Desenvolupament d'una eina de planificació col·laborativa de *releases* orientada a comunitats àgils.

Pau Campaña Soler

19 de juny de 2018

Director del projecte: Carles Farre

Codirector del projecte: David Ameller

Resum

En aquest projecte de final de carrera s'ha millorat una eina ja existent anomenada Replan. Replan és una aplicació web que ajuda en la planificació i gestió d'entregues de software, principalment pensada per projectes que adopten metodologies àgils.

L'objectiu d'aquest treball ha sigut adaptar l'aplicació perquè sigui multiusuari, i fer que cada usuari tingui un contingut personalitzat. Per fer-ho, per un lloc s'ha modificat la pàgina web existent que s'havia realitzat amb Angular 2, i per altre s'ha creat una API que permet fer tota la gestió dels usuaris. Per fer la API, les tecnologies que s'han utilitzat han sigut NodeJS i MongoDB.

El resultat ha estat que l'aplicació ha mantingut totes les funcionalitats anteriors i ha afegit contingut personalitzat a l'usuari. L'usuari, a l'entrar les seves credencials veu només aquella informació que li interessa, com són els projectes dels quals és membre i les seves notificacions. També pot modificar els seus atributs, modificar la contrasenya i pot continuar fent les mateixes coses que feia dins un projecte amb la versió anterior de Replan.

En aquest treball s'ha documentat tota la feina realitzada i el seu estudi corresponent com són l'abast del projecte, les fases, el pressupost, els requisits o la implementació, entre d'altres.

Resumen

En este proyecto de final de carrera se ha mejorado una herramienta ya existente llamada Replan. Replan es una aplicación web que ayuda en la planificación y gestión de entregas software, principalmente pensada para proyectos que adoptan metodologías ágiles.

El objetivo de este trabajo ha sido adaptar la aplicación para que sea multiusuario, y hacer que cada usuario tenga un contenido personalizado. Para ello, por un lado se ha modificado la página web existente que se había realizado con Angular 2, y por otro se ha creado una API que permite hacer toda la gestión de los usuarios. Para hacer la API, las tecnologías que se han utilizado han sido NodeJS y MongoDB.

El resultado ha sido que la aplicación ha mantenido todas las funcionalidades anteriores y ha añadido contenido personalizado al usuario. El usuario, al entrar sus credenciales ve sólo aquella información que le interesa, como son los proyectos de los que es miembro y sus notificaciones. También puede modificar sus atributos, modificar la contraseña y puede continuar haciendo las mismas cosas que hacía en un proyecto con la versión anterior de Replan.

En este trabajo se ha documentado todo el trabajo realizado y su estudio correspondiente como son el alcance del proyecto, las fases, el presupuesto, los requisitos o la implementación, entre otros.

Abstract

In this final degree project, I have improved an existing tool called Replan. Replan is a web application that helps in the planning and management of software deliveries, mainly designed for projects that adopt agile methodologies.

The objective of this work has been to adapt the application to be multi-user, and to make each user has a personalized content. To do this, on the one hand I have modified the existing website that had been done with Angular 2, and on the other hand I have created an API that allows managing all users. To develop the API, the technologies that have been used have been NodeJS and MongoDB.

The result has been that the application has maintained all the previous functionalities and has added personalized content to the user. The user, when entering his credentials can only see the information that interests him, such as the projects of which he is a member and his notifications. He can also modify his attributes, modify the password and he can continue doing the same things that he did in a project with the previous version of Replan.

In this project we have documented all the work done and its corresponding study such as the scope of the project, the phases, the budget, the requirements or the implementation, among others.

Índex

1 Context

1.1 Introducció

Aquest projecte està definit com un Treball de Final de Grau d'Enginyeria Informàtica, en l'especialitat d'Enginyeria del Software de la Facultat d'Informàtica de Barcelona (Universitat Politècnica de Catalunya). Es tracta d'un projecte de modalitat A on es millorarà un aplicatiu web ja existent anomenat Replan.

Replan forma part de SUPERSEDE, un projecte europeu on col·laboren diferents empreses d'arreu del món. És una aplicació web per a la planificació i gestió d'entregues de software, principalment pensada per projectes que adopten metodologies àgils. És una eina que a partir d'uns recursos troba una òptima combinació de tasques a implementar per vàries entregues de software.

1.2 Formulació del problema

Tot i que aquesta aplicació ja podria ser d'utilitzada per alguns equips de treball, aquesta té un principal dèficit, i és que aquesta mostra un mateix contingut per tots els membres de l'equip. Per alguns equips potser no és cap problema, però per altres organitzacions on hi hagi una jerarquia en els rols de cada desenvolupador sí. Actualment té els mateixos privilegis tant un empleat com el cap de projecte, ja que el sistema no diferència qui esta fent servir Replan.

Així doncs, l'objectiu del projecte és millorar l'eina actual, fent que passi a ser una aplicació multiusuari, i així cada persona d'un equip podrà veure quines tasques té assignades, en quins projectes participa, i les dates límits de les entregues, entre altres coses. Amb aquesta funcionalitat, aconseguirem que el contingut mostrat sigui personalitzat, i es podrà implementar una jerarquia i uns rols dins d'un projecte a Relpan.

1.3 Actors implicats

En aquest apartat es definiran els diferents stakeholders que tenen relació amb aquest projecte.

- Empreses: Seran els usuaris d'aquesta aplicació. Serà utilitzada per gestionar els seus equips de treball. A partir d'uns treballadors, una certa dedicació de temps i unes habilitats, aquest software proporciona una òptima planificació de com dur a terme les tasques, sempre respectant la dependència entre elles. Dins les empreses, podem trobar dos tipus d'actors que es veuran beneficiats:

    - Cap de projecte: Tindrà la possibilitat de fer tots els canvis que cregui corresponents. Per exemple, podrà canviar l'assignació de tasques, modificar la data d'una entrega o recalcular la repartició de tasques si s'afegeix un nou empleat a l'equip. El software està pensat perquè d'una manera fàcil i intuïtiva, pugui gestionar amb total llibertat tot allò que estigui relacionat amb el projecte.

    - Treballador: Podrà veure tots els projectes en els quals participa, quina és la pròxima entrega que ha de fer, i quines són les seves tasques. En cas que ho necessiti, també podrà modificar el seu perfil, i si ho trobes convenient, sol·licitar al cap de projecte un canvi en la realització de tasques.

- Directors del projecte: En Carles Farre Tost serà el director d'aquest projecte i en David Ameller el codirector . Per una part, ells tindran el rol de product owner o client, i voldran que una sèrie de funcionalitats siguin implementades de forma satisfactòria. Per altra banda, com a responsables d'aquest treball, supervisaran la feina realitzada per tal que aquest projecte es dugui a terme satisfactòriament i així l'alumne pugui assolir la titulació.

- Desenvolupador del projecte: En Pau Campaña Soler serà l'encarregat de dur a terme aquest projecte que li servirà com a Treball Final de Grau.

1.4 Estat de l'art

1.4.1 Contextualització

Aquest projecte té com a objectiu principal millorar Replan, que és una aplicació que ja existeix. És un sistema que ajuda en la gestió del projecte i serveix per planificar entregues de software a equips de treball, i s'emmarca dins d'un projecte europeu anomenat SUPERSEDE. A partir d'uns determinats treballadors i una sèrie de tasques a realitzar, troba una òptima combinació de com s'han de realitzar les tasques per tal que es puguin dur a terme dins el termini de la data de l'entrega. Per fer aquesta repartició de tasques el programa té en compte molts factors com són les habilitats que cada empleat té i el temps que té disponible, quines habilitats requereixen les tasques per ser implementades, el nivell d'esforç de cada tasca, les dependències entre les tasques i les dates límit. Un cop s'ha calculat com s'han de repartir les tasques, aquesta repartició es pot canviar sempre que es respecti la dependència entre tasques. En el cas que hi haguessin nous treballadors o algun dels existents adquirís alguna nova habilitat, es podria tornar a calcular una nova repartició de les tasques si es cregues convenient.

En aquest projecte es vol millorar aquesta eina fent que sigui multiusuari. Actualment no hi ha cap registre ni s'ha de proporcionar cap identificació a l'entrar, pel que la web i la informació que es mostra és la mateixa independentment de qui sigui la persona que l'utilitzi. Amb aquesta millora s'aconseguirà que el software sigui més personal, ja que l'usuari veurà només aquells projectes amb els quals està involucrat. També permetrà definir rols i privilegis, aspecte que fins ara era inexistent.

1.4.2 Gestor de projectes

Fa temps que els equips de treball han vist que els projectes no poden dependre de la seva documentació i memòria per funcionar, ja que aquests no sempre estan actualitzats, o si ho estan, es perd molt temps canviant-lo perquè la informació sigui correcte. Així doncs, el que es fa és utilitzar gestors de projectes software per poder organitzar la feina. Una de les principals avantatges d'utilitzar aquest tipus de software en comptes de mantenir un document escrit és la canviabilitat. És molt més ràpid modificar el pressupost, el temps del projecte o les dates d'entrega en aquest tipus de software que en una documentació escrita.

Els gestors de projecte són sistemes en línia per treballar i col·laborar en projectes en temps real. Permet organitzar la informació del projecte perquè tots els membres puguin veure els detalls del projecte actualitzats, i serveixen, com el seu nom indica, per gestionar projectes software. Això vol dir que permeten crear una data d'inici, una data final i entregues, entre altres coses.

1.4.3 Estudi del mercat

Per situar el projecte dins d'un marc de solucions existents, cal fer un estudi de mercat per veure quines són les alternatives a Replan. S'han estudiat tant eines destinades a la gestió del software com eines que tenen com a objectiu ajudar a la gestió, sense ser específicament pensades per projectes software. Actualment hi ha moltes eines que ajuden a la gestió de projectes pel que només estudiarem algunes d'elles.

Trello

Trello[1] és una eina de gestió de projectes. Pot ser utilitzada tant en grans equips com per una sola persona, i serveix per organitzar projectes. És una pàgina web que conté llistes de manera que es pot apreciar amb facilitat tot el que hi ha al projecte. És fàcil de fer servir, ja que pots afegir, esborrar, modificar o canvia un element de lloc amb rapidesa i d'una manera intuïtiva. Permet tenir diferents escriptoris, pel que és fàcil separar projectes. És una eina molt completa però no pensada específicament per la gestió de projectes software, sinó projectes en general. Això fa que en alguns aspectes sigui incompleta.

Slack

Slack[2] és un sistema de missatgeria en temps real per la comunicació entre equips que inclou tots els mitjans de comunicació un mateix lloc. Té la característica que integra altres eines com Dropbox o Google Drive. Les empreses ho utilitzen com a canal per parlar i discutir sobre el projecte en general o algun aspecte en particular. Es poden definir diferents canals i així tractar de temes diferents a cada un. És una eina ideal per la comunicació en grups de treball, i per mantenir relació client-servidor. Tot i això, no és una eina pensada per gestionar projectes software en la seva totalitat.

Jira

Jira[3] és una eina per l'administració de tasques en un projecte, el seguiment d'incidències i per a la gestió operativa de projectes. Incorpora funcions per l'organització del flux de treball, pel que pot ser utilitzada per la gestió i millores de processos. També s'utilitza per la gestió del desenvolupament de software, i té en compte la gestió dels requisits, l'estat del desenvolupament i la gestió d'errors. Al contrari que les eines mencionades anteriorment, aquesta sí que està pensada per projectes software.

Redmine

Redmine[4] és una eina basada en la gestió de projectes. Utilitza gràfics de Gantt i calendaris per ajudar a la representació visual dels projectes i els seus terminis. És compatible amb múltiples projectes. Inclou un sistema de seguiment d'incidències amb seguiments d'errors. És una eina pensada específicament per la gestió de projectes software.

Asana

Asana[5] és una eina per gestionar els projectes i les seves tasques, i permet als membres d'un equip compartir, planificar i organitzar el progrés de les diferents tasques en les quals cada membre treballa. És una aplicació web on destaca una senzilla i usable interfície. Una de les seves característiques és que s'integra amb moltes altres eines com són Github o Dropbox.

1.4.4 Conclusions de l'estudi del mercat

Actualment hi ha moltes eines que poden ajudar a gestionar un equip de treball, però com hem vist, no totes estan pensades per projectes software. Si comparem Replan amb les altres aplicacions, veiem que totes ofereixen la funcionalitat de multiusuari, on el contingut que s'ofereix és personalitzat. Actualment Replan no ho ofereix, però és l'objectiu d'aquest treball que ho acabi fent.

Si ho comparem amb eines com Trello, veiem que tot i que aquestes eines ofereixen més funcionalitats, no s'ajusta al que una gestió d'un projecte software necessita. No està pensat

per definir tasques amb uns requisits i dependència entre elles, pel que és difícil utilitzar-les, sobretot en equips grans, per aquest propòsit.

Pel que fa a les altres, veiem que sí que s'ajusta més al que una gestió de projectes software necessita. Comparant l'actual sistema de Replan amb els altres softwares, veiem que aquests són més evolucionats, i ofereixen més funcionalitats. Per altra banda, un punt negatiu que tenen és que molts d'ells són gratuïts fins a arribar a un cert límit de membres, pel que equips grans han de pagar per utilitzar el seu sistema.
Replan està pensat per ser una alternativa a aquestes eines, pel que les empreses puguin utilitzar-la si creuen que s'adapta bé a la seva manera de treballar. Els grans punts forts de Replan són la funcionalitat de calcular automàticament un òptim repartiment de tasques (moltes eines actuals no ho tenen), i que té l'objectiu de ser una aplicació totalment gratuïta.

1.4.5 Replan

Com hem vist, Replan pot tenir lloc dins el mercat actual. Com que Replan és una eina que ja existeix, hem de valorar si val la pena continuar utilitzant les mateixes tecnologies fins ara utilitzades o si s'ha de canviar. Dins el projecte trobem dos grans tasques, la implementació d'una API que permeti la gestió multiusuari, i l'aplicació web.
L'aplicació web ja existeix, i s'ha desenvolupat en Angular 2. En l'actualitat, hi ha diversos frameworks popular pel desenvolupament d'aplicacions web com són React, Vue o el mateix Angular 2. Com que Angular 2 es tracta d'un framework encara molt utilitzat per la comunitat, i que ofereix un bon rendiment, varietat d'exemples i documentació, no s'ha trobat necessari el canvi de tecnologia. Aquest canvi implicaria una gran dedicació de temps tornant a construir la web amb el nou framework. En canvi, utilitzant Angular 2 podem continuar l'aplicació i dedicar aquest temps a implementar noves funcionalitats.
La API es desenvolupa des de zero, així que no tenim cap restricció alhora d'escollir una tecnologia. Hem estudiat diferents tecnologies, i finalment hem escollit NodeJS, MongoDB i ExpressJS principalment per dos motius. Un és la amplia comunitat que té, i les llibreries que ofereix, fet que simplifica les coses alhora d'implementar. L'altre motiu és per poder utilitzar MEAN Stack[6] (acrònim de les inicials de MongoDB, Express.js, Angular 2, i NodeJS). Es tracta d'un conjunt de subsistemes de software que tenen en comú que es programen en JavaScript. Això fa que tot el projecte estigui programat en JavaScript.

2 Abast del projecte

Aquest treball es basa en millorar un sistema que actualment ja existeix anomenat Replan. L'objectiu principal és afegir en el sistema un sistema de registre i autentificació d'usuaris, perquè així la informació que mostri Replan sigui la relacionada amb l'usuari corresponent. També es volen introduir elements de gamificació, per tal de incentivar als usuaris a usar l'aplicació.

Per fer-ho, partirem de l'aplicació actual feta amb Angular 2 com a base. Haurem d'implementar una API per tal de poder gestionar totes les funcionalitats que són necessari per tenir diversos usuaris. Els mètodes d'aquesta API seran cridats des de l'aplicació web.

Per poder analitzar que s'ha assolit aquest objectiu, s'han definit una sèrie d'objectius d'ordre menor. Si aquests objectius es compleixen, es considerarà que l'objectiu principal també es compleix.

2.1 Objectius específics

Registre: Els usuaris s'han de poder registrar a l'aplicació, oferint un correu electrònic, una contrasenya i altra informació necessària. Aquesta informació ha de servir per després poder accedir al sistema.

Entrar al sistema: Un usuari registrat ha de ser capaç d'entrar al sistema, per així poder fer ús de Replan.

Gestió de l'usuari: Un usuari ha de ser capaç de modificar les seves dades i la contrasenya d'accés. També, en el cas que l'usuari no recordes la seva contrasenya, s'ha d'implementar una funcionalitat perquè pugui tornar a entrar al sistema.

Afegir, modificar o eliminar habilitats a un usuari: Un usuari ha de ser capaç d'afegir les habilitats que té, modificar-les, i eliminar-les.

Crear, modificar i eliminar projectes: Un usuari ha de poder crear un projecte, modificar-lo i eliminar-lo.

Unir-se a un projecte: Un usuari ha de poder unir-se a un projecte.

Adaptar funcionalitats de l'aplicació actual: S'haurà de modificar la web actual per tal que les vistes i les funcionalitats ofertes concordin amb el fet que ara l'aplicació sigui multiusuari.

Rebre recompenses per participar: Quan un usuari participi en un projecte i finalitzi tasques, aquest haurà de rebre recompenses que s'aniran acumulant.

2.2 Possibles obstacles

Durant el transcurs d'aquest treball final, pot ser que apareguin alguns problemes. Els principals problemes que ens podem robar són els següents:
Temps limitat: El treball final de grau té una durada establerta d'un quadrimestre, és a dir, 4 mesos. En el cas que durant aquest temps aparegui algun problema rellevant que estanqui o dificulti la realització del projecte, caldrà analitzar-lo amb el director del treball, i si fos necessari, replanificar el treball a realitzar perquè no tingui repercussions majors.
Tecnologies: En aquest treball s'han d'utilitzar diferents tecnologies per complir amb els requisits de frontend i backend. L'avanç en el desenvolupament del sistema estarà relacionat amb el grau d'aprenentatge que es vagi adquirint sobre Angular 2 pel frontend i NodeJS pel backend.
Internet: Per tal que Replan sigui una aplicació útil, és necessari que l'ordinador des del qual s'està accedint a la web tingui connectivitat a Internet.

2.3 Metodologia

Per realitzar aquest treball, s'ha decidit adoptar una metodologia àgil. Moltes de les metodologies àgils estan pensades per equips de treball formats per diversos membres, i no per una sola persona com és el cas d'aquest projecte. Així doncs, no adoptaré una metodologia en si, però sí que aprofitaré alguns conceptes com són els de fer iteracions curtes i tenir feedback constant.

Aprofitant que cada dues setmanes hi ha una reunió amb el director i codirector del projecte, he trobat adient que cada iteració tingues la mateixa durada, és a dir, dues setmanes. Durant cada iteració, s'implementaran noves pantalles, i es faran els canvis corresponents a la API perquè es puguin provar el seu funcionament. Així a cada reunió tant el director com el codirector ho podran provar, i així podran dir si ho troben bé o és necessari fer algun canvi.

2.3.1 Eines de seguiment

Per aquest projecte s'utilitzarà un repositori en línia per poder accedir al codi des de qualsevol dispositiu. Utilitzaré Git com a software de control de versions, i Bitbucket com a repositori tant per la API com per la pàgina web. Es partirà de l'última versió de Replan, i a partir d'allà es començarà a desenvolupar.
Per la comunicació amb el director i codirector del projecte, s'utilitzarà tant les reunions que farem cada dues setmanes com el correu electrònic.

2.3.2 Mètode de validació

Durant la implementació de nous mètodes a la API, s'utilitzarà l'aplicació Postman per tal de comprovar que els mètodes funcionin com s'espera d'ells. Així, en el cas que apareguin erros, podrem diferenciar fàcilment si es troben a la part del frontend o backend.

Per altra banda, s'aniran fent reunions cada dues setmanes amb el director i codirector del projecte. S'utilitzaran per validar la feina feta durant l'última iteració de feina, i discutir qualsevol problema o dubte que pugui sorgir. Aquestes reunions també faran que hi hagi una comunicació constant i així es podran solucionar amb rapidesa els dubtes i problemes que vagin sorgint durant el projecte.

3 Planificació temporal

3.1 Planificació general

3.1.1 Disponibilitat temporal

El projecte té una durada aproximada de quatre mesos i mig, començant el 12 de febrer de 2018 i acabant el 25 de juny, dia de la defensa oral. Tot i això, es procurarà acabar alguns dies abans per així poder fer front a possibles imprevistos que puguin aparèixer durant el transcurs del projecte. El projecte, considerant que té una càrrega de 18 ECTS i que cada ECTS són 30 hores, té un total de 540 hores de dedicació, englobant totes les tasques necessàries per la realització del projecte.

3.1.2 Recursos

Per la realització d'aquests projecte, es necessiten tant recursos personals com materials.

3.1.2.1 Recursos personals

Una persona amb una dedicació de 30 hores setmanals durant tot el projecte. Aquesta persona serà l'encarregada de totes les feines que siguin necessàries per dur a terme el projecte correctament com pot ser planificar, desenvolupar i documentar, entre altres feines.

3.1.2.2 Recursos materials

- **Portàtil Asus ZenBook UX330UA-FC143T amb Windows 10**: Utilitzat per la realització del projecte.
- **Google Drive Docs**: Utilitzat per generar la documentació.
- **Draw.io**: Utilitzat per generar els diagrames de classe.
- **Git**: Utilitzat per mantenir un control de versions.
- **Bitbucket**: Utilitzat com a repositori del codi.
- **Angular 2**: Framework de JavaScript utilitzat pel desenvolupament de la web.
- **NodeJS**: Utilitzat pel desenvolupament de la API.
- **MongoDB**: Base de dades NoSQL utilitzada per la API.
- **Visual Code Studio**: Editor de codi utilitzat per desenvolupar la web.
- **Atom**: Editor de codi utilitzat per desenvolupar la API.

- **Postman**: Utilitzat per testejar les funcionalitats de la API.

3.1.3 Consideracions generals

Cal dir que com que aquest treball es durà a terme per una sola persona, no es podran fer tasques en paral·lel. Això implica que el camí crític és el únic camí possible, i per tant el temps total del projecte equival a la suma total de les hores de totes les tasques. És per aquest motiu que no s'ha considerat necessari fer un diagrama de PERT.
Per totes les histories d'usuari, tant el seu desenvolupament com el testeig es faran paral·lelament. Per aquest motiu, per totes les tasques de desenvolupament que hi hagi, incoluran implicitament els corresponents testeijos.

3.1.4 Valoració d'alternatives

Pot ser que durant el projecte apareguin imprevistos que canviïn la planificació del projecte. És per això que dedicarem un temps abans de l'entrega final per a millores. En el cas que no hi hagui cap imprevist, es dedicarà a millorar Replan. Altrament, si han aparegut imprevistos que han fet anar més lenta la implementació d'alguna funcionalitat, el temps dedicat a millores serà dedicat a poder acabar el projecte.
També pot ser que alguna funcionalitat s'implementi més de pressa del previst. Si fos així es començaria la següent iteració abans del previst.

3.2 Descripció de les tasques

3.2.1 Aprenentatge de les tecnologies i posada en marxa

Aquesta serà la primera fase del projecte. Al ser un projecte on el punt de partida és un treball anterior, caldrà veure i estudiar com es va desenvolupar la eina, i decidir si continuar usant les mateixes tecnologies o no. D'altra banda, la API que s'haurà de desenvolupar no té punt de partida, pel que s'haurà de escollir una tecnologia per implementar-la. La primera part del treball és doncs, d'aprenentatge autònom de les tecnologies que s'utilitzaran com són Angular 2 per la web i NodeJS per la API. Aquesta fase no te ninguna dependència de procedència.

3.2.2 Planificació del projecte

Aquesta fase es basa en l'elaboració del document de gestió del projecte. Es desenvoluparà principalment en l'assignatura de GEP, i s'aniran afegint tots els aspectes tècnics que no es demanen a l'assignatura, però que s'hauran d'explicar al final. és una fase important perquè es definirà tot el context del projecte com és l'abast, objectiu final, l'estat de l'art o de que tracta i com es farà el projecte, entre altres coses. Aquesta fase no té cap dependència de procedència, però si que té unes dates límits, marcades pel calendari de GEP i l'entrega del treball final de grau.

3.2.3 Desenvolupament de la web i la API

Aquesta tasca té com a dependència de procedència les dues tasques anteriors. Es tracta de desenvolupar l'aplicació web, seguint tots els requisits que s'han especificat a la planificació del projecte. Consta de quatre fases que són l'anàlisi de requisits, l'especificació i disseny, implementació i testos. Cada una d'aquestes té com a dependència de procedència l'anterior, i es realitzaran de forma seqüencial. Es desenvoluparà a la vegada la pàgina web i la API, ja que una necessita l'altra per poder funcionar. El testeig es farà mentre es va desenvolupant, i posteriorment en les reunions amb el director i codirector es provarà de nou que tot funcioni com s'espera i es valorarà si s'han de fer modificacions.

3.2.4 Millores (Opcional)

Aquesta tasca serà opcional, i dependrà de la rapidesa amb la qual s'ha desenvolupat el sistema i els seus requeriments. Tracta de desenvolupar noves millores en el producte, ja siguin millorant el disseny, el funcionament o afegint noves funcionalitats. La dependència de procedència és la tasca anterior.

3.2.5 Posada en marxa

Un cop s'hagi acabat el desenvolupament de totes les funcionalitats, el software s'haurà de penjar per tal que tots els usuaris el puguin fer servir. Això vol dir que tant la API com la pàgina web s'hauran de penjar a un servidor, ja que desenvolupament es treballarà en servidor local. Aquesta tasca té com a procedència les dues anteriors.

3.2.6 Documentació i preparació de la defensa

Aquesta serà l'última fase i tractarà en finalitzar la documentació, i comprovar que sigui correcte. S'anirà redactant mentre es van realitzant les tasques anteriors. La memòria del projecte inclourà la documentació que es farà durant l'assignatura de GEP, així com la documentació derivada de la realització de totes les tasques, i un manual d'usuari del producte desenvolupat. Aquesta documentació servirà per preparar la defensa del treball final de grau, que és l'ultima fase del projecte. Aquesta tasca té com a procedència totes les tasques anteriors.

3.2.7 Observacions

Com s'ha comentat a l'apartat de metodologia, es treballarà amb iteracions curtes, per tal que els directors a cada reunió puguin donar feedback. Serà la tasca *Desenvolupament de la web i la API* la que ocuparà més temps, i també serà on hi haurà les iteracions. Les dates de les iteracions es poden veure al diagrama de Gant, i el contingut que s'implementarà a cada iteració pot canviar durant el transcurs del projecte. La distribució de les iteracions i el seu contingut és el següent:

Iteració 1 i 2: Dedicades a la implementació del registre a la pàgina. L'usuari podrà donar-se d'alta, entrar al sistema introduint les seves credencials, canviar la contrasenya i recuperar la contrasenya.

Iteració 3 i 4: Dedicades a la implementació de la pàgina principal. L'usuari podrà veure els seus projectes, veure els membres, i unir-se a projectes.

Iteració 5: Dedicades a la implementació de la configuració de l'usuari. L'usuari podrà modificar els seus atributs i afegir o esborrar habilitats.

Iteració 6: Dedicades a la implementació de la gamificació.

Iteració 7: Iteració reservada per si hi ha hagut algun imprevist que hagi fet retardar la implementació. En el cas que tot hagi anat correctament, s'aprofitarà per millorar Replan.

3.3 Calendari

3.3.1 Estimació d'hores

Tasca	Responsable	Hores
Aprenentatge de les tecnologies i posada en marxa		**40**
Posada en marxa	Cap del projecte	10
Aprenentatge autònom	Programador	30
Planificació del projecte		**100**
Definició de l'abast	Cap del projecte	20
Gestió econòmica	Cap del projecte	11
Gestió de sostenibilitat	Cap del projecte	11
Presentació preliminar	Cap del projecte	11
Contextualització i Bibliografia	Cap del projecte	12
Entrega d'especialitat	Cap del projecte	15
Document final	Cap del projecte	20
Desenvolupament de la web i la API		**300**
Anàlisis de requisits	Analista	50
Disseny i especificació	Dissenyador	75
Implementació i testeig	50% programador / 50% tester	175
Millores		**35**
Anàlisis de requisits	Analista	5
Disseny i especificació	Dissenyador	10
Implementació i testeig	50% programador / 50% tester	20
Posada en marxa		**15**
Posada en marxa	Programador	15
Documentació i preparació de la defensa		**50**
Redacció de la memòria	Cap del projecte	**35**
Presentació oral	Cap del projecte	**15**
Total		**540h**

Taula 1. Estimació d'hores

3.3.2 Diagrama de Gantt

El diagrama de Gantt correspon a la planificació de tasques explicada anteriorment, respectant les hores corresponents.

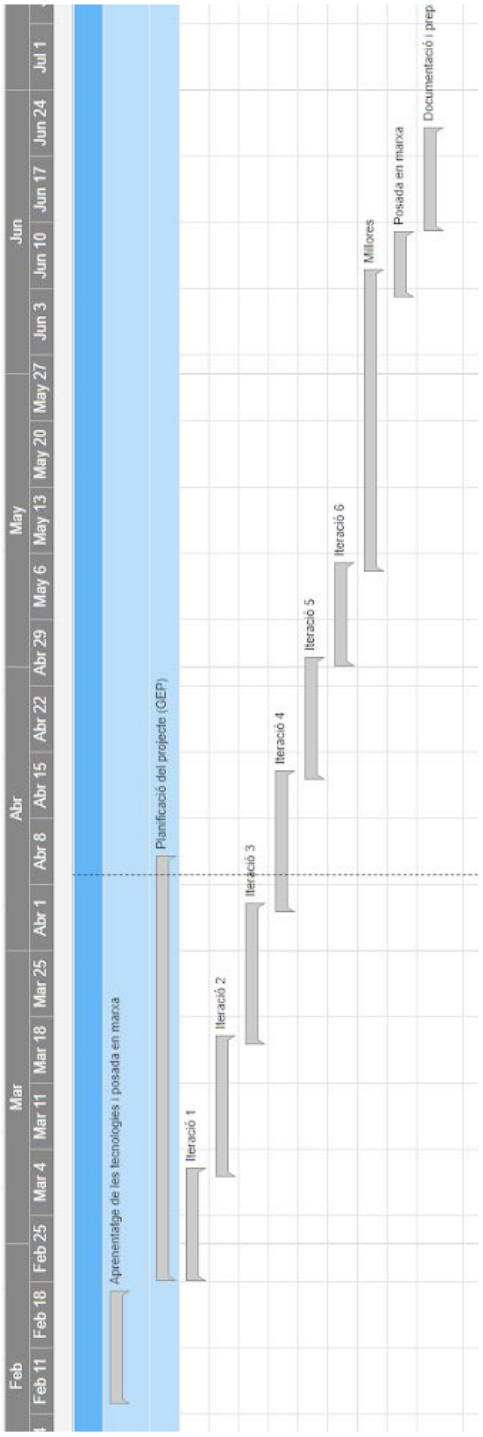


Figura 1. Diagrama de Gantt simplificat

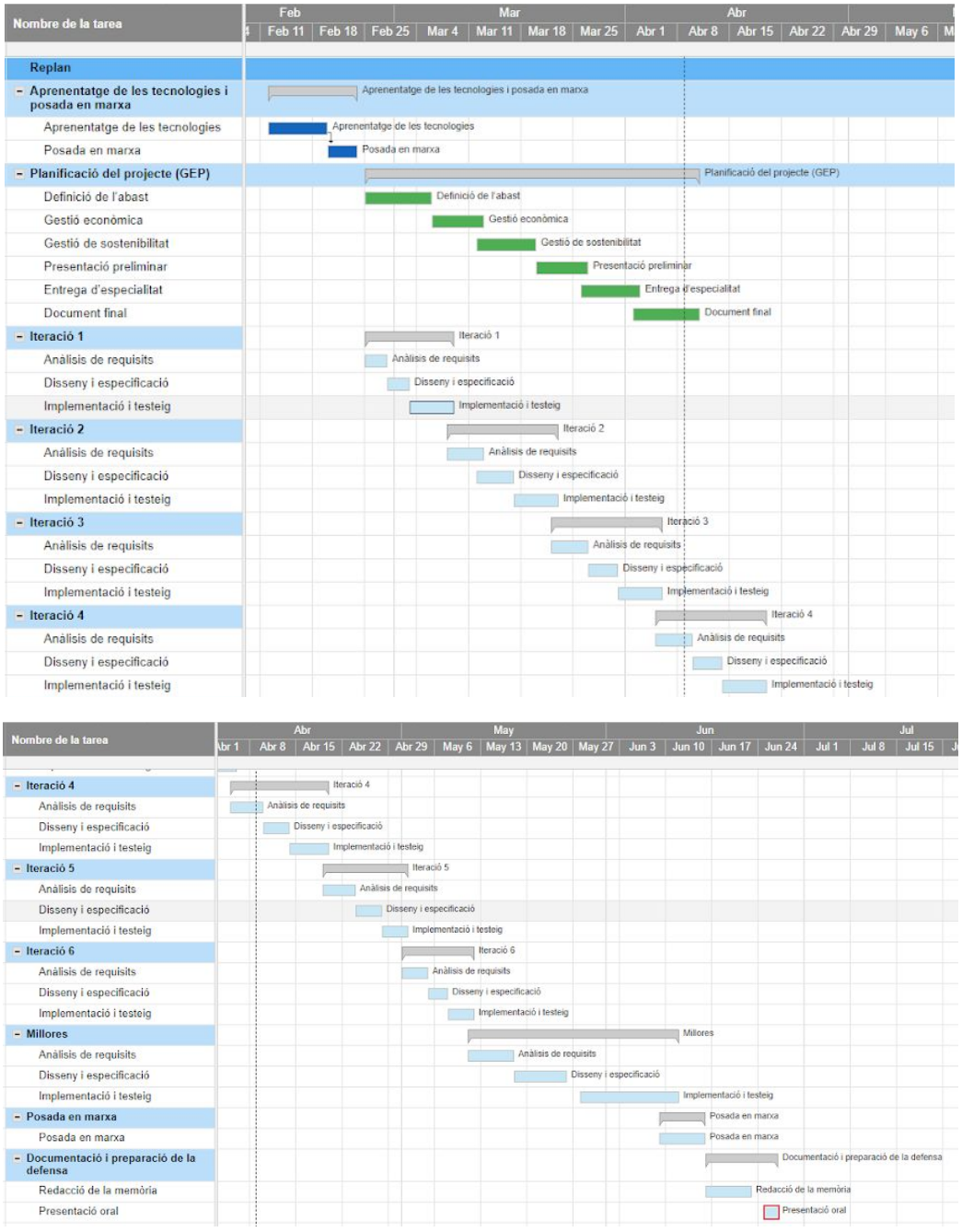


Figura 2,3. Diagrama de Gantt, part 1 i 2

3.4 Canvis respecte la planificació inicial

Per la realització del projecte, s'ha respectat tant la disponibilitat temporal com els recursos utilitzats. Pel que fa a la planificació del treball, hi ha hagut un canvi en els objectius que repercuteix en la iteració 6. En un principi, aquesta iteració havia d'estar dedicada a introduir elements de gamificació per fer l'aplicació més atractiva pels usuaris, però es va acordar amb el director i codirector del projecte modificar-lo. Aquesta iteració es va destinar a afegir diferents tipus d'usuari. En concret, implementar un tipus d'usuari d'administrador per tal de que pugues fer més coses a la web que un usuari normal.

Aquesta modificació no te cap repercussió en els costos del projecte, ja que les hores previstes per la dedicació de la iteració 6 seran les mateixes.

4. Gestió econòmica i sostenibilitat

4.1 Enquesta de sosteniblitat

Aquesta enquesta tracta d'analitzar el nivell que tenim els alumnes en sostenibilitat, tenint en compte tres àmbits com són el social, ambiental i econòmic. Una solució a un problema es considera sostenible quan es consideren aquestes tres dimensions simultàniament, i cada un d'ells alhora es considera sostenible.

És per això, que en pensar en com solucionar un problema, a banda d'intentar pensar en fer un software que compleixi tots els requisits funcionals i no funcionals, també s'ha de veure quin impacte tindrà, ja que per exemple, uns costos elevats de manteniment podria fer que el projecte passes de ser un èxit a un fracàs.

No s'ha de considerar només la sostenibilitat quan es crea el producte, sinó al llarg de tota la seva vida útil, i com posteriorment es reciclarà.

S'ha de veure si els beneficis de dur a terme el projecte són més grans que les pèrdues. De manera senzilla, això es pot veure contestant a la pregunta: Aportarà valor el producte a la societat?

Si la resposta és si, vol dir que val la pena aquell producte, ja que les repercussions que aquest tindrà sobre la societat, el medi ambient i l'economia serà més petit que el valor que aportarà a la societat. Si pel contrari, la resposta és no, s'hauria d'estudiar una alternativa, que no és un producte beneficiós.

Un altre aspecte important en un projecte és la planificació, i un posterior control. Al començar s'ha de fer una planificació i unes previsions de com serà el transcurs del projecte. Posteriorment, s'hauran d'anar fent revisions per anar controlant si la planificació es va complint.

4.2 Identificació dels costos

Els costos del projecte estan directament relacionats amb l'apartat de planificació, on s'explica quins són els recursos del projecte, i com està plantejada la repartició de feina al llarg del temps. Per fer l'estimació econòmica i calcular els diners que val la realització del sistema, es tindran en compte salaris mitjans per les diferents posicions dels membres de l'equip.

Com ja vam comentar, el Pau Campaña serà l'únic desenvolupador de l'aplicació. Tot i això, s'ha trobat convenient dividir les tasques segons la seva naturalesa per càrrecs que es

troben en projectes software, i així poder fer un pressupost més complet. Per a fer la gestió de despeses, classificarem els recursos entre humans i altres.

4.2.1 Recursos humans

Com hem comentat, el projecte serà dut a terme per una sola persona. Aquesta persona serà l'encarregada de fer totes les tasques, que s'ha agrupat en diferents rols que es troba en projectes software com són cap de projecte, analista, programador, tester i dissenyador.
Per fer el pressupost, s'han fet dues estimacions dels costos dels recursos humans. La primera (variant 1) és tenint en compte la remuneració que recomana la FIB per estudiants en practiques o fent el TFG en empresa, que és de 8 euros l'hora. La segona (variant 2) és tenint en compte salaris reals segons cada posició[7].
Per fer una aproximació del que es cobra per hora cada rol, s'ha analitzat diferents ofertes de treball per cada posició, i s'ha fet una aproximació a la mitjana. L'estimació de salaris és la següent:

Rol	**Salari**(€/hora)$^{\text{Variant 1}}$	**Salari**(€/hora)$^{\text{Variant 2}}$
Project manager	8	34
Analista	8	30
Programador	8	22
Tester	8	24
Dissenyador	8	24

Taula 2. Salaris de cada rol.

A partir de la planificació temporal, podem calcular el cost estimat del projecte per les dues variants.

Rol	**Dedicació** (en hores)	**Cost estimat** (en euros) (Variant 1)	**Cost estimat** (en euros) (Variant 2)
Cap de	160	1.280,00	5.440,00

projecte			
Analista	55	440,00	1.650,00
Programador	142,5	1.140,00	3.135,00
Tester	97,5	780,00	2.340,00
Dissenyador	85	680,00	2.040,00
Total	540	**4.320,00**	**14.605,00**

Taula 3. Cost estimat segons rol

Per tenir més exactitud, s'ha trobat convenient dividir els costos per cada tasca.

Tasca	**Dedicació** (en hores)					**Cost estimat** (en euros) (Variant 1)	**Cost estimat** (en euros) (Variant 2)
	Cap de projecte	Analista	Programador	Tester	Dissenyador		
Aprenentatge	10	0	30	0	0	320,00	1.000,00
Planificació	100	0	0	0	0	800,00	3.400,00
Desenvolupament	0	50	87,5	87,5	75	2.400,00	7.325,00
Millores	0	5	10	10	10	280,00	850,00
Posada en marxa	0	0	15	0	0	120,00	330,00
Preparació de la defensa	50	0	0	0	0	400,00	1.700,00
Total	160	55	142,5	97,5	85	**4.320,00**	**14.605,00**

Taula 4. Cost estimat segons tasca

4.2.2 Altres recursos

A part dels recursos humans, també hem de tenir en compte els recursos hardware i software que es fan servir, així com altres despeses.

4.2.2.1 Hardware i software

Cal tenir en compte tot el hardware i software que utilitzarem. Segons la vida útil del producte i el temps que disposem per aquest projecte (4 mesos), es calcularan els costos. Quasi tots els recursos materials anomenats a l'apartat de recursos són gratuïts, així que només posarem a la taula aquells que suposen un cost.

Producte	Preu(en euros)	Unitats	Vida útil (en anys)	Cost (en euros)
Asus ZenBook UX330UA-FC143T	899,00	1	5	59,93
Windows 10.1	45,00	1	5	3,00
Total	944,00			**62,93**

Taula 5. Cost estimat de hardware i software

4.2.2.2 Altres despeses

A part de les despeses mencionades anteriorment, es podrien també considerar les despeses indirectes. Per poder realitzar el treball, es necessita internet i llum. Tot i això, no s'ha inclòs dins el pressupost, ja que aquests serveis no han estat contractants per fer el treball sinó que ja es disposava d'ells. Quan es treballa des de l'UPC, s'aprofita la xarxa i les instal·lacions de la universitat. Quan es treballa des de casa, s'utilitzen els serveis existents en l'habitatge.

Tampoc s'han considerat despeses de desplaçament per fer les reunions presencials, ja que es va acordar amb els directors del projecte fer reunions els dies que Pau Campaña tingués classes d'alguna assignatura.

4.2.3 Contingència

Al cost obtingut, li afegim un percentatge de contingència d'un 10 per cent per poder fer front a possibles riscos no identificats.

Recursos	Percentatge de contingència	Cost (en euros) (Variant 1)	Cost (en euros) (Variant 2)
Recursos humans	10	432,00	1.460.50
Altres recursos	10	6,29	6,29
Total		**438,29**	**1.466,79**

Taula 6. Cost estimat de contingència

4.3 Pressupost total

Un cop hem vist tots els costos, els agrupem per veure el cost total, tant per la variant 1 com la 2.

Tipus	Cost (en euros) (Variant 1)	Cost (en euros) (Variant 2)
Recursos humans	4.320,00	14.605,00
Altres recursos	62,93	62,93
Contingència	438,29	1.466,79
Total	**4.821,22**	**16.134,72**

Taula 7. Cost estimat total

4.4 Viabilitat econòmica

Veiem que hi ha una gran diferència de preu entre les dues variants. Per analitzar la viabilitat econòmica, tindrem en compte la variant 1, ja que el desenvolupador del treball, segons les seves experiències i habilitats, es troba encara fent la carrera. Per tant, s'ajusta més al salari de 8 euros l'hora del conveni de la UPC, que no salaris del món laboral on els treballadors ja tenen més experiència.

Hem de tenir en compte que el hardware i software d'aquest projecte ja està amortitzat, ja que ja es disposava d'ell abans de començar el treball. Tots els softwares que s'han hagut d'instal·lar pel treball són gratuïts.
Per altra banda, els recursos humans s'emmarquen dins d'un projecte de 15 crèdits ECTS, i veiem que les hores estan dins dels límits calculats en l'entrega anterior.
Així doncs, podem concloure que el projecte és viable tant des del punt de vista del temps com dels diners.

4.5 Control

La part dels recursos no humans no es pot fer un seguiment de les despeses, ja que es fa un únic pagament per aconseguir el hardware i software. Com hem comentat, ja disposàvem d'ell abans de començar el treball.

Respecte als recursos humans, es farà un control de seguiment constant. Al final de cada tasca, s'estudiarà si la dedicació en temps és molt diferent de la del pressupost. Si fos així, s'estudiarà a què és degut a aquesta diferència d'hores i s'intentarà corregir les estimacions d'hores i la seva repartició.
Durant la fase de desenvolupament, al tractar-se de la més llarga, es farà un control especial. A cada reunió presencial amb els directors l'estudiant valorarà si alguna tasca en concret està ocupant-li més hores de les previstes, i es proposaran alternatives.

5. Sostenibilitat

5.1 Economica

Es considera que un projecte és viable econòmicament quan es realitza una avaluació de tots els costos, i s'estableixen protocols per controlar i ajustar el pressupost del projecte. Aquests aspectes estan explicats en els apartats anteriors. Si es tractes d'un projecte real, per fer-lo més competitiu es podria realitzar en menys temps. Això es pot aconseguir de dues maneres. O ve l'estudiant dediqués més hores a fer el treball per dia (per exemple a temps complet), o bé augmentar el personal.

5.2 Social

Com s'ha vist a l'estat de l'art, aquesta aplicació pot afegir valor a les empreses, ja que veuran com la gestió dels projectes software millora. Podran fer servir Replan, sense que els suposi un gran esforç inicial, per organitzar el projecte i així estalviar temps.
Per altra banda, les companyies que ofereixen softwares similars seran les perjudicades, ja que podrien veure com les empreses canvien el software actual per Replan.

5.3 Ambiental

S'ha de tenir en compte el consum elèctric de l'ordinador amb el qual desenvoluparem el treball. Aquest ordinador s'utilitza també per altres tasques, però el fet d'haver de fer el sistema l'augmenta. L'ordinador té una antiguitat d'un any, pel que el seu cost encara no està amortitzat. Tot i això, les previsions són de continuar fent servir l'ordinador en finalitzar el treball.
Pel que fa a la documentació, es fa de manera totalment virtual. Només es necessitaran papers i tinta per fer la impressió del document final. El contacte amb els directors, quan no és a la reunió presencial, es fa mitjançant correus electrònics pel que tampoc es fa ús de paper.
Replan és un producte software, pel que no requereix cap cost de fabricació ni genera cap tipus de contaminació. Tampoc ens hem de preocupar del reciclatge, ja que quan una empresa decideixi deixar de fer servir l'eina, podrà continuar fent servir l'ordinador sense cap canvi.

6. Anàlisis de requisits

En aquesta secció es detallaràn tots el requisits funcionals i no funcionals que té el sistema. Algunes de les funcionalitats ja existien previament al treball, però moltes s'han hagut d'adaptar a que ara l'aplicació és multiusuari.

6.1 Requisits funcionals

A continuació es detallen els requisits funcionals que ha de complir el sistema.

Número	Requisit
RF1	**Descripció**: El software ha de permetre a un usuari a crear, modificar, eliminar i veure una Feature de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). **Validació:** Quan un usuari ha iniciat sessió i es troba dins un projecte, aquest pot crear, modificar, eliminar i veure una Feature.
RF2	**Descripció:** El software ha de permetre a un usuari crear, modificar i veure una Skill de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, dins la secció de perfil pot crear, modificar i veure una Skill.
RF3	**Descripció:** El software ha de permetre a un usuari administrador crear, modificar, eliminar i veure una Skill de la base de dades del sistema. **Prioritat:** Baixa. **Dependència:** L'usuari administrador ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari administrador ha iniciat sessió, dins la secció de perfil pot crear, modificar, eliminar i veure una Skill.
RF4	**Descripció:** El software ha de permetre a un usuari crear, modificar, eliminar i veure Jobs de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear alguna feature (**RF7**). L'usuari ha de poder crear i veure una Release (**RF8**). L'usuari ha de poder veure el Plan (**RF6**).

	Validació: Quan un usuari ha iniciat sessió, dins d'un projecte, pot veure el Plan d'una Release, on apareixen els Jobs i els pot crear, modificar veure i eliminar.
RF5	**Descripció:** El software ha de permetre a un usuari crear, modificar, eliminar i veure un Project de la base de dades. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, pot crear, modificar, eliminar i veure un projecte.
RF6	**Descripció:** El software ha de permetre a un usuari crear i eliminar un Plan per a una Release específica. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear alguna Feature (**RF7**). L'usuari ha de poder crear i veure una Release (**RF8**). **Validació:** de poder veure el Plan (**RF6**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot veure el Plan d'una Release, crear-ne un de nou o eliminar-lo.
RF7	**Descripció:** El software ha de permetre a un usuari veure totes les Features d'un determinat projecte de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). **Validació:** Quan un usuari ha iniciat sessió i es troba dins un projecte, aquest pot veure totes les Features del projecte.
RF8	**Descripció:** El software ha de permetre a un usuari veure totes les Releases d'un determinat projecte de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). **Validació:** Quan un usuari ha iniciat sessió i es troba dins un projecte, aquest pot veure totes les Releases del projecte.
RF9	**Descripció:** El software ha de permetre a un usuari veure totes les Skills de la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, dins la secció de perfil pot veure totes les Skills.
RF10	**Descripció:** El software ha de permetre a un usuari afegir Features a una determinada Release base de dades del sistema.

	Prioritat: Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear alguna Feature (**RF7**). L'usuari ha de poder crear una Release (**RF8**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot afegir Features a una Release.
RF11	**Descripció:** El software ha de permetre a un usuari veure totes les Features d'una determinada Release de la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder veure una Release (**RF8**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot veure totes les Features d'una determinada Release.
RF12	**Descripció:** El software ha de permetre a un usuari veure tots els Projectes en els que és membre de la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió des de la pantalla principal pot veure tots els Projectes en els que és membre.
RF13	**Descripció:** El software ha de permetre a un usuari afegir usuaris a una determinada Release. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear i veure una Release (**RF8**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot afegir usuaris a una determinada Release.
RF14	**Descripció:** El software ha de permetre a un usuari afegir Skills a una determinada Feature. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder veure alguna Feature (**RF7**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot afegir Skills a una determinada Feature.
RF15	**Descripció:** El software ha de permetre a un usuari afegir-se Skills. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**).

	Validació: Quan un usuari ha iniciat sessió, dins la secció de perfil pot afegir-se Skills.
RF16	**Descripció:** El software ha de permetre a un usuari afegir dependències entre Features. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear, veure i modificar Features (**RF7**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot afegir dependències entre Features.
RF17	**Descripció:** El software ha de permetre a un usuari eliminar Features d'una determinada Release. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder veure i modificar una Release (**RF8**). L'usuari ha de poder veure el Plan (**RF6**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot veure el Plan d'una Release, des d'on pot eliminar les Features.
RF18	**Descripció:** El software ha de permetre a un usuari eliminar usuaris d'una determinada Release. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder veure i modificar una Release (**RF8**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot eliminar usuaris d'una determinada Release.
RF19	**Descripció:** El software ha de permetre a un usuari eliminar Skills d'una determinada Feature. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder veure i modificar Features (**RF7**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot eliminar Skills d'una determinada Feature.
RF20	**Descripció:** El software ha de permetre a un usuari eliminar-se Skills. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, dins la secció de perfil pot eliminar-se Skills.

RF21	**Descripció:** El software ha de permetre a un usuari eliminar dependències entre Features. **Prioritat:** Baixa. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder entrar a un projecte (**RF5**). L'usuari ha de poder crear, veure i modificar Features (**RF7**). **Validació:** Quan un usuari ha iniciat sessió, dins d'un projecte, pot eliminar dependències entre Features.
RF22	**Descripció:** El software ha de permetre registrar-se a la base de dades del sistema. **Prioritat:** Molt alta. **Dependència:** Cap. **Validació:** Qun un usuari entra a Replan i no té la sessió iniciada, pot registrar-se al sistema.
RF23	**Descripció:** El software ha de permetre a un usuari verificar el seu usuari de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari s'ha d'haver registrat (**RF22**). **Validació:** Quan un usuari es registre, rep un correu des d'on es pot verificar el seu usuari.
RF24	**Descripció:** El software ha de permetre a un usuari demanar que es torni a verificar el seu usuari de la base de dades del sistema. **Prioritat:** Baixa. **Dependència:** L'usuari s'ha d'haver registrat (**RF22**). **Validació:** Un usuari registrat pot demanar que es torni a verificar el seu usuari des de la pàgina inicial, a la secció de tornar a verificar el compte.
RF25	**Descripció:** El software ha de permetre a un usuari entrar al sistema oferint les seves credencials. **Prioritat:** Molt alta. **Dependència:** L'usuari s'ha d'haver registrat (**RF22**). L'usuari ha d'haver validat el seu compte (**RF23**). **Validació:** Un usuari registrat i amb el compte verificat pot iniciar sessió des de la pàgina principal.
RF26	**Descripció:** El software ha de permetre a un usuari obtenir una nova contrasenya. **Prioritat:** Normal. **Dependència:** L'usuari s'ha d'haver registrat (**RF22**). **Validació:** Un usuari registrat pot obtenir una nova contrasenya des de la pàgina inicial, a la secció de no recordo la contrasenya.
RF27	**Descripció:** El software ha de permetre a un usuari veure i modificar les seves dades d'usuari.

	Prioritat: Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, dins la secció de perfil pot veure i modificar les seves dades d'usuari.
RF28	**Descripció:** El software ha de permetre a un usuari modificar la seva contraseya. **Prioritat:** Baixa. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, dins la secció de perfil pot modificar la seva contraseya.
RF29	**Descripció:** El software ha de permetre a un usuari membre d'un Project acceptar o rebutjar una sol·licitud d'un altre usuari per unir-se. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, de de la pantalla principal pot acceptar o rebutjar una sol·licitud d'un altre usuari per unir-se.
RF30	**Descripció:** El software ha de permetre a un usuari veure les Notifications de la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, de de la pantalla principal pot veure les Notifications.
RF31	**Descripció:** El software ha de permetre a un usuari cercar projectes de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari ha iniciat sessió, de de la pantalla principal pot cercar projectes des del cercador.
RF32	**Descripció:** El software ha de permetre a un usuari unir-se a un projecte de la base de dades del sistema. **Prioritat:** Alta. **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari ha de poder cercar projectes (**RF31**). **Validació:** Quan un usuari ha iniciat sessió, de de la pantalla principal pot cercar projectes des del cercador, des d'on pot unir-se a un projecte.
RF33	**Descripció:** El software ha de permetre a un usuari deixar de ser membre de un projecte. **Prioritat:** Normal **Dependència:** L'usuari ha de poder iniciar sessió (**RF22, RF25**). L'usuari s'ha d'haver unit a un projecte (**RF33**). **Validació:** Quan un usuari ha iniciat sessió, de de la pantalla principal pot

	deixar de ser membre de un projecte.
RF34	**Descripció:** El software ha de permetre a un usuari administrador eliminar Skills de la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari administrador ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari administrador ha iniciat sessió, dins la secció de perfil pot eliminar Skills de la base de dades del sistema.
RF35	**Descripció:** El software ha de permetre a un usuari administrador veure tots els usuaris que hi ha a la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari administrador ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari administrador ha iniciat sessió, dins la secció d'administrador pot veure tots els usuaris registrats.
RF36	**Descripció:** El software ha de permetre a un usuari administrador veure tots els projectes que hi ha a la base de dades del sistema. **Prioritat:** Normal. **Dependència:** L'usuari administrador ha de poder iniciar sessió (**RF22, RF25**). **Validació:** Quan un usuari administrador ha iniciat sessió, dins la secció d'administrador pot veure tots els projectes registrats.

Taula 8. Requisits funcionals.

6.2 Requisits no funcionals

A continuació es detallen els requisits no funcionals que ha de complir el sistema. Es presenten en una taula i es detallen a continuació.

Número	Descripció
RNF1	**Seguretat**: L'acces a l'informació d'un usuari ha d'estar restringit a aquell usuari.
RNF2	**Rendiment**: El sistema ha de suportar una certa quantitat d'informació i continuar oferint un rendiment òptim.
RNF3	**Portabilitat**: El software ha de poder ser executat en diferents navegadors web i sistemes operatius.
RNF4	**Usabilitat**: El usuari ha de ser capaç d'entre i usar totes les funcionalitats del sofware d'una manera ràpida.
RNF5	**Disponibilitat**: El software ha d'estar accesible en tot moment per l'usuari.
RNF6	**Canviable**: El software ha de ser fàcilment canviable per poder evolucionar en un futur.

Taula 9. Requisits no funcionals.

6.2.1 Justificacions

RNF1: Un usuari, per poder entrar al sistema i veure la seva informació ha de proporcionar les seves credencials. Per altra banda, quan es fa una crida a la API, es comprova que l'usuari tingui els drets necessaris per poder fer tal acció.

RNF2: Es minimitzen els accessos a la base de dades per tal fer el programa el màxim eficient possible. El sistema s'ha provat amb 150 usuaris registrats i 100 projectes, i continuava funcionant ràpidament. També s'han fet proves amb diversos usuaris connectats al mateix temps.

RNF3: El software funciona correctament amb els sistemes operatius Windows, Linux i IOS, i amb els navegadors Google Chrome, Mozilla Firefox i Internet Explorer.

RNF4: S'ha seguit mantenint dins del possible el disseny anterior per tal que els usuaris continuïn trobant les funcionalitats al mateix lloc on estaven. Per a les pantalles totalment noves, s'ha intentat seguir el patró que segueixen altres aplicacions de gestió d'equips software. Pel que fa a l'aspecte, s'ha canviat el to de colors que s'utilitzava. S'ha esculli't uns tons de blau en comptes dels grocs i verd que hi havia anteriorment. S'ha utilitzat la web *Color Wheel* d'Adobe per escollir els colors.[8] S'ha fet un test d'usabilitat d'usabilitat per assegurar aquest requisit no funcional, com podrem veure més endavant.

RNF5: Tant la pàgina web com la API s'han desplegat a Heroku. La disponibilitat de Replan depèn doncs de Heroku.

RNF6: L'aplicació s'ha fet seguint les recomencacions de Angular 2 i NodeJS. És un software fàcil d'usar ja que està estructurat i el codi és de qualitat.

7. Aplicació antiga

7.1 Especificació

7.1.1 Model conceptual

A les següent pàgina es mostra el model conceptual previ a que jo implementes la funcionalitat de multiusuari.

Veiem com un Project pot tenir una o més Release, i cada una d'aquestes Releases té un Plan. Aquest Plan es genera a partir de les Features i els Resources associats. Les Features són el que s'ha de fer, i tenen uns Skills que són necessaris per implementar-ho. Els Resources són les persones, i cada Resource té definit unes Skills. A partir de les Features que es demanen a una Release, i els Resources que té associats, Replan intenta trobar la millor repartició de les tasques generant un Plan. Un Plan té un conjunt de Jobs, i cada Job té assignat una data d'inici i de fi, i és on s'assigna una determinada Feature a un Resource. Les Features poden tenir dependències entre elles ja que pot ser que una s'hagi de fer abans que una altra. Els Resources, a part de tenir un conjunt de Skills, també té una availability, que indica la disponibilitat que té el Resource.

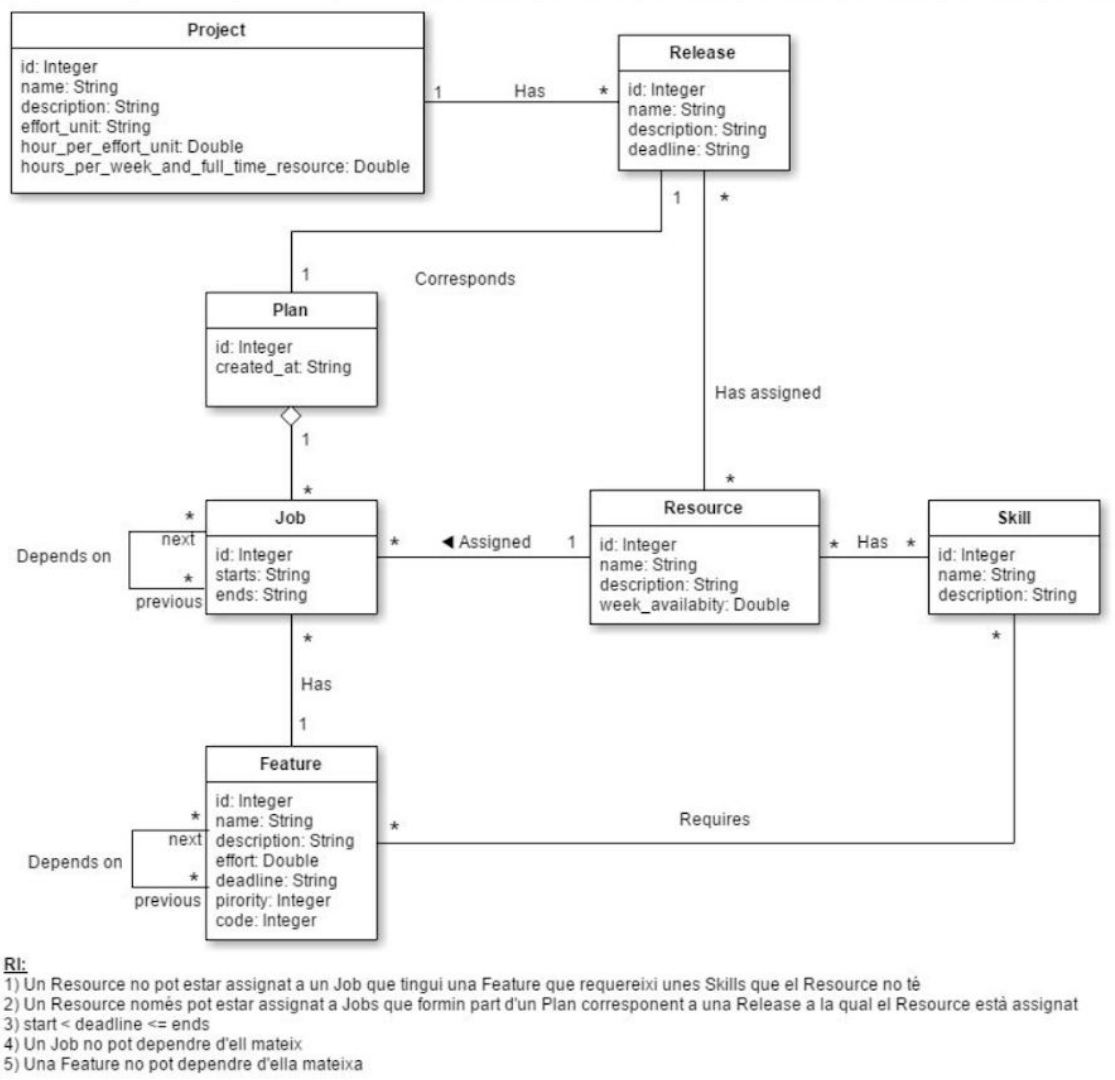


Figura 4. Model conceptual antic.[9]

7.1.2 Diagrama de casos d'ús

A continuació es mostra el diagrama de casos d'ús del software antic. Mostra totes les funcionalitats que l'aplicació antiga permetia realitzar. Com que no hi ha encara cap implementació de les funcionalitats multiusuari, només hi ha un únic usuari que en aquest cas representa el cap de projecte que és qui planifica les entregues d'un projecte i de gestionar els seus recursos.

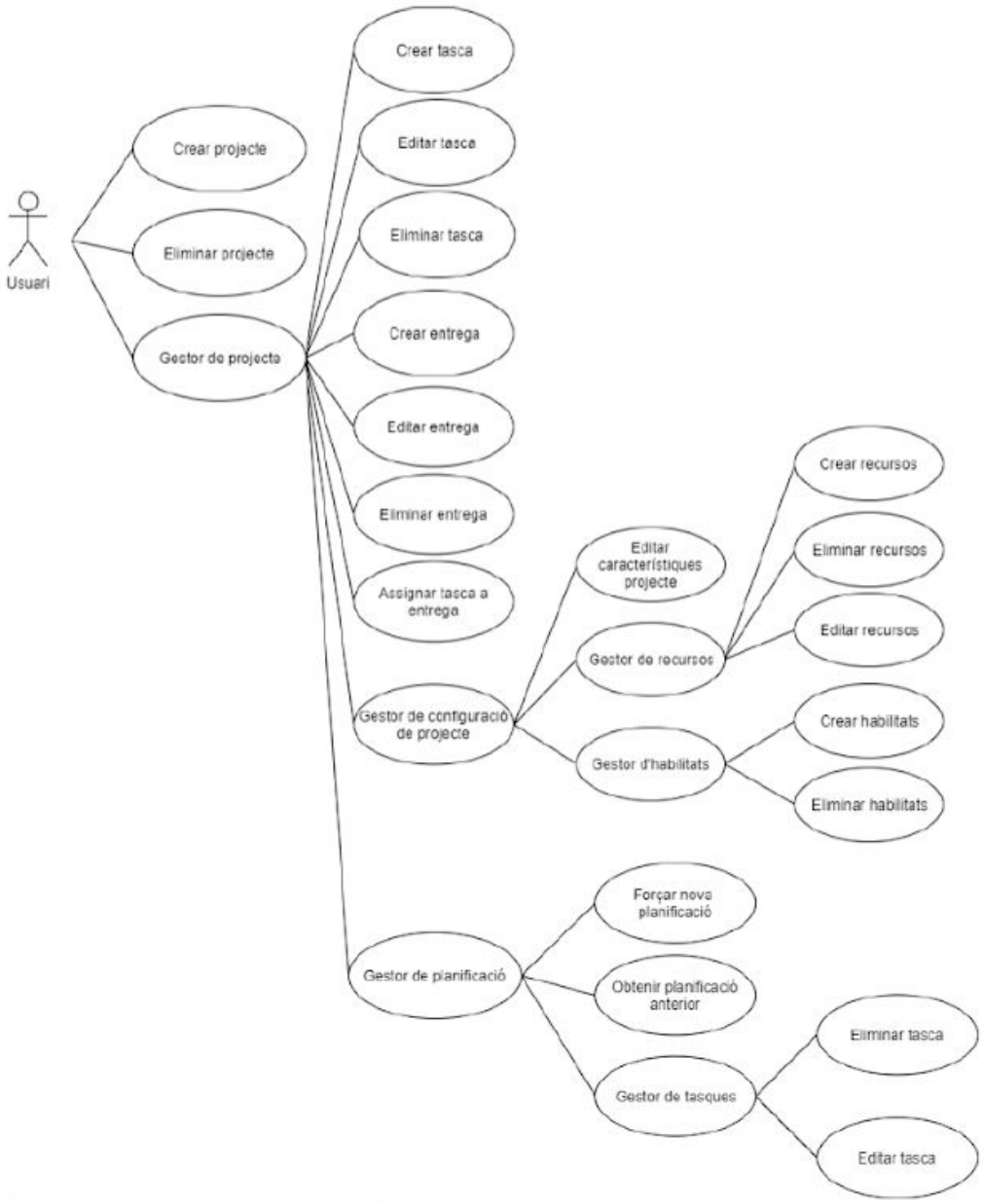


Figura 5. Diagrama de casos d'ús antic.

7.1.3 Disseny de l'aplicació

A continuació es mostra el disseny de les pantalles antigues, per així poder veure els canvis que s'ha fet per adaptar les diferents pantalles a les noves funcionalitats.

Pantalla principal

En la pantalla principal es podien veure tots els projectes pitjant les fletxes que apareixen a l'esquerra i a la dreta del quadre verd. De cada projecte es veu el seu títol i descripció. També és en aquesta pantalla on es pot crear un nu projecte, o eliminar-ne un de concret.

Figura 6. Pantalla principal antiga

Pantalla principal del projecte

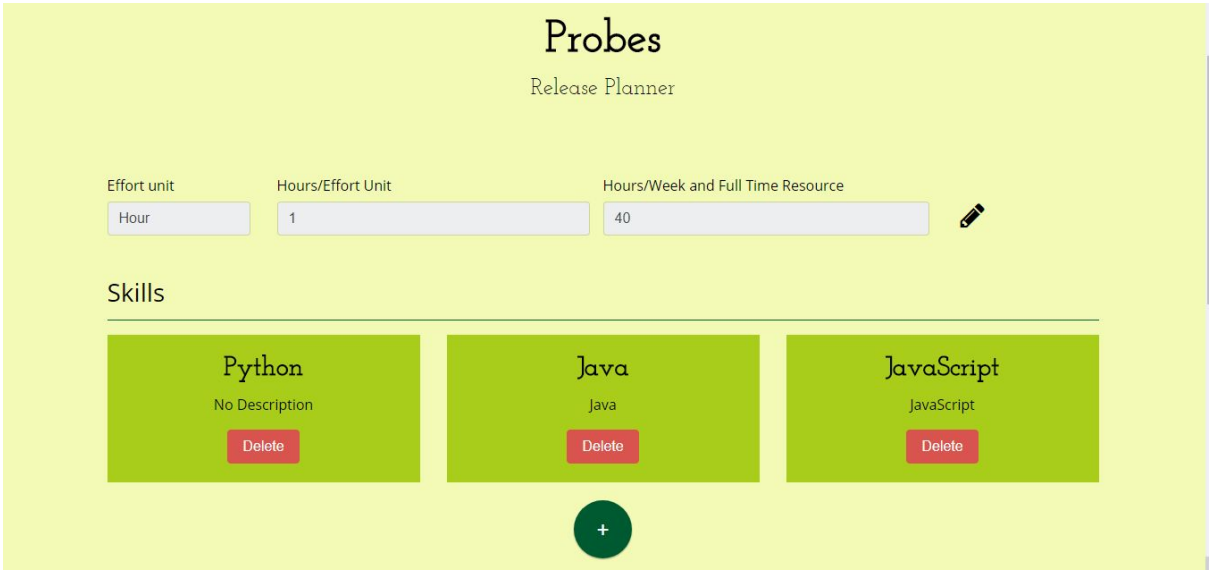
Aquí veiem com un cop hem seleccionat un projecte en la pantalla anterior, apareix tota la informació relacionada amb aquell projecte. A la part superior de la pantalla trobem un menú que ens ofereix tres opcions. Podem tornar enrere, veure la configuració o veure el projecte en si. Aquest menú apareix en totes les pantalles un cop s'ha seleccionat un projecte.

Sota el menú veiem la pantalla principal del projecte on es veu el títol, les Features i les Releases. També hi ha botons per poder crear, eliminar o editar les Features i les Releases.

Figura 7. Pantalla principal del projecte antiga.

Pantalla de configuració del projecte

En aquesta pantalla veiem en primer lloc que podem editar les característiques del projecte. Seguidament veiem com apareixen les Skills definides en aquell projecte, i com també tenim les opcions d'editar-les, eliminar-les o crear-ne de noves. Per últim veiem els Resources que té el projecte, quines habilitats de cada Resource i la seva disponibilitat. Igual que amb les Skills, tenim les opcions d'editar-los, eliminar-los o crear-ne de nous.



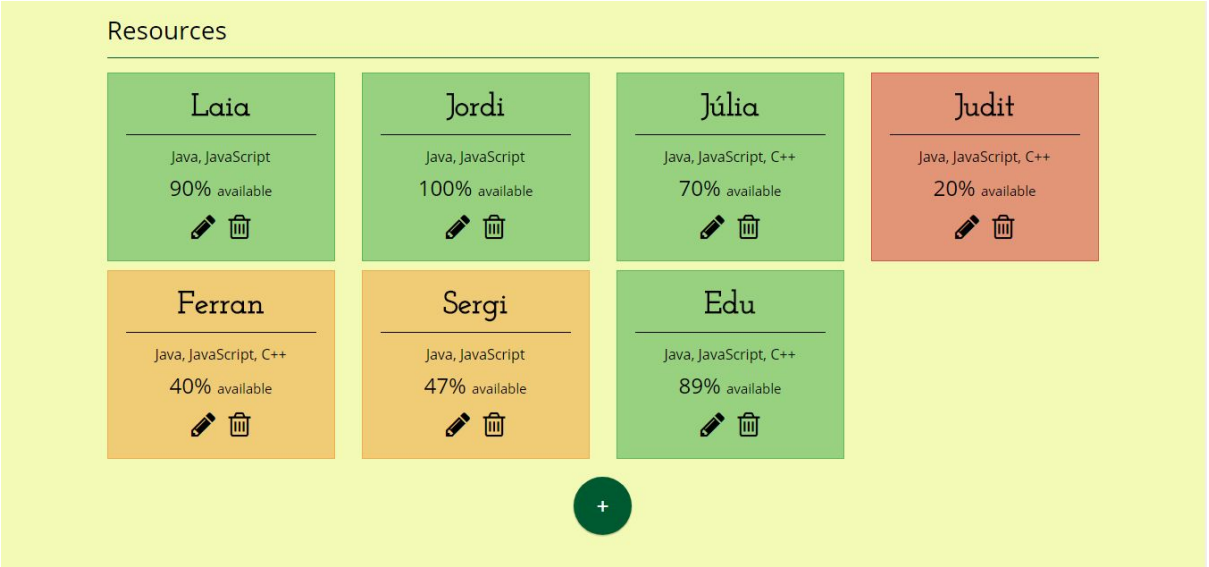
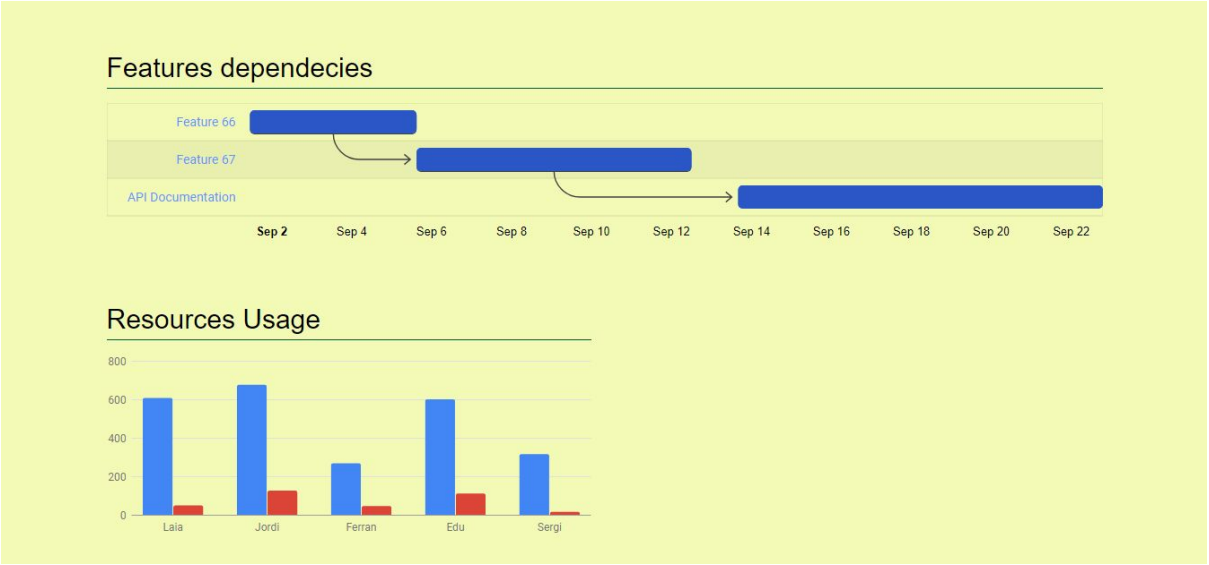


Figura 8 i 9. Pantalla de configuració del projecte antiga.

Pantalla de planificació

En aquesta pantalla podem veure quin és la planificació que ha fet Replan. Es mostra en un gràfic on cada fila correspon a un Resource. Hi ha l'opció de tornar al pla anterior o bé fer-ne un de nou. Sota veiem un altre gràfic que mostra les diferents dependències de les tasques, en cas que existeixin. L'últim gràfic que veiem mostra quina quantitat de temps han de dedicar els diferents Resources al projecte. Per últim, apareixen les diferents Features que no han sigut assignades.



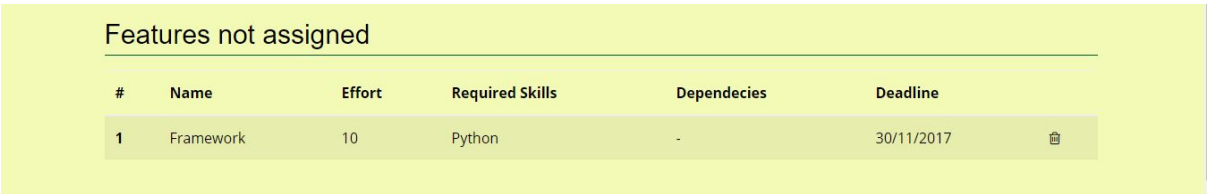


Figura 10, 11 i 12. Pantalla de la planificació del projecte antiga.

8. Aplicació nova

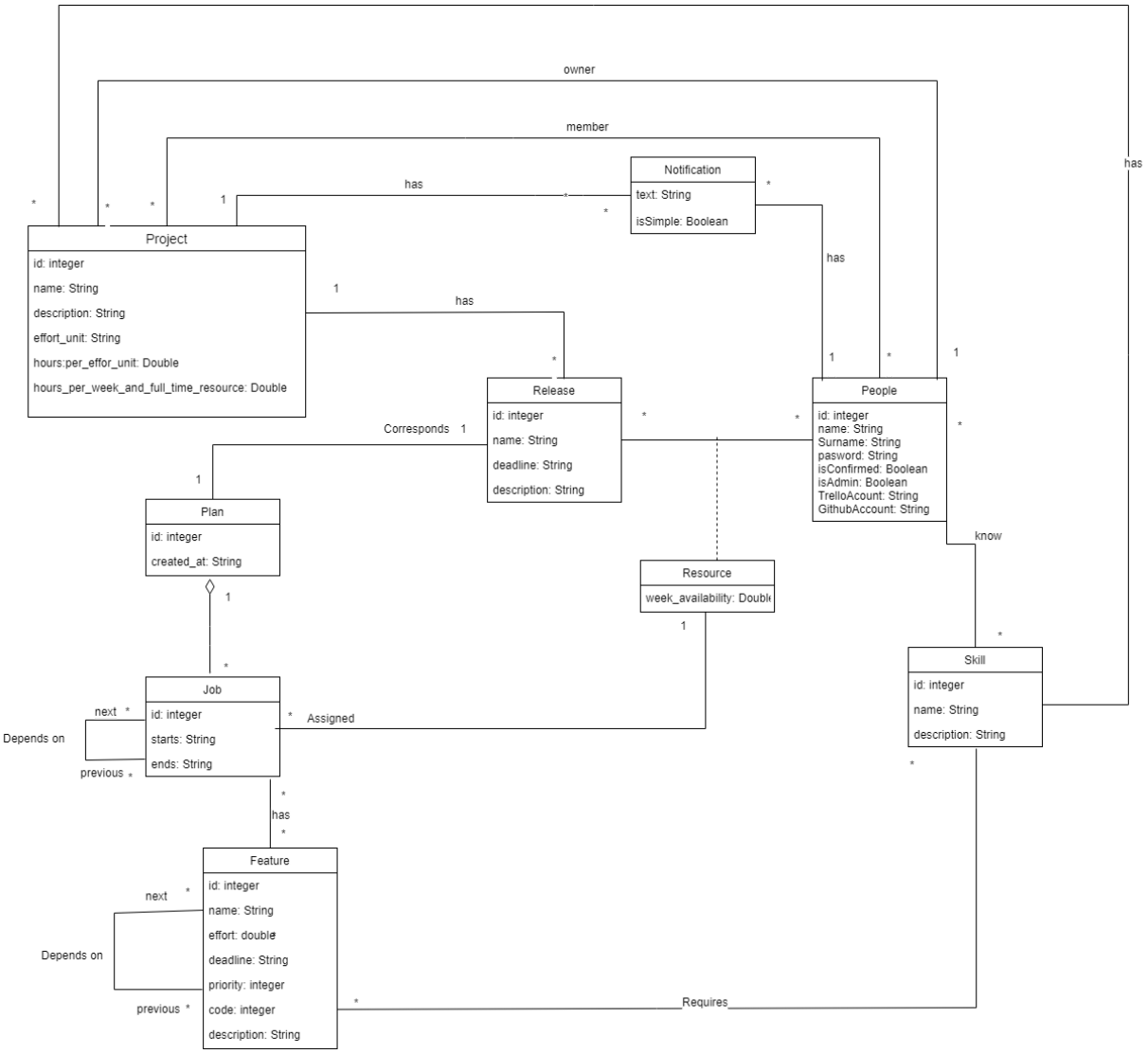
8.1 Especificació

8.1.1 Model conceptual

A partir del model conceptual vell, es van fer un seguit de modificacions per tal de fer el nou. La classe Resource ha desaparegut, i ara tenim People. Hi ha la diferència que ara cada People té un conjunt de Projects, metres que abans a cada Project s'havia de crear tots els Resources que es volia. Per posar un exemple, imaginem que tenim dos Projects, Proj1 i Proj2. Si Proj1 i Proj2 l'ha de fer una mateixa People Peop1. Abans havíem de registrar dos Resources, un a cada projecte, mentre que ara és el mateix Peop1 que és membre o owner dels dos Projects.

Un altre canvi és que ara els Skills són globals. Abans es definien els Skills a cada projecte, però vam veure que seria millor fer-los globals, ja que així no s'havien de registrar cada cop que es creava un Project.

L'altre canvi important és l'aparició de la classe Notification. Una Notification és el que es fa servir per notificar coses als usuaris. Actualment l'ús que se l'hi ha donat és per tenir un registre de quins membres s'han unit o han sol·licitat unir-se a un Projecte.



RI:

1) Una People no pot estar assignada a un Job que tingui una Feature que requereixi unes Skills que la People no té.
2) Una People només pot estar assignada a Jobs que formin part d'un Plan correspoent a una Release a la qual el Resource està assignat.
3) start < deadline < ends.
4) Un Job no pot dependre d'ell mateix.
5) Una Feature no pot dependre d'ella mateixa.

Figura 13. Model conceptual nou.

8.1.2 Adaptació del model conceptual nou

Es va decidir amb els directors del treball no modificar l'API que ja estava funcionant, sinó que era millor crear una altra API que es dediques exclusivament mantenir les funcionalitats relacionades amb el multiusuari. Posteriorment s'explica més detalladament com és l'estructura del sistema, però el fet que hi hagi dues APIs fa que la informació estigui en dos llocs diferents. Això va ser un inconvenient a l'hora d'implementar la classe People, ja que l'altra API continuava treballant amb Resources. El que es va fer per solucionar aquest problema va ser quan registràvem una nova People, també registràvem un Resource, i s'afegia un atribut a People que era el ID del Resource.

Un problema similar passava amb Project. A Project s'havien d'afegir més atributs que els que tenia a la API antiga. La solució que es va adoptar va ser crear una classe Project a la API multiusuari on es guardaven tots els atributs que faltaven, i s'afegia un atribut on es guardava el ID del Project a l'altra API.

Així aconseguíem que la API anterior continues funcionant com fins ara, i l'API multiusuari tenia la informació necessària per poder oferir les seves funcionalitats.

8.1.3 Diagrama de casos d'ús

S'han afegit noves funcionalitats al software, pel que s'ha hagut de tornar a dissenyar l'aplicació. Alguna de les funcionalitats s'accedeixen des del mateix lloc que abans, però altres s'han canviat. S'ha creat un nou diagrama de casos d'ús que com es pot veure ofereix més opcions que l'anterior.

Ara, un usuari pot tenir diferents rols. Un usuari pot ser membre o creador d'un projecte, i la diferència entre aquests és que un membre pot marxar d'un projecte, mentre que un creador no. També hi ha la possibilitat que un usuari tingui poders d'administració. En aquest cas, a part de fer el que podria fer com a creador o membre d'un projecte, pot també eliminar Skills, i accedir a funcionalitats específiques per administradors.

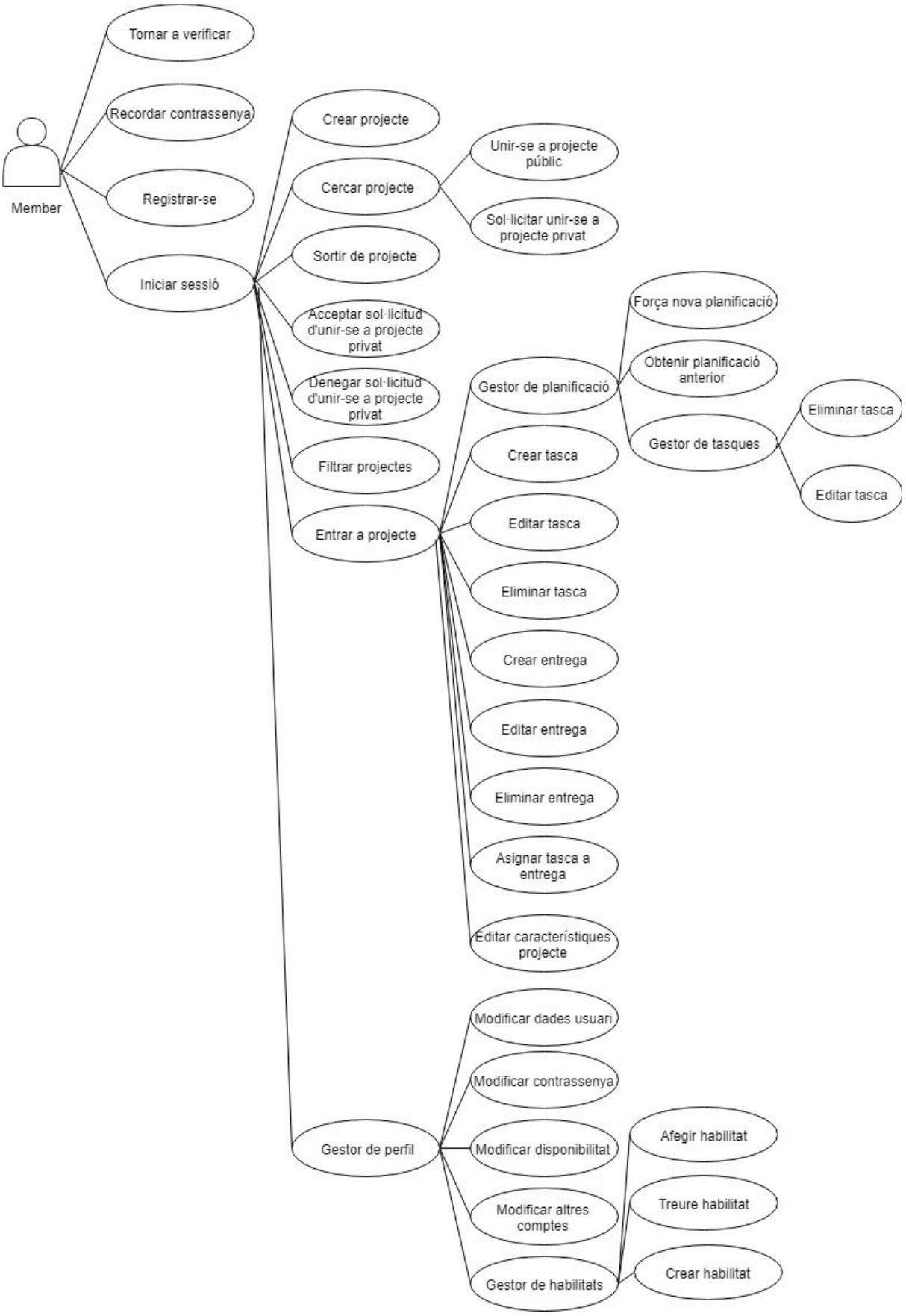


Figura 14. Diagrama de casos d'ús nou per usuari membre.

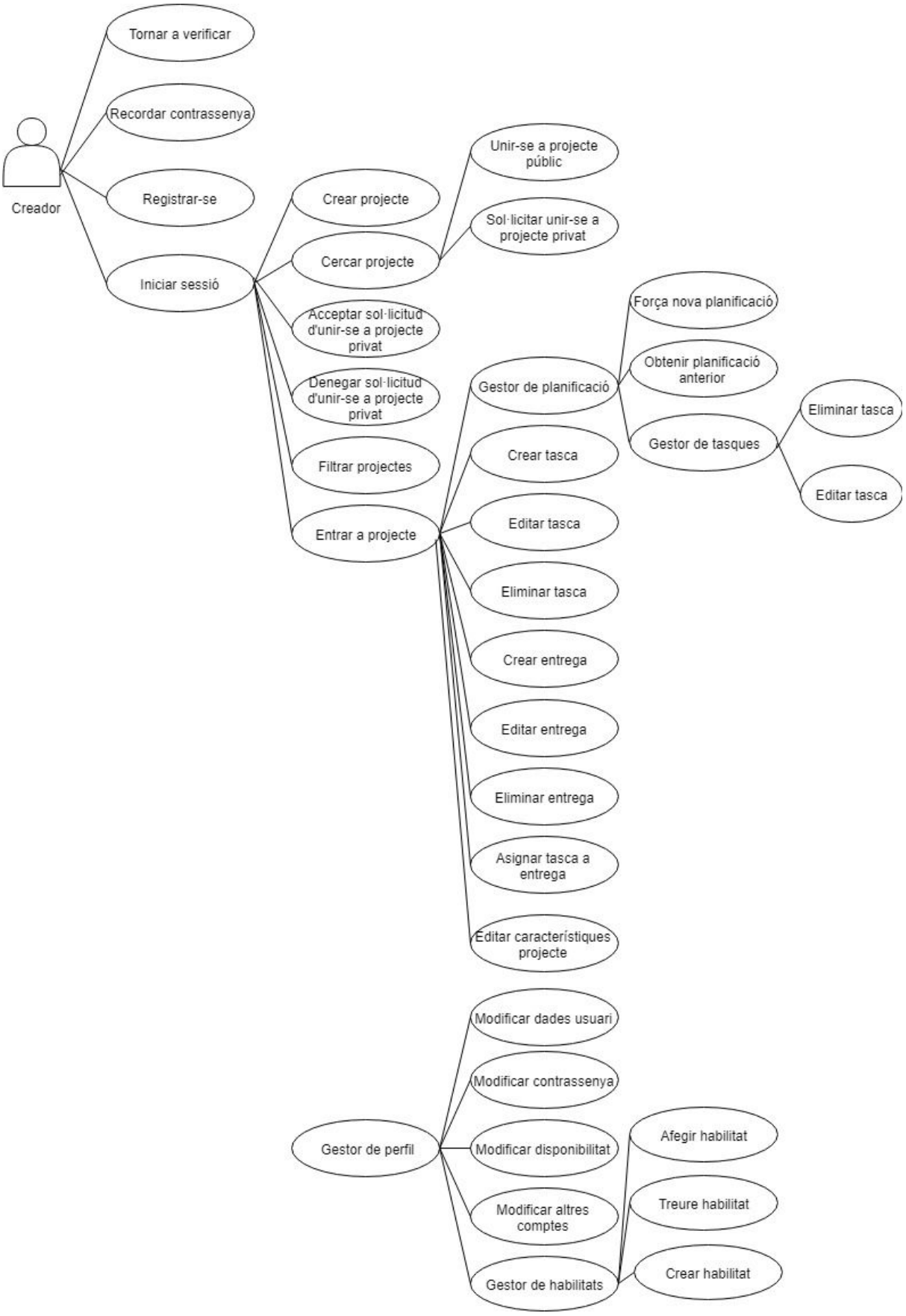


Figura 15. Diagrama de casos d'ús nou per usuaris creador.

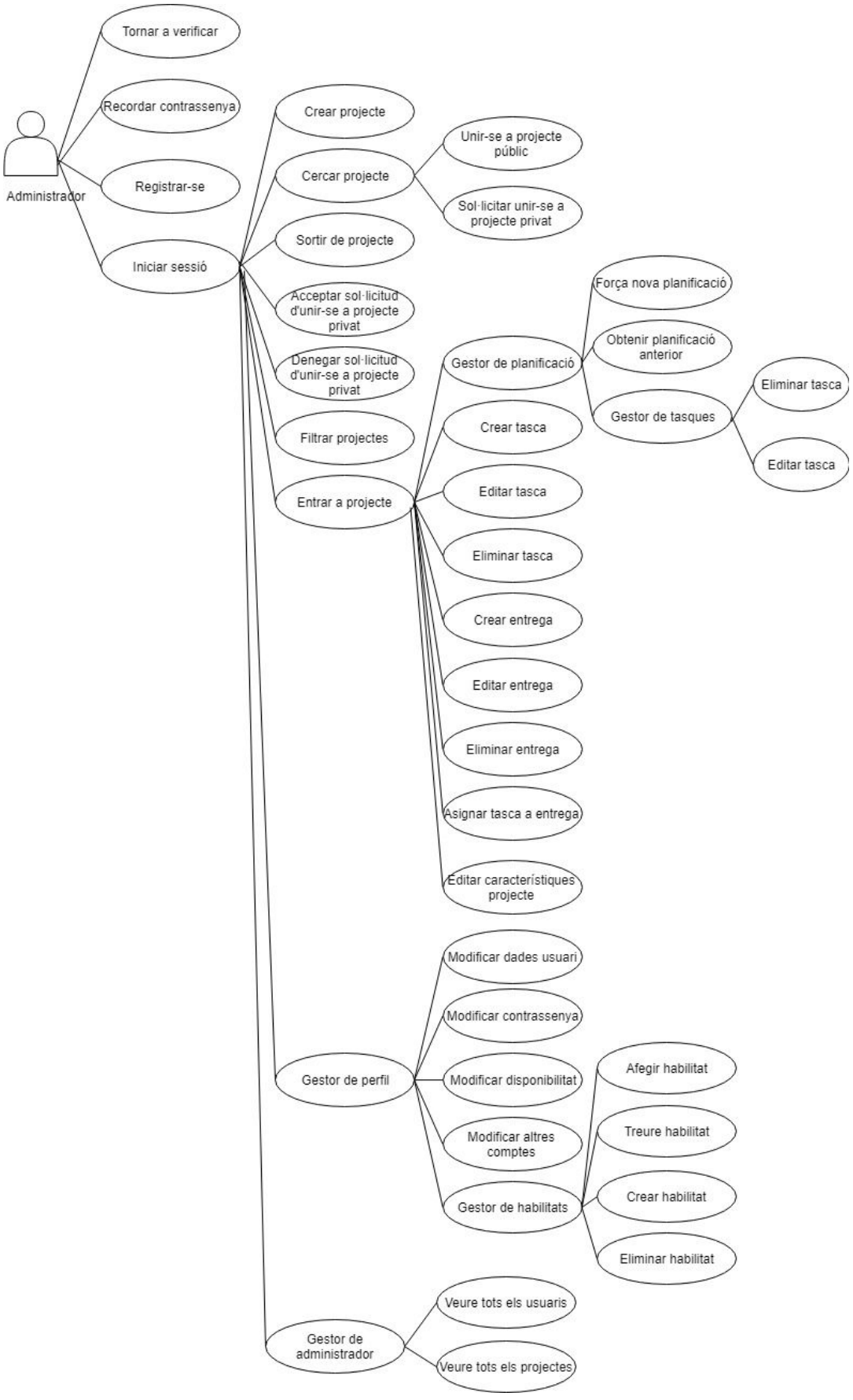


Figura 16. Diagrama de casos d'ús nou per usuaris administradors.

8.2 Disseny de l'aplicació

A continuació es mostra el disseny de les pantalles noves. Aquestes han hagut de ser adaptades per poder oferir correctament totes les funcionalitats a l'usuari. En cada pantalla s'expliquen els canvis que s'han fet.

Pantalla d'iniciar sessió

Aquesta pantalla no existia en la versió anterior. Com es pot veure, s'ha canviat els colors de fons. Juntament amb els directors, vam acordar canviar la tonalitat de totes les pantalles, ja que creiem que la combinació de verds amb grocs no aconseguia l'estètica que ens agradaria. S'ha utilitzat la web Color Wheel d'Adobe per escollir els colors.8

Observem també que aquesta és la pantalla on s'han d'introduir el correu i la contrasenya per tal d'iniciar sessió. S'ha seguit el mateix disseny tant per aquesta pantalla com per la de registre per fer que l'usuari vegi que l'aplicació té un disseny que segueix una mateixa línia.

En el cas que no estiguem registrats, tenim un enllaç que ens porta a la pàgina de registre. També hi ha dos enllaços més per si no has rebut el mail de confirmació del compte o per si no recordes la contrasenya.

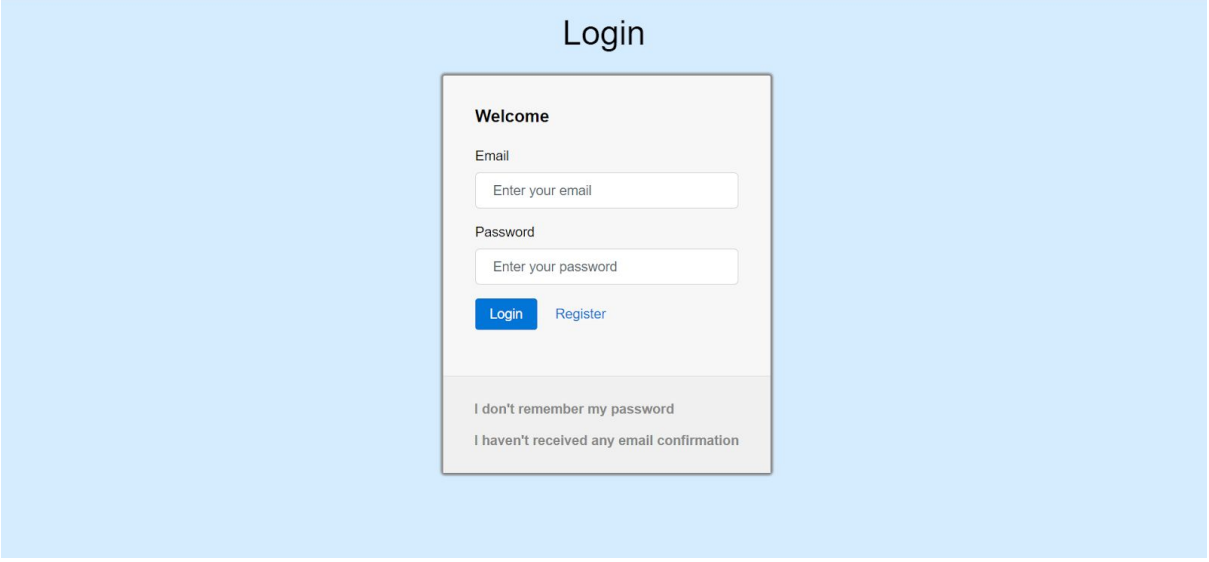


Figura 17. Pantalla d'iniciar sessió.

Una altra cosa que s'ha modificat respecte a la versió anterior és com es mostren els errors. Anteriorment només es mostrava un missatge general que deia "hi ha hagut algun error". S'ha intentat que els errors fossin més concrets per tal d'ajudar a l'usuari, tal com es pot veure en la següent imatge.

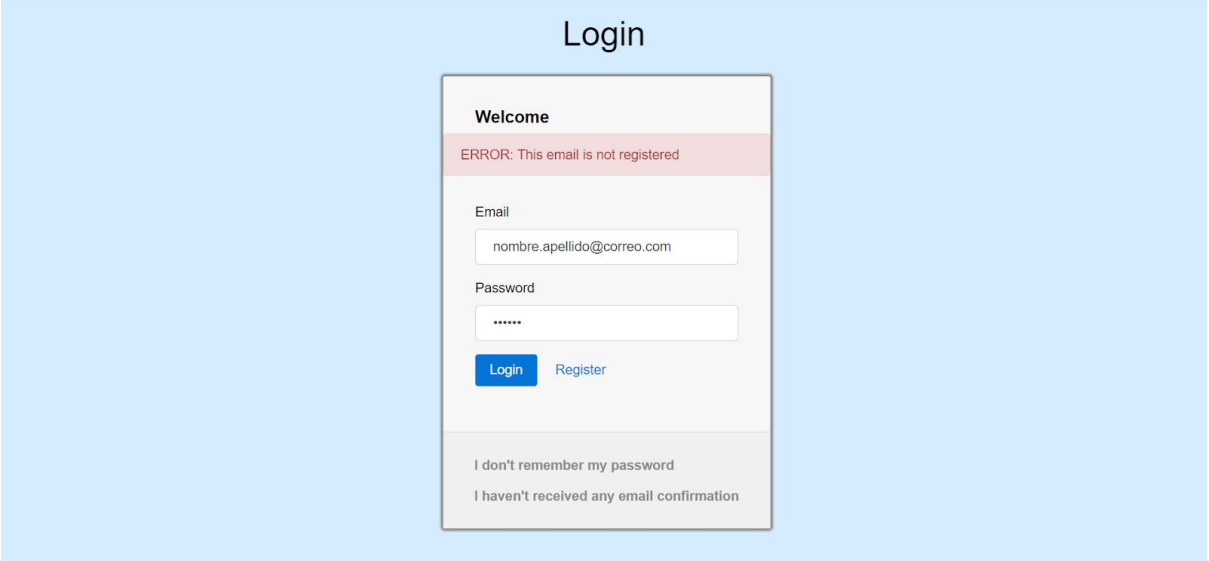


Figura 18. Pantalla d'iniciar sessió amb un error.

Pantalla de registre

Veiem com aquesta pantalla segueix la mateixa línia que la pantalla anterior. L'usuari ha d'introduir el seu nom, cognom, correu i contrasenya per registrar-se. Un cop ho ha fet, s'envia un correu a l'usuari perquè pugui verificar el seu compte. A part de registrar-se, l'usuari té l'opció de tornar a la pàgina de registre mitjançant l'enllaç.

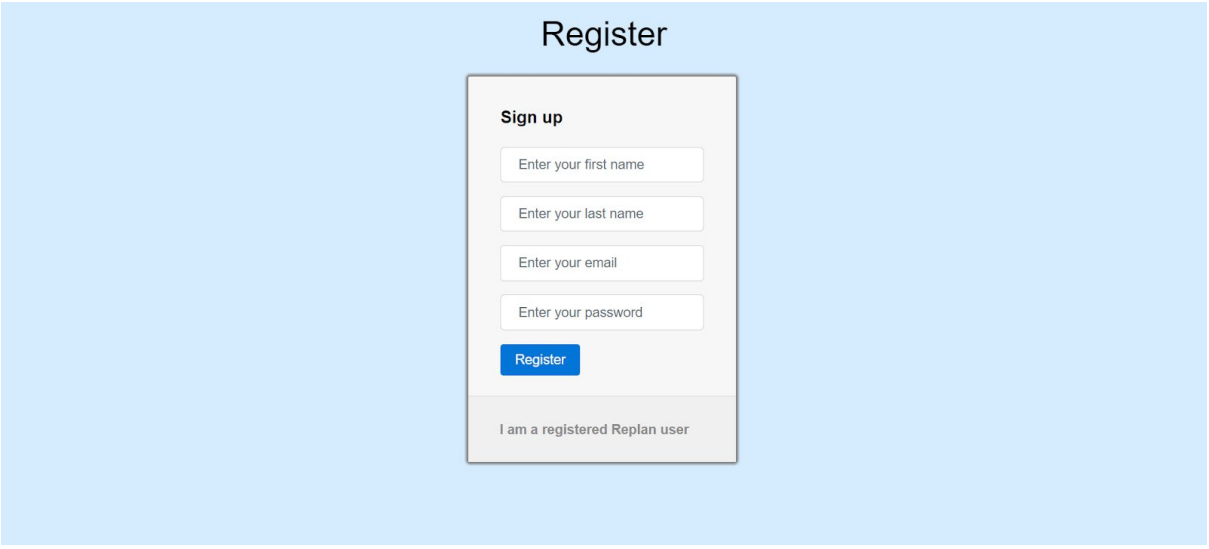


Figura 18. Pantalla d'iniciar sessió amb un error.

Pantalla de recuperar contrasenya

Aquesta pantalla serveix per recuperar la contrasenya. Intenta seguir la mateixa línia de disseny que l'anterior. L'usuari ha d'introduir el seu correu i pitjar el botó per tal de rebre una nova contrasenya d'usuari al seu correu.

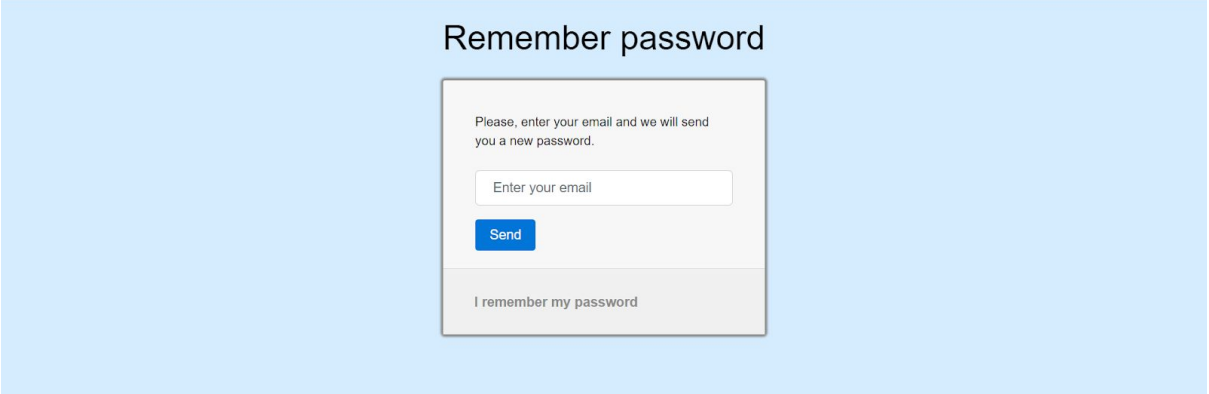


Figura 19. Pantalla de recuperació de la contrasenya.

Pantalla de sol·licitud de validació

Aquesta pantalla serveix per sol·licitar que es torni a enviar el correu electrònic amb l'enllaç per validar el compte.

L'usuari, si tot funciona correctament no hauria d'utilitzar mai aquesta pantalla, ja que quan es registri s'envia automàticament un correu amb l'enllaç per validar el seu compte. Però si pel que fos no el rebés, o elimines el correu sense voler, pot utilitzar aquesta pantalla per tornar a demanar-lo.

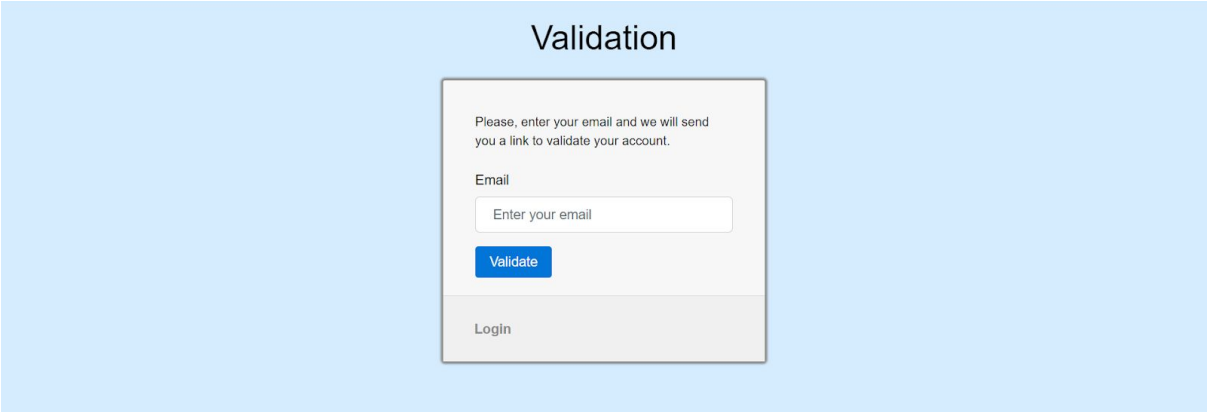


Figura 20. Pantalla de sol·licitud de validació del compte.

Pantalla de resposta a la validació del compte

Aquesta pantalla s'accedeix mitjançant un enllaç que es rep al correu electrònic. L'usuari simplement l'ha de pitjar i veure si s'ha validat correctament. Quan es mostra el missatge el compte ja està preparat per poder iniciar sessió.

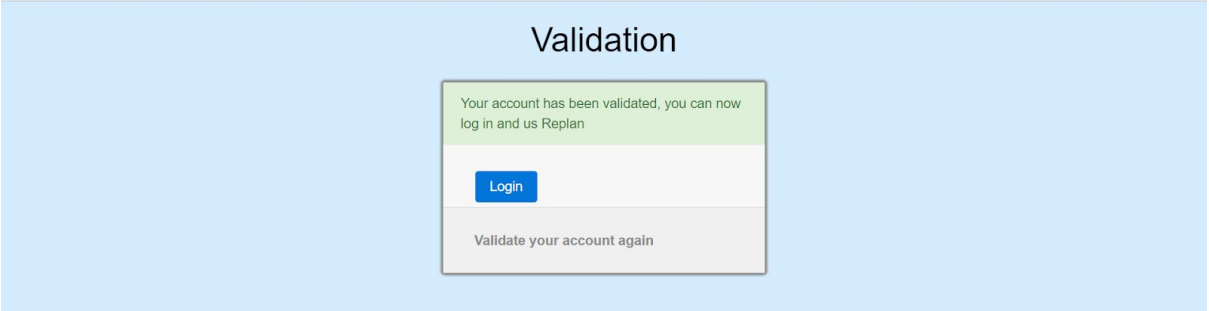


Figura 21. Pantalla de resposta de validació del compte.

Pantalla d'inici quan s'inicia sessió

Veiem que aquesta pantalla és completament diferent de la que hi havia a la versió anterior (figura 6). De fet, es va fer des de zero, sense reaprofitar res del disseny de la pantalla anterior. En primer lloc, veiem que el menú canvia. Hi ha tres opcions, una que és la que veiem a la següent figura on es veuen tots els projectes als quals participa un usuari, una altra que és on l'usuari pot modificar les dades del seu perfil, i per últim un botó per tancar sessió. La resta de pantalla també és molt diferent. S'aprofita molt més l'espai, ja que abans només podies veure un projecte, mentre que ara pots veure'n més. A més a més, abans l'aplicació et mostrava tots els projectes, mentre que ara només et mostra aquells que t'interessen que són als que ets membre.

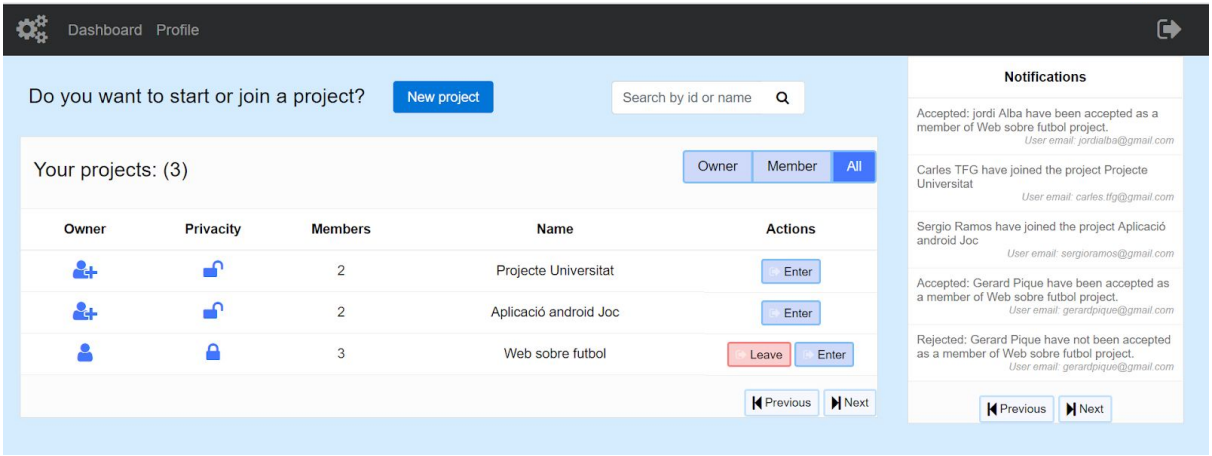


Figura 22. Pantalla d'inici quan s'inicia sessió.

Començant des de dalt, veiem que l'usuari pot crear un nou projecte pitjant un botó. Si ho desitja, pot utilitzar el cercador per buscar un projecte bé sigui pel nom del projecte o el ID si el sap. Un cop es pitja el botó de cercar, apareixen tots els projectes que coincideixen amb la cerca, i l'usuari si pot unir si és públic, o pot sol·licitar unir-se si és privat.

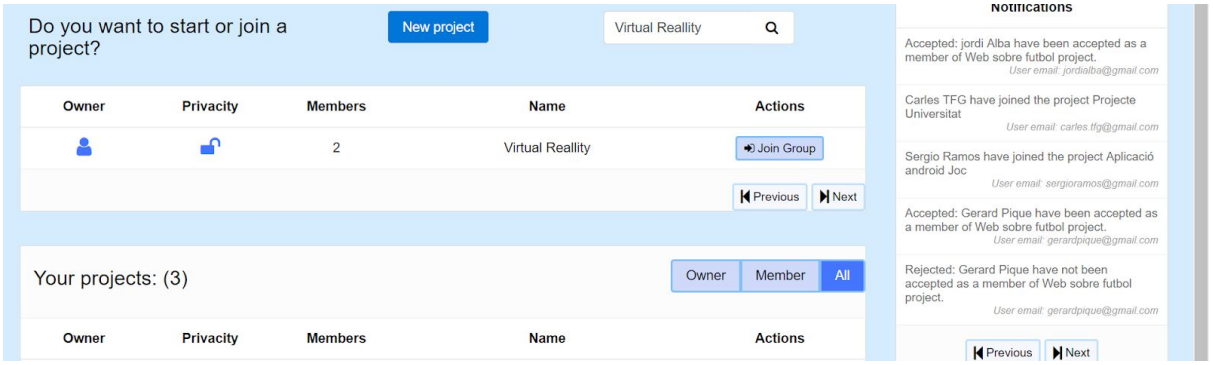


Figura 23. Pantalla d'inici quan s'inicia sessió i s'utilitza el cercador.

Quan es pitja al botó d'unir-se a un projecte privat, tots els membres reben una notificació nova, i des d'allà tenen l'opció d'afegir o no al membre.

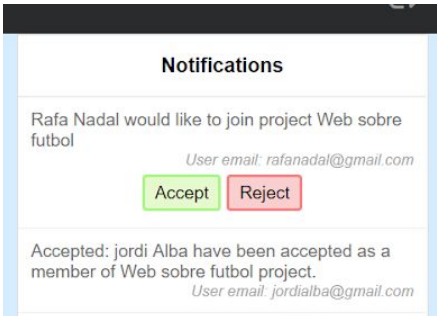


Figura 24. Notificació d'un usuari per unir-se a un projecte privat.

Sota del cercador veiem que apareixen una llista de projectes. Aquests són els projectes que has creat o bé que t'has unit com a membre. Es mostra la informació bàsica de cada un com són si ets el creador o no, si és públic, el nombre de membres i el nom. Per últim, per cada projecte pots o bé marxar o bé entrar per veure'l amb més detall. Si l'usuari ho desitja té l'opció de filtrar els projectes per veure només els privats o els públics.

Com a última novetat d'aquesta pantalla, veiem que a la part dreta hi apareixen les notificacions. És per allà on un usuari veu quins membres s'ha unit i quins no a cada un dels seus projectes.

Tant pels projectes com per les notificacions, es mostra un màxim de 5 files per no sobrecarregar a l'usuari amb informació. Si ho desitja, pitjant els botons d'enrere i endavant pot veure més informació.

Pantalles de perfil

Quan pitges a l'opció de perfil al menú superior, et mostra la informació relacionada amb el teu usuari, i t'ofereix un seguit de possibles accions que pots fer relacionades amb el teu compte. Totes segueixen un patró similar. Tenen un menú lateral que et mostra les diferents opcions, i a la resta de la pantalla es mostra el contingut corresponents. Menys la secció d'habilitats que l'analitzarem a part, la resta segueixen un patró similar. Mitjançant formularis l'usuari pot canviar-se el nom o cognom, la contrasenya, la disponibilitat i pot també afegir el nom d'usuari que té a Github i Trello (en cas de tenir-ne). Un cop omplerts els diferents caps, pitjant un botó pots canviar la informació de l'usuari.

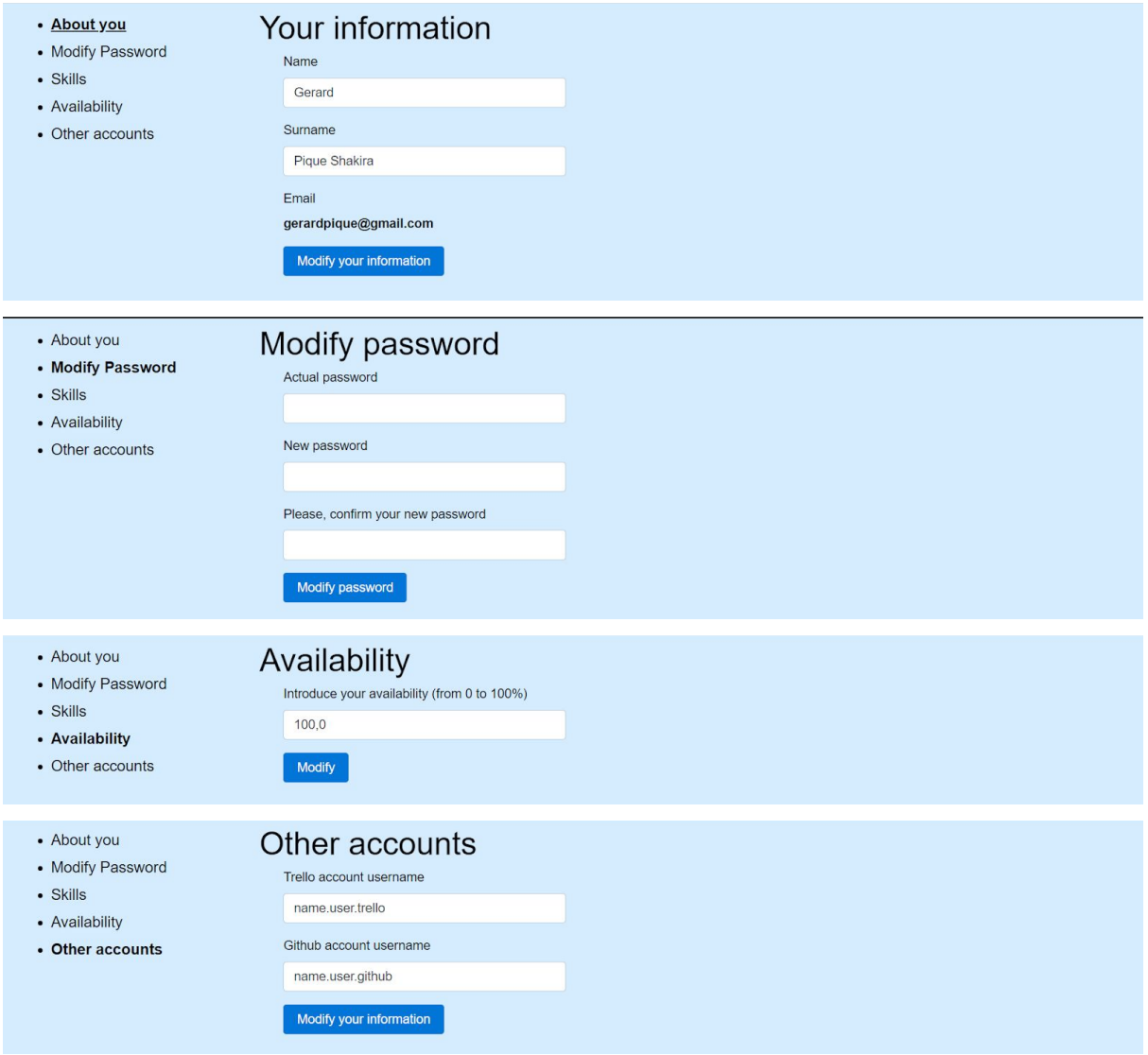
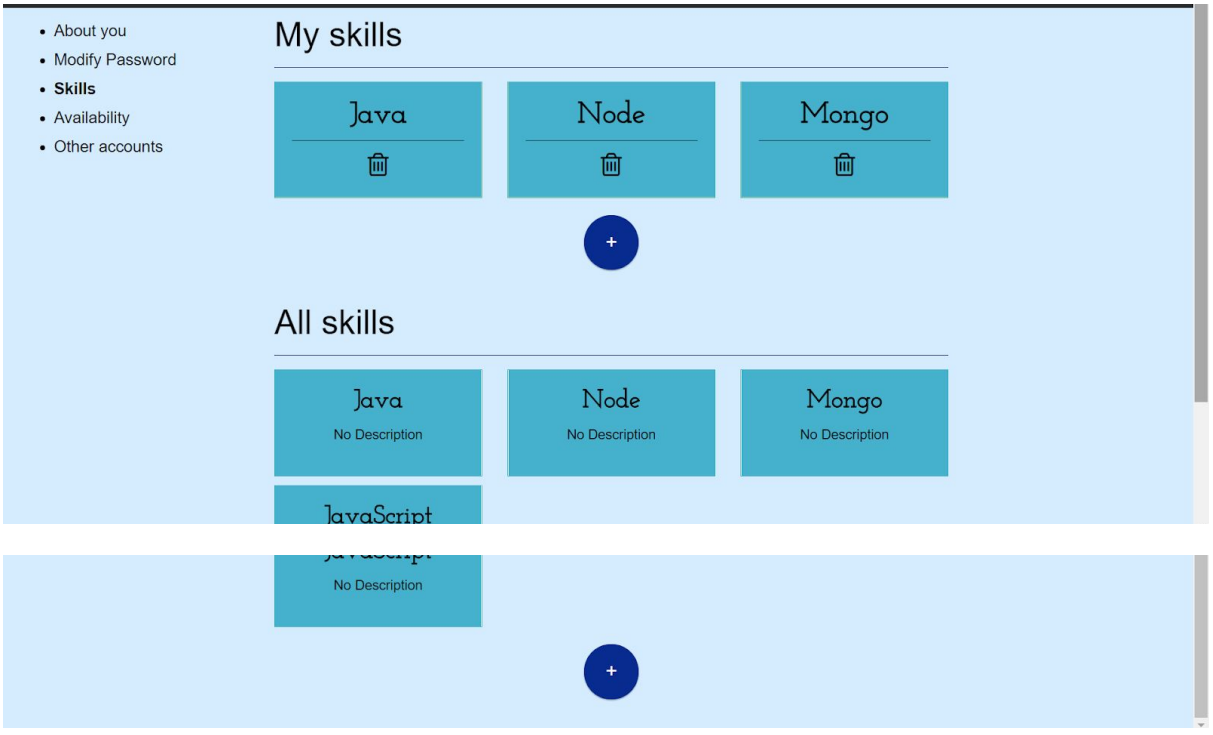


Figura 25, 26, 27 i 28. Diferents pantalles de les opcions de perfil.

Pantalla d'habilitats

Aquesta pantalla es troba dins la secció de perfil. Aquesta pantalla s'ha hagut de modificar també respecte a la pantalla inicial per dos motius. Un és que ara ja no hi ha Resources sinó que hi ha People, i la informació d'un usuari no es mostra tota en una pantalla com es feia abans, ja que ara hi ha molta més informació que només un nom i una descripció. El segon motiu és que ara les habilitats són globals, i això ha fet modificar també la manera de crear habilitats.

És aquí on l'usuari es pot afegir o eliminar habilitats que prèviament han d'estar creades. Si es vol donar d'alta una habilitat que no tenia prèviament l'ha de crear primer. Per evitar que es donin d'alta diferents habilitats que realment són la mateixa (com pot ser majúscules o minúscules, espais o errades ortogràfiques) s'ha afegit un recomanador que et suggereix una habilitat si el que estàs escrivint s'assembla a alguna activitat ja creada. A l'exemple que es pot veure en les següents figures, es veu com JavaScript ja està creat, i l'usuari intenta donar d'alta Java Script amb un espai.



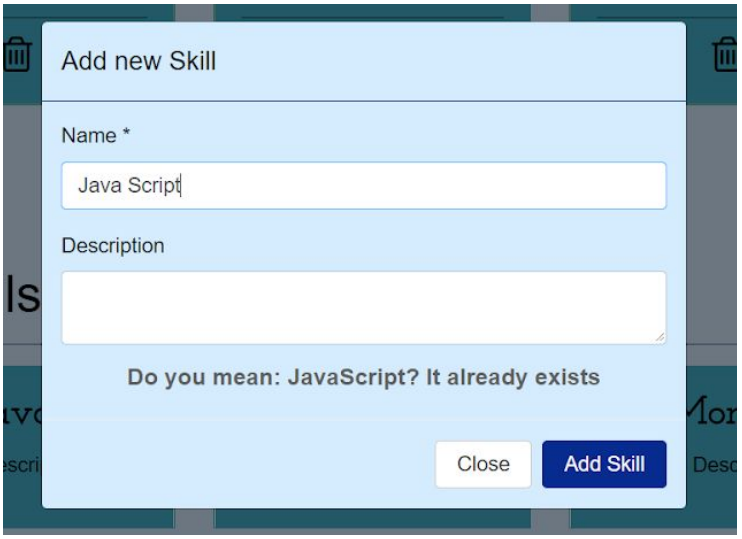


Figura 29, 30 i 31. Pantalla d'habilitats.

Pantalla de projecte

A aquesta pantalla s'accedeix mitjançant un botó que hi ha a la pàgina principal. Quan s'entra al projecte, la pantalla és bastant similar a la versió anterior. L'únic que s'ha afegit és que ara a la part superior apareix l'opció de modificar les característiques de l'aplicació, quan abans es mostrava dins de la secció de característiques. Pel que fa a la resta, s'ha mantingut el disseny anterior.

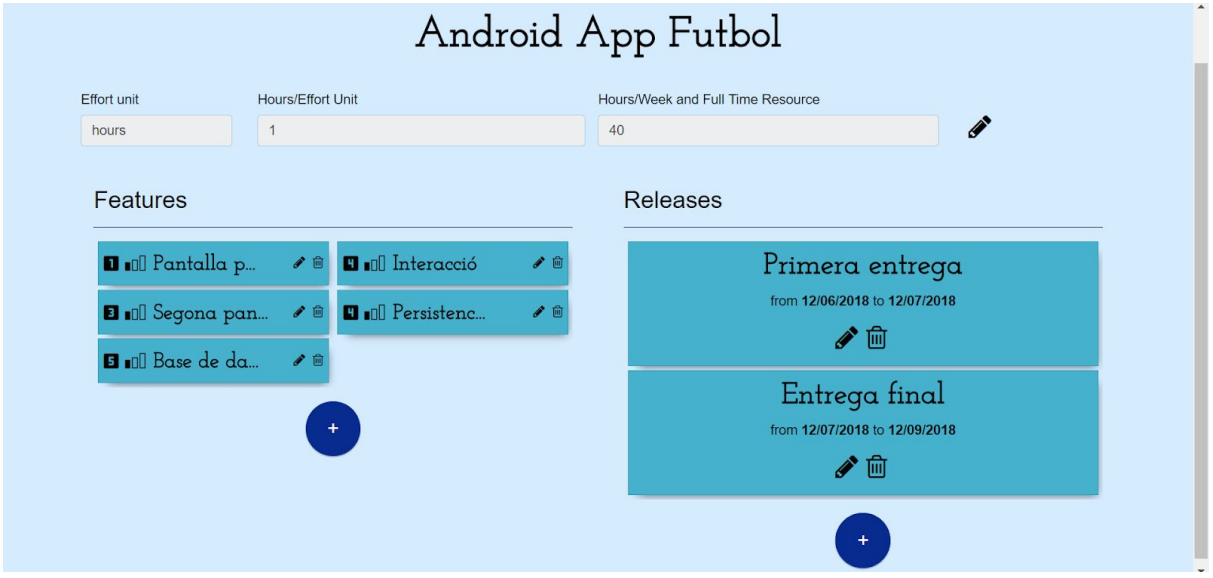


Figura 32. Pantalla de projecte.

Pantalla planificació

Igual que a la pantalla anterior, en general s'ha mantingut el disseny que hi havia a la versió antiga, ja que es considerava que la informació que es mostrava era suficientment clara. El que s'ha afegit, per fer-ho més personal per l'usuari, una secció a dalt de tot on es mostra quines són les tasques que ha de fer l'usuari. En la següent imatge es veu com primer es veu les tasques que ha de fer l'usuari Marc Bartra (que és l'usuari amb la sessió iniciada) i posteriorment ja es veu la feina de tot l'equip.

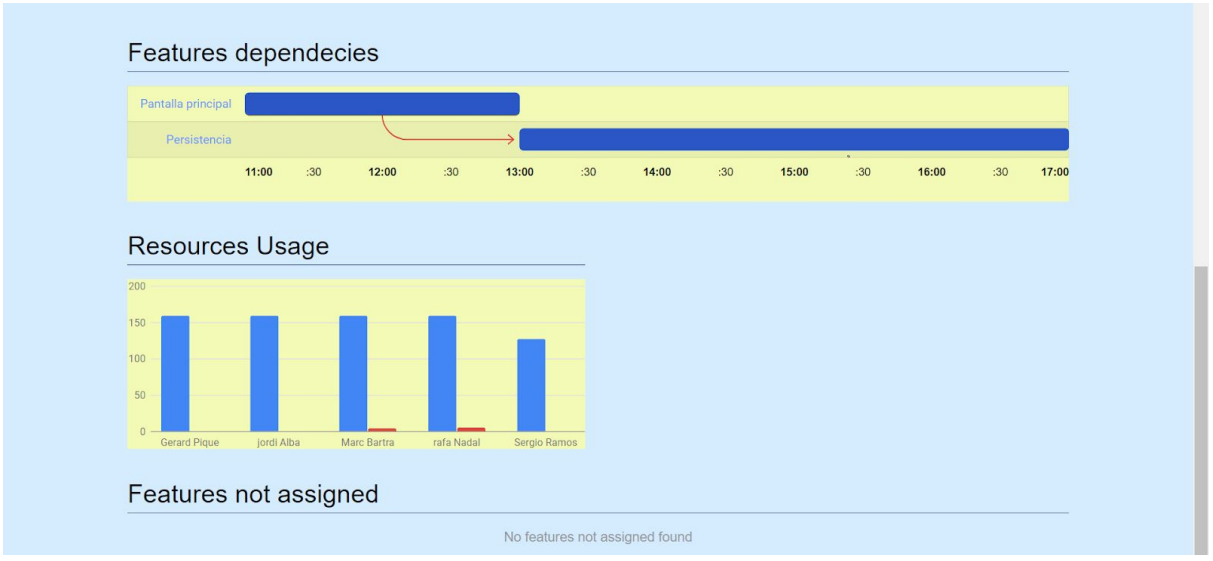


Figura 33 i 34. Pantalla de planificació.

8.3 Disseny de l'aplicació per l'administrador

Com ja hem comentat, hi ha diferents tipus d'usuari. Els més comuns seran els membres i creadors dels projectes, i les pantalles que aquests veuran són les que hem vist prèviament. Però pel que fa als administradors, hi ha petits canvis en la seva aplicació. En primer lloc, ells tenen el poder per poder esborrar habilitats globals cosa que els usuaris normals no

poden fer. La pantalla que veuen ells es tracta de la mateixa pantalla que hem vist prèviament, però s'inclou un botó per poder eliminar les habilitats.

Figura 35. Fragment de pantalla d'habilitats pels administradors.

Per altra banda, al menú que hem vist anteriorment, pels administradors apareix un enllaç que porta a la secció d'administrador.

Figura 36. Menú que veuen els administradors.

Si accedeixen a aquest menú, poden veure que tenen dues funcionalitats que la resta d'usuaris no tenen com és veure tots els projectes que hi ha creats i veure tots els usuaris que hi ha. Es mostra d'una manera similar que a la secció del perfil, per tal de mantenir la mateixa línia de disseny en tota l'aplicació. Al menú lateral es mostren les dues opcions, mentre que a la resta de pantalla es mostra en una taula la informació corresponent. Pel que fa a la vista de projectes, l'administrador té l'opció d'escollir com vol veure la informació, si bé veient el nom dels usuaris o el correu.

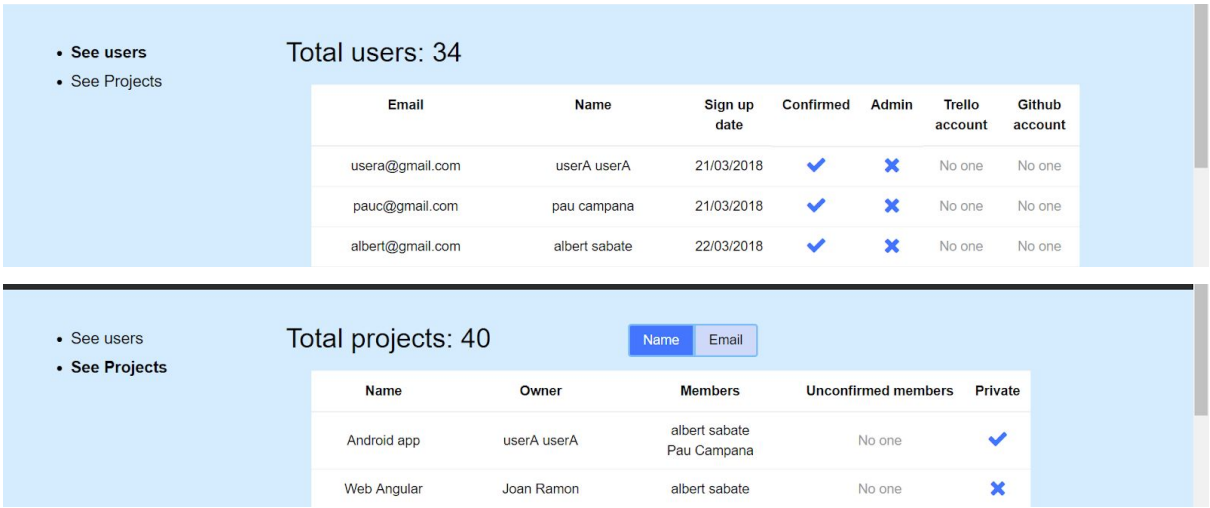


Figura 36 i 37. Pantalles de l'administrador.

9. Disseny tecnològic

9.1 Arquitectura de Replan

9.1.1 Abans del treball

Com ja hem comentat, Replan era una eina que ja funcionava abans que fes jo aquest treball. Així doncs he trobat convenient explicar breument com era el funcionament del sistema per així poder explicar els canvis que he fet. El sistema tenia dues grans parts. La primera era la del front-end que consistia en l'aplicació web en si, i la segona, la del back-end, que era la API que s'encarregava de tota la lògica de l'aplicació i la persistència necessària corresponent. Aquesta API està connectada amb un altre element, l'optimizer, que és l'encarregat d'executar la principal funcionalitat de l'aplicació. En la següent figura es pot veure com era l'arquitectura de Replan.

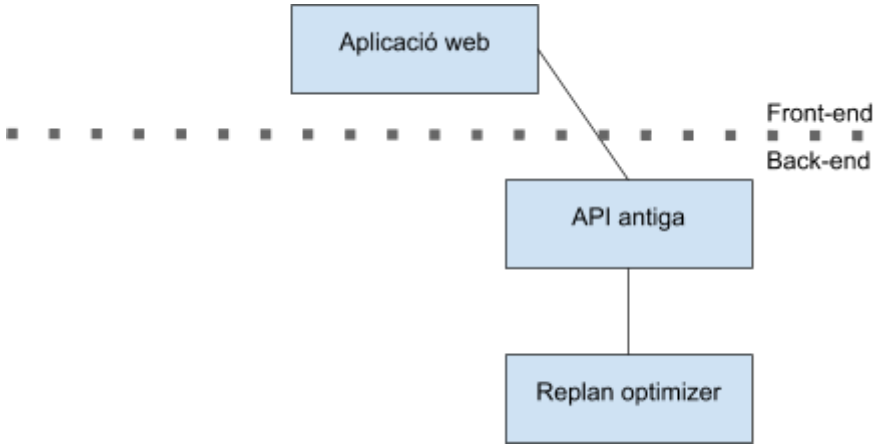


Figura 38. Arquitectura de Replan abans del meu projecte.

9.1.2 Després del treball

Per implementar les funcionalitats de multiusuari tenia principalment dues opcions. Una era modificar i ampliar la API antiga per tal que continues oferint les funcionalitats que tenia i afegir-ne més. L'altre era crear una API independent a aquesta des de zero que es dediques exclusivament a donar suport a les funcionalitats de multiusuari i la seva corresponent persistència. Es va acabar acordant amb el director i codirector fer una API nova, principalment per dos motius. Un d'ells era per separar la feina que feia de la que ja hi havia feta, i així tenir llibertat a l'hora d'escollir amb quina tecnologia ho volia fer. L'altre és perquè ja hi havia gent treballant amb una nova versió de l'API antiga, i treballar dues persones

sobre el mateix podia portar problemes després a l'hora d'ajuntar-ho. Així doncs, a l'arquitectura que hem vist anteriorment hi apareix un nou element que és la API multiusuari. L'altre element que he modificat durant el treball ha estat l'aplicació web que s'ha hagut d'adaptar. Els altres dos elements no s'han modificat. Cal dir que per mantenir la validesa de les dades de les dues API correctament, he hagut de fer algunes crides des de la API multiusuari a la API antiga. Un exemple és al registrar-se un usuari. Com que a la API antiga continuen existint els Resources, he de donar d'alta un Resource, i si tot funciona correctament, creo a la nova API una People que tindrà les dades que necessita i guardarà en un atribut el ID del Resource creat a l'altre API, per si en un futur es necessita.

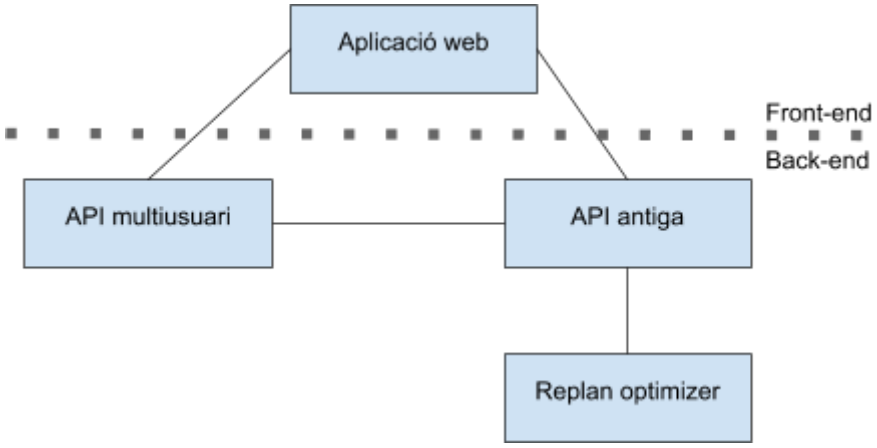


Figura 39. Arquitectura de Replan després del meu projecte.

9.2 Tecnologies utilitzades

Tant per fer l'aplicació web com per la API, hem utilitzat un seguit de tecnologies i eines que es detallen a continuació.

9.2.1 Angular 2

Aquest ha sigut el framework que s'ha utilitzat a l'hora de treballar en l'aplicació web. Es va decidir aquesta tecnologia perquè la pàgina web ja estava desenvolupada en Angular 2, i en segon lloc perquè continua sent una de les millors eines per fer pàgines web així que no es va trobar cap motiu per canviar de framework.

Una de les principals característiques d'Angular 2 és que treballa amb Components, que vindrien a ser un controlador que va lligat a una vista. Cada pantalla que l'usuari veu està formada per un Component o més, i cada Component té el seu fitxer HTML, CSS i TypeScript. Per aquest treball, el menú superior que l'usuari veu quan ha iniciat secció és un Component compartit entre les diferents pàgines. Així aconseguim que no hi hagi repetició de codi i sigui més fàcil i ràpid de canviar.

Pel que fa a la distribució dels arxius dins d'Angular 2, s'ha seguit utilitzant i ampliant la que ja hi havia feta. Algun dels arxius rellevants són els següents:

- App.components: dins d'aquest directori es troben tots els Components que hem explicat abans.
- Package.json: És un fitxer on e li defineixen característiques de com ha de compilar-se i desplegar-se el projecte. Pel cas del meu projecte no s'ha modificat, ja que estava correctament configurat.
- App.constants.ts: És on es guarden totes les constants de l'aplicació. S'ha utilitzat per guardar per exemple totes les URL per fer les crides a la API. És accessible des de tots els components de l'aplicació
- Carpeta Services: Aquesta carpeta conté tots els serveis que s'utilitzen a l'aplicació. Conté en primer lloc dos fitxers, replanAPIUser.service.ts i replanAPI.service.ts, per comunicar-se amb les dues APIs. Aquests contenen totes les crides a les API. També trobem en aquesta carpeta el fitxer globaldata.service.ts que s'utilitza per guardar algunes variables globals, i AuthentificationService.ts que s'utilitza per mantenir la sessió iniciada o bé per tancar-la.
- AuthGuard.ts: Aquest arxiu és l'encarregat de determinar si un usuari pot accedir a una determinada pàgina. Per exemple, si volem accedir a la pàgina de perfil, el que farà aquest arxiu és mirar si estem amb la sessió iniciada. Si és així podrem accedir sense problemes, però sinó ens portarà a la pàgina d'iniciar sessió.

Pel desenvolupament d'aquest projecte es va treballar en servidor local, i posteriorment quan ja funciona correctament es va desplegar l'aplicació perquè fos accessible per a tothom. Per treballar en local, es feia simplement des de la línia de comandes, accedint a la carpeta del projecte, i executant la comanda ng.serve. Amb això es desplegava l'aplicació en un entorn local, i així aconseguíem que poguéssim provat l'aplicació des de la web: http://localhost:4200/.

9.2.2 Visual Studio Code

Vaig fer servir aquest editor de text per desenvolupar l'aplicació web. El motiu principal va ser que ja tenia certa experiència amb el software. Aquest editor de text inclou bastantes funcionalitats que ajuden al programador.

9.2.3 Bitbucket

Va ser utilitzat com a repositori per a control de versions del projecte. Vaig triar aquesta aplicació que que era coneguda per mi i m'oferia tot el que necessitava pel projecte. Per començar el projecte vaig fer un clone del repositori del Sergi Orra que era qui habia desenvolupat la web per ultim cop. El enllaç del seu repositori és: https://bitbucket.org/sergiorra/replan. A partir d'aqui vaig anar millorant el software i utilitzant Bitbucket per guardar el codi font de l'aplicacio, tant per la pagina web com per la API. Es pot trobar el meu projecte a https://bitbucket.org/paucampana/tfg_replan/.

9.2.4 NodeJS, Express i MongoDB

Aquestes han estat les tecnologies principals que he utilitzat per desenvolupar la API. NodeJS és una plataforma o entorn d'execució per desenvolupar amb JavaScript al servidor, ExpressJS és un framework sobre NodeJS que permet treballar amb el protocol http i tenir sistemes de rutes i MongoDB és una base de dades NoSQL que té la principal característica que no treballa amb taules i relacions, sinó que ho fa amb uns arxius anomenats documents, i fan que sigui molt ràpid. Es va decidir treballar amb aquestes tecnologies ja que és una solució molt popular actualment per la seva eficiència, i també perquè així fèiem que tot el treball es fes amb un mateix llenguatge de programació JavaScript (Hem dit anteriorment que la part de la web la programàvem amb TypeScript, que és un llenguatge que amplia les funcionalitats de JavaScript, així que són molt similars). Té la avantatge que hi ha molta comunitat al darrere, i si necessitem implementar alguna cosa podem descarregar-nos llibreries que ofereixen funcionalitats vàries.

A continuació expliquem breument algunes de les carpetes i arxius que trobem en la nostra API:

- Carpeta models: És on definim els models, quins atributs tindrà, les restriccions i les relacions. En el nostre cas tenim tres models: Notification, Project i User. Per cada un definim un Shema, que és com s'anomena el model.
- Carpeta controllers: És on estan programades totes les funcionalitats, separades pels tres models: Notification, Project i User.
- Carpeta routes: És on estan totes les rutes. Quan la API rep una consulta, mira a quina funcionalitat està associada a aquella crida i la redirigeix.

- auth.js: És l'encarregat de dir si la crida l'està fent un usuari que ha iniciat sessió o no. Posteriorment ho explicarem amb més detall, però s'utilitzen tokens per veure quin usuari fa la crida.

9.2.5 Atom

Atom va ser l'editor de codi utilitzat per la part de la API. Vaig decidir utilitzar Atom i Visual Studio Code per poder així diferenciar que era del back-end i que era del front-end, ja que a l'hora d'implementar ho trobava més entenedor. Tot i això es podria haver usat només un dels dos editors de codi sense cap problema.

9.2.6 Postman

Postman és una eina que s'utilitza per provar les API i ajuda a determinar si una funcionalitat està funcionant com volíem. Al software se li proporciona l'URL per on vols fer la crida, el tipus (GET, POST, PUT, etc.), la capçalera i el cos, i en fer la crida retorna un resultat. Això ha sigut de gran utilitat per poder detectar on es produïen els errors en la fase de desenvolupament. Quan una funcionalitat no funcionava, gràcies a aquest software podíem determinar si el problema es trobava a l'aplicació web o pel contrari es trobava a la API.

També utilitzàvem l'eina per fer testos, ja que d'una manera fàcil i senzilla permet provar la crida amb valors d'entrada diferent.

10. Disseny de la base de dades

A continuació es mostra el disseny de la base de dades de l'aplicació, que en aquest cas està composta de tres taules, Project, Notification i User. La base de dades que hem utilitzat és MongoDB. Hem definit uns Shema (nom que s'utilitza a NodeJS per definir els diferents models que hi haurà) que comentem a continuació:

Project:

```
const ProjectSchema =  Schema({
  id_project: { type: String, unique: true},
  name: String,
  owner: { type: Schema.Types.ObjectId, ref: 'User' },
  members: [{ type: Schema.Types.ObjectId, ref: 'User' }],
  unconfirmedMembers: [{ type: Schema.Types.ObjectId, ref: 'User' }],
  is_private: Boolean
})
```

Figura 40. Shema de Project.

Podem veure en aquesta figura els camps del Shema de Project i els atributs que té. Podem observar diferents aspectes:

- La primera curiositat que veiem és que ens guardem un id_project tot i que cada Project que es registri al sistema tindrà un _id. Això ho fem perquè en aquest camp ens guardem l'identificador del projecte a l'altre API, que com ja hem comentat continuarà tenint Projects també.
- L'atribut name ja es conté a l'altra API, però s'ha trobat convenient emmagatzemar-lo també en l'API multiusuari. El motiu és que sinó a la pantalla principal on es veuen tots els projectes d'un usuari, s'hauria de fer una crida per cada projecte a l'API antiga per saber el nom.
- Veiem que tant el camp owner, com members com unconfirmedMembers fa referencia a un Shema User.

Notification:

```
const NotificationSchema =  Schema({
  text: String,
  is_simple: Boolean,
  email: String,
  signupDate: {type: Date, default: Date.now()},
  user: { type: Schema.Types.ObjectId, ref: 'User' },
  project: String,
})
```

Figura 41. Shema de Notification.

Aquí veiem que una Notification té un User i un Projecte. No hem utilitzat Shema.Types.ObjectId per fer la referència a un projecte perquè estem utilitzant la identificació de la API vella, per si ens hem de comunicar amb ella. També ens guardem una data per poder saber quan s'ha donat d'alta cada notificació.

User:

```
const UserSchema = new Schema({
  email: { type: String, unique: true, lowercase: true},
  displayName: String,
  displaySurname: String,
  resource: String,
  password: String,
  signupDate: {type: Date, default: Date.now()},
  lastLogin: Date,
  confirmed: {type: Boolean, default: false},
  isAdmin: {type: Boolean, default: false},
  trelloAccount:  {type: String, default: ""},
  githubAccount: {type: String, default: ""}
})
```

Figura 42. Shema de User.

Aquesta és l'últim Shema. Veiem com per defecte un usuari no és administrador i tampoc té el compte confirmat. Ens guardem també una data de registre per saber quan s'ha registrat cada usuari.

11. Gestió d'usuaris

Ja que el projecte tractava de fer l'aplicació multiusuari, he trobat adient fer un apartat que ho tractes amb profunditat.

11.1 Procés de registre

Un usuari necessita iniciar sessió si vol poder utilitzar Replan, i per iniciar sessió, primer ha d'haver-se registrat i confirmat el compte. A continuació detallem els passos a seguir per un usuari:

1. L'usuari es registre proporcionant un nom, un cognom, un correu i una contrasenya. Si un usuari intenta registrar un correu que ja està donat d'alta, el software mostrarà un error per pantalla. Si les dades proporcionades són correctes, la API registrarà a l'usuari, i enviarà un mail a l'adreça que ha proporcionat amb un enllaç per verificar el compte. Si no verifica el compte, tot i que l'usuari estarà registrat a la base de dades no podrà iniciar sessió, ja que l'usuari comprova si el compte ha estat verificat. S'utilitza la verificació per correu electrònic perquè així assegures que un usuari ha proporcionat el seu correu, i que no s'ha equivocat en escriure'l.
2. L'usuari rep un correu que li dóna la benvinguda a Replan, i demana a l'usuari que confirmi el compte. Quan l'usuari pitja l'enllaç, es crida a una funció de la API que confirma el compte. Això internament el que fa és canviar l'estat d'un boolean. El programa mostrarà un missatge a l'usuari conforme el seu compte ha estat validat i pot ja iniciar sessió. Si hi hagués algun error durant la confirmació, el programa indicaria les instruccions per tornar a rebre el correu de validació.
3. Un cop ja ha confirmat el compte, l'usuari ja pot iniciar sessió al sistema. Proporcionant correctament el seu correu i contrasenya ja podrà usar Replan.

11.2 Ús de JSON Web Token

Per tal de mantenir la sessió iniciada un cop l'usuari ha iniciat sessió, s'utilitza JSON Web Token. El funcionament és el següent. L'usuari s'autentifica a l'aplicació, i quan ho fa la API retorna un token que es guarda al costat del client, a la base de dades local del navegador. A partir de llavors cada petició HTTP que faci l'usuari va acompanyada del token a la capçalera de la crida. Aquest token és una firma xifrada que permet a la API identificar a l'usuari. Per cada crida que es fa, la API s'encarrega de descifrar el token i fer la

funcionalitat corresponent. Si el token no apareix a la capçalera d'una crida, o és erroni retorna un missatge d'error al client.

11.3 Tipus d'usuari

Quan un usuari es registra, ho fa com usuari normal, i per cada projecte que té fa un rol de creador o de membre. No hi ha cap manera que a partir de la web es pugui convertir en administrador. Per fer-ho s'ha d'entrar a la base de dades i convertir aquell usuari a administrador. Es fa així per un tema de seguretat, ja que no es vol que qualsevol persona es pugui canviar el tipus d'usuari.

12.Seguretat

S'ha intentat tenir cura de la seguretat en tot moment. Aquest són els aspectes claus:

- Token: Com hem comentat anteriorment, s'utilitzen tokens per saber que hem iniciat sessió. S'ha utilitzat aquesta alternativa ja que és més segura que guardar en el client el correu i la constrasenya, ja que algú podria doncs fàcilment veure les credencials de l'usuari. Una altra alternativa que es va considerar era tenir una base de dades on es guardessin els usuaris que tenien la sessió iniciada, però es va valorar que era una solució massa costosa i que no oferia cap avantatge respecte als tokens.
- Comprovació de permisos a la API: Quan la API rep una crida, verifica si l'usuari que la fa té els drets necessaris. Això ens permet tenir més seguretat, ja que si per exemple un usuari normal aconseguís fer una crida a la API que requereix permisos d'administrador, la API detectaria que no té els permisos i no ho faria.
- Encriptació de la contrasenya: Quan un usuari es registra, abans de desar la informació de l'usuari a la base de dades s'encripta la contrasenya. Es fa utilitzant bcrypt, una funció de hashing de contrasenyes. Així aconseguim que si algú aconseguís veure la informació de la nostra base de dades no podria saber les contrasenyes.

13. Identificació de lleis i regulacions

La llei que podria afectar a l'aplicació podria ser la següent:

"La Llei Orgànica 15/1999 de 13 de desembre de Protecció de Dades de Caràcter Personal, (LOPD), és una llei orgànica espanyola que té per objecte garantir i protegir, pel que fa al tractament de les dades personals, les llibertats públiques i els drets fonamentals de les persones físiques, i especialment del seu honor, intimitat i privacitat personal i familiar."

Per tal de poder analitzar correctament si estem complint aquesta llei, estudiarem les diferents parts que el componen:

Aplicació web: Aquesta s'encarrega de gestionar les dades que obté d'alguna de les APIs. Per aquesta part, no hi ha cap normativa que o llei que afecti l'aplicació. L'aplicació web, per tal de mantenir la sessió activa i no haver d'obtenir cada cop la informació de l'usuari, es guarda els atributs de l'usuari en la base de dades del navegador, en local. L'API ja gestiona que quan es vol obtenir l'usuari es retorni tota la informació menys la contrasenya. Amb això s'aconsegueix mantenir la privacitat de l'usuari i no incomplir cap llei.

API multiusuari: S'encarrega d'enviar les dades des del servidor fins a l'aplicació web. Aquí sí que tenim dades privades de l'usuari que no han de ser accessibles per la gent. És per això que les crides a funcions de l'API que retornin informació de l'usuari no són accessibles per tothom. La API només retornara la crida amb la informació corresponent si la crida es fa amb el token de seguretat corresponent a un usuari registrat.
En registrar-se un usuari, la contrasenya s'encripta abans de desar-la a la base de dades amb l'algoritme HS256. Amb això fem que si algú aconseguís entrar a la base de dades, no podria veure les contrasenyes dels usuaris, sinó que només les podria veure encriptades.

14. Usabilitat

Per tal de poder justificar el requisit no funcional de la usabilitat, hem trobat adient fer un test d'usabilitat per saber l'opinió de la gent. A continuació es mostra el test, les respostes obtingudes i els resultats que hem extret.

14.1 Test d'usabilitat

Aquest va ser un test que vam fer a 10 persones per així poder saber l'opinió de la gent. Era gent sense cap tipus de coneixement previ sobre el funcionament de l'aplicació. Per cada test que fèiem, es començava explicant breument de què tractava el projecte i perquè servia el software, i a partir d'aquí se li entregava el test, que havia d'intentar resoldre sense cap ajuda. A continuació es mostra el test que vam fer.

Test d'usabilitat sobre Replan:

1. Registra un usuari i inicia sessió proporcionant les teves dades reals.
2. Canvia el teu nom i cognom per les següents dades:
   NOM: Pau
   COGNOM: Campaña
3. Canvia la teva disponibilitat per les següents dades:
   DISPONIBILITAT: 50%
4. Afegeix un compte de Trello amb les següents dades:
   COMPTE DE TRELLO: trellouser
5. Crea una habilitat general amb les següents dades:
   NOM: Android
   DESCRIPCIÓ: Especialitzat en jocs.
6. Afegeix-te les dues següents habilitats:
   HABILITAT 1: Java
   HABILITAT 2: Android
7. Eliminar-te la següent habilitat:
   NOM: Java
8. Crea un nou projecte amb les següents dades:
   NOM: Test project
   UNITAT D'ESFORÇ: Hores
   HORES PER UNITAT D'ESFORÇ: 1

DISPONIBILITAT SETMANAL EN HORES: 40

DESCRIPCIÓ: Això és un test.

9. Uneix-te al projecte amb les següents dades:

   NOM: Web de futbol

   * És un projecte privat, així que jo des d'un altre ordinador acceptaré la seva sol·licitud.

10. Accepta la sol·licitud d'un usuari per unir-se al projecte anterior amb les següents dades:

    NOM: Joan

11. Rebutja la sol·licitud d'un usuari per unir-se al projecte anterior amb les següents dades:

    NOM: Pere

12. Entra al projecte Web de futbol i crea les següents features:

Nom	Esforç	Prioritat	Deadline
Pantalla d'inici	10	3	15/06/2018
Connexió a internet	20	4	17/06/2018
Base de dades	25	2	21/06/2018

13. Afegeix a la feature Base de dades com a dependència Pantalla d'inici.
14. Crea una release amb les següents dades:

    NOM: Entrega 1

    DATA INICI: 2/06/2018

    DATA FI: 4/07/2018

    DESCRIPCIÓ: Petita demo

15. Assigna les features creades anteriorment a la release.
16. Visualitza la planificació de la release.

Full de respostes del test d'usabilitat:

TASCA	Molt dificil	Dificil	Normal	Fàcil	Molt fàcil
1					
2					
3					

4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Taula 10. Plantilla per valorar el test.

* Els textos darrera aquest símbol (*) no es mostraven a la persona fent el test:

14.2 Resultats

A continuació mostrarem les respostes més votades per cada tasca i posteriorment ho analitzarem.

TASCA	Molt dificil	Dificil	Normal	Fàcil	Molt fàcil
1				**x**	
2					**x**
3					**x**
4				**x**	
5			**x**		
6					**x**
7					**x**

8					**x**
9			**x**		
10				**x**	
11					**x**
12					**x**
13				**x**	
14					**x**
15			**x**		
16					**x**

Taula 11. Resultats del test.

Una primera conclusió que podem treure és que els resultats han estat positius. Tot i això, hem vist com alguns usuaris presentaven alguns problemes per trobar algunes funcionalitats. Els problemes més comuns han estat:

- 1 usuari ha provat d'iniciar sessió sense haver confirmat el compte abans. A l'intentar-ho, s'ha mostrat un missatge que deia que havia de confirmar el compte. Seguidament ha anat al correu per confirmar el compte i ho ha fet.
- 2 usuaris no han trobat el projecte Web de futbol en un primer intent perquè ho havien escrit amb minúscules. Al següent intent ho han escrit correctament i ho han trobat
- 1 usuari ha intentat crear una habilitat general des del botó on un usuari s'afegeix una habilitat. Ha vist que el sistema no li permetia entrar un text, i s'ha adonat que no es feia des d'allà. Posteriorment ho ha aconseguit fer.
- A 2 usuaris els ha costat veure que per afegir features a una release s'havien d'arrossegar.
- En general ha costat entendre que significava que una tasca depengues d'una altra.

A partir d'aquests resultats i problemes que hem pogut apreciar, podem veure diferents llocs per on es podria millorar l'aplicació. Per aquest treball no s'ha disposat de temps per fer-ho, ja que el test d'usabilitat s'ha realitzat pocs dies abans de l'entrega.

Una primera millora seria que el cercador de projectes fos més permissiu. Actualment el text entrat ha de ser idèntic al nom del projecte. No pots equivocar-te ni amb les majúscules, ni amb espais i amb l'ortografia. Una possible millora seria que el cercador oferís com a

projectes trobats aquells que tinguessin com a títol un text semblant al que l'usuari ha entrat. Així aconseguiríem resoldre el problema que hem descrit anteriorment.

Fer que les features s'hagin d'arrossegar per afegir-les a una release és fàcil i ràpid pels usuaris que ja ho saben fer, però pels que usen Replan per primer cop no. Una solució podria ser mostrar un petit tutorial de com funciona l'aplicació als nous usuaris.

15. Desplegament del projecte

Per fer el projecte es va treballar primer en local, i un cop es va veure que estava tot implementat, i que era una versió estable i sense errors, es va decidir penjar-ho al Cloud. Indicarem un seguit de passos per explicar com desplegar el projecte.

15.1 En local

Com ja hem començat aquest treball consta de dues parts, una web i una API. La API la vam crear des de zero, i la web es va continuar del repositori de Sergi Orra, que havia fet un treball final de grau prèviament sobre Replan i tenia l'última versió del web. Ho vam fer fent una còpia del repositori https://bitbucket.org/sergiorra/replan. A continuació es detallen els passos que se seguien per desplegar el projecte.

15.1.1 Pàgina web

1. Descarregar NodeJS a l'ordinador. Es pot fer des de la seva pàgina web: https://nodejs.org/es/download/.
2. Clonar el projecte del repositori. En el repositori s'hi troba tant la API com la pàgina web. Aquest repositori es pot trobar en el següent enllaç: https://bitbucket.org/paucampana/tfg_replan.git.
3. Des d'un terminal anar a la carpeta arrel del repositori clonat, i entrar a la carpeta /app_replan. Aquesta és la carpeta on hi ha tota la web. Des del terminal, executa la comanda npm install, que automàticament farà que s'instal·lin totes les llibreries que fan falta pel projecte.
4. Un cop s'hagi acabat d'instal·lar tot, ja podrem desplegar la pàgina web en local. Per fer-ho, obrim un terminal a /app_replan i des d'allà executem la comanda ng serve. Ja tindrem disponible Replan a http://localhost:4200/.

15.1.1 API multiusuari

Per fer corra l'aplicació anterior i que funcioni amb normalitat, la API ha d'estar en execució. Aquests són els passos per executar-la.

1. Descarregar NodeJS a l'ordinador. Es pot fer des de la seva pàgina web: https://nodejs.org/es/download/. (Si ja s'ha fet anteriorment no fa falta tornar-hi).

2. Descarregar MongoDB a l'ordinador i segueix les instruccions que es proporcionen a la seva pàgina web: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/
3. Clonar el projecte del repositori. En el repositori s'hi troba tant la API com la pàgina web. Aquest repositori es pot trobar en el següent enllaç: https://bitbucket.org/paucampana/tfg_replan.git. (Si ja s'ha fet anteriorment no fa falta tornar-hi).
4. Des d'un terminal anar a la carpeta arrel del repositori clonat, i entrar a la carpeta /api. Aquesta és la carpeta on hi ha tota la API. Des del terminal, executa la comanda npm install, que automàticament farà que s'instal·lin totes les llibreries que fan falta pel projecte.
5. Un cop s'hagi acabat d'instal·lar tot, ja podrem desplegar la API en local. Per fer-ho, primer haurem d'obrir un terminal a /api per obrir la base de dades MongoDB. Això es fa amb la comanda mongod --dbpath C:/RUTA_DE_LA_BD.
6. Per acabar, obrim un terminal a /api i des d'allà executem la comanda npm start. Ja tindrem disponible la API a la ruta: http://localhost:3001/.

15.2 Al Cloud

Un cop es va acabar de desenvolupar l'aplicació, es va decidir penjar-la a internet perquè així estigues disponible per la gent. Per fer-ho, s'ha utilitzat Heroku, que és una de les eines més utilitzades en l'actualitat que permet desplegar aplicacions de manera gratuïta.

Per fer-ho, en primer lloc vam haver de penjar la API, que podem trobar disponible a la URL https://api-rest-replan-multiuser.herokuapp.com/api. Per la API, vam fer servir Swagger per fer la documentació ja que és una eina que et permet provar totes les crides. La API està disponible a https://app.swaggerhub.com/apis/paucampana/APIRelplanMultiuser/v1.

Seguidament, vam haver de canviar algunes constants de la nostra pàgina web perquè tinguessin la direcció de la API penjada a Heroku, i llavors ja vam poder també penjar la pàgina web. Aquesta està disponible a: https://web-replan-multiuser.herokuapp.com.

16. Implementacions futures

Pel que fa al software resultant del treball, tot i que compleix amb l'objectiu del treball, s'indiquen alguns punts que es podrien millorar en un futur per tal de fer millor Replan.

1. Oferir més funcionalitats a un administrador: Com hem vist, s'ha introduït en el sistema una figura d'administrador. Tot i que té més funcionalitats disponibles que la resta d'usuari, aquestes encara es podrien ampliar. Actualment les funcionalitats extres són les d'esborrar habilitats globals i de veure tots els Projectes i Usuaris. Però crec que altres funcionalitats podrien afegir-se per tal que l'administrador tingues més capacitats i poder en el software. Per exemple, poder eliminar usuari que estan fent un mal ús del sistema o eliminar projectes.
2. Restringir les capacitats de cada usuari per cada projecte: Actualment, tots els usuaris d'un projecte poden fer tots els canvis que desitgin. Això pot estar bé pels projectes on es confia en tots els membres. Però altres projectes potser trobarien més adient que només el cap de projecte pugues fer tots els canvis, i que a cada membre se l'hi assignessin uns permisos dels drets que té.
3. Implementar les millores suggerides del test d'usabilitat: S'ha intentat seguir una mateixa línia de disseny i fer que les funcionalitats fossin intuïtives i accessibles per l'usuari. Tot i això, a partir del test d'usabilitat hem pogut saber com podríem millorar Replan tal com hem explicat a l'apartat de resultats del test d'usabilitat.

17. Conclusions

Com a conclusió general podem dir que s'ha complert l'objectiu d'aquest projecte que era implementar la pàgina web per suportar a diferents usuaris.

Inicialment ens vam marcar uns objectius més específics per tal de poder dir que si es complien tots hauríem complert l'objectiu general. Aquests objectius s'han complert tots amb l'excepció de l'últim que era que els usuaris rebessin recompenses pel que anaven fent a Replan. Com hem explicat anteriorment, es va acordar amb el director i codirector del treball canviar aquest objectiu per afegir la figura de l'administrador. I com hem vist, aquest objectiu també s'ha assolit.

Podem dir doncs que hem millorat Replan fent el software multiusuari, ja que com hem vist a l'estudi de marcat tots els productes similars incorporen aquesta funcionalitat. També hem intentat millorar la interfície d'usuari mitjançant un millor tractament dels errors i oferint vistes agradables. Una vegada acabat el projecte, hem obtingut una eina que permet a l'usuari veure només aquella informació que és rellevant per ell, evitant així que perdi temps veient informació de projectes d'on no forma part. Oferim també una primera pantalla on es mostren tant els projectes de l'usuari com les notificacions perquè així només en entrar a la pàgina web ja pugui veure quin és l'estat dels projectes.

Per fer aquest projecte s'han hagut d'aplicar conceptes que hem anat veient durant el grau, i concretament durant l'especialitat d'enginyeria del software. Aquests conceptes han servit en primer lloc per poder fer un correcte model conceptual, per després poder ja poder fer el diagrama de casos d'ús, les bases de dades, i finalment tot el codi.

Cal dir que abans del projecte no havia fet servir en profunditat cap de les eines utilitzades per aquest treball així que vaig haver de fer una primera fase d'aprenentatge a les tecnologies Angular 2, NodeJS, MongoDB i el llenguatge de programació JavaScript.

Com he dit, crec que el treball ha estat finalitzat amb èxit, però encara hi ha coses de Replan que es poden millorar com s'ha dit a l'apartat d'implementacions futures.

17.1 Justificació de les competències tècniques

Per acabar les conclusions, he trobat adient fer un apartat on s'expliques com s'ha treballat cada una de les competències.

17.1.1 CES1.1: Desenvolupar, mantenir i avaluar sistemes i serveis software complexos i/o crítics.

Nivell esperat: En profunditat.

Ha sigut necessari veure i avaluar Replan, per tal de veure com funciona, quines parts té el sistema i com es comuniquen entre elles. També s'ha analitzat les funcionalitats que cal afegir-hi tenint en compte els objectius i l'abast del projecte. Aquestes s'hauran d'integrar al sistema Replan i haurà d'oferir tant les noves funcionalitats implementades, com les que ja existien prèviament.

Replan es pot considerar un sistema software complex, ja que es compon de vàries parts connectades entre elles com són la interfície web, la API de les funcionalitats existents fins ara, i la nova API que gestiona les funcionalitats d'usuaris. Perquè el sistema hagi acabat funcionant correctament, ha sigut necessari un enllaç correcte entre les diferents parts.

17.1.2 CES1.2: Donar solució a problemes d'integració en funció de les estratègies, dels estàndards i de les tecnologies disponibles.

Nivell esperat: En profunditat.

Com que el sistema consta de diferents elements, ha sigut important que la connexió entre ells es fes de manera correcta. S'han integrat diferents softwares i tecnologies en el sistema. La API dels usuaris s'ha fet sobre Node.js i utilitza com a base de dades MongoDB, que es programa en Javascript. Aquesta a vegades necessita comunicar-se amb la API que permetia totes les funcionalitats prèvies al treball, i rep les respostes a les crides en format JSON. Les crides a la API dels usuaris es fan des de AngularJS, que es programa en TypeScript. La API retorna a les crides els resultats en JSON, que cal processar per poder mostrar per pantalla a l'usuari, on la intefície web utilitza HTML, CSS i JavaScript.

17.1.3 CES1.3: Identificar, avaluar i gestionar els riscos potencials associats a la construcció de software que es poguessin presentar.

Nivell esperat: Una mica.

Ha sigut necessari identificar i avaluar els riscos associats al software que s'ha implementat. S'han identificat possibles riscos que pogessin sorgir i es va reservar un temps a la

planificació per poder igualment acabar el projecte en el cas que aparegués alguna complicació.

17.1.4 CES1.5: Especificar, dissenyar, implementar i avaluar bases de dades.

Nivell esperat: En profunditat.

Per la implementació de totes les funcionalitats de la API i poder guardar la informació, ha fet falta la creació de bases de dades. Ha fet falta especificar-les, dissenyar-les, implementar-les i avaluar-les. Ha sigut necessari decidir quina era una manera correcta de desar la informació i com s'haurien de connectar les bases de dades entre elles. El fet que alguna de les dades s'hagin d'accedir mitjançant una altra API, ha afegit complexitat al problema.

17.1.5 CES2.1: Definir i gestionar els requisits d'un sistema software.

Nivell esperat: Bastant.

S'ha hagut de definir i gestionar els requisits de les funcionalitats a desenvolupar, tenint en compte que partíem d'un sistema ja existent, pel que calia assegurar que fossin compatibles amb Replan.

18. Bibliografia

1. Trello.com. (2018). *Trello*. [online] Available at: https://trello.com/ [Accessed 5 Mar. 2018].
2. Slack. (2018). *Where work happens*. [online] Available at: https://slack.com/ [Accessed 5 Mar. 2018].
3. Atlassian. (2018). *Jira | Software de seguimiento de proyectos e incidencias | Atlassian*. [online] Available at: https://es.atlassian.com/software/jira [Accessed 5 Mar. 2018].
4. Redmine.org. (2018). *Overview - Redmine*. [online] Available at: http://www.redmine.org [Accessed 5 Mar. 2018].
5. Asana. (2018). *Utiliza Asana para gestionar tus proyectos y monitorear el trabajo de tu equipo · Asana*. [online] Available at: https://asana.com/es/ [Accessed 5 Mar. 2018].
6. Es.wikipedia.org. (2018). *MEAN*. [online] Available at: https://es.wikipedia.org/wiki/MEAN [Accessed 12 Mar. 2018].
7. Experteer.es. (2018). *¿Cuál es el salario medio en 2018? El comparador salarial de Experteer*. [online] Available at: https://www.experteer.es/salary_calculator/ [Accessed 8 Apr. 2018].
8. Color.adobe.com. (2018). *Adobe Color CC*. [online] Available at: https://color.adobe.com/create/color-wheel/ [Accessed 11 Jun. 2018].
9. Anàlisi de l'activitat dels desenvolupadors en plataformes de suport al desenvolupament col·laboratiu [Available at]: <http://www.essi.upc.edu/dameller/publications/TFG.miquel.xamani.pdf> [Accessed 11 Jun. 2018].