Aplicació per a la gestió dels comptes bancaris i pressupostos dels usuaris

Marc Sala Manzano

Resum– En aquest article, s'exposa el desenvolupament dut a terme per tal de realitzar una aplicació per dispositius Android que té com a objectiu facilitar als usuaris el seguiment de les seves despeses i ingressos juntament amb la capacitat de generar pressupostos. Per tant, a continuació es troba una introducció del tema per posar en context la necessitat d'aquesta aplicació juntament amb els objectius ben definits, l'estat de l'art, és a dir, aplicacions existents que compleixen amb alguns dels objectius anteriors, la metodologia que s'ha seguit per tal de realitzar el desenvolupament, la planificació seguida, la qual comença amb un anàlisi d'on s'extreuen els requisits i dissenys necessaris, continua amb la planificació del desenvolupament i finalitza amb la posada en marxa. Per últim es troben els resultats obtinguts del desenvolupament finalitzant amb unes conclusions del treball realitzat i la bibliografia utilitzada durant el treball.

Paraules clau– Desenvolupament Software, Aplicació Android, API, Pressupostos, Despeses i Ingressos, Endpoints, Arquitectura del Software

Abstract– This article explains the development carried out in order to create an application for Android devices that aims to make it easier for users to track their expenses and incomes along with the ability to generate budgets. Therefore, below we can find an introduction to the topic, which puts in context the needing of this application along with the goals, the state of the art, i.e. existing applications that meet some of the above objectives, the methodology that has been followed in order to carry out the development, the planning followed, which begins with an analysis of the requirements and the designs developed, continues with the development planning and ends with the implementation. Finally, we can find the results obtained from the development, ending with conclusions from the work carried out and the bibliography used during the work.

Keywords– Software development, Android application, API, Budgets, Expenses and Incomes, Endpoints, Software Architecture

1 Introducció - Context del treball

Avui en dia, en una societat on tot costa diners i on, degut al creixement del e-commerce[1], tenim l'habilitat de comprar qualsevol producte des de qualsevol dispositiu i en qualsevol moment, a vegades només pel simple fet de sentir-nos millor[2], és molt fàcil perdre el control sobre les despeses que generem, sobretot si no tenim alguna eina que ens permeti fer un seguiment d'aquestes. Tenir una bona gestió financera ens permet estalviar i arribar a final de mes amb un coixí econòmic que ens permet fer front a imprevistos d'una forma més relaxada. Una de les tècniques per tenir una millora financera és la pressupostació[3], és per això que han sorgit una gran quantitat d'aplicacions que ens permeten tenir aquest seguiment necessari de les despeses i que ens permeten realitzar pressupostos, però moltes d'aquestes ens aporten una funcionalitat limitada les quals s'han de pagar per tal desbloquejar totes les funcionalitats, fet que es contradiu amb l'objectiu d'aquestes aplicacions, el qual és controlar les despeses dels usuaris.

Per tant, els objectius establerts en el projecte són els següents:

- Donar una eina als usuaris que els hi permeti fer un seguiment de les seves despeses i ingressos separades per categories o temps, millorant així la gestió financera d'aquests.

• E-mail de contacte: salamarc6@gmail.com
• Mención realitzada: Enginyeria del Software
• Treball tutoritzat per: Helena Bolta Torrell (Department of Computer Science)
• Curs 2021/22

- Mostrar a l'usuari la informació de les seves despeses i ingressos a través de gràfiques perquè pugui veure d'una forma molt visual i clara el moviment de diners en les diferents categories.
- Oferir a l'usuari la capacitat de generar pressupostos basats en categories amb l'objectiu que l'usuari pugui accedir ràpidament a la quantitat de despesa que li queda per a cada categoria i que se li pugui avisar quan s'estigui arribant a un límit.
- Permetre que l'usuari pugui gestionar diferents comptes bancaris de forma independent i poder accedir a la informació de cada un de forma ràpida.
- Donar la possibilitat a l'usuari de compartir un compte amb un altre usuari, habilitant així el seguiment de les despeses entre diferents usuaris, el que facilita la gestió dels comptes compartits.
- Permetre a l'usuari la capacitat de configurar noves categories que s'adaptin millor al seu estil de vida, oferint així un seguiment més personalitzat.

2 Estat de l'art

Actualment existeixen aplicacions que compleixen certs objectius dels esmentats anteriorment. Les principals aplicacions que ens podem trobar són les de les pròpies entitats bancàries, les quals ens permeten tenir un seguiment de les despeses i dels ingressos per als comptes que estiguin registrats. Aquestes, però, solen tenir a l'hora de compartir comptes bancaris entre persones si aquests pertanyen a diferents entitats bancàries o, per altra banda, no et permeten configurar les categories dels moviments, el que comporta una insatisfacció dels usuaris.

També ens podem trobar amb aplicacions externes a les entitats bancàries, com Toshl Finanzas[4] o Wallet, d'on s'ha extret gran part de les funcionalitats del projecte degut a que compleixen amb tots els objectius anteriors, ja que ens proporcionen un seguiment dels comptes juntament amb els moviments que s'han fet en aquests, a part de poder crear pressupostos. El problema d'aquestes aplicacions és que venen limitades en una versió gratuïta, el que implica que si es vol portar una bona gestió financera, s'ha de realitzar un pagamanent.

Per últim, també existeixen eines que no estan pensades per fer una gestió financera, però que es poden utilitzar, com fulls de càlcul, però a causa de la complexitat que comporta tenir un seguiment amb aquest tipus d'eines, és millor no utilitzar-les.

3 Metodologia

En aquesta secció s'exposa la metodologia utilitzada per tal de realitzar totes les tasques, i per tant, la forma en la que s'ha treballat per tal desenvolupar l'aplicació.

Per tal de dur a terme el projecte, s'ha començat amb la realització d'una planificació inicial mitjançant un diagrama de Gantt on s'han repartit totes les tasques ha realitzar durant el desenvolupament del projecte. Aquesta planificació inicial serveix com a línia base per les revisions setmanals per a veure si el projecte està avançant de forma correcta o no.

Pel que fa al seguiment del dia a dia, s'ha utilitzat l'eina Trello per implementar la metodologia àgil Kanban[5], la qual es basa a tenir un taulell amb diferents columnes (To Do, Doing, Done) per on van passant les tasques que s'estan duent a terme, obtenint de forma molt visual l'estat actual del projecte i les tasques en les quals s'estan treballant, a part de limitar la càrrega de treball, que en aquest cas esdevindria a la quantitat de tasques diferents que s'estan realitzant a la vegada. Per tal d'adaptar Kanban en el desenvolupament del projecte, s'ha decidit utilitzar quatre columnes:

- To Do: En aquesta columna es posen totes les tasques que s'han de realitzar durant el llarg de la setmana.
- Doing: En aquesta columna es posen aquelles tasques que s'estan realitzant actualment i, per tant, per tal d'evitar tenir una saturació de tasques diferents no hi poden haver més de tres a la vegada.
- Done (Falta revisió): En aquesta columna es posen aquelles tasques que s'han acabat de realitzar però que necessiten una revisió per concloure que s'han acabat correctament, principalment aniran tasques de documentació. Aquestes tasques poden tornar a la columna de Doing si és necessari.
- Done: En aquesta columna es posen totes les tasques que ja s'han acabat i han estat revisades i, per tant, les tasques que acabin aquí no es podran moure.

Per altra banda, a principis de cada setmana s'ha realitzat una revisió del taulell Kanban per tal de veure si s'està seguint la planificació inicial que es va fer o veure si queden tasques per fer i en el cas que sigui necessari fer una petita replanificació per tal de tornar a seguir dins dels temps marcats en la línia base.

També s'han dut a terme reunions amb el tutor del TFG cada dues setmanes per tal d'obtenir un feedback sobre les funcionalitats generades i si s'escau, aplicar canvis en les diferents pantalles per tal de millorar l'experiència d'usuari.

Per últim, per tal de testejar l'aplicació i assegurar que aquesta funciona correctament, s'han desenvolupat un seguit de casos de test, tant per la part del backend, on es comproven diferents crides cap als endpoints de l'aplicació les quals han de generar diferents resultats, com per la part del frontend, on es comproven diferents inputs de l'usuari sobre els formularis per tal verificar que l'aplicació es comporta com és degut. Degut a la gran varietat de llibreries que es troben en el projecte, s'ha optat per executar els tests de forma manual, ja que l'automatització d'aquests comportaria una càrrega de feina molt elevada. Es poden trobar exemples d'aquests casos de test en la secció A.1 de l'apèndix.

4 Planificació i treball realitzat

En aquesta secció s'exposa la planificació i el treball realitzat en tres apartats diferents:

4.1 Anàlisi i disseny

En aquest apartat s'explica les tasques inicials portades a terme abans de començar amb el desenvolupament, les quals es basen en la realització d'un anàlisi sobre les eines a utilitzar i un disseny dels requisits necessaris per complir amb els objectius.

Respecte l'anàlisi d'eines s'ha optat per realitzar una aplicació mitjançant el framework React Native[6], basat en Javascript, el qual permet desenvolupar aplicacions per Android i iOS. Per la part del Back-End s'ha optat per realitzar una API mitjançant Node.js i Express.js[7], també basats en Javascript, els quals permeten muntar un servidor operatiu que escolta i respon a les peticions HTTP. També s'ha decidit utilitzar MySQL com a gestor de base de dades, i per tant, utilitzar una BD relacional. Per altre banda, també s'utilitzen eines com Firebase[8] per permetre la sincronització entre Back-End i Frontend i l'API de Google per permetre l'accés a l'aplicació utilitzant directament el compte de Google. Aquesta combinació d'eines la podem representar amb el diagrama de la figura 1.

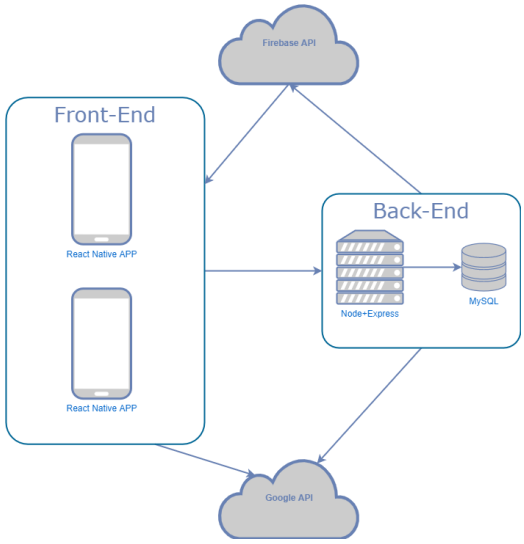


Fig. 1: Diagrama d'estructura

Respecte l'anàlisi i disseny dels requisits, s'ha generat un llistat a partir dels objectius establerts i un wireframe de les pantalles principals com a suport per tal de definir-los correctament i amb més detall. Aquests s'han agrupat en 5 seccions diferents les quals són:

- Funcionalitats d'accés: En aquest grup es troben aquelles funcionalitats encarregades de donar accés a les funcionalitats dels altres grups, és a dir, són aquelles encarregades de registrar i connectar als usuaris apart de permetre realitzar la desconnexió:
  - El sistema ha de permetre a l'usuari la capacitat de registrar-se al sistema a partir de nom/mail/password.
  - El sistema ha de permetre a l'usuari fer login a través de mail/password per tal d'accedir a les funcionalitats.
  - El sistema ha de permetre a l'usuari fer login a través Google per tal d'accedir a les funcionalitats.
  - Si es fa login a través de google per primera vegada, el sistema ha de registrar a l'usuari.
  - El sistema ha de permetre a l'usuari la opció de desconectar-se en tot moment.
  - El sistema ha de permetre a l'usuari la opció de eliminar el seu perfil.
  - El sistema ha de permetre a l'usuari la capacitat de canviar la contrasenya en tot moment, enviant un mail amb una clau autogenerada i un link on posar la nova contrasenya i la clau.
- Funcionalitats de gestió de comptes bancaris: En aquest grup es troben aquelles funcionalitats encarregades de gestionar els comptes bancaris dels usuaris.
  - Tan aviat com es registri un usuari, se li ha de crear un compte bancari per defecte amb import 0€.
  - El sistema ha de permetre a l'usuari crear nous comptes, on podrà afegir una quantitat inicial de diners.
  - El sistema ha de permetre a l'usuari la capacitat de eliminar un compte, eliminant així tota la informació associada a aquest.
  - El sistema ha de permetre a l'usuari la capacitat d'afegir un altre usuari al compte bancari. (Comptes bancaris compartits entre diferents usuaris). Es comparteix indicant el mail.
  - El sistema ha de mostrar un llistat amb tots els comptes d'un usuari, juntament amb el seu balanç.
  - El sistema ha de permetre a l'usuari editar el nom del compte i la compartició d'aquesta.
- Funcionalitats de gestió de moviments: En aquest grup es troben aquelles funcionalitats encarregades de gestionar les despeses i ingressos dels usuaris:
  - El sistema ha de permetre a l'usuari la capacitat de crear una nova despesa/ingrés, on s'indica: quantitat, categoria, etiqueta (opcional), compte associada, data i descripció (opcional).
  - El sistema ha de permetre eliminar una despesa/ingrés creada, afegint/eliminant els diners en el compte.
  - El sistema ha de mostrar una gràfica de les despeses/ingressos, separant-les per categories. La gràfica es mostra sobre les despeses/ingressos d'un compte o de tots els comptes, el qual es pot triar a través d'un desplegable.
  - El sistema ha de mostrar el llistat de les despeses/ingressos en un mes i un compte on per cada una s'indica quantitat, categoria i etiqueta. La llista està separada per temps. (Avui, ahir, aquesta semana, aquest mes). També es pot mostrar les despeses separades per categories en comptes de temps.
  - Si es selecciona una despesa/ingrés de la llista, el sistema ha de mostrar tota la informació d'aquesta, juntament amb les opcions d'editar i eliminar.

  - El sistema ha de permetre a l'usuari la capacitat d'editar la informació de la despesa/ingrés. Si s'edita l'import de la despesa/ingrés, s'ha d'actualitzar la quantitat del compte amb la diferència.

- Funcionalitats de gestió de pressupostos: En aquest grup es troben aquelles funcionalitats encarregades de gestionar els pressupostos dels usuaris a la vegada de notificar-los en el moment en que s'estigui arribant a un límit:

  - El sistema ha de permetre a l'usuari la capacitat de crear un pressupost, indicant a quin compte està associat, la quantitat del pressupost, a les categories que afecta, el nom del pressupost, si es repetitiu (diari, semanal, mensual, anual) i en el cas que sigui repetitu, si es vol transferir la quantitat restant al següent periode. Apart també s'ha d'indicar la data d'inici i en el cas que no sigui repetitiu, la data final.
  - Si un pressupost és repetitu, quan finalitzi un periode, el sistema haurà de crear un nou pressupost amb la informació corresponent.
  - Si un pressupost es repetitu i està marcat amb el traspàs de diners al finalitzar, al generar el nou pressupost haurà de incrementar la quantitat base d'aquest amb el que ha sobrat en el pressupost finalitzat.
  - Si un pressupost arriba al 80% de la quantitat del pressupost, haurà de notificar a l'usuari.
  - El sistema ha de permetre a l'usuari la capacitat de eliminar/modificar un pressupost.
  - El sistema ha de mostrar la llista de pressupostos actius on s'indica per cada: dates, el pressupost que queda, el compte que esta associat, les categories a les que pertany.

- Funcionalitats de gestió de categories: En aquest grup es troben aquelles funcionalitats encarregades de gestionar les categories i subcategories que poden generar els usuaris:

  - El sistema ha de permetre al usuari la capacitat de crear noves categories i etiquetes (aquestes ultimes han d'estar associades a una categoria).
  - El sistema ha de permetre al usuari la capacitat de eliminar o modificar les categories i etiquetes-. Si s'elimina una categoria, s'eliminen totes les seves etiquetes.
  - El sistema ha de mostrar totes les categories amb les seves etiquetes.

També s'ha realitzat un disseny de la base de dades el qual, gràcies a aquest, s'han considerat sis entitats principals que tindran molt pes en l'estructura: usuaris, comptes bancaris, despeses, ingressos, categories i pressupostos. Aquestes entitats ens dictaminen gran part de l'estructura del codi que s'explica més endavant. Es pot trobar el disseny de la base de dades en la secció A.2 de l'apèndix.

Per últim, s'ha desenvolupat un disseny inicial de les pantalles de l'aplicació en format wireframe, el qual ens serveix com una guia a l'hora de desenvolupar les pantalles, facilitant la programació d'aquestes al tenir una estructura definida des del principi. Es poden trobar algun dels dissenys en la secció A.3 de l'apèndix.

4.2 Desenvolupament

En aquest apartat s'expliquen les tasques realitzades per tal d'elaborar el codi de l'aplicació (tant de la part de frontend com la part de backend). Tal com s'ha explicat anteriorment, el desenvolupament està dividit en diferents mòduls, els quals depenen dels requeriments que compleixen.

Abans del desenvolupament dels diferents mòduls, s'ha realitzat una tasca de preparació d'entorn, la qual consisteix en la creació de la base de dades, en la instal·lació de les llibreries principals necessàries i en la generació de l'esquelet tant de frontend com de backend. Amb aquesta tasca s'ha obtingut la composició de carpetes de la figura 2, la qual ens marca l'arquitectura del software, és a dir, ens dictamina a on ha d'anar cada part del codi depenent de la funcionalitat que compleixi.

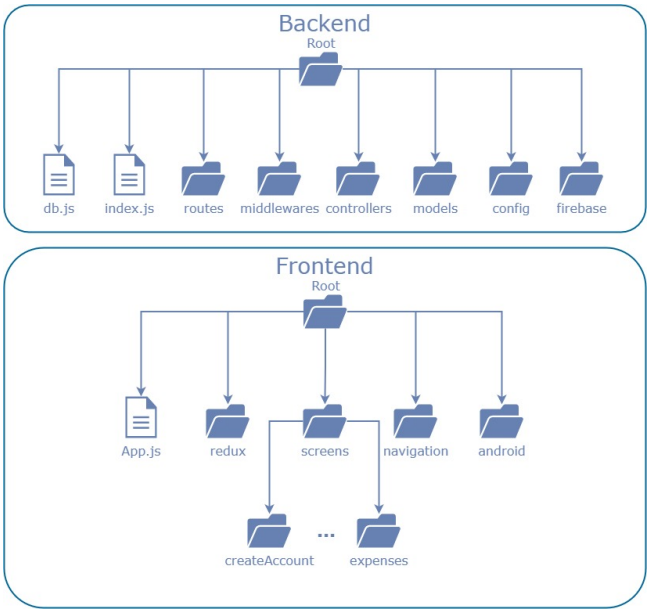


Fig. 2: Estructura del codi

Pel que fa a l'estructura del backend, tenim el directori root on es troba el fitxer db.js el qual s'encarrega de proporcionar connexions a la base de dades i el fitxer index.js el qual és el punt d'inici del servidor encarregat tant d'iniciar el servidor com d'iniciar el servei de Firebase utilitzat per l'enviament de notificacions per tal de sincronitzar el backend amb l'aplicació. Per altra banda, en el directori routes es troben els diferents fitxers (un per cada entitat, com ara comptes, despeses, etc...) que s'encarreguen de declarar els endpoints per tal d'accedir a les funcionalitats del backend des de fora. En aquesta declaració es determina quins controladors i middlewares s'utilitzen. En el directori middlewares es troben les funcionalitats que es criden en la majoria d'endpoints i que han d'actuar abans que els controladors, un exemple de middleware és la comprovació del token en la petició per veure si l'usuari que està fent la crida està autoritzat i en el cas que ho estigui en la identificació d'aquest. En el directori controllers, es troben els

diferents fitxers (un per cada objecte) que contenen diverses funcionalitats. Cada funcionalitat s'encarrega d'extreure la informació de la petició, comprovant que aquesta sigui correcta i juntament amb els models s'encarreguen de realitzar la funcionalitat pertinent. Per últim, el controlador determina el resultat de la petició, és a dir, si es considera que s'ha completat tot correctament resol la petició amb un codi de resposta satisfactori, enviant també la informació necessària en el cos de la resposta i si ha sorgit un error o la informació de la petició és incorrecta, resol la petició amb un codi de resposta insatisfactori. Un exemple de controlador és l'encarregat de gestionar els comptes, el qual conté diferents funcionalitats com ara crear, modificar, etc... relacionat amb els comptes bancaris dels usuaris. Respecte al directori models, és on es troben les definicions dels diferents objectes, és a dir, dels comptes, usuaris, pressupostos, etc.... Per altra banda, en els models també es troben les funcions encarregades de generar les crides a la base de dades mitjançant la connexió otorgada pel fitxer db.js.

Respecte a l'estructura del frontend, tenim el directori root on es troba el fitxer App.js el qual es el punt d'entrada a l'aplicació i l'encarregat de definir el tema de l'aplicació i de declarar els proveidors interns per tal d'obtenir la navegació i l'emmagatzematge intern del dispositiu per mantenir l'estat i no perdre informació durant l'ús de l'aplicació. També es troba el directori redux, en el qual es troba el codi encarregat de declarar els diferents estats de l'aplicació mitjançant la llibreria Redux[9], és a dir, s'encarrega d'emmagatzemar certa informació que ha d'estar de forma global en tota l'aplicació, com el token d'autentificació de l'usuari, i també es declara la forma d'accés a aquest emmagatzematge. En el directori screens és on es troben les diferents pantalles que componen l'aplicació i, per tant, és on es troba el codi que serveix com a punt d'entrada de l'usuari, és a dir, amb el que interactua directament. Per a cada pantalla es genera un directori on es troba, per una banda, el codi encarregat de generar la pantalla juntament amb el codi encarregat d'actuar davant dels inputs de l'usuari i, per altra banda, el codi que defineix el disseny d'aquesta. En el directori navigation es troba tot el codi referent a la navegació entre pantalles, el qual, mitjançant la informació de Redux, controla si un usuari té accés a certes pantalles o no. Per últim, es troba la carpeta android, en la qual es troba els fitxers que permeten muntar l'aplicació per a sistemes Android.

Respecte a la planificació i a la feina desenvolupada, un cop finalitzada l'estructura del sistema s'ha començat generant les rutes, controladors i models encarregats de proporcionar el registre i el login de l'usuari i retornar el token d'autenticació, on s'han obtingut un total de dos endpoints. S'ha utilitzat la llibreria jwt[10] per tal de generar el token, a partir de l'email de l'usuari, amb un dia de validesa, implicant que els usuaris s'hauran de loguejar cada dia. Per altra banda, s'ha utilitzat la llibreria bcrypt[11] per tal d'encriptar les contrasenyes a l'hora de guardar-les a la base de dades. També s'han generat dos triggers de MySQL que s'encarreguen de crear el compte bàsic amb un import de 0 euros i un seguit de categories i subcategories per defecte, cosa que permet a l'usuari començar a utilitzar les funcionalitats de l'aplicació sense necessitat de configurar res. Per últim, a part del token, també es retornen les dades pertinents als comptes bancaris i a les categories i subcategories de l'usuari.

Pel que fa al registre i el login en la part de frontend s'han generat dues pantalles, on a cada una apareix un formulari amb la informació necessària. Per tal de validar els inputs dels formularis, s'han utilitzat expressions regulars[12], el qual serveix per a tots els inputs de totes les pantalles explicades a continuació.

A continuació s'ha generat el middleware que s'encarrega de validar el token d'autenticació situat en la petició i que decideix si es denega la petició o si es continua amb aquesta, afegint l'email extret del token en la petició per tal que els controladors tinguin coneixement de l'usuari que ha fet la petició. Aquest middleware s'utilitza en totes les rutes que venen a continuació.

Posteriorment, s'han desenvolupat les rutes, controladors i el model encarregats de gestionar tot el relacionat amb els comptes bancaris, és a dir, permetre la creació, eliminació i modificació d'aquests juntament amb l'obtenció de les dades necessàries per mostrar els comptes en el frontend, on s'han obtingut un total de quatre endpoints. Respecte a la part de frontend s'han generat tres pantalles que permeten a l'usuari fer el seguiment i la gestió dels seus comptes: una pantalla on es mostra el llistat dels comptes de l'usuari, indicant l'import i el propietari d'aquest, una pantalla amb un formulari per tal de crear un nou compte i per últim una pantalla que s'accedeix a partir del llistat i que permet modificar els camps del compte a partir d'un formulari.

Seguidament, s'han desenvolupat les rutes, controladors i el model que s'encarreguen de gestionar tot el relacionat amb les despeses dels usuaris, el que permeten crear, eliminar i modificar-les juntament amb la capacitat d'obtenir les dades necessàries pel frontend per tal de mostrar-les, on s'han obtingut un total de quatre endpoints. També s'han desenvolupat un total de tres triggers de MySQL encarregats de modificar els imports dels comptes bancaris en el moment en què s'afecti una despesa associada a aquests. Per altra banda, en el frontend s'han desenvolupat un total de cinc pantalles que permeten a l'usuari tenir un seguiment de les seves despeses juntament amb proporcionar-li la capacitat de gestionar i crear-ne de noves: una pantalla on es mostren totes les despeses de l'últim mes separades per quatre espais temporals diferents (Avui, ahir, aquesta setmana i aquest mes), on es mostra la categoria i subcategoria de la despesa i el cost, una pantalla on es mostren els costos totals de les despeses de l'últim mes per categories i internament per subcategories, una pantalla on es mostra a partir d'una gràfica la informació de les despeses per categories juntament amb un percentatge de quantitat de despesa d'una categoria respecte a les altres, per tal de realitzar la gràfica s'ha utilitzat la llibreria react-native-svg-charts[13], una pantalla amb un formulari que permet crear una nova despesa configurant els camps explicats en els requisits, per tal de fer la selecció de la data, ja que la gestió d'aquestes sol ser molt tediosa s'ha utilitzat la llibreria Moment.js[14], la qual permet fer operacions matemàtiques sobre aquestes, i, per altra banda, s'ha utilitzat la llibreria react-native-date-picker[15], la qual permet crear un component de selecció de dates configurable, i per últim, una pantalla amb un formulari ja omplert que conté la informació d'una despesa i que permet modificar els camps d'aquesta o eliminar-la si s'escau.

Respecte al desenvolupament de les funcionalitats sobre

la gestió dels ingressos, s'ha seguit el mateix procediment que en la gestió de les despeses a causa de la similitud que comparteixen, per tant, per la part del backend s'han generat noves rutes, controladors i models, obtenint quatre endpoints. Pel que fa al frontend, també s'han generat cinc pantalles, tres que mostren informació dels ingressos i dues que permeten crear-los, modificar-los i eliminar-los.

A continuació s'han creat les rutes, controladors i model que permeten administrar les categories i subcategories, és a dir, que permeten crear-les, modificar-les, eliminar-les i obtenir-les pel frontend, on s'han obtingut un total de quatre endpoints. A causa del disseny de la base de dades, com que existeix una taula per categories i un altre per subcategories i la modificació i creació de categories implica també la creació, eliminació i modificació de les subcategories, s'han utilitzat transaccions MySQL per tal d'agrupar les diferents peticions que hi ha entre el backend i la base de dades, aconseguint així mantenir una consistència en les dades en casos on pugui sorgir un error. Pel que fa a la part de frontend s'han desenvolupat tres pantalles que permeten a l'usuari fer la gestió de les categories i subcategories. Per tal facilitar la feina a l'usuari a l'hora d'administrar les subcategories, s'ha fet un redisseny respecte al disseny inicial de l'aplicació, on en comptes de tenir un sol input on es posen totes les subcategories juntes, hi ha un input per a cada subcategoria.

Seguidament, s'han desenvolupat les rutes, controladors i models que permeten administrar els pressupostos de l'usuari, obtenint un total de quatre endpoints. Per tal d'actualitzar els pressupostos en el moment en què es generi, modifiqui o elimini despesa, s'han modificat els tres triggers de MySQL que controlaven els imports dels comptes bancaris, afegint la funcionalitat de canviar la quantitat de pressupost restant. Per altra banda, s'ha desenvolupat un event de MySQL que s'activa a cada hora encarregat de seleccionar aquells pressupostos que ja han estat finalitzats i que estan marcats com a repetitius, i modificar-los per tal d'obtenir un nou pressupost amb un import de 0€ o de la resta que quedava en el pressupost en el cas que aquest s'hagi marcat amb l'opció de traspàs i una nova data de finalització corresponent al període establert en el pressupost. Respecte a l'apartat de frontend, s'han desenvolupat tres pantalles per tal de visualitzar els pressupostos actius, juntament amb l'administració d'aquests. Per tal de facilitar la selecció de comptes bancaris i categories que afecten als pressupostos, s'ha utilitzat l'element chips de la llibreria react-native-paper, al contrari que el disseny inicial de la pantalla.

A continuació, s'ha desenvolupat tot el codi que permet mantenir una sincronització de la informació entre el backend i el frontend, degut a que al tenir comptes bancaris compartits, quan un usuari faci una modificació que afecti a un compte, aquesta s'ha de replicar en els usuaris que tinguin accés a aquest. Per tal d'aconseguir aquesta sincronització s'ha fet ús de Firebase el qual permet enviar notificacions des de l'API fins a l'App juntament amb la llibreria React Native Firebase[16], la qual permet treballar amb Firebase des de React Native. El que s'assoleix amb aquestes llibreries és que en el moment en què un usuari faci un canvi que afecti a un compte, s'envia una notificació a tots els usuaris actius que tenen accés a aquell compte indicant el canvi que s'ha fet i d'aquesta forma, aconseguir sincronitzar les dades en temps real. Per altra banda, també s'ha habilitat que els usuaris puguin utilitzar els comptes que se li han compartit a l'hora de crear noves despeses, ingressos i pressupostos, el que ha requerit una modificació en els controladors del login i dels comptes bancaris. Per tal d'utilitzar Firebase com a solució de missatgeria s'ha generat un nou projecte en la consola de Firebase del qual s'ha obtingut una clau privada necessària per la comunicació entre el backend i el firebase. Després d'implementar la clau, s'han modificat els controladors de despeses i ingressos on s'ha afegit la funcionalitat de notificar als usuaris afectats per un canvi en el moment en què els hi arriba una petició. Per tal de que la notificació arribi als usuaris correctes, s'ha d'utilitzar el token FCM, el qual és un codi únic per a cada dispositiu, per tant, s'ha modificat el login i el registre tant en la part de frontend, on s'obté aquest token a partir de la llibreria React Native Firebase, com en backend, on es guarda el token en la base de dades per poder recuperar-lo després. A partir d'aquí, s'utilitza la llibreria firebase-admin en el backend per enviar les notificacions al frontend amb infor amb prioritat alta amb les dades de la despesa/ingrés en el camp data, on aquest s'encarrega d'actualitzar la informació que veu l'usuari. Per altra banda, s'ha desenvolupat la funcionalitat de notificació en quan un pressupost està superant el límit de despesa del 80% del pressupost inicial, en la qual s'envia una notificació des de el backend cap al frontend quan es compleix el requisit anterior en el moment en què es faci un moviment amb una despesa. Al contrari que la notificació de sincronització, les dades van en el camp notification en format de missatge, indicant així que no es tracta d'un missatge normal amb dades sinó d'una notificació a mostrar en el dispositiu.

Respecte a la navegació dins de l'aplicació, tal com s'ha comentat anteriorment, s'ha utilitzat React Navigation[17], el qual proporciona diferents tipus de navegació entre components i pantalles React. Pel projecte s'ha realitzat la jerarquia de navegadors de la figura 3. Aquesta jerarquia comença amb el navegador d'autenticació, el qual es de tipus Stack i s'encarrega de mostrar i navegar entre les pantalles de registre i login en el cas que l'usuari no estigui connectat al sistema, és a dir, quan no hi ha un token d'autenticació en el Redux, o de deixar que el navegador principal s'encarregui de la navegació. Respecte al navegador principal, de tipus Drawer (el que implica que aquest és el navegador que apareix al lateral de l'aplicació al clicar el botó del menú), és el que s'utilitza per recórrer entre els diferents mòduls que té l'aplicació. Per a cada mòdul hi ha un navegador, de tipus Stack, que conté les diferents pantalles d'aquest. Aquesta jerarquia permet tenir un millor control sobre els diferents aspectes de la navegació, degut a que d'aquesta forma, mantenim un historial diferent per a cada navegador a part que es pot configurar diferents opcions per a cada un, com per exemple l'aspecte del header de l'App, on per certes pantalles no es mostra, o en altres pantalles apareix el nom amb el botó de menú, però en altres ha d'aparèixer el botó de tirar enrere.

A continuació s'ha generat la funcionalitat de desconnexió de l'usuari, on s'ha afegit un botó en el header de l'app que s'encarrega d'eliminar tota la informació emmagatzemada en l'aplicació a partir del Redux. Per altra banda, també s'ha afegit la funcionalitat d'eliminar el perfil, el qual se situa en el menú lateral de l'aplicació, el qual fa una crida al backend perquè aquest elimini l'usuari. Per tant, s'ha ge-

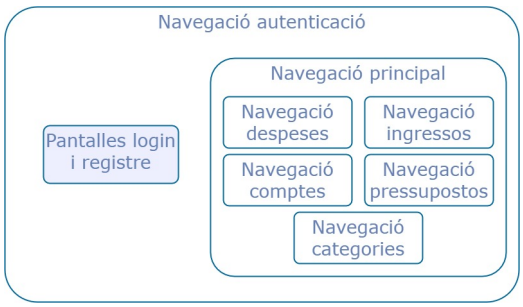


Fig. 3: Jerarquia de navegadors

nerat una nova ruta i controlador, traduït a un nou endpoint, encarregat d'eliminar l'usuari, el que implica eliminar tota la informació associada a aquest.

Per últim s'ha desenvolupat l'autenticació via Google, és a dir, poder registrar o loguejar un usuari utilitzant l'API de Google sense necessitat de contrasenyes. Per això, s'ha desenvolupat una nova ruta i controlador sobre els usuaris, generant un nou endpoint, en el qual es rep una ID generada per Google i mitjançant la llibreria google-auth-library, es fa una petició a l'API de Google per obtenir l'email de l'usuari, cosa que permet identificar-lo. Un cop obtingut l'email, es registra a l'usuari en el cas que no ho estigui ja i es logueja com un usuari normal. Respecte al frontend s'ha utilitzat la llibreria react-native-google-signin la qual proporciona el component que actua com a botó juntament amb la crida a la API de Google que retorna la informació de l'usuari on es troba la ID utilitzada en el backend.

4.3 Posada en marxa

A continuació s'exposa les tasques realitzades en la posada en marxa del servidor de backend en un entorn més real. Primer de tot, s'ha modificat l'inici del servidor per tal de treballar via HTTPS, el que afegeix una capa de seguretat en la comunicació entre les aplicacions i el backend. Per això s'ha generat un certificat SSL autofirmat mitjançant OpenSSL juntament amb una clau privada, el qual ens genera un seguit de fitxers que s'utilitzen en el servidor. Un cop afegits en el servidor, s'ha utilitzat la llibreria https en el fitxer index.js del backend per iniciar el servidor utilitzant el certificat i clau anteriors i escoltant pel port 443, obtenint així una comunicació més segura.

Respecte al servidor, s'ha utilitzat una instància de AWS EC2, la qual proporciona una infraestructura modificable amb un sistema Linux, amb la qual s'ha pogut comprovar el funcionament del sistema en un entorn real.

5 Resultats

El resultat obtingut en aquest treball ha estat una aplicació Android que juntament amb un servidor backend és capaç de complir amb els objectius establerts al principi del projecte i, per tant, permetre a l'usuari gestionar els seus comptes bancaris al tenir un seguiment de les despeses i els ingressos i a la vegada, poder generar pressupostos per tenir un millor control de les despeses generades.

Tal i com s'ha explicat en el desenvolupament, es pot dividir l'aplicació en els següents mòduls, on cada un compleix un seguit de requisits:

- **Mòdul d'accés**: Aquest mòdul permet registrar nous usuaris al sistema, permet fer l'autentificació al sistema per accedir a les funcionalitats de l'aplicació i s'encarrega de gestionar els canvis de contrasenya. També s'encarrega de la desconnexió de l'usuari del sistema i per últim de l'eliminació de l'usuari juntament amb tota la seva informació.
- **Mòdul de comptes**: Aquest mòdul permet visualitzar els comptes, tant els propis com els compartits, mostrant tant l'import actual del compte com l'usuari propietari en el cas que sigui compartit. També permet crear nous comptes o modificar-los a partir d'un formulari on es pot indicar si aquest serà compartit, juntament amb el nom i import inicial del compte. Per altra banda, els permet eliminar. També s'encarrega de crear un compte base amb import zero en el moment en què es registri un nou usuari al sistema.
- **Mòdul de ingressos/despeses**: Aquests mòduls permeten visualitzar els moviments de tres formes diferents: separats en 4 espais temporals, separats per categories i internament per subcategories i per últim, en forma de gràfica amb informació addicional com el percentatge de moviment d'una categoria respecte a les altres. Per altra banda, també permet crear nous moviments o modificar-los a partir d'un formulari. També permet eliminar els moviments. Aquest mòdul és l'encarregat d'actualitzar la informació dels comptes i pressupostos de forma automàtica en el moment de tractar un moviment.
- **Mòdul de categories**: Aquest mòdul serveix per visualitzar les diferents categories i subcategories d'un usuari. També permet crear noves categories amb les subcategories relacionades o modificar-les. Per altra banda, permet eliminar categories tenint en compte les relacions amb els moviments i les subcategories. Aquest mòdul s'encarrega també de generar les categories per defecte per tal que l'usuari pugui començar a utilitzar l'aplicació des de el primer minut.
- **Mòdul de pressupostos**: Aquest mòdul s'encarrega de mostrar els pressupostos amb l'import inicial i l'import actual, és a dir, la quantitat de pressupost que queda fins que es finalitzi, juntament amb la data de finalització de l'import. També permet a l'usuari generar o modificar nous pressupostos, on pot definir en quin espai de temps estarà actiu a part de relacionar-los amb comptes i categories. Addicionalment permet eliminar-los o modificar-los. Per altra banda, s'encarrega de generar els nous pressupostos en cas que n'hagi finalitzat un de categoria repetitiu i de fer la transferència de l'import sobrant d'un pressupost finalitzat si aquest s'ha marcat com a transferible. Per últim, és el mòdul encarregat de notificar als usuaris del fet que un pressupost està arribant al límit d'import.
- **Mòdul de sincronització**: Aquest mòdul, el qual no té interfície gràfica, s'encarrega de mantenir sincronitzada la informació entre el backend i el frontend a causa de modificacions en comptes compartits o moviments que afectin aquests mitjançant una comunicació via notificacions.

6 Linies de futur

Pel que fa a la continuitat del projecte, s'hauria de desenvolupar l'aplicació per sistemes iOS, el qual implicaria afegir certes modificacions en la part de Front-End amb la finalitat de que l'aplicació funcioni correctament amb aquest sistema juntament amb certes modificacions en el disseny per tal de complir amb els seus estàndards. També s'hauria de desenvolupar una versió web per facilitar el accés a les funcionalitats sense necessitar un dispositiu mòbil.

Per últim, per millorar la satisfacció dels clients, es podria connectar el compte bancari real dels usuaris amb l'aplicació i d'aquesta forma generar de forma automàtica els moviments d'aquests.

7 Conclusions

Com a conclusions finals podem extreure que l'aplicació generada ha estat capaç de complir amb els objectius proposats a l'inici, gràcies al fet que compleix amb tots els requisits funcionals els quals estan fortament relacionats amb aquests.

Per tant, s'ha desenvolupat una aplicació que permet als usuaris gestionar els seus comptes, a partir de tenir un control tant de les seves despeses com dels seus ingressos. També permet crear pressupostos per tenir un millor seguiment de les despeses a part de tenir la capacitat de compartir comptes amb altres usuaris.

Respecte a les eines utilitzades durant el desenvolupament del projecte, es pot extreure que frameworks com React Native o Node.js, ens poden ajudar molt respecte a la creació d'aplicacions, a causa de la gran comunitat que tenen darrera, on cada dia apareixen més llibreries compatibles amb aquests. Això implica que cada cop s'hagi de generar menys codi des de zero de termes que ja han estat tractats per la comunitat i que per tant ens puguem centrar en generar funcionalitats que ens diferenciïn de la competència, el que acaba aportant una millor satisfacció en els usuari finals.

Per últim, pel que fa al treball realitzat, si hagués de fer un projecte similar a aquest, optaria per treballar directament amb Firebase en comptes de generar un backend des de zero, ja que al no tenir funcionalitats molt complexes, es podria haver desenvolupat a partir de la base de dades que proporciona el mateix Firebase, el que agilitzaria molt el treball a realitzar.

Agraïments

Primer de tot, vull agrair a la meva família per ajudar-me econòmicament a cursar una carrera que volia fer des de que vaig descobrir el que era la programació. Per altra banda, agrair també els companys de la carrera, amb els quals he compartit molts bons moments. Per últim, agrair a tot el professorat i en especial a la tutora del TFG, l'Helena Bolta Torrell, per compartir els seus coneixements sobre l'Enginyeria amb mi.

Referències

[1] Jessica Young, Don Davis and Fareeha Ali, "US ecommerce grows 14.2% in 2021," Digital Commerce 360, 21-Feb-2022. [Online]. Available: https://www.digitalcommerce360.com/article/us-ecommerce-sales/

[2] R. Cruze, "Retail therapy: Are emotions draining your wallet?," Ramsey Solutions, 26-Jan-2022. [Online]. Available: https://www.ramseysolutions.com/budgeting/retail-therapy-when-emotions-fill-shopping-bag

[3] J. Feldman and M. Caldwell, "7 reasons why you should budget your money," The Balance, 26-Jan-2022. [Online]. Available: https://www.thebalance.com/reasons-to-budget-money-2385699

[4] T. Inc., "Monitoriza Todas Tus Tarjetas y dinero en un solo lugar.," Toshl Finance. [Online]. Available: https://toshl.com/es/

[5] "Kanban principles," Knowledge Train, 08-Dec-2021. [Online]. Available: https://www.knowledgetrain.co.uk/agile/agile-project-management/agile-project-management-course/kanban-principles.

[6] "React native · learn once, write anywhere," React Native RSS. [Online]. Available: https://reactnative.dev/

[7] "Node.js web application framework," Express. [Online]. Available: https://expressjs.com/

[8] "Firebase," Google. [Online]. Available: https://firebase.google.com/?hl=es

[9] "Redux - a predictable state container for JavaScript apps.: Redux," A predictable state container for JavaScript apps. [Online]. Available: https://redux.js.org/

[10] auth0.com, "Jwt.io," JSON Web Tokens. [Online]. Available: https://jwt.io/ç

[11] "Bcrypt," npm. [Online]. Available: https://www.npmjs.com/package/bcrypt

[12] "Expresión regular," Wikipedia, 14-Oct-2021. [Online]. Available: https://es.wikipedia.org/wiki/Expresi%C3%B3n_regular

[13] JesperLekland, "Jesperlekland/react-native-SVG-charts: One library to rule all charts for react native," GitHub. [Online]. Available: https://github.com/JesperLekland/react-native-svg-charts

[14] "Moment.js," Moment.js — Home. [Online]. Available: https://momentjs.com/

[15] Henninghall, "Henninghall/react-native-date-picker: React native date picker is datetime Picker for Android and IOS.," GitHub. [Online]. Available: https://github.com/henninghall/react-native-date-picker

[16] "React native Firebase," React Native Firebase. [Online]. Available: https://rnfirebase.io

[17] "React navigation: React navigation," React Navigation RSS. [Online]. Available: https://reactnavigation.org/

APÈNDIX

A.1 Casos de test

En aquesta secció del apèndix es troben alguns dels tests realitzats durant el desenvolupament del projecte per tal de comprovar el correcte funcionament d'aquest.

Pel que fa als tests de backend:

Globals		
Fet	**Esperat**	**Exemple**
Cas 1		
En end-points on es necessita autenticació, no hi ha token en el header o aquest és erroni	S'ha de tornar una resposta 401 indicant que la autenticació és incorrecte	Request sense header o token inventat.
Cas 2		
En end-points referents a despeses, ingressos i pressupostos, s'utilitza una id de categoria, subcategoria o compte que no existeix en l'espai del usuari.	S'ha de tornar una resposta 409 indicant que la informació no es correspon amb el que hi ha guardat, s'ha de retornar la llista de comptes o la de categories amb tota la informació sobre aquestes (similar al fer login o registre)	Request amb una id que no existeixi a la bd.

Crear comptes		
Fet	**Esperat**	**Exemple**
Cas 1		
Els parametres name i/o import no existeixen	S'ha de tornar una resposta 400 indicant que els camps estam buits	{shared_user:"test@test.com"} {import: 0} {name: "New account"} {}
Cas 2		
Els paràmetres son correctes però el nom del compte ja existeix	S'ha de retornar una resposta 400 indicant que aquell compte ja existeix	{import: 0, name: "basic"}
Cas3		
Els paràmetres son correctes però el usuari de shared_user no existeix o es el propi usuari	S'ha de retornar una resposta 400 indicant que el usuari compartit es incorrecte.	{import: 0, name:"New account", shared_user: "wrong@email.com"} {import: 0, name:"New account", shared_user: "test1@test.com"}
Cas 4		
Els paràmetres són correctes i en el cas que existeixi el paràmetre shared_user, aquest és correcte	S'ha de retornar una resposta 200 retornant el compte creat juntament amb el id. S'ha de haver generat un nou compte a la base de dades amb la informació corresponent.	{import: 0, name:"New account"} {import: 5, name:"New account", shared_user: "test@test.com"}

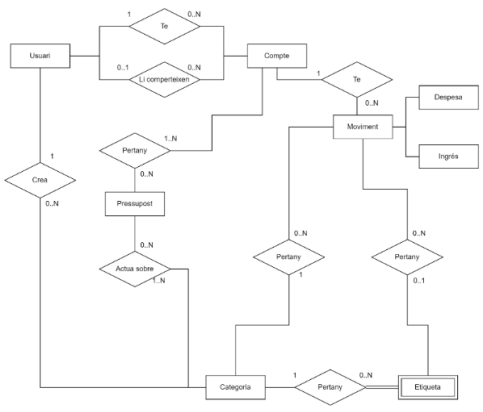
Respecte als tests de frontend:

Login	
Fet	**Esperat**
Cas 1	
Els camps estàn buits. Es prem el botó de loguejar.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "Please fill in all the required fields and check that they are ok."
Cas 2	
El camp email està buit i el password està omplert o viceversa. Es prem el botó de loguejar	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "Please fill in all the required fields and check that they are ok."
Cas 3	
El camp email esta omplert però no compleix la següent expresió regular (el camp password pot estar omplert o no): /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w\w+)+$/	S'ha de mostrar un missatge d'error sota l'input del email que indiqui el següent: "Email address is invalid!"
Cas 4	
Els dos camps estan omplerts però l'autenticació és incorrecte	S'ha de mostrar un missatge d'error sota el formulari que indiqui el següent : "Wrong username or password."
Cas 5	
Els dos camps estan omplerts i l'autenticació és correcte	S'ha d'accedir a les funcionalitats internes de l'aplicació

Crear compte	
Fet	**Esperat**
Cas 1	
Els camps estàn buits. Es prem el botó de crear.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "Please fill in all the required fields and check that they are ok."
Cas 2	
El camp de name està buit i els d'import està ple o viceversa. Es prem el botó de crear.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "Please fill in all the required fields and check that they are ok."
Cas 3	
S'omplen els camps name i import de forma correcta i es sel·lecciona el camp de compartir sense posar un email. Es prem el botó de crear.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "Please fill in all the required fields and check that they are ok."
Cas 4	
Es posa un import que no compleix amb la següent expresió regular: /^\-?\d+(\.\d{1,2})?$/	S'ha de mostrar un missatge d'error sota del input que indiqui el següent: "Invalid number format. Example: 42.51"
Cas 5	
S'omplen els camps name i import de forma correcta i es sel·lecciona el camp de compartir posant un email no registrat o el email del propi usuari. Es prem el botó de crear.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "The shared user doesn't exist." en el cas que sigui un email no registrat o "Form content is wrong!" en el cas que sigui el propi usuari.
Cas 6	
S'omplen els camps name i import on el name es el mateix que un altre compte de l'usuari. Es prem el botó de crear.	S'ha de mostrar un missatge d'error al final de formulari que indiqui el següent: "The account alredy exist."
Cas 7	
S'omplen els camps name i import de forma correcta i es sel·lecciona el camp de compartir posant un email. Es desselecciona el camp de compartir. O s'omplen els camps name i import de forma correcta	S'ha de crear el compte bancari sense tenir en compte el email compartit. S'ha de tornar a la pantalla on es mostren els comptes.

A.2 Disseny de la base de dades

A continuació es mostra el disseny de la base de dades realitzat al inici del projecte i que ha servit com a referència per declarar les entitats principals del projecte.



A.3 Disseny de pantalles wireframe

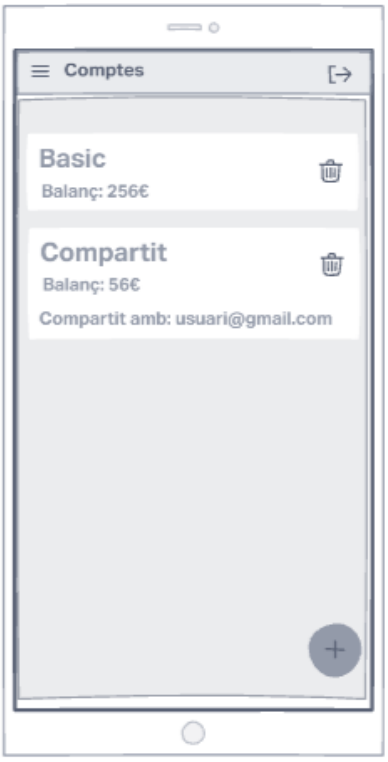
A continuació es mostren alguns dels dissenys de les diferents pantalles que es van fer en l'inici del projecte i que serveixen tant per definir la navegació dins de l'aplicació com per facilitar el desenvolupament del frontend de l'aplicació. Cal dir que aquests dissenys han servit com a una referència i que no representen al disseny final de l'aplicació, és per això que el resultat final no ha mantingut la mateixa estructura o disseny.

Pantalla Login

Compleix requisits RF1.2, RF1.3 i RF1.7.
Si es prem el botó Sign Up va a "Pantalla Registre"
Si es prem el botó Login o Sign Up with Google va a "Pantalla Despeses" si tot es correcte.

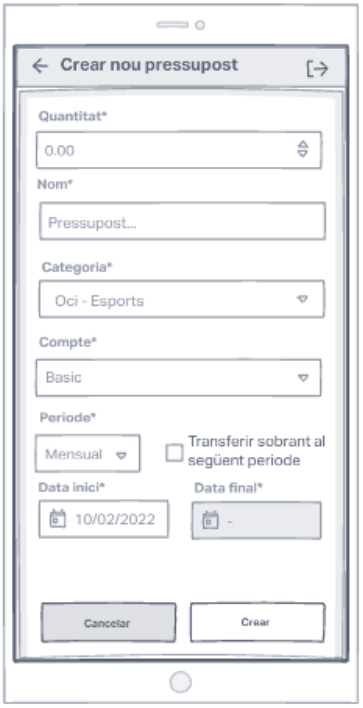
Pantalla Despeses

Compleix requisits RF3.4.
Si es prem el botó "+" va a "Pantalla crear despesa"
Amb la barra de navegació superior es pot anar a "Pantalla gràfica" i "Pantalla Despeses per categories".
Si es clica una de les despeses es va a "Pantalla Informació despesa"



Pantalla Comptes

Compleix requisits RF2.3, RF2.5.
Si es prem el botó "+" va a "Pantalla crear compte"
Si es prem el icone de la paperera s'elimina el "Compte associat".



Pantalla crear pressupost

**Compleix requisits RF4.1, RF4.2, RF4.3.
Si es prem el botó "Crear" genera un nou pressupost amb l'informació associada i torna a la pantalla anterior, per altra banda si es prem el botó "Cancelar" o "<-" torna a la pantalla anterior sense fer res.**