han fet possible la implementació de l'aplicació, des de les més bàsiques fins a les que han servit com a suport, així com els programes necessaris per construir la documentació corresponent.

7.1 SOFTWARE SERVIDOR

En aquesta secció s'explica cada un dels programes que s'utilitzen en el servidor per al funcionament del projecte.

7.1.1 MYSQL

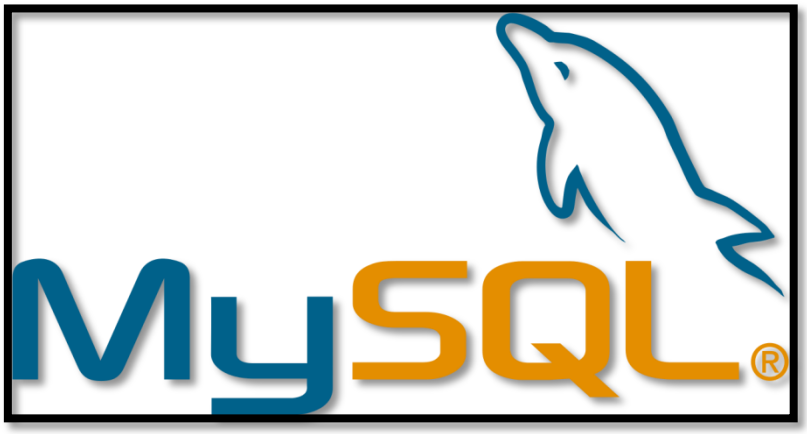

Figura 12: Logotip MySQL

MySQL és un sistema de gestió de bases de dades relacional, multifil i multiusuari amb més de sis milions d'instal·lacions. Està desenvolupat en la seva major part en ANSI C. MySQL és una base de dades molt ràpida en la lectura quan utilitza el motor no transaccional MyISAM, però pot provocar problemes d'integritat en entorns d'alta concurrència en la modificació.

En aplicacions web hi ha baixa concurrència en la modificació de dades i en canvi l'entorn és intensiu en lectura de dades , el que fa a MySQL ideal per a aquest tipus d'aplicacions.

Sigui quin sigui l'entorn en el que s'utilitzarà MySQL, és important monitoritzar per endavant el rendiment per detectar i corregir errors tant de SQL com de programació.

7.1.2 APACHE

Figura 13: Logotip Apache

El servidor HTTP Apache és un servidor web HTTP de codi obert, per a plataformes Unix, Microsoft Windows, Macintosh i altres, que implementa el protocol HTTP/1.12 i la noció de lloc virtual.

Apache presenta entre altres característiques altament configurables, bases de dades d'autenticació i negociat de contingut, però va ser criticat per la manca d'una interfície gràfica que ajudi en la configuració.

Apache té àmplia acceptació a la xarxa: des de 1996 és el servidor HTTP més usat. Va aconseguir la seva màxima quota de mercat en 2005 sent el servidor emprat en el 70% dels llocs web al món.

7.1.3 NODE.JS

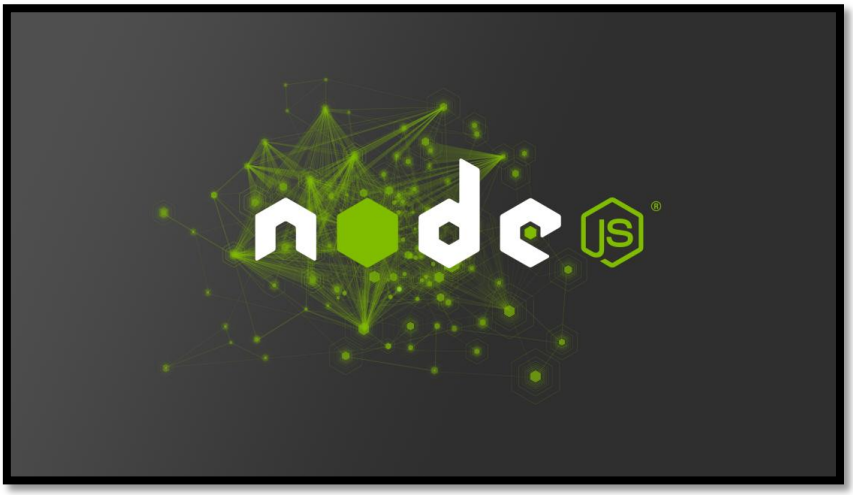

Figura 14: Logotip NodeJS

Com s'ha esmentat anteriorment en el punt 5.2, Node.js és un entorn de programació en la capa del servidor basat en el llenguatge de programació ECMAScript.

Les llibreries de Node.js que es fan servir al servidor són:

7.1.3.1 FOREVER

Forever és una llibreria global de Node.js que s'executa desde el moment en que s'engega el servidor. L'objectiu és mantenir un procés fill (el servidor web de Node.js) que s'executa de manera contínua i automàtica i reiniciar quan aquest procés fill es tanca inesperadament.

7.1.3.2 EXPRESS

Figura 15: Logotip express

Express és un framework web mínim i flexible per a Node.js que proporciona un conjunt robust de característiques necessàries per a aplications web i mòbils.

Amb una gran varietat de mètodes d'utilitat HTTP i middleware, el desenvolupament d'una APIRest potent és ràpid i fàcil.

7.1.3.3 NODE-MYSQL2

L'unió entre la base de dades MYSQL i el nostre servidor Node.js és una de les coses principal que es necessita per a l'aplicació web.

Node-mysql2 gestiona la connexió del servidor a la base de dades i permet fer consultes i operacions de forma directa o a través de vistes i processos, sempre controlant de forma automàtica típics problemes com els atacs per SQL injection.

7.1.3.4 CRYPTO

El mòdul Crypto de Node.js és un embolcall de les funcions criptogràfiques de l'OpenSSL. S'encarrega del càlcul de hash, autenticacions HMAC, i altres algoritmes de codificació.

En el servidor s'utilitza principalment per calcular el hash de les contrassenyes dels usuaris pel registre i l'autenticació.

7.1.3.5 UNDERSCORE

Figura 16: Logotip Underscore.js

Underscore proporciona més d'un centenar de funcions complexes (la majoria funcionant sobre taules) ja preparades per utilitzar, tipus map o filter.

També per al client interessa fer servir el seu mòdul intern per interpretar templates i generar l'string resultant (generalment HTML).

7.1.3.6 SOCKET.IO

Figura 17: Logotip Socket.IO

Socket.io és un mòdul de node.js que permet controlar events en temps real mitjançant connexió TCP.

A part d'escoltar i disparar events en temps real, permet l'intercanvi d'objectes JSON, cosa que el converteix en una llibreria extremadament útil en creació d'aplicacions en temps real per controlar avisos a la web, xats...

7.1.3.7 SESSION.SOCKET.IO

Aquest petit mòdul simplifica la connexió del socket al servidor, associant de forma automàtica al socket la sessió (que pertany a express) de l'usuari. D'aquesta manera es pot saber en tot moment de quin usuari es tracta i accedir a la seva sessió.

7.2 SOFTWARE CLIENT

En aquesta secció s'explica cada un dels programes que s'utilitzen en el client per al funcionament del projecte.

7.2.1 HTML5

Figura 18: Logotip HTML5

HTML5 (HyperText Markup Language, version 5) és la cinquena revisió del llenguatge bàsic de tota la web, HTML.

HTML5 estableix una sèrie de nous elements i atributs que reflecteixen l'ús típic dels nous llocs webs moderns. Alguns d'aquests elements proporcionen noves funcionalitats a través d'una interfície estandaritzada, com els elements <audio> i <video>. També s'ha millorat l'element <canvas>, capaç de renderitzar elements 3D en els navegadors més importants.

7.2.2 CSS3

Figura 19: Logotip CSS3

Cascading Style Sheets (CSS) és un llenguatge de fulls d'estil utilitzat per descriure l'aspete i el format d'un document HTML. Juntament amb HTML i Javascript, CSS és una tecnologia fonamental utilitzada per la majoria dels llocs web per crear pàgines web visualment atractives, interfícies d'usuari per a aplicacions web, i interfícies d'usuari per a moltes aplicacions mòbils.

7.2.3 JAVASCRIPT

Figura 20: Logotip JavaScript

JavaScript (abreujat comunament "JS") és un llenguatge de programació interpretat, dialecte de l'estàndard ECMAScript. Es defineix com orientat a objectes basat en prototips, imperatiu, dèbilment tipat i dinàmic.

S'utilitza principalment en la forma del costat del client (client-side), implementat com a part d'un navegador web permetent millores en la interfície d'usuari i pàgines web dinàmiques. L'ús en aplicacions externes a la web, per exemple en documents PDF, aplicacions d'escriptori (majoritàriament widgets) també és significatiu.

JavaScript es va dissenyar amb una sintaxi similar al C, encara que adopta noms i convencions del llenguatge de programació Java. No obstant això Java i JavaScript no estan relacionats i tenen semàntiques i propòsits diferents.

Tots els navegadors moderns interpreten el codi JavaScript integrat en les pàgines web. Per interactuar amb una pàgina web es proveeix al llenguatge JavaScript d'una implementació del Document Object Model (DOM). Tradicionalment es venia utilitzant en pàgines web HTML per realitzar operacions i únicament en el marc de l'aplicació client, sense accés a funcions del servidor. JavaScript s'interpreta en l'agent d'usuari, al mateix temps que les sentències van descarregant junt amb el codi HTML.

Les principals llibreries de JavaScript usades en el projecte són:

7.2.3.1 JQUERY

Figura 21: Logotip jQuery

JQuery és una biblioteca de JavaScript que permet simplificar la manera d'interactuar amb els documents HTML, manipular l'arbre DOM, manejar esdeveniments, desenvolupar animacions i afegir interacció amb la tècnica AJAX a les pàgines web.

JQuery és la biblioteca de JavaScript més utilitzada i igual que altres biblioteques, ofereix una sèrie de funcionalitats basades en JavaScript que d'altra manera requeririen de molt més codi, és a dir, amb les funcions pròpies d'aquesta biblioteca s'aconsegueixen grans resultats en menys temps i espai.

7.2.3.2 BOOTSTRAP

Figura 22: Logotip Bootstrap

Twitter Bootstrap és un framework o conjunt d'eines de programari lliure per a disseny de llocs i aplicacions web . Conté plantilles de disseny amb tipografia, formularis, botons, quadres, menús de navegació i altres elements de disseny basat en HTML i CSS , així com extensions de JavaScript opcionals addicionals.

Una de les característiques principal és les eines que dona per poder preparar molt més fàcilment elements responsive a la pàgina per la navegació desde dispositius mòbils.

En aquest projecte no s'utilitza bootstrap en la seva totalitat, sinó que s'aprofiten alguns dels estils que implementa (com en els botons) i en aprofitar l'apilat de caixes automàtic segons la resolució de pantalla.

7.2.3.3 UNDERSCORE

Comentat anteriorment a l'apartat 7.1.3.5.

A més d'utilitzar-se per les funcions que implementa i el seu sistema de templates, és un requisit intern de la llibreria Backbone.

7.2.3.4 BACKBONE

Figura 23: Logotip BackBone.js

Backbone.js és una llibreria JavaScript que proporciona una estructura a les aplicacions web a través de la implementació de models amb identificador i events personalitzables, col·leccions amb una API per al control i vistes amb gestió declarativa d'events.

En el projecte principalment s'utilitza el mòdul d'enrutament, el qual permet gestionar totes les rutes de la web singlepage i el control sobre el redireccionament en cas d'accedir a una url invàlida.

7.2.3.5 SOCKET.IO

Explicat a l'apartat 7.1.3.6. Connecta el client amb el servidor per poder traspassar missatges en temps real.

7.2.3.6 COUNTDOWN

Figura 24: Exemple del plugin The Final Countdown

The Final Countdown és un petit plugin que duu a terme un compte enrere amb el format predeterminat.

En el projecte estalvia la gestió dels comptes enrere dels edificis en procés de millorar.

7.3 PROGRAMACIÓ I DOCUMENTACIÓ

En aquesta secció s'explica cada un dels programes que s'han utilitzat durant el desenvolupament del projecte.

7.3.1 GIT

Figura 25: Logotip GIT

Git és un programari de control de versions pensant en l'eficiència i la fiabilitat del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font.

Al principi, Git es va pensar com un motor de baix nivell sobre el qual altres poguessin escriure la interfície d'usuari o front end com Cogito o StGIT . No obstant això, Git s'ha convertit des de llavors en un sistema de control de versions amb funcionalitat plena. Hi ha alguns projectes de molta rellevància que ja fan servir Git, en particular, el grup de programació del nucli Linux .

En aquest projecte s'ha utilitzat un repositori Git funcionant en un servidor bitbucket per poder tenir una revisió de les versions del codi, així com poder actualitzar-lo de manera àgil a la RaspberryPi.

7.3.2 PHPMYADMIN

Figura 26: Logotip phpMyAdmin

PhpMyAdmin és una eina de programari lliure escrita en PHP, amb la intenció de controlar l'administració de MySQL a través del web. PhpMyAdmin és compatible amb una àmplia gamma d'operacions en MySQL, MariaDB i plugim.

S'utilitza amb freqüència per dur a terme operacions (gestió de bases de dades, taules, columnes, relacions, índexs, usuaris, permisos, etc) a través de la interfície d'usuari.

En aquest projecte ha estat molt útil per poder fer proves i comprovacions ràpides localment mentre es programa un mòdul.

7.3.3 INTELLIJ IDEA

Figura 27: Logotip IntelliJ Idea

Intellij IDEA és un Entorn de desenvolupament interactiu (IDE) per al desenvolupament de programes informàtics. Proporciona les eines bàsiques per programar i debuggar un projecte client/servidor amb node.js.

En aquest projecte s'ha utilitzat la versió de prova del programa.

7.3.4 GOOGLE CHROME

Figura 28: Logotip Google Chrome

Google Chrome és un navegador web desenvolupat per Google.

Tot i que l'aplicació s'ha provat en diversos navegadors i entorns, s'ha utilitzat durant el desenvolupament principal, degut als plugins que ajuden a testejar alguns aspectes concrets:

7.3.4.1 POSTMAN

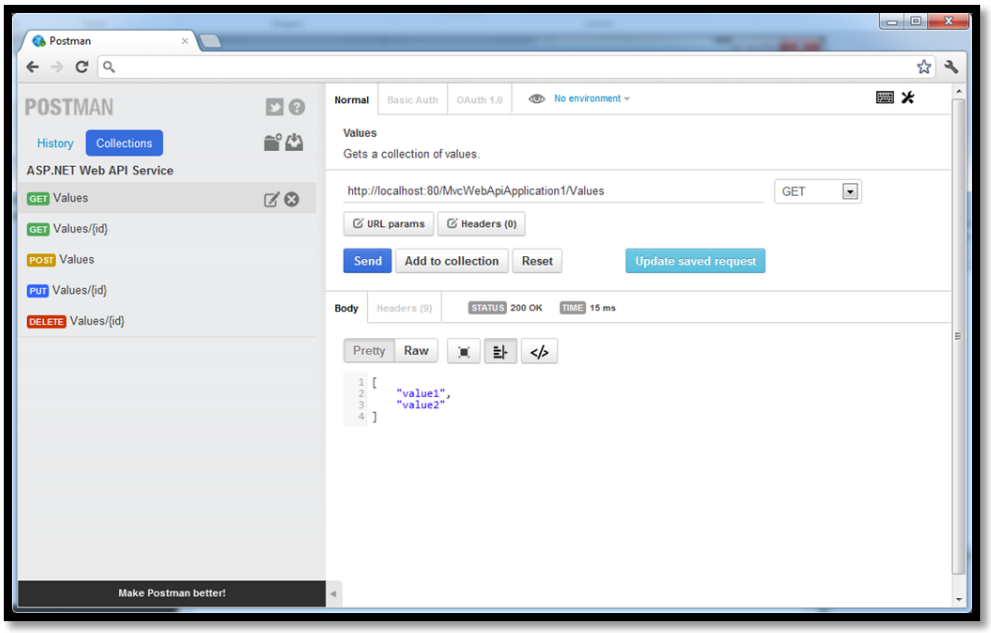

Figura 29: Exemple d'una petició desde la finestra de POSTMan

Postman és un plugin per Google Chrome que permet generar diverses peticions GET, POST, PUT, etc. amb els paràmetres que l'usuari hi insereix.

És molt útil en el moment de programar les funcions APIRest, ja que no és necessari improvisar cap vista en un client de prova per poder-les testejar.

7.3.4.2 MOBILE/RESPONSIVE WEB DESIGN TESTER

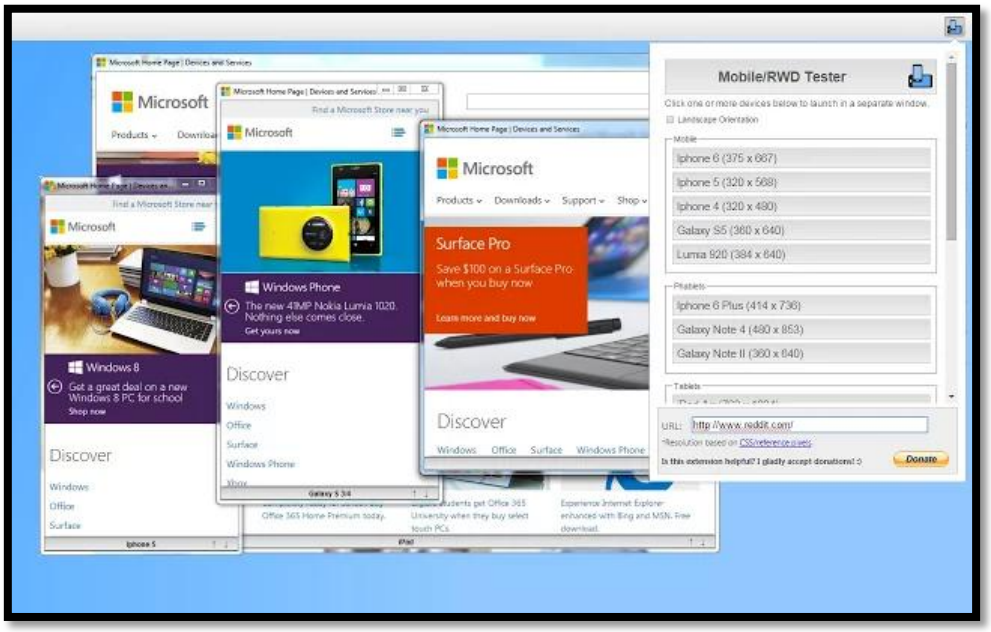

Figura 30: Exemple d'ús del plugin Mobile/RWD Tester

Mobile/RWD Tester és un plugin per Google Chrome que des deel menú permet canviat la mida del navegador a l'equivalent al de diferents tipus de pantalles de mòbils i tablets.

Tot i que generalment els aspectes de responsive es poden provar només canviant manualment la mida de la finestra, hi ha algunes configuracions de mòbils petits que no es poden recrear a mà sense l'ajuda del plugin.

7.3.5 MYSQL WORKBENCH

Figura 31: Pantalla d'inici de MySQL Workbench

MySQL Workbench és una eina visual de disseny de bases de dades que integra desenvolupament de programari, administració de bases de dades, disseny de bases de dades i creació/manteniment per a un sistema de base de dades MySQL.

En aquest projecte ha estat de gran ajuda en el procés de disseny del diagrama E/R de la base de dades i la autogeneració del codi MYSQL del joc.

7.3.6 VISUAL PARADIGM

Figura 32: Logotip Visal Paradigm

Visual Paradigm per UML és una eina CASE que suporta el modelatge mitjançant UML i proporciona assistència als analistes, enginyers de software i desenvolupadors durant tots els passos del cicle de vida del desenvolupament d'un software.

Els avantatges que proporciona Visual Paradigm for UML són:

- Dibuix. Facilita el modelatge UML ja que proporciona eines específiques per això. Això també permet la estandardització de la documentació.

- Correcció sintàctica. Controla que el model UML sigui correcte.

- Coherència entre diagrames. Al disposar de un repositori comú, es possible visualitzar el mateix element en diferents diagrames, evitant duplicitats.

- Integració amb altres aplicacions. Es pot integrar amb altres aplicacions, com eines ofimàtiques, la qual cosa augmenta la productivitat.

- Treball multiusuari. Permet el treball en grup, proporcionant eines de compartició de treball.

- Reutilització. Facilita la reutilització, ja que es disposa d'una eina centralitzada on es troben els models utilitzats per altres projectes.

- Generació de codi. Permet generar codi de forma automàtica reduint el temps de desenvolupament i evitant errors en la codificació.

- Generació de informes. Permet generar diversos informes a partir de la informació introduïda a la eina.

7.3.7 GANTT PROJECT

Figura 33: Logotip GanttProject

Gantt project és una aplicació de software lliure que permet realitzar diagrames de Gantt.

Un diagrama de Gantt és una eina gràfica que té per objectiu mostrar el temps de dedicació previst per les diferents tasques o activitats al llarg del desenvolupament d'un projecte.

El diagrama de Gantt no indica les relacions existents entre activitats, però la posició de cada tasca al llarg del temps respecte les altres fa que es puguin identificar aquests relaciones i dependències.

7.3.8 OFFICE ONLINE

Figura 34: Logotip Office Online

Office Online és una versió gratuïta al web del conjunt d'aplicacions de Microsoft Office. Inclou Word Online, Excel Online, PowerPoint Online, i OneNote Online. En aquesta versió d'Office s'han reduït les funcionalitats respecte al programari que s'instal·la en el disc dur.

Degut la facilitat d'utilitzar-lo en qualsevol tipus de sistema operatiu, s'ha anat escrivint la memòria desde Word Online.

8 ANÀLISI I DISSENY DEL SISTEMA

En aquest apartat es descriurà la temàtica i com ha de ser el videojoc, i s'exposarà a nivell conceptual i sense entrar en cap moment en la implementació, els elements més importants per la comprensió del disseny del videojoc i del projecte.

8.1 DESCRIPCIÓ GENERAL

El videojoc programat es basa en els MMORTS que es van popularitzar a finals del segle passat. En aquest tipus de joc, cada jugador controla una civilització a través de diversos menús, i l'objectiu és anar millorant aquesta civilització i evolucionar-la a partir de l'obtenció de diversos recursos. Podem veure un exemple clàssic de la història dels MMORTS en la Figura 35.

Figura 35: Travian (2004)

8.2 HISTORIA I AMBIENTACIÓ

En un món on existeixen els déus i la màgia, apareixen milers de poblats humans que, a partir dels seus recursos, habilitats i creences, evolucionaran en diversos tipus de civilitzacions.

Com sempre passa amb els humans, les civilitzacions lluitaran entre elles per obtenir els recursos de les altres i formaran pactes d'intercanvis de recursos per continuar evolucionant amb l'objectiu de ser la major força del món.

8.3 DISSENY DEL FUNCIONAMENT

El joc consisteix en arribar a evolucionar el màxim possible la seva ciutat.

Així doncs, per evolucionar-la al màxim, el jugador haurà de millorar els edificis per augmentar la producció de recursos/hora. El jugador també podrà lluitar contra altres usuaris per robar-los els recursos.

Cada ciutat pot pertànyer a dues civilitzacions simultàniament. Quan el jugador compleix el requisit mínim de recursos, pot escollir cap a quina de les diverses possibilitats pot evolucionar la civilització.

Depenguent de quines civilitzacions es trobin associades a la ciutat, els edificis necessitaran un tipus de recurs o un altre per millorar-se.

8.3.1 Exemple de prova

L'exemple de prova són les dades introduïdes al sistema per tenir un referent mínim del funcionament. Representa una part del que serà el resultat final del producte que ajudarà a fer totes les proves necessàries.

En l'exemple de prova tindrem un conjunt de civilitzacions, tipus de recursos i edificis (fins a nivell 3).

8.3.1.1 Civilitzacions

L'exemple de prova del sistema inclou les següents civilitzacions:

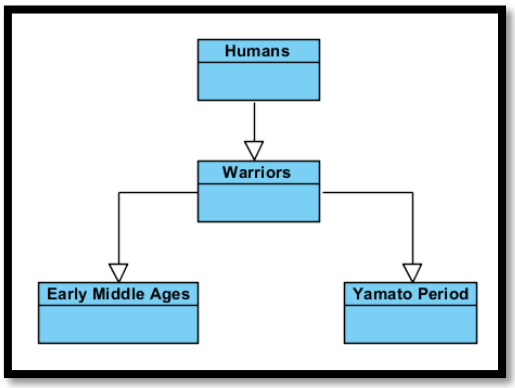

Figura 36: Arbre de civilitzacions de l'exemple de prova

8.3.1.1.1 Humans

Els humans són la civilització principal de les noves ciutats. En l'exemple de prova només pot evolucionar a Warriors.

"Els humans han deixat les cavernes per evolucionar cap a una nova civilització. El futur que els hi espera depen de tú".

Figura 37: Fons civilització Humans

8.3.1.1.2 Warriors

Primera evolució dels humans. En l'exemple de prova el jugador pot evolucionar la civilització a Early Middle Ages o Yamato Period.

"Aquesta civilització ha demostrat tanta predilecció per a l'art de la guerra, que ha començat a fabricar les seves primeres armes i armadures".

Figura 38: Fons civilització Warriors

8.3.1.1.3 Early Middle Ages

Una de les possibles evolucions dels humans guerrers. En l'exemple de prova no té cap més branca.

"L'economia feudal és rural, basada en la terra, amb una mínima divisió de la feina i escassos intercanvis comercials.

Les estructures polítiques han canviat. Comandar l'exèrcit i administrar justícia recau sobre el rei".

Figura 39: Fons civilització Early Middle Ages

8.3.1.1.4 Yamato period

Una de les possibles evolucions dels humans guerrers. En l'exemple de prova no té cap més branca.

"La població va créixer amb més rapidesa, i la política descentralitzada s'ha canviat per una patriarcal i militar.

Els diferents clans es disputen el poder mitjançant les armes, buscant aliances amb altres guerrers. Amb l'apirició d'un clan imperial, cada guerrer ha passat a desenvolupar una funció dintre de Yamato, sorgint la figura del Shogun, càrrec que se li atorga al general amb el deure de sofocar l'amenaça dels pobles bàrbars existents".

Figura 40: Fons civilització Yamato Period

8.3.1.2 Tipus de recursos

A l'exemple de prova es poden obtenir diversos tipus de recursos, uns només començar la partida, i uns altres a mesura que s'evoluciona la civilització.

8.3.1.2.1 Recursos primaris:

Aquests recursos es poden obtenir amb un edifici generador de recursos.

8.3.1.2.1.1 Pedra

És un material bàsic de construcció. Es pot obtenir des de la primera civilització.

Figura 41: Recurs - Pedra

8.3.1.2.1.2 Aliment

Indispensable per a que la població pugui sobreviure. Es pot obtenir des de la primera civilització.

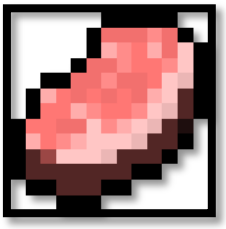

Figura 42: Recurs - Aliment

8.3.1.2.1.3 Fusta

Juntament amb la pedra, és un material bàsic de construcció. Es pot obtenir des de la primera civilització.

Figura 43: Recurs - Fusta

8.3.1.2.1.4 Or

Material preciós, freqüentment utilitzat com a moneda d'intercanvi. Es pot obtenir des de la primera civilització.

Figura 44: Recurs - Or

8.3.1.2.1.5 Ferro

Material bàsic, freqüentment utilitzat en la fabricació d'armes, estructures i altres objectes. Per poder obtenir-ne de forma automàtica cal evolucionar a la civilització Early Middle Ages.

Figura 45: Recurs - Ferro

8.3.1.2.1.6 Seda

Fibra natural formada de fibres, freqüentment utilitzat per fabricar diversos tipus de tela. Per poder obtenir-ne de forma automàtica cal evolucionar a la civilització Yamato Period.

Figura 46: Recurs - Seda

8.3.1.2.2 Recursos secundaris:

Aquests recursos es poden obtenir a partir d'una fàbrica, és a dir, un tipus especial d'edifici que transforma una part dels guanys per hora d'un o més recursos en un de nou. D'aquesta manera es poden fabricar coses més complexes com armes o objectes.

8.3.1.2.2.1 Armes

Eines d'agressió útils per a la caça i l'autodefensa. Es poden fabricar a partir de la civilització Warriors.

Figura 47: Recurs - Armes

8.3.1.3 Edificis

Per a l'exemple de prova s'ha reaprofitat l'aspecte de tots els edificis, tot i que en el resultat final cada edifici tindria el seu propi disseny.

Figura 48: Imatge usada en els edificis

8.3.1.3.1 Edificis generadors de recursos

Els edificis generadors de recursos són els que, simplement pagant una certa quantitat de recursos, es millora la quantitat de recurs/h que s'obté d'ell.

8.3.1.3.1.1 Granja d'ovelles

Una granja d'ovelles bàsica. Genera aliment per a la població.

Nivell	Preu millorar	Genera(h)
1	50 aliment 25 fusta	50 aliment
2	100 aliment 80 fusta	70 aliment
3	-	100 aliment

8.3.1.3.1.2 Pedrera

Explotació minera a cel obert d'on s'obté pedra.

Nivell	Preu millorar	Genera(h)
1	60 fusta	20 pedra
2	110 fusta	40 pedra
3	-	80 pedra

8.3.1.3.1.3 Serradora

Instal·lació artesanal dedicada a la serrada de fusta.

Nivell	Preu millorar	Genera(h)
1	40 pedra	30 fusta
2	100 pedra	80 fusta
3	-	110 fusta

8.3.1.3.1.4 Mina d'or

Mina d'on s'extreu or del terra.

Nivell	Preu millorar	Genera(h)
1	40 pedra 40 fusta	30 or
2	100 pedra 100 fusta	60 or
3	-	100 or

8.3.1.3.1.5 Mina de ferro

Mina d'on s'extreu ferro del terra.

Nivell	Preu millorar	Genera(h)
1	100 fusta 100 or	50 ferro
2	150 fusta 150 or	70 ferro
3	-	110 ferro

8.3.1.3.1.6 Granja de cucs de seda

En elles s'aplica un conjunt de tècniques per produir capolls i, amb ells , la seda com a producte tèxtil per processar.

Nivell	Preu millorar	Genera(h)
1	120 aliment 100 or	50 seda
2	200 aliment 150 or	70 seda
3	-	110 seda

8.3.1.3.2 Fàbriques

Els edificis de tipus fàbrica són els que, a part de tenir un cost en el moment de millorar, a cada nivell consumeixen un o varis materials per hora per produir-ne un de nou.

8.3.1.3.2.1 Armeria

Forja on es fabriquen armes a partir de matèria prima.

Nivell	Preu millorar	Cost producció (h)	Genera(h)
1	120 fusta 120 pedra	10 pedra 10 or	40 armes
2	200 fusta 200 pedra	20 pedra 20 or	80 armes
3	-	30 pedra 30 or	110 armes

8.3.2 Interfície d'usuari

Per poder gestionar la seva ciutat, el jugador tindrà en tot moment una sèrie de menús tal com es mostra a la figura 48.

Figura 49: Pantalla inicial de la ciutat amb els menús a esquerra i dreta

El menú de la interfície està formada pels següents elements:

1. Accedir a la pantalla principal de la ciutat.
2. Accedir a la vista de les civilitzacions de la ciutat.
3. Accedir a la vista de recursos de la ciutat.
4. Accedir a la vista d'edificis (recursos primaris).
5. Accedir a la vista de fàbriques (recursos secundaris).
6. Accedir a la vista d'atacs.

8.4 IDENTIFICACIÓ DELS ACTORS

A continuació s'identifiquen els actors de l'aplicació. Un actor és una entitat externa (persona, sistema, subsistema...) que interactua amb els sistema interpretant un determinat rol o estat.

Ara mateix en el videojoc no hi ha cap tipus de manteniment, al tractar-se d'un programa en el qual no existeix la distinció entre els possibles usuaris que el poden utilitzar. D'aquesta manera, podem deduir la existència de dos actors:

- L'usuari, que interactuarà en tot moment amb el sistema.
- El servidor, que s'encarregarà d'actualitzar les dades temporals.

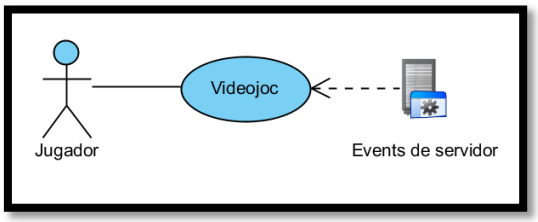

Figura 50: Identificació dels actors

Com es pot veure en la Figura 50, l'esquema resultant és senzill ja que de moment no disposem de rol d'administrador ni de manteniment (veure apartat de treball futur).

8.5 CASOS D'ÚS

8.5.1 Inici

Quan el jugador entra a l'aplicació, el sistema comprova si té una sessió guardada. Si en té, el sistema entra directament a “Veure ciutat”, en cas contrari el redirecciona a “Login Registre”.

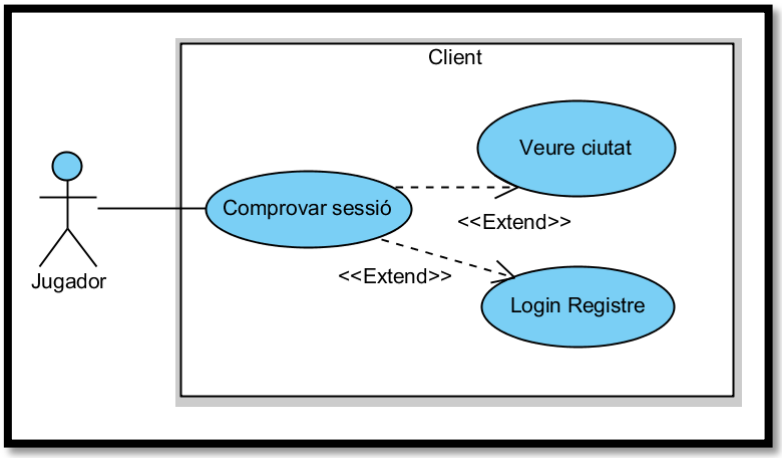

Figura 51: Diagrama de cas d’ús d’inici

8.5.2 Login Registre

Quan el jugador entra a aquest cas d'ús, es carrega la vista de Login/Registre. Un cop loguejat/registrat correctament redirecciona a “Veure ciutat”.

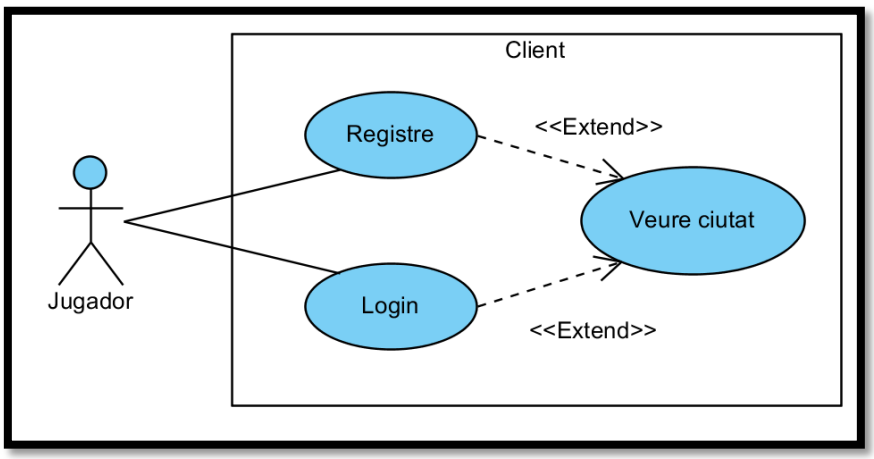

Figura 52: Diagrama de cas d’ús de Login Registre

8.5.3 Mostrar menú principal

Aquest cas d'ús representa el menú del joc presentat al punt 8.3.2. Per això de sde la resta de casos d'ús es pot accedir al cas d'ús del menú principal.

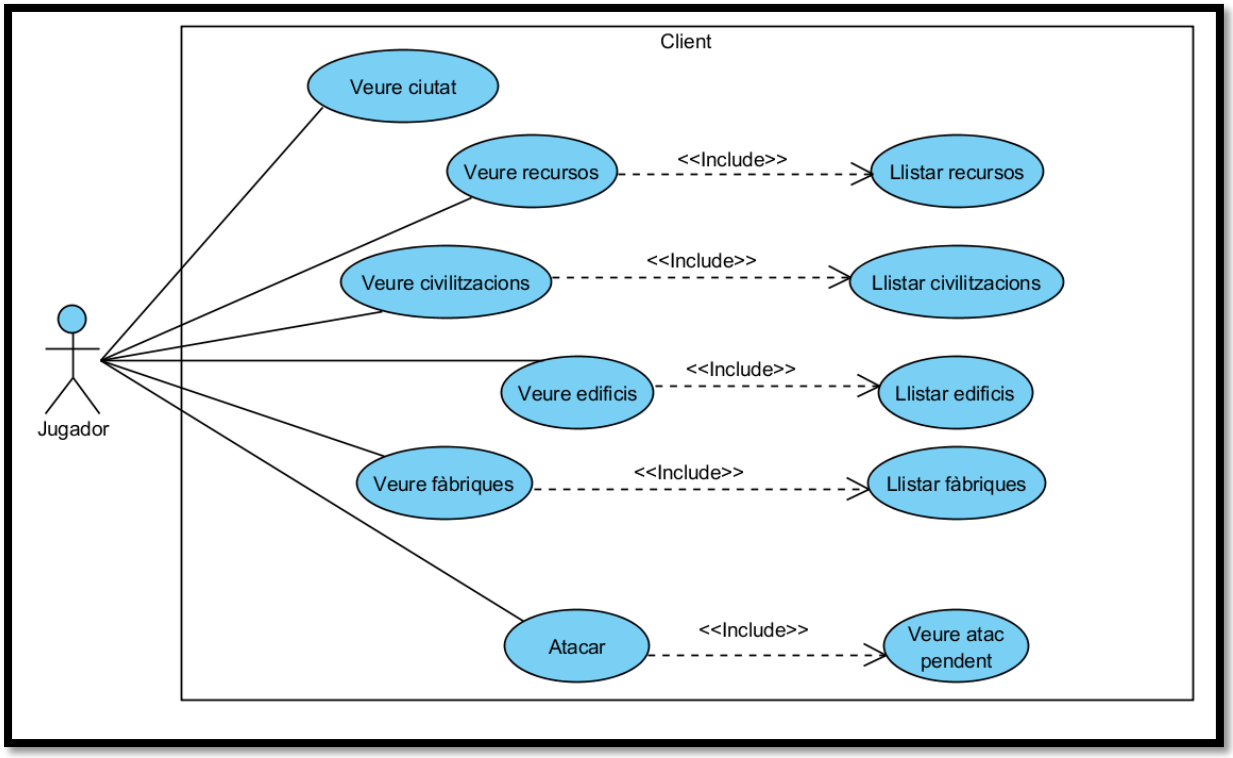

Figura 53: Diagrama de cas d'ús de Mostrar menú principal

8.5.4 Veure ciutat

Aquest cas d'ús mostra la pantalla principal de la ciutat. Mostra informació bàsica a l'úsuari sobre els recursos dels que disposa. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació.

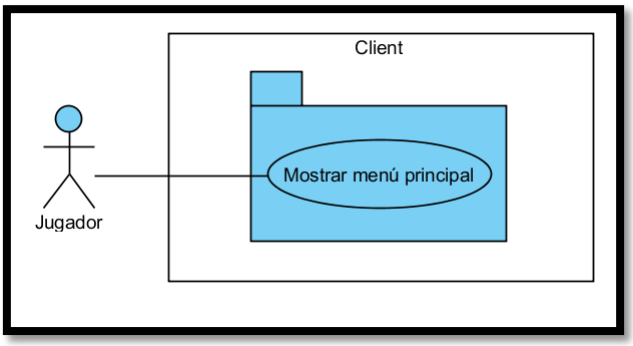

Figura 54: Diagrama de cas d'ús de Veure ciutat

8.5.5 Veure recursos

Aquest cas d'ús mostra un llistat de tots els recursos existents, amb la descripció, la quantitat de la que el jugador en disposa, i la producció/hora. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació.

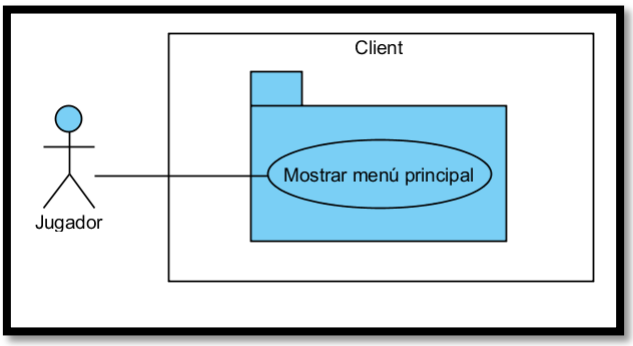

Figura 55: Diagrama de cas d'ús de Veure recursos

8.5.6 Veure edificis

Aquest cas d'ús mostra un llistat dels edificis de la ciutat, amb la descripció, nivell, cost per millorar i beneficis per fer-la. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació. Un cop es comença a millorar un edifici es mostra el compte enrere per a que s'acabi.

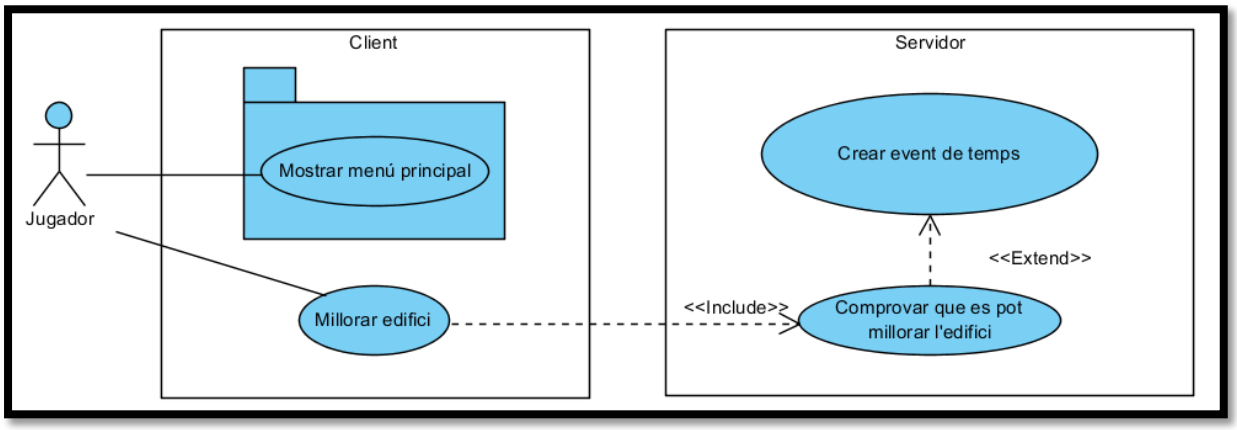

Figura 56: Diagrama de cas d'ús de Veure edificis

8.5.7 Veure fàbriques

Aquest cas d'ús mostra un llistat de les fàbriques de la ciutat, amb la descripció, nivell, cost per millorar i beneficis/costos per fer-la. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació.

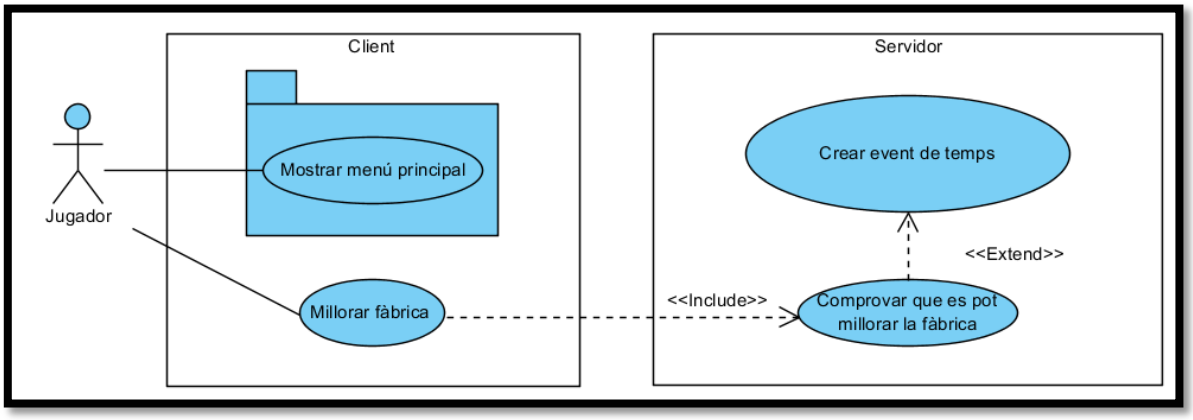

Figura 57: Diagrama de cas d'ús de Veure fàbriques

8.5.8 Veure civilitzacions

Aquest cas d'ús mostra la civilització principal i la civilització secundària de la ciutat, juntament amb les descripcions i les possibilitats de millora per cada una. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació.

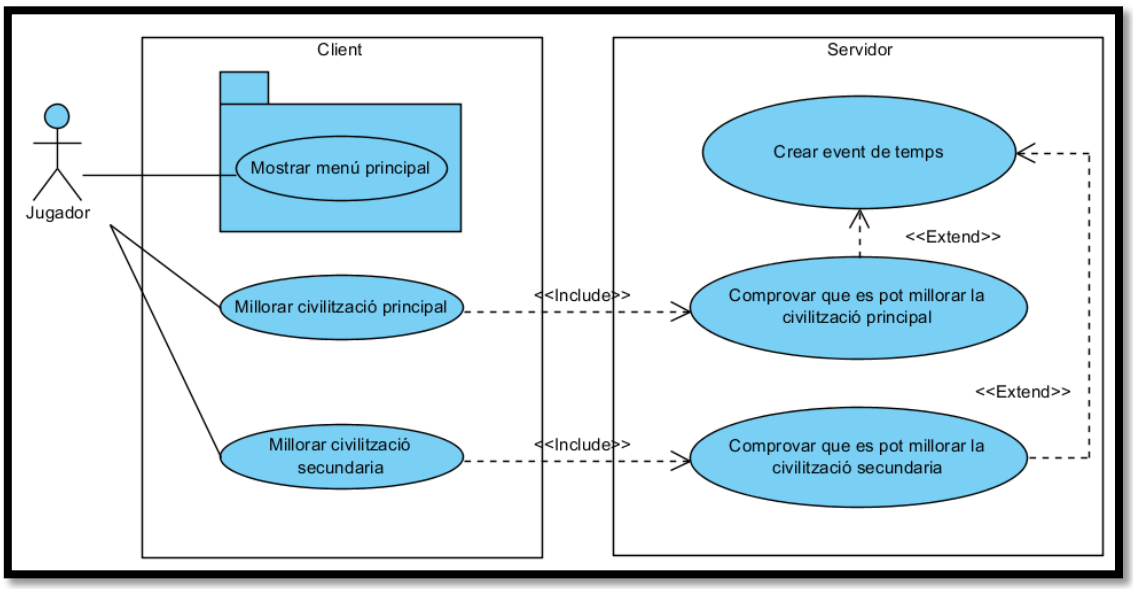

Figura 58: Diagrama de cas d'ús de Veure civilitzacions

8.5.9 Veure atacs

Aquest cas d'ús mostra l'atac en procés o la possibilitat d'efectuar-ne un de nou. Inclou el menú principal, d'on es pot anar a la resta de l'aplicació.

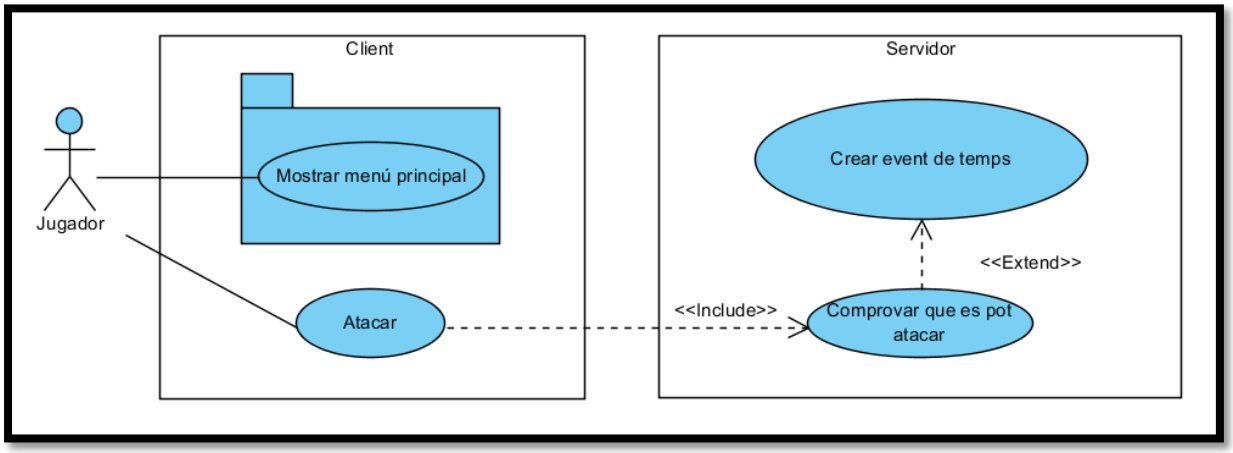

Figura 59: Diagrama de cas d'ús de Veure atacs

8.5.10 Events servidor

Aquest cas d'ús mostra les interaccions que tenen els events del servidor sobre el sistema. Inclou accions com gestionar l'increment de recursos cada hora o disparar els events de temps quan s'activa el timeout.

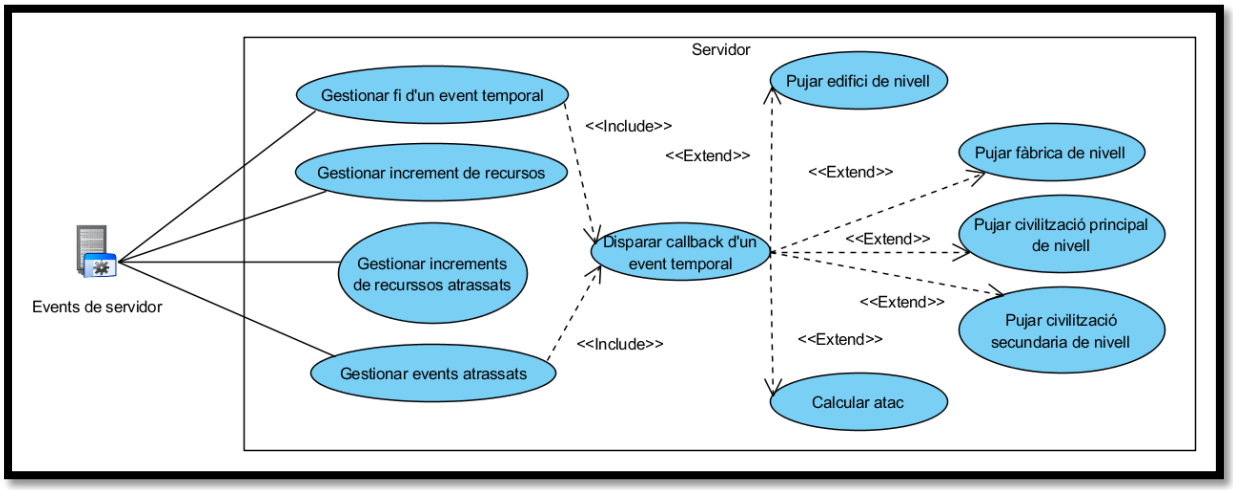

Figura 60: Diagrama de cas d'ús d'Events servidor

8.6 FITXES DE CASOS D'ÚS

8.6.1 INICI

Inici	
Actor	Jugador
Versió	1.0
Descripció	Redirecciona el jugador al cas d'ús corresponent quan entra a l'aplicació.
Precondició	-
Flux principal	1 – Comprovar la sessió de l'usuari. 2 – Si l'usuari té una sessió vàlida. 2.1 – Veure ciutat. 3 – Altrament . 3.1 – Login Registre.
Flux alternatiu	-
Post condició	L'usuari és redireccionat al cas d'ús corresponent.

8.6.2 LOGIN REGISTER

Login Register	
Actor	Jugador
Versió	1.0
Descripció	Mostra la pantalla de registre/login i passa al cas d'ús corresponent.
Precondició	Usuari no té sessió.
Flux principal	1 – Apareix la pantalla inicial de registre / login. 2 – Si l'usuari clica a registre. 2.1 – Registre. 3 – Altrament si clica a login. 3.1 -Login. 4 – Si registre o login han acabat correctament. 4.1 – Veure ciutat.
Flux alternatiu	2.1 – Error en el registre. 2.1.1 – Tornar al punt 1. 3.1 – Error en el login. 3.1.1 – Tornar al punt 1.
Post condició	L'usuari ha entrat al sistema.

8.6.3 REGISTRE

Registre	
Actor	Jugador
Versió	1.0
Descripció	Registra a l'usuari al sistema
Precondició	Usuari no té sessió
Flux principal	1 – Comprova que l'email és correcte i no existeix. 2 – Comprova que nick és correcte i no existeix. 3 – Comprova que la contrassenya és correcta. 4 – Comprova que el nom de la ciutat és correcte.
Flux alternatiu	1 – Si email és incorrecte o ja existeix. 1.1 – Sortir. 2 – Si nick és incorrecte o ja existeix. 2.1 – Sortir. 3 – Si contrassenya és incorrecta. 3.1 – Sortir. 4 – Si el nom de la ciutat és incorrecte. 4.1 – Sortir.
Post condició	S'ha retornat error o ok.

8.6.4 LOGIN

Login	
Actor	Jugador
Versió	1.0
Descripció	Logueja l'usuari al sistema
Precondició	Usuari no té sessió
Flux principal	1 – Comprova que nick existeix. 2 – Comprova que el hash de la contrassenya existeix pel nick.
Flux alternatiu	1 – Si el nick no existeix. 1.1 – Sortir. 2 – Si el hash de la contrassenya no correspon amb la del nick. 2.1 – Sortir.
Post condició	S'ha retornat error o ok.

8.6.5 VEURE CIUTAT

Veure ciutat	
Actor	Jugador
Versió	1.0
Descripció	Mostra una imatge general de la ciutat.
Precondició	Usuari té sessió
Flux principal	1 – Mostrar imatge de la civilització principal. 2 – Mostrar resum de recursos de la ciutat. 3 – Mostrar menú principal.
Flux alternatiu	-
Post condició	S'ha mostrat la vista principal de la ciutat amb el menú principal.

8.6.6 VEURE RECURSOS

Veure recursos	
Actor	Jugador
Versió	1.0
Descripció	Mostra la vista de recursos
Precondició	Usuari té sessió
Flux principal	1 – Mostrar menú principal. 2 – Llistar recursos.
Flux alternatiu	-
Post condició	S'han llistat tots els recursos de forma detallada amb el menú principal.

8.6.7 LLISTAR RECURSOS

Llistar recursos	
Actor	Jugador
Versió	1.0
Descripció	Fa un llistat de tots els recursos existents, amb descripció i quantitat que posseeix la ciutat,
Precondició	Usuari té sessió
Flux principal	1 – Obtenir llistat de recursos amb quantitat. 2 – Montar vista del llistat de recursos.
Flux alternatiu	-
Post condició	S'han llistat els recursos

8.6.8 VEURE EDIFICIS

Veure edificis	
Actor	Jugador
Versió	1.0
Descripció	Mostra la vista dels edificis
Precondició	Usuari té sessió
Flux principal	1 – Mostrar menú principal. 2 – Llistar edificis.
Flux alternatiu	-
Post condició	S'han llistat tots els edificis de forma detallada amb el menú principal.

8.6.9 LLISTAR EDIFICIS

Llistar edificis	
Actor	Jugador
Versió	1.0
Descripció	Fa un llistat de tots els edificis de la ciutat, amb descripció, nivell, i informació sobre la millora.
Precondició	Usuari té sessió
Flux principal	1 – Obtenir llistat d'edificis amb nivell i si es poden millorar. 2 – Montar vista del llistat d'edificis.
Flux alternatiu	-
Post condició	S'han llistat els edificis

8.6.10 MILLORAR EDIFICI

Millorar edifici	
Actor	Jugador
Versió	1.0
Descripció	Posa en cua la millora d'un edifici d'una ciutat.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que l'edifici pertany a la civilització. 2 – Si l'edifici pertany a la civilització. 2.1 – Comprovar que es pot millorar l'edifici. 2.2 – Si l'edifici es pot millorar. 2.2.1 – Actualitza vista amb el compte enrere.
Flux alternatiu	1 – Si l'edifici no pertany a la civilització. 1.1 – Error. 2.2 – Si l'edifici no es pot millorar. 2.2.1 – Sortir.
Post condició	Si es pot millorar l'edifici, s'ha generat el compte enrere per a que s'acabi de millorar l'edifici.

8.6.11 COMPROVAR QUE ES POT MILLORAR L'EDIFICI

Comprovar que es pot millorar l'edifici	
Actor	Jugador
Versió	1.0
Descripció	Comprova si es pot millorar l'edifici, i en cas afirmatiu genera l'event de temps.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que l'edifici pertany a la civilització. 2 – Comprovar que l'usuari disposa dels recursos per millorar-lo. 3 – Restar els recursos necessàris per la millora a l'usuari. 4 – Obtenir el temps de la millora. 5 – Crear event de temps.
Flux alternatiu	1 – Si l'edifici no pertany a la civilització. 1.1 – Retornar error. 2 – Si l'usuari no disposa dels recursos necessàris. 2.1 – Retornar error.
Post condició	Si es pot millorar l'edifici, s'ha generat l'event de temps i s'han restat els recursos a l'usuari.

8.6.12 VEURE FÀBRIQUES

Veure fàbriques	
Actor	Jugador
Versió	1.0
Descripció	Mostra la vista de les fàbriques
Precondició	Usuari té sessió
Flux principal	1 – Mostrar menú principal. 2 – Llistar fàbriques.
Flux alternatiu	-
Post condició	S'han llistat totes les fàbriques de forma detallada amb el menú principal.

8.6.13 MILLORAR FÀBRICA

Millorar fàbrica	
Actor	Jugador
Versió	1.0
Descripció	Posa en cua la millora d'una fàbrica d'una ciutat.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que la fàbrica pertany a la civilització. 2 – Si la fàbrica pertany a la civilització. 2.1 – Comprovar que es pot millorar la fàbrica. 2.2 – Si la fàbrica es pot millorar. 2.2.1 – Actualitza vista amb el compte enrere.
Flux alternatiu	1 – Si la fàbrica no pertany a la civilització 1.1 – Sortir. 2.2 – Si la fàbrica no es pot millorar. 2.2.1 – Sortir.
Post condició	Si es pot millorar la fàbrica, s'ha generat el compte enrere per a que s'acabi de millorar la fàbrica.

8.6.14 LLISTAR FÀBRIQUES

Llistar fàbriques	
Actor	Jugador
Versió	1.0
Descripció	Fa un llistat de totes les fàbriques de la ciutat, amb descripció, nivell, i informació sobre la millora.
Precondició	Usuari té sessió
Flux principal	1 – Obtenir llistat de fàbriques amb nivell i si es poden millorar. 2 – Montar vista del llistat de fàbriques.
Flux alternatiu	-
Post condició	S'han llistat les fàbriques.

8.6.15 COMPROVAR QUE ES POT MILLORAR LA FÀBRICA

Comprovar que es pot millorar la fàbrica	
Actor	Jugador
Versió	1.0
Descripció	Comprova si es pot millorar la fàbrica, i en cas afirmatiu genera l'event de temps.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que la fàbrica pertany a la civilització. 2 – Comprovar que l'usuari disposa dels recursos per millorar-la. 3 – Comprovar que l'usuari pot afrontar el cost horari de la fàbrica. 4 – Restar els recursos necessàris per la millora a l'usuari. 5 – Obtenir el temps de la millora. 6 – Crear event de temps.
Flux alternatiu	1 – Si la fàbrica no pertany a la civilització. 1.1 – Retornar error. 2 – Si l'usuari no disposa dels recursos necessàris. 2.1 – Retornar error. 3 – Si l'usuari no pot afrontar el cost horari. 3.1 – Retornar error.
Post condició	Si es pot millorar la fàbrica, s'ha generat l'event de temps i s'han restat els recursos a l'usuari.

8.6.16 VEURE ATACS

Veure atacs	
Actor	Jugador
Versió	1.0
Descripció	Mostra la vista d'atacs.
Precondició	Usuari té sessió.
Flux principal	1 – Mostrar menú principal. 2 – Veure atac pendent.
Flux alternatiu	-
Post condició	S'han mostrar la possibilitat d'efectuar un atac o l'atac pendent.

8.6.17 VEURE ATAC PENDENT

Veure atac pendent	
Actor	Jugador
Versió	1.0
Descripció	Mostra si hi ha un atac pendent quan acabarà, sino el cost de fer un nou atac.
Precondició	Usuari té sessió.

Flux principal	1 – Obtenir atac pendent i cost de fer un atac.
	2 – Si hi ha un atac pendent.
	  2.1 – Mostrar atac pendent.
	3 – Altrament
	  3.1 – Mostrar nou atac.

Flux alternatiu	-
Post condició	Es mostra la vista d'atacs.

8.6.18 ATACAR

Atacar	
Actor	Jugador
Versió	1.0
Descripció	Posa en cua l'atac.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que es pot atacar. 2 – Si es pot atacar. 2.1 – Actualitza vista amb el compte enrere.
Flux alternatiu	2 – Si no es pot atacar. 2.1 – Sortir.
Post condició	Si es pot atacar, s'ha generat el compte enrere per a que s'acabi l'atac.

8.6.19 COMPROVAR QUE ES POT ATACAR

Comprovar que es pot atacar	
Actor	Jugador
Versió	1.0
Descripció	Comprova si es pot atacar, i en cas afirmatiu genera l'event de temps.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que l'usuari disposa dels recursos per atacar. 2 – Restar els recursos necessàris per la millora a l'usuari. 3 – Obtenir el temps d'atac. 4 – Crear event de temps.
Flux alternatiu	1 – Si l'usuari no disposa dels recursos necessàris. 1.1 – Retornar error.
Post condició	Si es pot atacar, s'ha generat l'event de temps i s'han restat els recursos a l'usuari.

8.6.20 VEURE CIVILITZACIONS

Veure civilitzacions	
Actor	Jugador
Versió	1.0
Descripció	Mostra la vista de civilitzacions
Precondició	Usuari té sessió.
Flux principal	1 – Mostrar menú principal. 2 – Llistar civilitzacions.
Flux alternatiu	-
Post condició	S'han llistat les civilitzacions de forma detallada amb el menú principal.

8.6.21 LLISTAR CIVILITZACIONS

Llistar civilitzacions	
Actor	Jugador
Versió	1.0
Descripció	Mostra la civilització principal i la civilització secundària, juntament amb la descripció i les possibles millores de cada una.
Precondició	Usuari té sessió.
Flux principal	1 – Obtenir informació de la civilització principal. 2 – Obtenir les millores de la civilització principal. 3 – Obtenir informació de la civilització secundària. 4 – Obtenir les millores de la civilització secundària. 5 – Montar vista de les civilitzacions.
Flux alternatiu	-
Post condició	S'han mostrat les civilitzacions.

8.6.22 MILLORAR CIVILITZACIÓ PRINCIPAL

Millorar civilització principal	
Actor	Jugador
Versió	1.0
Descripció	Posa en cua la millora de la civilització principal.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que es pot millorar la civilització principal. 2 – Si la civilització principal es pot millorar. 2.1 – Actualitza vista amb el compte enrere.
Flux alternatiu	2 – Si la civilització principal no es pot millorar. 2.1 – Sortir.
Post condició	Si es pot millorar la civilització principal, s'ha generat el compte enrere per a que s'acabi la millora.

8.6.23 COMPROVAR QUE ES POT MILLORAR LA CIVILITZACIÓ PRINCIPAL

Comprovar que es pot millorar la civilització principal	
Actor	Jugador
Versió	1.0
Descripció	Comprova si es pot millorar la civilització principal, i en cas afirmatiu genera l'event de temps.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que l'usuari disposa dels recursos per millorar-la. 2 – Restar els recursos necessàris per la millora a l'usuari. 3 – Obtenir el temps de la millora. 4 – Crear event de temps.
Flux alternatiu	1 – Si l'usuari no disposa dels recursos necessàris. 1.1 – Retornar error.
Post condició	Si es pot millorar la civilització principal, s'ha generat l'event de temps i s'han restat els recursos a l'usuari.

8.6.24 MILLORAR CIVILITZACIÓ SECUNDÀRIA

Millorar civilització secundària	
Actor	Jugador
Versió	1.0
Descripció	Posa en cua la millora de la civilització secundària.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que es pot millorar la civilització secundària. 2 – Si la civilització secundària es pot millorar. 2.1 – Actualitza vista amb el compte enrere.
Flux alternatiu	2 – Si la civilització secundària no es pot millorar. 2.1 – Sortir.
Post condició	Si es pot millorar la civilització secundària, s'ha generat el compte enrere per a que s'acabi la millora.

8.6.25 COMPROVAR QUE ES POT MILLORAR LA CIVILITZACIÓ SECUNDÀRIA

Comprovar que es pot millorar la civilització secundària	
Actor	Jugador
Versió	1.0
Descripció	Comprova si es pot millorar la civilització secundària, i en cas afirmatiu genera l'event de temps.
Precondició	Usuari té sessió.
Flux principal	1 – Comprovar que l'usuari disposa dels recursos per millorar-la. 2 – Restar els recursos necessàris per a la millora a l'usuari. 3 – Obtenir el temps de la millora. 4 – Crear event de temps.
Flux alternatiu	1 – Si l'usuari no disposa dels recursos necessàris. 1.1 – Retornar error.
Post condició	Si es pot millorar la civilització secundària, s'ha generat l'event de temps i s'han restat els recursos a l'usuari.

8.6.26 CREAR EVENT E TEMPS

Crear event de temps	
Actor	Jugador
Versió	1.0
Descripció	Genera un event de temps al servidor amb un callback del seu tipus d'event.
Precondició	Duració > 0. Tipus d'event conegut.
Flux principal	1 – Afegir a la base de dades el registre de l'event desde el temps actual fins temps actual + duració. 2 – Obtenir l'última id inserida a la taula d'events. 3 – Montar l'objecte json de l'event. 4 – Generar la funció que tindrà de callback a partir del tipus d'event. 5 – Crear el timeout que disparará el servidor.
Flux alternatiu	-
Post condició	S'ha afegit la informació de l'event a la base de dades i s'ha creat el timeout al servidor amb el callback corresponent.

8.6.27 GESTIONAR INCREMENT DE RECURSOS

Gestionar increment de recursos	
Actor	Events de servidor
Versió	1.0
Descripció	S'incrementen els recursos de les ciutats segons el seu guany/hora.
Precondició	-
Flux principal	1 – Per cada ciutat. 1.1 – Per cada tipus de recurs. 1.1.1 – Obtenir l'increment/hora del recurs a la ciutat. 1.1.2 – Sumar l'increment/hora a la quantitat del recurs que hi ha a la ciutat. 1.2 – Actualitzar data d'últim increment de recursos realitzar a la ciutat.
Flux alternatiu	-
Post condició	S'han incrementat correctament tots els recursos de totes les ciutats,

8.6.28 GESTIONAR INCREMENT DE RECURSOS ATRASSATS

Gestionar increment de recursos atrassats	
Actor	Events de servidor
Versió	1.0
Descripció	S'incrementen els recursos pendents (segons el nombre d'hores) de les ciutats segons el guany/hora.
Precondició	El servidor s'està engegant.
Flux principal	1 – Per cada ciutat. 1.1 – Obtenir la data de l'última actualització. 1.2 – Si ha passat més d'una hora. 1.2.1 – Per cada tipus de recurs. 1.2.1.1 – Obtenir l'increment/hora del recurs a la ciutat. 1.2.1.2 – Sumar l'increment/hora multiplicat pel nombre d'hores pendents a la quantitat del recurs que hi ha a la ciutat. 1.2.2 – Actualitzar data d'últim increment de recursos realitzar a la ciutat.
Flux alternatiu	-
Post condició	S'han incrementat correctament tots els recursos de totes les ciutats.

8.6.29 GESTIONA FI D'UN EVENT TEMPORAL

Gestionar fi d'un event temporal	
Actor	Events de servidor
Versió	1.0
Descripció	Es crida el callback d'un event que s'ha disparat
Precondició	Tipus d'event conegut.
Flux principal	1 – Obtenir tipus d'event. 2 – Disparar calback d'un event temporal.
Flux alternatiu	-
Post condició	S'ha cridat correctàment el callback de l'event.

8.6.30 GESTIONAR EVENTS ATRASSATS

Gestionar events atrassats	
Actor	Events de servidor
Versió	1.0
Descripció	Es crida el callback dels events que han acabat anteriorment i encara estan a la base de dades.
Precondició	Tipus d'event conegut. El servidor s'està engegant.
Flux principal	1 – Per cada event pendent a la base de dades. 1.1 – Mostar json de l'event a partir de la informació de l'event i del tipus. 1.2 – Generar la funció que tindrà de callback a partir del tipus d'event. 1.3 – Disparar calback d'un event temporal.
Flux alternatiu	-
Post condició	S'ha cridat correctàment el callback de l'event.

8.6.31 DISPARAR CALLBACK D'UN EVENT TEMPORAL

Disparar callback d'un event temporal	
Actor	Events de servidor
Versió	1.0
Descripció	Es redirecciona a un altre cas d'ús depenent del tipus de l'event.
Precondició	Tipus d'event conegut.
Flux principal	1 – Si l'event és d'un edifici. 1.1 – Pujar edifici de nivell. 2 – Si l'event és d'una fàbrica. 2.1 – Pujar fàbrica de nivell 3 – Si l'event és d'una civilització principal. 3.1 – Pujar civilització principal de nivell 4 – Si l'event és d'una civilització secundària. 4.1 – Pujar civilització secundària de nivell 5 – Si l'event és d'un atac. 5.1 – Calcular atac. 6 – Esborra l'event de la base de dades.
Flux alternatiu	-
Post condició	S'ha cridat correctàment el callback corresponent al tipus d'event i s'ha esborrat de la base de dades.

8.6.32 PUJAR EDIFICI DE NIVELL

Pujar edifici de nivell	
Actor	Events de servidor
Versió	1.0
Descripció	Puja de nivell l'edifici.
Precondició	L'efici té més nivells disponibles per pujar.
Flux principal	1 – Obtenir dades de l'event. 2 – Actualitzar el nivell de l'edifici a la base de dades.
Flux alternatiu	-
Post condició	S'ha pujat de nivell l'edifici.

8.6.33 PUJAR FÀBRICA DE NIVELL

Pujar fàbrica de nivell	
Actor	Events de servidor
Versió	1.0
Descripció	Puja de nivell la fàbrica.
Precondició	La fàbrica té més nivells disponibles per pujar.
Flux principal	1 – Obtenir dades de l'event. 2 – Actualitzar el nivell de la fàbrica a la base de dades.
Flux alternatiu	-
Post condició	S'ha pujat de nivell la fàbrica.

8.6.34 PUJAR CIVILITZACIÓ PRINCIPAL DE NIVELL

Pujar civilització principal de nivell	
Actor	Events de servidor
Versió	1.0
Descripció	Puja de nivell la civilització principal.
Precondició	La civilització a la que es vol millorar és derivada de l'actual.
Flux principal	1 – Obtenir dades de l'event. 2 – Actualitzar la civilització principal.
Flux alternatiu	-
Post condició	S'ha pujat de nivell la civilització principal.

8.6.35 PUJAR CIVILITZACIÓ SECUNDÀRIA DE NIVELL

Pujar civilització secundària de nivell	
Actor	Events de servidor
Versió	1.0
Descripció	Puja de nivell la civilització secundària.
Precondició	La civilització a la que es vol millorar és derivada de l'actual.
Flux principal	1 – Obtenir dades de l'event. 2 – Actualitzar la civilització secundària.
Flux alternatiu	-
Post condició	S'ha pujat de nivell la civilització secundària.

8.6.36 CALCULAR ATAC

Calcular atac	
Actor	Events de servidor
Versió	1.0
Descripció	Calcula el resultat d'un atac.
Precondició	-
Flux principal	1 – Obtenir dades de l'event. 2 – Cerca (de moment aleatòria) d'oponent. 3 – Calcular resultat de l'atac (guanyador o perdedor i quins recursos). 4 – Actualitzar els recursos de la ciutat1. 5 – Actualitzar els recursos de la ciutat2.

8.7 DIAGRAMA D'ENTITAT RELACIÓ

8.7.1 DIAGRAMA COMPLERT

El primer diagrama que es mostra correspon al disseny complert de la base de dades final, del qual hi ha mòduls i funcionalitats no implementades en el projecte. Com es pot observar, s'han agrupat vàries taules segons al mòdul general al que pertanyen. Així, a excepció de les taules que no pertanyen a cap mòdul en particular, tenim els següents grups:

- Ciutat
- Civilització
- Edifici
- Edifici Recurs
- Edifici Fàbrica
- Edifici Investigació
- Edifici Defensa
- Edifici Exèrcit
- Batalla
- Traducció
- Intercanvi
- Històric

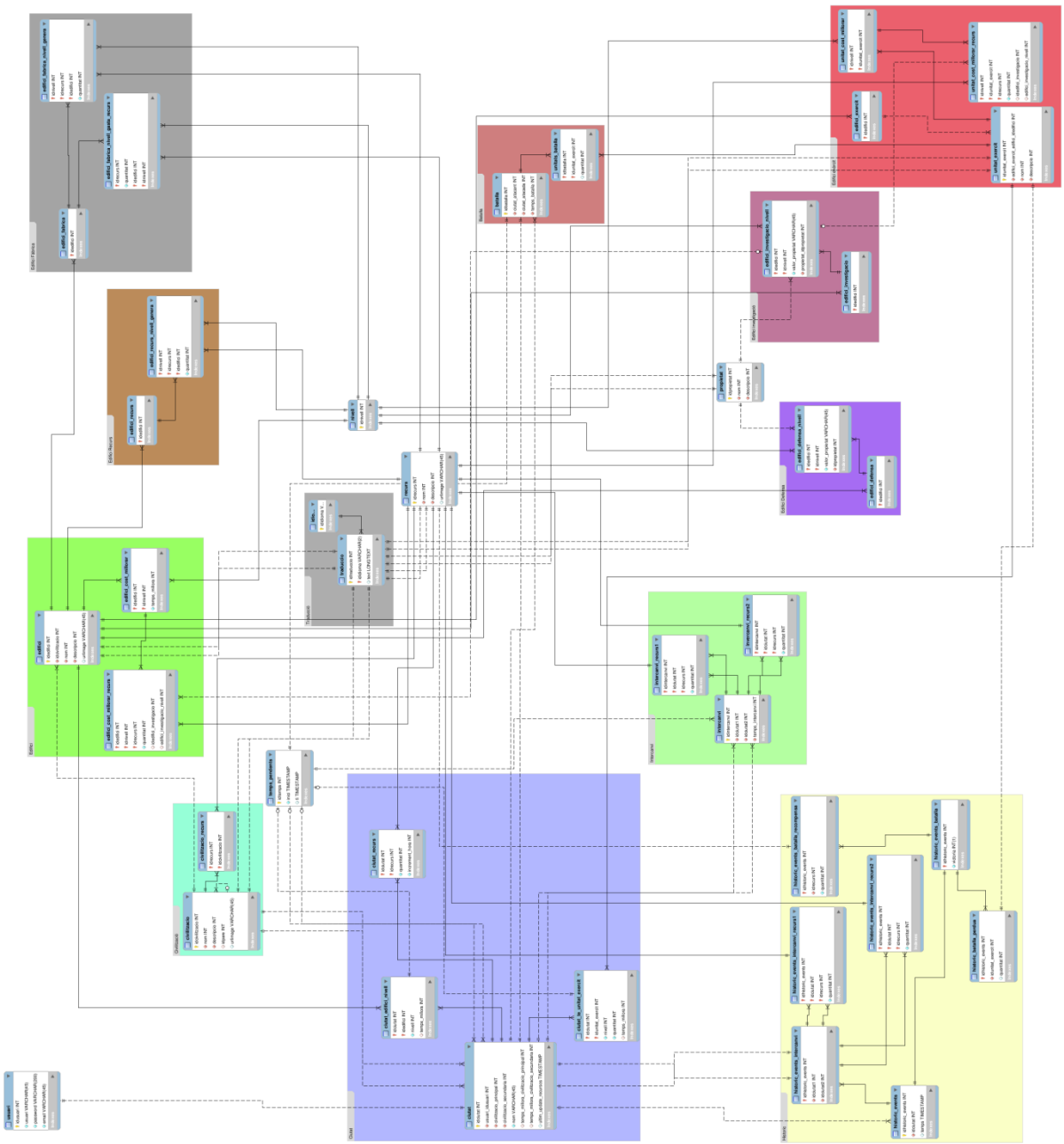

Figura 61: Diagrama Entitat-Relació complert

8.7.2 DIAGRAMA DE LA VERSIÓ ACTUAL

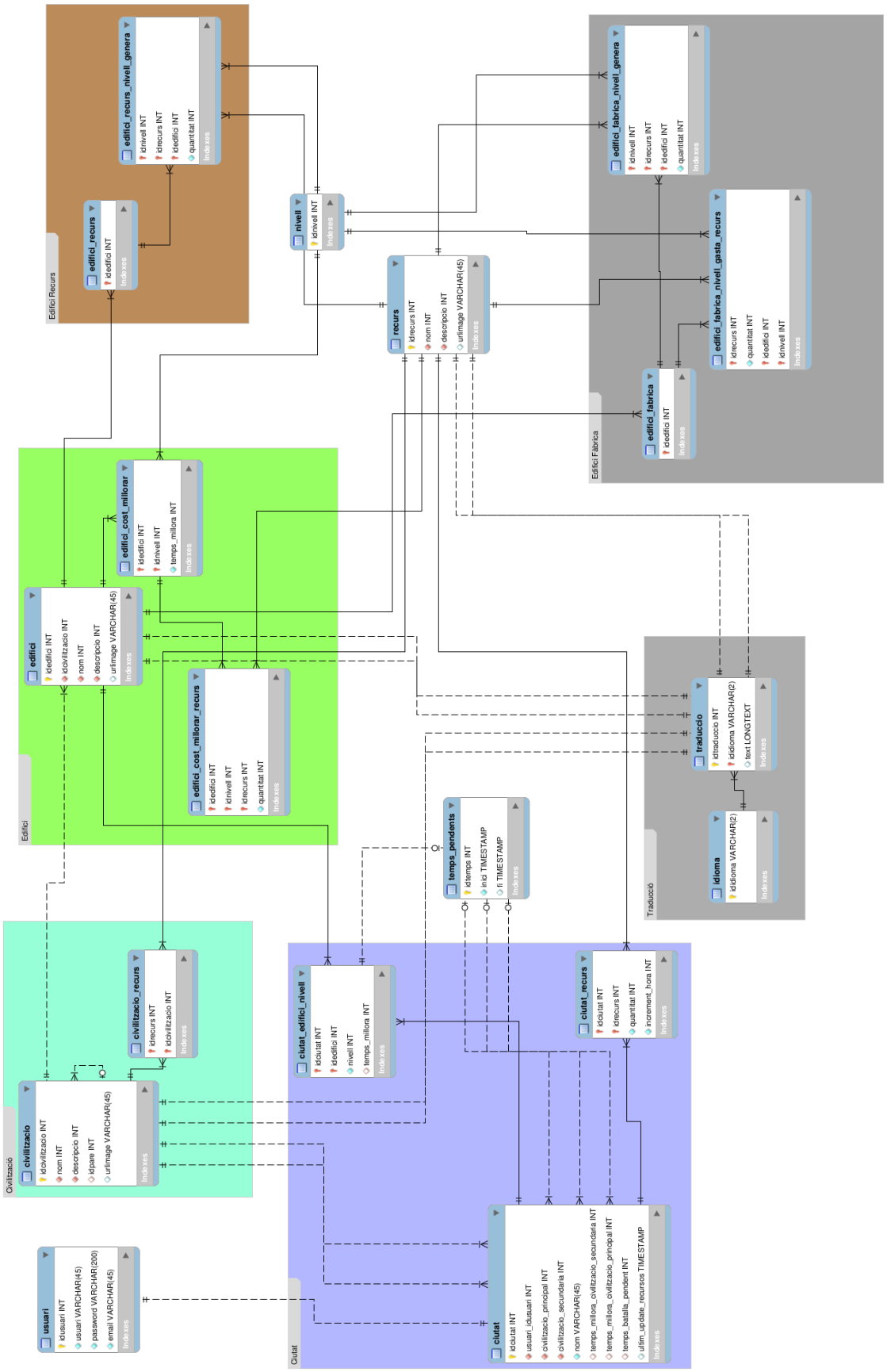

Figura 62: Diagrama Entitat-Relació del projecte actual

8.7.2.1 TAULA USUARI

Figura 63: Taula usuari

idusuari: clau primària. Autoincremental.

usuari: nick de l'usuari.

password: hash de la password de l'usuari.

email: correu electrònic de l'usuari. En un futur es pot implementar un sistema automàtic d'enviament de correus per afegir un mòdul de recuperació de contrassenya.

Segons el disseny de la base de dades, un usuari pot tenir diverses ciutats. Tot i que de moment un usuari només pot tenir una sola ciutat, el disseny ja està preparat per a futures ampliacions.

8.7.2.2 TAULA CIUTAT

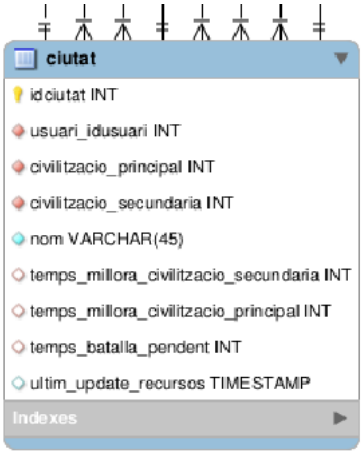

Figura 64: Taula ciutat

idciutat: clau primària. Autoincremental.

usuari_idusuari: id de l'usuari al que pertany la ciutat.

civilitzacio_principal: Id de la civilització principal de la ciutat.

civilitzacio_secundaria: Id de la civilització secundària de la ciutat.

nom: Nom de la ciutat.

temps_millora_civilitzacio_principal: id de l'event de temps si la civilització principal s'està millorant.

temps_millora_civilitzacio_secundaria: id de l'event de temps si la civilització secundària s'està millorant.

temps_batalla_pendent: per a la primera implementació s'ha simplificat al mínim el mòdul de batalla. Es guarda a la ciutat la id de l'event de temps de la batalla pendent i l'oponent, i el resultat es genera al servidor un cop acabat aquest event.

ultim_update_recursos: timestamp que guarda l'últim cop que s'han actualitzat els recursos de la ciutat. S'utilitza per poder generar les actualitzacions passades en engegar el servidor.

8.7.2.3 TAULA TEMPS_PENDENTS

Figura 65: Taula temps_pendents

idtemps: clau primària. Autoincremental.

inici: timestamp del moment en que comença l'event.

fi: timestamp del moment en que acaba l'event.

En aquesta taula es guarden tots els events que el servidor ha de'executar en algún moment. Si el servidor per qualsevol problema s'ha d'apagar, gràcies a aquesta taula tindrà constància de preparar de nou els timeouts i disparar els events que ja han acabat.

8.7.2.4 TAULA IDIOMA

Figura 66: Taula idioma

ididioma: text representatiu de l'idioma. En el sistema de proves s'emplena amb 'en' i 'es'.

8.7.2.5 TAULA TRADUCCIO

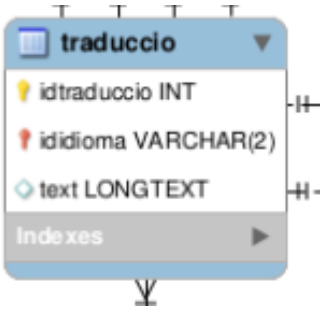

Figura 67: Taula traduccio

idtraduccio: part de la clau primària. Autoincremental.

ididioma: part de la clau primària. Fa referència a ididioma de la taula Idioma.

text: text de l'idtraduccio amb l'idioma corresponent.

Amb aquesta taula s'aconsegueix tenir els noms i descripcions de tots els textos de la base de dades en els diversos idiomes que es vulguin afegir al sistema.

8.7.2.6 TAULA CIVILITZACIO

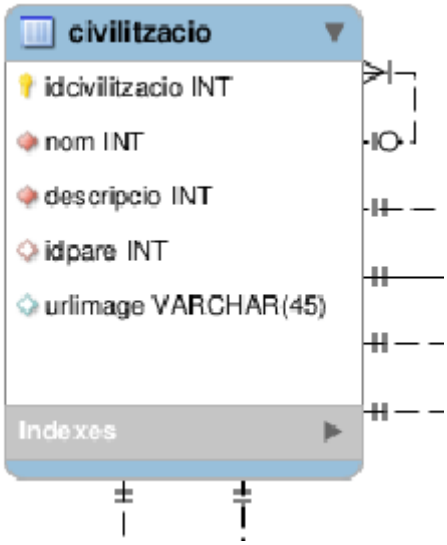

Figura 68: Taula civilitzacio

idcivilitzacio: clau primària. Autoincremental.

nom: nom de la civilització. Fa referència a un idtraduccio de la taula Traduccio.

descripcio: descripció de la civilització. Fa referència a un idtraduccio de la taula Traduccio.

idpare: id de la civilització predecessora. Fa referència a una idcivilitzacio de la taula Civilitzacio. L'arrel de l'arbre és l'única civilització que pot tenir el camp a NULL.

urlimage: nom de l'arxiu amb el fons de la civilització.

Com es pot veure, en aquesta taula es monta l'arbre jeràrquic de civilitzacions.

8.7.2.7 TAULA RECURS

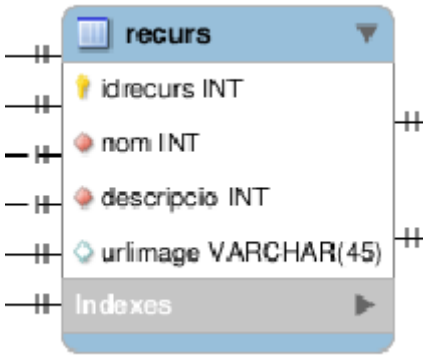

Figura 69: Taula recurs

idrecurs: clau primària. Autoincremental.

nom: nom del recurs. Fa referència a un idtraduccio de la taula Traduccio.

descripcio: descripció del recurs. Fa referència a un idtraduccio de la taula Traduccio.

urlimage: nom de l'arxiu amb la icona del recurs.

8.7.2.8 TAULA CIVILITZACIO_RECURS

Figura 70: Taula civilitzacio_recurs

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

idcivilitzacio: part de la clau primària. Fa referència a idcivilitzacio de la taula Civilitzacio.

Tot i que la idea és que hi hagi molts tipus de recursos en el joc, un jugador utilitzarà i incrementarà principalment els que estan directament relacionats amb la seva civilització. Aquesta taula s'encarrega de lligar quins són els recursos principals de cada civilització. D'aquesta manera, quan es mostren els recursos actuals de forma resumida, només aparéixen els que pertànyen a la unió de les dues civilitzacions de la ciutat.

8.7.2.9 TAULA CIUTAT_RECURS

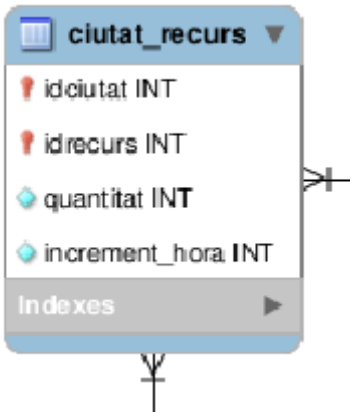

Figura 71: Taula ciutat_recurs

idciutat: part de la clau primària. Fa referència a idciutat de la taula Ciutat.

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

quantitat: El nombre d'unitats del que la ciutat disposa del recurs.

increment_hora: El nombre d'unitats del recurs que guanya la ciutat cada hora.

L'increment_hora s'utilitza es la funció horària per incrementar els recursos de cada ciutat. Tot i que el càlcul es podria fer directament accedint a tots els edificis de recursos i el seu nivell en la ciutat, i restant la producció per hora que poden produïr les fàbriques també segons el seu nivell a la ciutat, la complexitat del càlcul és massa elevada per fer-ho cada hora amb totes les ciutats del sistema. Per això és millor fer el càlcul un cop quan canvia el nivell d'algun edifici o fàbrica i guardar l'accés directe a ciutat_recurs.

8.7.2.10 TAULA NIVELL

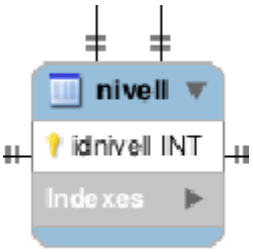

Figura 72: Taula nivell

idnivell: clau primària. El propi número del nivell.

El propòsit d'aquesta taula és el de marcar quins nivell hi ha disponibles al sistema, així com tenir sempre indexat el nivell, factor que agilitzarà moltes cerques pesades en el càlcul de millores d'edificis.

Així doncs, un nivell es el marcador numèric dels estats en els que l'edifici canvia de propietats.

8.7.2.11 TAULA EDIFICI

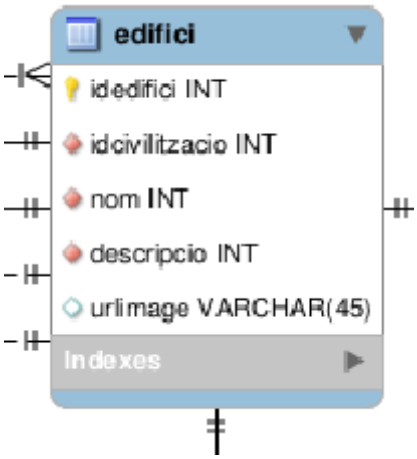

Figura 73: Taula edifici

idedifici: clau primària. Autoincremental.

idcivilitzacio: civilització a la que pertany l'edifici. Fa referència a idcivilitzacio de la taula Civilitzacio.

nom: nom de l'edifici. Fa referència a un idtraduccio de la taula Traduccio.

descripcio: descripció de l'edifici. Fa referència a un idtraduccio de la taula Traduccio.

urlimage: nom de l'arxiu amb la imatge de l'edifici.

En l'àmbit de la base de dades, quan es fa referència a un edifici, es considera tota construcció que augmenta de nivell. A la resta del projecte, quan es parla d'edifici, es fa referència a un edifici_recurs de la base de dades.

Cal tenir en compte que un jugador pot veure i millorar tots els edificis de la unió de les dues civilitzacions i tots els pares d'aquestes. És a dir, si un edifici pertany a la civilització arrel, tots els jugadors més avançats que tindran altres civilitzacions el posseixen igualment.

8.7.2.12 TAULA CIUTAT_EDIFICI_NIVELL

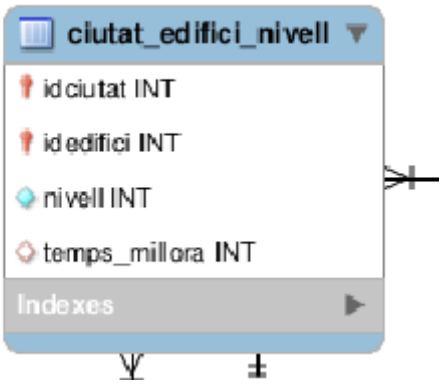

Figura 74: Taula ciutat_edifici_nivell

idciutat: part de la clau primària. Fa referència a idciutat de la taula Ciutat.

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici.

nivell: nivell que té l'edifici a la ciutat.

temps_millora: id de l'event de temps si l'edifici s'està millorant.

8.7.2.13 TAULA EDIFICI_COST_MILLORAR

Figura 75: Taula edifici_cost_millorar

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici.

idnivell: part de la clau primària. Fa referència a idnivell de la taula Nivell.

temps_millora: segons que costa realitzar la millora cap al següent nivell.

Cal saber que qualsevol edifici pot requerir un o més tipus de recursos per fer la millora. És per això que en aquesta taula només es guarda el temps de fer la millora.

8.7.2.14 TAULA EDIFICI_COST_MILLORAR_RECURS

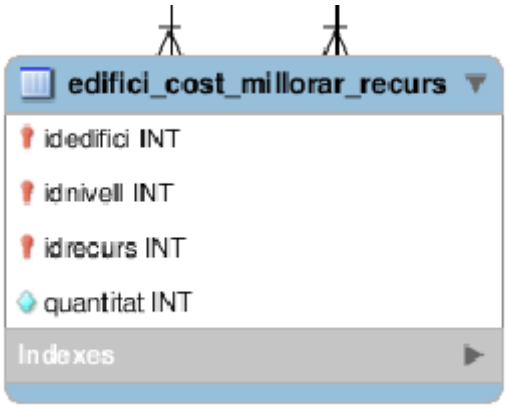

Figura 76: Taula edifici_cost_millorar_recurs

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici.

idnivell: part de la clau primària. Fa referència a idnivell de la taula Nivell.

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

quantitat: nombre d'unitats del recurs que es requereix per pujar de nivell.

Com s'ha esmentat anteriorment, en el disseny es permet que en la millora d'un edifici siguin necessàris nombrosos recursos. En aquesta taula es tracta quins recursos i amb quina quantitat són necessàris per cada nivell de l'edifici.

8.7.2.15 TAULA EDIFICI_RECURS

Figura 77: Taula edifici_recurs

idedifici: clau primària. Fa referència a idedifici de la taula Edifici.

Cal tenir en compte que, tot i que els diversos tipus d'edificis tenen una part comuna, arriba un moment en el que cada tipus d'edifici es comporta de forma diferent, cosa que provoca canvis en les dades que s'han de guardar d'ells.

En aquesta taula s'enregistren les ids dels edificis que són del tipus de generar recursos. Gràcies a ella es poden fer cerques més ràpides per tipus d'edifici, ja que cada vista només tracta sobre un tipus diferent en cada moment.

8.7.2.16 TAULA EDIFICI_RECURS_NIVELL_GENERA

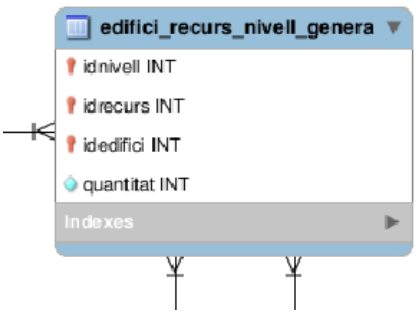

Figura 78: Taula edifici_recurs_nivell_genera

idnivell: part de la clau primària. Fa referència a idnivell de la taula Nivell.

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici_recurs, i a la vegada a idedifici de la taula Edifici.

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

quantitat: nombre d'unitats que l'edifici genera del recurs cada hora.

El disseny de la base de dades permet que un sol edifici_recurs pugui generar diferents tipus de recursos amb diverses quantitats.

8.7.2.17 TAULA EDIFICI_FABRICA

Figura 79: Taula edifici_fabrica

idedifici: clau primària. Fa referència a idedifici de la taula Edifici.

En aquesta taula s'enregistren les ids dels edificis que són del tipus fàbrica (generar recursos a canvi de gastar-ne d'altres). Gràcies a ella es poden fer cerques més ràpides per tipus d'edifici, ja que cada vista només tracta sobre un tipus diferent en cada moment.

8.7.2.18 TAULA EDIFICI_FABRICA_NIVELL_GENERA

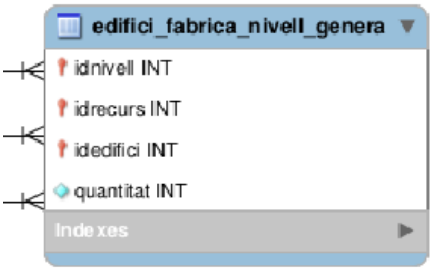

Figura 80: Taula edifici_fabrica_nivell_genera

idnivell: part de la clau primària. Fa referència a idnivell de la taula Nivell.

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici_recurs, i a la vegada a idedifici de la taula Edifici.

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

quantitat: nombre d'unitats que l'edifici genera del recurs cada hora.

D'igual manera que edifici_recurs_nivell_genera, el disseny de la base de dades permet que un sol edifici_fabrica pugui generar diferents tipus de recursos amb diverses quantitats.

8.7.2.19 TAULA EDIFICI_FABRICA_NIVELL_GASTA_RECURS

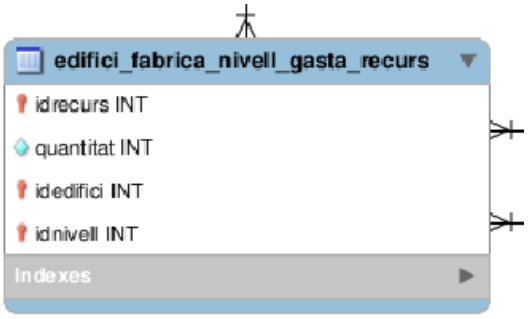

Figura 81: Taula edifici_fabrica_nivell_gasta_recurs

idnivell: part de la clau primària. Fa referència a idnivell de la taula Nivell.

idedifici: part de la clau primària. Fa referència a idedifici de la taula Edifici_fabrica, i a la vegada a idedifici de la taula Edifici.

idrecurs: part de la clau primària. Fa referència a idrecurs de la taula Recurs.

quantitat: nombre d'unitats que la fàbrica gasta del recurs cada hora.

El disseny de la base de dades permet que una sola fabrica pugui gastar nombrosos tipus de recursos amb diverses quantitats.

8.8 DISSENY DE MÒDULS

Javascript, siguent un llenguatge orientat a events, no disposa d'un sistema estricte de classes. Per això, en aquesta secció es tracten dels diferents mòduls dels que disposa l'aplicació, separant si pertànyen al client o al servidor.

8.8.1 DISSENY DELS MÒDULS DEL CLIENT

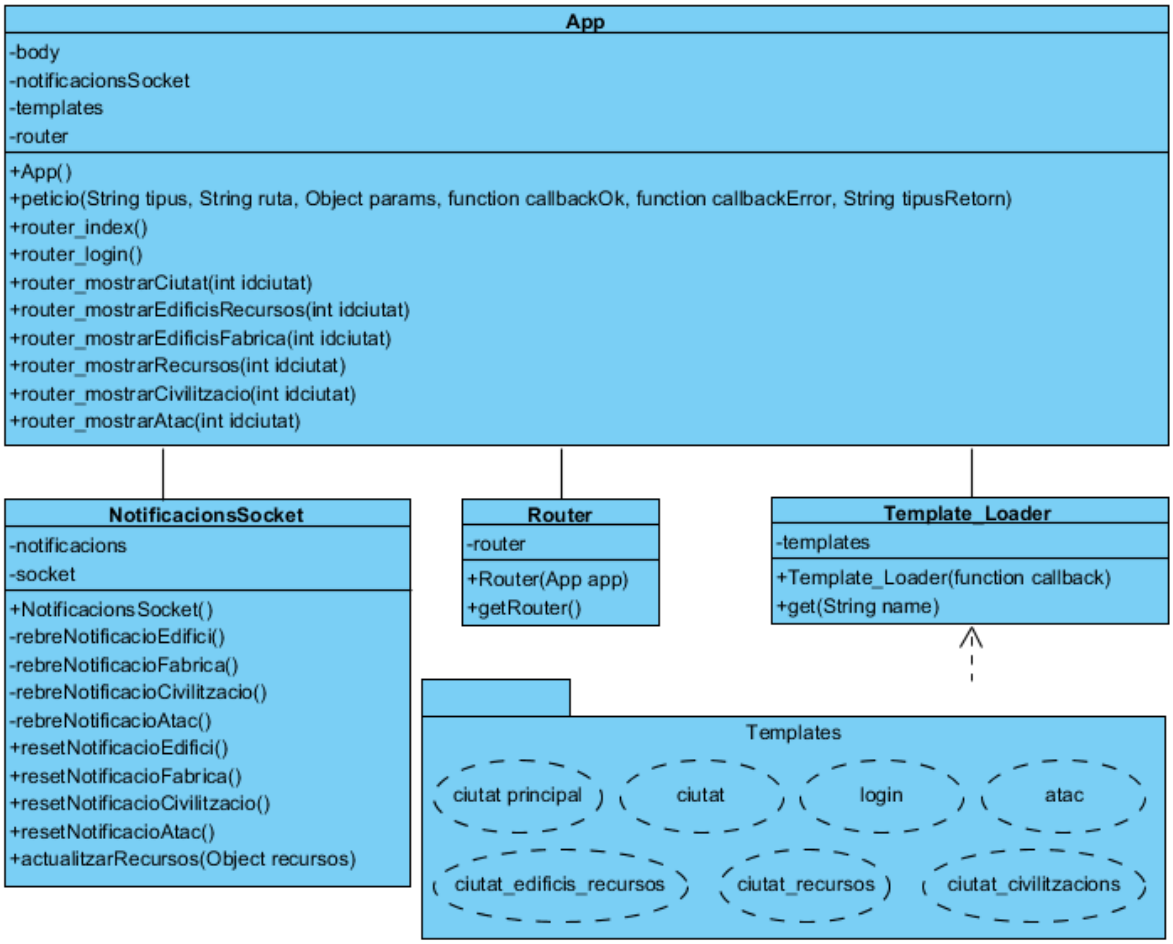

Figura 82: Diagrama de mòduls del client

8.8.1.1 APP

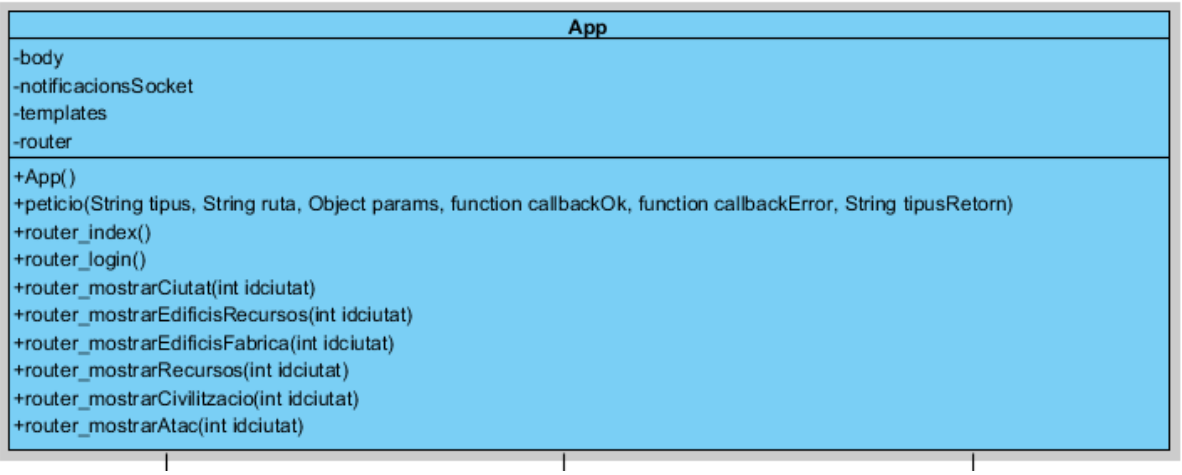

Figura 83: Mòdul App

Atributs:

body: variable jQuery on es carreguen les vistes.

notificacionsSockets: inicialitza el mòdul NotificacionsSockets.

templates: inicialitza el mòdul Template_Loader.

router: incialitza el mòdul Router.

Funcions públiques:

App: constructor del mòdul.

peticio: funció general per duur a terme peticions ajax.

router_index: comprova la sessió de l'usuari per redireccionar a router_mostrarCiutat o router_login.

router_login: prepara la vista de login.

router_mostrarCiutat: prepara la vista de la ciutat.

router_mostrarEdificisRecursos: prepara la vista dels edificis de recursos.

router_mostrarEdificisFabrica: prepara la vista dels edificis fàbrica.

router_mostrarRecursos: prepara la vista de recursos.

router_mostraCivilitzacio: prepara la vista de les civilitzacions de la ciutat.

router_mostrarAtac: prepara la vista d'atac.

Mòdul principal del client. S'encarrega d'inicialitzar tots els altres mòduls i de preparar les diverses vistes del sistema.

8.8.1.2 NOTIFICACIONSSOCKET

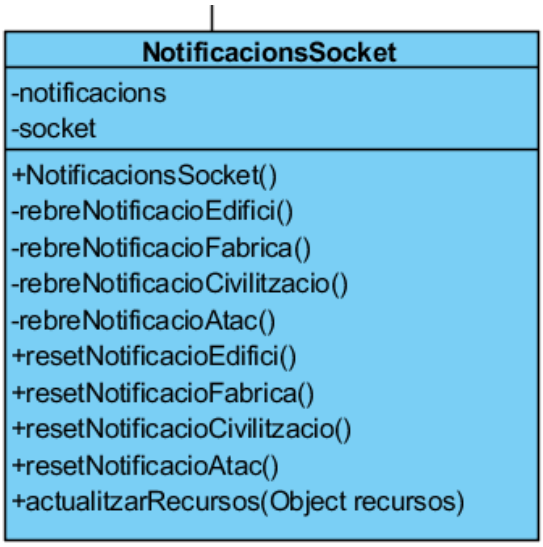

Figura 84: Mòdul NotificacionsSocket

Atributs:

notificacions: taula d'enters. Cada posició indica quantes alertes hi ha de cada tipus (0 edifici, 1 fàbrica, 2 civilització i 3 atac).

socket: socket de la connexió.

Funcions privades:

rebreNotificacioEdifici: augmenta en 1 la posició 0 de la taula notificacions i l'actualitza a la interfície.

rebreNotificacioFabrica: augmenta en 1 la posició 1 de la taula notificacions i l'actualitza a la interfície.

rebreNotificacioCivilitzacio: augmenta en 1 la posició 2 de la taula notificacions i l'actualitza a la interfície.

rebreNotificacioAtac: augmenta en 1 la posició 3 de la taula notificacions i l'actualitza a la interfície.

Funcions públiques:

NotificacionsSocket: constructor del mòdul.

resetNotificacioEdifici: posa a 0 la posició 0 de la taula notificacions i treu la notificació de la interfície.

resetNotificacioFabrica: posa a 1 la posició 0 de la taula notificacions i treu la notificació de la interfície.

resetNotificacioCivilitzacio: posa a 2 la posició 0 de la taula notificacions i treu la notificació de la interfície.

resetNotificacioAtac: posa a 3 la posició 0 de la taula notificacions i treu la notificació de la interfície.

actualitzarRecursos: actualitza els recursos a la interfície.

Mòduls encarregat en les funcions relacionades amb les actualitzacions per socket. En el sistema només es rep informació en temps real, en cap cas se n'envia.

8.8.1.3 ROUTER

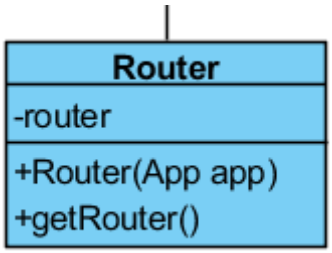

Figura 85: Mòdul Router

Atributs:

router: extensió del mòdul Router de BackBone. Conté totes les possible urls del sistema i crida la vista corresponent.

Funcions públiques:

Router: constructor del mòdul.

getRouter: retorna l'atribut router.

Mòduls encarregat d'encapsular el router de Backbone.

8.8.1.4 TEMPLATE_LOADER

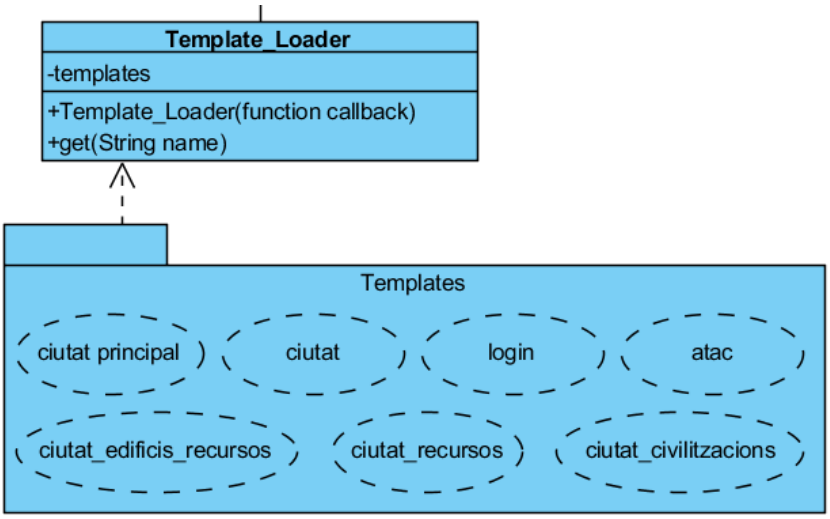

Figura 86: Mòdul Template_Loader i la llista de templates

Atributs:

templates: conjunt de templates carregats.

Funcions públiques:

Template_Loader: constructor del mòdul. Carrega tots els templates a dintre la variable templates.

get: Retorna un template a partir del nom.

Mòdul encarregat de carregar i servir els diversos templates d'Underscore utilitzats a les vistes.

8.8.2 DISSENY DELS MÒDULS DEL SERVIDOR

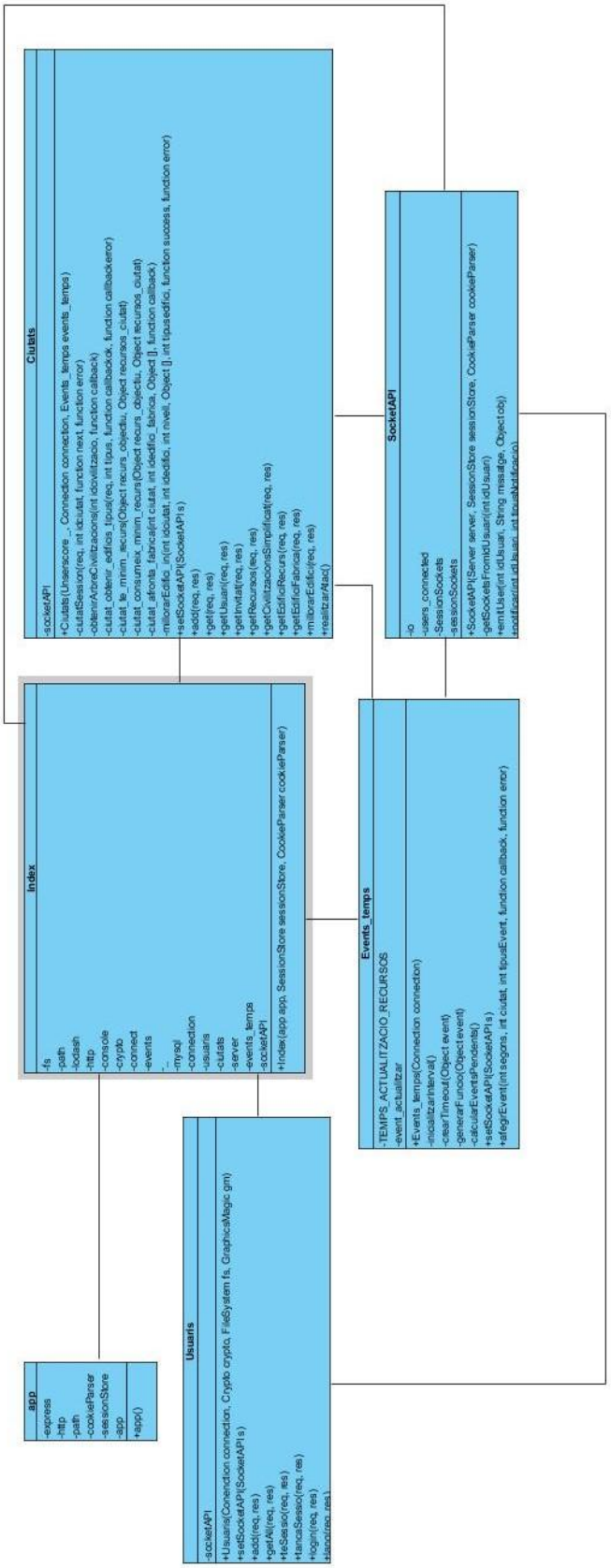

Figura 87: Diagrama de mòduls del servidor

8.8.2.1 APP

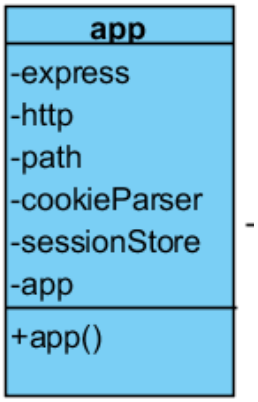

Figura 88: Mòdul App

Atributs:

express: mòdul express.

http: mòdul http.

path: mòdul path.

cookieParser: cookieParser de l'express generat amb la clau del servidor.

sessionStore: memoryStore de la sessió d'express.

app: inicialització de l'express.

Funcions públiques:

App: constructor del mòdul.

Aquest mòdul s'encarrega d'inicialitzar les llibreries i configuràcions bàsiques que tindrà el servidor, com express i la gestió de sessions.

8.8.2.2 INDEX

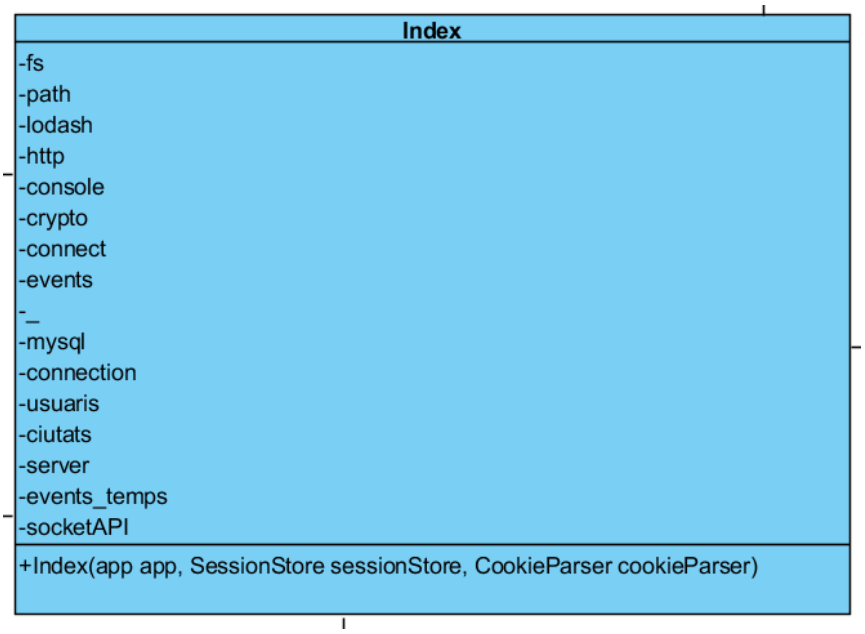

Figura 89: Mòdul Index

Atributs:

fs: mòdul fs.

path: mòdul path.

lodash: mòdul lodash.

http: mòdul http.

console: mòdul console.

crypto: mòdul crypto.

connect: mòdul connect,

events: mòdul events.

_: mòdul underscore.

mysql: mòdul mysql2.

conenction: connexió a mysql.

usuaris: mòdul usuaris.

ciutats: mòdul ciutaats.

server: servidor http.

evemts_temps: mòdul events_temps.

socketAPI: mòdul socketAPI.

Funcions públiques:

Index: constructor del mòdul.

S'encarrega d'inicialitzar el servidor, preparar l'APIRest, conenctar-se a la base de dades, inicialitzar els sockets, així com inicialitzar el mòdul de temps_pendents.

8.8.2.3 SOCKETAPI

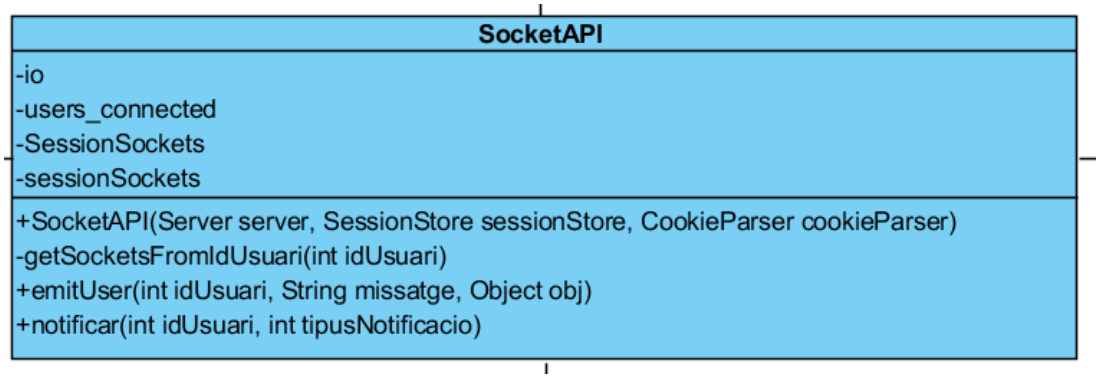

Figura 90: Mòdul SocketAPI

Atributs:

io: mòdul socket.io.

users_connected: Taula que relaciona els usuaris connectats al sistema amb els sockets oberts.

SessionSockets: mòdul session.socket.io.

sessionSockets: inicialització de session.socket.

Funcions privades:

getSocketsFromIdUsuari: retorna la taula de sockets connectats amb la sessió de l'usuari.

Funcions públiques:

SocketAPI: constructor del mòdul.

emitUser: delega un emit a tots els sockets que hi ha online d'un usuari.

notificar: decideix quin emit s'ha de disparar quan es millora alguna cosa a partir del tipus de millora.

Mòdul que s'encarrega de gestionar les connexions simultànies del mateix usuari des de diversos dispositius i la generació de missatges en temps real del sistema.

8.8.2.4 EVENTS_TEMPS

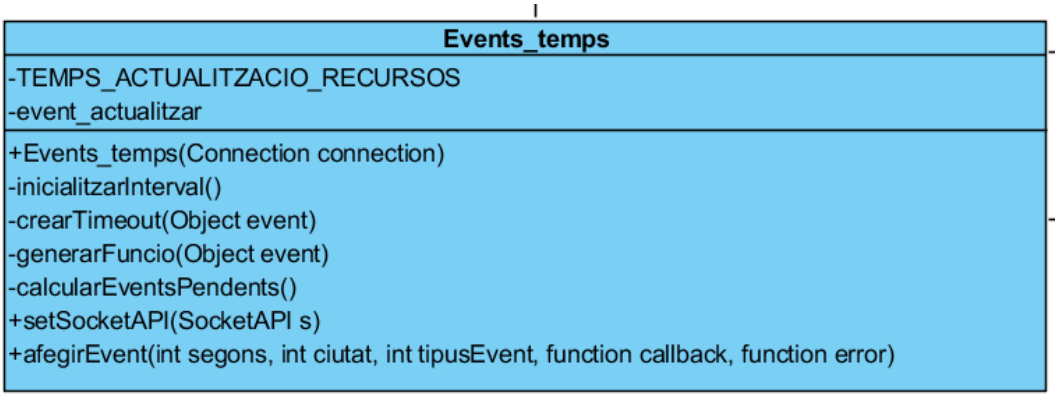

Figura 91: Mòdul Events_temps

Atributs:

TEMPS_ACTUALITZACIO_RECURSOS: temps de durada per tornar a disparar l'interval. Es pot canviar per motius de testeig.

event_actualitzar: id de l'interval d'actualitzar els recursos de totes les ciutats del sistema.

Funcions privades:

inicialitzarInterval: inicialitza l'interval d'actualització horària de recursos.

crearTimeout: donat un event genera el seu timeout al sistema.

generarFuncio: genera la funció de callback de l'event a partir del tipus.

calcularEventsPendents: resol tots els temps d'events pendents. Els que ja han passat els activa i dels que encara s'han de completar en genera el timeout.

Funcions públiques:

Events_temps: constructor del mòdul.

setSocketAPI: assigna el mòdul SocketAPI.

afegirEvent: serveix per afegir un nou event al sistema.

Aquest mòdul s'encarrega de gestionar tots els events de temps, ja sigui l'interval horari d'augmentar recursos com els timeouts de les millores i atacs de les ciutats.

8.8.2.5 USUARIS

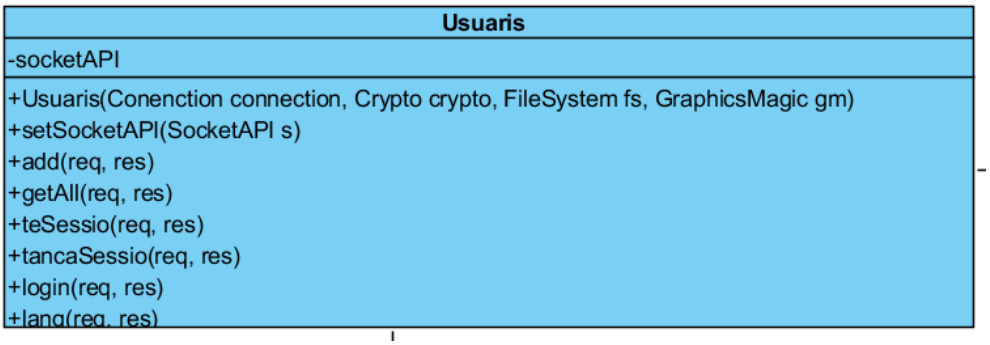

Figura 92: Mòdul Usuaris

Atributs:

socketAPI: assigna el mòdul SocketAPI.

Funcions públiques:

Usuaris: constructor del mòdul.

setSocketAPI: assigna el mòdul SocketAPI.

add: registre de l'usuari. Genera el hash de la contrassenya.

getAll: retorna un llistat de tots els usuaris del sistema.

teSessio: comprova que l'usuati té sessió.

tancaSessio: esborra la sessió de l'usuari.

login: logueja l'usuari al sistema i li genera la sessió.

lang: canvia l'idioma a la sessió de l'usuari, per escollir en quin idioma cal recuperar els textos a la base de dades.

Mòduls d'usuaris. S'encarrega de la gestió dels usuari del sistema a través de l'API Rest.

8.8.2.6 CIUTATS

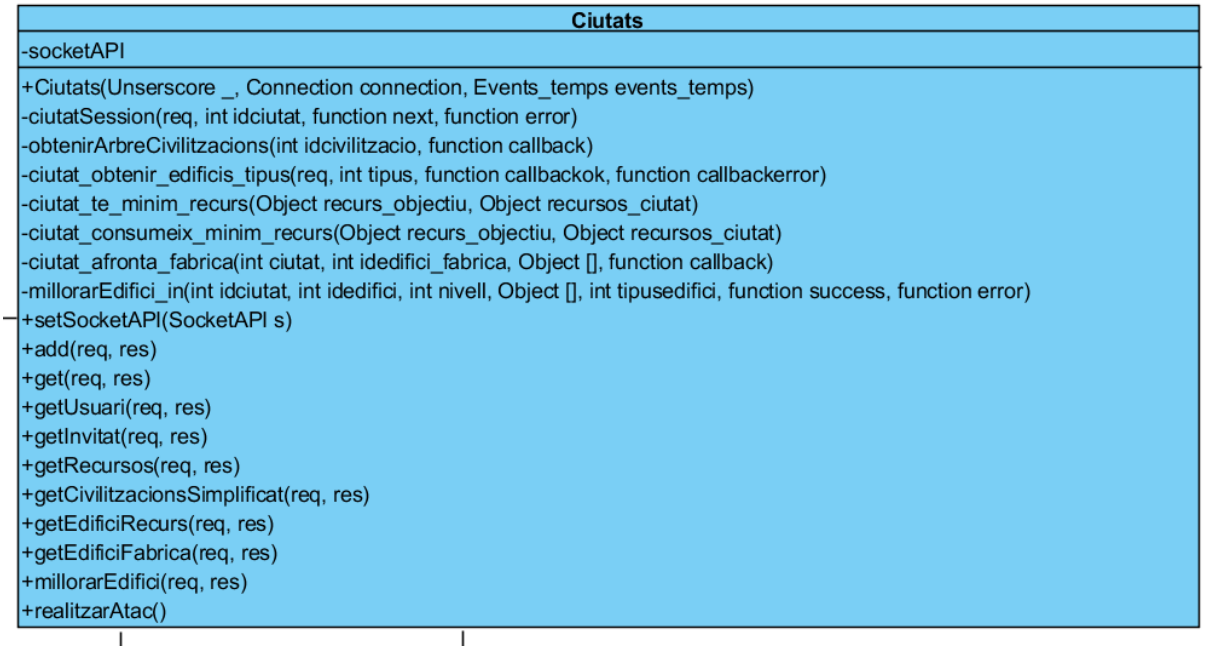

Figura 93: Mòdul Ciutats

Atributs:

socketAPI: assigna el mòdul SocketAPI.

Funcions privades:

ciutatSession: comprova si la ciutat pertany a l'usuari.

obtenirArbreCivilitzacions: es crida recursivament per generar un la llista de pares de la civilització objectiu.

ciutat_obtenir_edificis_tipus: s'obté tota la informació necessària per mostrar tots els edificis del tipus demanat, sigui del tipus que sigui.

ciutat_te_minim_recurs: retorna si dintre dels recursos de la ciutat es compleix el recurs objectiu.

ciutat_conzumeix_minim_recurs: retorna si dintre dels recursos de la ciutat es compleix (en guany/hora) el recurs objectiu.

ciutat_afronta_fabrica: comprova si una ciutat pot soportar el cost horari d'una fàbrica.

millorarEdifici_in: insereix un event de temps per pujar de nivell un edifici d'una ciutat.

Funcions públiques:

Ciutats: constructor del mòdul.

setSocketAPI: assigna el mòdul SocketAPI.

add: creació d'una ciutat nova.

get: comprova si l'usuari és propietari de la ciutat o no. Si n'és el propietari retorna la informació de la ciutat i dels seus recursos. En cas que no sigui propietari només retorna la informació referent a la ciutat.

getUsuari: es retorna la tota la informació principal de la ciutat.

getInvitat: es retorna una part de la informació principal de la ciutat.

getRecursos: es retorna la informació dels recursos de la ciutat.

getCivilitzacionsSimplificat: retorna les civilitzacions a les que pertany la ciutat i les possibles millores.

getEdificiRecurs: es retornen els edificis de recursos disponibles.

getEdificiFabrica: es retornen els edificis de fàbrica disponibles.

millorarEdifici: comprova que l'edifici es pugui millorar i prepara l'event de millora.

realitzarArac: comprova que pot fer un atac i prepara l'event d'atac.

Aquest mòdul gestiona totes les funcions relacionades amb la gestió de la ciutat. Les funcions públiques corresponen a l'API Rest de la ciutat.

8.9 MISSATGES DE SOCKETS

Els missatges de sockets especificats en el sistema són els següents:

- connection: Missatge predeterminat de connexió. Quan es rep un missatge connection es comprova si l'usuari té sessió i el vincula a l'estructura users_connected (veure 8.8.2.3).
- disconnect: Missatge predetermina de desconnexió. Quan es rep aquest missatge s'esborra el socket de l'estructura users_connected.
- notificacioEdifici: S'envia a un usuari quan un edifici s'ha acabat de millorar.
- notificacioFabrica: S'envia a un usuari quan una fàbrica s'ha acabat de millorar.
- notificacioCivilitzacio: S'envia a un usuari quan una civilització s'ha acabat de millorar.
- notificacioAtac: S'envia a un usuari quan s'ha realitzat un atac.

- actualitzarRecursos: S'envia a l'usuari quan s'ha provocat una acció sobre els recursos (sigui perdre recursos per fer una millora o guanyar-ne cada hora).

9 IMPLEMENTACIÓ I PROVES

En aquesta secció s'explica com s'han implementat els elements més representatius del projecte, per poder arribar al resultat final.

9.1 CLIENT

9.1.1 ESTRUCTURA DE DIRECTORIS

Observant la figura 94 es pot veure l'estructura de directoris del projecte del client.

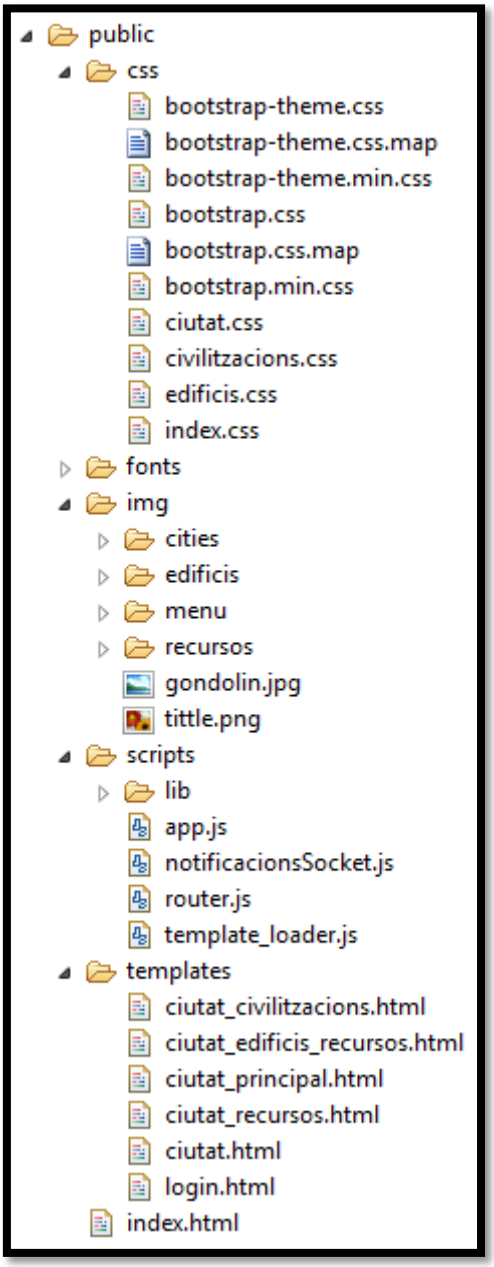

Figura 94: Estructura de direcoris del client

css: carpeta que conté diversos fitxers d'estils del projecte.
fonts: carpeta que conté els fitxers de les fonts personalitzades de la plana web.
img: carpeta que conté totes les imatges de la plana web.

- **cities**: carpeta que conté la iamtge de fons de cada civilització.
- edificis: carpeta que conté la imatge representativa dels edificis.
- menu: carpeta que conté les icones del menú general.
- **recursos**: carpeta que conté les icones de cada tipus de recurs.

scripts: carpeta que conté els diversos fitxers javascript del projecte.

- **lib**: carpeta que conté els fitxers javascript provinents de llibreries i plugins externs.

templates: carpeta que conté tots els templates de les vistes del projecte.
index.html: codi inicial de la plana web.

9.1.2 RESPONSIVE

Tota la pàgina web és responsive, és a dir, que la informació que es mostra i com es mostra depèn del dispositiu i de les propietats de la pantalla.

Primer de tot, per poder reconéixer de forma correcta els paràmetres de la pantalla en un dispositiu mòbil, s'ha d'afegir un tag meta en el fitxer index.html:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Una de les característiques del disseny responsive és que hi ha certes regles (media queries) que només s'apliquen quan es dóna una coincidència amb les propietats establertes. En aquest cas per separar el disseny s'han establert aquestes condicions:

```
/*************************************************/
/*************** Media Queries *****************/
/*************************************************/
/* MQ1: Desde mòbil horitzontal fins tablet vertical */
@media (max-width: 767px){ }
/* MQ2: Mòbils horitzontals i menys */
 @media (max-width: 480px) { }
 /* MQ3: Mòbils ultraminis vertical*/
@media (max-height: 300px) and (orientation : portrait) { }
 /* MQ4: Desde tablet vertial fins a escriptori */
@media (min-width: 768px) and (max-width: 979px) { }
/* MQ5: Escriptori gran */
@media (min-width: 1200px) { }
```

Utilitzant aquestes media queries s'han pogut fer que el menú general de l'aplicació es vagi reescalant fins arribar a un punt on al estar en vertial canvia de lloc:

```
.ciutat_bar1{  width: 7%; height: 100%; position: absolute;
background-color: rgb(214, 142, 62);
}
.ciutat_bar2{ width: 7%; height: 100%; right: 0px; position: absolute;
background-color: rgb(214, 142, 62);
}
.ciutat_background{ position: absolute; left: 7%; right: 7%; height: 100%;
background-size: cover; background-position: 50%;
}
/* MQ1: Desde mòbil horitzontal fins tablet vertical */ @media (max-width: 767px){ .ciutat_bar1{
width: 15%; }
.ciutat_bar2{ width: 15%; }
 .ciutat_background{ left: 15%; right: 15%; }
}
/* MQ2: Mòbils horitzontals i menys */ @media (max-width: 480px) {
.ciutat_bar1{ width: 100%; height: 15%; top: 0px; }
.ciutat_bar2{ width: 100%; height: 15%; bottom: 0px; }
.ciutat_background{ left: 0px; right: 0px; top: 15%; bottom: 15%; height: auto; }
}
/* MQ3: Mòbils ultraminis vertical*/ @media (max-height: 300px) and (orientation : portrait) {
.ciutat_bar1{ width: 100%; height: 25%; top: 0px; }
.ciutat_bar2{ width: 100%; height: 25%; bottom: 0px; }
.ciutat_background{ left: 0px; right: 0px; top: 25%; bottom: 25%; height: auto; }
}
/* MQ4: Desde tablet vertial fins a escriptori */ @media (min-width: 768px) and (max-width:
979px) { .ciutat_bar1{ width: 10%; } .ciutat_bar2{ width: 10%; } .ciutat_background{ left: 10%; right:
10%; } }
```

Els resultat d'aquest fragment juntament amb altres subcomponents de les barres és el següent:

Figura 95: Barres de menú responsive – pantalla gran

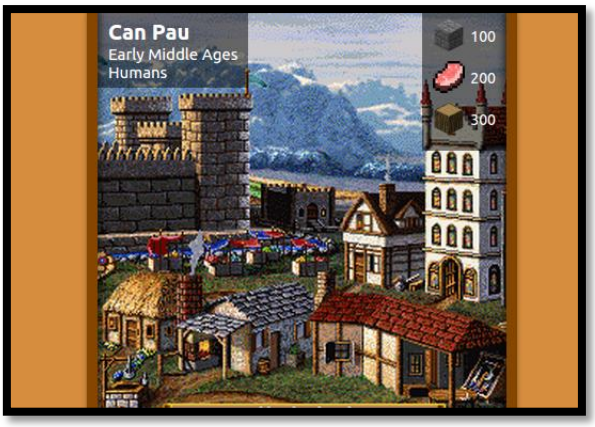

Figura 96: Barres de menú responsive – pantalla mitjana

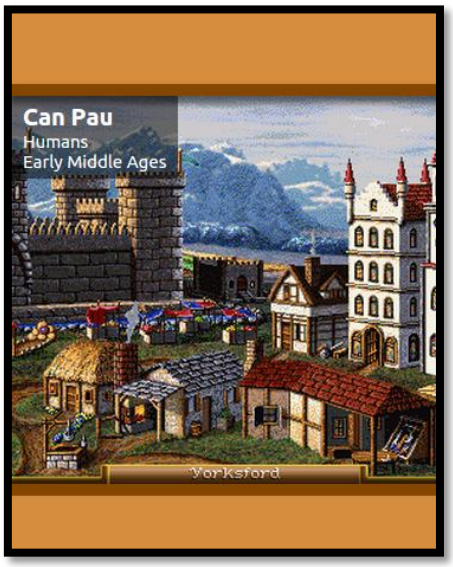

Figura 97: Barres de menú responsive – pantalla petita

Com es pot observar, a mesura que més estreta es fa la pantalla més importància guanyen les barres, fins que arriba a un punt (a la pantalla d'un mòbil) on aquestes passen a estar a les zones superior i inferior de la pantalla per a que sigui més còmode d'utilitzar.

Tot i que hi ha bastants elements responsive utilitzant aquesta tècnica, en les vistes en les que s'ha de llistar un conjunt d'objectes (edificis, recursos...) es fa realment complicat el poder apilar els elements. És per això que s'ha inclòs la llibreria de Bootstrap, que proporciona un sistema de columnes que s'encarrega d'apilar automàticament aquests elements.

Figura 98: Sistema de columnes Bootsrap – Pantalla gran

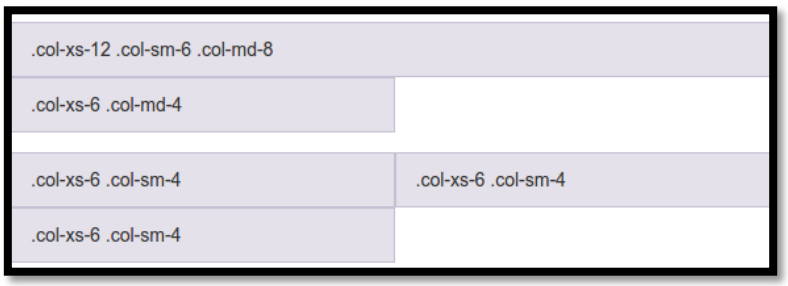

Figura 99: Sistema de columnes Bootsrap – Pantalla mitjana

En el projecte aquestes columnes es generen dintre dels diversos templates (veure apartat 9.1.3), aquest és un exemple d'html d'un template ja generat amb dos elements a dintre:

```
<div class="row edificis_recursos_llista">
 <div class="col-md-3 col-sm-4 edificis_recursos_llista_col">
  <div class="edificis_recursos_llista_item">
   <div class="edificis_recursos_llista_item_title">Exemple de Recurs</div>
   <div class="recursos_llista_item_img" style=" background-image:
        url('img/recursos/exemple.png'); ">
   </div>
   <div class="edificis_recursos_llista_item_text">Descripció</div>
  </div>
 </div>
 <div class="col-md-3 col-sm-4 edificis_recursos_llista_col">
  <div class="edificis_recursos_llista_item">
   <div class="edificis_recursos_llista_item_title">Exemple de Recurs2</div>
   <div class="recursos_llista_item_img" style="background-image:
        url('img/recursos/exemple2.png'); ">
   </div>
   <div class="edificis_recursos_llista_item_text">Descripció2</div>
  </div>
 </div>
</div>
```

El resultat dels fragments de codi generats de la mateixa manera són els següents:

Figura 100: Columnes de Bootsrap per dibuixar edificis – Pantalla gran

Figura 101: Columnes de Bootsrap per dibuixar edificis – Pantalla mitjana

Figura 102: Columnes de Bootsrap per dibuixar edificis – Pantalla petita

9.1.3 TEMPLATES

La pàgina web parteix d'index.html, però en el moment de carregar l'enrutador aquest crida la vista corresponent a la url.
Cada vista demana un seguit de dades al servidor per API Rest, i les utilitza per generar el codi html final a partir dels templates de la llibreria Underscore.
Un template és un fragment de codi html enriquit amb un llenguatge específic sobre ell, de manera que aplicant un objecte sobre un template es poden emplenar camps a partir de la informació de l'objecte, o fins i tot recórrer un array de l'objecte per generar codi html en bucle. En resum, utilitzar templates facilita la programació d'interfícies implicades amb camps dinàmics, i proporcionen una lectura més fàcil i ordenada.

En l'apartat 9.1.2 s'ha mostrat un codi html generat a partir d'un template. L'equivalent del template del codi anterior és:

```
<div class="row edificis_recursos_llista">
 <% _.each(recursos, function(recurs) { %>
  <div class="col-md-3 col-sm-4 edificis_recursos_llista_col">
   <div class="edificis_recursos_llista_item">
    <div class="edificis_recursos_llista_item_title"><%= recurs.nom %></div>
    <div class="recursos_llista_item_img" style="background-image:
       url('img/recursos/<%= recurs.urlimage %>');"></div>
    <div class="edificis_recursos_llista_item_text">
     <%= recurs.descripcio %>.
    </div>
   </div>
  </div>
 <% }); %>
</div>
```

Com es pot observar, tot el codi javascript dintre del template està delimitat per <% … %>. En aquest cas, el template espera tenir una taula d'objectes anomenada recursos com a paràmetre. A partir d'aquesta taula, underscore la recorre la taula i per cada paràmetre d'aquesta genera un fragment d'html més, és a dir, genera l'html de cada recurs de la taula recursos. Més endevant es pot veure com utilitza la informació dintre de recurs per generar tota la informació. D'aquesta manera, es pot generar el contingut de la pàgina automàticament.
Per aplicar el template i dibuixar-lo a la pàgina la vista fa el següent:

```
that.peticio("get","/usuaris/ciutat/"+idciutat+"/recursos",{},function(data2){
  var compiledTemplate = _.template(that.templates.get('ciutat_recursos'))( {
    user:data, ciutat: {background:'img/cities/'+data.urlimage}, recursos:data2
  });
  that.body.find(".ciutat_background").html( compiledTemplate );
  $(window).resize();
});
```

En aquest exemple, primer es fa la petició API Rest per obtenir els recursos de la ciutat. Un cop completada, genera el template compilat de ciutat_recursos, passant-li com a pàmetre un objecte amb els camps user, ciutat (que ja es disposaven anteriorment) i els recursos que s'han obtingut ara. Un cop compilat, s'afegeix a l'html de ciutat_background.

Amb l'exemple anterior (lleugerament modificat per mostrar més camps) obtenim el següent resultat.

Figura 103: Elements generats dinàmicament a través del template

9.1.4 SOCKETS

En el client el sistema de sockets està encapsulat al mòdul notificacionsSocket.js. Quan es crida el constructor del mòdul, el sistema es connecta:

```
this.socket= io('http://localhost:3000',{
  'reconnection': true,
  'reconnectionDelay': 1000,
  'reconnectionDelayMax' : 5000,
  'reconnectionAttempts': 100
});
```

Cal tenir en compte que, en el moment de pujar al servidor, s'ha de canviar localhost per el nom del domini.
Un cop connectat, el socket pot començar a rebre informació en temps real, usant els missatges esmentats a l'apartat 8.9.

```
this.notificacions=[0,0,0,0];//edifici,fabrica,civilitzacio,atac
var rebreNotificacioEdifici= function(){
  that.notificacions[0]++;
  $(".notificacioEdifici").show().html(that.notificacions[0]);
}
var rebreNotificacioFabrica= function(){
  that.notificacions[1]++;
  $(".notificacioFabrica").show().html(that.notificacions[1]);
}
var rebreNotificacioCivilitzacio= function(){
  that.notificacions[2]++;
  $(".notificacioCivilitzacio").show().html(that.notificacions[2]);
}
var rebreNotificacioAtac= function(){
  that.notificacions[3]++;
  $(".notificacioAtac").show().html(that.notificacions[3]);
}
this.socket.on('notificacioEdifici',function(obj){rebreNotificacioEdifici();});
this.socket.on('notificacioFabrica',function(obj){rebreNotificacioFabrica();});
this.socket.on('notificacioCivilitzacio',function(obj){rebreNotificacioCivilitzacio();});
this.socket.on('notificacioAtac',function(obj){rebreNotificacioAtac();});
this.socket.on('actualitzarRecursos',function(obj){that.actualitzarRecursos(obj);});
```

D'aquesta manera, depenguent de quin sigui el missatge que rep l'aplicació, actualitza i mostra un contador d'alertes o un altre.
També es pot veure el missatge actualitzarRecursos, que crida el mètode del mòdul amb el mateix nom que s'encarrega d'actualitzar els htmls que contenen la informació del nombre de recursos de la ciutat.
D'altra banda, quan es carreguen les diferents vistes afectades, s'encarreguen d'avisar al mòdul que ha de resetejar alguna de les notificacions i actualitzar la resta de la interfície. Per exemple, en la funció router_mostrarEdificisFabrica del mòdul app.js, un cop recarregat el menú i recuperades les dades del servidor, s'executa la següent línia:

```
that.notificacionsSocket.resetNotificacioFabrica();
```

Les funcions de reset que es criden en aquest cas són:

```
NotificacionsSocket.prototype.resetNotificacioEdifici= function(){
  this.notificacions[0]=0;
  $(".notificacioEdifici").hide().html("");
  if(this.notificacions[1]!=0)$(".notificacioFabrica").show().html(this.notificacions[1]);
  if(this.notificacions[2]!=0)$(".notificacioCivilitzacio").show().html(this.notificacions[2]);
  if(this.notificacions[3]!=0)$(".notificacioAtac").show().html(this.notificacions[3]);
}
NotificacionsSocket.prototype.resetNotificacioFabrica= function(){
  this.notificacions[1]=0;
  $(".notificacioFabrica").hide().html("");
  if(this.notificacions[0]!=0)$(".notificacioEdifici").show().html(this.notificacions[0]);
  if(this.notificacions[2]!=0)$(".notificacioCivilitzacio").show().html(this.notificacions[2]);
  if(this.notificacions[3]!=0)$(".notificacioAtac").show().html(this.notificacions[3]);
}
NotificacionsSocket.prototype.resetNotificacioCivilitzacio= function(){
  this.notificacions[2]=0;
  $(".notificacioCivilitzacio").hide().html("");
  if(this.notificacions[1]!=0)$(".notificacioFabrica").show().html(this.notificacions[1]);
  if(this.notificacions[0]!=0)$(".notificacioEdifici").show().html(this.notificacions[0]);
  if(this.notificacions[3]!=0)$(".notificacioAtac").show().html(this.notificacions[3]);
}
NotificacionsSocket.prototype.resetNotificacioAtac= function(){
  this.notificacions[3]=0;
  $(".notificacioAtac").hide().html("");
  if(this.notificacions[1]!=0)$(".notificacioFabrica").show().html(this.notificacions[1]);
  if(this.notificacions[2]!=0)$(".notificacioCivilitzacio").show().html(this.notificacions[2]);
  if(this.notificacions[0]!=0)$(".notificacioEdifici").show().html(this.notificacions[0]);
}
```

Figura 104: Exemple de notificació

9.2 SERVIDOR

9.2.1 ESTRUCTURA DE DIRECTORIS

Observant la figura 105 es pot veure l'estructura de directoris del projecte del servidor.

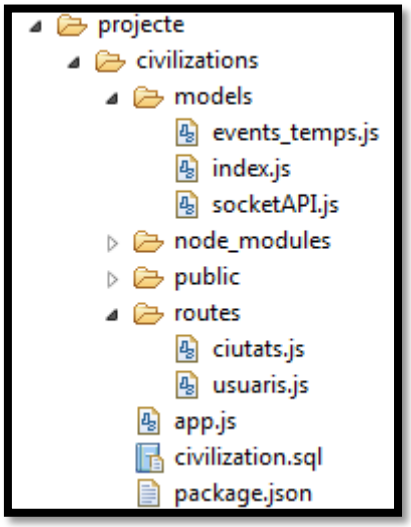

Figura 105: Estructura de directoris del servidor

.idea: carpeta generada per l'IDE IntelliJIdea. Ignorada pel Git a l'hora de pujar els fitxers.
models: a dintre de models hi ha index.js i els mòduls programats que implementen funcions necessàries per al funcionament del servidor.
node_modules: carpeta on s'instalen els mòduls necesàris. Ignorada pel Git a l'hora de pujar els fitxers.
public: carpeta on està guardada l'estructura de directoris del client (veure capítol 9.1.1).
routes: carpeta on hi ha els mòduls que apliquen un seguit de funcions relacionades amb algun objecte de la base de dades (usuari,ciutat...).
.gitignore conté les carpetes que no s'han de pujar al servidor.
app.js és l'executable de node i fa el set de les primeres variables del servidor.
civilization.sql: és l'script que genera el sistema de proves esmentat a l'apartat 8.3.1.
package.json: inclou el json amb les llibreries necessàries per al projecte i que l'npm les pugui instal·lar automàticament.

9.2.2 PROCEDURES I TRIGGERS

Per treure càrrega computacional a la llibreria node.js, així com ajudar a mantenir la consistència entre les taules, hi ha moltes petites funcionalitats que s'han fet amb procedures o triggers de MySQL. En aquest apartat es comenten alguns d'ells, la resta es poden trobar a l'Annex (apartat 14.1).

Un dels factors més importants en la estabilitat del sistema és mantenir actualitzat el guany/hora dels recursos d'una ciutat, per a això hi ha un conjunt de triggers que criden el procedure que s'encarrega de recalcular els guanys/hora.

trigger_afegir_edifici_recurs/trigger_modificar_edifici_recurs/trigger_eliminar_edifici_recurs:
Després d'afegir, canviar o eliminar el que genera un edifici es recalculen els recursos generats per les ciutats cada hora.

```
delimiter ;
DROP TRIGGER if exists trigger_inserir_recurs;
DELIMITER //
CREATE TRIGGER trigger_afegir_edifici_recurs AFTER INSERT ON edifici_recurs_nivell_genera
  FOR EACH ROW BEGIN
    CALL recalcular_guany_all_ciutats();
  END
//
DELIMITER ;
DROP TRIGGER if exists trigger_modificar_edifici_recurs;
DELIMITER //
CREATE TRIGGER trigger_modificar_edifici_recurs AFTER UPDATE ON edifici_recurs_nivell_genera
  FOR EACH ROW BEGIN
    CALL recalcular_guany_all_ciutats();
  END
//
DELIMITER ;
DROP TRIGGER if exists trigger_modificar_edifici_recurs;
DELIMITER //
CREATE TRIGGER trigger_eliminar_edifici_recurs AFTER DELETE ON edifici_recurs_nivell_genera
  FOR EACH ROW BEGIN
    CALL recalcular_guany_all_ciutats();
END
DELIMITER ;
```

trigger_modificar_edifici_nivell: Després de pujar el nivell d'un edifici d'una ciutat es recalculen els guanys.

```
delimiter ;
DROP TRIGGER if exists trigger_modificar_edifici_nivell;
DELIMITER //
CREATE TRIGGER trigger_modificar_edifici_nivell AFTER UPDATE ON ciutat_edifici_nivell
  FOR EACH ROW BEGIN
    CALL recalcular_guany_ciutat(NEW.idciutat);
  END
//
DELIMITER ;
```

Com es pot veure, hi ha dos procedures diferents que es criden en els triggers. El procedure recalcular_guany_all_ciutats només recorre la taula de ciutats i crida el procedure recalcular_guany_ciutat(idciutat_) que a la vegada, fa un recorregut per tots els tipus de recursos i crida recalcular_guany_ciutat_recurs(idciutat_,idrecurs_), el qual donada una ciutat i un recurs, recalcula el guany/hora a partir de tots els edificis que té la ciutat.

```
delimiter ;
DROP PROCEDURE if exists recalcular_guany_ciutat_recurs;
delimiter //
CREATE PROCEDURE recalcular_guany_ciutat_recurs (idciutat_ INT,idrecurs_ INT)
BEGIN
DECLARE quantitat_ INTEGER DEFAULT 0;
DECLARE quantitat_fabrica INTEGER DEFAULT 0;
DECLARE quantitat_resta INTEGER DEFAULT 0;
-- El que generen els edificis de recursos
SELECT IFNULL(SUM(quantitat),0) INTO quantitat_ FROM ciutat_edifici_nivell cen
,edifici_recurs_nivell_genera erng WHERE erng.idnivell=cen.nivell AND cen.idedifici=erng.idedifici
AND cen.idciutat=idciutat_ AND erng.idrecurs=idrecurs_;
-- El que generen les fàbriques
SELECT IFNULL(SUM(quantitat),0) INTO quantitat_fabrica FROM ciutat_edifici_nivell cen
,edifici_fabrica_nivell_genera erng WHERE erng.idnivell=cen.nivell AND cen.idedifici=erng.idedifici
AND cen.idciutat=idciutat_ AND erng.idrecurs=idrecurs_;
SET quantitat_ = quantitat_ + quantitat_fabrica;
-- El que consumeixen les fàbriques
SELECT IFNULL(SUM(quantitat),0) INTO quantitat_resta FROM ciutat_edifici_nivell cen
,edifici_fabrica_nivell_gasta_recurs erng WHERE erng.idnivell=cen.nivell AND
cen.idedifici=erng.idedifici AND cen.idciutat=idciutat_ AND erng.idrecurs=idrecurs_;
SET quantitat_ = quantitat_ - quantitat_resta;
UPDATE ciutat_recurs SET increment_hora=quantitat_ WHERE idciutat=idciutat_ AND
idrecurs=idrecurs_;
END //
DELIMITER ;
```

Un altre procés important és l'encarregat de computar els guanys/hora quan el servidor ho demana. Com que en ocasions ens interessa calcular diverses hores a la vegada per a algunes ciutats (si el

servidor s'ha aturat quan estava fent el càlcul d'una hora, hi haurà ciutats amb l'hora nova i les altres amb l'antiga), s'han creat dos procedures.
El procés extern és increment_recursos_tick(), que el crida el servidor cada hora. Recorre totes les ciutats i crida el segon procés, increment_recursos_tick_ciutat(idciutat_,nhores), posant nhores a 1. D'aquesta manera si una ciutat no té el càlcul de les últimes 6 hores el pot fer d'igual manera amb el segon procedure:

```
delimiter ;
DROP PROCEDURE if exists increment_recursos_tick_ciutat;
delimiter //
CREATE PROCEDURE increment_recursos_tick_ciutat (idciutat_ INT,ncops INT)
BEGIN
DECLARE v_finished INTEGER DEFAULT 0;
DECLARE idrecurs_ INTEGER DEFAULT 0;
DEClARE recurs_cursor CURSOR FOR SELECT idrecurs FROM recurs;
--NOT FOUND handler
DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
OPEN recurs_cursor;
get_recurs: LOOP
  FETCH recurs_cursor INTO idrecurs_;
  IF v_finished = 1 THEN
    LEAVE get_recurs;
  END IF;
  UPDATE ciutat_recurs SET quantitat=quantitat+(increment_hora*ncops) WHERE
idciutat=idciutat_ AND idrecurs=idrecurs_;
END LOOP get_recurs;
CLOSE recurs_cursor;
UPDATE `ciutat` SET `ultim_update_recursos`=NOW() WHERE idciutat=idciutat_;
END //
DELIMITER ;
```

9.2.3 VIEWS

D'igual manera que amb els procedures, per facilitar la complexitat del servidor en la mesura del possible s'han preparat diverses vistes a la base de dades. En aquest apartat es comenten algunes d'elles, la resta es poden consultar a l'Annex (arpartat 14.2).

Un exemple d'una de les vistes que utilitza el servidor és vista_edificis_fabrica_ciutat. Es crida dintre de la funció ciutat_obtenir_edificis_tipus del fitxer ciutats.js. La funció és retornar, a partir d'un idioma, una ciutat, i una civilització (pot ser una civilització pare de les civilitzacions de la ciutat), retorna tota la informació necessària per poder montar la taula d'objectes d'edificis de tipus fàbrica.

```
CREATE VIEW `vista_edificis_fabrica_ciutat` AS
SELECT e.idedifici, t1.text as nom, t2.text as descripcio,c.idciutat,c.nom as ciutatnom,IF(cen.nivell IS
NULL,0,cen.nivell) AS nivell, t.fi as fi_millora, r.idrecurs as idrecurs_gasta , ecmr.quantitat as
quantitat_gasta, ecm.temps_millora,r.urlimage as recurs_gasta_image,t3.text as
recurs_gasta_nom, t4.text as recurs_gasta_descripcio, r2.idrecurs as idrecurs_genera, r2.urlimage
as recurs_genera_image, t5.text as recurs_genera_nom, t6.text as
recurs_genera_descripcio,IF(cen.nivell IS NULL,0, IF(cen.nivell=0,0,erng.quantitat)) as
recurs_genera_quantitat, efngr.idrecurs as recurs_consumeix_id,efngr.quantitat as
recurs_consumeix_quantiat, r3.urlimage as recurs_consumeix_image, t7.text as
recurs_consumeix_nom, t8.text as recurs_consumeix_descripcio, e.idcivilitzacio, t2.ididioma as
ididioma
FROM traduccio t1, traduccio t2, traduccio t3, traduccio t4, traduccio t5, traduccio t6, traduccio t7,
traduccio t8,edifici_fabrica er, edifici_cost_millorar_recurs ecmr,edifici_cost_millorar ecm,recurs r,
edifici_fabrica_nivell_genera erng, recurs r2, edifici_fabrica_nivell_gasta_recurs efngr ,recurs r3,
ciutat c
LEFT JOIN edifici e ON e.idcivilitzacio IS NOT NULL
LEFT JOIN ciutat_edifici_nivell cen ON cen.idedifici=e.idedifici AND cen.idciutat=c.idciutat
LEFT JOIN temps_pendents t ON t.idtemps=cen.temps_millora
WHERE t1.idtraduccio=e.nom AND t2.idtraduccio=e.descripcio AND t1.ididioma=t2.ididioma AND
e.idedifici=er.idedifici AND ( (cen.nivell IS NULL AND ecmr.idnivell=1) OR cen.nivell=ecmr.idnivell-1 )
AND ecmr.idrecurs=r.idrecurs AND ecmr.idedifici=er.idedifici AND ecm.idnivell=ecmr.idnivell AND
ecm.idedifici=er.idedifici AND erng.idedifici=er.idedifici AND r2.idrecurs=erng.idrecurs AND ( (
(cen.nivell IS NULL OR cen.nivell=0) AND erng.idnivell=1) OR (cen.nivell IS NOT NULL AND
cen.nivell!=0 AND erng.idnivell=ecm.idnivell-1) ) AND r3.idrecurs=efngr.idrecurs AND
t3.idtraduccio=r.nom AND t4.idtraduccio=r.descripcio AND t3.ididioma=t4.ididioma AND
t4.ididioma=t2.ididioma AND t5.idtraduccio=r2.nom AND t6.idtraduccio=r2.descripcio AND
t5.ididioma=t4.ididioma AND t6.ididioma=t2.ididioma AND t7.idtraduccio=r3.nom AND
t8.idtraduccio=r3.descripcio AND t7.ididioma=t4.ididioma AND t8.ididioma=t2.ididioma
ORDER BY e.idedifici,c.idciutat,ididioma;
```

Al servidor la vista es crida d'aquesta manera amb la llibreria mysql2.

```
var obtenir_edificis_civilitzacio=function(c,callback) {
  connection.execute('SELECT * from vista_edificis_fabrica_ciutat WHERE idciutat=:x AND
ididioma=:y AND idcivilitzacio=:z', { x: req.params.idciutat, y: req.session.ididioma,z:c},
    function (err, rows) {
      if (!err) {
        ...
    }
  });
  ...
}
```

El cas de vista_recursos_ciutat és més senzill però amb el mateix procediment. S'utilitza dintre de getRecursos, i passant com a paràmetres l'idioma i la ciutat, retorna la informació de tots els recursos en l'idioma seleccionat juntament amb la quantitat que en poseeix la ciutat.

```
CREATE VIEW `vista_recursos_ciutat` AS
SELECT r.idrecurs, ciur.quantitat ,t1.text as nom, t2.text as descripcio, c.idciutat as idciutat,
r.urlimage, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, recurs r, ciutat c, ciutat_recurs ciur
WHERE t1.idtraduccio=r.nom AND t2.idtraduccio=r.descripcio AND t1.ididioma=t2.ididioma AND
ciur.idrecurs=r.idrecurs AND ciur.idciutat=c.idciutat
GROUP BY r.idrecurs,t2.ididioma,c.idciutat
ORDER BY c.idciutat,r.idrecurs;

getRecursos: function(req,res){
  if(!req.session.ididioma)req.session.ididioma='en';
  if( req.params.idciutat) {
    that.ciutatSession(req, req.params.idciutat, function () {
      connection.execute('SELECT * from vista_recursos_ciutat WHERE idciutat=:x AND
ididioma=:y', { x: req.params.idciutat, y: req.session.ididioma}, function (err, rows) {
        if (!err) { console.log("recursos: " + JSON.stringify(rows)); res.send(200, rows); }
        else { res.send(500, { error: err.messsage }); }
      });
    }, function (err) {
      res.send(500, { error: err });
    });
  } else res.send(500, "idciutat required");
}
```

9.2.4 EVENTS TEMPORALS

El mòdul d'events temporals es podria considerar el més essencial del joc, ja que és el que s'encarrega de disparar tots els events temporals en el moment correcte, i si per qualsevol raó el servidor no s'ha estat executant amb anterioritat, és l'encarregat de mantenir la consistència amb tots els events que queden pendents en el passat.
Abans d'explicar el comportament del mòdul, cal definir l'estructura de l'objecte event:

```
event = {
            id: id de l'event a la base de dades,
            idciutat: id de la ciutat que ha originat l'event,
            tipusEvent: tipus de l'event (millora d'edifici de recursos, fàbrica...),
            segons: segons per disparar-se,
            callback: generat a partir de tipusEvent.
};
```

Així doncs, la primera acció que el mòdul events_temporal duu a terme en la inicialització del servidor és reactivar l'interval d'actualització de recursos, cridant el procés increment_recursos_tick() esmentat a l'apartat 9.2.2. Sempre es parteix de l'hora en punt per començar l'interval. Per això es genera un timeout que s'espera fins al canvi s'hora per executar-lo.

```
var inicialitzarInterval=function() {
    setTimeout(function () {
      connection.execute('CALL increment_recursos_tick();', {}, function (err, rows) { });
      event_actualitzar = setInterval(function () {
        connection.execute('CALL increment_recursos_tick();', {}, function (err, rows) { });
      }, TEMPS_ACTUALITZACIO_RECURSOS * 1000);
    }, (60 - new Date().getMinutes()) * 60 * 1000);
};
inicialitzarInterval();
```

Després d'iniciar l'interval, el mòdul declara totes les seves funcions privades, i un cop acaba passa a recuperar els events de temps programats. Es poden donar dos casos amb aquests events:

- Ja haurien d'haver acabat. En aquest cas es torna a generar l'objecte event com si fós un event nou i es dispara amb temps 0 amb el procediment normal.
- Encara no ha passat tot el temps. En aquest cas s'ha de calcular el temps restant i generar l'objecte event amb el timeout corresponent.

```
connection.execute('SELECT * FROM vista_ciutats_edificis_pendents', {},
  function (err, rows) {
    if (!err) {
      for (var i = 0; i < rows.length; i++) {
        var event;
        switch (rows[i].tipus) {
          case 0:
            event = { id: rows[i].idtemps, idciutat: rows[i].idciutat, tipusEvent: 0, segons: 0 };
            event.callback = generarFuncio(event);
            break;
          case 1:
            event = { id: rows[i].idtemps, idciutat: rows[i].idciutat, tipusEvent: 1, segons: 0 };
            event.callback = generarFuncio(event);
            break;
        }
        if (rows[i].fi.getTime() < new Date().getTime()) {
          //Events que ja han passat
          crearTimeout(event);
        } else{
          //Dels events que no han passat s'ha de generar el seu timeout
          event.segons=(new Date().getTime()-rows[i].fi.getTime())/1000;
          crearTimeout(event);
        }
      }
    }
  }
);
```

Un cop ha tractat els events temporals pendents, el mòdul passa a calcular els recursos que s'haurien d'haver generat de format automàtica des de l'última actualització. Per fer-ho, recupera el camp ultim_update_recursos (taula Ciutat. Veure apartat 8.7.2.2) de totes les ciutats. Un cop calculades quantes unitats de temps té pendents per aplicar crida al procés increment_recursos_tick_ciutat (veure apartat 9.2.2), passant-li de paràmetres la ciutat i les unitats de temps pendents.

```
connection.execute('SELECT idciutat,ultim_update_recursos FROM ciutat', {},
function (err, rows) {
  if (!err) {
    for (var i = 0; i < rows.length; i++) {
      var diff = new Date().getTime() - (rows[i].ultim_update_recursos.getTime() -
(rows[i].ultim_update_recursos.getMinutes() * 60 * 1000));
      var unitatsdeTemps = parseInt(diff / (TEMPS_ACTUALITZACIO_RECURSOS * 1000));
      connection.execute('CALL increment_recursos_tick_ciutat(:x,:y);', {x: rows[i].idciutat, y:
unitatsdeTemps}, function (err, rows) { });
    }
  }
});
```

9.2.5 SOCKETS

Totes les funcions relacionades amb la gestió de la llibreria de sockets estan encapsulades dintre del mòdul socketAPI.js. Aquest mòdul es crea desde index.js i s'afegeix en al resta de mòduls. En el moment de crear-se, es vincula amb el servidor per escoltar les peticions de connexió.
Dintre del mòdul també es crea una estructura que guarda en tot moment la relació entre usuaris i sockets, és a dir, una taula d'usuaris on cada usuari té una taula de sockets connectats. És important tenir-ho en compte, ja que es pot donar el cas que un usuari estigui loguejat desde varis dispositius o navegadors simultàniament. En aquest cas quan s'hagi d'enviar un missatge ha de ser a través de tots els sockets de l'usuari per a que tots rebin la informació.

```
var io=require('socket.io').listen(server);
var users_connected=[];
var SessionSockets = require('session.socket.io') ,
sessionSockets = new SessionSockets(io, sessionStore, cookieParser);
sessionSockets.on('connection', function (err, socket, s) {
  var session=s;
  if(s) {
    if (!users_connected[s.idusuari])users_connected[s.idusuari] = {};
    users_connected[s.idusuari][socket.id] = socket;
  }
  socket.on('disconnect',function(data){
    if(socket.data && socket.data.idUser){
      delete users_connected[socket.data.idUser][socket.id];
      if(Object.getOwnPropertyNames(users_connected[socket.data.idUser]).length ==0)delete
users_connected[socket.data.idUser];
    }
  });
});
```

Un cop es genera l'estructura d'usuaris i es manté consistent, es pot programar una petita funció que envii un missatge i l'objecte a un usuari a partir de la seva id.

```
var getSocketsFromIdUsuari=function(idUsuari){ return users_connected[idUsuari]; }
emitUser: function(idUsuari,missage,obj){
  var sockets=getSocketsFromIdUsuari(idUsuari);
  for (var idSocket in sockets) { sockets[idSocket].emit(missage,obj); }
}
```

Per últim des de la resta de mòduls, s'ha afegit una funció per generar la crida emitUser de forma senzilla.

```
notificar:function(idUsuari,tipusNotificacio){
//tipusNotificacio: 0 edifici, 1 fabrica, 2 civilitzacio, 3 atac
  switch(tipusNotificacio){
    case 0: that.emitUser(idUsuari,'notificacioEdifici',{});
            break;
    case 1: that.emitUser(idUsuari,'notificacioFabrica',{});
            break;
    case 2: that.emitUser(idUsuari,'notificacioCivilitzacio',{});
            break;
    case 3: that.emitUser(idUsuari,'notificacioAtac',{});
            break;
  }
}
```

Amb socketAPI carregat, quan al mòdul events_temporals salta un event de millorar fàbrica, s'efectua una notificació que li arribarà en temps real a l'usuari.

```
socketAPI.notificar(event.idciutat,1);
```

9.3 ADMINISTRAR SERVIDOR DE PROVES

Com s'ha esmentat a l'apartat 1.2, s'ha montat un servidor de proves a la raspberryPi, on s'ha testejat el funcionament del servidor donant servei de forma continuada durant diversos dies.

9.3.1 OBTENIR CODI DEL REPOSITORI I CREAR BDD

Durant tot el projecte s'ha utilitzit GIT, ja que proporciona control de versions i permet que la raspberryPi pugui actualitzar de forma fàcil i còmoda els canvis en el servidor. Així doncs, un cop a la ruta ~/node/civilizations s'ha de fer pull del repositori i carregar el nou backup de la base de dades.

```
git pull
mysql -uroot -pquercus2012 -Dcivilization < civilization.sql
```

9.3.2 CONFIGURACIÓ DEL DOMINI I DE APACHE

Tenint en possessió de forma temporal el domini error404.cat, s'ha configurat el DNS per apuntar cap a la IP fixa del router on està connectada la raspberryPI. El router en qüestió està configurat per assignar IP local fixa a la raspberryPI i fer forward de les peticions de port 80.

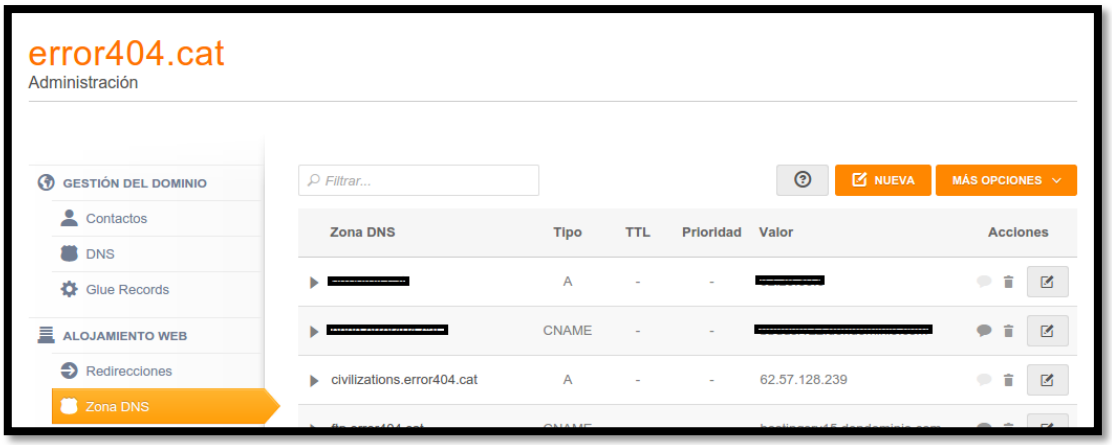

Figura 106: DNS del Host del servidor

Un cop direccionat el tràfic a la raspberryPI, és necessàri configurar el domini a l'Apache per donar-li a on redireccionar totes les peticions. Per fer-ho, cal anar al directori de la ruta /etc/apache/sites-available/, i allí crear un nou fitxer que s'anomenarà civilizations.error404.

```
<VirtualHost *:80>
    ServerName civilizations.error404.cat
    ServerAlias www.civilizations.error404.cat
    ServerAdmin webmaster@localhost
    ProxyRequests off
    <proxy *>
        Order deny,allow
        Allow from all
    </proxy>
    <location "/">
        ProxyPass http://localhost:3000/ retry=0
        ProxyPassReverse http://localhost:3000/
    </location>
</VirtualHost>
```

Amb aquest fitxer li estem comunicant a l'Apache que totes les peticions que es rebin pel port 80 i que vinguin amb el nom de civilizations.error404.cat o www.civilization.error404.cat es redireccionen amb un proxy al port 3000, que es el port on està obert el servidor del joc.
Per activar els canvis cal activar el nou fitxer que s'ha creat.

```
sudo a2ensite civilizations.error404
service apache2 reload
```

9.3.2 MANTENIR FUNCIONANT EL SERVIDOR

Per mantenir funcionant el servidor s'ha utilitzat una llibreria global de nodejs anomenada Forever. Aquesta llibreria, s'encarrega de mantenir obert de forma permanent un programari nodejs, o dit d'altra manera, si el programari acaba (de forma correcta o amb error), el torna a executar de forma contínua.

```
forever start app.js
```

D'aquesta manera s'assegura que encara que es doni un cas en el que el servidor es sobrecarrega o es troba amb alguna excepció, en pocs segons s'haurà tornat a obrir sense que l'usuari tingui temps de notar cap malfuncionament.

9.3.3 SERVEI PER CONTROLAR EL SERVIDOR I INICI AUTOMÀTIC

Tot i que en l'apartat 9.3.2 s'ha mostrat com executar el servidor de forma continuada amb la comanda forever, el servidor encara no es pot considerar que funciona de forma automàtica, ja que encara no s'obre juntament amb la raspberriPi, ni es genera cap tipus de log, ni es té control directe sobre el procés.
Per solucionar tots aquests problemes s'ha creat un servei a /etc/init.d anomenat executa_servidor que s'encarrega de tot el que s'ha esmentat.

```
NAME="executa_servidor"
NODE_BIN_DIR=""
NODE_PATH="/home/Super_P91/node/civilizations"
APPLICATION_PATH="/home/Super_P91/node/civilizations/app.js"
PIDFILE="/var/run/executa_servidor.pid"
LOGFILE="/var/log/executa_servidor.log"
MIN_UPTIME="5000"
SPIN_SLEEP_TIME="2000"

PATH=$NODE_BIN_DIR:$PATH
export NODE_PATH=$NODE_PATH
start() {
  echo "Starting $NAME"
  forever \
    --pidFile $PIDFILE \
    -a \
    -l $LOGFILE \
    --minUptime $MIN_UPTIME \
    --spinSleepTime $SPIN_SLEEP_TIME \
    start $APPLICATION_PATH 2>&1 > /dev/null &
  RETVAL=$?
}
stop() {
  if [ -f $PIDFILE ]; then
    echo "Shutting down $NAME"
    forever stop $APPLICATION_PATH 2>&1 > /dev/null
    rm -f $PIDFILE
    RETVAL=$?
  else
    echo "$NAME is not running."
    RETVAL=0
  fi
}
...
```

D'aquesta manera el servidor s'obre automàticament, i es pot reobrir fàcilment amb

```
sudo service executar_servidor restart
```

10 RESULTATS

S'han realitzat totes les tasques planificades inicialment, amb la inclusió d'algunes funcionalitats que estàven previstes per al futur. A continuació es mostren varies captures del videojoc realitzat, mostrant com varia la interfície segons la mida del navegador.
La primera pantalla correspon al login/registre, on l'usuari pot entrar al sistema amb les seves credencials o registrar-se de nou i crear una ciutat nova.

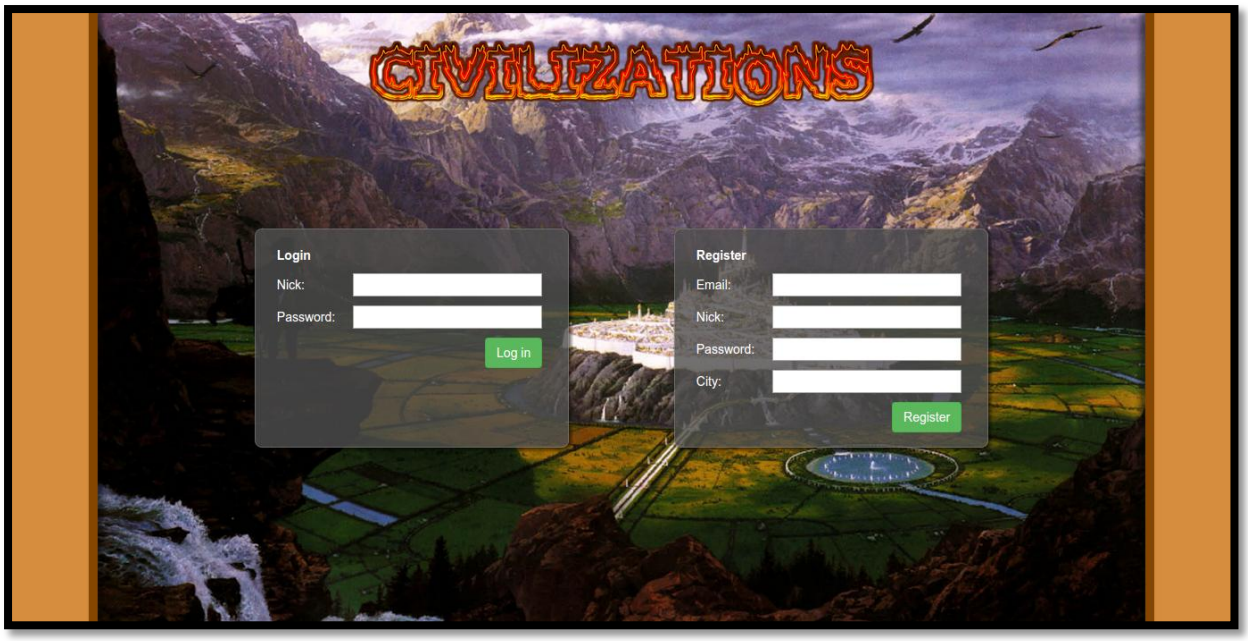

Figura 107: Vista Login/Registre – Pantalla gran

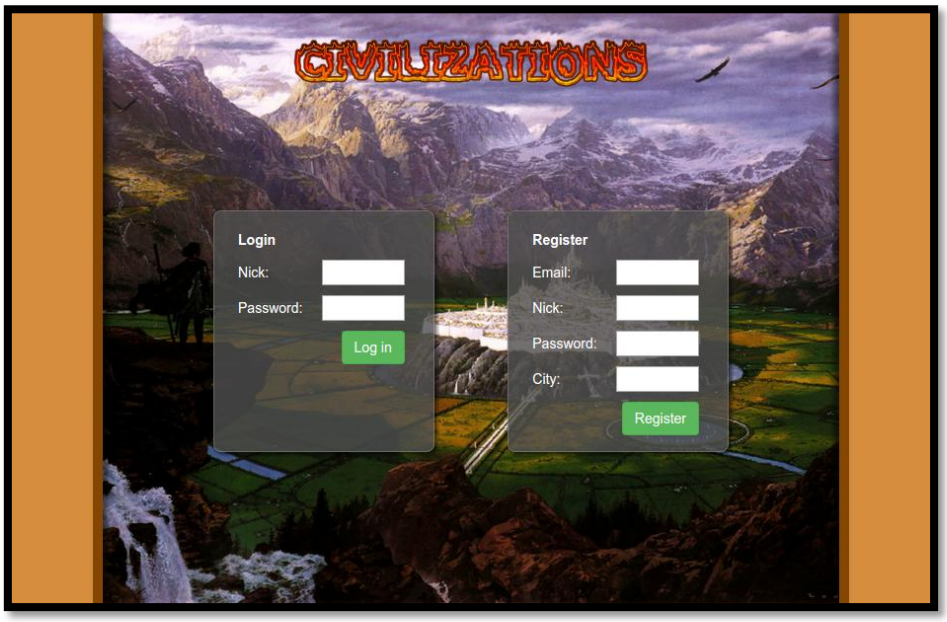

Figura 108: Vista Login/Registre – Pantalla mitjana

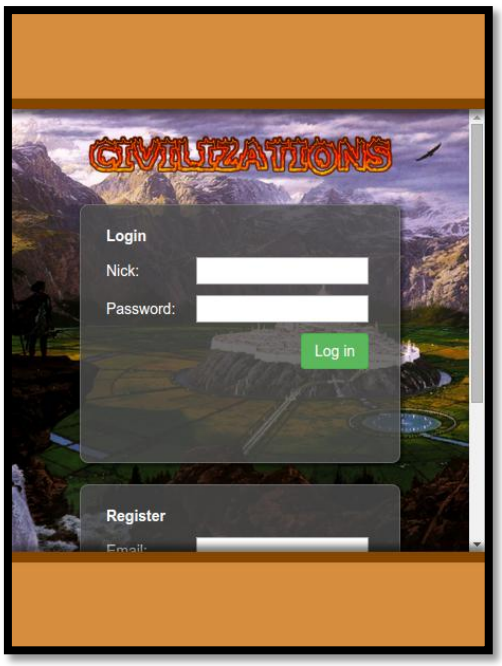

Figura 109: Vista Login/Registre – Pantalla petita

Un cop dintre la ciutat, l'usuari veu una primera vista d'aquesta i el llistat dels recursos més importants dels que disposa.

Figura 110: Vista ciutat – Pantalla gran

Figura 111: Vista ciutat – Pantalla mitjana

Figura 112: Vista ciutat – Pantalla petita

Com es pot veure a la figura 113, el fons de la ciutat varia segons la civilització principal d'aquesta.

Figura 113: Vista ciutat amb la civilització Yamato Period

Si l'usuari va a la vista de civilitzacions, podrà veure les seves civilitzacions principal i secundària, juntament amb les possibles millores.

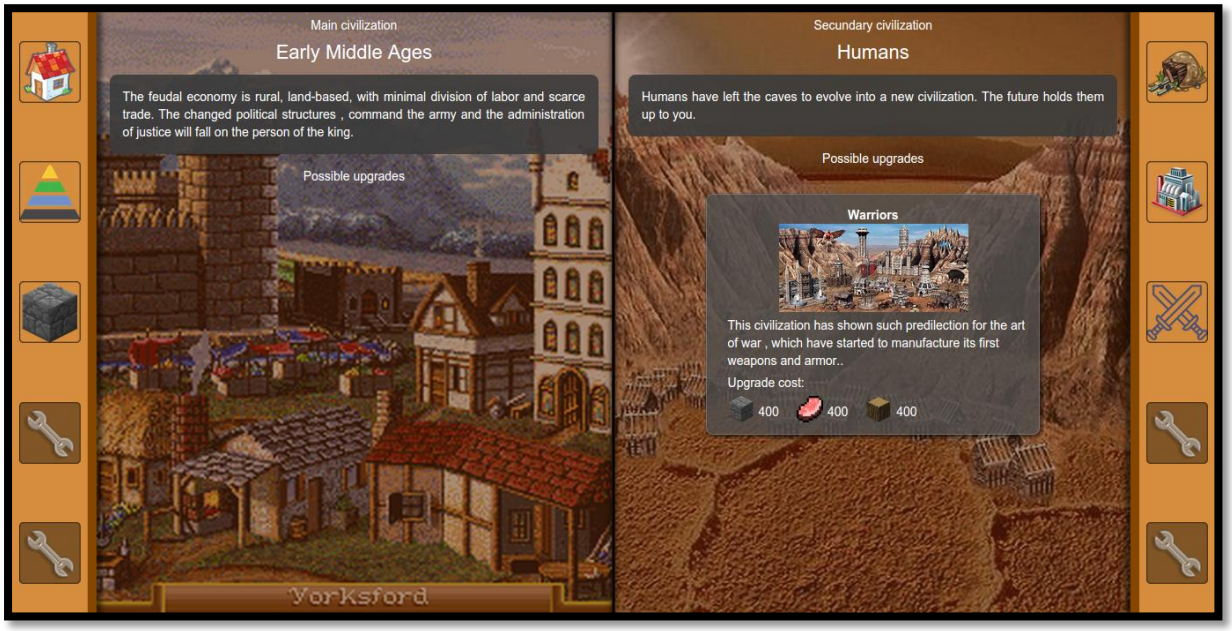

Figura 114: Vista civilitzacions – Pantalla gran

Figura 115: Vista civilitzacions – Pantalla mitjana

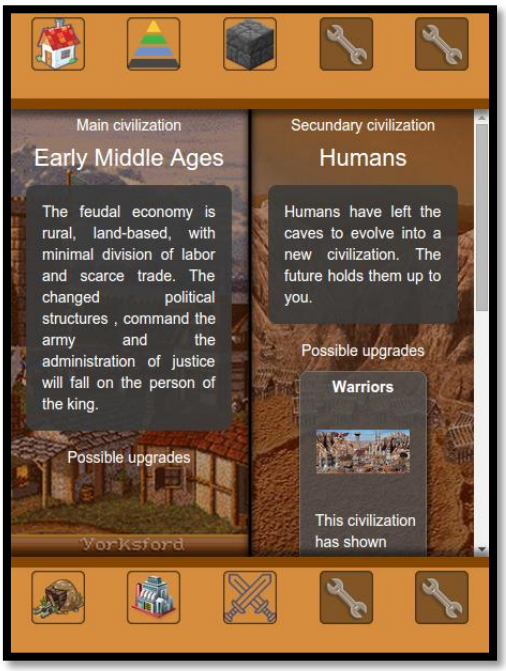

Figura 116: Vista civilitzacions – Pantalla petita

Com es pot observar a la figura 117, si una ciutat té una civilització a Warriors li apareixen les dues possibilitats per millorar.

Figura 117: Vista civilitzacions amb dues possible millores

Si l'usuari va a la vista de recursos veurà llistats tots els recursos del sistema, inclosos els que no estan directament relacionats amb les seves civilitzacions, així com la quantitat i guany/hora.

Figura 118: Vista recursos – Pantalla gran

Figura 119: Vista recursos – Pantalla mitjana

Figura 120: Vista recursos – Pantalla petita

Si l'usuari va a la vista d'edificis, li apareixeran llistats tots els edificis amb tota la informació complerta, és a dir, en quin nivell estan a la ciutat, el cost de recursos i temps de millorar-los i el guany/hora després de la millora.

Figura 121: Vista edificis – Pantalla gran

Figura 122: Vista recursos – Pantalla mitjana

Figura 123: Vista recursos – Pantalla petita

Si un edifici s'està millorant apareixerà amb una animació on va canviant el tamany i color del seu quadre.

Figura 124: Un edifici té l'animació de millora

Si l'usuari va a la vista vista de fàbriques, li apareixeran llistades totes les fàbriques amb la informació complerta, que és la mateix que en el cas d'edificis però afegint el cost/hora que suposa aplicar la millora.

Figura 125: Vista fàbriques – Pantalla gran

Figura 126: Vista fàbriques – Pantalla mitjana

Figura 127: Vista fàbriques – Pantalla petita

SI l'usuari va a la vista d'atacs, tindrà la opció de fer un atac a un altre usuari de forma aleatòria pagant els costos d'aliment i armes.

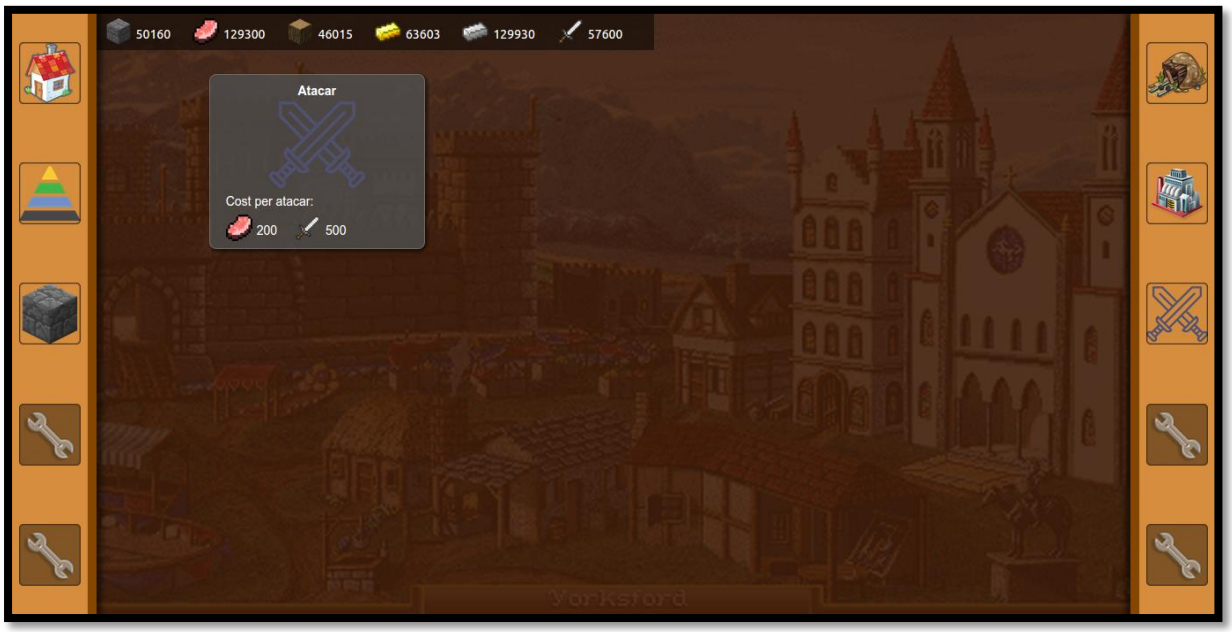

Figura 128: Vista atacs – Pantalla gran

Figura 129: Vista atacs – Pantalla mitjana

Figura 130: Vista atacs – Pantalla petita

En tot el sistema es poden rebre notificacions en temps real, les quals apareixen a sobre de la icona d'on provenen. Aquestes notificacions tenen una petita animació on cada 10 segons fa 2 petits parpadejos en blanc per captar l'atenció de l'usuari.

Figura 131: Notificacions – Pantalla gran

Figura 132: Notificacions – Pantalla mitjana

Figura 133: Notificacions – Pantalla petita

Per últim, tot i que per aplicar de forma oficial la traducció cal afegir la traducció de la part del client. Cridant a l'API Rest es pot canviar l'idioma per a que tota la informació del servidor es retorni en castellà.

Figura 134: Traducció de textos de base de dades

11 CONCLUSIONS

11.1 TEMPORITZACIÓ

És difícil que la temporització planificada inicialment en un projecte es correspongui fidelment a la realitat. La temporització real del temps dedicat és com a es mostra en la figura 135.

Es pot observar que algunes tasques s'han allargat bastant, principalment implementar tot el mòdul d'edificis i la primera versió dels atacs. Tot i així, la temporalització final no s'ha vist afectada per aquests retrassos.

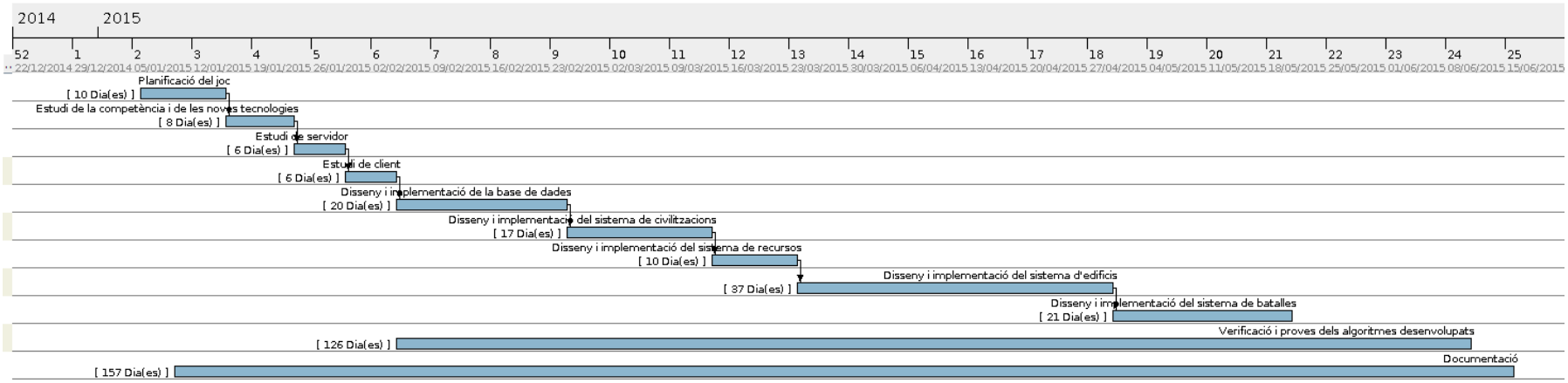

Figura 135: Diagrama de Gantt final

11.2 CONCLUSIONS

Durant l'elaboració del Projecte de Final de Carrera s'ha arribat a diversos conclusions:

- És molt important realitzar un bon anàlisis de la feina que comporta i dels problemes que poden sorgir. Tot i que sembli que és una inversió de temps extra, és important saber a on es vol arribar i fins a on es pot abarcar.

- Quan treballem amb projectes web és molt important saber aprofitar tot el que ja està fet, doncs l'objectiu de duur a terme un projecte és arribar al resultat esperat amb el menor cost econòmic, i en la programació web existeixen moltes llibreries que s'ocupen de tot tipus de funcionalitats.

- Un sola persona no pot desenvolupar un gran projecte. En el desenvolupament d'un videojocs hi intervenen persones amb diferents rols, com per exemple un dissenyador gràfic.

- Durant el projecte s'han tractat vàris llenguatges (javaScript, html5, css3, MySQL, ShellScript). Tot i que de la majoria ja disponc d'experiència laboral el projecte ha servit per repassar i consolidar coneixements.

12 TREBALL FUTUR

Un videojoc com aquest és molt gran, i es podria dir que en aquest projecte de final de carrera només s'han marcat les funcionalitats més bàsiques. Com s'ha vist a l'apartat 8.7.1, la base de dades original és molt més gran i complexa que la que s'ha implementat en aquest projecte. Així doncs, el treball futur més a curt termini del videojoc és:

- Acabar el mòdul de batalla: Com s'ha vist al diagrama d'Entitat-Relació, tot el mòdul de batalles és molt més gran, incloent diversos tipus d'unitats d'exèrcit per poder atacar o inclús edificis de tipus defensa.

- Fer el mòdul d'edifici d'investigació: Fer que per a certes millores d'edificis o unitats d'exèrcit sigui necessàri duur a terme alguna investigació relacionada.

- Fer el mòdul d'intercanvis: Permetre que els usuaris puguin comerciar entre ells, o inclús afegir un mercat on els usuaris puguin fer compra/venta dels productes. D'aquesta manera es poden posar millores de recursos que un usuari no pot aconseguir per si sol sense comerciar amb altres, augmentant de manera dràstica la jugabilitat.

- Fer el mòdul d'històrics: Un usuari ha de poder revisar totes les batalles i intercanvis que ha realitzat. Fins i tot es podria arribar a fer un mòdul d'estadístiques que grafiqués els diversos atacs o la fluctuació de recursos del jugador.

- Crear un backend suficientment potent com per permetre crear i gestionar tot el que hi ha i passa dintre del joc.

- Omplir la base de dades (amb el backend del punt anterior) amb totes les dades del joc. Totes les civilitzacions, edificis, nivells...

- Preparar el sistema per quan s'hagi d'escalar, ja que arribarà un moment que el tràfic augmentarà dràsticament i s'ha de preveure en quin moment ho farà per poder reaccionar i escalar el servei a temps, sino es podria arribar a "morir d'èxit".

13 BIBLIOGRAFIA

Statistia. (2015) *Statistics and facts about the Digital Music Industry*. 10 Maig 2015. Recuperat des de http://www.statista.com/topics/1386/digital-music/

Statistia. (2015) *Statistics and facts about the Film Industry*. 10 Maig 2015. Recuperat des de http://www.statista.com/topics/964/film/

Statistia. (2015) *Statistics and facts about the Video Game Industry*. 10 Maig 2015. Recuperat des de http://www.statista.com/topics/868/video-games/

Wikipedia. (2015) *Real-time strategy*. 12 Maig 2015. Recuperat des de http://en.wikipedia.org/wiki/Real-time_strategy

Wikipedia. (2015) *Cytron Masters*. 12 Maig 2015. Recuperat des de http://en.wikipedia.org/wiki/Cytron_Masters

14 ANNEX

14.1 PROCEDURES I TRIGGERS

```
SET GLOBAL binlog_format = 'MIXED';

-- ----------------------------------------------------------------------
-- crear_ciutat: Donat una id d'usuari i un nom crea una ciutat nova.
-- ----------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists crear_ciutat;
delimiter //
CREATE PROCEDURE crear_ciutat (usuari_ INT, nom_ VARCHAR(45))
BEGIN
        DECLARE idciutat INT DEFAULT 0;
        INSERT IGNORE
ciutat(idusuari,`civilitzacio_principal`,`civilitzacio_secundaria`,nom)
                VALUES  (usuari_,1,1,nom_);
        SELECT LAST_INSERT_ID() AS id;
END
//

-- ----------------------------------------------------------------------
-- TRIGGER trigger_inserir_ciutat: Després de crear una ciutat genera les
seves dependències bàsiques
-- ----------------------------------------------------------------------
delimiter ;
DROP TRIGGER if exists trigger_inserir_ciutat;
DELIMITER //
CREATE TRIGGER trigger_inserir_ciutat AFTER INSERT ON ciutat
  FOR EACH ROW BEGIN
    CALL ciutat_definir_recursos (NEW.idciutat);
  END
//
DELIMITER ;
```

```
-- ----------------------------------------------------------------------
-- ciutat_definir_recursos: Donada una ciutat crea la seva relació amb els
recursos per cada recurs si aquesta no existeix.
-- ----------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists ciutat_definir_recursos;
delimiter //
CREATE PROCEDURE ciutat_definir_recursos (idciutat_ INT)
BEGIN
        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idrecurs_ INTEGER DEFAULT 0;
        DEClARE recurs_cursor CURSOR FOR SELECT idrecurs FROM recurs;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN recurs_cursor;
        get_recurs: LOOP
                FETCH recurs_cursor INTO idrecurs_;
                IF v_finished = 1 THEN
                        LEAVE get_recurs;
                END IF;
                -- build email list
                IF idrecurs_ < 5 THEN
                  INSERT IGNORE
ciutat_recurs(idciutat,idrecurs,quantitat,increment_hora)
                    VALUES  (idciutat_,idrecurs_,100,0);
                ELSE
                  INSERT IGNORE
ciutat_recurs(idciutat,idrecurs,quantitat,increment_hora)
                    VALUES  (idciutat_,idrecurs_,0,0);
                END IF;
        END LOOP get_recurs;
        CLOSE recurs_cursor;
END//
DELIMITER ;

-- ----------------------------------------------------------------------
-- TRIGGER trigger_inserir_recurs: Després de crear un recurs genera
dependències de les ciutats existents envers el nou recurs
-- ----------------------------------------------------------------------
delimiter ;
DROP TRIGGER if exists trigger_inserir_recurs;

DELIMITER //
CREATE TRIGGER trigger_inserir_recurs AFTER INSERT ON recurs
  FOR EACH ROW BEGIN
    CALL all_ciutats_definir_recursos();
  END
//
DELIMITER ;
```

```
-- ------------------------------------------------------------------------
-- all_ciutats_definir_recursos: Per cada ciutat crea la seva relació amb
els recursos per cada recurs si aquesta no existeix.
-- Serveix per actualitzar les ciutats antigues quan s'afegeix un nou
recurs.
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists all_ciutats_definir_recursos;
delimiter //
CREATE PROCEDURE all_ciutats_definir_recursos ()
BEGIN

        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idciutat_ INTEGER DEFAULT 0;
        DEClARE ciutat_cursor CURSOR FOR SELECT idciutat FROM ciutat;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN ciutat_cursor;
        get_ciutat: LOOP
                FETCH ciutat_cursor INTO idciutat_;
                IF v_finished = 1 THEN
                        LEAVE get_ciutat;
                END IF;
                CALL ciutat_definir_recursos(idciutat_);
        END LOOP get_ciutat;
        CLOSE ciutat_cursor;
END//
DELIMITER ;

-- ------------------------------------------------------------------------
-- obtenir_pares_civilitzacio: Donada una civilitzacio retorna tots els
seus pares.
-- Serveix per saber per quines civilitzacions ha passat l'usuari en un
arbre.
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists obtenir_pares_civilitzacio;
delimiter //
CREATE PROCEDURE obtenir_pares_civilitzacio (idcivilitzacio_inicial INT)
BEGIN
        SELECT T2.idcivilitzacio
        FROM (
            SELECT
                @r AS _id,
                (SELECT @r := idpare FROM civilitzacio WHERE idcivilitzacio
= _id) AS idpare,
                @l := @l + 1 AS lvl
            FROM
                (SELECT @r := idcivilitzacio_inicial, @l := 0) vars,
                civilitzacio c
            WHERE @r <> 0) T1
        JOIN civilitzacio T2
        ON T1._id = T2.idcivilitzacio
        ORDER BY T1.lvl DESC;
END//
DELIMITER ;
```

```
-- ----------------------------------------------------------------------
-- TRIGGER
trigger_afegir_edifici_recurs/trigger_modificar_edifici_recurs/trigger_elim
inar_edifici_recurs: Després de afegir,canviar o eliminar el que genera un
edifici es recalculen els valors generats per les ciutats
-- ----------------------------------------------------------------------
delimiter ;
DROP TRIGGER if exists trigger_inserir_recurs;
DELIMITER //
CREATE TRIGGER trigger_afegir_edifici_recurs AFTER INSERT ON
edifici_recurs_nivell_genera
        FOR EACH ROW BEGIN
                CALL recalcular_guany_all_ciutats();
        END
//
DELIMITER ;

delimiter ;
DROP TRIGGER if exists trigger_modificar_edifici_recurs;
DELIMITER //
CREATE TRIGGER trigger_modificar_edifici_recurs AFTER UPDATE ON
edifici_recurs_nivell_genera
        FOR EACH ROW BEGIN
                CALL recalcular_guany_all_ciutats();
                END
//
DELIMITER ;

delimiter ;
DROP TRIGGER if exists trigger_modificar_edifici_recurs;
DELIMITER //
CREATE TRIGGER trigger_eliminar_edifici_recurs AFTER DELETE ON
edifici_recurs_nivell_genera
        FOR EACH ROW BEGIN
                CALL recalcular_guany_all_ciutats();
                END
//
DELIMITER ;

-- ----------------------------------------------------------------------
-- TRIGGER trigger_modificar_edifici_nivell: Després de pujar el nivell
d'un edifici d'una ciutat es recalcular els seus guanys
-- ----------------------------------------------------------------------
delimiter ;
DROP TRIGGER if exists trigger_modificar_edifici_nivell;
DELIMITER //
CREATE TRIGGER trigger_modificar_edifici_nivell AFTER UPDATE ON
ciutat_edifici_nivell
        FOR EACH ROW BEGIN
                CALL recalcular_guany_ciutat(NEW.idciutat);
                END
//
DELIMITER ;
```

```
-- ------------------------------------------------------------------------
-- recalcular_guany_all_ciutats: Donada una ciutat recalcula tots els
recursos a partir de tots els edificis (de moment de recursos) el guany per
hora de materials
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists recalcular_guany_all_ciutats;
delimiter //
CREATE PROCEDURE recalcular_guany_all_ciutats ()
BEGIN
        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idciutat_ INTEGER DEFAULT 0;
        DEClARE ciutat_cursor CURSOR FOR SELECT idciutat FROM ciutat;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN ciutat_cursor;
        get_ciutat: LOOP
                FETCH ciutat_cursor INTO idciutat_;
                IF v_finished = 1 THEN
                        LEAVE get_ciutat;
                END IF;
                CALL recalcular_guany_ciutat(idciutat_);
        END LOOP get_ciutat;
        CLOSE ciutat_cursor;

END//
DELIMITER ;

-- ------------------------------------------------------------------------
-- recalcular_guany_ciutat: Donada una ciutat recalcula tots els recursos a
partir de tots els edificis el guany per hora de materials
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists recalcular_guany_ciutat;
delimiter //
CREATE PROCEDURE recalcular_guany_ciutat (idciutat_ INT)
BEGIN
        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idrecurs_ INTEGER DEFAULT 0;
        DEClARE recurs_cursor CURSOR FOR SELECT idrecurs FROM recurs;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN recurs_cursor;
        get_recurs: LOOP
                FETCH recurs_cursor INTO idrecurs_;
                IF v_finished = 1 THEN
                        LEAVE get_recurs;
                END IF;
                CALL recalcular_guany_ciutat_recurs(idciutat_,idrecurs_);
        END LOOP get_recurs;
        CLOSE recurs_cursor;
END//
DELIMITER ;
```

```
-- ------------------------------------------------------------------------
-- recalcular_guany_ciutat_recurs: Donada una ciutat i un recurs recalcula
el guany/hora a partir de tots els edificis el guany per hora de materials
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists recalcular_guany_ciutat_recurs;
delimiter //
CREATE PROCEDURE recalcular_guany_ciutat_recurs (idciutat_ INT,idrecurs_
INT)
BEGIN
        DECLARE quantitat_ INTEGER DEFAULT 0;
        DECLARE quantitat_fabrica INTEGER DEFAULT 0;
        DECLARE quantitat_resta INTEGER DEFAULT 0;
        -- El que generen els edificis de recursos
        SELECT IFNULL(SUM(quantitat),0) INTO quantitat_
                FROM  ciutat_edifici_nivell cen
,edifici_recurs_nivell_genera erng
                WHERE erng.idnivell=cen.nivell AND
cen.idedifici=erng.idedifici AND cen.idciutat=idciutat_ AND
erng.idrecurs=idrecurs_;
        -- El que generen les fàbriques
        SELECT IFNULL(SUM(quantitat),0) INTO quantitat_fabrica
                FROM  ciutat_edifici_nivell cen
,edifici_fabrica_nivell_genera erng
                WHERE erng.idnivell=cen.nivell AND
cen.idedifici=erng.idedifici AND cen.idciutat=idciutat_ AND
erng.idrecurs=idrecurs_;
        SET quantitat_ = quantitat_ + quantitat_fabrica;
        -- El que consumeixen les fàbriques
        SELECT IFNULL(SUM(quantitat),0) INTO quantitat_resta
                FROM  ciutat_edifici_nivell cen
,edifici_fabrica_nivell_gasta_recurs erng
                WHERE erng.idnivell=cen.nivell AND
cen.idedifici=erng.idedifici AND cen.idciutat=idciutat_ AND
erng.idrecurs=idrecurs_;
        SET quantitat_ = quantitat_ - quantitat_resta;
        UPDATE ciutat_recurs SET increment_hora=quantitat_ WHERE
idciutat=idciutat_ AND idrecurs=idrecurs_;
END//
DELIMITER ;
```

```
-- ------------------------------------------------------------------------
-- increment_recursos_tick: Un cop cridat, recorre totes les ciutats i els
hi suma un cop els seus recursos/h
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists increment_recursos_tick;
delimiter //
CREATE PROCEDURE increment_recursos_tick ()
BEGIN
        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idciutat_ INTEGER DEFAULT 0;
        DEClARE ciutat_cursor CURSOR FOR SELECT idciutat FROM ciutat;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN ciutat_cursor;
        get_ciutat: LOOP
                FETCH ciutat_cursor INTO idciutat_;
                IF v_finished = 1 THEN
                        LEAVE get_ciutat;
                END IF;
                CALL increment_recursos_tick_ciutat(idciutat_,1);
        END LOOP get_ciutat;
        CLOSE ciutat_cursor;
END//
DELIMITER ;

-- ------------------------------------------------------------------------
-- increment_recursos_tick_ciutat: Agafa una ciutat i suma ncops els seus
recursos/h
-- ------------------------------------------------------------------------
delimiter ;
DROP PROCEDURE if exists increment_recursos_tick_ciutat;
delimiter //
CREATE PROCEDURE increment_recursos_tick_ciutat (idciutat_ INT,ncops INT)
BEGIN
        DECLARE v_finished INTEGER DEFAULT 0;
        DECLARE idrecurs_ INTEGER DEFAULT 0;
        DEClARE recurs_cursor CURSOR FOR SELECT idrecurs FROM recurs;
        -- NOT FOUND handler
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_finished = 1;
        OPEN recurs_cursor;
        get_recurs: LOOP
                FETCH recurs_cursor INTO idrecurs_;
                IF v_finished = 1 THEN
                        LEAVE get_recurs;
                END IF;
                UPDATE ciutat_recurs SET
quantitat=quantitat+(increment_hora*ncops) WHERE idciutat=idciutat_ AND
idrecurs=idrecurs_;
        END LOOP get_recurs;
        CLOSE recurs_cursor;
        UPDATE `ciutat` SET `ultim_update_recursos`=NOW() WHERE
idciutat=idciutat_;
END//
DELIMITER ;
```

14.2 VIEWS

```
CREATE VIEW vista_civilitzacions AS
SELECT c1.idcivilitzacio,t1.text as nom, t2.text as descripcio,
c2.idcivilitzacio as idpare, t3.text as pare_descripcio,c1.urlimage,
t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, civilitzacio c1
LEFT JOIN civilitzacio c2
ON c1.idpare=c2.idcivilitzacio
LEFT JOIN traduccio t3
ON  c2.nom=t3.idtraduccio
WHERE t1.idtraduccio=c1.nom AND t2.idtraduccio=c1.descripcio AND
t1.ididioma=t2.ididioma AND (t3.ididioma IS NULL OR
t2.ididioma=t3.ididioma) ORDER BY c1.idpare

CREATE VIEW vista_edificis_civilitzacio AS
SELECT e.idedifici,
IF(ef.idedifici IS NOT NULL,'fabrica',
 IF(er.idedifici IS NOT NULL,'recurs',
   IF(ee.idedifici IS NOT NULL,'exercit',
    IF(ed.idedifici IS NOT NULL,'defensa',
     IF(ei.idedifici IS NOT NULL,'investigacio',NULL))))) as tipus,
t1.text as nom, t2.text as descripcio, c.idcivilitzacio, t2.ididioma as
ididioma
FROM traduccio t1, traduccio t2, civilitzacio c,edifici e
LEFT JOIN  edifici_fabrica ef
ON ef.idedifici=e.idedifici
LEFT JOIN  edifici_recurs er
ON er.idedifici=e.idedifici
LEFT JOIN  edifici_exercit ee
ON ee.idedifici=e.idedifici
LEFT JOIN  edifici_defensa ed
ON ed.idedifici=e.idedifici
LEFT JOIN  edifici_investigacio ei
ON ei.idedifici=e.idedifici
WHERE e.idcivilitzacio=c.idcivilitzacio AND t1.idtraduccio=e.nom AND
t2.idtraduccio=e.descripcio AND t1.ididioma=t2.ididioma
ORDER BY c.idcivilitzacio,tipus;

CREATE VIEW vista_recursos AS
SELECT r.idrecurs,t1.text as nom, t2.text as descripcio, t2.ididioma as
ididioma
FROM traduccio t1, traduccio t2, recurs r
WHERE t1.idtraduccio=r.nom AND t2.idtraduccio=r.descripcio AND
t1.ididioma=t2.ididioma
ORDER BY r.idrecurs;
```

```
CREATE VIEW `vista_recursos_civilitzacio` AS
SELECT r.idrecurs,t1.text as nom, t2.text as descripcio, cr.idcivilitzacio,
t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, recurs r,civilitzacio_recurs cr,
civilitzacio c
WHERE t1.idtraduccio=r.nom AND t2.idtraduccio=r.descripcio AND
t1.ididioma=t2.ididioma AND cr.idrecurs=r.idrecurs AND
cr.idcivilitzacio=c.idcivilitzacio
ORDER BY c.idcivilitzacio,r.idrecurs

CREATE VIEW `vista_ciutat_usuari` AS
SELECT c.nom as nom, t1.text as civilitzacio1, t2.text as civilitzacio2,
c.civilitzacio_principal,c.civilitzacio_secundaria,  c.idciutat as
idciutat,c1.urlimage, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, ciutat c, civilitzacio c1,civilitzacio c2
WHERE t1.idtraduccio=c1.nom AND t2.idtraduccio=c2.nom AND
c.civilitzacio_principal=c1.idcivilitzacio AND
c.civilitzacio_secundaria=c2.idcivilitzacio AND t1.ididioma=t2.ididioma
ORDER BY c.idciutat;


CREATE VIEW `vista_recursos_ciutat_simplificat` AS
SELECT r.idrecurs, ciur.quantitat ,t1.text as nom, t2.text as descripcio,
cr.idcivilitzacio, c.idciutat as idciutat,r.urlimage, t2.ididioma as
ididioma
FROM traduccio t1, traduccio t2, recurs r,civilitzacio_recurs cr, ciutat c,
civilitzacio c1, civilitzacio c2, ciutat_recurs ciur
WHERE t1.idtraduccio=r.nom AND t2.idtraduccio=r.descripcio AND
t1.ididioma=t2.ididioma AND cr.idrecurs=r.idrecurs AND
ciur.idrecurs=r.idrecurs AND ciur.idciutat=c.idciutat AND
c1.idcivilitzacio=c.civilitzacio_principal AND
c2.idcivilitzacio=c.civilitzacio_secundaria AND
(cr.idcivilitzacio=c1.idcivilitzacio OR
cr.idcivilitzacio=c2.idcivilitzacio)
GROUP BY r.idrecurs,t2.ididioma,c.idciutat
ORDER BY c.idciutat,r.idrecurs;

CREATE VIEW `vista_recursos_ciutat` AS
SELECT r.idrecurs, ciur.quantitat ,t1.text as nom, t2.text as descripcio,
c.idciutat as idciutat, r.urlimage, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, recurs r, ciutat c, ciutat_recurs ciur
WHERE t1.idtraduccio=r.nom AND t2.idtraduccio=r.descripcio AND
t1.ididioma=t2.ididioma AND
ciur.idrecurs=r.idrecurs AND ciur.idciutat=c.idciutat
GROUP BY r.idrecurs,t2.ididioma,c.idciutat
ORDER BY c.idciutat,r.idrecurs;
```

```
CREATE VIEW vista_edificis_recursos_ciutat AS
SELECT e.idedifici, t1.text as nom, t2.text as descripcio,c.idciutat,c.nom
as ciutatnom,IF(cen.nivell IS NULL,0,cen.nivell) AS nivell,  t.fi as
fi_millora,
r.idrecurs as idrecurs_gasta   , ecmr.quantitat as quantitat_gasta,
ecm.temps_millora,r.urlimage as recurs_gasta_image,t3.text as
recurs_gasta_nom, t4.text as recurs_gasta_descripcio,
r2.idrecurs as idrecurs_genera, r2.urlimage as recurs_genera_image, t5.text
as recurs_genera_nom, t6.text as recurs_genera_descripcio,IF(cen.nivell IS
NULL,0, IF(cen.nivell=0,0,erng.quantitat)) as recurs_genera_quantitat,
e.idcivilitzacio, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, traduccio t3, traduccio t4, traduccio t5,
traduccio t6,edifici_recurs er,
edifici_cost_millorar_recurs ecmr,edifici_cost_millorar ecm,recurs r,
edifici_recurs_nivell_genera erng, recurs r2,
ciutat c
LEFT JOIN edifici e
ON e.idcivilitzacio IS NOT NULL
LEFT JOIN ciutat_edifici_nivell cen
ON cen.idedifici=e.idedifici AND cen.idciutat=c.idciutat
LEFT JOIN temps_pendents t
ON t.idtemps=cen.temps_millora
WHERE t1.idtraduccio=e.nom AND t2.idtraduccio=e.descripcio AND
t1.ididioma=t2.ididioma AND e.idedifici=er.idedifici AND
 ( (cen.nivell IS NULL AND ecmr.idnivell=1) OR cen.nivell=ecmr.idnivell-1
) AND
ecmr.idrecurs=r.idrecurs AND ecmr.idedifici=er.idedifici AND
ecm.idnivell=ecmr.idnivell AND ecm.idedifici=er.idedifici AND
erng.idedifici=er.idedifici AND r2.idrecurs=erng.idrecurs AND  ( (
(cen.nivell IS NULL OR cen.nivell=0) AND erng.idnivell=1) OR (cen.nivell IS
NOT NULL AND cen.nivell!=0 AND erng.idnivell=ecm.idnivell-1) ) AND
t3.idtraduccio=r.nom AND t4.idtraduccio=r.descripcio AND
t3.ididioma=t4.ididioma AND t4.ididioma=t2.ididioma AND
t5.idtraduccio=r2.nom AND t6.idtraduccio=r2.descripcio AND
t5.ididioma=t4.ididioma AND t6.ididioma=t2.ididioma
ORDER BY e.idedifici,c.idciutat,ididioma;

CREATE VIEW `vista_edificis_genera_nivell` AS
SELECT erng.idrecurs,t1.text as nom, t2.text as descripcio,r.urlimage,
erng.quantitat,erng.idedifici,erng.idnivell,t2.ididioma
FROM `edifici_recurs_nivell_genera` erng, traduccio t1,traduccio t2 ,
recurs r
WHERE r.idrecurs=erng.idrecurs AND t1.ididioma=t2.ididioma AND
r.nom=t1.idtraduccio AND r.descripcio=t2.idtraduccio
ORDER BY erng.idedifici,erng.idnivell;
```

```
CREATE VIEW vista_ciutats_edificis_pendents AS
SELECT idciutat,fi,idtemps,
IF(ef.idedifici IS NOT NULL,1,
 IF(er.idedifici IS NOT NULL,0,
   IF(ee.idedifici IS NOT NULL,4,
    IF(ed.idedifici IS NOT NULL,3,
     IF(ei.idedifici IS NOT NULL,2,NULL))))) as tipus
FROM `ciutat_edifici_nivell` cen,temps_pendents,edifici e
LEFT JOIN  edifici_fabrica ef
ON ef.idedifici=e.idedifici
LEFT JOIN  edifici_recurs er
ON er.idedifici=e.idedifici
LEFT JOIN  edifici_exercit ee
ON ee.idedifici=e.idedifici
LEFT JOIN  edifici_defensa ed
ON ed.idedifici=e.idedifici
LEFT JOIN  edifici_investigacio ei
ON ei.idedifici=e.idedifici
WHERE temps_millora=idtemps AND e.idedifici=cen.idedifici

CREATE VIEW vista_civilitzacio_necessita_recurs AS select
cv.idcivilitzacio,`r`.`idrecurs` AS `idrecurs`,cv.quantitat,`t1`.`text` AS
`nom`,`t2`.`text` AS `descripcio`, r.urlimage,`t2`.`ididioma` AS `ididioma`
from `traduccio` `t1`, `traduccio` `t2`, `recurs` `r`
,civilitzacio_necessita_recurs cv
where ((`t1`.`idtraduccio` = `r`.`nom`) and (`t2`.`idtraduccio` =
`r`.`descripcio`) and (`t1`.`ididioma` = `t2`.`ididioma`)) and
cv.idrecurs=r.idrecurs order by `r`.`idrecurs`;
```

```
CREATE VIEW `vista_edificis_fabrica_ciutat` AS
SELECT e.idedifici, t1.text as nom, t2.text as descripcio,c.idciutat,c.nom
as ciutatnom,IF(cen.nivell IS NULL,0,cen.nivell) AS nivell,  t.fi as
fi_millora,
r.idrecurs as idrecurs_gasta   , ecmr.quantitat as quantitat_gasta,
ecm.temps_millora,r.urlimage as recurs_gasta_image,t3.text as
recurs_gasta_nom, t4.text as recurs_gasta_descripcio,
r2.idrecurs as idrecurs_genera, r2.urlimage as recurs_genera_image, t5.text
as recurs_genera_nom, t6.text as recurs_genera_descripcio,IF(cen.nivell IS
NULL,0, IF(cen.nivell=0,0,erng.quantitat)) as recurs_genera_quantitat,
efngr.idrecurs as recurs_consumeix_id,efngr.quantitat as
recurs_consumeix_quantiat, r3.urlimage as recurs_consumeix_image, t7.text
as recurs_consumeix_nom, t8.text as recurs_consumeix_descripcio,
e.idcivilitzacio, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, traduccio t3, traduccio t4, traduccio t5,
traduccio t6, traduccio t7, traduccio t8,edifici_fabrica er,
edifici_cost_millorar_recurs ecmr,edifici_cost_millorar ecm,recurs r,
edifici_fabrica_nivell_genera erng, recurs r2,
edifici_fabrica_nivell_gasta_recurs efngr ,recurs r3,
ciutat c
LEFT JOIN edifici e
ON e.idcivilitzacio IS NOT NULL
LEFT JOIN ciutat_edifici_nivell cen
ON cen.idedifici=e.idedifici AND cen.idciutat=c.idciutat
LEFT JOIN temps_pendents t
ON t.idtemps=cen.temps_millora
WHERE t1.idtraduccio=e.nom AND t2.idtraduccio=e.descripcio AND
t1.ididioma=t2.ididioma AND e.idedifici=er.idedifici AND
 ( (cen.nivell IS NULL AND ecmr.idnivell=1) OR cen.nivell=ecmr.idnivell-1
) AND
ecmr.idrecurs=r.idrecurs AND ecmr.idedifici=er.idedifici AND
ecm.idnivell=ecmr.idnivell AND ecm.idedifici=er.idedifici AND
erng.idedifici=er.idedifici AND r2.idrecurs=erng.idrecurs AND  ( (
(cen.nivell IS NULL OR cen.nivell=0) AND erng.idnivell=1) OR (cen.nivell IS
NOT NULL AND cen.nivell!=0 AND erng.idnivell=ecm.idnivell-1) ) AND
r3.idrecurs=efngr.idrecurs AND
t3.idtraduccio=r.nom AND t4.idtraduccio=r.descripcio AND
t3.ididioma=t4.ididioma AND t4.ididioma=t2.ididioma AND
t5.idtraduccio=r2.nom AND t6.idtraduccio=r2.descripcio AND
t5.ididioma=t4.ididioma AND t6.ididioma=t2.ididioma AND
t7.idtraduccio=r3.nom AND t8.idtraduccio=r3.descripcio AND
t7.ididioma=t4.ididioma AND t8.ididioma=t2.ididioma
ORDER BY e.idedifici,c.idciutat,ididioma;
```

```
CREATE VIEW vista_edificis_generic_ciutat AS
SELECT e.idedifici, t1.text as nom, t2.text as descripcio,e.tipus
,c.idciutat,c.nom as ciutatnom,IF(cen.nivell IS NULL,0,cen.nivell) AS
nivell,  t.fi as fi_millora,
r.idrecurs as idrecurs_gasta   , ecmr.quantitat as quantitat_gasta,
ecm.temps_millora,r.urlimage as recurs_gasta_image,t3.text as
recurs_gasta_nom, t4.text as recurs_gasta_descripcio,
e.idcivilitzacio, t2.ididioma as ididioma
FROM traduccio t1, traduccio t2, traduccio t3, traduccio t4,
edifici_cost_millorar_recurs ecmr,edifici_cost_millorar ecm,recurs r,
ciutat c
LEFT JOIN edifici e
ON e.idcivilitzacio IS NOT NULL
LEFT JOIN ciutat_edifici_nivell cen
ON cen.idedifici=e.idedifici AND cen.idciutat=c.idciutat
LEFT JOIN temps_pendents t
ON t.idtemps=cen.temps_millora
WHERE t1.idtraduccio=e.nom AND t2.idtraduccio=e.descripcio AND
t1.ididioma=t2.ididioma AND
 ( (cen.nivell IS NULL AND ecmr.idnivell=1) OR cen.nivell=ecmr.idnivell-1
) AND
ecmr.idrecurs=r.idrecurs AND ecmr.idedifici=e.idedifici AND
ecm.idnivell=ecmr.idnivell AND ecm.idedifici=e.idedifici AND
t3.idtraduccio=r.nom AND t4.idtraduccio=r.descripcio AND
t3.ididioma=t4.ididioma AND t4.ididioma=t2.ididioma
ORDER BY e.idedifici,c.idciutat,ididioma;
```