APLICACIO MÒBIL DE PRODUCTIVITAT EN GRUP BASADA EN EL MÈTODE POMODORO

TREBALL FINAL DE GRAU

Facultat d'Informàtica de Barcelona

Estudiant:

Enric Ayguadé Baró

Director:

Carles Farré Tost

26 d'octubre de 2015

Resum

Actualment la organització de tasques i temps ha pres gran protagonisme, ja que és ben sabut que l'èxit en aquests punts pot implicar l'èxit en el projecte en que s'està treballant. Existeixen moltes aplicacions que permeten gestionar tasques i moltes d'altres que permeten gestionar el temps. Si ens centrem en aquest últim, la Tècnica Pomodoro és un sistema que guanya adeptes però és, en la seva naturalesa, simple.

El que pretén aquest projecte, **TikTak Team**, és portar les funcionalitats de la Tècnica Pomodoro a la dimensió del treball en grup i les aplicacions de gestió de tasques, intentant unir el millor dels dos conjunts.

Resumen

Actualmente la organización de tareas y tiempo ha tomado un gran portagonismo, ya que es sabido que el éxito en estos puntos puede implicar el éxito en el proyecto en el que se está trabajando. Existen muchas aplicaciones que permiten gestionar tareas y muchas otras que permiten gestionar el tiempo. Si nos centramos en este último, la Técnica Pomodoro es un sistema que gana adeptos pero es, en su naturaleza, simple.

Lo que pretende este proyecto, **TikTak Team**, es portar las funcionalidades de la Técnica Pomodoro a la dimensión del trabajo en grupo y las aplicaciones de gestión de tareas, intentando unir el mejor de los dos conjuntos.

Abstract

Nowadays, the management of tasks and time has taken timelight, as it is known that the success in this points may mean the success in the project you are working on. There are many applications that allow you to manage tasks and many others that allow you to manage time. If we focus on the last one, the Pomodoro Technique is a system that is gaining followers but that is simple by it's nature.

What this project called **TikTak Team** wants to achieve is to bring the functionalities of the Pomodoro Technique to the teamwork and tasks managing dimension, trying to get the best of each set.

Índex de figures

Índex de taules

Índex de codis

Índex

Capítol 1

Introducció i estat de l'art

1.1 Context

Aquest projecte es realitza com a Treball de Final de Grau dels estudis de Grau en Enginyeria Informàtica, especialitat en Enginyeria del Software, a la Facultat d'Informàtica de Barcelona. La idea del projecte és millorar la productivitat de treballadors i que aquests estiguin més contents. El sistema es basa en la Tècnica Pomodoro, una tècnica en que el temps es divideix en torns. El primer torn és de 25 minuts i en ell es treballa en la tasca que sigui sense descansar i evitant distraccions. El segon torn és de 5 minuts i serveix per descansar. El tercer torn torna a ser igual que el primer, etc. La tècnica s'explica posteriorment de forma més detallada. En el que innova el meu sistema és en permetre crear grups de treball amb diferents usuaris, on cada grup tindrà els seus propis torns de treball, que seran visibles alhora per tots els membres. Aquesta solució que es proposarà vol que els treballadors treballin i descansin alhora, millorant el control de l'ansietat i la concentració. Per exemple, l'Arnau és el cap de programació d'una empresa i crea un grup de treball amb els seus programadors; programa un seguit de torns que són automàticament visibles per a tothom del grup. Quan estan en torns de treball, tots els empleats (i ell mateix) treballen i eviten distraccions, quan estan en temps de descans, tots ho fan alhora i poden conversar entre ells i relaxar-se. Per a la nostra aplicació mòbil de productivitat en grup basada en el mètode Pomodoro cal primer explicar alguns conceptes clau per entendre completament el context d'aquest TFG, que s'expliquen al glossari.

1.1.1 Glossari

- **Productivitat [2]**: Com a mesura econòmica, es defineix la productivitat com la relació entre la quantitat de productes obtinguts per un sistema productiu i els recursos utilitzats per obtenir aquesta producció. Segons la investopedia, els guanys de la productivitat són vitals per a l'economia perquè ens permeten aconseguir més amb menys. A més, es menciona que aquestes millores vénen d'avenços tecnològics (com ara ordenadors i internet), millores en la cadena de producció i logística, i per millora de les habilitats del personal.

- **Treball en equip [3]**: Definim treball en equip com el treball fet per diversos individus on cada un fa una part però tots tenen un mateix objectiu. Els 5 components del treball en grup són el lideratge, la mesura mútua del rendiment, backup behavior, adaptabilitat i l'orientació de l'equip. [4]

- **Control del Temps [5]**: Es coneix el *time management* com l'acte o procés de planificar i exercitar un control conscient sobre el temps dedicat a activitats específiques, especialment per incrementar

l'efectivitat, eficiència o la productivitat. Inicialment s'encarava només al món dels negocis, però va anar expandint-se també a l'àmbit personal. El time management és una combinació de processos, eines, tècniques i mètodes.

- **Tècnica Pomodoro [6]**: La Tècnica Pomodoro és un mètode per l'administració del temps desenvolupat per Francesco Cirillo a finals dels anys 1980. La tècnica utilitza un rellotge per dividir el temps dedicat a una feina en intervals de 25 minuts, que s'anomenen *pomodoros*, separats per pauses.

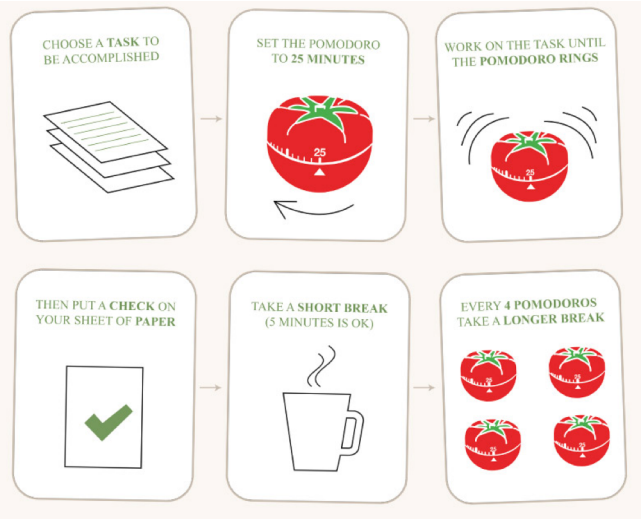


Figura 1.1: Funcionament de la Tècnica Pomodoro [1]

- **Sessió privada**: En el nostre sistema, considerem una sessió privada un conjunt d'un o més pomodoros individual, que no es comparteix amb ningú i que només és visible per l'usuari que l'ha creat.
- **Grup d'usuaris**: En el nostre sistema, un grup d'usuaris és un conjunt de persones que tenen un "espai"on posar-hi les seves sessions en grup.
- **Sessió en grup**: En el nostre sistema, considerem una sessió en grup un conjunt d'un o més pomodoros que es comparteix i pot ser visualitzada pels membres del grup al que pertany. Les sessions en grup les crea l'administrador del grup.
- **Configuració de sessió**: En el nostre sistema, és la configuració respecte als temps que té una sessió (ja sigui privada o de grup). En ella hi ha la durada del torn de treball, la durada del descans curt i la durada del descans llarg (que es fa cada 4 descansos curts).
- **Client [7]**: En informàtica, el client és una aplicació o ordenador que consumeix un servei remot d'un altre ordenador (servidor), habitualment a través d'una xarxa de telecomunicacions.
- **Servidor web [8]**: Un servidor web o servidor HTTP és un programa informàtic que processa una aplicació del costat del servidor, realitzant connexions bidireccionals i/o unidireccionals i sincrones o asíncrones amb el client i generant o cedint una resposta en qualsevol llenguate o aplicació del costat del client.
- **Framework [9]**: Un framework és una estructura de suport definit, normalment amb artefactes o mòduls de software concrets, que acostuma a incloure suport de programes, biblioteques, entre altres eines, per ajudar a desenvolupar i unir diferents components d'un projecte.

- **API (Application Programming Interface) [10]**: Una API és el conjunt de subrutines, funcions i procediments (o mètodes) que ofereix certa biblioteca per ser utilitzat per un altre software com una capa d'abstracció.
- **JSON [11]**: És un acrònim de *JavaScript Object Notation*, és un format lleuger per l'intercanvi de dades.
- **REST [12]**: REST és una tècnica d'arquitectura del software per sistemes hipermedia distribuïts com la World Wide Web.
- **Object Relational Mapping (ORM) [13]**: ORM és una tècnica de programació per convertir dades entre sistemes amb tipus incompatibles en els llenguatges orientats a objectes. Crea l'efecte d'una "base de dades virtual"que pot ser utilitzada des del llenguatge de programació.

  En el nostre cas utilitzarem el *loopback-datasource-juggler* a Loopback.io (servidor) i *Realm* al client Android, per guardar la informació local.
- **Software Development Kit (SDK) [14]**: Un SDK és un conjunt d'eines de desenvolupament de software que permeten la creació d'aplicacions per a un tipus de software, framework, plataforma de hardware, etc.

1.1.2 Actors implicats

A continuació, identificarem quines són les parts interessades en el nostre sistema. És a dir, aquelles persones, organitzacions i organismes que tenen algun interès en el sistema. Cadascuna d'aquestes parts tenen un interès per tal d'assolir els seus objectius (benefici que esperen aconseguir mitjançant l'ús o el funcionament del sistema i el preu màxim que estan disposats a pagar).

- **Emprenedors**: Els emprenedors que desenvolupem el sistema. El nostre interès és proporcionar una eina que augmenti la productivitat i benestar de les altres parts interessades, aconseguir un benefici econòmic a llarg plaç, desenvolupar un sistema que ens serveixi de mostra per al currículum i realitzar el TFG.
- **Director del projecte**: El director d'aquest TFG. Té interès en què el projecte es realitzi correctament, en ajudar el seu alumne i en ser reconegut per un bon treball.
- **Empresa**: Les empreses que utilitzaran l'aplicació. El seu interès és aconseguir una millora de la productivitat sense ressentir la qualitat de vida dels seus treballadors, motivant-los.
- **Treballadors**: Els treballadors que utilitzaran l'aplicació. Volen treballar més sincronitzats i poder descansar junts.
- **Altres grups de treball**: Els altres grups de treball, com ara estudiants, associacions, etc. podran beneficiar-se del servei de la mateixa manera, millorant la seva manera d'aprendre o desenvolupar noves idees.

1.2 Estat de l'art

Dividirem l'estat de l'art en l'estudi de les tecnologies que es podrien aplicar i l'estudi de serveis similars.

1.2.1 Estudi de tecnologies

Per al client de l'aplicació, existeixen com a opcions una aplicació nativa (majoritàriament les plataformes iOS i Android, seguides per Windows Phone), una aplicació web o una aplicació híbrida.

Aplicació mòbil nativa

L'aplicació nativa és la forma més habitual de consumir continguts al smartphone o tablet. Ofereix una experiència nativa (en sintonia amb la resta d'apps, fàcil d'interpretar per l'usuari), és més eficient en quant a gestió de recursos i ens facilita la gestió de notificacions. Ens permet tenir visibilitat a les botigues mòbils. A més, en un futur es podria extendre l'aplicació a wearables (rellotges intel · ligents). Com a contra, és necessària una app per a cada sistema operatiu o resignar-se a no desenvolupar la app per a tothom.

Aplicació web

Una *webapp* és una aplicació que s'executa en un navegador web. És més fàcil de desenvolupar i funciona en tot tipus de dispositius, a més de poder ser consumida a l'escriptori d'un ordenador convencional. Per contra, l'experiència d'usuari no és nativa i l'eficiència és ressenteix molt d'estar executant-se en un navegador. No tenim visibilitat a les botigues mòbils.

També existeixen les apps híbrides, que intenten ser un punt mig el suficientment decent.

S'ha decidit desenvolupar el client de l'aplicació en una app nativa Android. Gràcies a la seva quota de mercat ens dóna accés a una majoria d'usuaris, a més que l'accés al desenvolupament de la plataforma i desplegament a la *Play Store* és més econòmic.

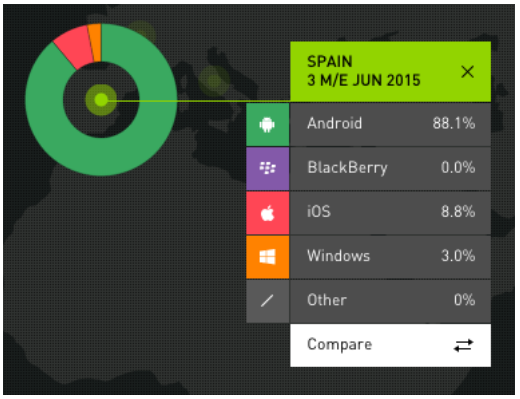


Figura 1.2: Quota de mercat d'Android a Espanya [2]

Quant a la tecnologia del servidor, tenim una gran quantitat de llenguatges i frameworks que s'adapten a qualsevol necessitat possible. En destaquen PHP, ASP.NET i Java. [15]

Per al servidor de l'aplicació s'ha decidit utilitzar el framework Javascript **Loopback.io**, que està construït sobre **express.js** i **node.js**. Especialment, la filosofia darrera el framework d'encarar-lo directament al desenvolupament d'APIs i l'objectiu d'evitar escriure codi innecessari em van fer decidir. A més, volia aprendre a utilitzar algun altre llenguatge i ja estava una mica familiaritzat amb Javascript de cara al client, a més que la popularitat de cara al servidor estava creixent i em semblava molt interessa aprendre'n. Tot i ser un framework força nou, té el suport de l'empresa StrongLoop, principals sponsors d'express.js, també hi ha alguns *commiters* de node.js. [16]

1.2.2 Serveis i filosofies similars

Actualment la productivitat és un tema de moda i n'existeix molta informació al respecte. La nostra aplicació podria ser utilitzada en un espectre molt ampli d'organitzacions i associacions que voldrien millorar la seva productivitat i organització. En la literatura podem trobar diverses propostes per a la gestió de tasques i del temps.

- **Anàlisi ABC [17]**: En informàtica, el client és una aplicació o ordenador que consumeix un servei remot d'un altre ordenador (servidor), habitualment a través d'una xarxa de telecomunicacions.

  Aquesta tècnica, utilitzada des de fa anys en el control de negocis, es basa en categoritzar informació en grups per prioritat, habitualment A, B i C; A: Tasques urgents i importants. B: Tasques importants però no urgents. C: Tasques urgents però no importants.

- **Principi de Pareto i anàlisi de Pareto [18]**: El Principi de Pareto diu que, per a molts events, el 80% dels efectes venen del 20% de les causes. En el context del time management tenim l'anàlisi de Pareto: es pot dir que el 80% de les tasques poden ser completades en el 20% del temps. Per tant, el 20% restant es completarà amb el 80% del temps. Habitualment, es combina l'anàlisi ABC i el de Pareto; si el 80% de la productivitat es pot aconseguir amb el 20% de les tasques, aquestes haurien de ser prioritzades com a 'A' en l'anàlisi ABC.

- **El Mètode Eisenhower [19]**: En el Principi de Decisió d'Eisenhower, es valoren totes les tasques utilitzant un criteri de important / no important i urgent / no urgent, per a després posar-les a una Matriu d'Eisenhower. Finalment, es realitzen les tasques d'acord amb l'ordre que es pot veure a la

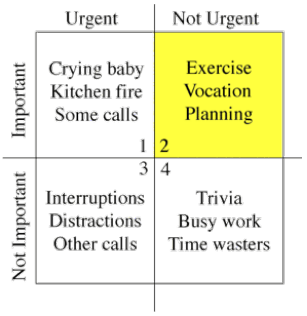


Figura 1.3: Matriu d'Eisenhower

  figura.

- **Tècnica Pomodoro**: La Tècnica Pomodoro que hem explicat a la contextualització. Existeixen aplicacions mòbils i d'escriptori que ajuden a utilitzar-la, de forma individual.

- **Getting Things Done [20]**: Literalment "Fer les Coses". La idea d'aquest mètode és fer immediatament les tasques petites i dividir les tasques grans en altres de més petites. La intenció que té aquest mètode és la de distribuir la informació i evitar "saturar-nos".

- **To-do List [21]**: En aquest mètode, s'acostuma a dividir les tasques entre To Do per a les que s'han de començar, Doing per a les que s'estan realitzant en aquest mateix moment i Done per a les acabades. És habitual utilitzar aquest mètode utiltzant una pissara amb post-its o utilitzant eines digitals.

- **Software de gestió**: En l'entorn del software que implementa millores i filosofies del time management és on encaixa la nostra aplicació. Existeix un gran nombre d'alternatives que permeten fer un gestió similar, però diferent. [22] [23] [24]

1.2.3 Conclusions

Totes els serveis tenen en comú una pissarra on es mostren les tasques, gestió per a grups d'usuaris, gestió per a grups de treball, control dels temps d'entrega de les tasques i, com a mínim, un client web o aplicació mòbil.

Entre elles es diferencien, però, àmpliament. Per exemple, Jira està enfocada a empreses més grans, és molt més complexa i també té un preu més alt. Todoist enfoca la gestió com un joc, obtenint estadístiques i punts. KanbanFlow permet utilitzar el Mètode Pomodoro de forma personal. Etc. Si comparem els sistemes existents amb el nostre, veiem com ens diferenciem amb la gestió del temps en grup, que és un concepte que no presenten les altres solucions. En canvi, tot i existir solucions molt més completes, hem optat per desenvolupar un sistema més humil inicialment, que abraci menys, però que faci la seva funcionalitat millor.

Capítol 2

Abast del projecte i metodologia

2.1 Formulació del problema

La gestió del temps és sempre un tema difícil. Si ens ho passem bé passa ràpid, si ens avorrim passa lent. A l'hora de treballar, una mala gestió del temps pot fer fracassar tot un projecte, incloient la teva feina i la dels teus companys. És habitual, sinó obligatori, fer una planificació temporal per identificar el temps que es tardarà en fer quelcom. Però, així com sabem fer una planificació, podem fer-la cumplir? És aquest l'objectiu de l'aplicació que us presentaré, aconseguir complir una planificació temporal mitjançant el Mètode Pomodoro.

El principal objectiu del projecte és crear una aplicació perquè la gent utilitzi la Tècnica Pomodoro en els seus grups de treball, de manera que podran beneficiar-se de la tècnica i, a més, treballar al mateix moment, de forma sincronitzada. Això ha de fer, per una part, que els diferents integrants del grup evitin distraccions i es concentrin durant els temps de treball i, per una altra, que tinguin els descansos al mateix temps, podent-los aprofitar per descansar junts, conversar, etc.

2.2 Abast

El sistema resultant d'aquest projecte es durà a terme a partir de la integració de dos components clarament identificats: Primerament, una **aplicació per smartphone** en la que l'usuari podrà utilitzar la Tècnica Pomodoro prèviament explicada de forma çlàssica"segons la Tècnica (individual, amb sessions privades) i també en grup (sessions en grup), que és la innovació que aquest projecte aporta. Per a utilitzar les funcionalitats en grup, l'usuari s'haurà de registrar i entrar al sistema. Si l'usuari és el creador d'un grup en serà l'**administrador**: serà qui afegeixi altres usuaris al grup i qui hi programi sessions per una hora determinada. Si en canvi, l'usuari és un **invitat** en un grup, simplement podrà seguir l'estat de les sessions i els seus pomodoros. També podrà sortir del grup. L'usuari també podrà crear les seves configuracions de temps (com he dit al glossari, triar quin temps de treball, de descans curt i de descans llarg tindrà cada sessió).

Per altra banda, un **servidor** amb una **base de dades** que permetrà la comunicació amb l'aplicació per smartphone, amb el propòsit de permetre la interacció entre usuaris i fer possible la part social (grups) de l'aplicació, aconseguir una persistència de les dades i emmagatzemar informació útil per a futures millores.

2.2.1 Possibles obstacles

Els possibles obstacles i riscos als quals s'haurà de fer front durant el desenvolupament del projecte són:

- **Limitació del temps**: La durada del Treball Final de Grau és limitada i és possible que durant el desenvolupament del sistema aparegui algun imprevist tècnic, conceptual o de qualsevol altre tipus que faci endarrerir de manera inesperada el projecte. Una bona base conceptual i un bon disseny en l'apartat tècnic ajudarà a evitar el risc que el temps suposa. Si s'arribés a aquest punt, s'hauria de repensar la feina programada per tal d'ajustar-s'hi i complir amb els temps previstos.
- **Desconeixement de la tecnologia**: Relacionat amb la limitació del temps tenim el possible obstacle que pot ser tractar amb una tecnologia molt nova o que no es domina totalment. És evident que es necessita un temps d'aprenentatge i una bona comunitat en qualsevol de les tecnologies utilitzades perquè el sistema tingui èxit.
- **Limitacions d'internet**: Per aconseguir una experiència d'usuari òptima és necessari disposar d'internet al smartphone. El risc d'aquest punt és baix, ja que la gran majoria d'usuaris de smartphones disposen de connexió (3g o superior, punt Wi-fi...) i s'acostuma a treballar en llocs amb cobertura suficient.
- **Desacord amb els socis participants**: El disseny gràfic de l'aplicació mòbil serà realitzat per una persona externa al TFG, un soci del projecte. Existeix per tant un risc si hi ha un desacord amb aquesta persona, que podria endarrerir part del projecte.

2.3 Metodologia i rigor

2.3.1 Mètode de treball

Com a metodologia del projecte s'ha escollit **SCRUM-Agile**, amb iteracions freqüents que tindran tasques definides. Els motius són la incertesa que representa desenvolupar un sistema de forma pròpia (no hi ha un client que sap exactament el que vol al darrera) i l'oportunitat d'adaptació que això mateix ens brinda. Gràcies al sistema agile es podrà controlar en petits espais de temps el desenvolupament del projecte i el correcte camí d'acord amb l'entorn (amb especial interès en el feedback rebut pels potencials clients) i les sensacions personals, que canvien constantment. A més, els riscos relacionats amb el temps i la tecnologia podran ser superats ràpidament. La comunicació amb altres socis del projecte també se'n veurà beneficiada gràcies a la comentada velocitat d'adaptació a possibles canvis.

2.3.2 Eines de seguiment

Per tal del correcte seguiment del projecte s'utilitzarà el sistema de control de versions Git, amb un gestor de repositoris online com és GitHub, que també permetrà que el codi del sistema estigui disponible arreu. També s'utilitzarà Trello com a suport en la gestió de tasques del projecte. Per a la documentació i connexió amb el director del projecte i amb els altres socis implicats (dissenyador), s'utilitzarà Google Drive i e-mail.

2.3.3 Mètodes de validació

El projecte disposarà de diferents mètodes de validació per comprovar que aquest avança correctament: Primerament, després de cada **iteració** es valorarà la completesa de les tasques definides i les seves possibles

alteracions, en què ha repercutit això en el producte (experiència d'usuari -UX- i/o rendiment) i es validarà la iteració de forma conjunta amb els socis. Quan existeixi un canvi important en la usabilitat i UX es mesurarà l'aprovació dels potencials usuaris. A part, hi haurà **reunions** freqüents, cada dues setmanes, amb el tutor del projecte, per tal de buscar el consell d'aquest i l'aprovació de nous canvis. Per acabar, l'assignatura de GEP és molt important al principi del projecte per confirmar que els fonaments d'aquest són correctes. Els diversos **lliuraments** asseguren un seguiment constant.

Capítol 3

Requisits

3.1 Requisits funcionals

Registre d'usuari

El registre ha de permetre a l'usuari crear un compte al sistema. Aquest registre es realitzarà amb un correu electrònic, un nom d'usuari i una contrasenya. La contrasenya es guardarà de forma segura amb encriptació hash.

Login

L'usuari ha de poder entrar al sistema amb les seves credencials: email i contrasenya.

Logout

L'usuari ha de poder sortir del sistema en qualsevol moment.

Crear configuració

L'usuari ha de poder crear configuracions de temps per a les sessions. Si l'usuari no ha fet login, aquesta preferència es guardarà només de forma local en el client. Si l'usuari ha fet login, la configuració es guardarà també online. Quan un usuari faci login, les seves configuracions locals passaran a estar online.

Crear sessió privada

L'usuari ha de poder crear una sessió de pomodoros privada per utilitzar la Tècnica Pomodoro de forma *offline*.

Editar sessió privada

L'usuari ha de poder editar una sessió privada i modificar-ne els seus atributs.

Eliminar sessió privada

L'usuari ha de poder eliminar una sessió privada.

Crear grup de treball

L'usuari, si ha fet login, ha de poder crear un grup de treball.

Editar grup de treball

L'usuari, si ha fet login i és el creador del grup, ha de poder editar-ne la seva informació.

Eliminar grup de treball

L'usuari, si ha fet login i és el creador del grup, ha de poder eliminar el grup de treball.

Afegir usuaris a un grup de treball

L'usuari, si ha fet login i és el creador del grup, ha de poder afegir usuaris al grup de treball, mitjançant una cerca per nom o email.

Visualitzar un grup de treball

L'usuari, si ha fet login i és membre del grup, ha de poder veure'n la seva informació: nom del grup, usuaris d'aquest, sessions i estat del comptador de temps (si alguna sessió està en marxa).

Sortir d'un grup de treball

L'usuari, si ha fet login, ha de poder sortir d'un grup de treball al que ha estat afegit.

Crear sessió de grup

L'usuari, si ha fet login i és el creador del grup, ha de poder crear una sessió de pomodoros *online*.

Eliminar sessió de grup

L'usuari, si ha fet login i és el creador del grup, ha de poder eliminar una sessió de pomodoros *online*.

Enviament de notificacions

El sistema ha d'enviar notificacions a un usuari quan se l'hagi afegit a un grup, quan s'hagi creat una sessió a un grup al qual pertany o quan una sessió d'algun grup estigui apunt de començar.

3.2 Requisits de usabilitat i humanitat

Interfície

La interfície ha de ser simple i intuïtiva per facilitar l'ús de l'aplicació a l'usuari.

Actualització de dades

L'aplicació mostrarà sempre les últimes dades disponibles, actualitzant-se automàticament (serà necessària una connexió a internet per actualitzar les dades *online*).

Informació d'errors

El sistema ha d'estar preparat per gestionar errors i mostrar-los-hi a l'usuari, evitant que s'aturi l'aplicació. S'ha de validar els formularis en el client abans d'enviar-los al servidor i donar el *feedback* corresponent a l'usuari.

3.3 Requisits de rendiment i escalabilitat

Temps de resposta

El sistema ha de reaccionar als canvis dels usuaris en menys de 3 segons si s'està en les condicions adequades (smartphone amb 1GB de RAM que mogui fàcilment l'aplicació i que tingui connexió a internet estable).

Escalabilitat

El sistema ha de suportar tenir 1000 usuaris registrats i 10 de connectats simultàniament, tenint em compte la versió gratuïta del hosting que utilitzo. Si el hosting fos de pagament, els límits d'usuaris podrien ser molt més alts.

Extensibilitat

El sistema ha de permetre que s'afegeixin noves funcionalitats o que les existents es modifiquin en el temps.

3.4 Requisits polítics i/o culturals

Llei Orgànica de Protecció de Dades de Caràcter Personal d'Espanya (LOPD)

El sistema ha d'assegurar que es compleix la LOPD.

Capítol 4

Especificació

4.1 Jerarquia d'actors

Anem a mostrar la jerarquia d'actors, que és important per entendre com s'estructuren els rols d'usuari dins de l'aplicació:

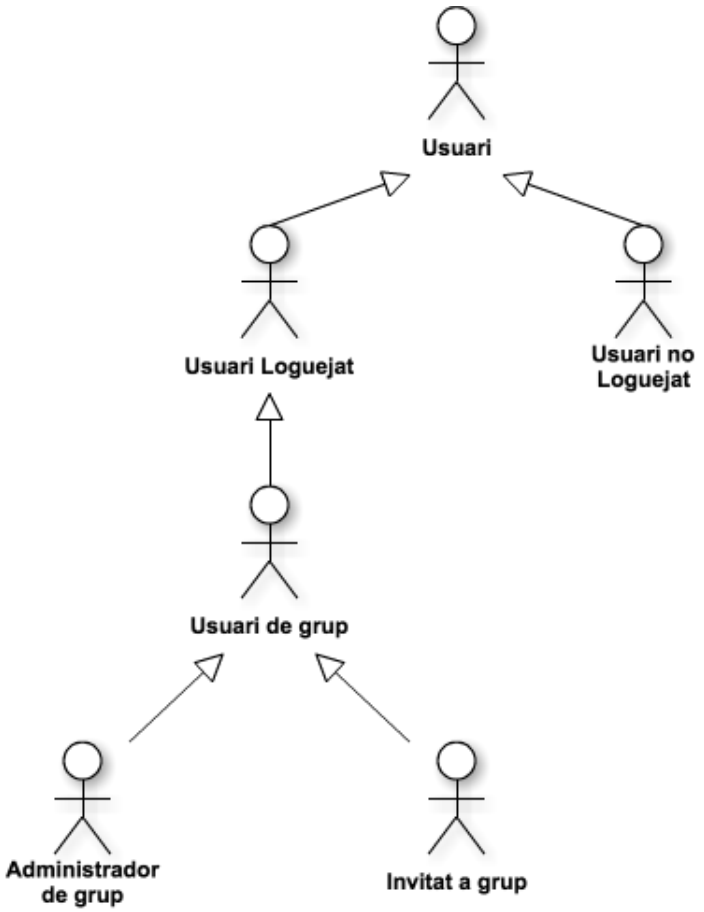


Figura 4.1: Jerarquia d'actors

4.2 Casos d'ús

Un cop mostrada la jerarquia, podem primer visualitzar el diagrama de casos d'ús simplificat:

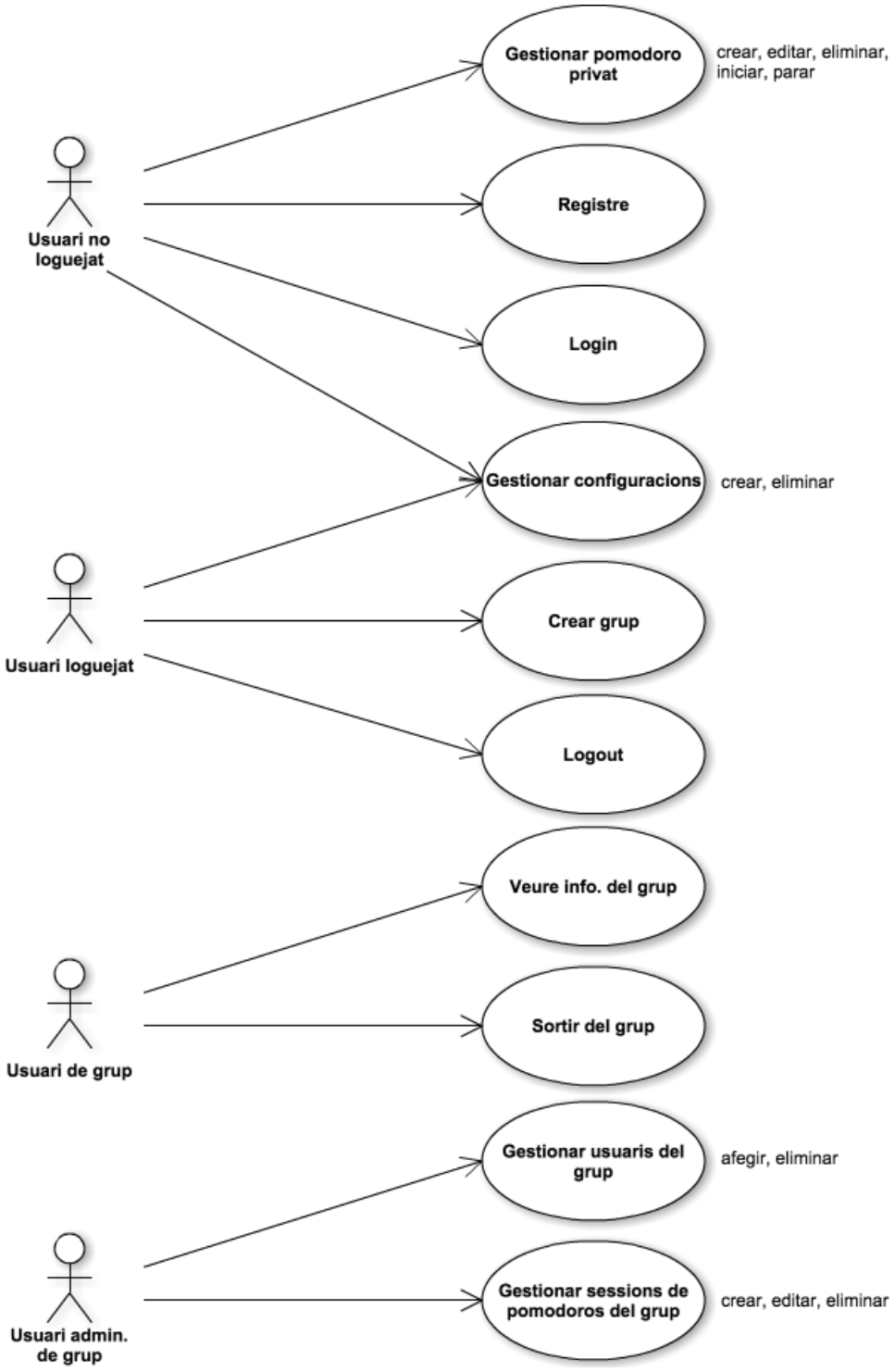


Figura 4.2: Diagrama de casos d'ús

Cas d'ús: Registre d'usuari
Actor: Usuari no loguejat
Precondició: L'usuari no ha fet login al sistema
Disparador: L'usuari vol registrar-se al sistema.
Escenari principal:

1. L'usuari entra a la pantalla de registre.
2. L'usuari completa el formulari amb el seu correu electrònic, nom d'usuari, contrasenya i confirmació de contrasenya.
3. L'usuari prem el botó de confirmació.
4. El sistema fa les corresponents comprovacions i guarda les dades.
5. El sistema porta l'usuari a la pantalla de login i informa a l'usuari de que el registre ha estat correcte.

Cas d'ús: Login
Actor: Usuari no loguejat
Precondició: L'usuari no ha fet login al sistema i hi té un compte.
Disparador: L'usuari vol loguejar-se al sistema.
Escenari principal:

1. L'usuari entra a la pantalla de login.
2. L'usuari completa el formulari amb el seu correu electrònic i contrasenya.
3. L'usuar prem el botó de confirmació.
4. El sistema fa les corresponents comprovacions.
5. El sistema refresca l'aplicació per mostrar la informació de grups i preferències corresponent al seu compte.
6. Si el dispositiu des d'on s'ha fet login pot connectar-se amb GCM (Google Cloud Messaging), el sistema guarda una nova **instal · lació**, que té la informació del dispositiu associada a la informació d'usuari, per poder posteriorment enviar-li notificacions.

Cas d'ús: Logout
Actor: Usuari loguejat
Precondició: L'usuari ha fet login al sistema.
Disparador: L'usuari vol fer logout al sistema.
Escenari principal:

1. L'usuari indica al sistema que vol fer logout.

2. El sistema fa les corresponents comprovacions i confirma el logout.
3. El sistema refresca la informació de l'aplicació perquè es mostri com a no loguejat.

Cas d'ús: Crear configuració
Actor: Usuari
Precondició: -
Disparador: L'usuari vol crear una configuració.
Escenari principal:

1. L'usuari entra a la pantalla de configuracions.
2. L'usuari indica al sistema que vol crear una configuració.
3. L'usuari completa el formulari amb el nom, temps de treball, temps de descans i temps llarg de descans de la configuració.
4. El sistema fa les corresponents comprovacions i guarda les dades. Si l'usuari està loguejat, es guarda la configuració creada al servidor; sinó, només es guarda de forma local.
5. El sistema porta l'usuari a la pantalla de configuracions, on ja es mostra la nova configuració.

Cas d'ús: Crear sessió privada
Actor: Usuari
Precondició: -
Disparador: L'usuari vol crear una sessió de pomodoros privada.
Escenari principal:

1. L'usuari entra a la pantalla de sessions privades.
2. L'usuari indica al sistema que vol crear una sessió privada.
3. El sistema mostra a l'usuari la pantalla de crear sessió privada.
4. L'usuari completa el formulari amb el nom, el número de pomodoros i la configuració de la sessió.
5. El sistema fa les corresponents comprovacions i guarda les dades.
6. El sistema porta l'usuari a la pantalla de sessions privades, on es mostra la nova sessió afegida.

Cas d'ús: Editar sessió privada
Actor: Usuari
Precondició: La sessió privada que es vol editar existeix.
Disparador: L'usuari vol editar una sessió de pomodoros privada.
Escenari principal:

1. L'usuari entra a la pantalla de sessions privades.
2. L'usuari indica al sistema que vol editar una sessió privada.
3. El sistema mostra a l'usuari la pantalla d'editar sessió, on es mostra el nom, el nombre de pomodoros i la configuració de la sessió que es vol modificar.
4. L'usuari completa el formulari amb el nom, el número de pomodoros i la configuració de la sessió.
5. El sistema fa les corresponents comprovacions i guarda les dades.
6. El sistema porta l'usuari a la pantalla de sessions privades, on es mostra la sessió ja editada.

Cas d'ús: Eliminar sessió privada
Actor: Usuari
Precondició: La sessió privada que es vol eliminar existeix.
Disparador: L'usuari vol eliminar una sessió de pomodoros privada.
Escenari principal:

1. L'usuari entra a la pantalla de sessions privades
2. L'usuari indica al sistema que vol eliminar una sessió privada.
3. El sistema pregunta a l'usuari si realment la vol eliminar.
4. L'usuari confirma.
5. El sistema fa les corresponents comprovacions i guarda les dades.
6. El sistema porta l'usuari a la pantalla de sessions privades, on ja no es mostra la sessió privada eliminada.

Cas d'ús: Iniciar sessió privada
Actor: Usuari
Precondició: La sessió privada que es vol iniciar existeix i està la primera a la pila de sessions.
Disparador: L'usuari vol iniciar una sessió de pomodoros privada.
Escenari principal:

1. L'usuari entra a la pantalla de sessions privades.
2. L'usuari indica al sistema que vol iniciar una sessió privada.
3. S'inicia la sessió privada.

Cas d'ús: Parar sessió privada
Actor: Usuari
Precondició: La sessió privada que es vol parar existeix i està en marxa.
Disparador: L'usuari vol parar una sessió de pomodoros privada.
Escenari principal:

1. L'usuari entra a la pantalla de sessions privades.
2. L'usuari indica al sistema que vol parar una sessió privada.
3. S'atura la sessió privada.

Cas d'ús: Crear grup de treball
Actor: Usuari loguejat
Precondició: -
Disparador: L'usuari vol crear un grup de treball.
Escenari principal:

1. L'usuari entra a la pantalla de grups de treball.
2. L'usuari indica al sistema que vol crear un grup de treball.
3. El sistema mostra a l'usuari la pantalla de creació d'un grup de treball
4. L'usuari completa el formulari amb el nom, la descripció i els usuaris a afegir al grup.
5. L'usuari prem el botó de confirmar.
6. El sistema fa les corresponents comprovacions i guarda les dades.
7. El sistema porta l'usuari a la pantalla de grups, on es mostra el nou grup.

Cas d'ús: Editar grup de treball
Actor: Administrador de grup
Precondició: El grup que vol editar existeix.
Disparador: L'usuari vol editar un grup de treball.
Escenari principal:

1. L'usuari entra a la pantalla de grups de treball.
2. L'usuari indica al sistema que vol editar un grup de treball.
3. El sistema mostra a l'usuari la pantalla d'edició d'un grup de treball, amb la informació pertinent del grup.
4. L'usuari completa el formulari amb el nom, la descripció i els usuaris a afegir al grup (d'acord amb el cas d'ús *afegir usuari a un grup de treball*.
5. L'usuari prem el botó de confirmar.
6. El sistema fa les corresponents comprovacions i guarda les dades.
7. El sistema porta l'usuari a la pantalla de grups, on es mostra el grup amb les noves dades.

Cas d'ús: Eliminar grup de treball
Actor: Administrador de grup
Precondició: El grup que vol editar existeix
Disparador: L'usuari vol eliminar un grup de treball.
Escenari principal:

1. L'usuari entra a la pantalla de grups de treball.
2. L'usuari indica al sistema que vol eliminar un grup de treball.
3. El sistema pregunta a l'usuari si realment vol eliminar el grup.
4. L'usuari prem el botó de confirmar.
5. El sistema fa les corresponents comprovacions i guarda les dades.
6. El sistema porta l'usuari a la pantalla de grups, on ja no es mostra el grup.

Cas d'ús: Afegir usuari a un grup de treball
Actor: Administrador de grup
Precondició: El grup on vol afegir usuaris existeix.
Disparador: L'usuari vol afegir usuaris a un grup de treball.
Escenari principal:

1. L'usuari indica al sistema que vol afegir usuaris al grup.
2. L'usuari escriu el nom d'usuari o email del nou usuari.
3. El sistema mostra dinàmicament els usuaris que encaixen amb la búsqueda.
4. L'usuari prem en un dels usuaris mostrats.
5. L'usuari prem el botó de confirmar.
6. El sistema afegeix el nou usuari al grup.

Cas d'ús: Visualitzar un grup de treball
Actor: Usuari de grup
Precondició: El grup que vol visualitzar existeix.
Disparador: L'usuari vol visualitzar un grup de treball.
Escenari principal:

1. L'usuari entra a la pantalla de grups de treball.
2. L'usuari indica al sistema que vol visualitzar un grup de treball.
3. El sistema mostra a l'usuari la informació de grup de treball i les seves sessions de grup.

Cas d'ús: Sortir d'un grup de treball
Actor: Invitat a grup
Precondició: El grup del que es vol sortir existeix.
Disparador: L'usuari vol sortir d'un grup de treball.
Escenari principal:

1. L'usuari entra a la pantalla d'un grup de treball.
2. L'usuari indica al sistema que vol sortir del grup de treball.
3. El sistema preguna a l'usuari si realment vol sortir del grup.
4. L'usuari prem el botó de confirmar.
5. El sistema indica a l'usuari que s'ha sortit del grup i mostra la pantalla de grups sense el grup d'on s'ha sortit.

Cas d'ús: Crear sessió de grup
Actor: Administrador de grup
Precondició: El grup en el que es vol afegir una sessió existeix.
Disparador: L'usuari vol crear una sessió de grup.
Escenari principal:

1. L'usuari entra a la pantalla de grup.
2. L'usuari indica al sistema que vol crear una sessió.
3. El sistema mostra a l'usuari la pantalla de crear sessió.
4. L'usuari completa el formulari amb el nom, el número de pomodoros, la data d'inici i la configuració de la sessió.
5. L'usuari prem el botó de confirmar.
6. El sistema fa les corresponents comprovacions, guarda les dades i envia una notificació a cada membre del grup.
7. El sistema porta l'usuari a la pantalla de grup, on es mostra la nova sessió afegida.

Cas d'ús: Eliminar sessió de grup
Actor: Administrador de grup
Precondició: El grup en el que es vol eliminar una sessió existeix.
Disparador: L'usuari vol eliminar una sessió de grup.
Escenari principal:

1. L'usuari entra a la pantalla de grup.

2. L'usuari indica al sistema que vol eliminar una sessió.
3. El sistema pregunta a l'usuari si realment vol eliminar la sessió.
4. L'usuari prem el botó de confirmar.
5. El sistema fa les corresponents comprovacions i guarda les dades.
6. El sistema porta l'usuari a la pantalla de grup, on ja no es mostra la sessió.

Cas d'ús: Enviar notificació d'accés a un grup
Actor: Sistema
Precondició: El grup, l'administrador del grup i l'usuari afegit existeixen.
Disparador: Un administrador ha afegit un usuari a un grup.
Escenari principal:

1. L'administrador afegeix un usuari a un grup.
2. El sistema rep que s'ha afegit i li envia una notificació.
3. La notificació apareix al mòbil de l'usuari.

Cas d'ús: Enviar notificació de nova sessió
Actor: Sistema
Precondició: El grup, l'administrador del grup i els seus usuaris existeixen.
Disparador: Un administrador ha afegit una sessió de grup a un grup.
Escenari principal:

1. L'administrador afegeix una sessió de grup.
2. El sistema rep que s'ha afegit i envia una notificació a tots els usuaris d'aquest grup.
3. La notificació apareix al mòbil dels usuaris.

Cas d'ús: Enviar notificació d'inici de sessió
Actor: Sistema
Precondició: La sessió de grup existeix.
Disparador: Falta un minut perquè comenci una sessió de grup.
Escenari principal:

1. Falta un minut perquè comenci una sessió de grup.
2. La notificació apareix al mòbil de l'usuari.

4.3 Model conceptual

A continuació es mostrarà el model conceptual de tot el sistema. S'hi inclouen els models necessaris per poder proporcionar notificacions a l'usuari.

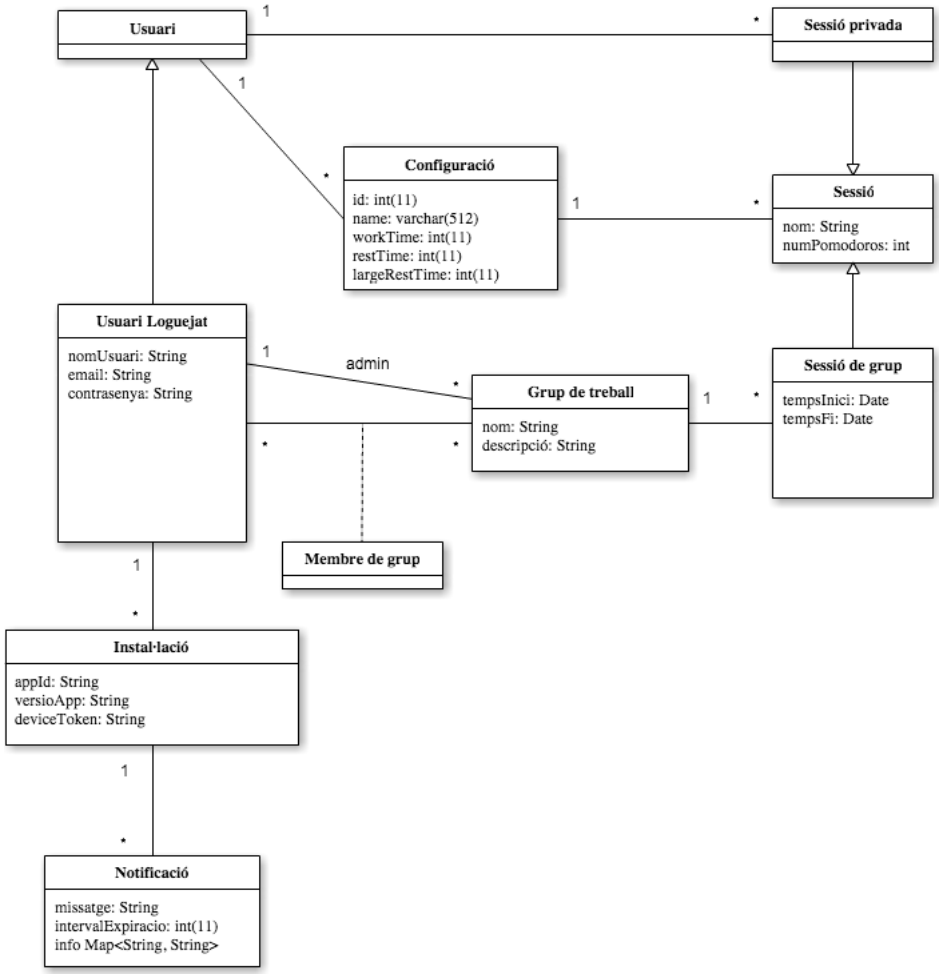


Figura 4.3: Model conceptual

Capítol 5

Disseny

5.1 Arquitectura general

Inicialment, per fer-nos una idea de l'estructura del projecte, ens ajudarà aquesta figura on es mostra com està organitzada i com es transmet la informació entre els diferents elements del sistema.

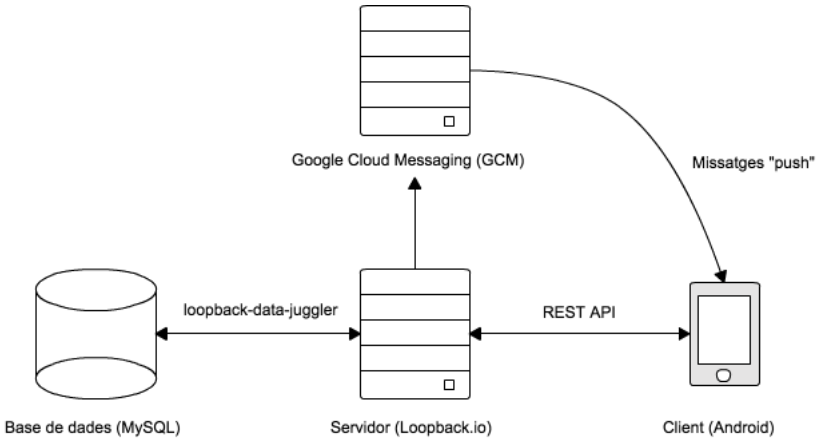


Figura 5.1: Arquitectura general

- **Base de dades**: La base de dades és on es guarda tota la informació registrada al sistema.
- **Servidor**: El servidor s'encarrega de rebre les peticions del client, processar-les i fer les consultes i modificacions corresponents a la base de dades.
- **Client**: El client és l'aplicació que arriba a l'usuari final i amb qui aquest interactua. Quan l'usuari realitza accions, el client es preocupa d'actualitzar la seva vista i de realitzar les peticions necessàries al servidor, si n'hi ha.
- **Google Cloud Messaging**: També és un servidor, en aquest cas de la pròpia Google. Quan el servidor li ho demana, envia els missatges necessaris al client.

5.2 Disseny del servidor i la base de dades

El servidor està construït sobre el framework Loopback.io, un framework de Javascript del costat del servidor. Aquest servidor consulta i modifica les dades de la base de dades, que és MySQL. Finalment, el servidor s'exposa al client mitjançant una REST API. Anem a explicar-ho en més detall.

5.2.1 Disseny intern: Loopback.io

Primerament cal parlar de Javascript i Node.js. JavaScript és un llenguatge de programació interpretat, dialecte de l'estàndard ECMAScript. Es defineix com orientat a objectes, basat en prototips, imperatiu, de tipat dèbil i dinàmic. Node.js és un entorn en temps d'execució multi-plataforma, de codi obert, per la capa del servidor. Està basat en ECMAScript (Javascript), asíncron, amb una arquitectura orientada a events i basada en el motor V(de Google. Va ser creat amb la intenció de ser útil en la creació de programes de xarxa molt escalables, com ara servidors web.

A partir de Node.js i també de Express.js, un altre framework basat en Node.js es crea Loopback.io, un framework per connectar aplicacions amb informació a través d'APIs. Adopta el paradigma de disseny de software *convention over configuration* (convenció sobre configuració) per tal de facilitar i agilitzar el desenvolupament de software. La creació d'una REST API bàsica amb els mètodes *CRUD* es pot realitzar per terminal sense haver de programar en 5 minuts, per posar un exemple del pragmàtic que és.

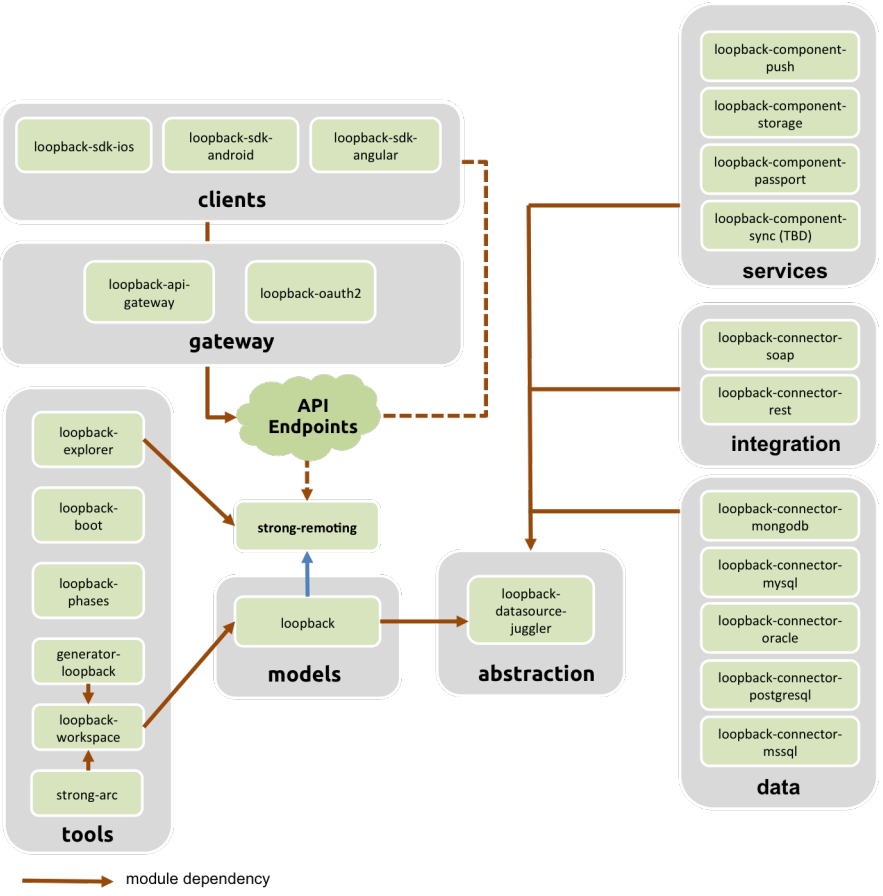


Figura 5.2: Arquitectura general

En la Figura 5.2 veiem els diferents mòduls de Loopback.io i com estan connectats:

- **API Endpoints**: Són les urls finals de la REST API.
- **models**: Els models representen dades que hi ha al back-end, com ara la informació de les bases de dades.
- **abstraction**: Com ja hem comentat abans, *loopback-datasource-juggler* és la interfície per connectar-se a altres fonts d'informació. En el nostre cas, utilitzem el component *loopback-connector-mysql*, perquè la nostra base de dades és MySQL. També utilitzem *loopback-component-push* per enviar notificacions al client.
- **tools**: Són diferents eines que proporciona loopback. Per exemple, he utilitzat *loopback-explorer*, que ve activat per defecte, i què genera una web on visualitzar i provar els *endpoints* de l'API. També he utilitzat *strong-arc* que proporciona una interfície per terminal per poder crear ràpidament un projecte, afegir-li models, relacions entre models, permisos, etc.
- **clients**: Loopback.io també té SDK's per als clients més habituals: iOS, Android i Angular.js (web). He utilitzat el d'Android perquè facilitava la gestió de notificacions.

El més important i representatiu són els models, que defineixen la base de dades i l'estructura autogenerada de la REST API.

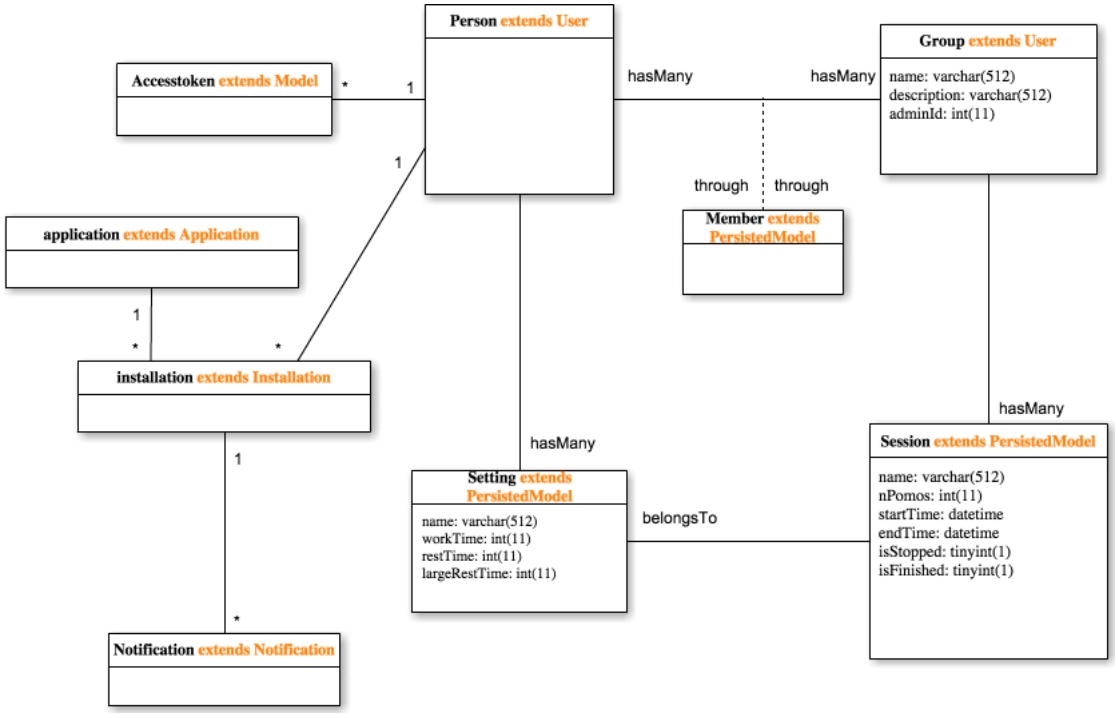


Figura 5.3: Models de Loopback del sistema

Veiem com és similar al model conceptual. Quan un model té un extends, significa que té un model base ja creat pel propi Framework. Molt útils ja que eviten feina redundant. Només es mostren els atributs explícitament creats per aquest model. Les relacions entre els altres models s'expressen en *hasMany* (té molts), *belongsTo* (pertany a) i *through* (a través de). La resta de relacions es creen automàticament pel framework. Ho he expressat en cardinalitat per facilitar-ne la lectura.

Autenticació i *Access Tokens*

El sistema d'autenticació del servidor és simple i està basat en *Tokens* o tiquets, que simbolitzen una acreditació temporal de que es té el consentiment de l'usuari per les peticions que els ensenyin. Per exemple:

1. Un usuari fa login des del client Android, introduïnt email i password.
2. Si les dades són correctes, el sistema genera un tiquet (un String) que té també un ttl (*Time To Live*, o temps de vida) i està associat a aquest usuari.
3. El client Android es guardarà aquest tiquet i en les accions en que es necessiti, es passarà a la petició.
4. Llavors, el servidor comprovarà si el tiquet es vàlid i a qui pertany, i si realment pot fer l'acció que es vol fer comprovant els permisos necessaris (si, per exemple, el tiquet sigués vàlid però es volgués eliminar un grup en que l'usuari no sigués administrador, el servidor retornaria un 401 Authorization Required).

Tal com es veu hi ha un model que es diu Accesstoken. És on es guarden els *tiquets*.
És important que els endpoints de l'API siguin HTTPS, per tal de millorar la seguretat del sistema.

5.2.2 Disseny de la base de dades

MySQL és un sistema de gestió de bases de dades relacional, multi-fil i multi-usuari. La base de dades MySQL es genera a partir dels models de Loopback definits que hem comentat prèviament gràcies a *LoopBack DataSource Juggler*, un ORM integrat al framework Loopback.io.

5.2.3 Disseny extern: REST API

La REST API és el que "obre"el sistema a l·l'exterior". El client es comunicarà amb el servidor a partir d'aquesta.

Automàticament, Loopback.io crea moltes crides possibles a la API, però en general m'he basat en les operacions CRUD (*Create, Read, Update, Delete*). Per exemple, l'estructura resultant d'una part de la API és la següent:

Taula 5.1: Exemple de disseny de l'API

Model	Verb	Endpoint	Descripció
Group	POST	/Groups	Crea un grup.
	<GET, PUT, DELETE>	/Groups/{id}	<Obté, modifica, elimina >el grup amb id.
	GET	/Groups/{id}/people	Obté les persones del grup amb id.
	GET	/Groups/{id}/sessions	Obté les sessions del grup amb id.
Person	POST	/People	Crea una persona (registre).
	<GET, PUT, DELETE>	/People/{id}	<Obté, modifica, elimina >el grup amb id.
	GET	/People/{id}/groups	Obté els grups de la persona amb id.
	GET	/People/{id}/settings	Obté les configuracions de la persona amb id.
	POST	/People/login	Login. Se li passen les credencials i retorna un token.

Es veu com, de manera general, cada model té les seves operacions CRUD i GETs per obtenir els seus models relacionats.

A `https://pomoapp.herokuapp.com/explorer` es pot utilitzar la documentació dinàmica autogenerada per Loopback.io. Permet visualitzar endpoints, atributs i interactuar-hi.

5.3 Google Cloud Messaging (GCM)

Google Cloud Messaging permet enviar informació des del servidor als dispositius dels usuaris i rebre missatges dels dispositius en la mateixa connexió. El servei de GCM controla tots els aspectes de les cues de missatges i l'entrega al client. És totalment gratuït. A la Figura 5.4 es mostra el funcionament general.

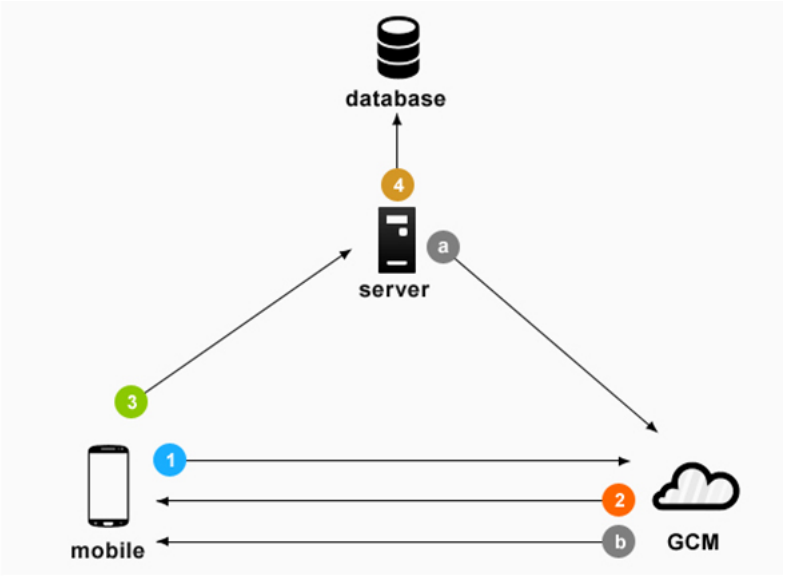


Figura 5.4: Arquitectura general

1. El dispositiu Android envia el *senderId* i l'*applicationId* al servidor GCM per poder registrar-se.
2. El servidor torna un *registrationId*.
3. Després de rebre el *registrationId* el dispositiu l'envia al servidor.
4. El servidor guarda el *registrationId* a la base de dades per un ús posterior.
   (a) Quan s'ha d'enviar una notificació push, el servidor envia un missatge al servidor GCM junt amb el *registrationId* del dispositiu.
   (b) El servidor GCM entrega el missatge al dispositiu corresponent a aquest *registrationId*.

5.4 Disseny del client

El client ha estat creat en Android. Android és un sistema operatiu basat en el nucli de Linux. Va ser dissenyat principalment per dispositius mòbils amb pantalla tàctil (telèfons o tablets), però també per a rellotges intel · ligents, televisions i automòbils. Va ser desarrollat inicialment per Android Inc., empresa que va comprar Google.

5.4.1 Disseny intern: Estructura de carpetes i funcionalitats

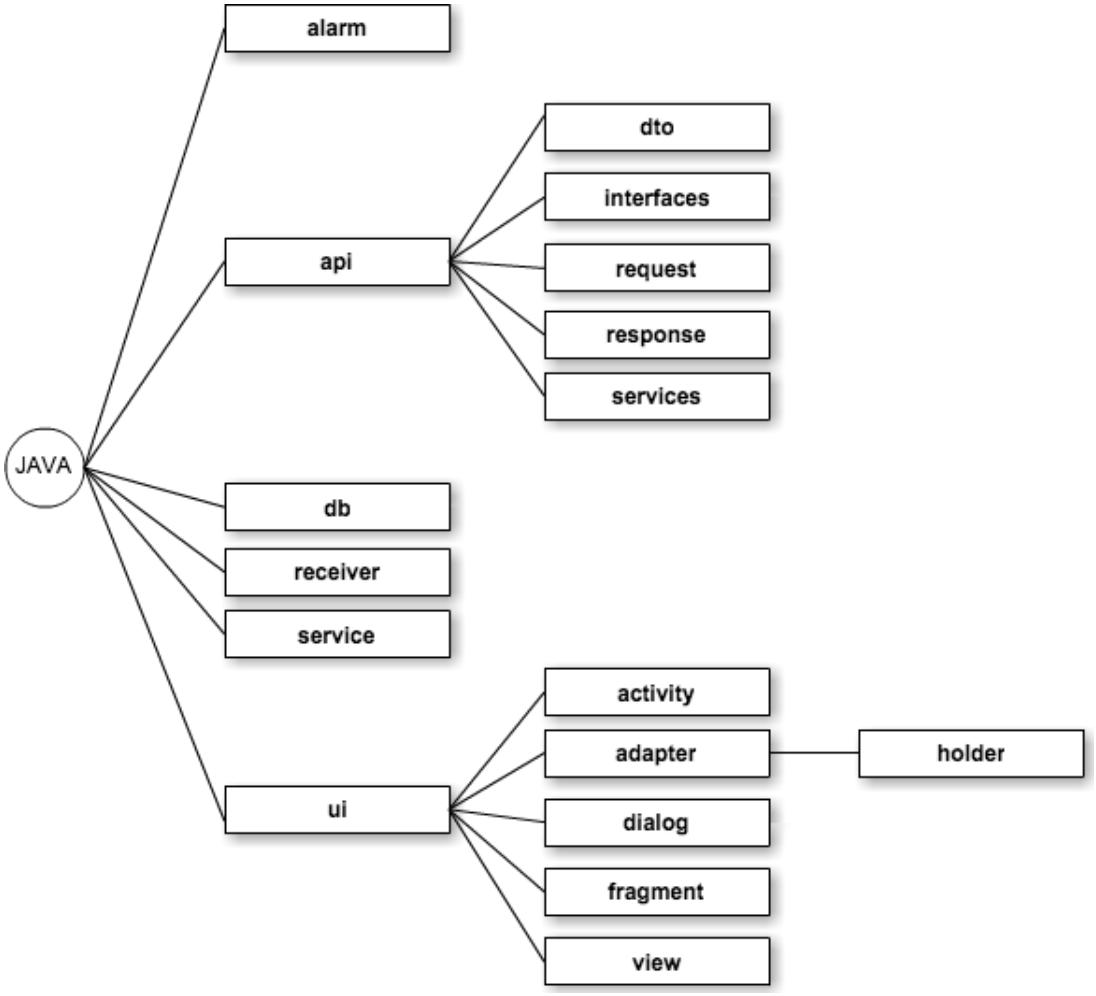


Figura 5.5: Estructura de carpetes Java

En el codi java:

- **alarm**: Només hi ha AlarmUtils.java, utilitzat per programar alarmes al sistema.
- **api**: Totes les classes relacionades amb la interacció amb la API.
  - **dto**: Data Transfer Objects. Classes Java equivalents a la resposta JSON (i, per tant, als models de Loopback).
  - **interfaces**: Les diferents interfaces per connectar amb els diferents endpoints de la API, gràcies a la biblioteca Retrofit, com explicaré més endavant.
  - **request**: Data Transfer Objects específics per fer requests (peticions).
  - **response**: Data Transfer Objects específics per rebre responses (respostes).
  - **services**: Els diferents serveis per a realitzar les operacions de la API.
- **db**: Classes que representen els diferents models de la base de dades local Realm. També hi ha DBHandler.java, per realitzar les peticions.

- **receiver**: Els diferents BroadcastReceiver, per rebre events com ara alarmes, notificacions o l'arrancada del dispositiu.
- **service**: Només hi ha un servei, GcmIntentService.java, que s'encarrega de manejar la rebuda de notificacions per part de GCM.
- **ui**: Totes les classes relacionades amb la part visual.
  - **activity**: Les diferents acitivities del sistema (vista completa).
  - **adapter**: Els diferents adapters encarregats de mostrar llistats de dades. Per exemple el llistat de grups a la pantalla de grup.
    - **holder**: Només hi ha GroupViewHolder.java. S'ha creat per utilitzar el patró ViewHolder amb GroupAdapter, que és un RecyclerView.
  - **dialog**: Els diferents popups que surten a l'aplicació.
  - **fragment**: Els fragments (vistes parcials) que hi ha al sistema.
  - **view**: Vistes personalitzades creades per a la app.

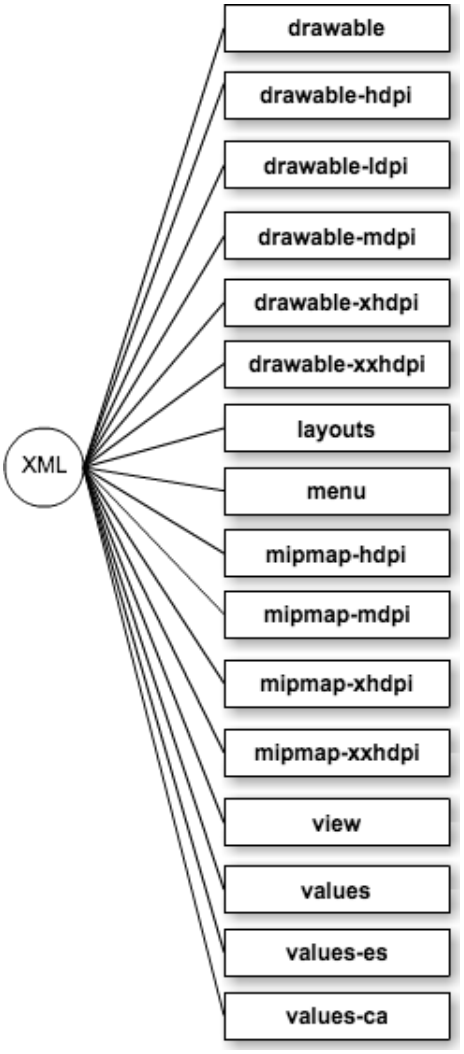


Figura 5.6: Estructura de carpetes XML

Quant a resources, s'hi troba:

- **drawables**: Carpetes on s'hi guarda imatges, formes, colors de fons, etc. per diferents densitats de pantalla.
- **layout**: XMLs que defineixen les vistes de la app.
- **mipmaps**: Carpetes on s'hi guarda la icona de la app per diferents densitats de pantalla.
- **values**: Carpetes on s'hi gaurden valors com ara texts (strings), dimensions, colors, estils de text o de vistes, etc. Depenent del sufix, s'hi poden guardar continguts per diferents idiomes.

5.4.2 Disseny extern: Mapa navegacional

Com a disseny extern del client Android, s'ha realitzat un mapa navegacional amb mockups, que ens ha permès al dissenyador i a mi pensar en la forma que havia de tenir l'aplicació. També ens ha servit com a pont per intercanviar idees i portar en la mateixa direcció el disseny gràfic i la implementació.

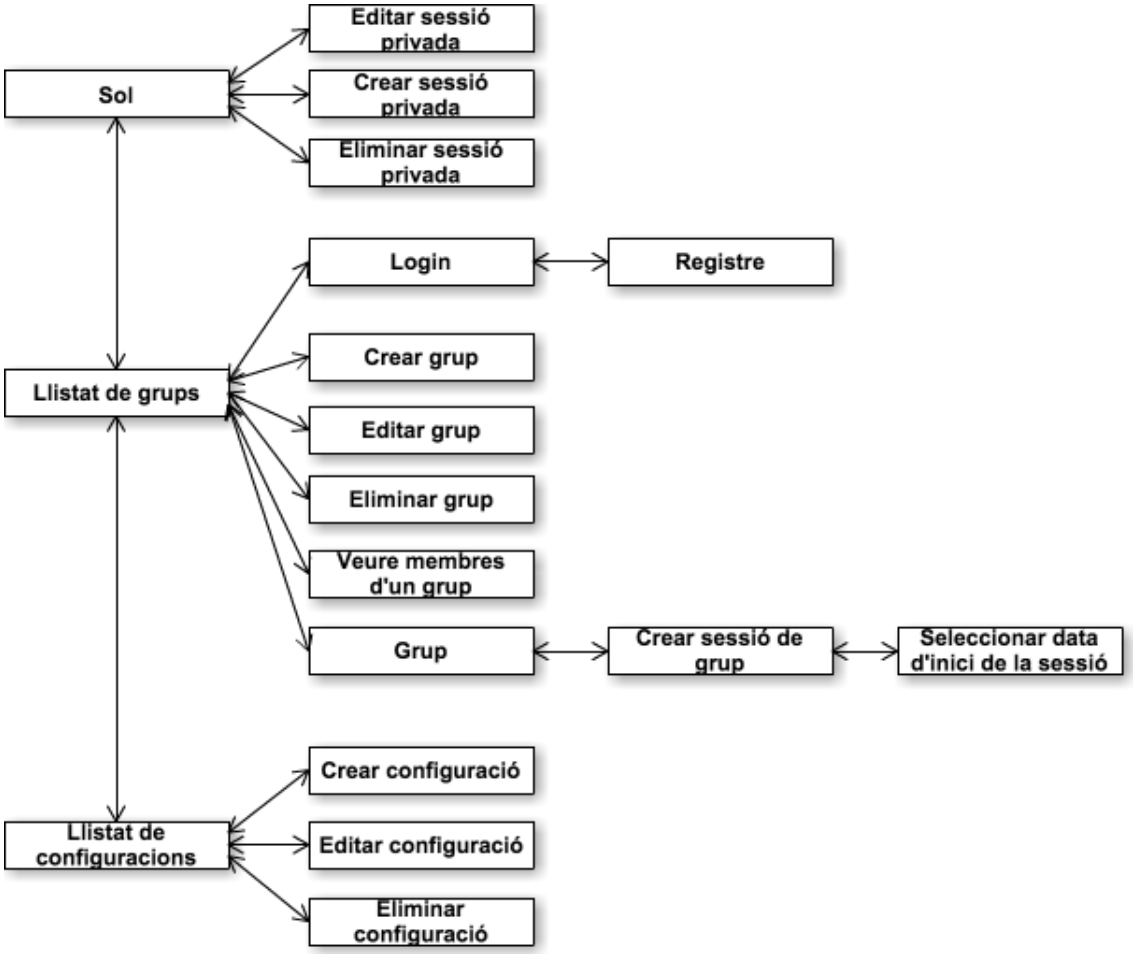
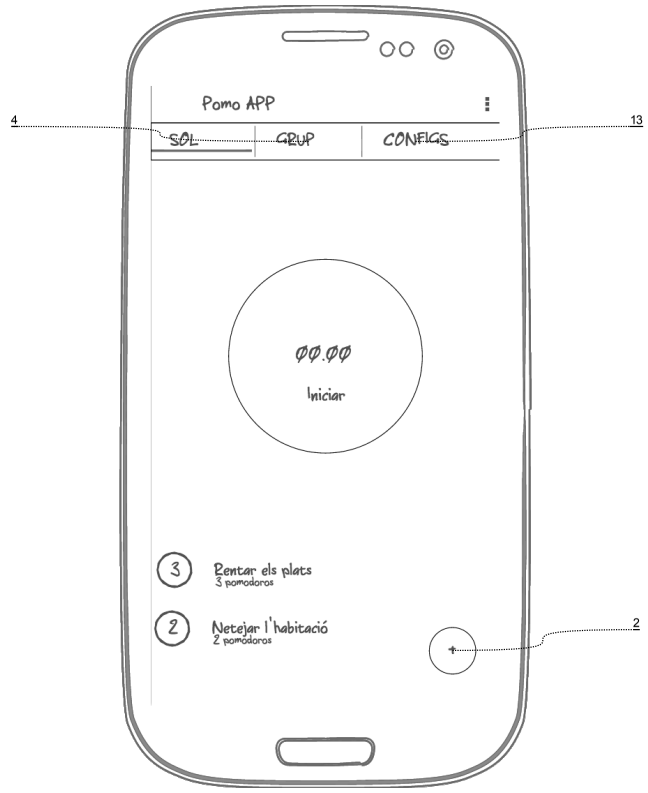
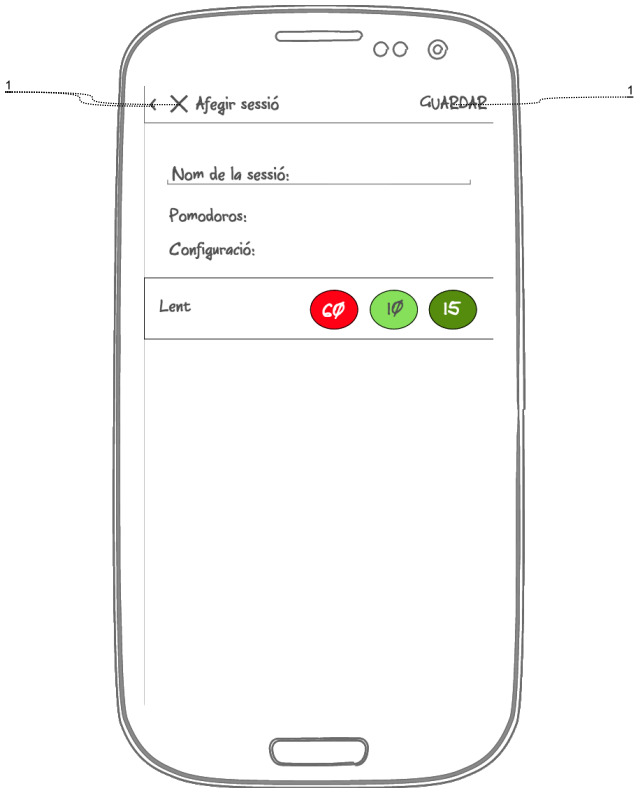


Figura 5.7: Mapa navegacional

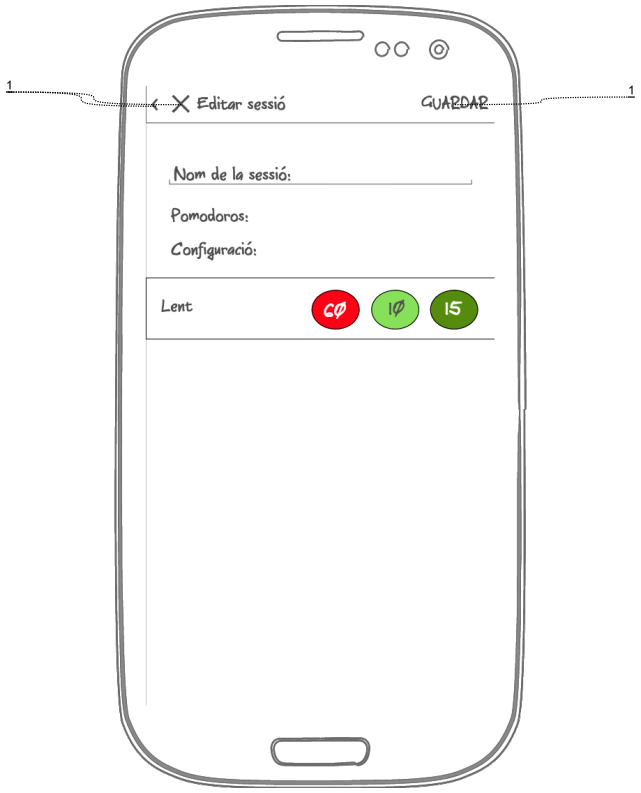
1 - alone



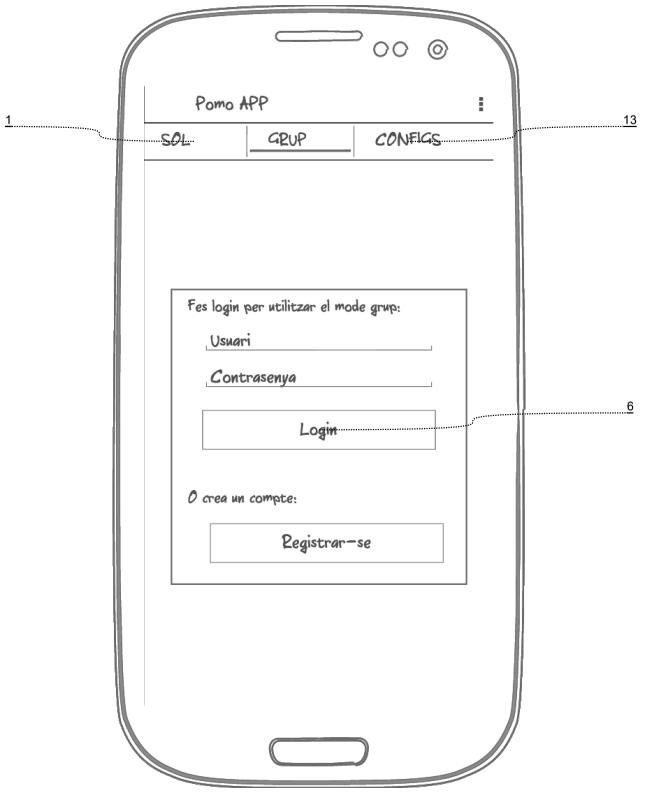
2 - addAloneSession

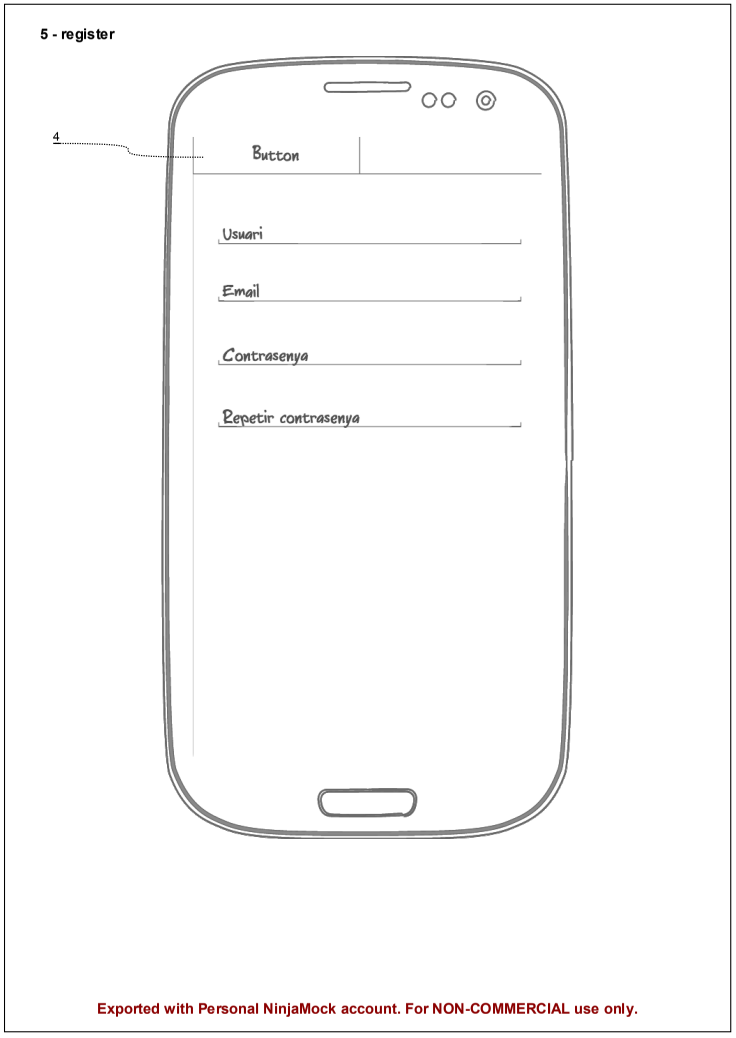


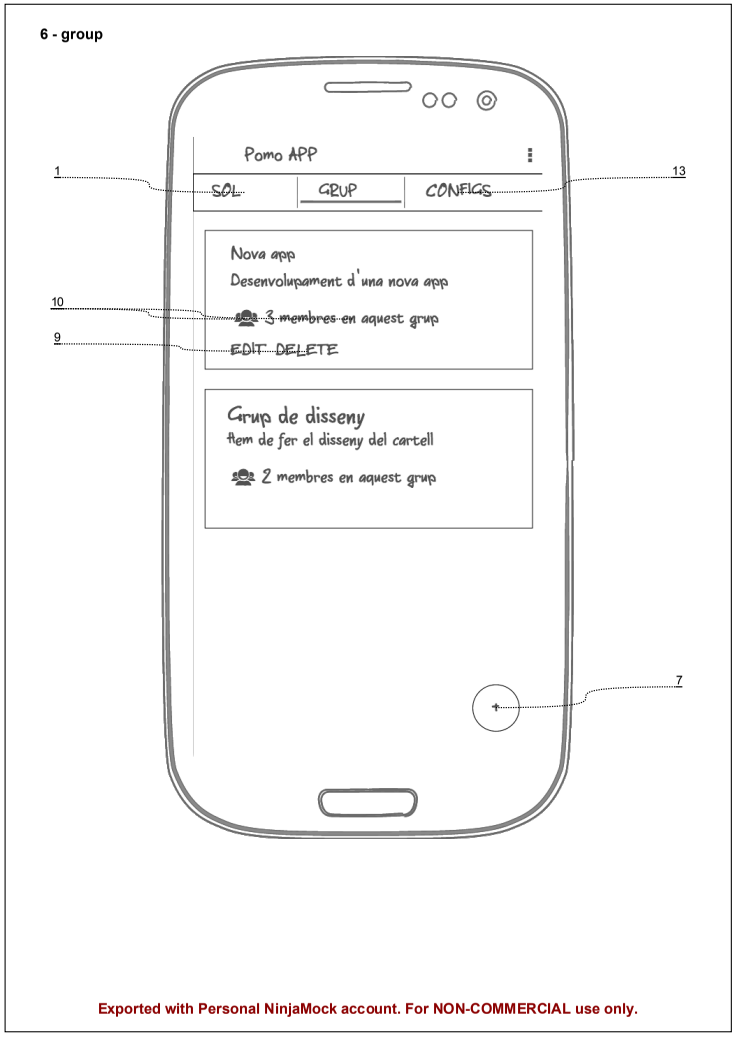
3 - editAloneSession



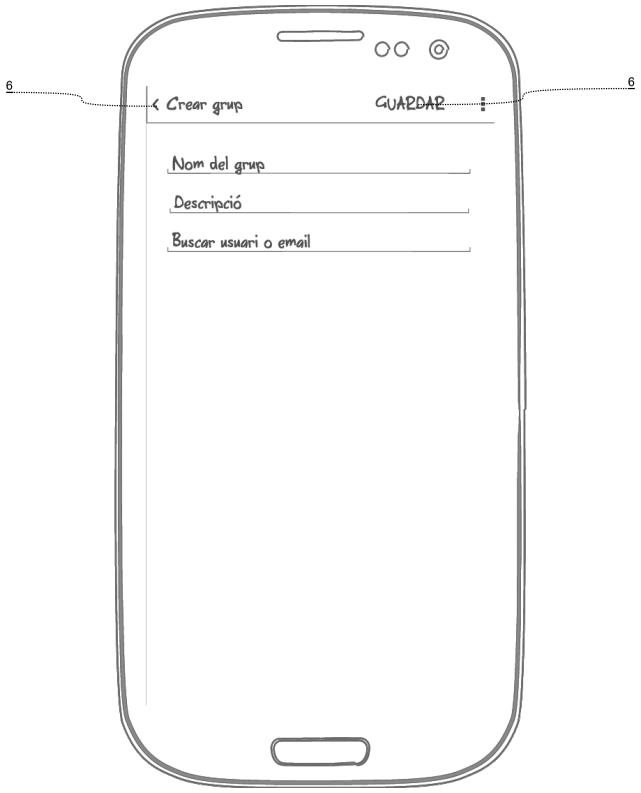
4 - groupLoginDialog



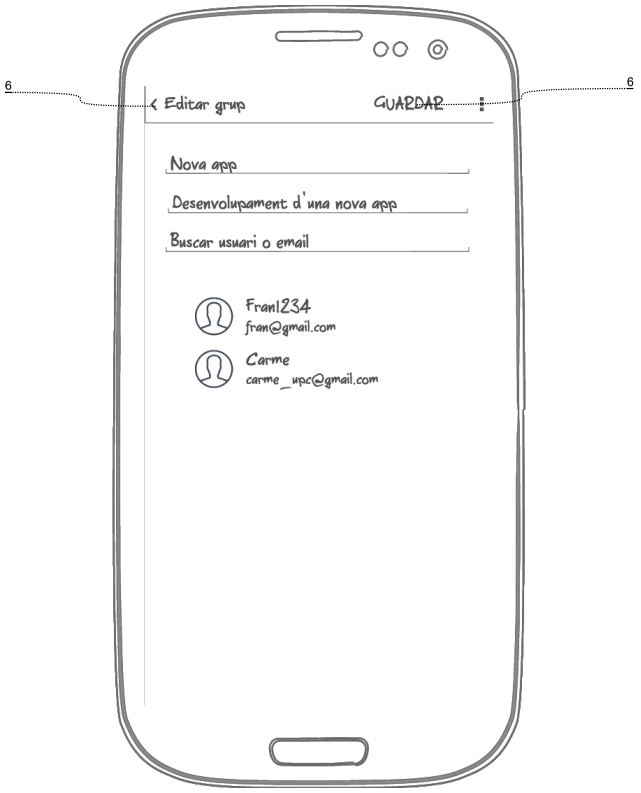

5 - register
4
Button
Usuari
Email
Contrasenya
Repetir contrasenya
Exported with Personal NinjaMock account. For NON-COMMERCIAL use only.


6 - group
Pomo APP
1
SOL
GRUP
CONFIGS
13
Nova app
Desenvolupament d'una nova app
10
3 membres en aquest grup
9
EDIT DELETE
Grup de disseny
Item de fer el disseny del cartell
2 membres en aquest grup
7
Exported with Personal NinjaMock account. For NON-COMMERCIAL use only.

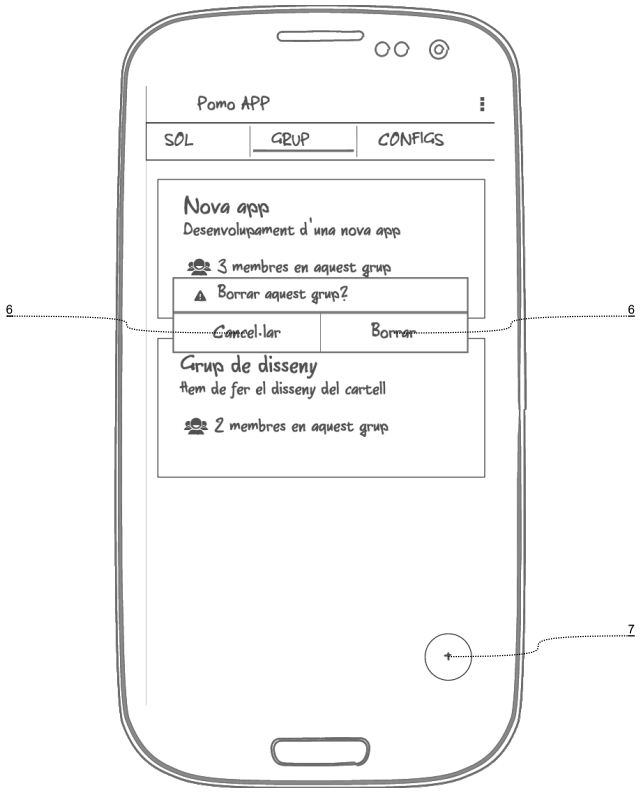
7 - addGroup



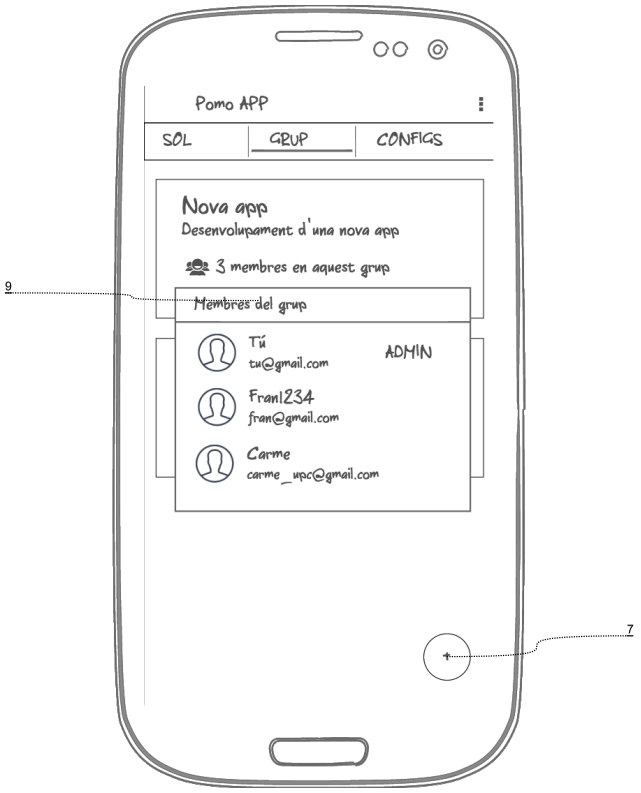
8 - editGroup

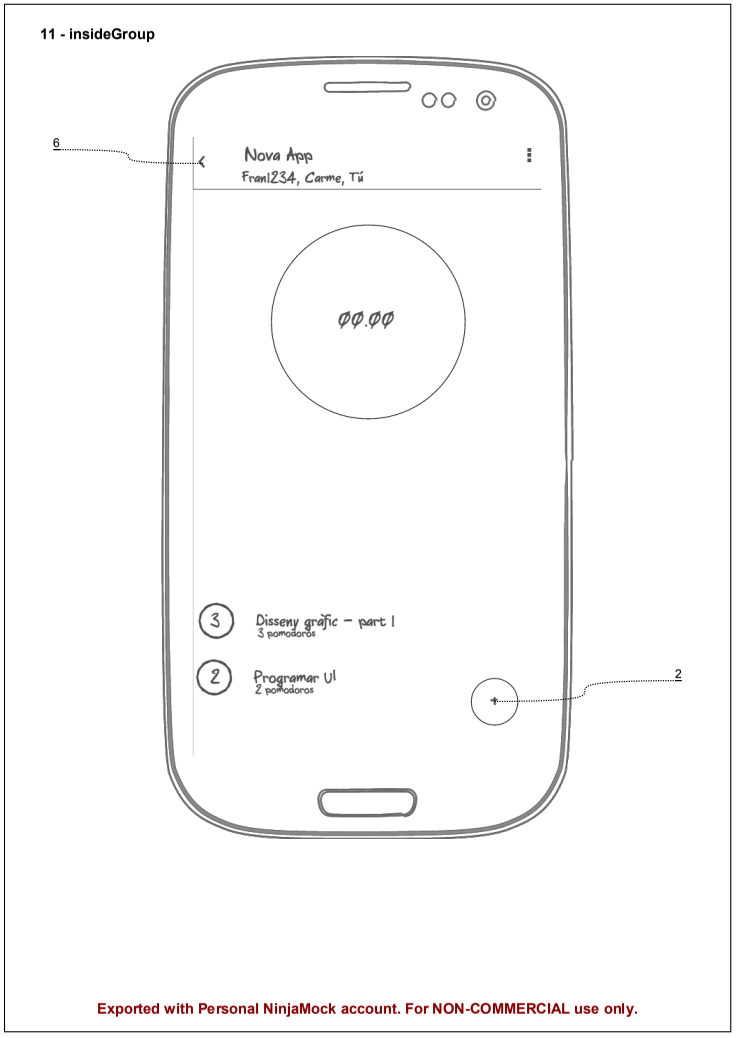


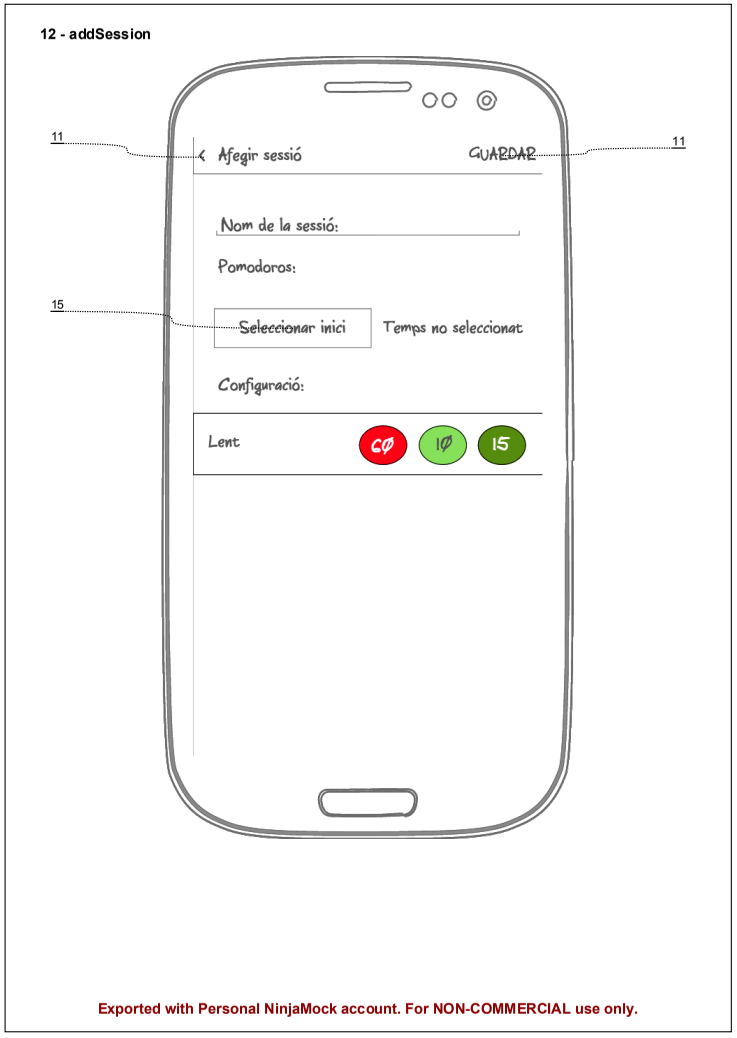
9 - deleteGroup



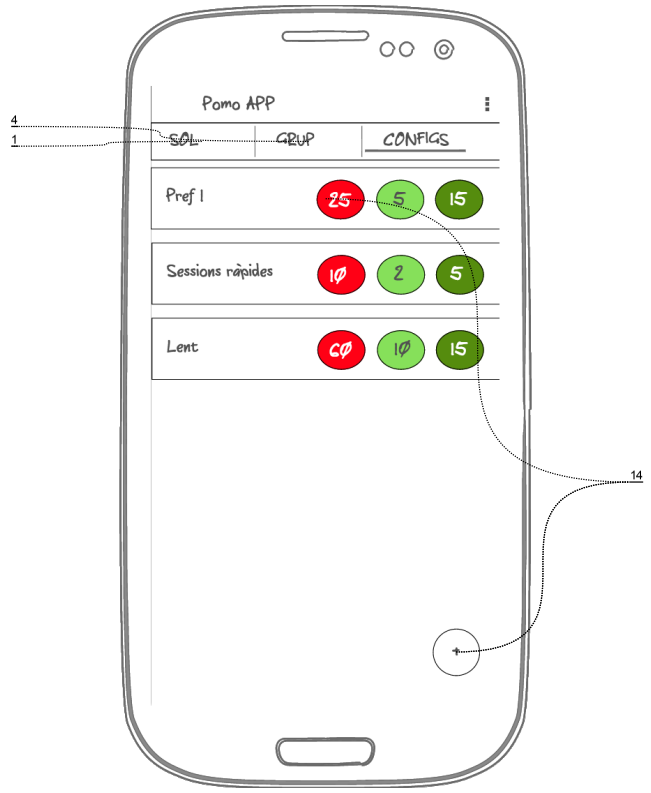
10 - groupMembers



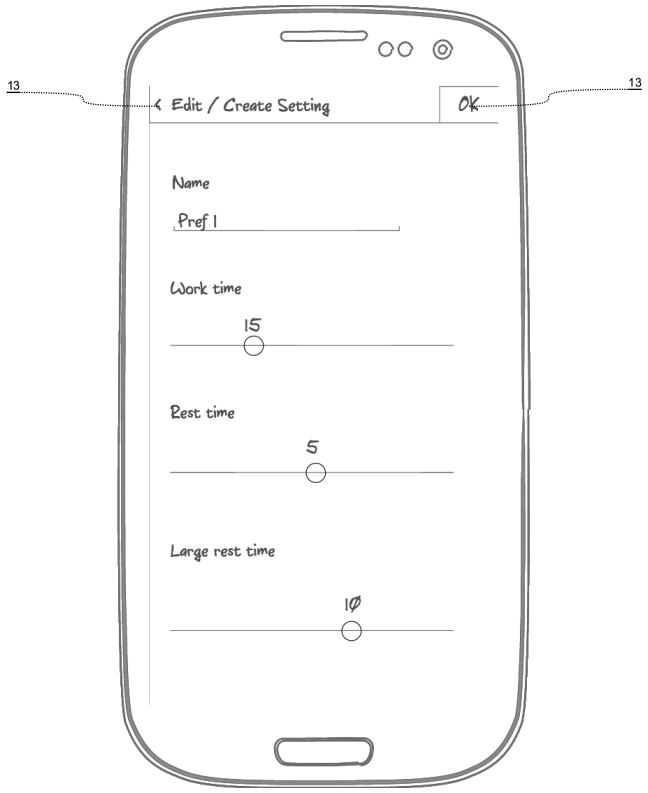

11 - insideGroup
6
Nova App
Fran1234, Carme, Tú
ØØ.ØØ
3
Disseny gràfic - part I
3 pomodoros
2
Programar UI
2 pomodoros
2
Exported with Personal NinjaMock account. For NON-COMMERCIAL use only.

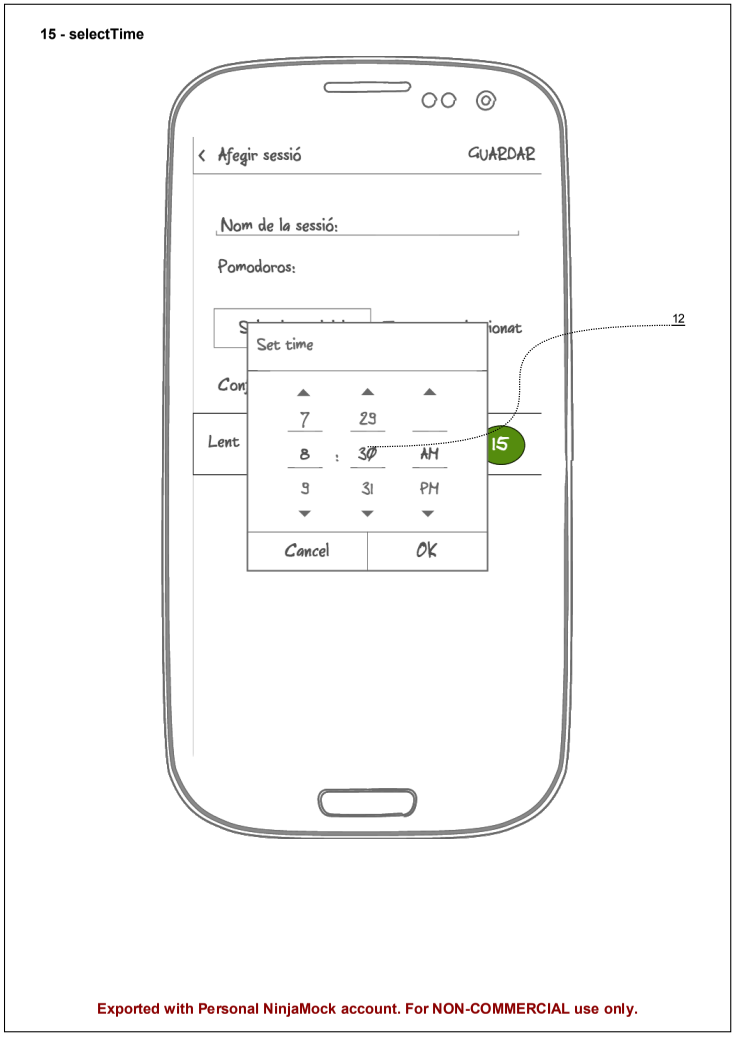

12 - addSession
11
Afegir sessió
GUARDAR
11
Nom de la sessió:
Pomodoros:
15
Seleccionar inici
Temps no seleccionat
Configuració:
Lent
CØ
1Ø
15
Exported with Personal NinjaMock account. For NON-COMMERCIAL use only.

13 - settings



14 - editOrCreateSetting




15 - selectTime
Afegir sessió
GUARDAR
Nom de la sessió:
Pomodoros:
Set time
7
29
8
30
AM
9
31
PM
Cancel
OK
12
15
Exported with Personal NinjaMock account. For NON-COMMERCIAL use only.

5.5 Recorregut d'exemple per tot el sistema

Per entendre el disseny de tots els components i les relacions entre ells, crec que és bo de posar un exemple de funcionalitat i analitzar tot el recorregut que tindrà pel sistema. S'adjunta un diagrama de seqüència per visualitzar millor tot el procés.

Cas d'ús: Afegir usuari a un grup de treball.

1. L'**usuari** administrador indica al **client** Android que vol afegir un usuari al grup. Introdueix el seu username o email.
2. El **client** fa una petició al **servidor**: /People?filter[limit]=10&filter[where][username][like]=userIntroduït&filter[where][email][like]=emailIntroduït
3. El **servidor** processa la petició, fa una consulta a la **base de dades** i retorna el resultat de la petició.
4. El **client** rep el resultat i el processa per mostrar els possibles usuaris a afegir per al text introduït.
5. L'**usuari** administrador selecciona un usuari a afegir i prem acceptar.
6. El **client** fa una petició al **servidor**: /Groups/id/people/rel/fk, on id és l'id del grup i fk és l'id de l'usuari a afegir.
7. El **servidor** processa la petició, actualitza la base de dades i retorna el resultat al client. Després processa la informació del nou usuari per preparar un missatge "push", que després envia al **GCM**. item El client processa la informació rebuda i mostra al client el nou estat dels seus grups.
8. Paral · lelament al punt 8, el **GCM** fa les comprovacions pertinents i envia el missatge als dispositius pertinents de l'**usuari** afegit.
9. El **client**, al mòbil de l'**usuari** afegit, rep el missatge, el processa i mostra una notificació.
10. Si l'**usuari** afegit clica la notificació, el **client** mostra la pestanya de grups.

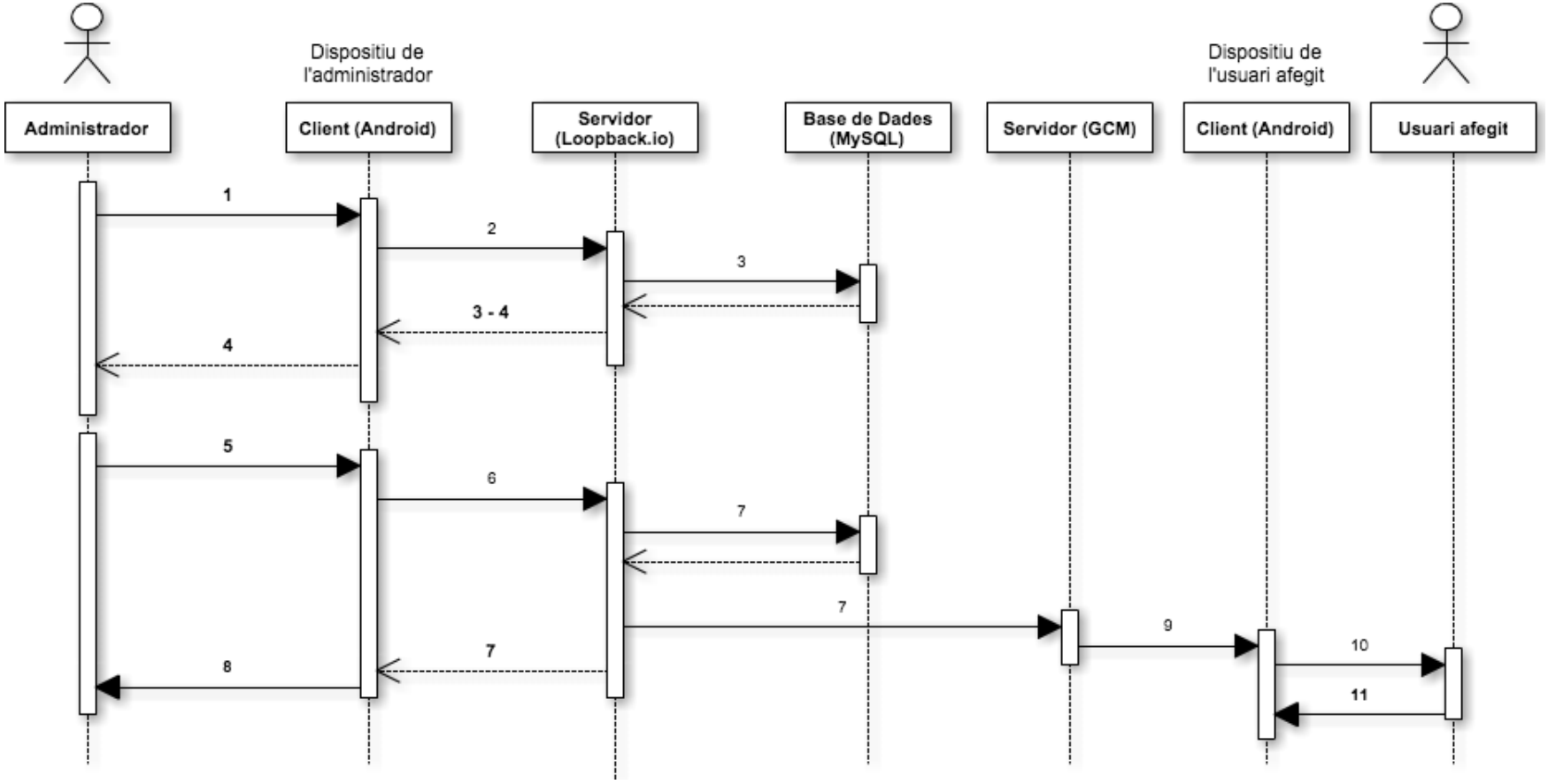


Figura 5.8: Diagrama de seqüència del recorregut d'exemple

Capítol 6

Desplegament

El model de desplegament es pot veure a continuació. En ell podem veure com tant el codi del servidor com el del client Android estan pujats a GitHub per facilitar el desenvolupament. El codi de servidor també està pujat a Heroku, des d'on s'executa l'aplicació i està disponible. La base de dades MySQL també està en una eina associada a Heroku, ClearDB. L'aplicació Android realitza les crides a la API REST del servidor, és a dir, al projecte pujat i desplegat a Heroku. Si l'aplicació Android es desplegués per a que estigués disponible públicament, s'hauria de pujar a Google Play, la botiga d'aplicacions d'Android.

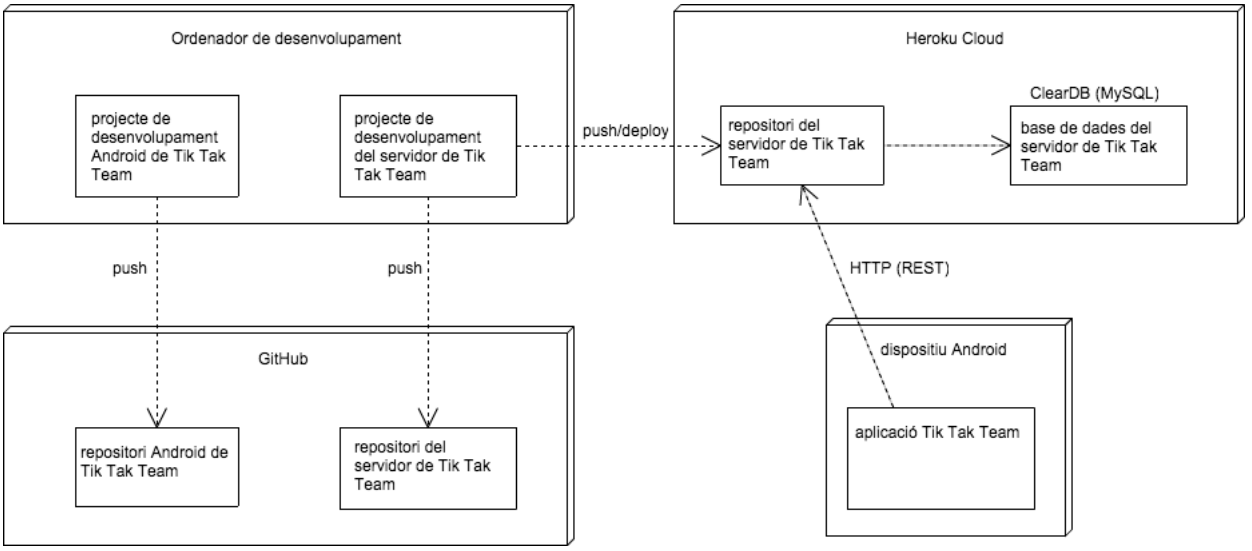


Figura 6.1: Model de desplegament

Capítol 7

Implementació

7.1 Servidor

7.1.1 Loopback.io

Comencem al terminal, la comanda de la línia 1 del codi 7.1 inicia un assistent per començar ràpidament amb el projecte. Es creen un seguit d'arxius i carpetes per evitar picar codi innecessari.
La comanda de la línia 2 és un assistent per afegir una font de dades al sistema.
La comanda de la línia 3 és un assistent per afegir un model al sistema. Al crear-ne un, s'afegeixen, per defecte, les operacions CRUD (Create, Read, Update, Delete) a la REST API.
La comanda de la línia 4 és un assistent per afegir relacions entre models.
La comanda de la línia 5 és un assistent per afegir controls d'accés als models.

```
1 $ slc loopback
2 $ slc loopback:datasource
3 $ slc loopback:model
4 $ slc loopback:relation
5 $ slc loopback:acl
```

Codi 7.1: Comanda per crear el projecte

Models

Com ja hem dit en l'apartat d'arquitectura, els models representen dades d'alguna font d'informació. En el nostre cas, cada model encaixa amb una taula de la base de dades MySQL. La configuració per connectar amb la font d'informació es troba al fitxer datasources.json, on s'especifica la informació necessària (host, base de dades, usuari, password...).

Si passem ja als models, he posat com a exemple el model *group*. Té, per una part, la configuració a group.json i per altra, certes funcionalitats extra afegides a group.js.
A group.json es defineix:

- **name**: El nom (Group).
- **base**: El model base de que hereda (PersistedModel, que és el model base per connectar la informació amb una base de dades).

- **idInjection**: Si es vol assignació automàtica d'IDs (true)
- **properties**: Les diferents propietats del model. Se'n defineix el tipus i si és una propietat necessària. (name, description, adminId).
- **validations**: Per afegir validacions en formularis (tot i que amb la comanda que genera models es crea aquest apartat, encara no està implementat en la última versió de Loopback.io a data d'aquest TFG.
- **relations**: Les relacions amb altres models. (Group hasMany Session, Group hasMany People through Member).
- **acls**: Els permisos necessaris per realitzar cada operació. (Es comença denegant el permís de tots per després otorgar permisos a cada operació explícitament). En el següent apartat s'expliquen els possibles rols existents.
- **methods**: Altres mètodes afegits, a part de les operacions CRUD.

A group.js es deshabiliten els mètodes no utilitzats amb **disableRemoteMethod** i s'afegeix un codi a executar després d'un mètode en concret **afterRemote**.

En el primer cas, després del mètode *'create'*, s'afegeix l'adminId al grup i es crea un objecte del model *Member* que relaciona Person i Group.

En el segon, després del mètode *'link_people'*, s'envia una notificació a l'usuari afegit al grup.

```
module.exports = function(Group) {
  Group.disableRemoteMethod('__create__people', false);
  ...

  Group.afterRemote('create', function (ctx, group, next) {
    var userId = ctx.req.accessToken.userId;
    group.adminId = userId;
    group.save();

    var app = Group.app;
    var Member = app.models.Member;
    Member.create({'personId': userId, 'groupId': group.id}, function (err, member
    ) {
      if (err) console.log(err);
    });
    next();
  });

  var utils = require('../../server/utils');

  Group.afterRemote('prototype.__link__people', function (ctx, member, next) {
    var info = {};
    info.userId = member.personId;
    info.groupId = member.groupId;
    info.groupName = ctx.instance.__data.name;
    utils.notifyGroupAdded(Group, info);
    next();
```

```
27   });
28 };
```

Codi 7.2: group.js

Sistema de rols i controls d'accés

Com ja he comentat, es poden definir controls d'accés per als mètodes de cada model, de manera que només cert tipus d'usuari pugui utilitzar-lo. S'ha utilitzat tres tipus de rols definits per defecte i dos propis:

- **$everyone**: Aquest rol significa es refereix a qualsevol usuari.
- **$authenticated**: Aquest rol significa que l'usuari ha d'haver fet login al sistema.
- **$owner**: Aquest rol significa que l'usuari que realitza l'operació és qui li pertany l'objecte.
- **groupMember**: Un dels dos rols propis, significa que l'usuari ha de ser membre del grup implicat en la operació.
- **groupAdmin**: També un rol propi, significa que l'usuari ha de ser l'administrador del grup implicat en la operació.

Push

Per a què el sistema push explicat prèviament funcioni, hem realitzat el següent: Primer, un arxiu config.js on es guarden les dades de connexió amb GCM.

```
1  exports.appName = 'thePomoAPP';
2
3  exports.appId = 'thePomoAPPId';
4
5  exports.gcmServerApiKey = 'XXXXXX';
6
7  exports.gcmSessionNew = 'session_new';
8  exports.gcmSessionUpdated = 'session_updated';
9  exports.gcmSessionDeleted = 'session_deleted';
10 exports.gcmGroupAdded = 'group_added';
```

Codi 7.3: config.js

Tal com es veu, es guarda el nom que hem decidit per a la app (*thePomoAPP*) i l'ID (*thePomoAPPId*). També hi guardem la gcmServerApiKey, que és la clau que ens genera GCM per poder connectar-nos-hi. Els Strings gcmSessionNew, gcmSessionUpdated, gcmSessionDeleted i gcmGroupAdded que he definit són enviats al missatge de GCM que arriba al client i s'utilitzen perquè el client sàpiga quin tipus de missatge és.

Després, tenim el fitxer push.js, que s'executa al iniciar el servidor i que registra (o actualitza si ja existeix) els diferents clients que podrien interactuar amb el nostre servidor i GCM (en el nostre cas, només una, *thePomoAPP*).

Finalment, tenim el fitxer anomenat utils.js, on hi ha les diferents funcions que envien missatges al client amb el sistema push. Per exemple, podem veure en aquest fragment de codi la funció que notifica a un usuari de que ha estat afegit a un grup. (La funció es crida a la línia 34 del codi 7.2, en el que s'executa després de la funció link_people, que relaciona una persona amb un grup).

```
1  notifyGroupAdded: function(Group, info) {
2      var app = Group.app;
3      var Notification = app.models.notification;
4      var PushModel = app.models.push;
5      var Installations = app.models.installation;
6
7      var arrayMemberIds = [info.userId];
8
9      var devicesFilter = {
10       where: {
11         userId: {
12           inq: arrayMemberIds
13         }
14       }
15     };
16
17     Installations.find(devicesFilter, function(err, installations) {
18       if (err) {
19         console.log("err in Installation.find");
20       }
21
22       var note = new Notification({
23         expirationInterval: 3600, // Expires 1 hour from now.
24         message: config.gcmGroupAdded,
25         data: {
26           groupId: info.groupId,
27           groupName: info.groupName
28         }
29       });
30
31       var arrayDeviceTokens = filterDeviceTokens(installations);
32
33       PushModel.notifyMany(config.appId, 'android', arrayDeviceTokens, note,
     function(err) {
34         if (err) console.log('err in notifyMany');
35       });
36     });
37   }
```

Codi 7.4: Fragment d'utils.js

1. A la funció se li passa (en una estructura anomenada info) l'userId de l'usuari associat al grup i el groupId i el nom de grup al que s'ha associat.

2. A les línies 9-15 es crea un filtre amb l'userId.

3. A la línia 17 es crida a la funció find sobre el model Installation, passant-li el filtre previ. Això retorna, si ha anat bé, installations, que són totes les instal · lacions de l'usuari amb userId.

4. A les línies 22-29 es crea una notificació, amb el temps d'expiració, el missatge i informació extra (groupId i groupName).

5. A la línia 31 es filtra la variable installations per quedar-nos només amb els deviceTokens, ja que és necessari per a la següent crida. (El mètode filterDeviceTokens és un mètode propi que he creat).

6. A la línia 33 es crida notifyMany, per notificar un o més dispositius. Se li passa l'id de la app, el tipus de dispositiu, la llista de deviceTokens i la notificació.

7. Quan el client Android rebi aquest missatge, tindrà la informació suficient per mostrar la informació a l'usuari.

7.2 Base de dades

Quant a implementació, de la base de dades es veu a la Figura 7.1 l'esquema SQL resultant. Cal recordar que aquest es genera automàticament per l'ORM de Loopback (*loopback-datasource-juggler* i més concretament per *loopback-connector-mysql*) a partir dels models definits, tal com s'ha mostrat a l'apartat de disseny.

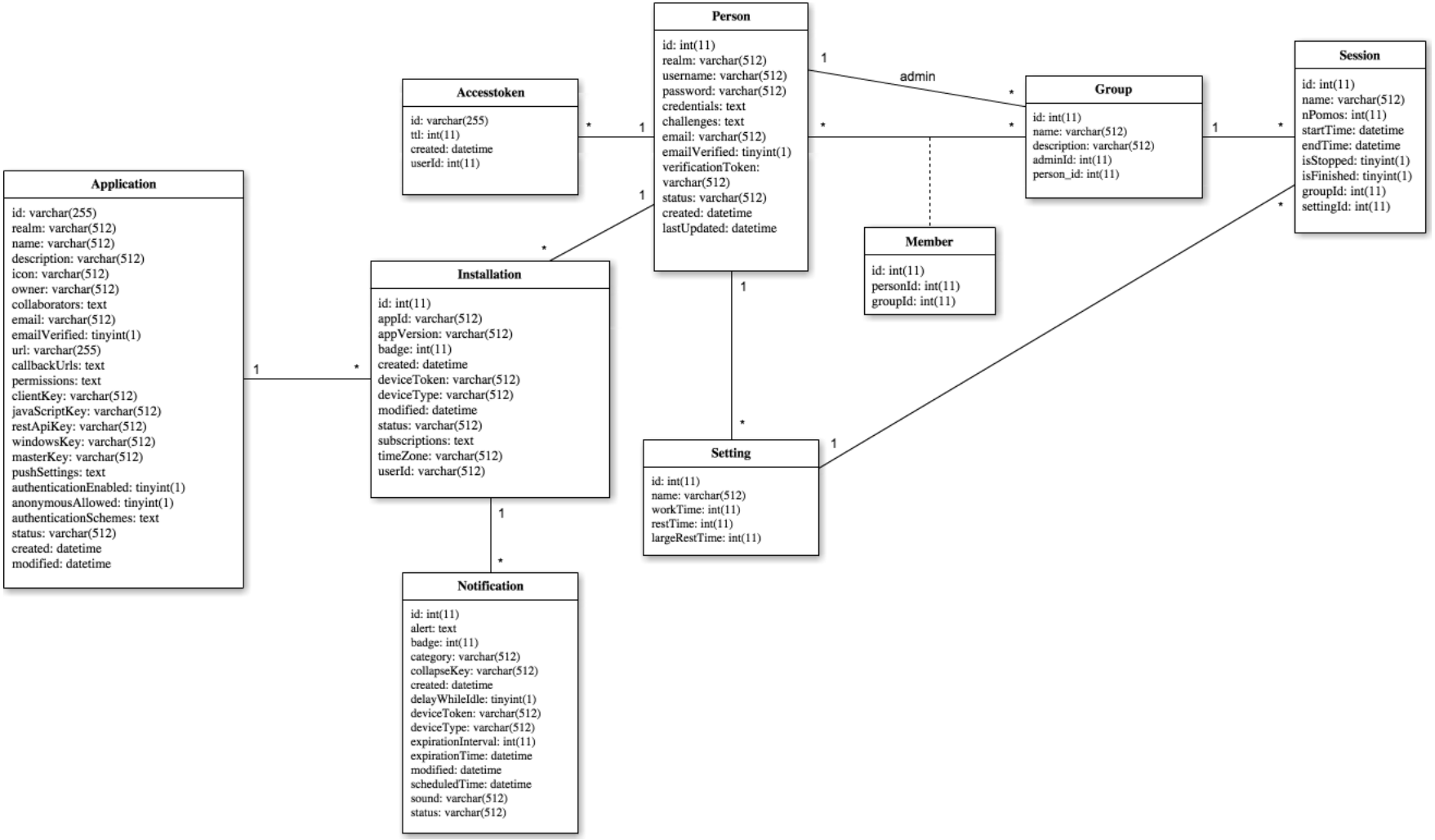


Figura 7.1: Esquema de la base de dades MySQL

7.3 Client Android

Anem ara amb la implementació del client Android.

7.3.1 Llibreries externes per a Android

Primerament, vull anomenar un seguit de biblioteques externes, tant de Google com de tercers, que permeten agilitzar el desenvolupament del software:

- **v7 appcompat**: Biblioteca de suport per tenir el disseny i funcionalitats de Lollipop a versions anteriors.
- **support-v13**: Biblioteca de suport per tenir noves funcionalitats en dispositius a partir de l'API v13.
- **CircleView**: Vista personalitzada que mostra un text amb un cercle al voltant.
- **om.melnykov:floatingactionbutton**: Vista personalitzada que mostra el *floating action button* característis d'Android Lollipop.
- **com.jpardogo.materialtabstrip:library**: Biblioteca que permet afegir *tabs* amb Material Design.
- **OkHttp**: Substitut del client HTTP per defecte. És més eficient.
- **Retrofit**: Biblioteca per gestionar crides a APIs.
- **Gson**: Biblioteca per serialitzar i deserialitzar objectes Java.
- **Realm**: Base de dades NoSQL per Android.
- **android.support.v7.cardview**: Biblioteca de suport per a mostrar les Cards de Lollipop.
- **org.adw.library:discrete-seekbar**: Vista personalitzada per seleccionar un valor.
- **Material DateTime Picker**: Biblioteca per implementar la selecció de dates i temps.
- **net.danlew:android.joda**: Biblioteca per utilitzar les classes Joda Time de gestió del temps.
- **loopback-sdk-android**: SDK del framework Loopback.io per Android que permet comunicar-se més fàcilment amb el servidor.
- **play-services**: Biblioteca per utilitzar els Google Play Services.

7.3.2 Consumint l'API: Retrofit

L'API per connectar-nos al servidor es consumeix mitjançant la biblioteca externa Retrofit. Exposaré el cas complet del model Session per mostrar els diferents elements que són necessaris per utilitzar Retrofit i aconseguir passar d'una simple crida al servidor a tenir objectes Java que utilitza el client.
Primerament, al codi 7.5 es veu el *mapping* que es realitza per passar de Java a JSON i de JSON a Java. Les classes DTO s'han generat automàticament gràcies a l'eina online `http://www.jsonschema2pojo.org/` a partir del JSON resultant de la REST API. Les annotacions @Expose signifiquen que l'atribut ha de sortir reflexat al JSON.

Després, al codi 7.6 es veu com es declaren els diferents endpoints com a funcions. Les annotacions @GET, POST i DELETE són propies de Retrofit, que ell interpreta internament. Per exemple, findById equival a

”GET /Sessions/id”. En els paràmetres de la funció hi ha l'id de la sessió que volem fer GET (@Path indica que l'atribut es correspon a l'id del path de la url) i un Callback¡SessionDTO¿, perquè aquesta crida ens retorna una sessió, que queda ”mapejada”a un SessionDTO. En el cas de create, @Body significa que tot el JSON resultant de l'objecte es passa pel body. En el cas de delete, es retorna un Callback¡ResponseCallback¿, que és un tipus de callback de Retrofit expressament per crides que no retornen res.

Finalment al codi 7.8 hi ha el ”servei”. El que es fa és definir una funció per a cada mètode on es realitza la crida. Aquest servei hereda d'un servei base (codi 7.7) on se li diu a Retrofit la url base de l'API. He creat la funció setAuthInterceptor per afegir el token d'accés a les crides que el necessitin. A codi 7.9 es pot veure un exemple de la crida al servei. Es veu com el callback té success i failure com a possibles funcions que es poden executar al rebre el resultat, depenent de si és favorable o no.

```
public class SessionDTO {

    @Expose
    private String name;
    @Expose
    private int nPomos;
    ...

    public SessionDTO(String name, int nPomos, String startTime, String endTime,
    int groupId, Integer settingId) {
        this.name = name;
        ...
    }

    //Getters i setters
```

Codi 7.5: SessionDTO.java

```
public interface SessionsInterface {

    @GET("/Sessions/{id}")
    void findById(@Path("id") int id, Callback<SessionDTO> callback);

    @POST("/Sessions")
    void create(@Body SessionDTO sessionDTO, Callback<SessionDTO> callback);

    @DELETE("/Sessions/{id}")
    void delete(@Path("id") int id, Callback<ResponseCallback>
    responseCallbackCallback);
}
```

Codi 7.6: SessionsInterface.java

```
public abstract class BaseService {

    protected RestAdapter restAdapter = new RestAdapter.Builder()
            .setEndpoint(Constants.API_ENDPONT)
            .build();

```

```
    protected RequestInterceptor authInterceptor;
    protected String ttl = "31556926";

    public interface OnRetrofitError {
        void onError(RetrofitError error);
    }

    protected void setAuthInterceptor(final String token) {
        authInterceptor = new RequestInterceptor() {
            @Override
            public void intercept(RequestFacade request) {
                request.addQueryParam("access_token", token);
            }
        };
        restAdapter = new RestAdapter.Builder()
                .setEndpoint(Constants.API_ENDPONT)
                .setRequestInterceptor(authInterceptor)
                .build();
    }
}
```

Codi 7.7: BaseService.java

```
public class SessionsService extends BaseService{
    ...

    public void findById(String token, int id, Callback<SessionDTO> callback) {
        setAuthInterceptor(token);
        SessionsInterface sessionsInterface = restAdapter.create(SessionsInterface
    .class);
        sessionsInterface.findById(id, callback);
    }

    public void create(String token, SessionDTO sessionDTO, Callback<SessionDTO>
    callback) {
        setAuthInterceptor(token);
        SessionsInterface sessionsInterface = restAdapter.create(SessionsInterface
    .class);
        sessionsInterface.create(sessionDTO, callback);
    }

    public void delete(String token, int id, Callback<ResponseCallback> callback)
    {
        setAuthInterceptor(token);
        SessionsInterface sessionsInterface = restAdapter.create(SessionsInterface
    .class);
        sessionsInterface.delete(id, callback);
    }
```

```
}
```

Codi 7.8: SessionsService.java

```
SessionsService.getInstance().findById(Utils.getToken(getApplicationContext()),
    intSessionId, new Callback<SessionDTO>() {
                    @Override
                    public void success(SessionDTO sessionDTO, Response response)
    {
                        refreshAlarm(sessionDTO);
                    }

                    @Override
                    public void failure(RetrofitError error) {

                    }
                });
```

Codi 7.9: Exemple de crida a SessionService (GcmIntentService.java)

Base de dades local: Realm

Realm és una base de dades NOSQL que funciona molt bé. S'ha utilitzat per guardar de forma local Sessions Privades, fer una *cache* de les sessions online i guardar Configuracions. Si per exemple mirem al codi 7.10, veiem la classe que representa una Sessió Privada. Simplement extén de RealmObject i té els seus atributs, getters i setters. L'annotació @PrimaryKey defineix quin és l'atribut que serveix de clau primària per a la "taula". Al codi 7.11 veiem un fragment de DBHandler, que és la classe que he creat per manejar les interaccions amb la base de dades local. S'hi pot veure com es crea, s'obté i s'elimina una Sessió Privada.

```
public class AloneSession extends RealmObject {

    @PrimaryKey
    private String name;
    private int num;
    private int state;
    private String settingUuid;

    // Getters i setters
}
```

Codi 7.10: AloneSession.java

```
public void createAloneSession(final String name, final int num, final int state,
    final String settingUuid) throws RealmException {
        realm.executeTransaction(new Realm.Transaction() {
            @Override
            public void execute(Realm realm) {
                AloneSession aloneSession = realm.createObject(AloneSession.class)
    ;
                aloneSession.setName(name);
```

```
                aloneSession.setNum(num);
                ...
            }
        });
    }

public AloneSession getAloneSession(String name) {
    RealmQuery<AloneSession> query = realm.where(AloneSession.class);
    query.equalTo("name", name);
    return query.findFirst();
}

public void deleteAloneSession(final String name) throws RealmException {
    try {
        realm.executeTransaction(new Realm.Transaction() {
            @Override
            public void execute(Realm realm) {
                AloneSession aloneSession = getAloneSession(name);
                aloneSession.removeFromRealm();
            }
        });
    } catch (RealmException e) {
        throw e;
    }
}
```

Codi 7.11: Fragment de DBHandler.java

Notificacions

La gestió de notificacions funciona de la següent manera: al rebre un missatge de GCM, GcmReceiver.java rep l'event (codi 7.12) i fa que GcmIntentService.java el gestioni. Aquest últim comença comprovant la correctesa del missatge, després extreu les dades del JSON per saber quin tipus de missatge és i quina informació porta, després dóna format als textos de la notificació, si s'escau i, finalment, realitza una acció (mostrar la notificació o en el cas de borrat de sessió, eliminar-la localment sense mostrar res). Al codi 7.13 podem veure com s'extreu la informació a "parseContentï com es formateja el text de la notificació, amb una funció auxiliar que he definit.

```
public class GcmReceiver extends WakefulBroadcastReceiver {

    public static final String LOGTAG = GcmReceiver.class.getSimpleName();

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.v(LOGTAG, "onReceive");
        // Explicitly specify that GcmIntentService will handle the intent.
        ComponentName comp = new ComponentName(context.getPackageName(),
                GcmIntentService.class.getName());
        // Start the service, keeping the device awake while it is launching.
```

```
        startWakefulService(context, (intent.setComponent(comp)));
        setResultCode(Activity.RESULT_OK);
    }
}
```

Codi 7.12: GcmReceiver.java

```
private void parseContent(Bundle extras) {
        try {
            JSONObject data = new JSONObject(extras.getString(Constants.GCM_DATA))
    ;
            gcmMessage = extras.getString(Constants.GCM_MESSAGE);
            if (data.has("groupId")) {
                groupId = data.getString("groupId");
            }
            if (data.has("groupName")) {
                groupName = data.getString("groupName");
            }
            ...
        } catch (JSONException e) {
            Log.e(LOGTAG, e.getMessage());
        }
    }

    private void formatNotification() {
        switch (gcmMessage) {
            case Constants.GCM_SESSION_NEW:
                notificationTitle = Utils.formatNotification(getApplicationContext
    (), R.string.session_new_title, groupName);
                notificationText = Utils.formatNotification(getApplicationContext
    (), R.string.session_new_content, sessionName);
                break;
            // segueixen els altres casos
        }
    }
```

Codi 7.13: Fragment de GcmIntentService.java

El procés de gestió de sessions

La gestió de les sessions es basa en dos classes, Pomodoro.java i GroupPomodoro.java, que hereda de la primera. La classe Pomodoro s'utilitza pels pomodoros privats i la de GroupPomodoro pels de grup.
En el primer cas, el cicle de gestió de la sessió és:

- **setSession**: La classe rep la informació de la sessió que estarà en funcionament.
- **startSession**: S'activa quan l'usuari fa click a INICIO a la seva pestanya de sessions privades. S'inicialitzen valors de l'objecte Pomodoro i es crida a *startPhase*.

- **startPhase**: Comença una fase, es comprova l'estat de la sessió (Treball, descans o descans llarg), s'assigna la duració de la fase i es canvia el color de la roda del temps. A partir d'aquí, s'inicia un comptador que actualitza el temps restant cada segon. Quan la fase acaba, es passa a *betweenPhase*.
- **betweenPhase**: Si ja no queden fases restants, es crida *finishSession*. Sinó, s'actualitza la fase vinent, els pomodoros que que han passat, es retorna una senyal de fi de la fase (*callback*) i es torna a cridar *startPhase*.
- **finishSession**: S'actualitza visualment la roda de temps i es retorna una senyal de la fi de la sessió (*callback*).

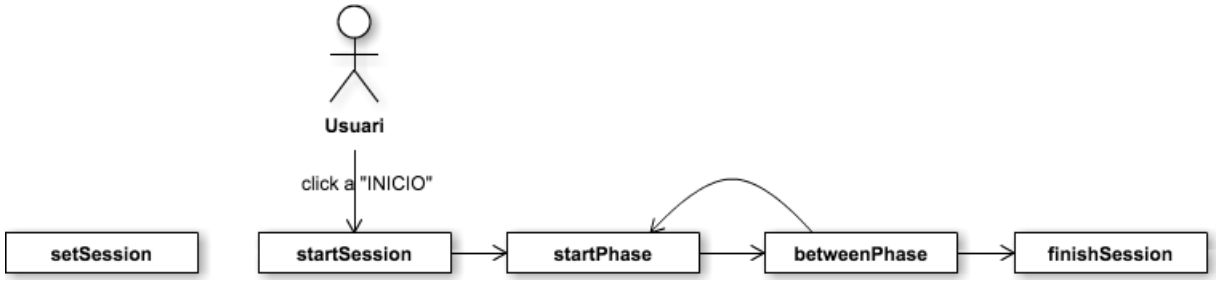


Figura 7.2: Seqüència de la classe Pomodoro.java

En el segon cas (cal dir que aquí l'inici de la sessió és imminent, ja que només es creen objectes GroupPomodoro de sessions quan és la data d'inici o quan ja s'han iniciat:

- **setSession**: Ídem que a Pomodoro.java. En aquest cas es guarda la data inicial de programació.
- **getSessionState**: Es calcula, a partir del moment actual i de la data de programació, l'estat actual de la sessió: és a dir, la fase actual i el segon en el que es troba. S'ajuda de la funció *countPhase*, que és recursiva.
- **restartPhaseFromState**: A partir del calculat a *getSessionState*, es modifica el color de la roda de temps i s'inicia el timer, similar a com ho fa startPhase a Pomodoro.java.
- **betweenPhase**: Si ja no queden fases restants, es crida *finishSession*. Sinó, s'actualitza la fase vinent, els pomodoros que que han passat i es torna a cridar *startPhase*.
- **finishSession**: S'actualitza visualment la roda de temps i es retorna una senyal de la fi de la sessió (*callback*).

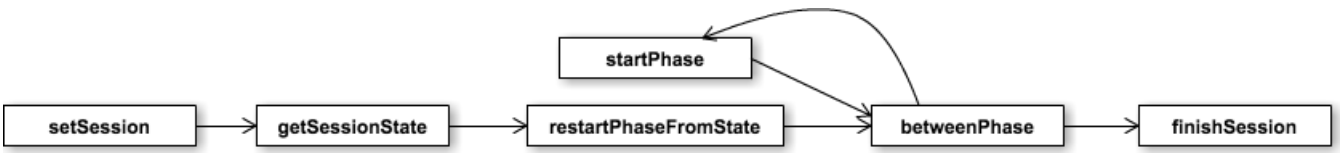


Figura 7.3: Seqüència de la classe GroupPomodoro.java

```
private long countPhase() {
    long leftSeconds = 0;
    int auxPhase = calculateResults.currentPhase;
    if (calculateResults.phaseNumber % 2 != 0) { // WORK
        int workSeconds = workTime.get(Calendar.MINUTE) * 60;
        leftSeconds = calculateResults.differenceInSeconds - workSeconds;
        auxPhase = WORK;
        ++calculateResults.currentPomodoro;
    } else if (calculateResults.phaseNumber % 8 != 0) { // BREAK
        int breakSeconds = breakTime.get(Calendar.MINUTE) * 60;
        leftSeconds = calculateResults.differenceInSeconds - breakSeconds;
        auxPhase = BREAK;
    } else { // LARGE BREAK
        int largeBreakSeconds = largeBreakTime.get(Calendar.MINUTE) * 60;
        leftSeconds = calculateResults.differenceInSeconds - largeBreakSeconds;
        auxPhase = LARGE_BREAK;
    }
    calculateResults.currentPhase = auxPhase;
    if (leftSeconds <= 0) {
        return -leftSeconds;
    }
    calculateResults.differenceInSeconds = leftSeconds;
    ++calculateResults.phaseNumber;
    return countPhase();
}
```

Codi 7.14: countPhase de GroupPomodoro.java

Capítol 8

Proves del sistema

8.1 Usabilitat

Per a comprovar si la usabilitat del client Android és bona, s'ha realitzat un petit test d'usabilitat a 5 persones.

Taula 8.1: Perfils dels usuaris utilitzats a la proves

Persona	Edat	Professió	Comentaris
C	56	Comptable, administrativa	
S	57	Bancari	Familiarització amb les noves tecnologies molt baixa.
M	20	Estudiant d'Antropologia	
A	23	Atenció telefònica	
E	50	Emprenedora, comercial	

Primerament s'ha contextualitzat als participants i se'ls hi ha explicat les possibilitats de l'aplicació. Aquí es mostren les preguntes o accions demanades i els resultats apreciats:

1. Què creus que pots fer a la pestanya "solo"?
    - **C**: Em puc programar les estones de feina i de descans per mi sola.
    - **S**: Crec que es refereix al temps que utilitzo per mi sol.
    - **M**: És el que fas que t'organitzes per tu sol.
    - **A**: És el mode clàssic, per exemple quan estudies.
    - **E**: Per un sol usuari que es programa les seves coses.
2. Crea't un nou usuari i fes login (entra al sistema).
    - **C**: 1 min. 51 s. Esperava que fent click a login funcionés directament
    - **S**: 1 min. 23 s.
    - **M**: 49 s.
    - **A**: 1 min. 30 s. Ha tardat per errors en la introducció de la contrasenya per teclat.
    - **E**: 1 min. 55 s.
3. Crea un grup de nom "Testï afegeix l'usuari Manolo.

- **C**: 34 s.
- **S**: 1 min. 4 s.
- **M**: 17 s.
- **A**: 15 s.
- **E**: 54 s.

4. Crea una configuració de nom Çonfig1"amb duració 25' 5' 15'.
    - **C**: 54 s. Volia posar els minuts de duració des del teclat.
    - **S**: 48 s.
    - **M**: 21 s.
    - **A**: 19 s.
    - **E**: 1 min. 8 s.
5. Programa una sessió de grup al grup "Test", de nom "Prova", que tingui 2 pomodoros per a les 10:00h el 31-10-2015 i que la seva configuració sigui "Per defecte".
    - **C**: 59 s.
    - **S**: 1 min. 04 s.
    - **M**: 55 s.
    - **A**: 30 s.
    - **E**: 52 s.
6. Fes logout (surt del sistema).
    - **C**: 9 s. Primer ha fet click a l'email i no al botó de sortir.
    - **S**: 18 s.
    - **M**: 4 s.
    - **A**: 4 s.
    - **E**: 4 s.

De les observacions realitzades i els comentaris dels participants se n'ha tret les següents conclusions:

- S'ha apreciat la necessitat d'un login social (Facebook, Google, Linkedin...) que permetria unificar el registre i la introducció de dades en un sol click.
- Hauria d'haver-hi la possibilitat d'introduïr els minuts d'una configuració personalitzada per teclat, juntament amb el selector.
- Els participants han comentat que un cop s'ha fet servir l'aplicació per primer cop, pensaven que en un hipotètic segon cop els seria molt més fàcil. Això s'evidència en la tasca de programar una sessió, que tot i ser més llarga i difícil, no té uns temps molt més elevats. Això suporta la idea de que el disseny d'un tema comú i coherent per tota la app funciona. A més, els usuaris de menys edat, més habituats a l'ús d'aplicacions mòbils tenen uns temps considerablement millors. Seria important crear un tutorial el primer cop que un usuari utilitza l'app, per familiaritzar-lo.
- Algun text s'hauria de canviar de color per tenir un millor contrast i millorar la visibilitat en persones més grans o amb problemes de visió.

8.2 Accessibilitat

Com a prova d'usabilitat, s'ha provat el dispositiu de forma física en un Nexus 4. De forma virtual, gràcies a Genymotion, s'ha provat a un Galaxy Nexus, Nexus 4, Nexus 5, Samsung Galaxy S2 i altres dispositius Samsung. L'aplicació funciona correctament a excepció dels dispositius Samsung amb Android 4.2.2., degut a les modificacions de la versió que realitza Samsung.

Capítol 9

Planificació temporal

9.1 Descripció de les tasques

9.1.1 Durada del projecte

Aproximadament, la durada del projecte és de quatre mesos i mig, des de l'inici de l'assignatura de GEP el 16/02/2015, fins al 03/07/2015, com a màxim. El projecte serà realitzat generalment per una persona, amb el suport en el disseny gràfic d'una altra (soci).

9.1.2 Planificació i Diagrama de Gantt

Per a la bona comprensió de la distribució del temps entre els diferents sprints (o iteracions) i tasques s'ha realitzat un diagrama de Gantt que engloba tot el projecte. És important considerar que tot i aquesta planificació, els temps poden variar dels mostrats en el diagrama, ja que la metodologia Agile permet adaptar-nos al que sigui necessari en un moment determinat. Igualment, és important haver especificat amb rigor els temps previstos per anar "sobre rails". Els temps de planificació i conclusió de cada sprint no es mostren, però es consideren implícits en el temps de cada sprint. Anem doncs a explicar les diferents activitats del desenvolupament del projecte:

Gestió del Projecte (GEP)

Aquesta assignatura associada al Projecte Final de Grau en marca l'inici. Les seves tasques són els diferents lliurables que hi ha definits. El seu calendari ha estat extret de la guia de l'assignatura, el qual assegura que la previsió de temps és correcta.

Sprint 1: Anàlisi i requisits

En aquest primer Sprint s'especificaran els requisits que ha de complir el sistema, els casos d'ús que existiran, es definirà el client al qual va dirigit aquest, quines restriccions existeixen i quines expectatives tenim.

Sprint 2: Disseny del software

En el segon Sprint es passarà a dissenyar el software. Això inclou l'esquema de la base de dades, el disseny del backend i l'especificació de la API, i el disseny de l'aplicació Android (inclòs el mapa navegacional).

Es considerarà superada si al final del Sprint es tenen els documents suficients per interpretar de manera conceptual el disseny total de l'aplicació.

Sprint 3: Disseny gràfic

En aquest tercer Sprint, el dissenyador gràfic realitzarà el disseny gràfic de l'aplicació Android. Es considerarà superada si al final del sprint es té el disseny de totes les pantalles de l'aplicació.

Sprint 4: Desenvolupament base de la API

En aquest Sprint es desenvoluparà bona part de la API REST. Es considerarà superada si al final del Sprint es funcionen tots els endpoints desenvolupats en aquest Sprint.

Sprint 5: Desenvolupament base Android (offline)

En aquest Sprint es desenvoluparan les funcionalitats offline de l'aplicació Android. Es considerarà superada si al final del Sprint es té una versió bàsica de l'aplicació Android que permeti utilitzar un pomodoro personal. Al final d'aquest Sprint s'aconseguirà una versió mostrable per comprovar la resposta del potencial usuari final.

Sprint 6: Desenvolupament base Android (online)

En aquest Sprint es desenvoluparan les funcionalitats online de l'aplicació Android. Es considerarà superada si al final del Sprint es té una versió de l'aplicació Android que permeti utilitzar els pomodoros en grup. Al final d'aquest Sprint s'aconseguirà una nova versió mostrable per comprovar la resposta del potencial usuari final.

Sprint 7: Desenvolupament del sistema de notificacions

En aquest Sprint es desenvoluparà el sistema de notificacions tant al servidor com a l'aplicació Android. Es considerarà superada si al final del Sprint es té una versió de l'aplicació Android que permeti rebre notificacions. Al final d'aquest Sprint s'aconseguirà una nova versió mostrable per comprovar la resposta del potencial usuari final.

Documentació i tancament

En aquesta última fase es completarà la documentació del projecte. També s'inclou la defensa del projecte. La planificació temporal es troba en un Diagrama de Gantt a l'ANNEX.

9.1.3 Estimació en hores

Nombre del recurso	Trabajo
Enric	454,5 horas
Lliurable 1: Abast del projecte	9,25 horas
Lliurable 2: Planificació temporal	8,25 horas
Lliurable 3: Gestió econòmica i sostenibilitat	9,25 horas
Lliurable 4: Presentació preliminar	6,25 horas
Lliurable 5: Contextualització i bibliografia	15,25 horas
Lliurable 6: Plec de condicions	8,5 horas
Lliurable 7: Document final i presentació oral	18,25 horas
Definició del client	3,5 horas
Especificació de requisits	10 horas
Casos d'ús del sistema	10 horas
Detecció de restriccions	3,5 horas
Espectatives del projecte	3,5 horas
Disseny de la BD	3,5 horas
Investigar sobre el disseny d'APIs REST	3,5 horas
Disseny de l'API	7 horas
Investigar sobre el framework Loopback	7 horas
Disseny del servidor	7 horas
Mapa navegacional de l'app Android	7 horas
Disseny de l'aplicació Android	7 horas
Desenvolupament API	70 horas
Testing dels endpoints	4 horas
Desenvolupament Android offline	70 horas
Testing de resposta a casos d'ús	4 horas
Desenvolupament Android online	70 horas
Testing de resposta a casos d'ús	4 horas
Investigar sobre la integració de notificacions amb Loopback	8 horas
Investigar sobre la integració de notificacions amb Android	8 horas
Desenvolupament notificacions servidor	30 horas
Desenvolupament notificacions Android	25 horas
Testing de notificacions	2 horas
Documentació i tancament	12 horas
Dissenyador	60 horas
Disseny de logo, icones i drawables	25 horas
Disseny gràfic de l'aplicació Android	35 horas

Figura 9.1: Estimació en hores

9.2 Recursos

Els recursos que s'utilitzaran per al desenvolupament del projecte són:

Taula 9.1: Recursos del projecte

Recursos humans	Finalitat
Una persona: emprenedor, arquitecte tècnic i desenvolupador amb una dedicació de 25h setmanals	Desenvolupar el projecte
Una persona: emprenedor i dissenyador gràfic	Dissenyar la part gràfica del projecte
Recursos materials	**Finalitat**
Ordenador portàtil Packard Bell Easynote TK85 amb Windows 8.1	Desenvolupar el projecte
LG Nexus 4	Desenvolupament de l'aplicació mòbil Android
Genymotion (màquina virtual)	Desenvolupament de l'aplicació mòbil Android
Sublime Text 3 (IDE)	Desenvolupament del backend
Android Studio (IDE)	Desenvolupament de l'aplicació mòbil Android
Loopback Framework	Desenvolupament del backend
Android SDK	Desenvolupament de l'aplicació mòbil Android
Heroku	Hosting
Google Drive	Desenvolupament de documentació, eina de seguiment
Git i GitHub	Control de versions del backend i l'aplicació mòbil Android
E-mail	Comunicació i seguiment
Microsoft Project	Planificar el projecte (Gantt)
Postman	REST Client per analitzar les rutes de la API

9.3 Valoració d'alternatives i pla d'acció

Com que pel desenvolupament del projecte s'utilitza la metodologia Agile - SCRUM, al final de cada sprint es pot valorar com ha anat aquest i si s'ha de canviar alguna cosa en el futur. Per tant, en el cas de que apareguessin problemes es podrien solucionar fàcilment i ràpidament. Es podria també en aquest moment, consultar amb el tutor del treball quina direcció agafar i replantejar els sprints següents amb ell. Tot i que l'últim sprint està aprop de la data de finalització del projecte, ja s'han assignat uns temps sobradament amplis a les últimes etapes del projecte, assumint que podria sobrar temps. De la mateixa manera, si un sprint intermedi s'acaba abans, es començarà amb l'altre per tal d'avançar feina.

9.4 Diagrama de Gantt

	Nombre de tarea	Duració	Comienzo	Fin	Predecesoras	Nombres de los recursos	Risc
10	**Iteració 1: Anàlisi i requisits**	**7 días**	**sáb 21/02/15**	**dom 01/03/15**			
11	Definició del client	1 día	sáb 21/02/15	sáb 21/02/15		Enric	Mitjà
12	Especificació de requisits	3 días	dom 22/02/15	mar 24/02/15	11	Enric	Baix
13	Casos d'ús del sistema	3 días	mié 25/02/15	vie 27/02/15	12	Enric	Baix
14	Detecció de restriccions	1 día	sáb 28/02/15	sáb 28/02/15	13	Enric	Baix
15	Espectatives del projecte	1 día	dom 01/03/15	dom 01/03/15	14	Enric	Baix
16	**Iteració 2: Disseny del software**	**11 días**	**lun 02/03/15**	lun 16/03/15	**10**		
17	Disseny de la BD	1 día	lun 02/03/15	lun 02/03/15		Enric	Mitjà
18	Investigar sobre el disseny d'APIs REST	1 día	mar 03/03/15	mar 03/03/15		Enric	Baix
19	Disseny de l'API	2 días	mié 04/03/15	jue 05/03/15	18	Enric	Alt
20	Investigar sobre el framework Loopback	2 días	vie 06/03/15	lun 09/03/15		Enric	Baix
21	Disseny del servidor	2 días	mar 10/03/15	mié 11/03/15	20	Enric	Alt
22	Mapa navegacional de l'app Android	2 días	jue 12/03/15	vie 13/03/15		Enric	Baix
23	Disseny de l'aplicació Android	2 días	sáb 14/03/15	lun 16/03/15	22	Enric	Mitjà
24	**Iteració 3: Disseny gràfic**	**12 días**	**lun 02/03/15**	**mar 17/03/15**	**10**		
25	Disseny de logo, icones i drawables	5 días	lun 02/03/15	vie 06/03/15		Dissenyador	Baix
26	Disseny gràfic de l'aplicació Android	7 días	sáb 07/03/15	dom 15/03/15		Dissenyador	Mitjà
27	**Iteració 4: Desenvolupament base de la API**	**20 días**	**mar 17/03/15**	**lun 13/04/15**	**16**		
28	Desenvolupament API	20 días	mar 17/03/15	lun 13/04/15		Enric	Alt
29	Testing dels endpoints	20 días	mar 17/03/15	lun 13/04/15		Enric	Baix

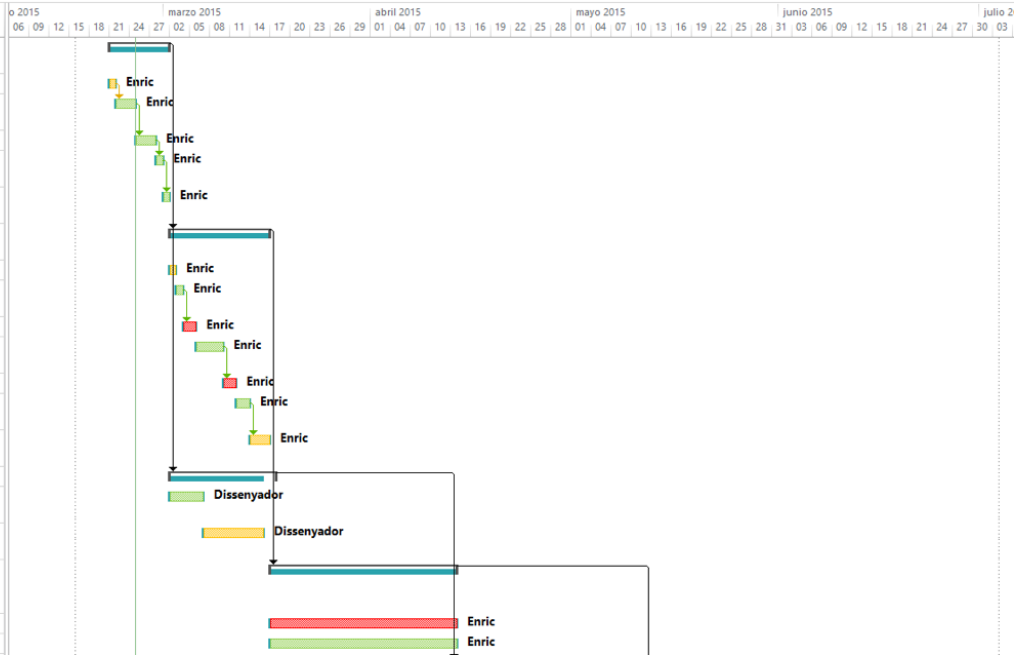
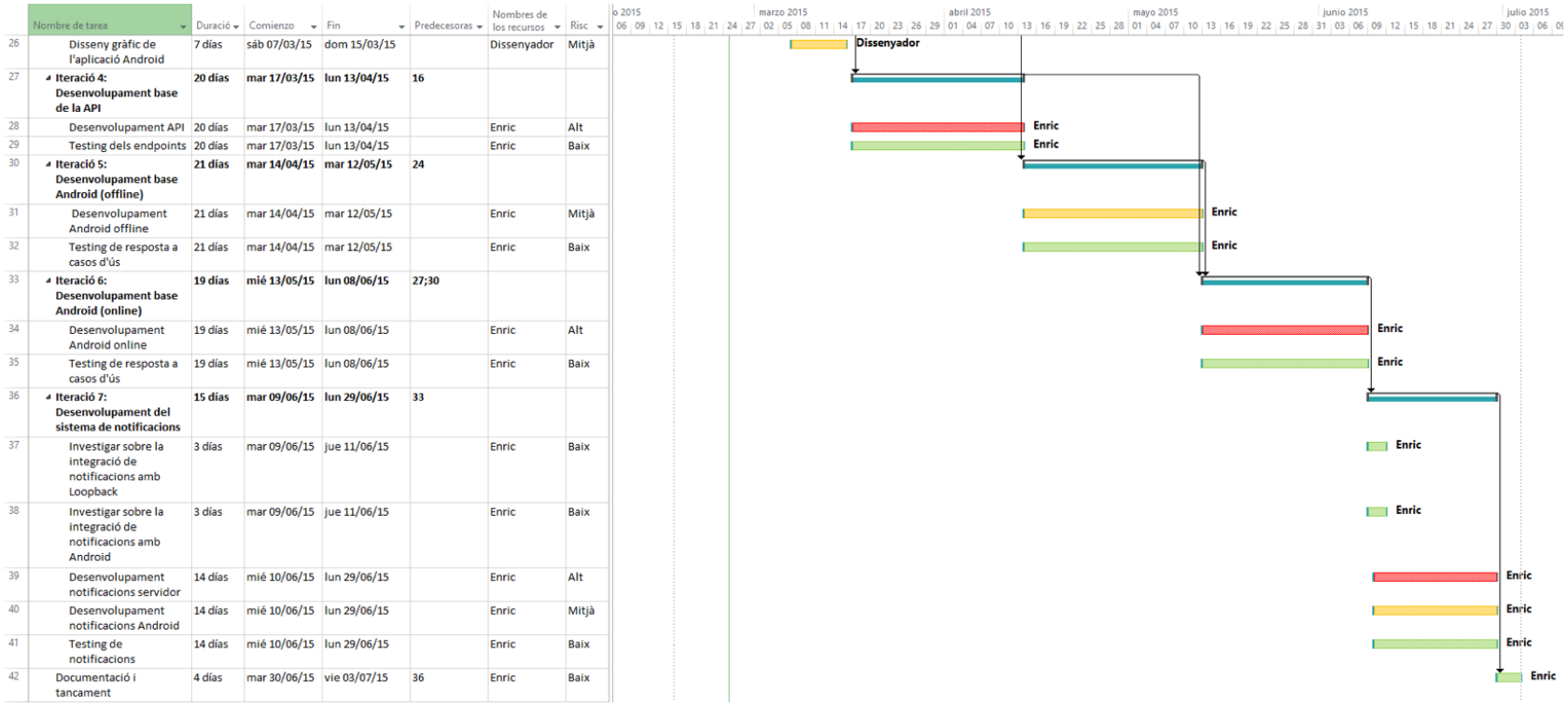


Figura 9.2: Diagrama de Gantt - primera part

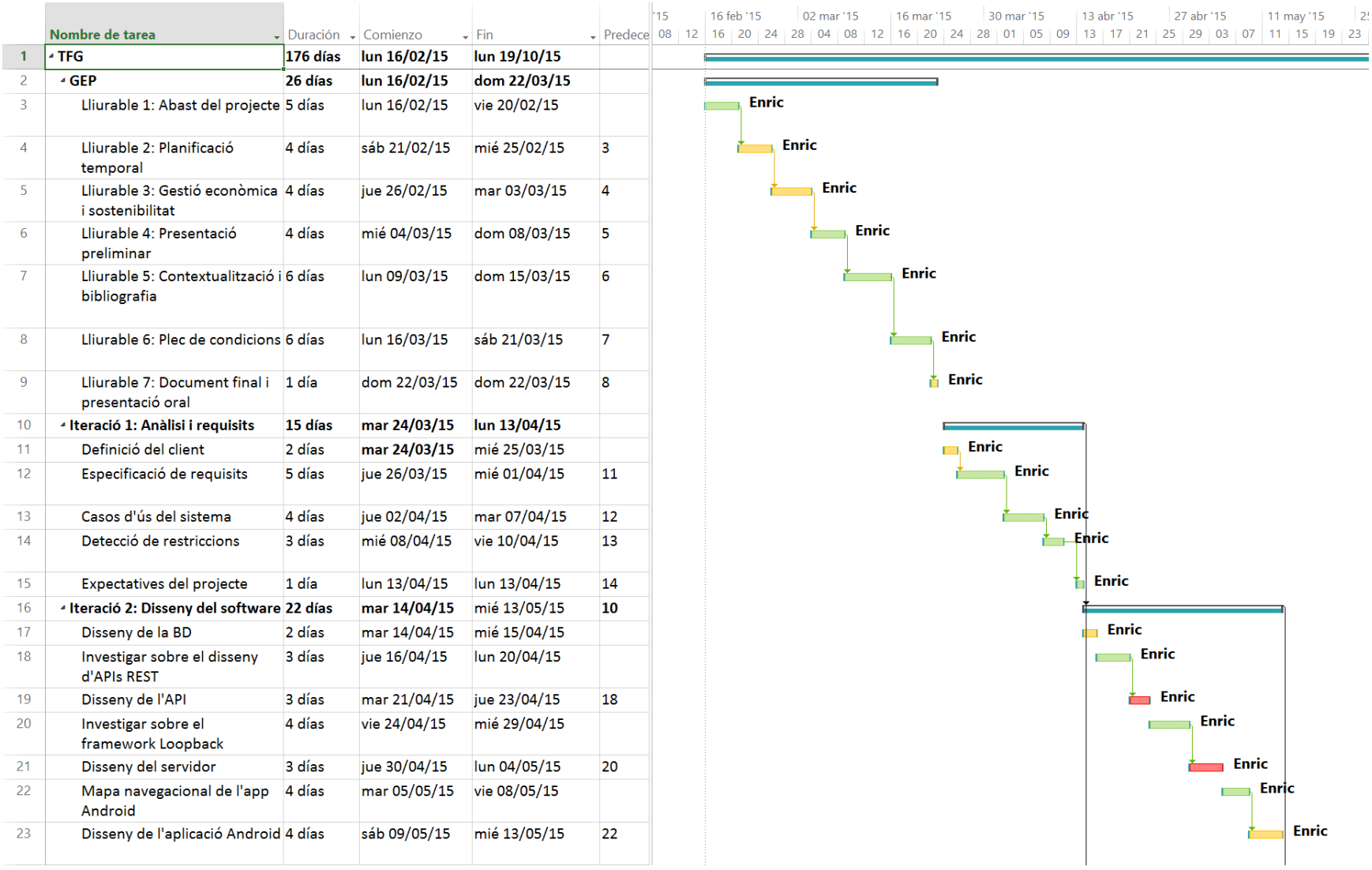


	Nombre de tarea	Duració	Comienzo	Fin	Predecesoras	Nombres de los recursos	Risc
26	Disseny gràfic de l'aplicació Android	7 días	sáb 07/03/15	dom 15/03/15		Dissenyador	Mitjà
27	**Iteració 4: Desenvolupament base de la API**	**20 días**	**mar 17/03/15**	**lun 13/04/15**	**16**		
28	Desenvolupament API	20 días	mar 17/03/15	lun 13/04/15		Enric	Alt
29	Testing dels endpoints	20 días	mar 17/03/15	lun 13/04/15		Enric	Baix
30	**Iteració 5: Desenvolupament base Android (offline)**	**21 días**	**mar 14/04/15**	**mar 12/05/15**	**24**		
31	Desenvolupament Android offline	21 días	mar 14/04/15	mar 12/05/15		Enric	Mitjà
32	Testing de resposta a casos d'ús	21 días	mar 14/04/15	mar 12/05/15		Enric	Baix
33	**Iteració 6: Desenvolupament base Android (online)**	**19 días**	**mié 13/05/15**	**lun 08/06/15**	**27;30**		
34	Desenvolupament Android online	19 días	mié 13/05/15	lun 08/06/15		Enric	Alt
35	Testing de resposta a casos d'ús	19 días	mié 13/05/15	lun 08/06/15		Enric	Baix
36	**Iteració 7: Desenvolupament del sistema de notificacions**	**15 días**	**mar 09/06/15**	**lun 29/06/15**	**33**		
37	Investigar sobre la integració de notificacions amb Loopback	3 días	mar 09/06/15	jue 11/06/15		Enric	Baix
38	Investigar sobre la integració de notificacions amb Android	3 días	mar 09/06/15	jue 11/06/15		Enric	Baix
39	Desenvolupament notificacions servidor	14 días	mié 10/06/15	lun 29/06/15		Enric	Alt
40	Desenvolupament notificacions Android	14 días	mié 10/06/15	lun 29/06/15		Enric	Mitjà
41	Testing de notificacions	14 días	mié 10/06/15	lun 29/06/15		Enric	Baix
42	Documentació i tancament	4 días	mar 30/06/15	vie 03/07/15	36	Enric	Baix

Figura 9.3: Diagrama de Gantt - segona part

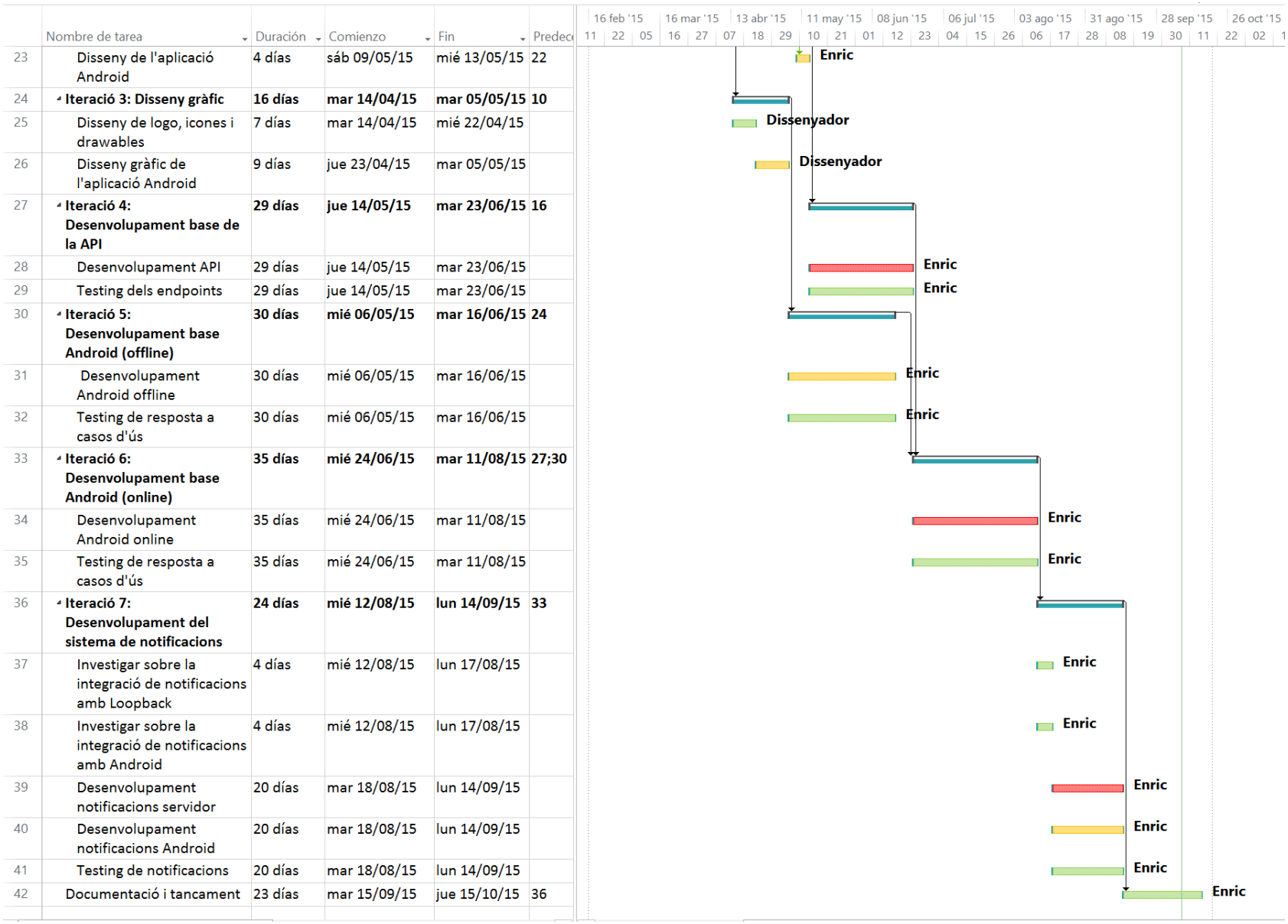
9.5 Canvis des de la planificació inicial

Quant als canvis des de la planificació inicial, hi ha hagut canvis importants. He decidit fer l'entrega del projecte a finals d'octubre i no a finals de juny - principis de juliol com estava previst inicialment. Els motius en són un augment de les hores de dedicació per una part: les tasques relacionades amb el servidor no tenien l'automatització desitjada inicialment pel framework Loopback.io i per tant hi havia una mica més de feina de l'esperada. Les tasques relacionades amb el client Android han tingut un augment d'hores més important, ja que hi havia certs aspectes que no s'havien considerat prou profundament, especialment la gestió de les dades de forma local i la seva sincronització amb el servidor i la gestió del temps de les sessions privades i en grup quan l'usuari no té l'aplicació oberta. A més, la correció de bugs ha prés un bon temps de desenvolupament final.
Per altra part, les pràctiques d'empresa no em permetien dedicar suficient temps al projecte i vaig decidir que allargar-lo i tenir tot l'estiu era una bona solució, dosificant les hores inicialment previstes en més dies. Era una solució que no perjudicava ningú. seguidament es mostra una nova versió del diagrama de Gantt més acurada a la distribució del temps real.



	Nombre de tarea	Duración	Comienzo	Fin	Predece
1	**TFG**	**176 días**	**lun 16/02/15**	**lun 19/10/15**	
2	**GEP**	**26 días**	**lun 16/02/15**	**dom 22/03/15**	
3	Lliurable 1: Abast del projecte	5 días	lun 16/02/15	vie 20/02/15	
4	Lliurable 2: Planificació temporal	4 días	sáb 21/02/15	mié 25/02/15	3
5	Lliurable 3: Gestió econòmica i sostenibilitat	4 días	jue 26/02/15	mar 03/03/15	4
6	Lliurable 4: Presentació preliminar	4 días	mié 04/03/15	dom 08/03/15	5
7	Lliurable 5: Contextualització i bibliografia	6 días	lun 09/03/15	dom 15/03/15	6
8	Lliurable 6: Plec de condicions	6 días	lun 16/03/15	sáb 21/03/15	7
9	Lliurable 7: Document final i presentació oral	1 día	dom 22/03/15	dom 22/03/15	8
10	**Iteració 1: Anàlisi i requisits**	**15 días**	**mar 24/03/15**	**lun 13/04/15**	
11	Definició del client	2 días	**mar 24/03/15**	mié 25/03/15	
12	Especificació de requisits	5 días	jue 26/03/15	mié 01/04/15	11
13	Casos d'ús del sistema	4 días	jue 02/04/15	mar 07/04/15	12
14	Detecció de restriccions	3 días	mié 08/04/15	vie 10/04/15	13
15	Expectatives del projecte	1 día	lun 13/04/15	lun 13/04/15	14
16	**Iteració 2: Disseny del software**	**22 días**	**mar 14/04/15**	mié 13/05/15	**10**
17	Disseny de la BD	2 días	mar 14/04/15	mié 15/04/15	
18	Investigar sobre el disseny d'APIs REST	3 días	jue 16/04/15	lun 20/04/15	
19	Disseny de l'API	3 días	mar 21/04/15	jue 23/04/15	18
20	Investigar sobre el framework Loopback	4 días	vie 24/04/15	mié 29/04/15	
21	Disseny del servidor	3 días	jue 30/04/15	lun 04/05/15	20
22	Mapa navegacional de l'app Android	4 días	mar 05/05/15	vie 08/05/15	
23	Disseny de l'aplicació Android	4 días	sáb 09/05/15	mié 13/05/15	22

Figura 9.4: Diagrama de Gantt Final - primera part



	Nombre de tarea	Duración	Comienzo	Fin	Predec
23	Disseny de l'aplicació Android	4 días	sáb 09/05/15	mié 13/05/15	22
24	**Iteració 3: Disseny gràfic**	**16 días**	**mar 14/04/15**	**mar 05/05/15**	**10**
25	Disseny de logo, icones i drawables	7 días	mar 14/04/15	mié 22/04/15	
26	Disseny gràfic de l'aplicació Android	9 días	jue 23/04/15	mar 05/05/15	
27	**Iteració 4: Desenvolupament base de la API**	**29 días**	**jue 14/05/15**	**mar 23/06/15**	**16**
28	Desenvolupament API	29 días	jue 14/05/15	mar 23/06/15	
29	Testing dels endpoints	29 días	jue 14/05/15	mar 23/06/15	
30	**Iteració 5: Desenvolupament base Android (offline)**	**30 días**	**mié 06/05/15**	**mar 16/06/15**	**24**
31	Desenvolupament Android offline	30 días	mié 06/05/15	mar 16/06/15	
32	Testing de resposta a casos d'ús	30 días	mié 06/05/15	mar 16/06/15	
33	**Iteració 6: Desenvolupament base Android (online)**	**35 días**	**mié 24/06/15**	**mar 11/08/15**	**27;30**
34	Desenvolupament Android online	35 días	mié 24/06/15	mar 11/08/15	
35	Testing de resposta a casos d'ús	35 días	mié 24/06/15	mar 11/08/15	
36	**Iteració 7: Desenvolupament del sistema de notificacions**	**24 días**	**mié 12/08/15**	**lun 14/09/15**	**33**
37	Investigar sobre la integració de notificacions amb Loopback	4 días	mié 12/08/15	lun 17/08/15	
38	Investigar sobre la integració de notificacions amb Android	4 días	mié 12/08/15	lun 17/08/15	
39	Desenvolupament notificacions servidor	20 días	mar 18/08/15	lun 14/09/15	
40	Desenvolupament notificacions Android	20 días	mar 18/08/15	lun 14/09/15	
41	Testing de notificacions	20 días	mar 18/08/15	lun 14/09/15	
42	Documentació i tancament	23 días	mar 15/09/15	jue 15/10/15	36

Figura 9.5: Diagrama de Gantt Final - segona part

Capítol 10

Gestió econòmica

10.1 Identificació i estimació de costos

10.1.1 Costos directes: recursos humans

quí s'inclouen els recursos humans especificats en el diagrama de Gantt de la planificació temporal. Realment, aquestes activitats estaran desenvolupades per dues persones com s'ha comentat prèviament: la persona que fa aquest TFG i el dissenyador, que només realitza la iteració 3. Tot i això, s'ha quantificat el cost de cada activitat d'acord amb el preu de mercat estimat que tindria la persona que hagués de desenvolupar aquesta.

10.1.2 Costos directes materials

Anem ara als costos materials, on diferenciem hardware i software.

Hardware

Tenim dos costos de hardware. L'ordinador portàtil utilitzat per desenvolupar el projecte. Suposarem que té una vida útil de 4 anys i que el 80% del seu ús anirà dedicat al projecte en els 5 mesos que dura. El smartphone Google Nexus 4. Suposarem que té una vida útil de 2 anys i que el 50% del seu ús anirà dedicat al projecte en els 5 mesos que dura.

Software

Durant el desenvolupament del projecte, el cost de software és 0. En un futur post-projecte aquest podria tenir costos associats a contractar llicències o servidors per a ús professional, però queda fora de l'abast del projecte.

10.1.3 Costos indirectes

Connexió a internet

En tot moment del projecte serà necessària una connexió a internet. La connexió ja està contractada actualment (ús domèstic, no exclusiu del projecte) i a més és compartida amb dues persones més (pis compartit). Assumirem que el projecte utilitza el 60% d'internet de la part compartida durant els 5 mesos que dura el projecte.

Taula 10.1: Costos de recursos humans

Tasca	Unitats	Preu de mercat	Cost de mercat
GEP	75h	50€/h	3750€
It1: Anàlisi de requisits	30,5h	50€/h	1525€
It2: Disseny del software	42h	35€/h	1470€
It3: Disseny gràfic	60h	25€/h	1500€
It4: Desenvolupament API	74h	30€/h	2220€
It5: Desenvolupament Android offline	74h	30€/h	2220€
It6: Desenvolupament Android online	74h	30€/h	2220€
It7: Desenvolupament Notificacions	74h	30€/h	2220€
Documentació i tancament	12h	50€/h	600€
TOTAL			**17725€**

Taula 10.2: Costos materials

Producte	Unitats	Preu unitari	Percentatge de dedicació	Cost estimat
Ordenador portàtil Packard Bell Easynote TK85	5 mesos	500€ / 4 anys	80%	42€
Google Nexus 4	5 mesos	250€ / 2 anys	50%	26€
TOTAL				**68€**

Taula 10.3: Tipus de software

	Software liure	Versió gratuïta	Versió de pagament de forma gratuïta
Windows 8.1			Microsoft Developer Network Academic Alliance (Dreamspark)
Genymotion		x	
Sublime Text 3		x	
Android Studio		x	
Loopback		x	
Android SDK		x	
Heroku		x	
Google Drive		x	
Git	x		
GitHub			Student Developer Pack
E-mail		x	
Microsoft Project			Microsoft Developer Network Academic Alliance (Dreamspark)
Postman	x		

Taula 10.4: Costos indirectes

Producte	Unitats	Preu unitari	Percentatge de dedicació	Cost estimat
Internet	5 mesos	40/3€ / mes	60%	40€
Impressions	1000 pàgines	0,05€ / pàgina	100%	50€
TOTAL				**90€**

Taula 10.5: Costos d'imprevistos

Imprevist	Probabilitat	Unitats	Preu de mercat	Cost de mercat
Retard de 10 dies	25%	35h	32,50€/h	1137.5€
Avaria ordinador	5%	1	500€	25€
Avaria smartphone	5%	1	250€	12.5€
TOTAL				**1175€**

Impressions en paper

S'haurà de lliurar la documentació del projecte als membres del tribunal, al director del projecte i una còpia personal. Assumirem una extensió de 200 pàgines.

10.1.4 Contingència

Es reservarà una part del pressupost per a la partida de contingència, que en el nostre cas suposarà el 12% de la suma dels costos directes i indirectes.

10.1.5 Imprevistos

Retard de 10 dies

Tot i que en el Gantt la finalització del projecte és el mateix dia d'exposició d'aquest, el 3 de juliol, ja es va assumir que acabaria abans i es va donar cert marge a les diferents tasques. Tot i això, podria haver-hi un retard en el desenvolupament del projecte de 10 dies. El preu per hora es comptarà a un preu intermig de 32,50€. La probabilitat de què passi és del 25%.

Avaria de hardware

És possible que durant el desenvolupament del projecte hi hagi una avaria en el portàtil o el smartphone i faci falta comprar nous aparells. Assignem a aquest imprevist una probabilitat del 5%.

10.1.6 Pressupost

Taula 10.6: Costos totals

Concepte	Cost de mercat
Costos directes recursos humans	17725€
Costos directes materials	68€
Costos indirectes	90€
Contingència	2145.96€
Imprevistos	1175€
TOTAL	**21203.96€**

10.2 Control de gestió

Per al control de software i hardware no podem fer res més que anotar si s'ha complert algun dels imprevists. Per al control dels recursos humans, es mantindrà un registre d'hores treballades en cada tasca per tal de veure la desviació temporal que s'ha sofert al final de cada iteració i des d'aquí calcular la desviació en el pressupost. D'aquesta manera es podrà corregir el rumb del projecte a partir de la següent iteració. Quant als costs indirectes, es podrà calcular la desviació en el nombre de folis impresos, comparant els prevists amb els reals. Al final del projecte s'agruparan aquests costos en una taula i es comprovarà si aquests han estat coberts en els imprevistos o si han estat per alguna altra causa. També es mirarà si la partida de contingència ho cobreix.

Capítol 11

Sostenibilitat i compromís social

11.1 Sostenibilitat econòmica

S'ha fet una avaluació dels costos, tant dels recursos humans com dels materials i s'ha calculat possibles imprevistos. La metodologia SCRUM ens permetrà adaptar aquests costos. Si el projecte acaba sent competitiu i surt al mercat, el cost del projecte serà viable si es pot crear un conjunt d'usuaris que utilitzin l'aplicació i permeti que aquesta es viralitzi (seria necessària una futura inversió en màrqueting), però arribats a aquest punt seria molt fàcil escalar-la perquè funcionés amb grans quantitats de gent sense un sobrecost important, aconseguint llavors un retorn de la inversió molt alt. El temps per a realitzar el projecte és bastant acotat i difícil de millorar, tenint en compte que s'espera un mínim de qualitat i la opció que aquest sigui escalable en un futur. La distribució de cada tasca és l'adequat i ja s'ha inclòs tecnologies que agilitzen el desenvolupament (per exemple Loopback, que és una solució més concreta per a desenvolupar APIs però que estalvia reinventar la roda). Actualment no es preveu col · laboració amb cap altre projecte.

11.2 Sostenibilitat social

L'objectiu del projecte és millorar la productivitat en grup, especialment en el sector empresarial. Creiem que existeix un lloc per a la nostra aplicació, que a un preu molt baix permetria una millora en la productivitat i contribuiria a la felicitat de treballadors, ja que podrien descansar tots junts i alhora estar motivats quan toca treballar. A més, la necessitat del servei és palpable, ja que des del sector empresarial es busca sempre la manera de millorar la productivitat, i la nostra solució no compromet la qualitat de vida del treballador, al contrari. En el cas d'estudiants i altres grups de treball, ajudaria tenir una eina que permetés realitzar la feina abans i amb més ganes. És difícil pensar que algun col · lectiu que sigui perjudicat pel nostre sistema. En tot cas, podria ser que la millora de la productivitat generés algun acomiadament perquè ja no és necessari tant blat. Més indirectament, aplicacions com Facebook podrien tenir menys tràfic perquè no es perd tant el temps, però és bastant inversemblant.

11.3 Sostenibilitat ambiental

Els recursos de hardware hauran consumit recursos en la seva construcció i gasten electricitat quan s'utilitzen. Els de software no directament, però al utilitzar-los és necessari fer-ho a partir d'un dels recursos de hardware. L'única contaminació que generen és la indirecta per l'obtenció d'electricitat. Si el projecte s'hagués fet

Taula 11.1: Matriu de sostenibilitat

Sostenibilitat	Econòmica	Social	Ambiental	TOTAL
Planificació	Viabilitat econòmica	Millora en la qualitat de vida	Anàlisi de recursos	
Valoració	8	8	6	22

sense TFG, s'hagueren estalviat els fulls necessaris per a imprimir-lo. Tot i això, la resta de documentació ja s'està utilitzant de forma digital. En el projecte es reaprofiten tecnologies i eines, que eviten haver-les de desenvolupar (amb el cost energètic que això comportaria). El projecte no augmenta ni disminueix la petjada ecològica, no es necessiten matèries primeres i no es preveu el desmantellament de cap recurs al final del projecte.

Capítol 12

Apèndix: Manual d'usuari

El manual d'usuari mostrarà les diferents pantalles i accions disponibles de l'aplicació Android de TikTak Team.

Figura 12.1: Pestanya sol

Aquí es pot veure la pestanya de sessions privades, que és la pantalla inicial. Amb les diferents pestanyes es pot canviar de pantalla. El botó de baix a la dreta permet afegir noves sessions privades. Si es prem el botó d'iniciar i hi ha una sessió pendent, el comptador s'inicia.

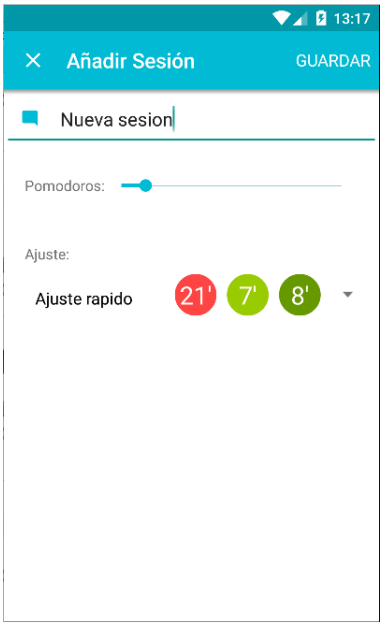


Figura 12.2: Afegir sessió privada

Aquesta és la pantalla per afegir sessions privades. S'ha de seleccionar el nom, el número de pomodoros i la configuració de temps desitjada.

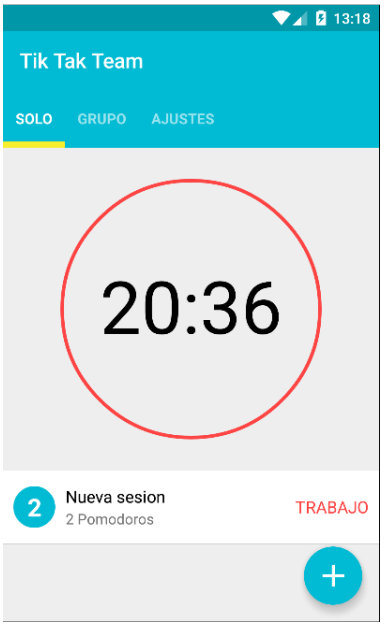


Figura 12.3: Pestanya sol, amb una sessió en procés

Tornem a ser a la pestanya de sessions privades, on hem premut ÏNICIOï la sessió afegida està en procés.

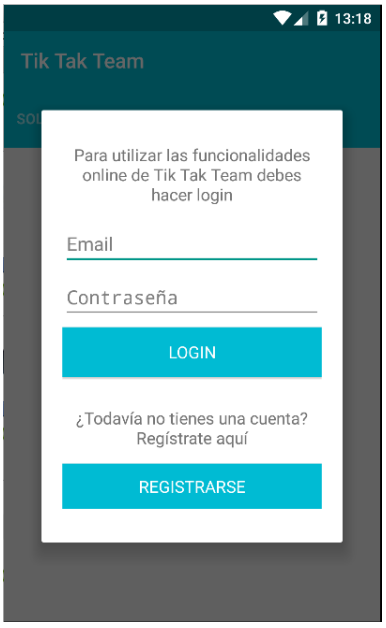


Figura 12.4: Popup de login

Al canviar a la pestanya de grups i no estar identificat, es demana a l'usuari que faci login.

Figura 12.5: Pantalla de registre

Si l'usuari no està registrat, pot crear-se un compte.

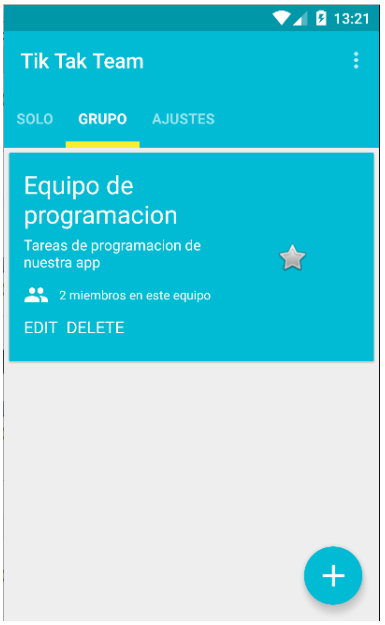


Figura 12.6: Pestanya de grups

Un cop ha fet login, es mostren els seus grups.

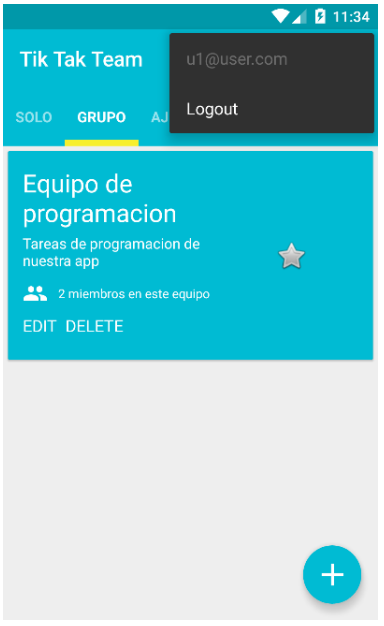


Figura 12.7: Logout

Si es prem als 3 botons de l'extrem superior esquerre, es veu informació del login i es té la opció de fer logout.

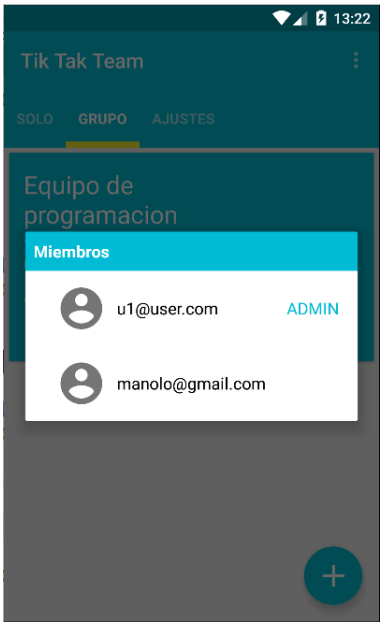


Figura 12.8: Popup de membres del grup

Si es prem el nombre de membres del grup, es mostra un popup on s'informa de qui són.

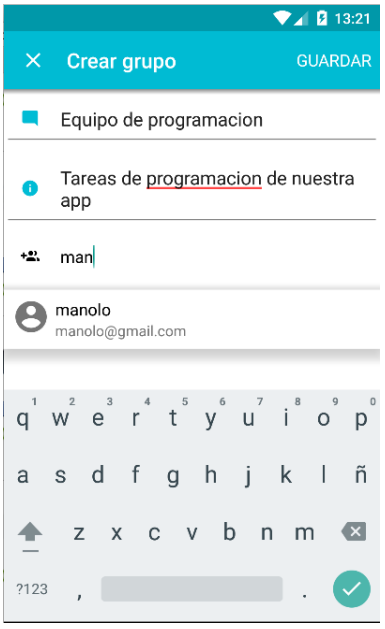


Figura 12.9: Pantalla per afegir un grup

En la pantalla per afegir un grup s'ha d'introduïr el nom, una descripció i els membres d'aquest grup.

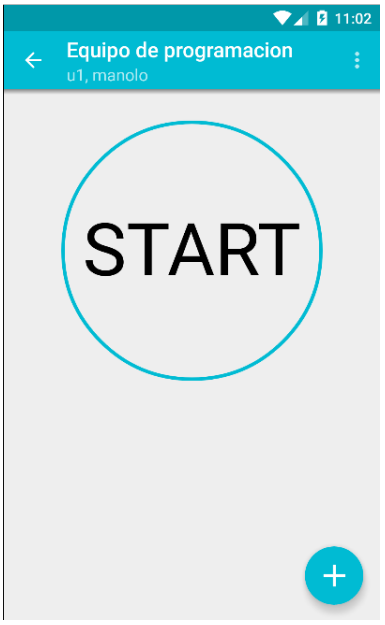


Figura 12.10: Pantalla d'un grup

Havent premut a un grup, entrem a la pantalla d'aquest grup.

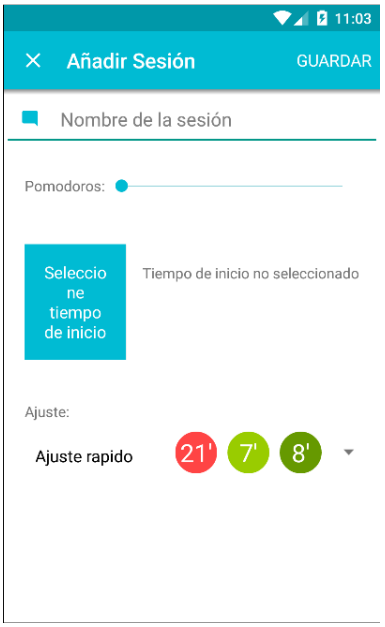


Figura 12.11: Pantalla per afegir una sessió de grup

A la pantalla per afegir un grup s'hi demana el nom, el nombre de pomodoros, la data d'inici i la configuració de temps que es vulgui.

Figura 12.12: Popup per seleccionar el dia

Per seleccionar el moment d'inici, primerament se'ns demana el dia.

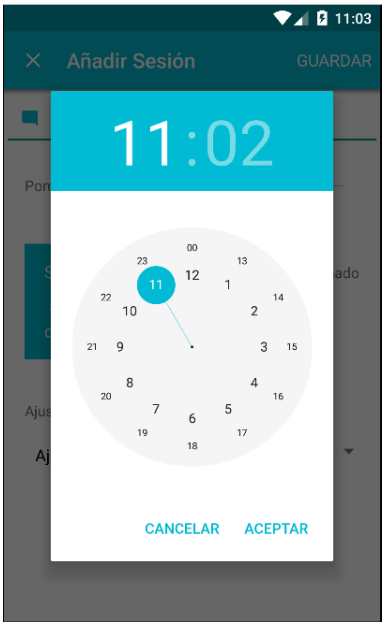


Figura 12.13: Popup per seleccionar l'hora

Posteriorment, l'hora.

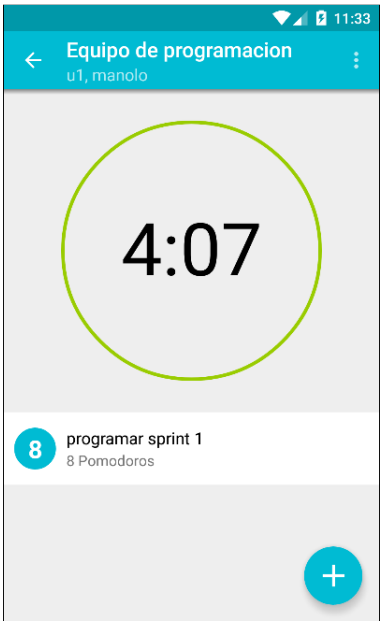


Figura 12.14: Pantalla d'un grup amb una sessió afegida en l'estat de descans

En aquesta pantalla tornem a ser a la pantalla de grup, però hi ha una sessió programada que ja s'ha iniciat.

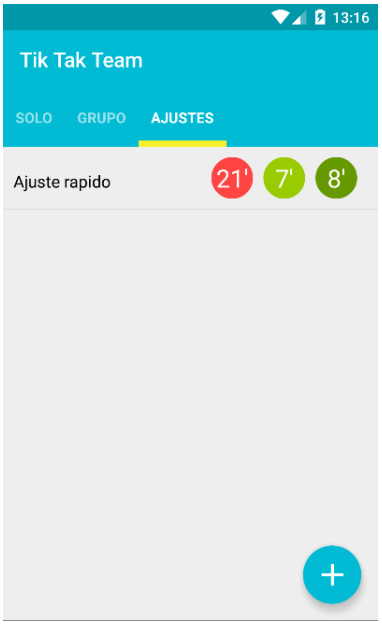


Figura 12.15: Pestanya de configuracions

La pestanya de configuracions.

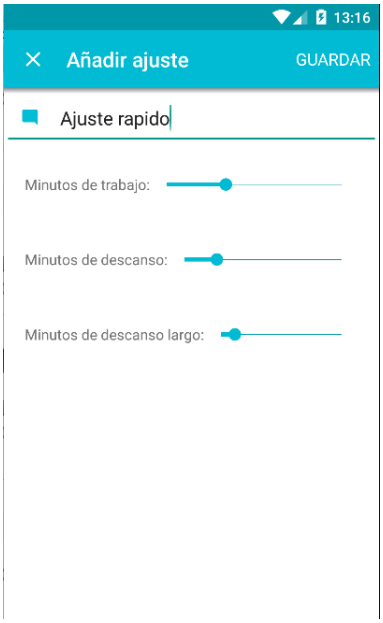


Figura 12.16: Pantalla per afegir una configuració

La pantalla d'afegir configuració demana un nom, els minuts de treball, els minuts de descans i els minuts de descans llarg.

Capítol 13

Apèndix: Imatge corporativa

Finalment, afegeixo un apèndix relacionat amb la imatge corporativa que ha treballat el dissenyador, Albert Mitjana.

Figura 13.1: Inici i estructura

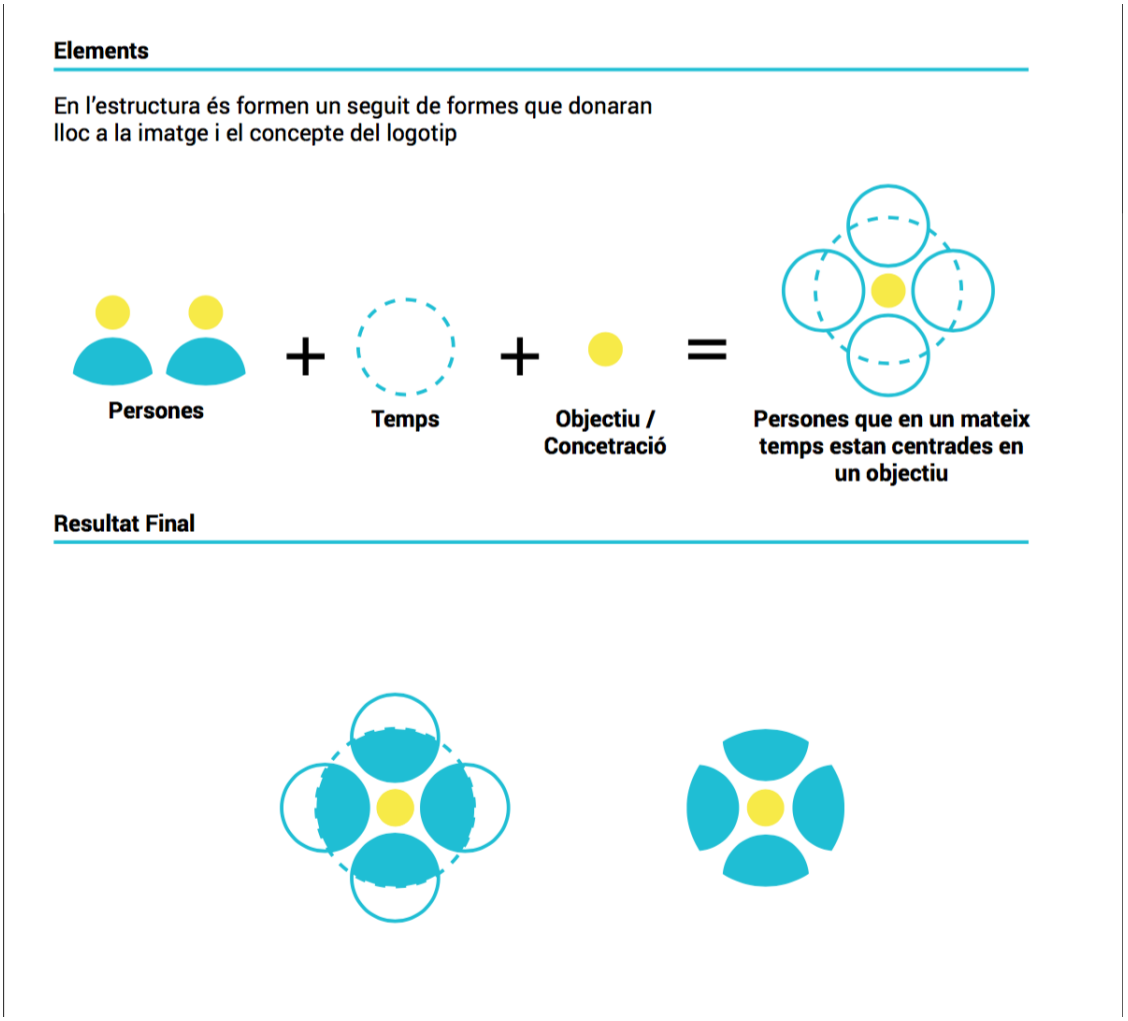


Figura 13.2: Elements i resultat final

Figura 13.3: Logotip

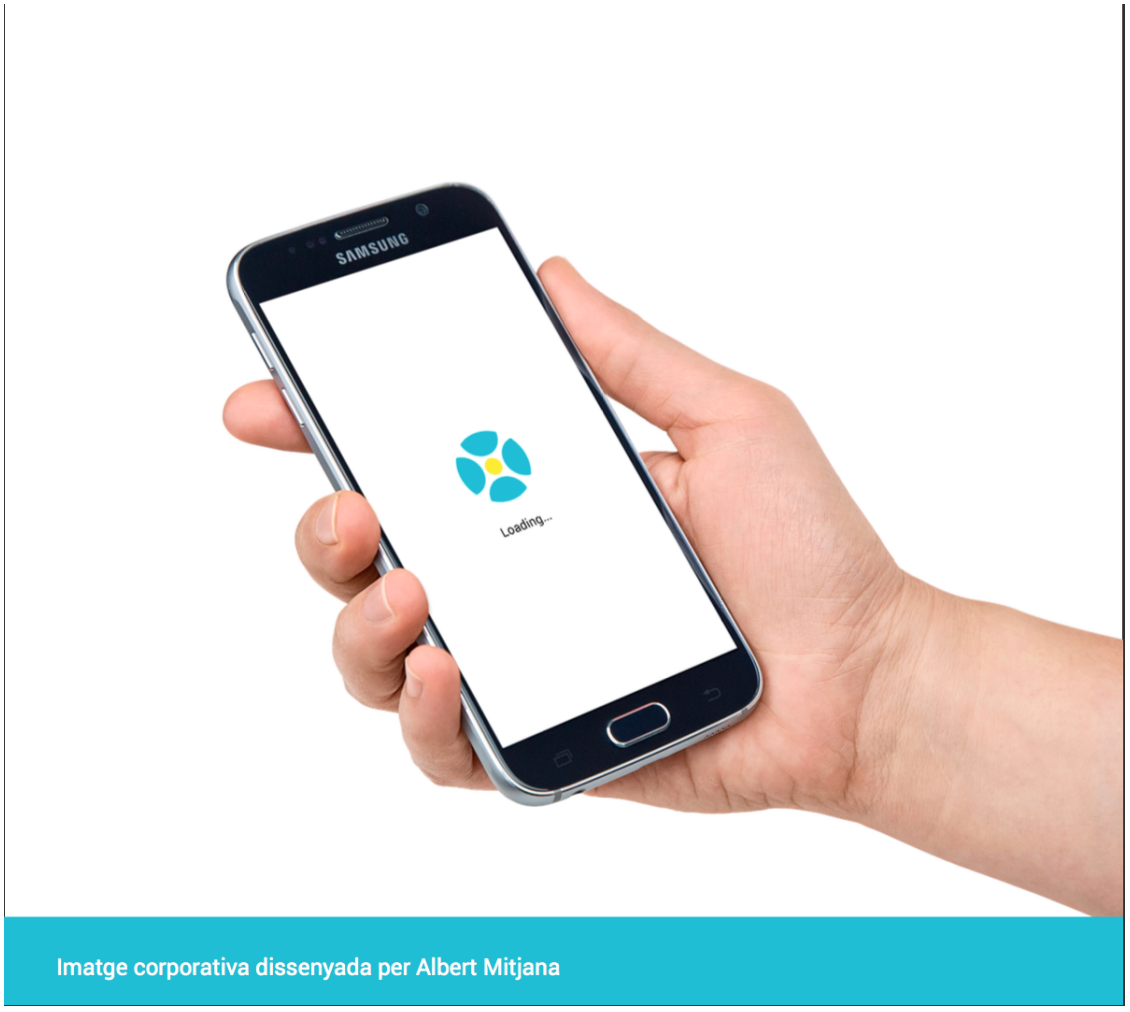


Figura 13.4: Exemple de mostra

Bibliografia

[1] Dhananjay Parkhe. Time management - the pomodoro technique. https://www.linkedin.com/pulse/20140719232001-671295-time-management-techniques-no-one-taught-me-b-schools-pomodoro-technique.

[2] Investopedia. Productivity definition. http://www.investopedia.com/terms/p/productivity.asp.

[3] Free Merriam-Webster Dictionary. Teamwork definition. http://www.merriam-webster.com/dictionary/teamwork.

[4] E. Salas. Is there a "big five" in teamwork? http://doi.org/10.1177/1046496405277134.

[5] Wikipedia. Time management. http://en.wikipedia.org/wiki/Time_management.

[6] F. Cirillo. The pomodoro technique. in agile processes in software engineering. (p. 46) Retrieved from baomee.info.

[7] Wikipedia. Client. http://es.wikipedia.org/wiki/Cliente%28inform%C3%A1tica%29.

[8] Wikipedia. Servidor web. https://es.wikipedia.org/wiki/Servidor_web.

[9] Wikipedia. Framework. http://es.wikipedia.org/wiki/Framework.

[10] Wikipedia. Api. https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones.

[11] Wikipedia. Json. https://es.wikipedia.org/wiki/JSON.

[12] Wikipedia. Representational state transfer. https://es.wikipedia.org/wiki/Representational_State_Transfer.

[13] Wikipedia. Object relational mapping. https://en.wikipedia.org/wiki/Object-relational_mapping.

[14] Wikipedia. Software development kit. https://en.wikipedia.org/wiki/Software_development_kit.

[15] W3Techs. Usage statistics and market share of side programming languages for websites. http://w3techs.com/technologies/overview/programminglanguage/all.

[16] Why StrongLoop? Usage statistics and market share of side programming languages for websites. https://strongloop.com/.

[17] Wikipedia. Análisis abc. https://es.wikipedia.org/wiki/An%C3%A1lisis_ABC.

[18] Wikipedia. Principio de pareto. https://es.wikipedia.org/wiki/Principio_de_Pareto.

[19] Gabriella Campbell. El método eisenhower. cómo elegir lo que es realmente importante. http://www.gabriellaliteraria.com/el-metodo-eisenhower-como-elegir-lo-que-es-realmente-importante/.

[20] Wikipedia. Getting things done. https://es.wikipedia.org/wiki/Getting_Things_Done.

[21] Ali Schiller. The do, doing, done board. http://alischiller.com/the-do-doing-done-board/.

[22] Todoist. Todoist. https://todoist.com/.

[23] Trello. What is trello? http://help.trello.com/article/708-what-is-trello.

[24] KanbanFlow. Kanbanflow. https://kanbanflow.com/.