Desenvolupament d'un Joc Seriós per HRM - Saxion

Víctor Oliva Vidal

Títol: *Desenvolupament d'un Joc Seriós per HRM - Saxion*

Autor: *Víctor Oliva Vidal*

Data: Juny del 2014

Director: *Teun Lucassen*

Institució del director: *Windesheim University of Applied Sciences - School of Information Sciences*

Ponent: *Lluís Solano Albajes*

Departament del ponent: *Llenguatges i Sistemes Informàtics*

Titulació: *Enginyeria Superior Informàtica*

Centre: *Facultat d'Informàtica de Barcelona (FIB)*

Universitat: *Universitat Politècnica de Catalunya (UPC)*

Institució on s'ha realitzat el PFC: *Windesheim University of Applied Sciences - School of Information Sciences*

Índex

Introducció

En aquest document es treballa el desenvolupament del videojoc seriós que he desenvolupat en la meva estada als Països Baixos.

Un videojoc seriós es pot resumir com a un joc interactiu on la finalitat del qual no és només la de oci, la de divertir, sinó que a més tingui un altre tipus d'efecte desitjat. Potser el primer efecte que ens pot vindre en ment és el de educació o formació, o d'altres com podrien ser jocs de simulació, però n'hi poden haver de molts altres tipus, com per exemple jocs que a través d'un exercici físic milloren les seves capacitats, o jocs que proporcionen una certa informació (notícies, etc.), o jocs que serveixin per oferir publicitat als usuaris.

En aquest cas, l'objectiu del videojoc és d'educació, concretament el d'un tema d'una assignatura d'una universitat.

Motivació

Els de l'assignatura d'administració de recursos humans de la *Saxion University of Applied Sciences*, dels Països Baixos, volen una forma de potenciar l'aprenentatge dels seus alumnes.

Cada vegada més, els plans d'estudis universitaris intenten incloure menys classes magistrals i fer participar de manera més activa als estudiants, doncs es potencia molt més l'aprenentatge de tots els estudiants. Hem passat de plans on la majoria eren classes magistrals, on tan sols participava el professor i els estudiants prenien apunts per després estudiar-s'ho tot per un o dos exàmens, a un pla que afavoreix més una avaluació continuada, amb la implicació dia a dia dels estudiants, fent que l'aprenentatge sigui més continu i uniforme.

Si a més a més d'això li afegim que l'estudiant obté una certa diversió mentre s'estan potenciant els seus coneixements i/o experiències, sembla lògic que els resultats han de ser millors que amb els mètodes habituals.

Objectius

Per tant, l'objectiu principal del projecte és desenvolupar un joc seriós pels estudiants de la Universitat de Saxion.

Aquest joc ha de complementar les hores de teoria dels alumnes d'administració de recursos humans per a que puguin aplicar-hi els coneixements que hagin après. En concret, el temari que s'ha de reforçar són una sèrie de conceptes sobre diferents estratègies que poden agafar les empreses. Una manera de representar l'estratègia d'aquestes empreses es mitjançant 4 quadrants o eixos: *business excellence, cohesion, proactivity i proficiency*, que depenen de

accions que prengui la companyia, la formació que apliquin o la manera de motivar als treballadors. El fet és que si en un moment donat, l'empresa vol canviar d'estratègia, molt sovint no ho pot fer de cop, sinó que cal que passi per una altra estratègia per poder arribar a la de destí. Llavors, donada una situació inicial, el jugador a partir de les seves decisions haurà d'arribar a la situació objectiu.

A més d'aquest objectiu principal, també n'hi ha d'altres que faran que quedi un joc més complet o serveixi per altres coses. Per exemple, el joc també ha d'incorporar un sistema per poder monitoritzar les accions dels jugadors, així com poder exportar en format csv per poder-ne fer estudis científics.

Això implica que el joc hagi de ser on-line, doncs totes aquestes dades han de ser accessibles des d'un servidor.

També hauria d'incorporar algun mecanisme per a poder canviar certes parts del joc sense haver de modificar el codi del programa, com podrien ser per exemple els paràmetres de les funcions matemàtiques, o els paràmetres de diferents funcions per poder balancejar el joc.

Aprofitant que el joc és on-line, s'hauria de poder jugar des d'un navegador, doncs facilitaria la feina per distribuir actualitzacions del projecte, i no faria falta instal·lar-lo als ordinadors de la Universitat.

Per tal d'afegir un punt de competició al joc, també hauria d'incorporar algun sistema de rànking entre els jugadors que hi participen.

Anàlisi i Metodologia

Problema

Abans de saber exactament què hem de fer exactament en aquest videojoc hem de veure i entendre quin és la part "seriosa", quins son els coneixements que el joc ha de contenir per poder-los transmetre als jugadors.

Tota la teoria està basada en un llibre de Kim S. Cameron i Robert E. Quinn, *Diagnosing and Changing Organizational Culture*, on explica com analitzar i canviar la forma de treball de les organitzacions d'una forma eficient.

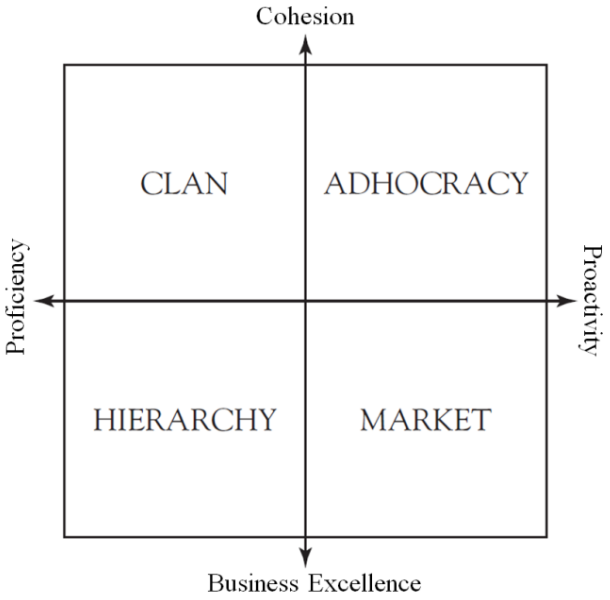


Fig. 1: Clústers i quadrants

Comença explicant que després d'un estudi on s'integraven 39 indicadors d'efectivitat, han arribat a la conclusió que hi ha 2 dimensions més importants que separen a les companyies en 4 clústers principals.

Per una banda hi ha una que diferencia l'efectivitat a base de flexibilitat, discreció i dinamisme de l'efectivitat a base de estabilitat, ordre i control. I d'altra banda diferencia l'efectivitat gràcies a l'organització interna, la integració i la unitat en vers a l'efectivitat per l'orientació externa, la diferenciació i la rivalitat.

Els quatre clústers, que fins a un cert punt els podem anomenar estratègies son els de la Fig.1:

- *Hierarchy* (Jerarquia): Organitzacions que es basen sobre la burocràcia: Normes, especialització, meritocràcia, propietat separada, impersonalitat i resultats predictibles. Com a exemple clar tindríem la cadena de restaurants Fast-Food de McDonald's.
- *Market* (Mercat): Organitzacions que funcionen com un mercat. Treballen sobre mecanismes econòmics del mercat, sobretot a base del intercanvi monetari. És a dir, el major objectiu d'aquestes organitzacions és de fer transaccions (intercanvis, ventes i contractes) amb altres per crear un avantatge competitiu. Els seus interessos estan

sobre la rendibilitat, resultats finals i assegurar-se una bona base de clients. Assumeixen que l'ambient extern no es benigne, sinó hostil.

- *Clan*: Organitzacions que treballen més com una família. Els integrants comparteixen valors i objectius, cohesió, participació activa, i el sentiment de "nosaltres". En comptes de normes i procediments del *Hierarchy* o la concentració sobre el benefici competitiu del *Market*, es basen en el treball en equip, la participació dels empleats, i els premis com a equip.
- *Adhocracy* (Adhocràcia): Aquestes organitzacions assumeixen que les claus per el succés son tindre iniciatives innovatives i pioneres, i es basen en crear nous productes i serveis en contínua preparació pel futur. Son organitzacions que posen l'enfasi en l'emprenedoria i la creativitat, i en crear una visió de futur.

Podríem llavors posar-hi nom als eixos que defineixen aquests clústers:

- *Business Excellence*: És l'eix que defineix els clústers *Hierarchy* i *Market*, i representa el grau d'estabilitat, l'ordre i el control mitjançant estructures elaborades d'empresa i normatives fixades
- *Cohesion*: Defineix els clústers *Clan* i *Adhocracy*. Representa el grau de flexibilitat, discreció i dinamisme, i es caracteritza perquè les organitzacions són molt més familiars. Dit d'alguna altra manera, és l'eix complementari al *Business Excellence*.
- *Proactivity*: Defineix els clústers *Adhocracy* i *Market*, i representa el grau d'orientació externa, diferenciació i rivalitat. Normalment ho tenen les organitzacions amb afany de competir contínuament dins del mercat.
- *Proficiency*: Defineix els clústers *Clan* i *Hierarchy*, i rempresenta el grau d'orientació interna, la integració i la unitat. Bàsicament cuiden de tindre la seva pròpia producció el més eficaç i/o eficientment possible.

Llavors en base a això, el que ha d'ensenyar el joc seriós són les transicions que fa una empresa d'un clúster a un altre. El llibre explica que per les empreses que s'han estudiat, sembla ser que la majoria segueixen un patró, sobretot les més noves. Resumin-t'ho tot, la idea és que donada la representació de la Fig.1, les empreses tan sols es desplacen en l'eix horitzontal o en el vertical, però mai en el diagonal.

Requisits

Requisits funcionals

Per tant, es tracta de fer un joc que donada una situació inicial en una organització i una situació objectiu, el jugador sigui capaç de reconduir l'estratègia per arribar a la situació objectiu. També cal que tingui una certa relació amb el contingut de HRM explicat, doncs la finalitat d'aquest videojoc és potenciar el coneixement dels alumnes.

Aquest joc ha de tindre una certa progressió, doncs l'estona seguida en que jugaran els jugadors és més aviat baixa, i han de ser capaços de guardar l'estat de la seva empresa per continuar més endavant.

També cal mantenir un log de les accions i decisions que prenen els jugadors, perquè més tard se'n voldrà fer un estudi.

Requisits no funcionals

- S'ha d'incorporar un sistema de puntuació per cada jugador i que es puguin comparar en un ranking.
- El joc ha de córrer sobre un navegador, per tal de poder distribuir l'aplicació de forma fàcil.
- Ha d'incloure un panell d'administració per tal que uns administradors puguin gestionar el joc. Per exemple:
    - Donar jugadors d'alta o baixa.
    - Poder importar una llista de jugadors per donar-los d'alta de cop.
    - Poder canviar certs paràmetres del funcionament del joc (Com podrien ser costos, efectes, etc.).
    - Poder afegir/modificar noves situacions inicials i finals.
    - Exportar en format csv el log de les diferents accions i/o decisions que hagin realitzat els jugadors.
- Ha de ser visualment atractiu, on els gràfics siguin 3D i un tema definit.
- Ha d'incloure sons i/o musica de fons per fer-lo més agradable.
- Com que el grup de treball és multilingüe, hauria de tindre un sistema per a poder canviar d'idiomes fàcilment.
- El servidor s'hauria de poder configurar sobre un hosting compartit.

Metodologia

Per a desenvolupar aquest projecte hem fet servir la metodologia *Scrum*, un marc de treball que donada la naturalesa d'aquest projecte (de curta durada i de requisits que poden canviar fàcilment) hem vist que era òptim.

La base de *Scrum* està en crear petits lliuraments cada un cert temps predefinit (normalment entre 1 setmana i un mes, depenent del projecte), cosa que permet al equip desenvolupador respondre als feedbacks i els canvis, per a poder crear exactament el que es necessita. Aquest període de temps s'anomena *Sprint*. El resultat d'una Sprint es un producte *fet*, que podria ser lliurat perfectament.

Es definieixen tres rols primaris:

- *Scrum Master*: Se'n encarrega de fer aplicar *Scrum* i millorar el mètode.
- *Product Owner*: Determina què es el que s'ha d'implementar en la següent *Sprint.*
- *Development Team*: Son els que desenvolupen les parts que s'ha fixat per el *Sprint*.

En cada *Sprint* es comença per l'*Sprint Planning*, on tot l'equip planifica el que es farà durant la *Sprint*. Aquesta etapa respon a dos preguntes, "Que es pot fer en aquesta *Sprint*?" i "Com es farà?" que l'equip les resolen a partir de una llista de les tasques que cal fer (*Product Backlog*) i d'una estimació del temps que comporta cada tasca.

Scrum també diu que cal fer una reunió diaria anomenada *Daily Scrum* o *Standup Meeting* que cal que tingui una durada molt curta (uns 15 minuts). De fet es suggereix fer-la de peu per ajudar a que no s'allargui massa. En aquesta reunió tots els integrants del equip expliquen que van fer ahir, que tenen planificat fer avui i si s'ha trobat qualsevol problema que impedeixi aconseguir l'objectiu del *Sprint*. Això permet al equip tindre una visió global del que està fent tothom, així com poder veure el progrés del equip cap al objectiu del *Sprint*.

Al final del *Sprint* es fan dos altres processos: *Sprint Review* i *Sprint Retrospective*. El *Sprint Review*

Grup de treball

L'equip que ha desenvolupat aquest joc seriós estava format per 8 persones. Molta part del joc la vam fer entre tots, sobretot parts com el disseny i el concepte, on es tracta de fer una pluja d'idees i treballar-ne un resultat final. Per les parts de desenvolupament en si, tot i que Scrum recomana que tots els integrants del grup participin en totes les tasques, sempre hi ha una

certa especialització, o un cert tipus de tasques en que cada un se sent més còmode. En concret, erem:

- Harm-Jan Hazelhorst: Scrum Master. S'especialitzava sobretot amb el disseny de l'arquitectura i la programació del panell d'administració.
- Tom uit de Bulten: Product Owner. Ha participat en el pla de Màrketing i en les interfícies del joc.
- Andok Melkonian: S'especialitzava en el pla de Màrketing i empresa, així com la gestió de les xarxes socials de l'empresa.
- Chiel Broeke: Encarregat dels models en 3D, així com dels sons.
- Christiaan van der Weegh: Encarregat en part de les animacions dels models 3D i interfícies del joc.
- Robin van Ee: Encarregat de crear les escenes i de la IA del joc.
- Youri Kamperman: Sobretot ha treballat sobre el joc en si, tant les interfícies com la lògica darrere.
- Víctor Oliva: Sobretot en el disseny i implementació de l'API, així com la lògica HRM en el client, l'estructura bàsica i algunes interfícies.

Donat que no vaig participar en absolut amb la creació dels models 3D ni del pla d'empresa, aquesta memòria no recull el procés de desenvolupament d'aquestes dos parts.

Anàlisi de Stakeholders

Per poder elaborar el projecte cal tenir molt clar quins son els Stakeholders del projecte i quin rol hi tenen, tant si és positiu com si és negatiu, doncs ens permetrà tindre un millor control de tot el projecte en si. Per a aquest projecte podem identificar els següents *Stakeholders*:

- Guido Bruinsma: És qui "paga" el projecte, el responsable de l'assignatura de HRM de la universitat de Saxion. Aquest és qui explica la teoria de HRM per tal d'aplicar-la al joc i qui influeix més en els requisits del projecte. Se'l informa del progrés del desenvolupament al final de cada *Sprint* (amb l'*Sprint Review*), i es concerten reunions puntuals per aclarir parts dels requisits del joc.
- Estudiants de HRM de Saxion: Són els clients finals del joc seriós producte d'aquest projecte. Tenen un paper molt important no només pel fet de que són els qui faran ús del joc i per tant s'ha de dissenyar-lo per a ells, sinó també seran subjectes en la fase

de proves en les últimes etapes del projecte, donant un feed-back que pot ser molt important.

- Professors de HRM: Són qui faran ús del panell d'administració per aportar-hi contingut al joc i poder ajustar paràmetres per balancejar el joc.
- Grup investigador: Interessats en treure conclusions tant de l'efectivitat del joc seriós com del comportament dels estudiants donades diferents situacions en el joc una vegada es posi en marxa.
- Premsa: Cal informar-los en les últimes fases del projecte, doncs pot ser que doni reputació a l'empresa.
- Spin Awards: Concurs de creativitat en mitjans digital dels Països Baixos, que també donarà reputació tant al joc en si com a l'empresa.

Tecnologies

Client

Donat que havíem de fer el joc en 3D per tal de que fos visualment atractiu, a més que s'havia d'executar en un navegador, vam desenvolupar-lo amb Unity3D, un motor de videojocs 3D multiplataforma, és a dir, que permet exportar els projectes per a poder-los executar tant en Windows, com Linux o Mac OS, per navegador (amb el plug-in de Unity) o fins i tot per mòbil o per consoles de joc.

Unity el que fa és agrupar tota una sèrie de recursos (models, objectes, textures, càmeres, etc.) relacionats entre ells i posats dins d'escenes, i tot relacionat amb scripts (que poden ser en Javascript, C# o Boo) se'n encarrega de tots els processos de renderització (amb Direct3D o OpenGL) per tal de fer córrer el joc.

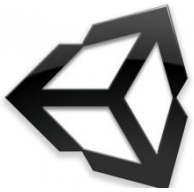


Fig. 2 Logotip Unity

En el nostre cas vam fer servir C# per a Unity, un llenguatge de programació orientat a objectes desenvolupat i estandarditzat per Microsoft, que per la seva semblança al Java en quan a la programació orientada a objectes i el gran nombre de llibreries que es poden trobar creiem que era idoni donat l'abast del projecte.

Servidor

Com que un dels requisits era que el servidor s'havia de poder configurar sobre un *hosting* compartit, vam decidir fer servir AMP (Apache - MySQL - PHP), al ser un sistema que es pot trobar en la majoria d'aquest tipus de hostings.

Apache es un servidor de HTTP *opensource*, és a dir, que està desenvolupat i mantingut per gent arreu del món. És un servidor on s'hi poden configurar moltes coses, a més d'estar basat en mòduls, on s'hi poden afegir o treure segons es necessitin.

Fig. 3 Logotip de Apache

En el nostre cas, hi cal posar el mòdul de PHP per tal de que les pàgines siguin preprocessades en PHP, un llenguatge originàriament dissenyat pels servidors per a poder oferir contingut dinàmic, normalment incrustant les parts dinàmiques en el HTML (en una mena de patró plantilla). PHP ha anat evolucionant moltíssim fins a tal punt que conté el paradigma de programació orientada a

Fig. 4 Logotip de PHP

objectes, i un gran nombre de llibreries i de frameworks externs que el deixen com a una molt bona opció per desenvolupar projectes.

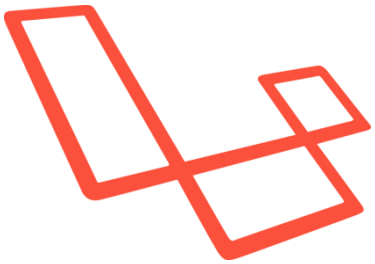


Fig. 5 Logotip de Laravel

Laravel és el framework que hem fet servir per desenvolupar el servidor en PHP. Laravel ajuda en la creació de projectes web en PHP amb tasques comunes per aquests tipus de projectes, com poden ser autentificació, routing, sessions, migracions (molt útils en el nostre cas, donat que per la naturalesa de Scrum l'estructura de la base de dades canvia molt sovint), tests unitaris, etc. i estableix una sèrie de patrons de disseny i/o tècniques que dona una certa organització i robustesa al projecte, com Inversion of Control, MVC, etc. També té molt bona integració amb algunes tecnologies molt útils, com Eloquent o Sentry.

Eloquent és un paquet ORM (Mapatge d'objectes relacional), que gestiona la connexió i la sincronització amb la base de dades, mapejant els objectes del model escrits en PHP a les taules de la base de dades. Permet per tant treballar directament amb el model sense haver-se de preocupar directament a haver de preparar les consultes per obtindre o modificar les dades de la base de dades.

Un altre paquet que hem fet servir per el servidor és Sentry, de Cartalyst. Es tracta d'un paquet d'autorització i autentificació que ens ha permès no només mantenir un nivell bo de seguretat (en general, és millor fer servir llibreries especialitzades en seguretat que no fer-s'ho un sol) sinó també incloure-hi conceptes avançats com per exemple grups d'usuaris (doncs tenim estudiants, investigadors i administradors) amb els seus respectius permisos.

Per mantenir les dependències i que més tard fos fàcil traslladar l'aplicació web a un altre servidor, també hem fet ús de Composer, un programa per a PHP que s'encarrega justament d'això, donada una certa llista de dependències, instal·la les versions que toquen dels paquets. Això fa que quan es vulgui migrar a una altra màquina, el procés d'instal·lació sigui pràcticament automàtic.

Fig. 6 Logotip de MySQL

Per últim, per la base de dades vam decidir fer servir la gran coneguda MySQL, una base de dades relacional que té un gran suport i un gran nombre de possibles configuracions, així com un fort nombre d'usuaris arreu del món.

Eines de comunicació

Després també tenim algunes poques tecnologies que, tot i que no serveixen directament per produir programes, ajuden molt a l'hora amb l'organització i la comunicació del projecte, tant internament com externament.

L'eina més important és el repositori Git, un sistema de control de versions que permet, resumint, que tot l'equip treballi de forma simultània sobre el mateix codi, i dona la possiblitat de revisar el treball de cada desenvolupador i desfer canvis. Si hi ha dos desenvolupadors que treballen sobre el mateix codi exactament, proporciona eines per poder resoldre el "conflicte" i donar una solució.

Per últim, per compartir documents entre nosaltres vam fer servir Google Drive, una plataforma *cloud* de google per compartir fitxers, i algunes vegades en que una reunió presencial amb el client no era possible fèiem servir Skype, un programa de telefonia i comunicació per Internet, per fer videoconferències.

Concepte

El primer joc que a un li pot vindre en ment quan es tracta de fer un joc seriós que ensenyi com administrar una empresa és bàsicament un simulador. És a dir, un videojoc en que el jugador vegi la seva empresa i l'hagi d'organitzar i gestionar per tal de poder complir amb els seus objectius.

Tot i així, nosaltres buscàvem alguna manera per tal de que fos més divertit i visual, doncs ens semblava que un "simple" simulador es podria fer avorrit. Vam decidir llavors emmascarar l'empresa com a una piràmide antiga d'Egipte, on dins hi ha uns monstres (*minions*) que el jugador els gestioni per a complir amb el seu objectiu.

Cada *minion* té 6 característiques. Quatre d'elles indiquen com treballen dins la companyia (seguint els 4 quadrants de la teoria, Business Excellence, Proactivity, Proficiency i Cohesion que el jugador pot canviar-les amb cursos (*training*), que cada un té efectes positius i negatius en els quatre quadrants del *minion*. Les altres dues indiquen el grau de motivació del *minion* i la productivitat resultant en funció de totes les seves característiques i com s'aproximin a les de la piràmide (empresa).

Fig. 7 Art concept de minion

La piràmide està estructurada en 4 plantes, que cada planta representa cada una dels treballs que han de fer els *minions*. La planta baixa és la de producció, on els *minions* recullen plàtans de les mines i les dipositen en uns ascensors per portar-les a la segona planta, on s'empaqueten en caixes per ser analitzades (control de qualitat) i venudes a la tercera planta. La quarta planta té el *Boss*, que monitoritza com va l'empresa i indica el usuari el seu progrés. També té les quatre característiques que defineixen quina és l'estratègia actual de la piràmide, que es tracta d'una mitjana de les característiques dels seus minions (Business Excellence, Proactivity, Proficiency i Cohesion).

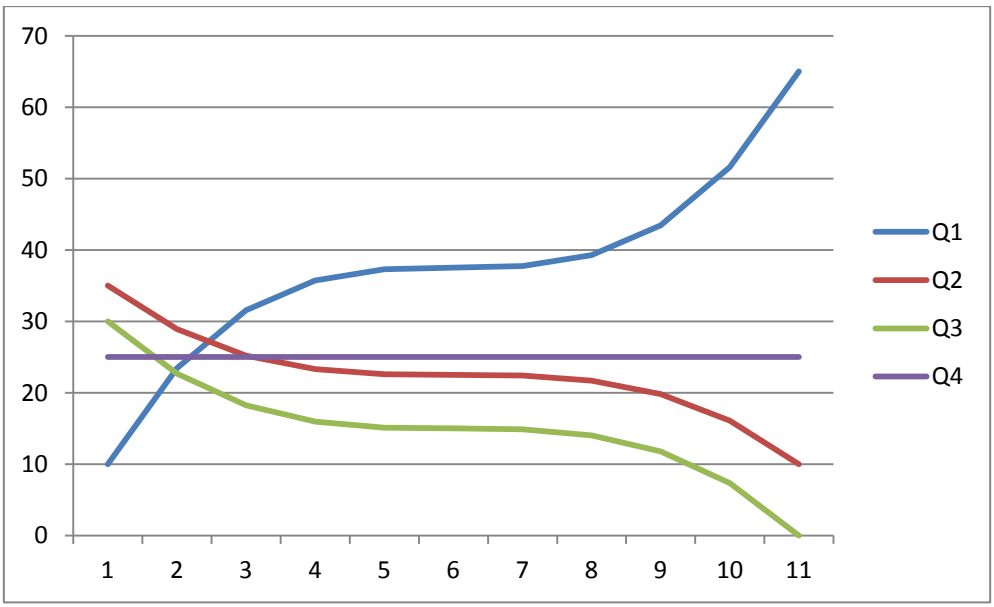
Quan un jugador comenci una partida tindrà una situació inicial (amb uns minions inicials amb les seves característiques) i l'haurà de reconduir cap a una altra estratègia a base tant de fer-la créixer, contractant nous minions o acomiadant d'altres, com re-entrenant els minions amb els cursos o motivacions, o decisions que afectin a tota l'empresa directament. El joc anirà informant al usuari del seu progrés cap al seu objectiu enviant-li correus al joc en nom del minion cap.

Per tal de que tingui l'efecte que s'explica en l'apartat "Problema" d'aquesta memòria, els influenciadors (cursos, motivacions o decisions sobre l'empresa) modifiquen les

característiques dels minions traient una certa quantitat d'un i passant aquests "punts" trets cap a les altres 2 característiques adjacents (veure Fig. 1, Clústers). Aquest efecte no té perquè ser inmediat, sinó que pot durar un cert temps i se li pot assignar una corba d'efecte:

- Lineal: $f(t) \approx \mathrm{t}$
- Quadràtica: $f(t) \approx \mathrm{t}^2$
- Exponencial: $f(t) \approx \mathrm{e}^{\mathrm{t}}$
- Logarítmica: $f(t) \approx \ln(\mathrm{t})$
- Amb forma de S: $f(t) \approx \mathrm{t}^3 + \mathrm{t}^2 + \mathrm{t}$

Òbviament aquestes funcions s'ajusten amb uns certs paràmetres per tal que tingui l'efecte desitjat amb el temps desitjat. Les funcions "amb forma de S" són polinomis de tercer grau però una mica restringits: La seva derivada és una paràbola que el mínim és zero i el seu centre està just al mig del temps total. A continuació un gràfic d'exemple que mostra un influenciador amb una corba d'efecte amb forma de S amb començament ràpid (per tant, el coeficient de segon grau de la derivada és positiu). Aquest influenciador dona 55 punts al Q1 al cost de treure'n 25 al Q2 i 30 al Q3.



Gràfic 1 Efecte del influenciador amb la corba S-Shape Fast Start.

Aquestes característiques no només serveixen per complir directament l'objectiu, sinó que també tenen un efecte sobre la productivitat de cada minion. Quan les característiques del minion són més semblants a les de la seva piràmide, millor serà la seva productivitat, que junt

amb el grau de felicitat (*happiness*) té un efecte sobre la productivitat física del minion (la velocitat en la que camina, el nombre de platans que processa, etc.)

Pel correcte creixement de l'empresa i per poder arribar al objectiu, també es important el sistema de reclutament (el procés de buscar minions per contractar-los). En aquest cas el jugador quan decideixi que cal contractar més minions, seleccionarà una sèrie de característiques que vol que tinguin els minions reclutats (que són configurables, p.e., "habilitats verbals", "Innovador", etc), i representa que el "departament de recursos humans" ja se'n encarregarà de buscar-li i contractar-li els minions amb les característiques seleccionades.

Una altra part que afecta directament al creixement de l'empresa és una que no venia en els requisits, però era necessària per donar-li sentit als diferents rols que poden agafar els minions, que es tracta d'una cadena de producció, es a dir, que lo que produeixen els minions miners no va directament al "capital" de la piràmide, sinó que s'envia a la següent planta perquè els empaquetin, i que els envien a la següent planta perquè facin un control de qualitat i venda als clients.

Hem decidit crear dos rànquings pel joc, per donar competitivitat entre els jugadors. D'una banda hi ha el rànquing per nivell i per producció. Mentre el rànquing per nivell indica quan d'aprop està cada jugador a completar el seu objectiu principal, el rànquing per producció a partir del nombre total de plàtans produïts i el temps total jugat treu una puntuació per poder equiparar les piràmides que més produeixen (anul·lant fins un cert punt l'efecte del temps, que és obvi que quan més temps un porta jugant, més plàtans haurà produit).

Per últim, tal i com manenc els requisits, també hi haurà un panell d'administració que permetrà configurar i ampliar el contingut del joc. En concret:

- Administrar influenciadors, afegint-ne de nous definint el seu efecte amb la corva que toca, etc. o modificant-ne o eliminant-ne ja existents
- Configurar correus automàtics que s'enviïn donada una certa condició en el joc (p.e. quan s'hagin contractat minions, quan s'està apropant al objectiu per N punts, etc.)
- Definir noves característiques pel reclutament.
- Crear nous nivells, definint una situació inicial i un (o varis) objectius.
- Canviar molts paràmetres del joc (p.e. ajustos sobre l'efecte dels influenciadors, productivitat dels minions, etc.).
- Administrar comptes d'administrador/jugador.

- Exportar les accions dels usuaris (Logs) en CSV.

Història

Per tal d'entrar més en el concepte i crear un tema (*theme*), vam escriure una historia:

Durant molt temps pensava que era l'únic de la seva espècie. Un ser baixet, sense pel i vermell, que semblava ser el contrari que l'estereotip humà. Havia d'estar constantment amagant-se de la gent, que segurament l'empresonarien com a un animal de companyia, o disseccionar-lo per estudiar-lo en un laboratori científic. Mentre sobrevivia com podia evitant els humans, llegia molt, per poder-se educar i aprendre moltíssimes coses sobre el món i la vida. Específicament, els llibres que més li interessaven eren els d'empresa, que explicaven el concepte sobre l'intercanvi i com fer-ne diners. Es sentia frustrat pel fet de que per la seva naturalesa no podia muntar una companyia.

Fins que un dia es va trobar dins d'una piràmide a Egipte, i va descobrir que en l'immens nombre de túnels a la part inferior de la piràmide, les parets estaven repletes de plàtans tot i que és xocant el fet de que els plàtans puguin créixer sota la terra en absència de llum. Tastant-ne una va veure que eren boníssims, en va guardar uns quants a la seva motxilla, i va seguir explorant.

Més endavant va trobar-se amb una piscina estranya amb forma de semicercle que tenia un líquid groc i viscós. Mentre investigava per a què servia exactament aquella piscina, i no només això, sinó també que era aquell líquid estrany, se li van caure de la seva motxilla uns quants plàtans dins. El que va passar a continuació era una cosa que a ningú se li hagués acudit mai: una criatura molt semblant a ell mateix va emergir de la piscina. Intentant interactuar amb ell va descobrir que la criatura no era gens intel·ligent: tan sols era capaç de dur a terme tasques simples que se li manaven.

No va perdre més el temps: juntament amb el servent va recollir tants plàtans com va poder i els va tirar a la piscina. Un gran nombre de minions van sortir de la piscina, i tots ells estaven encantantats de seguir ordres del seu nou cap. Totes aquestes oportunitats li van fer despertar la petita flama malèfica que cremava dins seu per fer-se amb tot el món. Feia que els seus servents recollissin tots els plàtans que poguessin, que les empaquetessin i les venguessin als clients.

L'organització, al créixer considerablement, es va trobar que no podia administrar tots els servents per ell sol, i va descobrir un ingredient secret que al posar-lo junt amb els plàtans a la piscina va fer sortir un servent molt més inteligent que els altres, capaç de reconduir l'organització de la piràmide per ell sol. Li donà tots els privilegis per tal de que tingués un control directe sobre els servents i li va permetre a ell lliurar-se d'una gran part del temps per poder-lo dedicar als seus plans per dominar el món, i venjar-se del món que en un principi el perseguia...

Arquitectura

MVC

El patró de disseny Model Vista Controlador defineix una manera d'organitzar el codi d'una aplicació amb interfícies en tres parts inter-comunicades:

- Model: Conté les dades de l'aplicació, i les normes, la lògica i les funcions bàsiques de l'aplicació.
- Vista: S'encarreguen de mostrar informació al usuari, així com oferir una interfície.
- Controlador: Transforma els inputs del usuari per tal que hi hagi un efecte en el model o en la vista. Dit d'alguna manera, és el que es comunica amb la vista i el model per lligar-ho tot. Rep tant avisos de la vista (p.e. que l'usuari ha pitjat un botó que hauria d'activar una funció del model, o fer un *scroll* a la mateixa vista) com del model (p.e. que el model ha canviat com a resultat d'una altra operació) i actualitza tot per mantindre tot sincronitzat.

Això permet tindre molt desacoblament i tindre el codi estructurat d'una forma molt modular, fent que es pugui canviar fàcilment una de les parts, o poder tindre varies vistes i controladors per un mateix model.

Domain Model

Per explicar el domain model, primer exclouré un cas d'us per simplificar el diagrama i després l'explicaré per separat. El cas d'ús que excloc es el de tindre emmagatzemat un log de les accions dels jugadors.

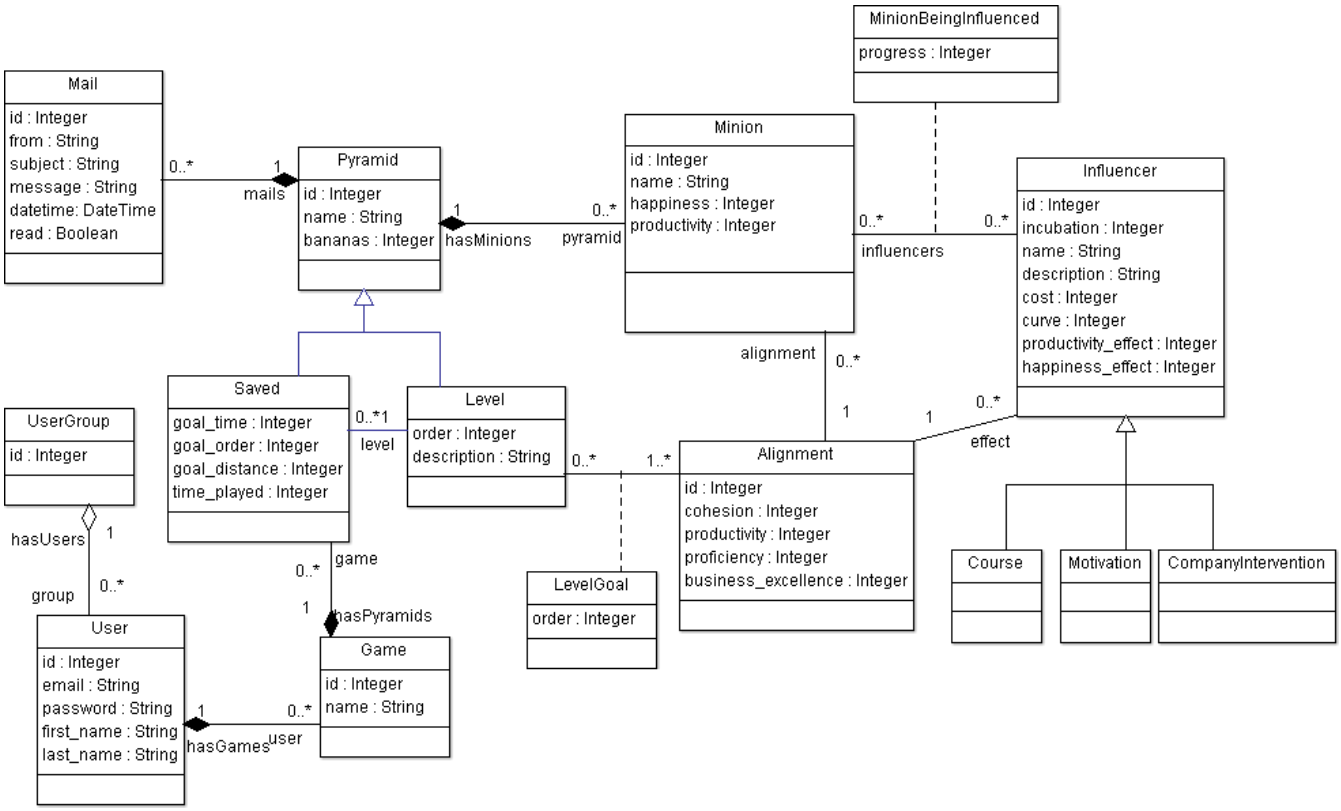


Fig. 8 Domain Model sense ActionLog

Comencem per la classe d'usuari (*user*). Els usuaris formen part d'un grup d'usuaris (*user_group,* per tindre els permisos que toquen), i tenen "jocs" (*game*), la classe que representa tot l'estat del joc en un moment donat.

D'altra banda tenim la classe piràmide (*pyramid*) que n'hem definit de dos tipus: la que representa un nivell (*level*, bàsicament el jugador es troba amb una situació inicial i l'ha de portar als objectius, que es representa amb una associació ordenada amb *alignments*) i la piràmide de cada un dels jugadors (*saved*), que manté una referència al nivell del que ha partit.

Després, tota piràmide pot tindre correus (*mail*) i minions, que les característiques d'aquests poden estar sent influïts per influenciadors (que es categoritzen en cursos, motivacions i intervencions a nivell d'empresa).

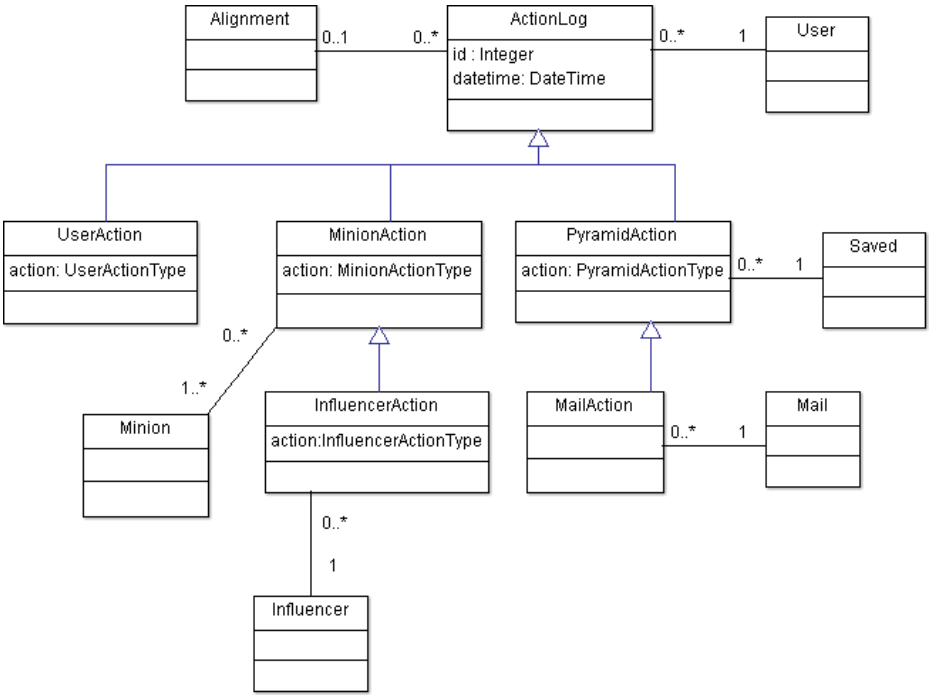


Fig. 9 Diagrama de classe de Action Logs

Per al cas d'us de emmagatzemar les accions que ha fet l'usuari ho separem en moltes parts. Per començar, totes les accions les ha fet algun usuari, i aquest pot tindre la seva piràmide en un estat, *alignment*, que també el guardem.

Emmagatzemem les següents accions:

1. Accions d'Usuari
    a. Conectarse (login).
    b. Desconectarse (logout).
2. Accions dels minions. Guardem el(s) minion(s).
    a. Minion(s) contractat(s).
    b. Minion(s) expulsat(s).
    c. Accions d'influencia. Guardem l'influenciador
        i. Influenciador començat
        ii. Influenciador acabat
3. Accions de la piràmide. Guardem la piràmide
    a. Piràmide creada
    b. Objectiu complert
    c. Accions de correu. Guardem el correu
        i. Correu llegit

Això ens permet tindre totes les accions del usuari en el seu moment, i mantenint quin era l'estat en el moment que la va fer.

Base de dades

Ara per portar aquest model a la base de dades (que hem decidit que seria SQL) cal normalitzar-lo, i hem fet alguna modificació per tal de que fos més fàcil i eficient fer consultes o modificacions.

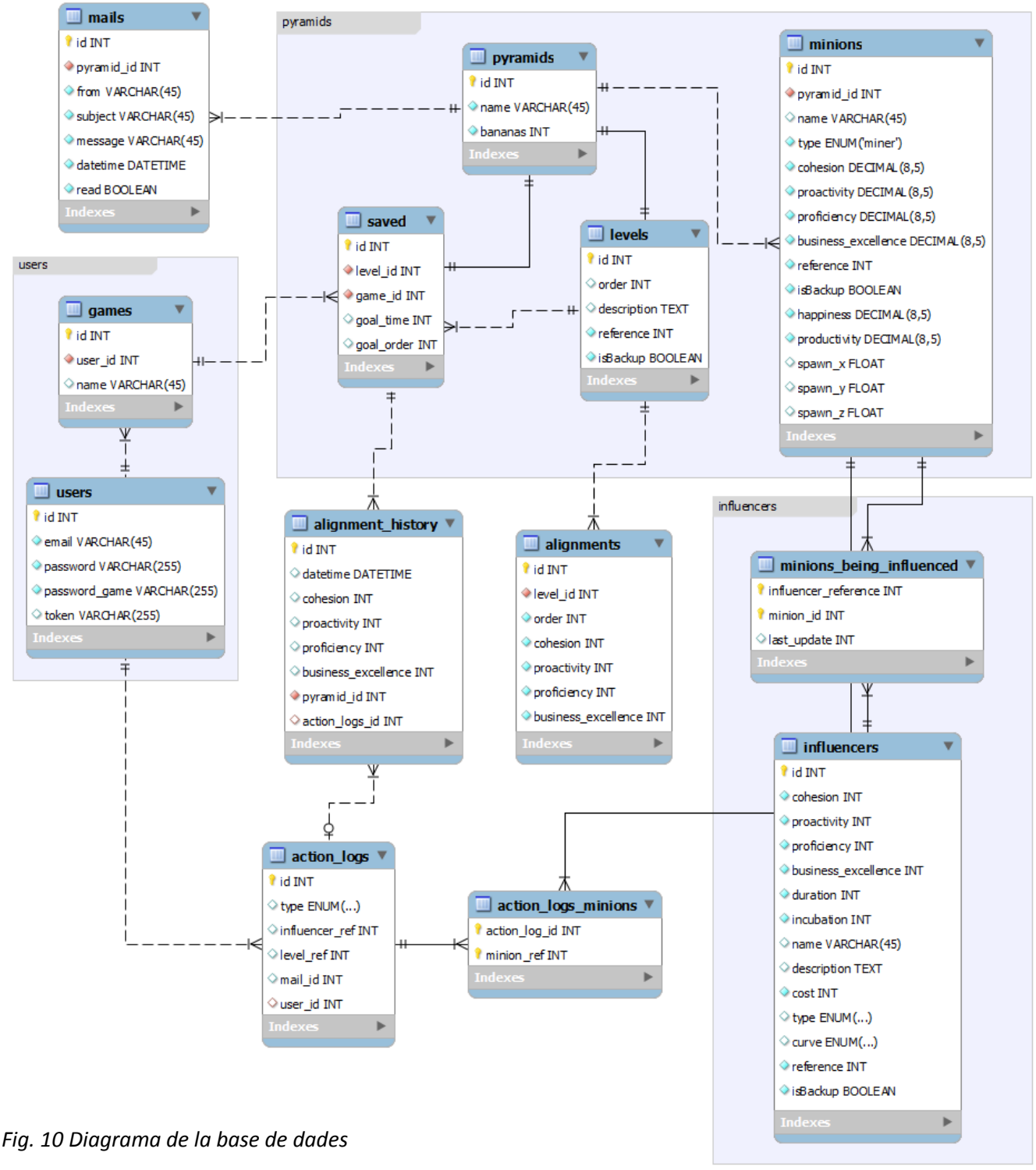


Fig. 10 Diagrama de la base de dades

En concret, les generalitzacions dels influenciadors les hem deixat com a una propietat "*type*", i les dels *Action logs* també les hem unificat tot en una sola taula, on cada propietat és opcional depenent del seu "tipus d'acció". També hem eliminat algunes associacions amb la classe *Alignments*, per evitar-nos joins quan s'haguessin de buscar minions o influenciadors. Es podria dir que vam canviar una mica la seva funció, passant de representar un alignment a un objectiu d'un nivell.

Per últim, hem normalitzat la generalització de *Pyramid* en *Saved* i *Levels* en una herència de taules, on les subclasses agafen la mateixa id que la superclasse i les associacions molts a molts hem creat una taula auxiliar per poder-les fer.

En aquest diagrama (Fig. 10) hem amagat algunes claus foranes dels action_logs per deixar-lo més net.

Comunicació en xarxa

Per a poder comunicar el servidor, que és el que emmagatzema l'estat de cada un dels jugadors, amb el client, que és el joc en si, vam haver de decidir varies coses. Per començar, entre TCP o UDP vam decidir fer servir TCP, doncs el volum de dades que s'han de passar (bàsicament, algunes vegades que l'usuari fa una acció) son relativament petites, i volem tindre la seguretat de que els paquets arriben correctament.

Una vegada decidit que faríem servir TCP, vam decidir fer servir el protocol HTTP, doncs el servidor del protocol en si ja n'hi ha un que és ben conegut i només caldria implementar l'aplicació en si. A més, es podria integrar amb el panell d'administració fàcilment, doncs aquest és programat en una pàgina web.

Hem volgut implementar una API REST per les accions que facin els jugadors al joc, fent servir els verbs propis d'HTTP, com GET, POST, PUT, DELETE.

Seguretat

Per a algunes entrades de la API és clar que fa falta alguna forma d'autorització (per exemple, no volem que un jugador faci una acció per a un altre jugador sense el seu consentiment), així que vam decidir fer servir HMAC com a autentificació. Explico primer el mètode i després el justificaré.

Per autentificar-se, el client envia al servidor la contrassenya del usuari en un *Hash*, encriptat amb una clau compartida (i pública) del servidor. Dit d'una altra manera, si HMAC(dades, contrasenya) representa el *hash* de les "dades" fent servir la contrasenya "contrasenya", i HMAC_TOKEN es la clau compartida del servidor, el client li envia:

```
HMAC(contrassenya_del_usuari, HMAC_TOKEN)
```

Llavors el servidor li retorna una altre clau (diguem-li user_token) que és un *string* aleatòri que servirà per calcular la resta de HMACs de la sessió.

Llavors per a cada petició al servidor que faci falta autorització, es fa servir el *Header Content-Signature* de HTTP, que conté una "firma" de les dades que s'han enviat. En concret, es tracta de:

```
Content-Signature: HMAC(path + ":" + dades + ":" + user_token, user_password)
```

On path és la URL de la petició (per exemple, "api/influencers/3") i dades son les dades que es passen dins del cos de la petició (en el cas que no hi hagin dades, aquest es una *string* buida).

A més d'aquest sistema d'HMAC, amb el Framework que fem servir (Laravel) per poder obtindre els permisos dels usuaris es fan servir sessions de PHP amb cookies.

Reconeixem que no és el millor sistema de seguretat, però creiem que és un bon equilibri entre complexitat/cost i seguretat. Com a efectes positius tenim:

- Al fer servir una espècie de signatura sobre les dades que s'envien resulta impossible per a un atacant modificar la petició si no té la contrasenya de l'usuari. I pels usuaris "tramposos" que modifiquen les peticions per treure'n benefici ja no ho tenen d'una forma immediata (cal que descobreixin com està fet el *hash* i que hagin de re-calcular la signatura).
- Tot i que el primer paquet d'autorització sempre és el mateix, un atacant no en treu res d'enviar-lo, doncs com que el *user_token* és diferent per a cada sessió i és generat pel servidor, necessita també la contrasenya del usuari per poder-se identificar com ell.

I com a negatius o que es podrien millorar amb més temps/cost:

- El punt que més falla és que el servidor necessita tindre la contrasenya del usuari en un primer moment per tal de poder comprovar la signatura HMAC, i la forma en que aquí ho fem (quan l'usuari es registra), es passant-ho per un canal insegur. La manera de mitigar aquest punt seria fer servir criptografia asimètrica, per crear un canal segur i poder transmetre-la.
- Tot i que els *user_tokens* són diferents entre sessions, dins la mateixa sessió segueixen sent vàlids. Això vol dir que un atacant pot repetir peticions que hagin enviat els usuaris seràn valides. Es podria resoldre afegint un comptador extra dins de la signatura

De totes formes, aquest últim punt dèbil queda resolt gràcies al Framework que fem servir (Laravel), doncs per a mantenir els permisos dels usuaris fa servir *cookies* de sessions PHP per autentificació.

API

Per a fer l'especificació de la API, definiré una sèrie de "parts" per tal de que quedi més petit. Si quan faig servir una d'aquestes parts posant-hi un asterisc (*) vol dir que es tracta d'un "resum" i que no surten les dades més enllà del primer nivell (cada nivell es representa amb una tabulació)

Totes les accions poden respondre amb un codi 5XX si hi ha algun problema amb el servidor.

També els codis d'error 401 (Unauthorized) o 403 (Forbidden) s'envien amb problemes d'autentificació, el 400 (Bad Request) si falten dades/parametres o el 404 si algun recurs no existeix.

```
$status:
{
      "cohesion":int,
      "proactivity":int,
      "proficiency":int,
      "business_excellence":int
}
```

Influenciadors

Necesitem entrades per obtindre els influenciadors que s'hagin introduït en el panell d'administració, així com entrades per aplicar/treure influenciadors a minions.

```
$influencer:
{
      "id":int,
      "cohesion":int,
      "proactivity":int,
      "proficiency":int,
      "business_excellence":int,
      "productivity":int,
      "happiness":int,
      "duration":int,
      "incubation":int
      "name":string
      "description":string
      "cost":int
      "type": 'course' | 'motivation' | 'company',
      "curve":  'linear'  |  'quadratic'  |  'negquadratic'  |
'exponential' | 'logarithmic' | 'sshapess' | 'sshapefs'
}
$influence_in
{
      "minion_id":int,
      "last_update":int
}
```

GET /influencers

Descripció: Retorna una llista amb tots els influenciadors

Codis de retorn: 200 OK

Contingut de retorn: `[$influencer, ...]`

GET /influencers/{id}

Descripció: Retorna l'influenciador amb id {id}.

Codis de retorn: 200 OK

Contingut de retorn: `$influencer`

GET /influencers/types/{type}/influencers

Descripció: Retorna tots els influenciadors del tipus {type}

Codis de retorn: 200 OK

Contingut de retorn: `[$influencer, ...]`

POST /influencers/{id}/minions

Descripció: Aplica influenciadors a minions.

Dades d'entrada: `{"influences": [$influence_in, ...], "status": $status}`

Requereix autentificació.

Codis de retorn: 201 Created

PATCH /influencers/{id}/minions

Descripció: Acaba influenciadors als minions

Dades d'entrada: `{"influences": [$influence_in, ...], "status": $status}`

Requereix autentificació.

Codis de retorn: 204 No Content

Nivells

Pels nivells també necesitem operacions de consulta (per obtindre els nivells que s'hagin configurat en el panell d'administració) i per modificar.

```
$level:
{
```

```
      "id":int,
      "description":string,
      "pyramid":$pyramid,
      "goals":[
            $goal,
            ...
      ]
}

$goal:
{
      "order":int,
      "cohesion":int,
      "proactivity":int,
      "proficiency":int,
      "business_excellence":int,
}
```

GET /levels

Descripció: Retorna tots els nivells

Codis de retorn: 200 OK

Contingut de retorn: `[$level, ...]`. Cada `$level` no conté els objectius (goals), i la piràmide "pyramid" és `$pyramid*` (no va més enllà del primer nivell).

GET /levels?order={order}

Descripció: Retorna el nivell que el seu número d'ordre és el {order}

Codis de retorn: 200 OK

Contingut de retorn: `$level`

GET /levels/{id}

Descripció: Retorna el nivell que el seu id es el {id}

Codis de retorn: 200 OK

Contingut de retorn: `$level`

POST /levels/{id}/goal/{order}/complete

Descripció: Emmagatzema que el objectiu amb número d'ordre {ordre} s'ha completat

Dades d'entrada: `{"pyramid":int, "status":$status}`

Requereix autentificació.

Codis de retorn: 204 No Content

Piràmides

```
$pyramid:
{
      "id":int,
      "name":string,
      "bananas":int,
      "goal_time":int,
      "goal_order":int,
      "minions":[
            $minion,
            ...
      ],
      "mails":[
            $mail,
            ...
      ],
}

$minion:
{
      "id":int,
      "name":string,
      "type":"miner"|"research"|"marketing"|"packaging"
      "cohesion":float,
      "proactivity":float,
      "proficiency":float,
      "business_excellence":float,
      "happiness":float,
      "productivity":float,
      "spawn_x":float | null,
      "spawn_y":float | null,
      "spawn_z":float | null,
      "influencers":[
            $influence,
            ...
      ]
}

$influence:
{
      "influencer_id":int,
      "last_update":int
}

$mail:
{
      "id":int,
      "from":string,
      "subject":string,
      "message":string,
      "datetime":"YYYY-MM-DD HH:MM:SS",
      "read":bool
}
```

GET /pyramids/{id}

Descripció: Retorna la piramide amb id {id}

Requereix autentificació.

Codis de retorn: 200 OK

Contingut de retorn: `$pyramid`

DELETE /pramid/{id}

Descripció: Elimina la piramide amb id {id}

Requereix autentificació.

Codis de retorn: 204 No Content

POST /pyramids/{id}

Descripció: Guarda la piramide amb la id indicada per recuperar-la més tard

Dades d'entrada: `$pyramid`

Requereix autentificació.

Codis de retorn: 204 No Content

POST /pyramids/{id}/minions

Descripció: La piramide amb id {id} contracta un nou minion

Dades d'entrada: `{"minion":$minion, "status":$status}`

Requereix autentificació.

Codis de retorn: 201 Created

Contingut de retorn: `{"minion":"/pyramids/{id_p}/minions/{id_m}"}`

DELETE /pyramids/{id_p}/minions/{id_m}

Descripció: Despideix el minion {id_m} de la piràmide {id_p}

Requereix autentificació.

Codis de retorn: 204 No Content

PUT /pyramids/{id_p}/mail/{id_m}

Descripció: Guarda el correu {id_m} a la piràmide {id_p}. {id_m} és "NEW" si el correu és nou.

Dades d'entrada: `{"mail":$mail, "status":$status}`

Requereix autentificació.

Codis de retorn: 201 Created, si el correu s'ha creat ({id_m} es "NEW"). 204 No Content si el correu ja exisitia.

Contingut de retorn: `{"mail":"/pyramids/{id_p}/mail/{id_m}"}, si 201`

DELETE /pyramids/{id_p}/mail/{id_m}

Descripció: Elimina el correu {id_m} de la piràmide {id_p}

Requereix autentificació.

Codis de retorn: 204 No Content

Partides

```
$game:
{
      "name":string,
      "pyramids":[
            {
                  "pyramid":$pyramid*
                  "level":$level*
            },
            ...
      ]
}

$game_in:
{
      "name":string,
      "pyramids":[
            $pyramid,
            ...
      ]
}
```

GET /games/{id}

Descripció: Retorna el joc amb id {id}

Requereix autentificació.

Codis de retorn: 200 OK

Contingut de retorn: `$game`

POST /games/{id}

Descripció: Guarda tota la partida amb id {id}. Els elements que no es passin s'asumiràn ser eliminats.

Dades d'entrada: `$game_in`

Requereix autentificació.

Codis de retorn:

- 204 No Content

- 201 Created, si hi ha hagut algun objecte nou creat (amb id NEW)
- 207 Multi Status, si hi ha varis codis de retorn (per exemple, poden haver-hi varis objectes creats, o alguns amb algun error (400, 403 o 404)). Contingut de retorn: `[{"code":int, "phrase":string, "content":object}, ...]`

POST /games/{id}/pyramids

Descripció: Crea una nova piràmide

Dades d'entrada: `{"level_id":int}`

Requereix autentificació.

Codis de retorn: 201 Created

Contingut de retorn: `{"pyramid":"/pyramids/{id}"}`

Altres entrades

POST /auth

Descripció: Entrada per fer log-in

Dades d'entrada: `{"user":string, "key":string}`

Codis de retorn: 200 OK si es correcte, 401 Unauthorized si la combinació és incorrecte.

Contingut de retorn: `{"token":"/auth/{token}"}`

DELETE /auth/{token}

Descripció: Entrada per fer log-out

Requereix autentificació.

Codis de retorn: 204 No Content

POST /users

Descripció: Registrar un usuari

Dades d'entrada:

```
{
      "email":string,
      "password":string,
      "first_name":string,
      "last_name":string
}
```

Requereix autentificació.

Codis de retorn: 201 Created

GET /recruitment/characteristics

Descripció: Retorna una llista amb les característiques pels reclutaments

Codis de retorn: 200 OK

Contingut de retorn:

```
[{
      "name":string,
      "cohesion":int,
      "proactivity":int,
      "proficiency":int,
      "business_excellence":int
}, ...]
```

GET /mails

Descripció: Retorna la llista de la configuració de correus.

Codis de retorn: 200 OK

Contingut de retorn:

```
[{
      "id":int,
      "sender":string,
      "subject":string,
      "message":string,
      "trigger":string,
      "trigger_comparisson":string,
      "trigger_value":int
}, ...]
```

GET /rankings?pyramid={id}

Descripció: Retorna una llista amb les 10 millors piràmides i les posicions [-2,+2] de la piràmide amb id {id}

Codis de retorn: 200 OK

Contingut de retorn:

```
{
      "bananas":[{
            "rank":int,
            "name":string,
            "bananas_produced":int,
```

```
            "time_played":int
      }, ...],
      "alignment"[{
            "rank":int,
            "name":string,
            "goal_distance":int
      }, ...]
}
```

Client

Com s'ha dit en l'apartat de tecnologies, el client s'ha desenvolupat íntegrament amb el motor Unity3D. Com que jo no he tocat res dels dissenys dels escenaris ni dels models 3d o els sons, explicaré la part de programació que més he treballat. En concret, l'estructura bàsica, la lògica d'empresa (la part teòrica de HRM), la IA dels minions i les interfícies.

Si fem un cop d'ull a l'estructura de carpetes dels scripts en c# del client en trobem bastantes. Els més importants, per ordre alfabètic, son:

- BusinessLogic: Conté els scripts que controlen la lògica de la piràmide (com a empresa o organització) en si (com els influenciadors, el comportament dels minions, el reclutament, etc.)
- DataCarriers: Conté classes que en general no tenen cap altra funcionalitat que emmagatzemar informació per fer-la servir per paràmetres.
- Enums: Conté les classes de tipus enumeració.
- Events: Conté els administradors d'events, on s'hi poden registrar *listeners* i disparar-los.
- GameUtils: Trobem diverses subcarpetes amb scripts d'utilitat que es fan servir a la resta del programa. Els més notables possiblement és el de internacionalització, i les façanes de comunicació.
- Global: Hi trobem el punt d'entrada del joc (i.e. el *main*), i totes les inicialitzacions que es fan al arrancar l'aplicació, com el IoC Container (explicat a continuació)
- GUIs: Conté totes les interfícies gràfiques (cada una de les finestres del joc).
- Models: En aquesta carpeta es troba tot el *Domain Model* del joc.
- Repositories: Té els scripts que implementen el patró repositori.
- UnityHTTP: Una llibreria externa que fem servir per enviar peticions HTTP.

Repository Pattern

Per donar un bon desacoblament entre el Domain model i la capa de negoci, hem decidit implementar el Repository Pattern (patró repositori). Aquest bàsicament el que fa es defineix una interfície que representa un repositori, i que té operacions bàsiques sobre l'element, com per crear-la, modificar-la, llegir-la i eliminar-la, i després tenim varies implementacions (realitzacions) sobre aquesta interfície, cada una adaptada sobre la tecnologia que es vol fer servir (ja sigui la memòria en si, el sistema de fitxers, fent servir una base de dades, o una connexió en xarxa).

Si en aquest repositori se li afegeix un IoC Container (Inversion of Control Container), que el que fa es configurar en temps d'execució quina de les implementacions es fa servir, ja tenim un sistema molt ben desacoblat, doncs si es vol canviar d'una implementació a una altre, només cal modificar el contenidor perquè agafi una altra implementació (i en aquest projecte no es fa servir, però a més permetria canviar-ho mentre l'aplicació s'està executant).

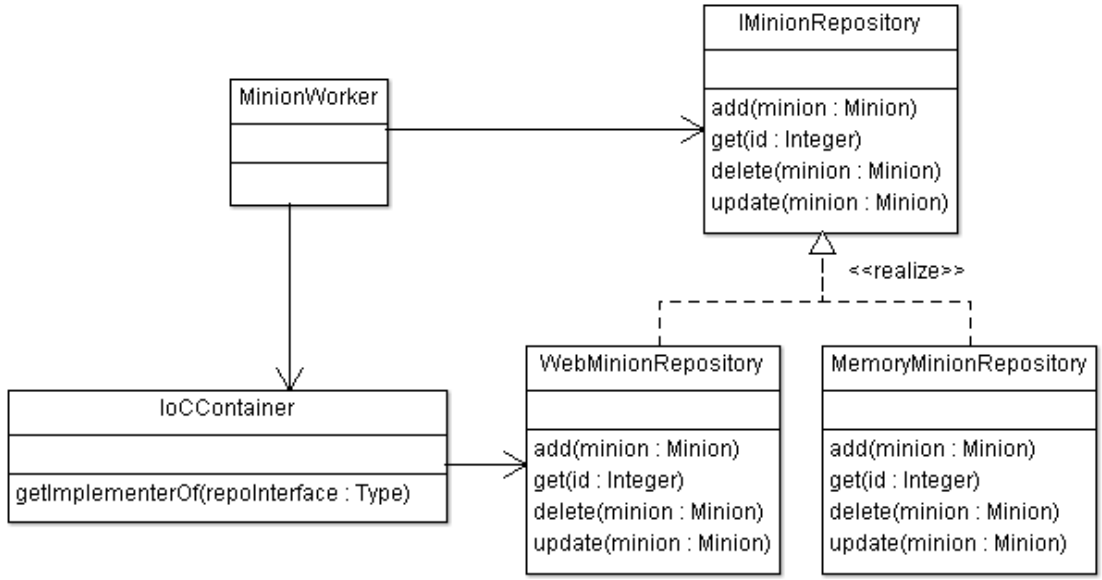


Fig. 11 Exemple de diagrama de classes de IoC + Repository Pattern

Això va ser de vital importància pel nostre projecte, doncs primer vam començar programant el client, i no va ser fins bastant més tard que vam programar el servidor. Llavors en un principi fèiem servir la memòria per emmagatzemar les dades (que eren esborrades al tancar l'aplicació i inicialitzades de nou al arrancar-la) i més tard, quan vam començar a implementar el servidor, el canvi a repositoris que fessin servir la API va ser relativament més fàcil. De fet, en el nostre cas encara hi ha menys desacoblament que en el de la Fig. 11, doncs el *IoC Container* agafa referències de "Interfícies" (el tipus, en si) a implementacions, i tot i que emmagatzema les implementacions per a cada una de les interfícies, ell no veu res més que el tipus de la interfície i de la implementació (Dit d'una altra manera, se suposa que algú li dirà "La realització d'aquesta interfície és aquesta classe", i més tard algú li demanarà "Donam la implementació d'aquesta interfície"). La classe que està acoblada amb les implementacions és la de inicialització, que és la que canviaríem quan volguéssim canviar la implementació.

En l'estructura de fitxers del projecte del client (en Unity3D), tots els repositoris es poden trobar dins la carpeta "Scripts/Repositories", que estan agrupats per carpetes. Dins de cada carpeta es troba la interfície del repositori i les diferents implementacions que s'han fet per cada un d'ells.

A la carpeta "Scripts/IoC" es pot trobar el IoC Container (una interfície del mateix i una implementació), i a la carpeta "Global" hi ha l'script IoCBinder.cs que configura el IoCContainer definint quina implementació li toca a cada repositori. Si es volgués canviar d'implementació, n'hi hauria prou en canviar-ho en aquest script.

Si fem un cop d'ull a la carpeta de repositoris hi trobem, per ordre alfabètic:

- CompanyWideIntervention: Serveix per obtindre els influenciadors a nivell d'empresa que s'hagin configurat des del panell d'administració. Hi ha implementacions tant per Memòria, per XML (que llavors treu les dades d'un fitxer .xml) i per Web.
- Course: El mateix que CompanyWideIntervention però per cursos per a minions
- Mail: Té dos repositoris que estan relacionats. Per un costat hi ha el MailTemplateRepository, que llegeix els correus que s'han configurat des del panell d'administració, i el MailRepository, que tracta amb els correus en si (per crear-ne, marcar-los com a llegits i eliminarlos.
- Minion: Per crear/modificar/eliminar minions. Hi ha una implementació per memòria directament (que es perd al reiniciar el joc) i per web, que es comunica amb el servidor per fer persistents els canvis.
- MinionInfluence: Registra l'acció de cada minion que està sent influenciat (la relació entre un Minion i un influenciador), que permet afegir-ne (quan comença) o eliminar-ne (per quan acaba)
- Motivation: El mateix que CompanyWideIntervention o Course, però per a motivacions.
- Recruitment: Serveix per obtindre les característiques dels minions que s'hagin configurat de del panell d'administració per al reclutament.
- Resource: Emmagatzema el nombre de plàtans que té la piràmide. De fet, està obert a diferents tipus de recursos per si en una altra versió se'n volen afegir.
- Vacancy: Per al sistema de reclutament, això guarda que el jugador ha començat/acabat de buscar nous minions per a la seva piràmide.

Events

Hem dissenyat un sistema d'events per poder connectar les diferents parts del joc d'una manera que redueix significativament l'acoblament. D'aquesta manera, una classe pot notificar un event, i tots els que s'hagin registrat a aquests rebran una crida al mètode que hagin indicat.

A la carpeta d'events podem trobar-ne els següents:

- ExceptionEventManager: Té l'event de quan hi ha hagut una excepció (d'aquesta manera quan hi ha alguna part de tot el joc que no funciona correctament se li pot mostrar al usuari un missatge)
- GameEventManager: Conté l'event de quan se surt del joc, per poder executar codi abans de que es tanqui.
- GoalEventManager: Conté els events que fan referència als objectius (quan s'han completat objectius, que recordem que es que la piràmide hagi agafat una certa alienació, i al nivell, que és quan s'han complert tots els objectius.
- MailEventManager: Té els events de quan es rep un nou mail o s'esborra.
- MinionEventManager: Té els events relacionats amb els minions, de quan s'han contractat, acomiadat, entrenats o motivats.
- PopupEventManager: Events per el sistema de Popups de la interfície (al obrir, tancar o quan l'usuari prem l'opció de confirmar).

Localització

Donat que l'equip de treball era multilingüe (set holandesos i un català) i que el joc està destinat per una classe d'una universitat dels Països Baixos, vam dissenyar un sistema molt bàsic per poder traduir fàcilment el joc.

A la carpeta GameUtils/Language hi ha la classe DisplayText que té un mètode estàtic (no fa falta instanciar la classe per poder-la fer servir) que donada una clau retorna el text associat a aquesta clau. Cada idioma s'emmaguetzema en un fitxer XML amb el següent format:

```
<?xml version="1.0" encoding="utf-8"?>
<strings>
   <string key="clau">Text</string>
   ...
</strings>
```

Quan DisplayText rep una petició per obtindre una traducció, amb l'ajuda de la classe XmlParser (que interpreta fitxers XML) llegeix del fitxer de traduccions (que va definit en una constant, PATH_LANGUAGES, però que es podria fer canviable si fos necessàri) per trobar-hi el text que toca. Per poder-hi posar salts de línia, DisplayText també substitueix `(br)` per un salt de línia.

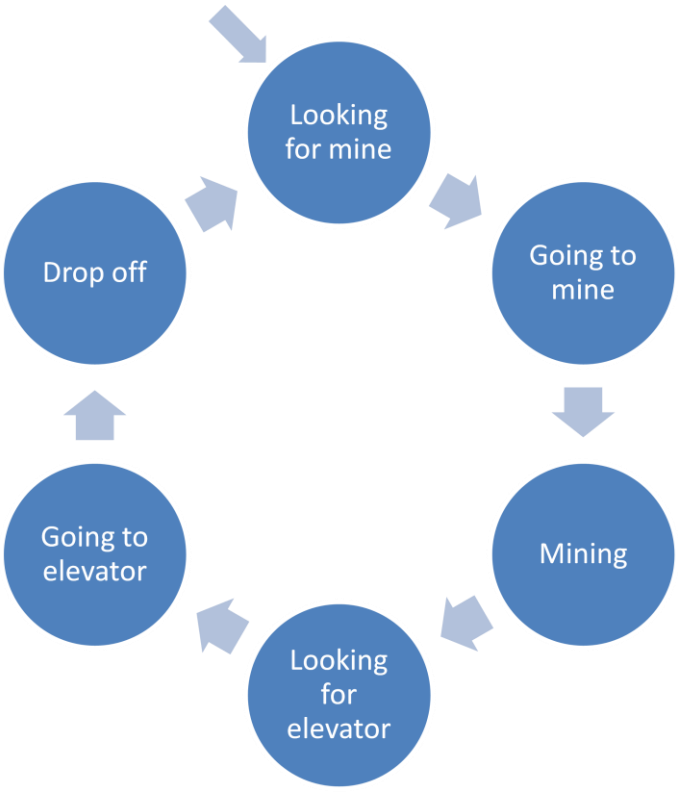
Minion IA

El nostre joc ha d'incloure una certa Intel·ligència Artificial per als minions, doncs aquests es representen com a objectes 3D dins de l'escena, i s'han de desplaçar per dur a terme la seva funció.

El motor que fem servir, Unity3D, ens estalvia molta part d'aquesta feina, donant-nos el *pathfinding* ja fet, mitjançant un *NavMesh*, que es un espai per on els objectes hi poden passar, i quan vols que un objecte es desplaci cap a una posició, Unity3D fa el pathfinding per trobar-hi el camí més curt. Dins del *NavMesh*, els objectes també poden colisionar entre ells, amb lo que tampoc ens hem de preocupar si dos minions es troben.

El que si que s'ha de programar és el comportament dels Minions. En tots els casos, es tracta d'un autòmat (màquina d'estats), però que les característiques del Minion tenen efectes sobre la seva productivitat. Anem per els 4 tipus de minions:

1. Miner



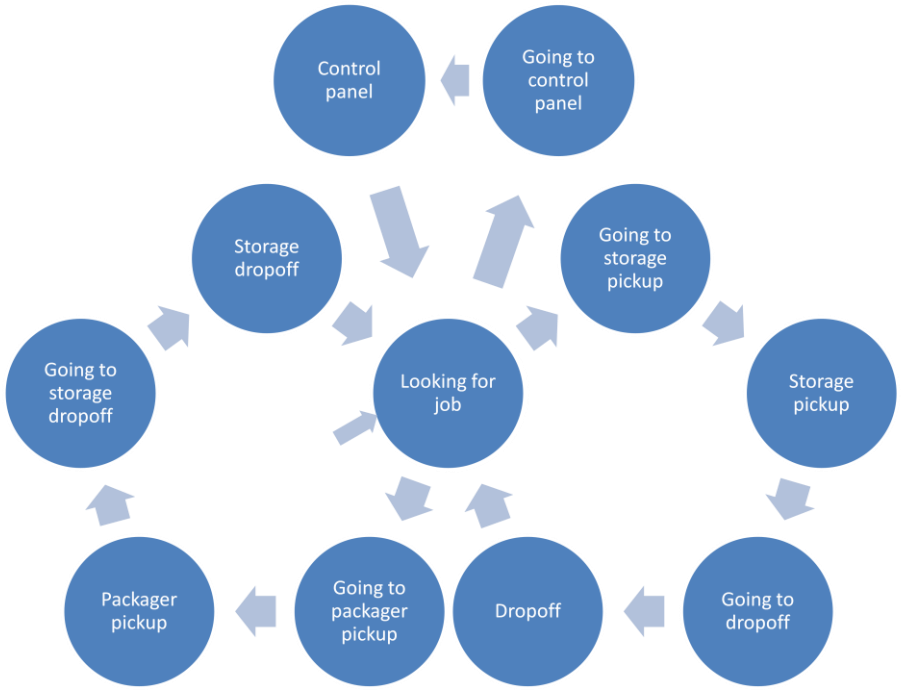
Gràfic 2: Autòmat del Minion miner

Tal i com es pot veure en el graf, el miner es molt circular. Al començar el joc, comença buscant una mina, buscant totes les mines que hi ha en l'escena aquella que no estigui completament ocupada (cada mina té un nombre màxim de minions assignada) i tingui

més a prop (amb les coordenades del minion i de la mina per Pitàgores s'obté la distància, i es busca el mínim de cada una). Incrementa el comptador de minions de la mina i la marca com a destinació (al *NavMesh*), passant al següent estat on es manté fins que no hi ha arribat.

En l'estat de *Mining*, s'espera un cert temps (que depèn de les característiques del minion) on representa que està treballant a la mina, i passa al següent estat, on busca un ascensor per deixar-hi els plàtans que ha extret i pujar-los a la següent planta (fent el mateix que abans amb les mines, pero ara amb els ascensors)

2. Empaquetador/producció:



Gràfic 3: Autòmat del minion empaquetador

Aquest és el autòmat més complex. Per entendre-ho, hi ha quatre llocs on poden estar els plàtans: Entrant o sortint de la màquina empaquetadora, als prestatges on s'emaguetzemen els plàtans empaquetats, o a la cinta per passar el control de qualitat o per enviar-los a vendre.

Inicialment els plàtans arriben del ascensor a la entrada de la màquina empaquetadora. Quan un minion des del "Looking for job" agafi el path del "Control panel" (explicaré més tard com pren aquesta decisió), aquest es dirigeix a la màquina per tal de que empaqueti els platans i els deixi a la sortida de la màquina.

Llavors quan un minion agafa el path de "Packager pickup", aquest agafa el paquet de la màquina i el diposita en els prestatges que estan en la mateixa planta, en l'estat

"Storage dropoff". I per últim, quan un minion agafa el path de "Storage pickup", aquest agafa el paquet que està en els prestatges i el porta a la cinta per tal de que passi a la següent fase de la cadena.

La part més complexa és la que pren la decisió de quin treball agafar en l'estat "Looking for job". Com que hi ha tres camins, el que fa es en una primera fase calcular la prioritat de cada camí segons bastantes variables, i després tria el camí que té més prioritat. Per exemple, si no hi ha paquets per agafar i no hi ha ningú treballant en el panell de control, va directament al panell de control. O si hi ha més minions treballant en treure paquets de la màquina que minions treballant al panell, llavors li dona una mica de prioritat, però podria ser que acabés triant els altres, si per exemple hi ha molts paquets a la sortida de la màquina i quasi bé cap en els prestatges.

Aquest sistema en la fase de proves va ser el que va donar millors resultats (en quant a que no hi havia cap coll d'ampolla), tant per quan només hi ha un minion treballant en aquesta planta com quan n'hi ha molts.

De fet, l'autòmat dibuixat en el gràfic 3 és una mica més simple del que està implementat. En el joc hi ha 2 estats més que són els inicials, per portar el minion a un punt en l'escena per tal de que comenci a treballar en aquesta planta.

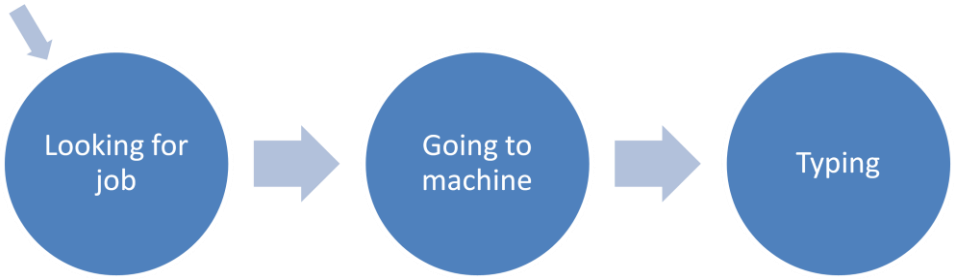
3. Control de qualitat:

Gràfic 4: Autòmat del minion de qualitat

Aquest també és molt bàsic, molt semblant al de miners però amb tres estats només. En aquest cas, en comptes de buscar mines o ascensors, en la planta hi ha unes quantes màquines i aparells que serveixen per passar el control de qualitat. En "Looking for job" s'escull la màquina o aparell més proper, al "Going to science" espera

a que el NavMesh dirigeixi el minion al aparell (la velocitat en la que es mou també es veu afectada per les característiques del minion) i al "Doing science" el minion treballa durant un temps per dur a terme el control de qualitat.

4. Màrqueting i vendes:



Gràfic 5: Autòmat de vendes

I per últim, aquest és el més simple, doncs és completament lineal. En la planta hi ha uns quants ordinadors on els minions de màrqueting venen els plàtans als clients finals. Els minions busquen l'ordinador que estigui lliure més proper, es dirigeixen a ell, s'asseuen i es queden escrivint, produïnt un cert nombre de plàtans cada cert temps (que tot això depèn, com sempre, de les característiques dels minions).

Interfícies

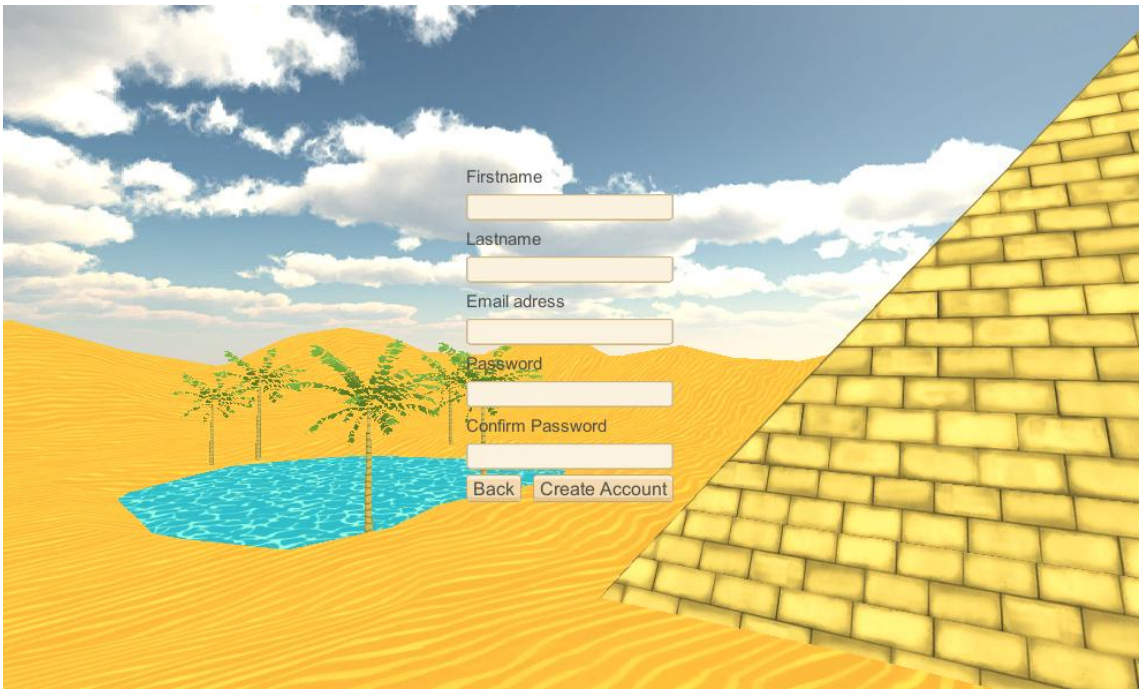
Per les interfícies vam decidir crear un tema que tingués una bona relació amb una piràmide, doncs era on passava el joc. Segons el llibre "Schell, Jesse. *The Art of Game Design, A Book of Lenses*", el tema d'un joc és una de les parts més obviades en general però més importants a l'hora de dissenyar un joc és el tema. Tot el joc ha de semblar que estigui en una mateixa harmonia per tal de que el jugador hi entri de ple. Per això vam estar pensant com dissenyar els botons, les textures, els shaders, etc. per tal de que al jugador li donés la impressió que realment està posat dins d'una piràmide, que en general té un estil bastant arcaic, però incorporant-hi tocs tecnològics i moderns.

Menú principal

Pantalla de login

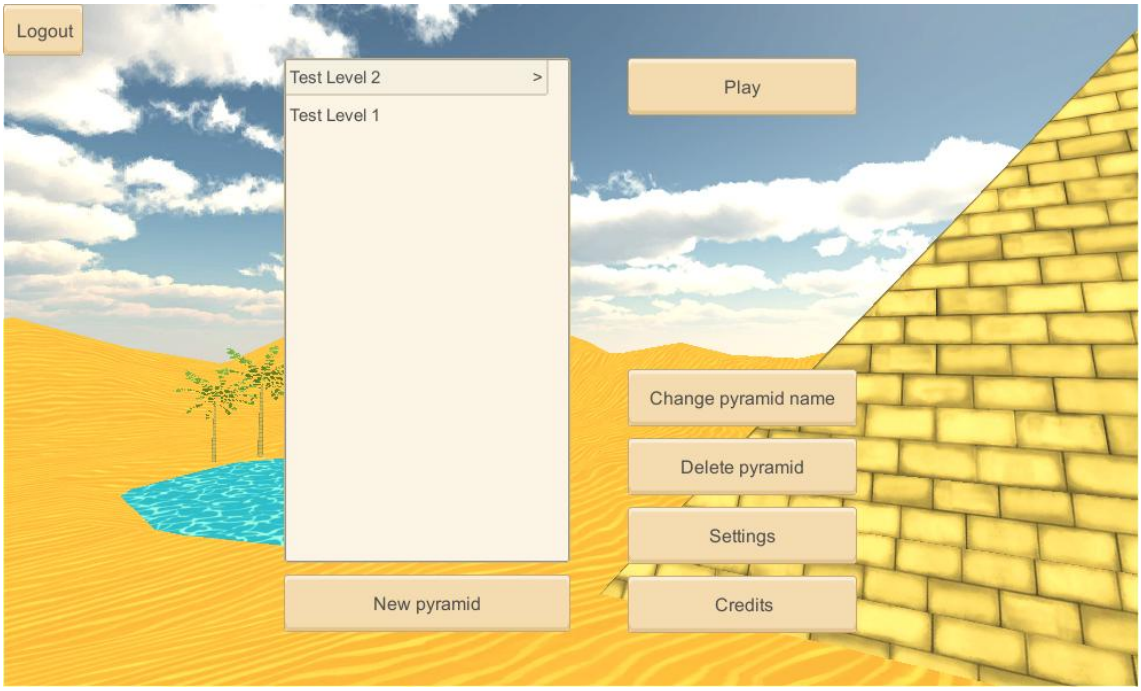
Vam voler donar un bon tema d'entrada, així que vam posar l'escena de la piràmide vista des de fora, amb el logotip del joc i el formulari per entrar-hi. Els jugadors amb el formulari poden fer login o poden crear una compte.

Crear un compte



Els usuaris al premer el botó de crear un compte, els hi surt un formulari bàsic per crear-ne un, demanant el seu nom (recordem que aquest joc està pensat per aplicar-lo a les aules), el correu electrònic i la contrassenya.

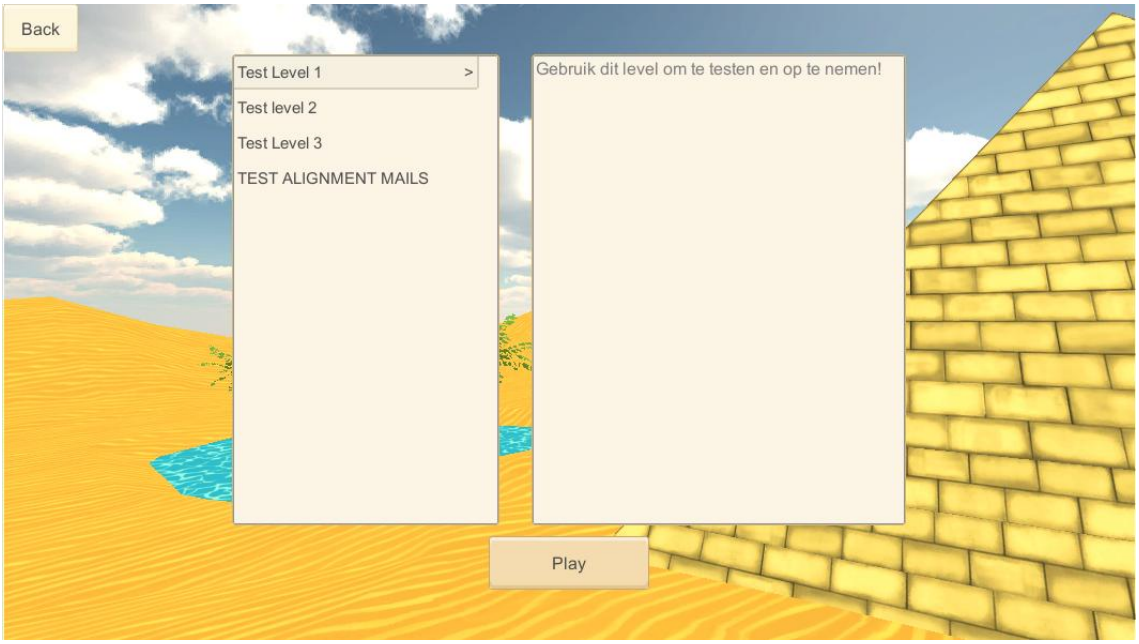
Selecció de partida



Quan l'usuari s'ha registrat o ha fet login, li apareix una pantalla on pot seleccionar una partida ja començada anteriorment per jugar-hi, crear-ne una de nova, canviar de nom la piràmide,

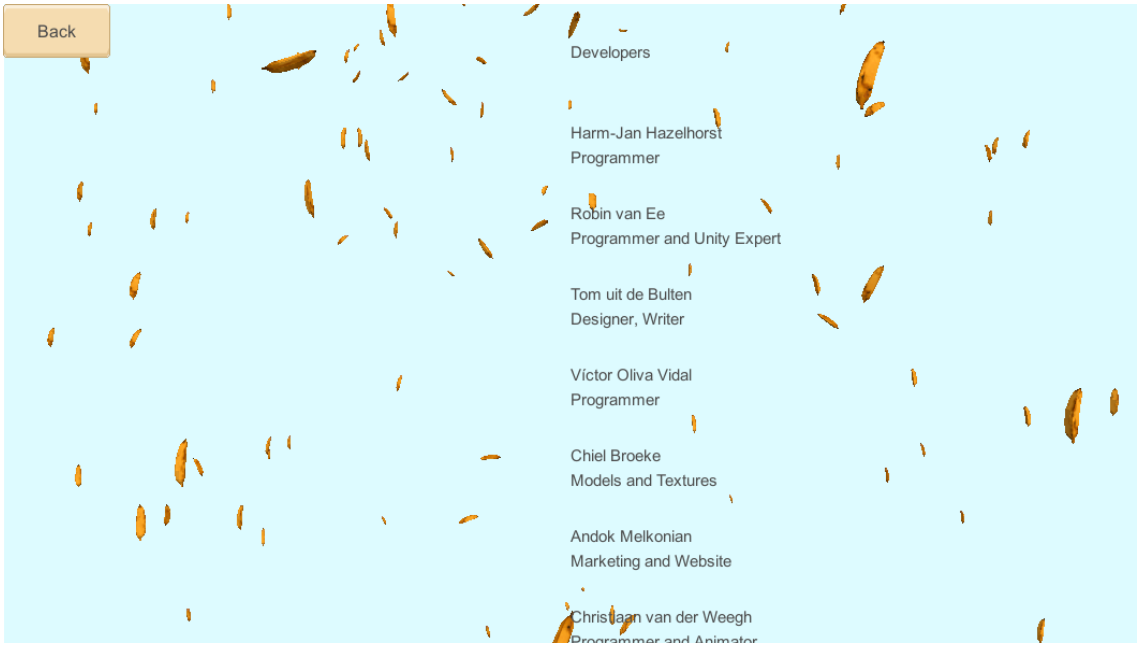
eliminar-la, sel·leccionar la configuració del joc (per augmentar/disminuïr la qualitat dels gràfics, o per canviar el volum dels sons) i accedir a la pantalla de crèdits

Nova piràmide



En la pantalla de selecció de piràmide, se li presenten al jugador els nivells que s'hagin configurat des del panell d'administració, mostrant el títol i una descripció (en aquesta captura les dades són d'exemple). El jugador al donar-li a Jugar se li demanarà que li posi un nom a la partida, oferint-li un d'aleatòri.

Crèdits



La pantalla de credits vam fer un efecte divertit de deixar caure plàtans infinitament mentre la llista de crèdits va fent un scroll amunt.

Joc

Interfície bàsica

L'interfície pròpia del joc es tracta en una barra informativa a la part superior de la pantalla, i d'un panell de control en la part inferior.

La barra superior indica la data del joc (que es pot fer servir com a condicions per disparar certs events), el nombre de plàtans que el jugador disposa actualment, i tres botons per accedir a funcions fora del joc, com l'ajuda, canviar la configuració del joc o sortir al menú principal.

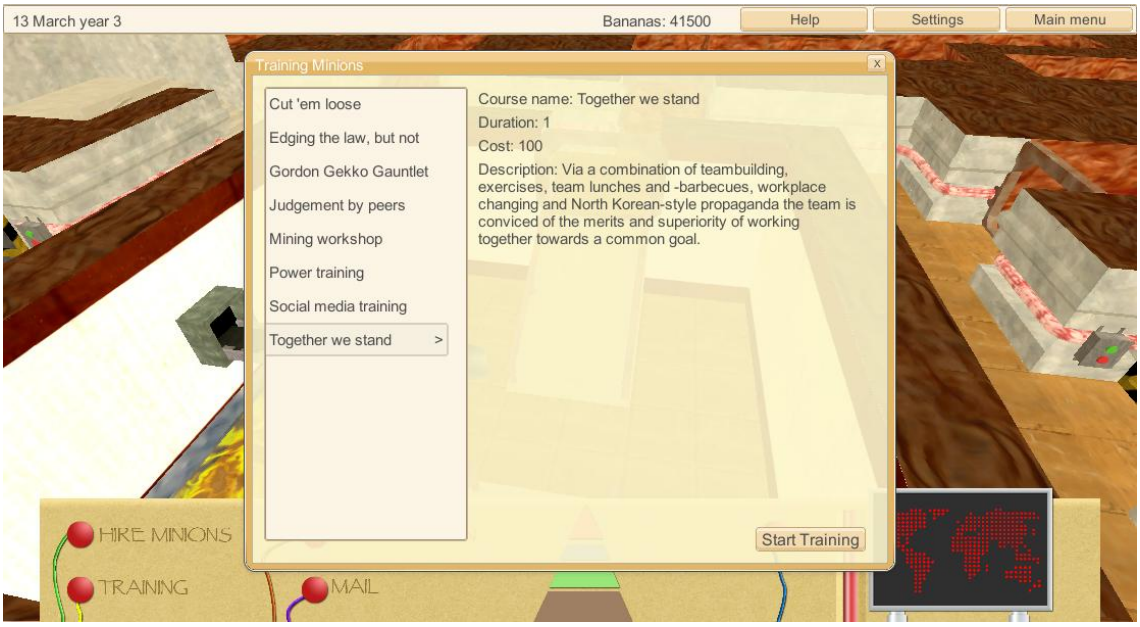
El panell inferior és el que l'usuari fa servir per donar ordres als minions o moure's entre les plantes, tot i que per donar-li utilitat en l'escena en si, també hi pot accedir mitjançant accions dins de l'escena (per exemple, clicant sobre un minion porta al detall del minion seleccionat, on també li pot donar ordres, motivar-lo, etc).

Contractar minions

La finestra per contractar minions permet al usuari sel·leccionar quants minions vol contractar i les seves característiques (que com sempre es poden configurar des del panell de control). També pot decidir per avançat quin càrrec tindràn els minions.

Influenciadors

Recordem que hi ha tres tipus d'influenciador: decisions a nivell d'empresa, formació i motivació. Tots els influenciadors comparteixen la mateixa pantalla:



En la que es veu una llista amb tots els influenciadors que s'han configurat en el panell de control, amb un detall que diu la duració que té aquest influenciador, el cost i una descripció que explica quin és l'efecte.

Llavors en el cas de la formació, surt la següent pantalla per escollir quins minions assistiran al curs:

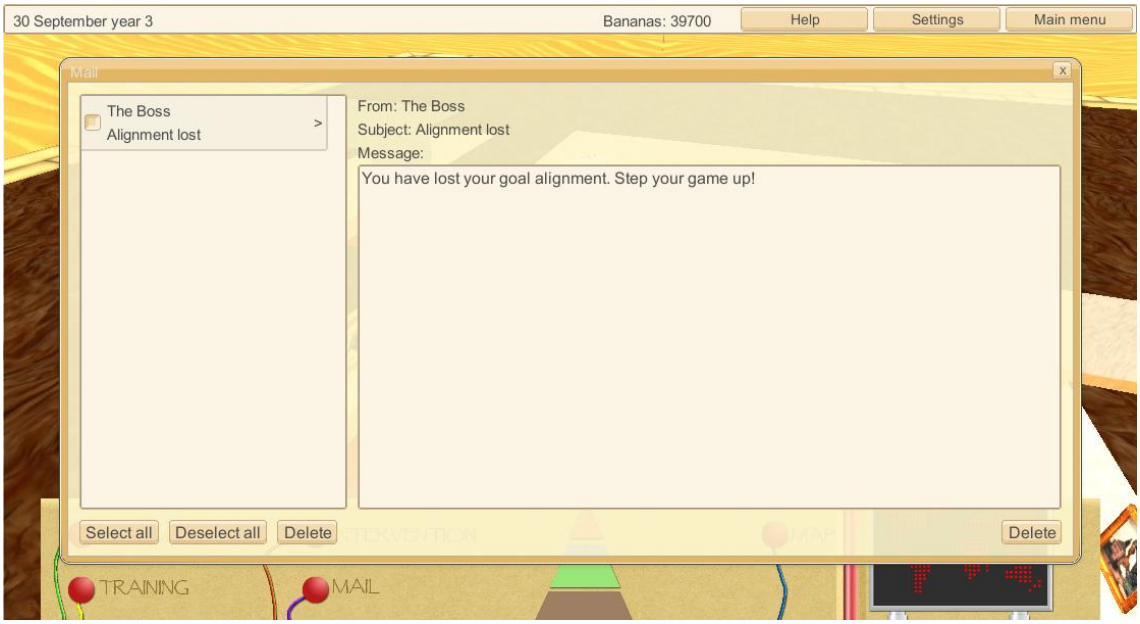
En el que es veu una llista amb tots els minions de la piràmide que poden fer el curs, i un petit resum dels seus detalls. El jugador selecciona els minions que vol i l'interfície l'informa del cost total.

Detall del minion

Com hem dit abans, quan el jugador fa clic sobre algun minion, aquest es resalta amb un efecte com d'energia que li surt del cap i apareix la finestra del minion, on surten totes les seves característiques i l'alineació que té actualment, junt amb els cursos o motivacions que està prenent.

Des d'aquesta finestra també pot canviar de posició de treball al minion, oferir-li formació, motivar-lo (obrint les interfícies que toquen) o acomiadar-lo.

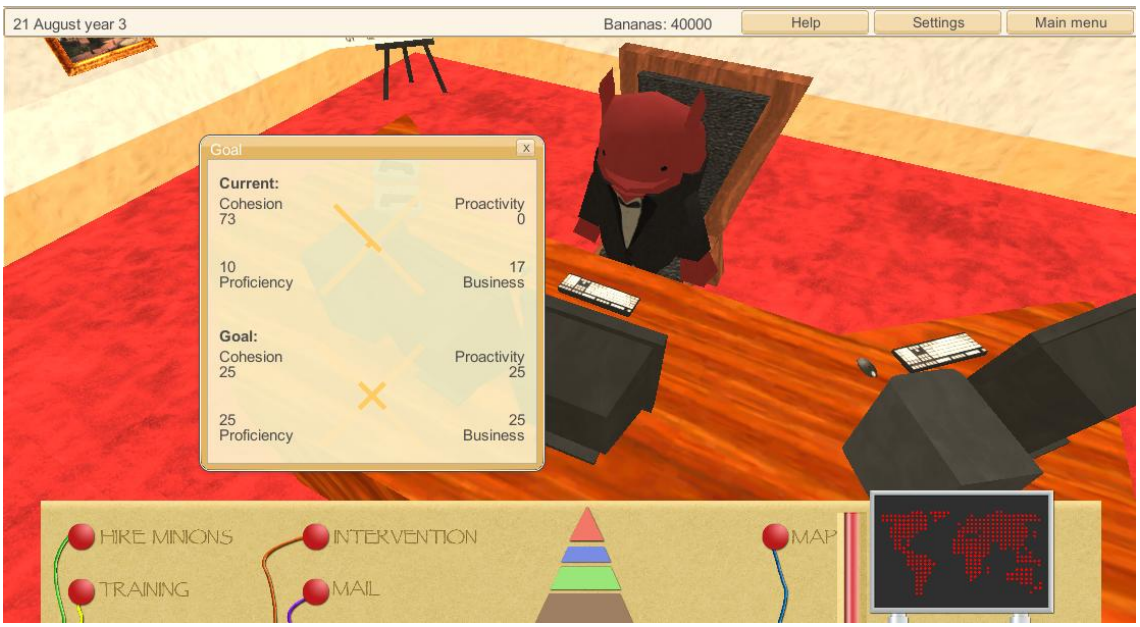
Correu



La pantalla de correu mostra una llista de tots els correus que hagi guardat el jugador, i un detall a la dreta que surt el correu en si. Se li ofereixen botons per seleccionar / des-selecionar els missatges i eliminar-los de cop, o eliminar-ne el que està llegint amb el botó de la dreta.

Accions a la planta del cap

A la planta superior, on es troba el cap (Boss), hi ha unes cuantes accions que l'usuari pot fer per veure com va la seva organització. Si, per exemple, clica sobre el *boss*:



Apareix una finestra que diu quina es la alineació actual de la seva piràmide i quin és el seu pròxim objectiu. Si en canvi fa clic sobre els arxivadors:

Apareix una finestra amb la llista de tots els minions, amb uns botons per poder acomiadar els que hagi seleccionat i un altre per contractar-ne de nous. I per últim, si li dona al gràfic de vendes:

Se li mostra una finestra amb el rànquing, que tal i com s'havia dit en el concepte, en té dos, un per la producció de plàtans i un altre per com de aprop estan de complir els seus objectius.

Servidor

L'altre part important del joc, però que pel jugador final no és visible, és el servidor. No només té la funció de mantindre totes les dades de tots jugadors per tal de que s'hi puguin connectar des de qualsevol lloc, sinó que a més ha de d'oferir una manera per tal de que els administradors puguin farcir el joc amb contingut relacionat amb l'assignatura i poder monitoritzar els jugadors que hi participen.

Configuració

Comencem parlant de la configuració del servidor. Com s'ha dit en la secció de tecnologies, cal instal·lar el Framework laravel, que demana tindre Composer (un gestor de dependències de PHP), PHP 5.3.7 i l'extensió MCrypt. En el cas del nostre codi, com que fem servir alguns *traits* (fragments de codi que es poden replicar en diferents classes, permetent així l'herència múltiple), el servidor necessita tindre la versió 5.4 de PHP.

Com a base de dades hem fet servir MySQL versió 5.5, tot i que no tenim una versió mínima, doncs la part de gestió de la base de dades la fa Eloquent.

I per últim cal configurar algun servidor per rebre peticions de Cross-Domain Requests. Vam estar estudiant diferentes possiblitats de comunicació que té Unity, i vam trobar que la que menys ens limitava era fer servir una llibreria que requeria Cross-Domain Requests (les altres ens limitaven en quant a suport per APIs REST). Unity demana al servidor la política de peticions entre dominis és pel port 23 (per defecte és el 843, però és configurable i el vam canviar al 23 perquè només certs ports estan oberts en la xarxa que ha de córrer el joc, i vam aprofitar el 23 que estava obert per Telnet), seguint el format d'Adobe per flash (enviant un XML i esperant rebre un altre XML).

Per no haver de fer córrer una altra aplicació escoltant al port 23, ja sigui programant un servidor en java o en un Node.js, hi ha un mòdul per Apache que muntant un host virtual rep les peticions i serveix un fitxer configurable de política Cross-Domain. Més informació sobre aquest mòdul es pot trobar a la bibliografia d'aquesta memòria.

Per obtindre el codi del servidor, també es necessita Git.

Instalació

Si tot està correctament configurat, instal·lar el servidor és molt senzill. Tant sols cal executar les següents comandes (per UNIX):

```
$ git clone -b web-backend https://github.com/harmjanh/flying-
panda-games.git minion-mundo
$ cd minion-mundo
$ sudo chmod -R 777 app/storage
$ mkdir public/export-dump
$ sudo chmod -R 777 public/export-dump
$ php artisan migrate
```

El que fan aquestes comandes es descarregar-se el codi des del repositori públic de github, i després dona permisos de control total a algunes carpetes que és necessari pel correcte funcionament del servidor. En concret, app/storage és una que fa servir Laravel per poder funcionar, i public/export-dump és on s'emmagatzemen temporalment els fitxers .csv i .zip per poder-los descarregar. Si es volgués més seguretat, seria suficient en donar permisos d'escriptura per l'usuari del sistema que corre el servidor Apache.

Per últim, "php artisan migrate" configura automàticament la base de dades, donant-li l'estructura que li toca i les funcions preprogramades.

Base de dades

Per augmentar la rapidesa d'alguns casos, vam decidir crear alguns *Procedures* de MySQL. Per exemple, per calcular el rànquing dels jugadors: si cada vegada que un jugador vol obtindre el rànquing el servidor ha de calcular la posició de cada un (que es fa ordenant les entrades de la base de dades per un cert valor), es lògic pensar que gastaria molts recursos al servidor. Llavors la primera solució que ens ve en ment es la de emmagatzemar el rànquing de cada jugador quan es demani (en el nostre cas, configurant un *cronjob* cada 8 hores, però podria ser perfectament quan ho demanés un jugador passat un cert temps del últim càlcul), i després servir-lo sense haver-lo de calcular (com si es tractés d'una *cache*). El problema que té aquest sistema, és que llavors necessitem N+1 peticions a la base de dades (doncs primer hem d'obtindre els jugadors ordenats i després hem de dir que el primer és el primer, el segon és el segon, etc.), amb lo que si el nombre de jugadors creix, el temps per fer el càlcul també creix. Si agrupem totes aquestes peticions en una mitjançant un *Procedure*, reduïm considerablement el temps. El codi d'aquest és:

```
CREATE ROCEDURE `update_ranking`()
BEGIN
 SET @effect = 10000000;
SET @r = 0;
 UPDATE saved
  SET bananas_rank = @r := (@r+1)
  WHERE deleted_at IS NULL
```

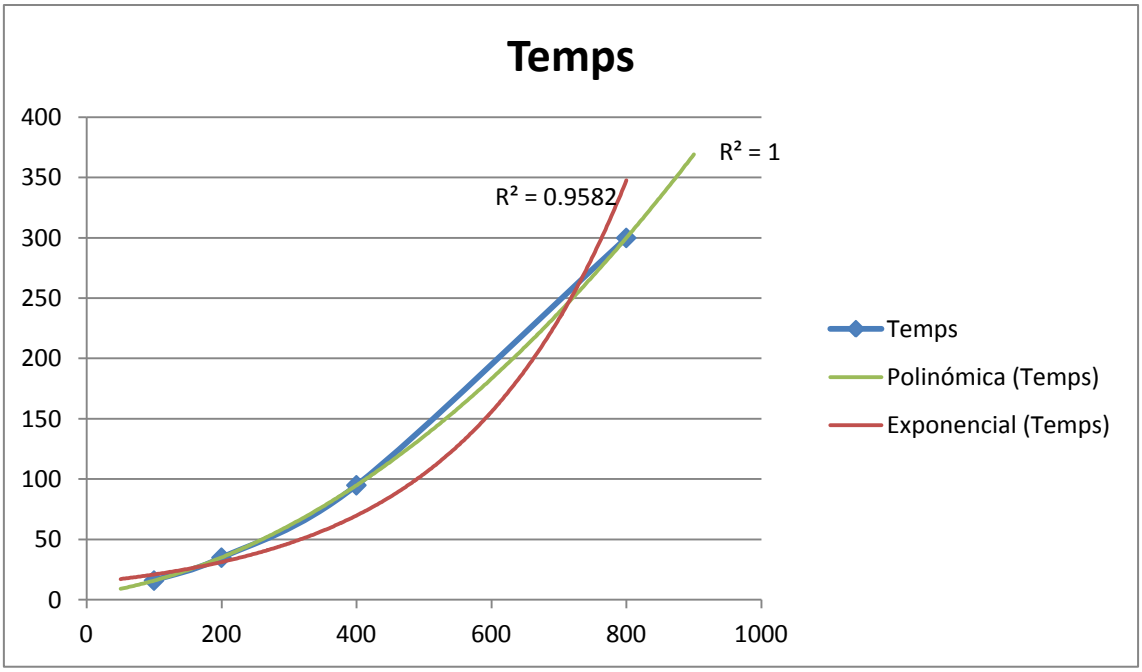
```
  ORDER BY bananas_produced * @effect / (@effect + time_played)
DESC;

 SET @prev := null;
 SET @r = 0;
 ALTER TABLE saved ADD _prev numeric;

 UPDATE saved
  SET
   alignment_rank = IF (
    @prev <> level_id,
    @r := 1,
    @r := (@r+1)),
   _prev = (@prev := level_id)
  WHERE deleted_at IS NULL
  ORDER BY level_id, goal_distance ASC;

 ALTER TABLE saved DROP _prev;
END
```

Com que hi ha 2 rànquings, hi ha 2 parts: Una pel nombre de plàtans produïts, i l'altre per la distància al proper objectiu.

Pel nombre de plàtans és relativament senzill: inicialitzem una variable @r a 0, i fem un *Update* ordenat per una funció (que depèn del nombre de plàtans i el temps jugat del usuari) posant el rànquing de plàtans com a @r, incrementant @r per 1.

Per al següent ja és més complicat, perquè si féssim el mateix que en el cas dels plàtans estaríem barrejant els diferents nivells, que tenen diferents objectius i per tant diferentes dificultats. Llavors hem de crear un rànquing per la distància al objectiu, però agrupat per nivells. Això ho fem ordenant primer un camp i després l'altre dins del mateix *UPDATE*, i comprovant que el nivell de la piràmide que estàvem abans segueix sent el mateix. Si no és el mateix, reinicialitzem el comptador @r a 1. En aquest cas, necessitem també una variable que se'n recordi del nivell de la piràmide anterior, i MySQL no permet fer assignacions dins d'un *UPDATE* si no es modifica algun valor de la fila. És per això que creem una columna temporal "_prev" per poder-hi emmagatzemar aquest valor, i després l'eliminem.

Com que la taula es redimensiona al afegir una columna, això pot portar problemes de rendiment depenent del motor que es faci servir. De totes formes hem fet algunes proves i hem vist que el temps que necessita és polinòmic amb el nombre de jugadors, amb lo que hem decidit mantenir-ho així, doncs no portava massa temps pel volum de jugadors que esperàvem. Si creés algun problema, es podria considerar l'opció de mantenir aquesta columna temporal en comptes de crear-la i destruir-la cada vegada que es vol fer el rànquing.



Gràfic 6: Temps de 'update_ranking()' en funció del nombre de jugadors

Els altres procedures que vaig crear eren per satisfer el requisit de mantenir un log de les accions dels usuaris. Hi ha accions d'aquestes que van vinculades amb una fila d'alguna taula que volem mantenir les seves dades del moment que s'ha fet l'acció, però no volem fer una copia de la fila cada vegada que hi ha una acció perquè és possible que entre varies accions la fila no hagi canviat i estem fent còpies redundants de la mateixa fila. Llavors la idea es fer la còpia quan es modifiqui la fila que se n'ha fet un log.

Però sorgeix un problema: al fer una copia de la fila, el camp identificador se'n crea un de nou i per tant totes les relacions (foreign keys) es perden. Per solventar això el que vam decidir va ser crear una columna en les taules implicades anomenada "*Reference*" que és lo que apunta la *foreign key* dels Actions Logs. Llavors quan es fa una copia al original se li assigna un nou reference, i per la copia es copien les relacions que es volen mantindre. Posem-ne un exemple: pels minions, volem mantenir els influenciadors que tenia en aquell moment, i suposem que partim del següent:

Influenciador		Minion			Actionlog
Id	Minion_id	Id	Nom	Reference	Minion_ref
4	70	70	Pepito	30	30

I suposem que algú vol canviar el nom del minion, llavors haurem de fer una copia:

Influenciador		Minion			Actionlog
Id	Minion_id	Id	Nom	Reference	Minion_ref
4	70	70	Pepito	30	30
		71	Pepito	31	

Però hem d'actualitzar la referència per fer que la copia es mantingui com a còpia (I no haguem de canviar totes les taules que apunten al Id del minion):

Influenciador		Minion			Actionlog
Id	Minion_id	Id	Nom	Reference	Minion_ref
4	70	70	Pepito	31	30
		71	Pepito	30	

I per últim, com volem mantenir els influenciadors que tenia en aquell moment, també hem de copiar el seu influenciador. I canviant-li el nom quedaria:

Influenciador		Minion			Actionlog
Id	Minion_id	Id	Nom	Reference	Minion_ref
4	70	70	Juanito	31	30
5	71	71	Pepito	30	

Fent això, el action_log està apuntant al mateix minion que abans, i es mantè la integritat de les dades. A continuació el codi d'aquest exemple:

```
CREATE PROCEDURE check_reference_minion(IN _id INT)
BEGIN
     SELECT @referencers := COUNT(*) FROM action_logs, minions
WHERE minions.reference = action_logs.minion_ref AND minions.id
= _id;
     IF @referencers > 0 THEN
          SELECT @maxref := MAX(reference) FROM minions;

          DROP TABLE IF EXISTS tmptable;
          CREATE TEMPORARY TABLE tmptable SELECT * FROM minions
WHERE id = _id;
          UPDATE tmptable SET id = NULL;
```

```
            UPDATE minions SET reference = @maxref+1 WHERE id =
_id;

            INSERT INTO minions SELECT * FROM tmptable;
            DROP TEMPORARY TABLE IF EXISTS tmptable;
            SET FOREIGN_KEY_CHECKS = TRUE;

            SET @copy = LAST_INSERT_ID();

            CREATE TEMPORARY TABLE tmptable SELECT * FROM
minions_being_influenced WHERE minion_id = _id;
            UPDATE tmptable SET minion_id = @copy;
            INSERT INTO minions_being_influenced SELECT * FROM
tmptable;
            DROP TEMPORARY TABLE IF EXISTS tmptable;
      END IF;
END;
```

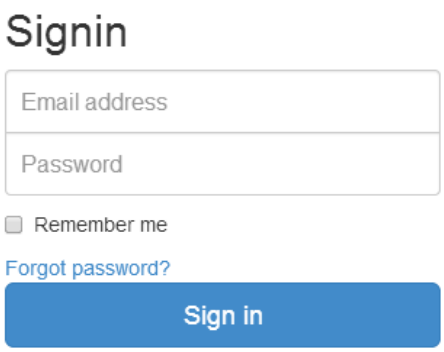
Primer el que fem es comprovar que hi hagi alguna referència als action logs del minion (perquè si no hi ha cap, no cal fer cap canvi). Llavors el que fem es fer una còpia sencera de la fila del minion a una taula temporal, perquè sinó tindríem el problema de que durant un temps, hi haurien dos files amb la mateixa Id en la mateixa taula, i deixem la Id a NULL en la còpia perquè quan la passem a la taula de minions se li assigni una Id automàticament (amb el *autoincrement*). Tot seguit, assignem una nova referència al nou minion i el copiem a la taula original. Per últim, cal copiar tots els influenciadors del minion i cal fer que apuntin a la nova copia.

Les altres funcions tenen el mateix aspecte que aquest mètode, però cada un s'adapta una mica als seus requisits, sobretot en el part de copiar les files de les taules que el referencia (per exemple, en aquest cas eren els influenciadors).

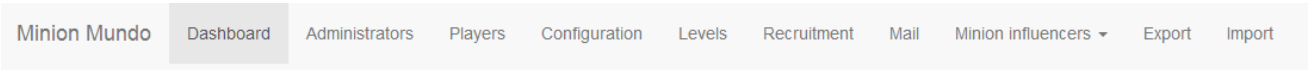
Frontend

Per al frontend del servidor, el panell d'administració, hem fet servir el framework de Twitter Bootstrap, al permetre un desenvolupament molt ràpid amb una *look-and-feel* decent.

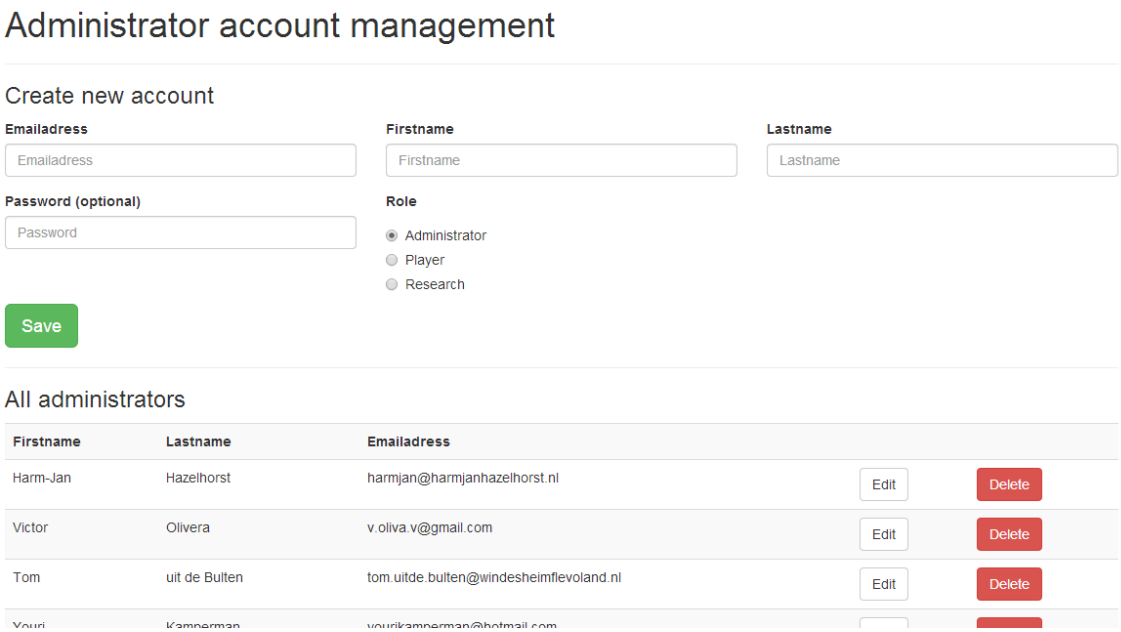
Al entrar al panell d'administració, a l'URI del servidor "/admin", si l'usuari no està autenticat se li mostra el formulari de login, amb les credencials del usuari, un checkbox per mantenir la sessió oberta i un link a una pàgina per recuperar la contrassenya (enviant un e-mail al que l'ha perduda):



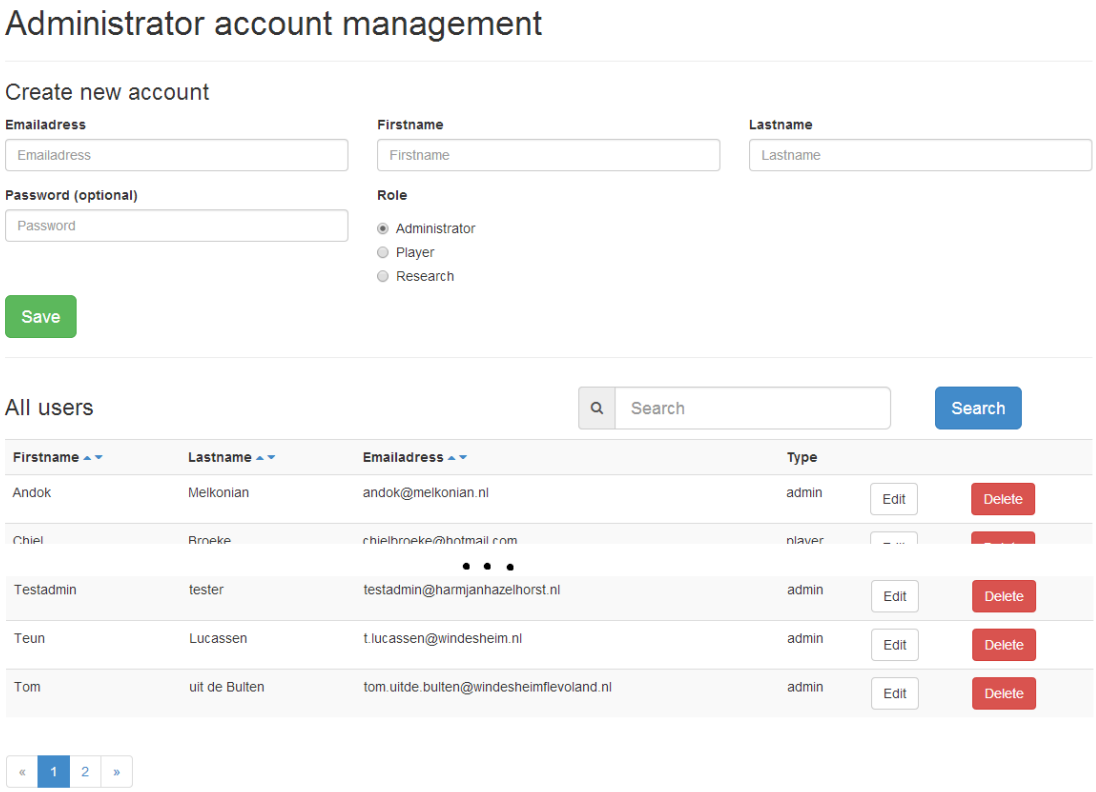
Una vegada l'usuari ha entrat correctament, entra a una pantalla de benvinguda i a la barra de navegació de dalt li apareixen una sèrie de links:



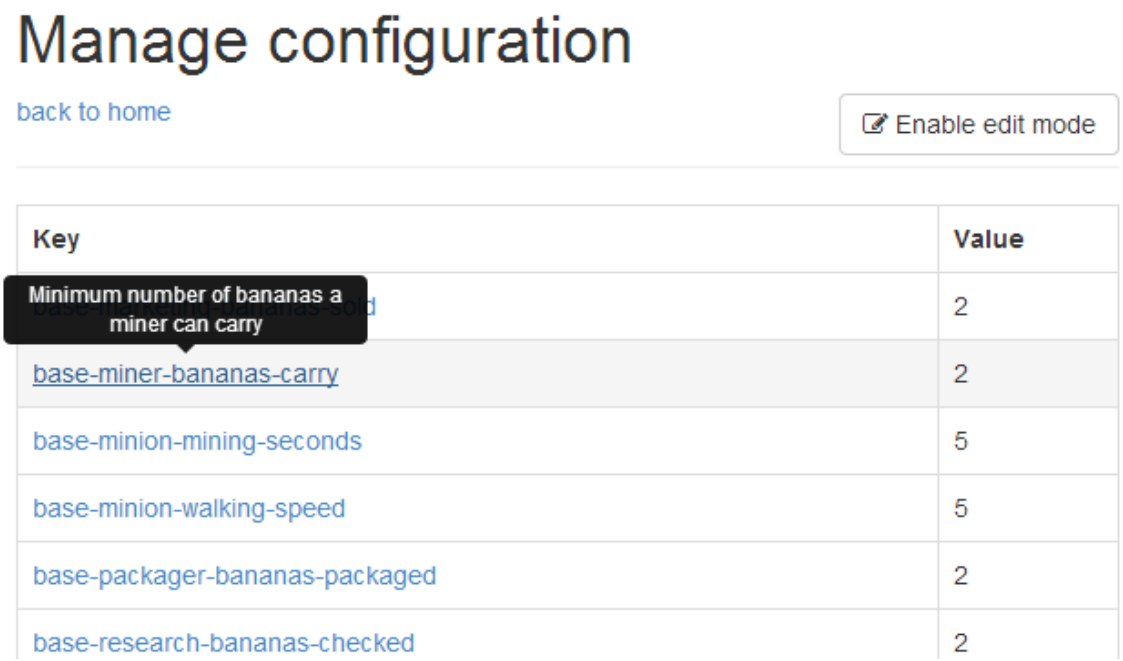
Anant per parts: A la pàgina "*Administrators*", permet veure una llista de tots els usuaris al grup d'administradors, així com gestionar-los (editar-los o eliminar-los). En la mateixa pàgina també permet crear noves comptes, siguin del grup que siguin.



La següent pàgina, la de jugadors, mostra la mateixa llista però per tots els usuaris (els administradors també), amb una columna extra que indica a quins grups pertany. Com que en aquest cas el nombre d'usuaris pot ser molt gran, hi ha un sistema de paginació, mostrant 20 jugadors a cada pàgina. Si es vol trobar un jugador, també hi ha un camp per a poder-los buscar:

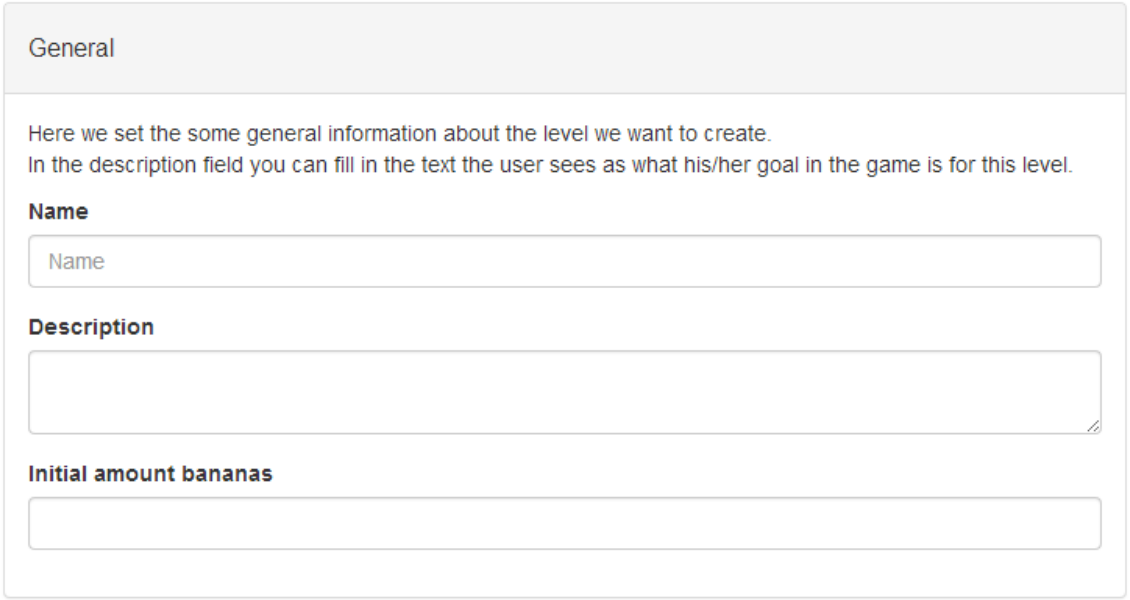


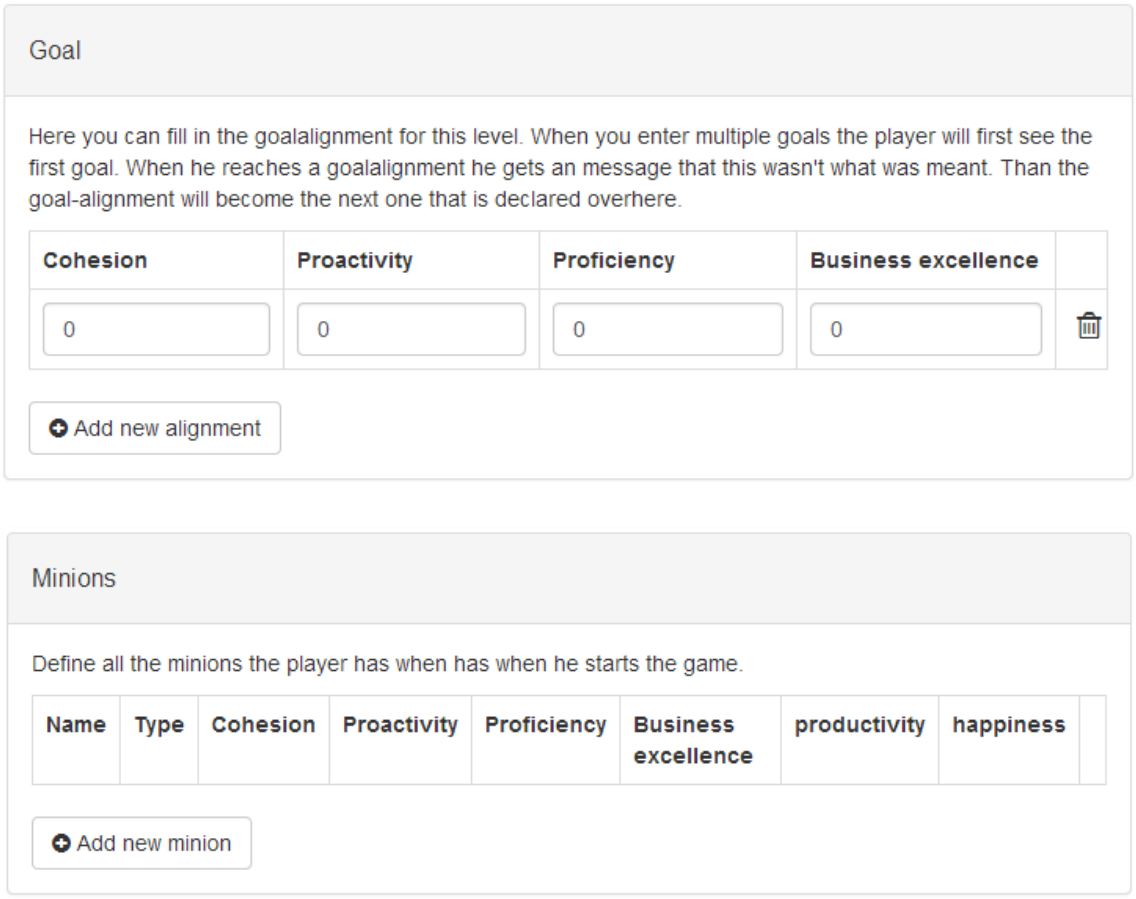
La pestanya de configuració permet veure els valors dels paràmetres del joc i els permet editar (clicant sobre el botó de editar converteix tots els valors en "inputs" per editar-lo, amb un botó a baix per guardar-ho. Passant el ratolí per sobre de cada clau de configuració, mostra una breu descripció per tal de que s'entenguin més aquells dels quals n'hi pugui haver dubtes.



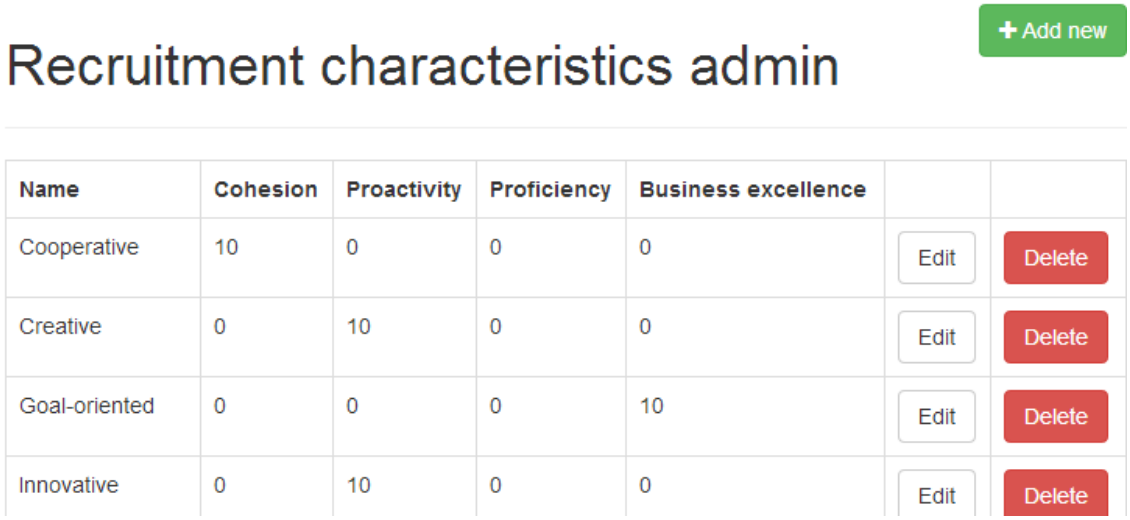
La següent pàgina és la d'administració dels nivells. En aquesta pàgina es poden veure, editar o eliminar els nivells que s'hagin creat, o crear-ne de nous.

Si cliquem per afegir un nou nivell (o editar-ne un) podem posar-li els valors bàsics (nom i descripció) i assignar-li un nombre inicial de plàtans, minions (amb les seves característiques i els objectius del nivell.

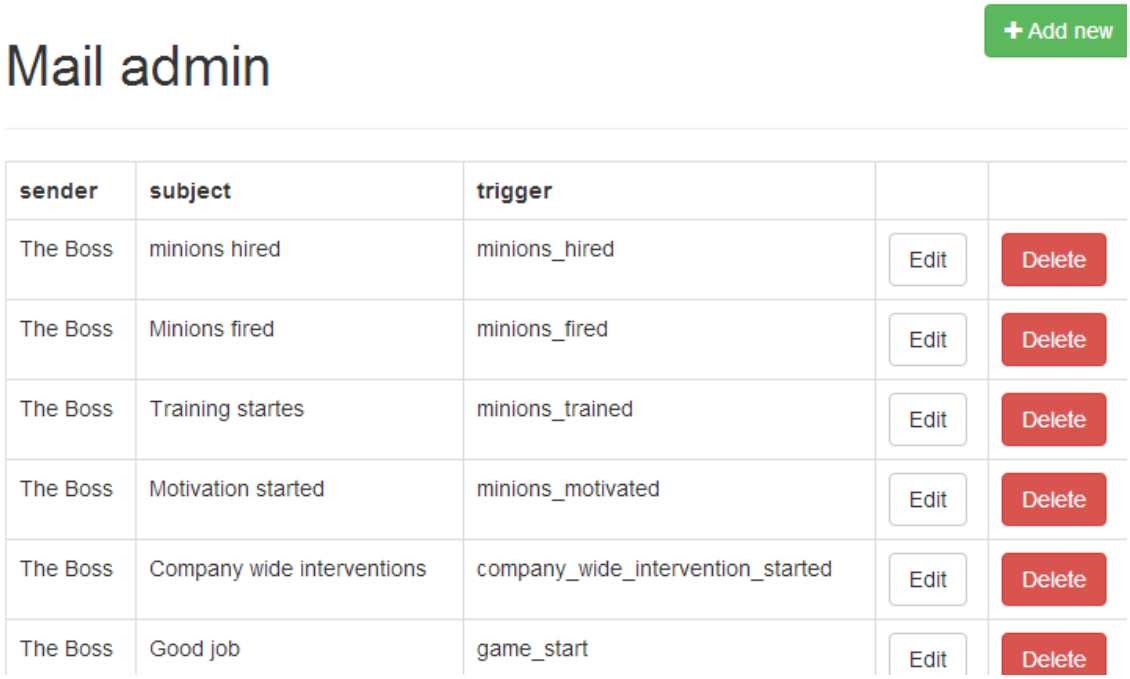




La següent pàgina és per configurar les variables de reclutament. Recordem que quan el jugador vol reclutar nous minions, ha d'escollir d'una sèrie de característiques que farà donar una certa alineació a cada minion (amb un petit factor aleatòri). Des d'aquesta pàgina es poden crear/editar/eliminar-los.

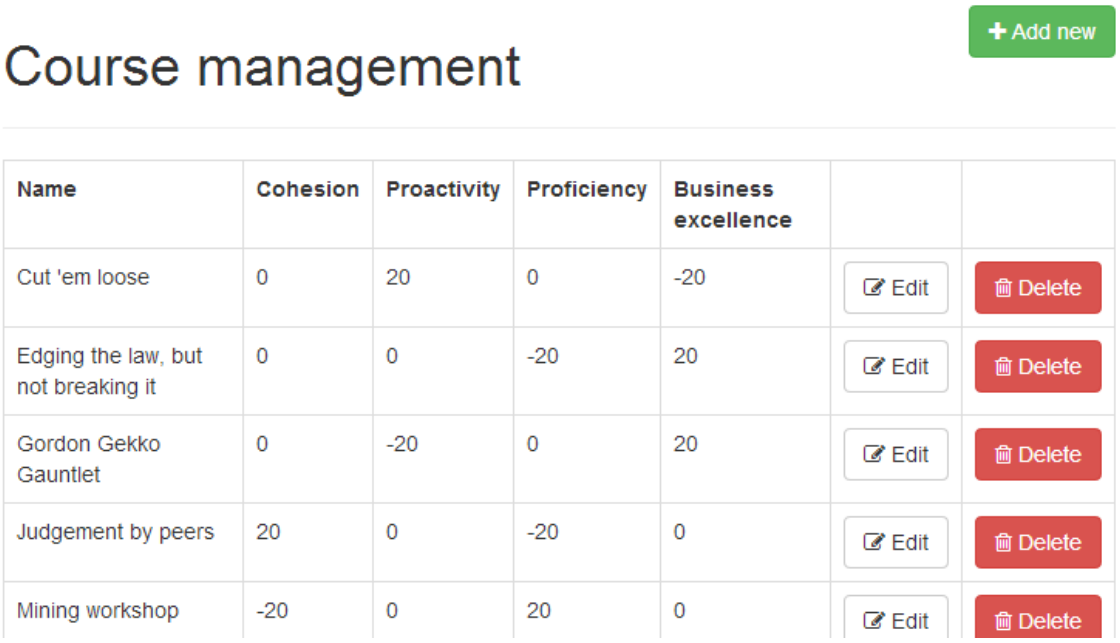


Després tenim la *secció Mail*, on es configuren els correus assignant qui els envia, el assumpte, el disparador i el contingut.

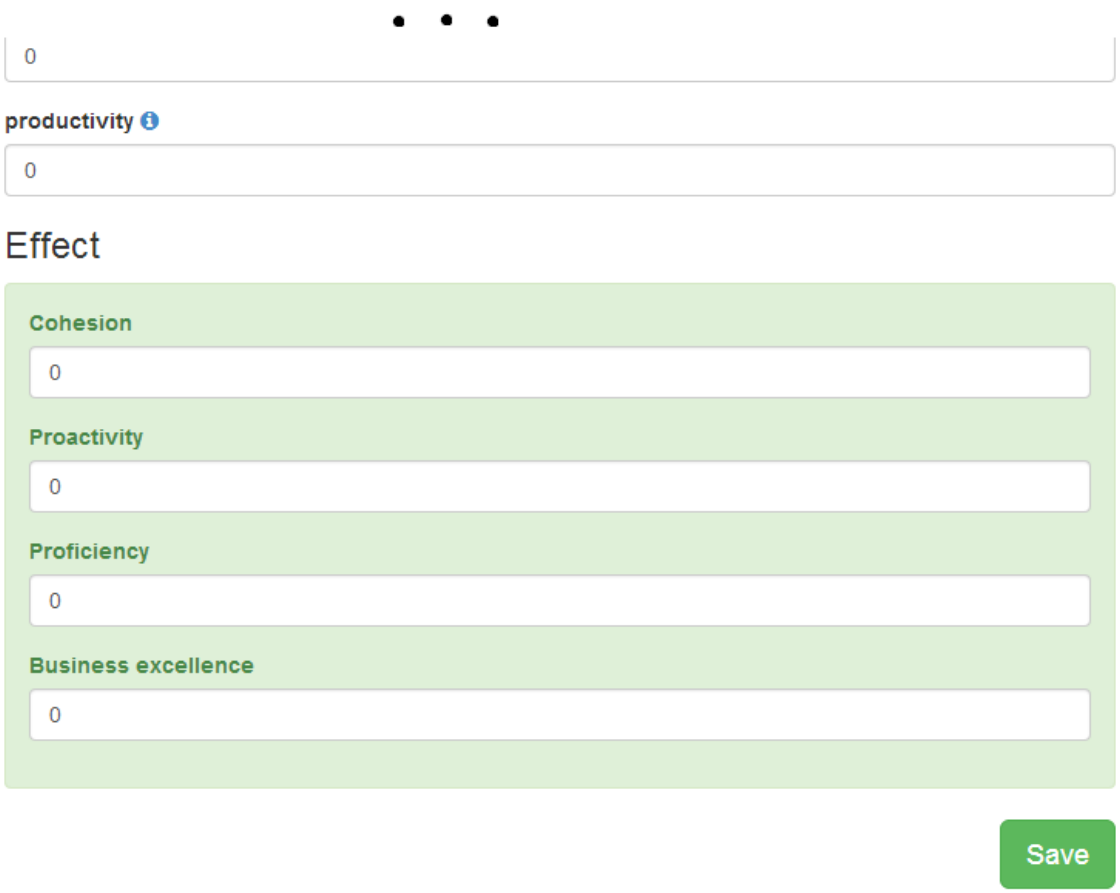


L'última pestanya de configuració del contingut del joc és la dels influenciadors. Aquest està separat en tres pàgines diferents per tindre'ls més ben separats (una per cada tipus, Motivacions, Formació i Decisions d'empresa).

Com a totes les altres pàgines, primer surt una llista amb els influenciadors actuals amb la possibilitat d'afegir-ne, editar-los o eliminar-los. Quan n'afegim o editem un podem posar-li tots les seus paràmetres (a part del nom, la descripció i del cost, també la corva d'aplicació que té, amb la duració i l'efecte sobre la felicitat i la productivitat base del minion). Pels efectes sobre l'alineació del minion comprova que compleixi amb les restriccions en temps real (i.e., que només un quadrant pot obtindre punts i els ha de treure dels quadrants adjacents en horitzontal o vertical.

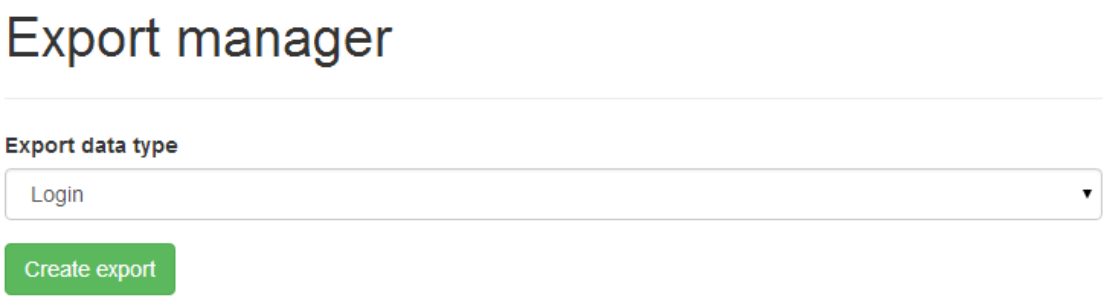


I la part del final de la pantalla d'editar / crear un nou influenciador:

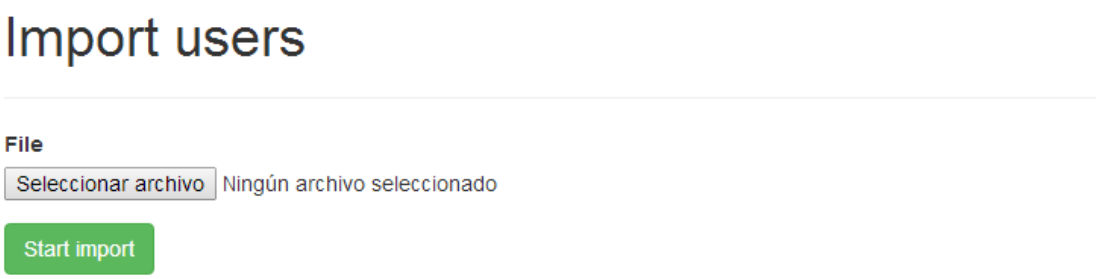


Aquesta part verda va validant l'efecte que l'usuari hagi introduït, i si és incorrecte segons les normes de HRM, el quadre es posa vermell per tal de que l'usuari el corregeixi abans de guardar-lo.

I per últim tenim les pestanyes per exportar els logs dels jugadors o importar un CSV amb la llista de jugadors per fer el registre automàticament. Per exportar, es tria quina és l'acció que es vol exportar i al donar-li a "*Create export*" el servidor genera el CSV, el comprimeix si es necessari, i el client se'l descarrega.



I per la pantalla d'importar usuaris:



Com es pot veure, *Twitter Bootsrap* dona un estil decent i bastant estàndard a la web, amb botons que pel seu color ja és fàcil saber quina serà l'acció.

Backend

Com ja he repetit bastants vegades, el servidor el vam programar amb Laravel. Per agafar un ordre, em basaré amb l'estructura de fitxers de la carpeta *app* (on està el codi del servidor en si) per anar explicant com està muntat, per ordre alfabètic. Només comentaré les carpetes rellevants.

La primera carpeta que trobem es config, que conté tots els paràmetres de configuració per a Laravel i els "plugins" que hi hagi, com Eloquent. Possiblement els fitxers més importants són app.php, que defineix tots els serveis de la nostra aplicació, i database.php que defineix la configuració per conectar-se a la base de dades.

A la carpeta *controllers* hi trobem tots els controladors, els mètodes que Laravel crida quan un usuari demana alguna acció mitjançant la API, i els tenim separats en dos carpetes, una per la

API i l'altre pel frontend (a la carpeta *website*). La responsabilitat d'aquestes classes és de validar la informació que proporciona l'usuari, fer les crides als serveis necessaris per tal de dur a terme l'acció i servir un resultat de l'operació al usuari que l'ha demanat. En el cas de la API, tenim un controlador per a cada grup d'accions que estiguin relacionades (per exemple, totes les operacions que estiguin relacionades amb les piràmides com contractar minions, expulsar-ne, crear una nova piràmide, etc. van a una mateixa classe controlador), i per a cada acció individual tenim un mètode que la serveix.

La següent carpeta és per l'estructura de la base de dades. Dins s'hi troben els fitxers de migració per aplicar l'estructura en altres servidors a l'hora d'instalar-lo. Cada fitxer conté els canvis que es necessitaven durant l'etapa de producció.

La carpeta *events* conté el codi que fa servir els events de Eloquent per tal de que al actualitzar el valor de les taules referenciades pels logs, faci la còpia (veure apartat de Bases de dades). Bàsicament el que fa és cridar directament el procediment de MySQL quan un dels models de Eloquent s'està modificant (*updating*), passant-li el identificador o quan es crea una instància d'un model que té una columna de referència (*Reference*) li assigna una nova referència única.

A la carpeta *hrmmanager* hi ha tots els models i serveis per tal de dur a terme les accions necessàries o demanades pel jugador o administrador. També estan separats per carpetes, on cada carpeta conté els serveis relacionats amb el model (p.e. *Game, Minion, MinionInfluencer, Pyramid,* etc.) i que dins d'aquestes també està separat per fitxers que defineixen el model, les accions del repositori (veure patró repositori, en aquest cas la major part de les implementacions fan servir la API de Eloquent) i els serveis definits fent servir el patró façana.

D'aquesta carpeta vull destacar els fitxers de l'arrel que extenen Eloquent. Vam tindre el problema de que Eloquent no suporta claus foranes a columnes que no siguin la clau primària, i nosaltres tenim en el nostre model de dades referències entre objectes mitjançant les columnes *Reference*. Llavors vaig haver de ampliar Eloquent (aprofitant que és de codi obert) per poder fer les relacions "pertany a molts" (*Belongs to many*), "pertany a" (*Belongs to*) i "té molts" (*Has many*) i extendre la classe que representa un Model base de Eloquent per tal de que els models tinguessin accés a aquestes relacions amb una API igual a la de Eloquent (afegint un mètode per a cada tipus de relació, amb el nom en anglès).

Vegem un model d'exemple per comentar la API de Eloquent. La classe de Model de minion, es (abreviant algunes parts, per no fer-ho massa llarg):

```
class Minion extends Model {
```

```
	public $table = "minions";
	public $visible = array(
		'id',
		[...]
		'spawn_z'
	);

	public $fillable = array(
		'business_excellence',
		[..]
		'spawn_z'
	);

	public $softDelete = true;

	public static $rules = array(
		'name' => 'required',
		[..]
		'productivity' => 'numeric|min:0|max:100',
	);

	public function influencers(){
		return $this->hasMany(
			"\hrmmanager\MinionInfluencer\Model\Influencing"
		);
	}

	public function referencers(){
		return $this->hasManyNoPK(
			"\hrmmanager\Logging\Model\ActionLog",
			"minion_ref", "reference"
		);
	}

	public static function getNextRef(){
		$maxRef = Minion::withTrashed()->max("reference");
		return $maxRef + 1;
	}

	public function delete(){
		$influencers = $this->influencers()->get();
		foreach ($influencers as $influencer) {
			$influencer->delete();
		}

		parent::delete();
	}
}
```

Primer li diem a Eloquent quina és la taula d'aquest model, per tal de que faci el mapeig. Les següents variables les tenim com a configuració d'aquest model, definint quines son les columnes que són visibles (per exemple, a un usuari voldriem mostrar el seu nom d'usuari,

però no la seva contrassenya), els modificables -automàticament- (Eloquent permet modificar el Model passant-li directament un *array* amb claus i valors. Per seguretat, es pot especificar quines columnes són les que es poden modificar mitjançant aquest mètode). Amb $softDelete diem que quan donem l'ordre d'eliminar una instància d'aquest model no l'elimini de la base de dades, sinó que només el marqui com a eliminat (perquè el preu de les dades és realment baix i permet recuperar objectes que s'hagin eliminat per equivocació o poder treure estadístiques més endavant), i $rules és un *array* que defineix les normes de validació de l'entrada del usuari per Eloquent.

Després venen les funcions que defineixen les relacions amb els altres models, per exemple, si volem accedir als influenciadors del minion, i fem servir les funcions de l'API d'eloquent (i de fet, en aquest cas fem servir també la funció que vaig fer per relacions sense claus primàries). *GetNextRef* serveix per obtindre el següent número únic per a assignar una nova referència, buscant la actual màxima referència, tenint en compte les eliminades (*::withTrashed*) i incrementant-lo.

Per últim, extenem el mètode delete per tal de que quan eliminem el minion també s'eliminin els seus influenciadors.

La següent carpeta relevant és la de tests. Aquí vam crear tots els tests unitaris per a cada un dels controladors del servidor. Laravel fa servir l'API de PHPUnit per crear els tests unitaris. En cada classe de Test tenim unes funcions que cada una defineix un cas de test, i per a cada test es crida una funció que deixa una base de dades mapejada a memòria (per tal de que siguin més ràpides les operacions) buida, així no tenim "contaminació" entre casos de test. Normalment abans d'executar cada test, creem les dades inicials de la base de dades que cal tindre per aquell test, li demanem a Laravel que executi una entrada de la API i comprovem que els resultats són els correctes mitjançant *asserts*.

Llavors, els fitxers que queden a la carpeta arrel "*app*" són *filters.php* i *routes.php*. A *filters* hi tenim declarats els filtres per l'enrutament per comprovar que l'usuari estigui autentificat i, en el cas del accés per l'API, que la signatura HMAC sigui correcte (veure Comunicació en Xarxa - Seguretat).

Per últim, en el fitxer *routes.php* tenim definides tots els enrutaments, tant del panell d'administració com per la API (que hem definit al apartat de Comunicació en Xarxa). Als que fa falta autorització, fem servir els filtres que hem definit al fitxer *filters.php*.

Planificació i cost

Planificació

El diagrama de Gantt de la planificació inicial es pot trobar en les següents pàgines després d'aquest apartat. A continuació comentaré els trets més importants del diagrama.

Cal tindre en compte però, que donada la naturalesa del projecte i del mètode aplicat (Scrum), aquesta planificació inicial ha estat aplicada de forma molt aproximada. Recordem que Scrum és un mètode que en el nostre cas feiem iteracions de 2 setmanes, amb lo que els canvis eren pràcticament constants. Però això ens ha permès fer un videojoc que satisfà més el que realment volia el client, que fent servir altres mètodes (de cascada, per exemple) molt possiblement no ho haguéssim aconseguit.

Pràcticament el primer mes el vam dedicar a recerca i anàlisi. Després de buscar informació sobre jocs seriosos i les seves aplicacions, havíem de separar els treballadors de l'empresa en dos grups de treball per dos projectes diferents. Un d'ells està basat en que l'any anterior l'empresa va produïr però que per falta de temps i/o personal va quedar molt poc madur. El seu objectiu era millorar les capacitats matemàtiques (sobretot de fraccions) dels estudiants de magisteri (Pabo), doncs els graduats tenen greus problemes a l'hora d'entrar a oposicions en els apartats de matemàtiques. Es tractava llavors de veure què fallava en aquell videojoc, replantejar-lo i desenvolupar-lo de nou. L'altre projecte era el que he descrit en aquesta memòria.

Cada un dels grups va estar buscant informació de cada projecte (tant l'informació bàsica com els requisits) i va acabar desenvolupant un concepte, que es va portar a presentar a cada client. Basada en l'opinió de cada client per a cada concepte, vam assignar que el nostre grup (anomenat Flying Panda Games) faria el projecte de Saxion i l'altre grup (anomenat Game2Win) faria el projecte de Pabo.

La següent fase l'hem anomenada de *Setup*. En aquesta fase es prepara tot l'entorn de treball per a dur a terme el desenvolupament, tals com escollir els rols del grup, de Scrum, planificar totes les tasques necessàries a fer cada una amb la seva prioritat, i elaborar la planificació del projecte.

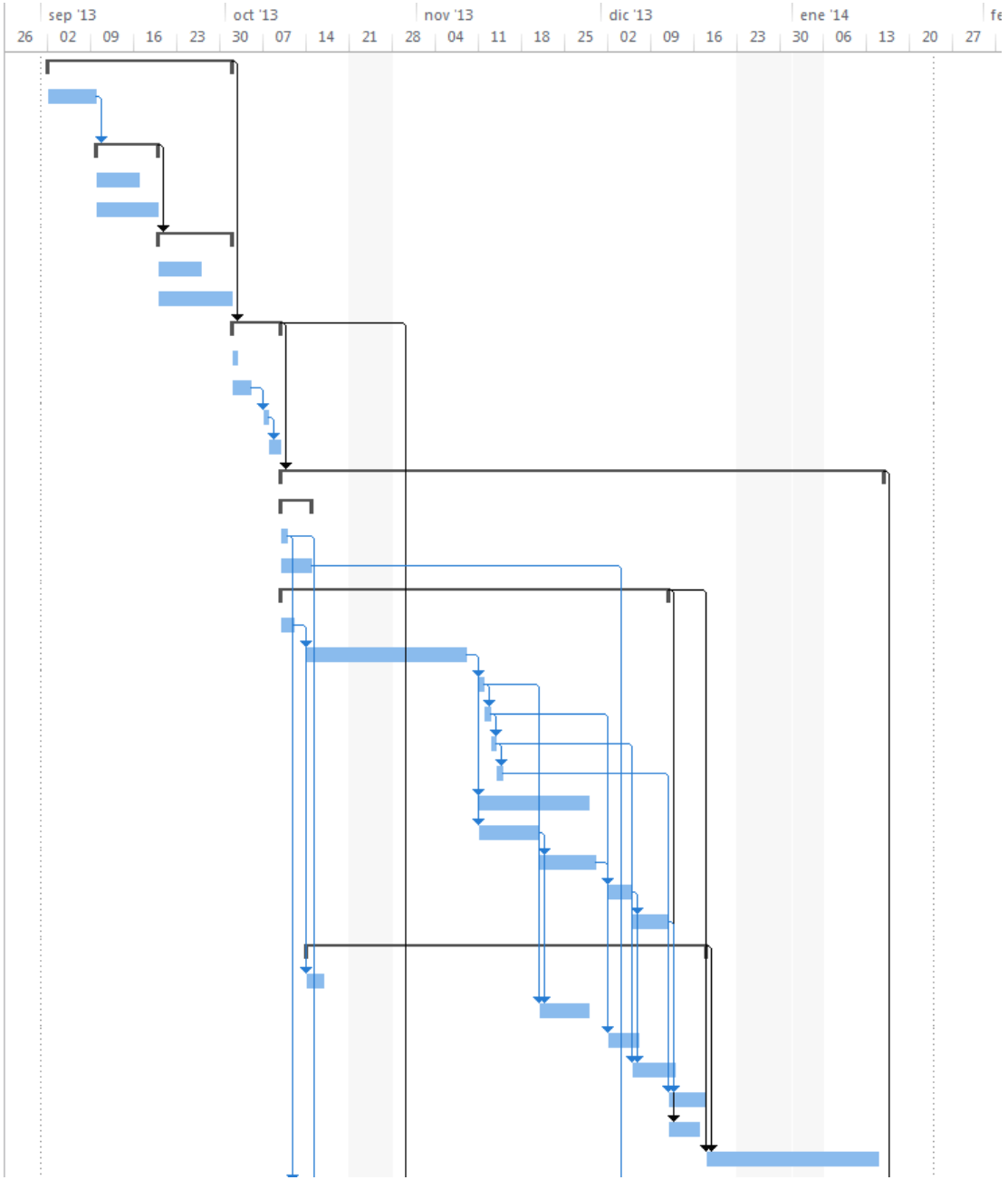
Llavors vam començar directament pel joc client, deixant el servidor per més tard, ja que volíem tindre el més feedback possible del client el més aviat millor, sinó els canvis que s'haurien d'haver fet haguessin estat més grans. El detall de la planificació es pot veure

directament al Gantt, però jo personalment pel principi em vaig dedicar a parts bàsiques d'arquitectura i components reutilitzables (com el sistema de finestres, el component d'entrada de nombres amb botons per augmentar/disminuïr, un altre component per representar-hi llistes, basada en el ListView d'Android, etc), i una primera versió de les interfícies que teníem clar que hi serien, com la de gestió de Minions o per reclutar-ne.

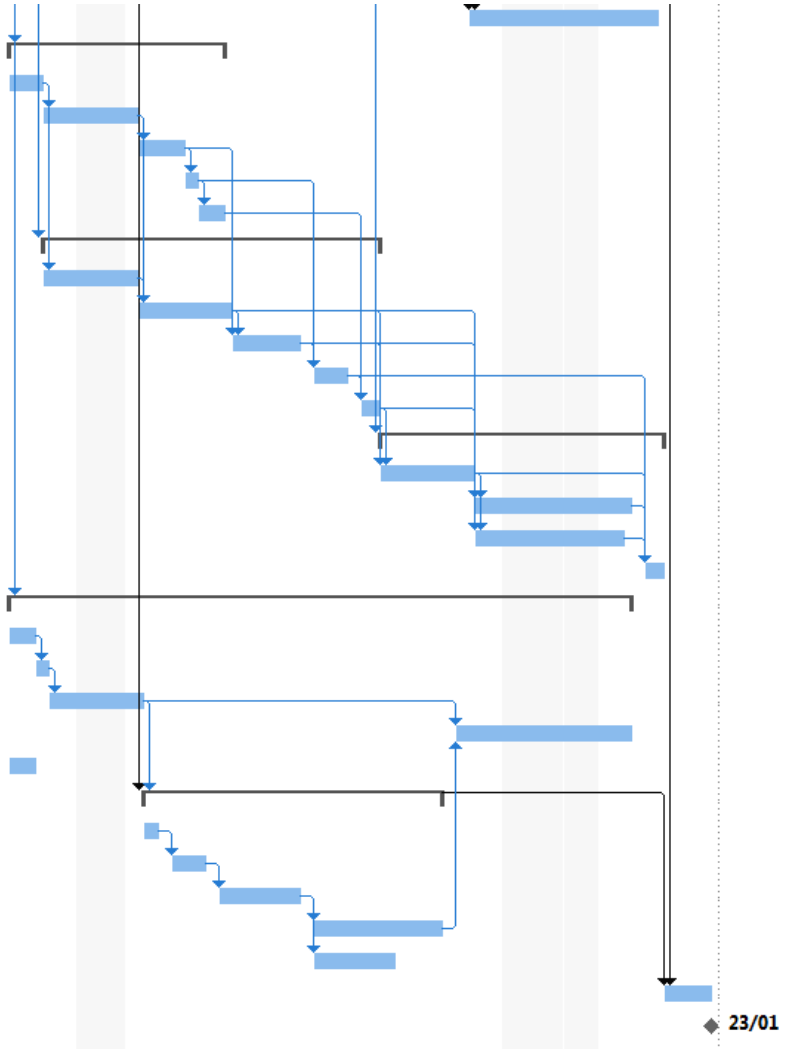
Una vegada ja vam rebre els primers canvis, ens vam veure en cor de començar a programar el servidor. Tal i com he dit anteriorment, el servidor el vam fer entre dos persones, en Harm-Jan es va dedicar sobretot a la part del panell d'administració i jo a la part de la API i de la comunicació amb el client.

L'última fase abans de l'entrega del producte, eren proves (amb gent real, i veient quines parts eren les que calia millorar), arreglar errors i poliment (acabar d'ajustar paràmetres o fer petites millores a les interfícies).

	Nombre de tarea	Duraci	Comienzo	Fin	Predeceso
1	**Seleccio, Recerca i Analisi**	**22 días**	**lun 02/09/13**	**mar 01/10/13**	
2	Recerca sobre serious games	6 días	lun 02/09/13	lun 09/09/13	
3	**Pabo**	**8 días**	**mar 10/09/13**	**jue 19/09/13**	**2**
4	Recerca projecte	5 días	mar 10/09/13	lun 16/09/13	
5	Concepte projecte	8 días	mar 10/09/13	jue 19/09/13	
6	**HRM**	**8 días**	**vie 20/09/13**	**mar 01/10/13**	**3**
7	Recerca projecte	5 días	vie 20/09/13	jue 26/09/13	
8	Concepte projecte	8 días	vie 20/09/13	mar 01/10/13	
9	**Setup**	**6 días**	**mié 02/10/13**	**mié 09/10/13**	**1**
10	Escollir rols del grup	1 día	mié 02/10/13	mié 02/10/13	
11	Definir les tasques a fer	3 días	mié 02/10/13	vie 04/10/13	
12	Prioritat de les tasques	1 día	lun 07/10/13	lun 07/10/13	11
13	Entorn de treball	2 días	mar 08/10/13	mié 09/10/13	12
14	**Videojoc**	**55 días**	**jue 10/10/13**	**mié 15/01/14**	**9**
15	**Utilitats**	**3 días**	**jue 10/10/13**	**lun 14/10/13**	
16	IoC Container	1 día	jue 10/10/13	jue 10/10/13	
17	Sistema de Finestres	3 días	jue 10/10/13	lun 14/10/13	
18	**Models gràfics**	**40 días**	**jue 10/10/13**	**mié 11/12/13**	
19	Minion bàsic	2 días	jue 10/10/13	vie 11/10/13	
20	Minion final	15 días	lun 14/10/13	vie 08/11/13	19
21	Minion Packaging	1 día	lun 11/11/13	lun 11/11/13	20
22	Minion Màrketing	1 día	mar 12/11/13	mar 12/11/13	21
23	Minion Cientific	1 día	mié 13/11/13	mié 13/11/13	22
24	Minion Líder	1 día	jue 14/11/13	jue 14/11/13	23
25	Animacions	14 días	lun 11/11/13	jue 28/11/13	20
26	Primera planta	8 días	lun 11/11/13	mié 20/11/13	20
27	Segona planta	7 días	jue 21/11/13	vie 29/11/13	26
28	Tercera planta	4 días	lun 02/12/13	jue 05/12/13	27
29	Quarta planta	4 días	vie 06/12/13	mié 11/12/13	28
30	**Escena**	**42 días**	**lun 14/10/13**	**mar 17/12/13**	
31	Escena bàsica	3 días	lun 14/10/13	mié 16/10/13	19
32	Primera planta	6 días	jue 21/11/13	jue 28/11/13	21;26
33	Segona planta	5 días	lun 02/12/13	vie 06/12/13	22;27
34	Tercera planta	5 días	vie 06/12/13	jue 12/12/13	23;28
35	Quarta planta	4 días	jue 12/12/13	mar 17/12/13	24;29
36	Interactivitat	3 días	jue 12/12/13	lun 16/12/13	18
37	Sons	10 días	mié 18/12/13	mar 14/01/14	18;30



37	Sons	10 días	mié 18/12/13	mar 14/01/14	18;30
38	**Data Model**	**17 días**	**vie 11/10/13**	**lun 11/11/13**	**16**
39	Piràmide	3 días	vie 11/10/13	mar 15/10/13	
40	Minions	5 días	mié 16/10/13	mar 29/10/13	39
41	Influenciadors	5 días	mié 30/10/13	mar 05/11/13	40
42	Correus	2 días	mié 06/11/13	jue 07/11/13	41
43	Recursos	2 días	vie 08/11/13	lun 11/11/13	42
44	**Business Model**	**31 días**	**mié 16/10/13**	**mié 04/12/13**	**16**
45	Piràmide	5 días	mié 16/10/13	mar 29/10/13	39
46	Minions	10 días	mié 30/10/13	mar 12/11/13	40;45
47	Influenciadors	8 días	mié 13/11/13	vie 22/11/13	41;46
48	Correus	5 días	lun 25/11/13	vie 29/11/13	42;47
49	Recursos	3 días	lun 02/12/13	mié 04/12/13	43;48
50	**Interfície**	**20 días**	**jue 05/12/13**	**mié 15/01/14**	**17**
51	MinionManager	10 días	jue 05/12/13	mié 18/12/13	46;47;49
52	ContractarMinions	7 días	jue 19/12/13	vie 10/01/14	46;49;51
53	Influenciadors	6 días	jue 19/12/13	jue 09/01/14	47;49;51
54	Correu	3 días	lun 13/01/14	mié 15/01/14	48;51;52;
55	**Comunicació**	**51 días**	**vie 11/10/13**	**vie 10/01/14**	**16**
56	Comunicació bàsica	2 días	vie 11/10/13	lun 14/10/13	
57	Protocol	2 días	mar 15/10/13	mié 16/10/13	56
58	Seguretat	5 días	jue 17/10/13	mié 30/10/13	57
59	API	10 días	lun 16/12/13	vie 10/01/14	65;58
60	Crides asíncrones	2 días	vie 11/10/13	lun 14/10/13	
61	**Servidor**	**32 días**	**jue 31/10/13**	**vie 13/12/13**	**9;58**
62	Configuració	2 días	jue 31/10/13	vie 01/11/13	
63	Bases de dades	5 días	lun 04/11/13	vie 08/11/13	62
64	Data Model	10 días	lun 11/11/13	vie 22/11/13	63
65	API	15 días	lun 25/11/13	vie 13/12/13	64
66	Panell d'administració	10 días	lun 25/11/13	vie 06/12/13	64
67	Proves, Bugs i Poliment	5 días	jue 16/01/14	mié 22/01/14	14;61
68	Presentació i lliurament	0 días	jue 23/01/14	jue 23/01/14	



Costos i recursos

Per dur a terme aquest projecte durant el període de mig any s'han necessitat tota una sèrie recursos. Com és obvi, no només recursos materials, com poden ser màquines, oficines, etc. sinó també personals, doncs es requereix un cert personal amb les habilitats necessàries per poder desenvolupar el projecte. En aquest apartat les enumero, i n'estimaré un cost.

Comencem per les màquines que necessitem per desenvolupar / provar el projecte:

Quantitat	Objecte	Quantitat	Objecte
7x	Ordinadors	7x	Pantalles
1x	Macbook	1x	LG Nexus 4 (per proves)
1x	Samsung Galaxy Tab 3 (per proves)	1x	iPad mini (per proves)
1x	Canó projector	1x	VPS per l'allotjament

Els ordinadors han de ser suficientment potents com per poder desenvolupar en Unity, 3ds max i Blender còmodament.

D'altra banda també necessitem una sèrie de llicències de software per al desenvolupament / publicació:

Quantitat	Objecte
6x	Llicències Unity3D
2x	Llicències 3ds Max
2x	Llicències Adobe Photoshop
8x	Llicències SourceTree
1x	Bitbucket/Github
1x	Llicència Microsoft Office
1x	Nom de domini

I per últim, necessitem una oficina suficientment gran i un equip de 8 persones equilibrat que treballin i s'autoorganitzin mitjançant la metodologia Scrum. Llavors 1 d'aquests haurà d'agafar el rol de *Scrum master*, per assegurar-se de que el mètode s'aplica correctament i confirmar el progrés de cada sprint i un *product owner*, que serà el que decidirà majoritàriament la importància de les coses que cal fer per implementar el projecte. De totes formes, aquests membres no han d'agafar aquest rol exclusivament, sinó que com els altres membres del

equip, també cal que agafin un rol d'entre desenvolupador del software (que se'n necessiten 5), un artista/dissenyador 3D, un de *game design* i un de màrqueting i comunicació.

Havent fixat els recursos necessaris podem calcular el cost del projecte. Per fer-lo amb rigor, assumirem que els que hem treballat en desenvolupar aquest projecte hem cobrat un sou i que hem tingut els diners suficients per a poder pagar tots els costos.

Un sou mitjà als Països Baixos, on s'ha desenvolupat el projecte, és d'uns 2.200€ al mes. Si multipliquem els 8 treballadors pels 6 mesos, ens dona un total de **105.600€**

Costos d'equipament de màquines:

Quantitat	Objecte	Cost per unitat (o mes)	Cost Total (o 6 mesos)
7x	Ordinadors	499€	3.493€
7x	Pantalles	144€	1.008€
1x	Samsung Galaxy Tab 3	329€	329€
1x	Canó projector	434€	434€
1x	Macbook	1.229€	1.229€
1x	LG Nexus 4 (per proves)	349€	349€
1x	iPad mini (per proves)	287,98€	287,98€
1x	VPS per l'allotjament	25€	150€
		Total	**7.279,98€**

Costos d'equipament de llicències:

Quantitat	Objecte	Cost per unitat (o mes)	Cost Total (o 6 mesos)
6x	Llicències Unity3D	3.324,47€	19.946,82€
2x	Llicències 3ds Max	144,06€	3.457,44
2x	Llicències Adobe Photoshop	12,29€	589,92€
8x	Llicències SourceTree	0€	0€
1x	Github	18,47€	110,82€
1x	Llicència Microsoft Office	458,59€	458,59€
1x	Nom de domini	12€	12€
		Total	**24.575,59€**

D'altra banda, l'edifici on estàvem treballant cobrava un lloguer de 1.468€ al mes per a cada oficina. Multiplicant-ho per 6 mesos, ens dona un total de **8.808€**.

I per últim, el que ha elaborat el business plan ha dit que fa falta crear publicitat en forma de flyers, pòsters, pel Facebook i LinkedIn (en el pla comenta que Twitter hagués estat bo, però requereixen un mínim de 5.000€ al mes, que surt massa car pel nostre projecte). Ha calculat un total de 5.000 flyers Din-A5, que a un preu de 13 cèntims per unitat acaba costant 650€ i 2 posters, que a un preu de 129€ acaba costant 258€. Pel que fa a les xarxes socials, tant pel Facebook com pel LinkedIn ha determinat de reservar un pressupost de 741.22€ repartit equitativament en els modes de pagament CPM i CPC. Si representem aquests valors en una taula ens queda:

Suport	Cost
Flyers A5	650€
Posters	258€
Facebook	741.22€
LinkedIn	741.22€
Total	**2.390,44€**

Resumint tots els costos, i separant-los entre variables i fixos:

Costos fixes		Costos variables		**Cost total**
Lloguer	8.808€	Publicitat	2.390,44€	
Salaris	105.600€			
Maquinari	7.279,98€			
Llicències	24.575,59€			
Total fixes	146.263,57€	Total variables	2.390,44€	**148.654.01€**

Conclusions

Si fem un repàs als objectius principals com als secundaris, podem veure que aquest projecte els ha complert. Hem creat un joc que ha incorporat tot el contingut d'administració de recursos humans que volia el client (doncs s'ha quedat més que satisfet), amb una potent arquitectura distribuïda, que permet monitoritzar les accions que fan els jugadors, així com poder configurar i distrbiuïr fàcilment el joc.

Una demostració de l'èxit del nostre projecte va ser que va tindre un cert impacte sobre els medis de comunicació, al sortir en dos dels medis importants dels Països Baixos: "RTV Oost" i "Nationale Onderwijsgids" (veure referències).

Com a autocrítica, el joc ha quedat una mica més seriós del que volíem en un principi. És un dels punts que al entrar en aquest gènere de videojocs cal tindre molt en compte, el balanç entre com de "seriós" el videojoc i com pot arribar a enganxar realment als jugadors, i moltes vegades es tracta de detalls molt petits. De fet, voldria fer referència a una conferència d'un congrés que vam assistir sobre desenvolupament de videojocs (Control conference) on en *Jan Willem Nijmank* va fer una presentació sobre *"Game Feel"* on partint d'un joc molt bàsic (i aborrit) afegint petits detalls, com vibració a la pantalla al disparar acaba fent un joc que tot i que segueix sent bàsic és divertit i et donen ganes de seguir jugant-hi. Doncs al nostre joc crec que li falta una mica més d'animació, de sons, de petits detalls que facin que sigui un joc millor.

De totes formes estic molt satisfet amb el treball que he aportat en aquest projecte. Acabant la carrera tinc tots els coneixements teòrics, i tot i que es fan pràctiques a les sessions de laboratori de la facultat, mai l'havia aplicat en un ambient real de treball, on has de treballar amb un grup important de gent per tal de que tot acabi funcionat, i començant el desenvolupament d'un software des del principi del tot (de anàlisi) fins al final, amb la prova i l'aplicació. M'han estat de especial ajuda les assignatures de enginyeria del software, per aplicar els patrons de disseny correctament, "Disseny de sistemes basats en el web" per dissenyar i implementar l'API, a més d'un extra en introducció al Scrum i Criptografia, per decidir el sistema criptogràfic de comunicació entre el client i el servidor, i per suposat les assignatures de programació. Haver participat en aquest projecte m'ha donat una visió professional de tot el que he après durant la carrera, que a més m'ha proporcionat un experiència molt valuosa per entrar al món laboral.

Referències

Llibres

Kim S. Cameron i Robert E. Quinn (2006), *Diagnosing and Changing Organizational Culture*.

Schell, Jesse (2008). *The Art of Game Design, A Book of Lenses*.

K. Schwaber i J. Sutherland (2013), *The Scrum Guide*.

Artícles de premsa / Congressos

http://controlconference.com/

http://www.nationaleonderwijsgids.nl/hbo/nieuws/21366-windesheim-en-saxion-bundelen-krachten-voor-unieke-hrm-game.html

http://www.rtvoost.nl/nieuws/default.aspx?nid=181042

Artícles i ajudes de programació

http://www.beamartyr.net/articles/adobepolicyfileserver.html

http://laravel.com/

http://unity3d.com/

Sebastian Bergmann (2014), *PHPUnit Manual*