UNIVERSITAT DE BARCELONA

Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

DEFINICIÓ DE RUTES SALUDABLES A PARTIR DE DADES DE GEOLOCALITZACIÓ I CONTAMINACIÓ

Albert Pérez Costa

Director: Laura Igual Muñoz i
Maria Grau Magaña
Realitzat a: Departament de
Matemàtiques i
Informàtica

Barcelona, 24 de gener de 2023

ÍNDEX

Abstract (Català)

Avui en dia, l'esperança de vida de les persones que viuen a Catalunya és de les més llargues del món. L'envelliment poblacional s'ha d'acompanyar d'una bona qualitat de vida. Tot i els avenços tecnològics i mèdics, aquestes persones són més vulnerables a malalties i infeccions, i si ajuntem això amb el factor dels grans nivells de contaminació que es tenen actualment i que s'ha demostrat que tenen un impacte directe en les malalties, ens dona un resultat a tenir en compte per lluitar per un envelliment saludable. Quant de temps vivim i quina proporció d'aquesta vida es passa en bona salut tenen implicacions importants per a les persones i les societats.

Per tant, en aquest projecte es vol intentar aportar un granet de sorra, per promoure un envelliment saludable a través de l'exercici. Concretament, tractarà d'una aplicació mòbil orientada principalment a persones grans, on es mostraran rutes per sortir a caminar i/o córrer, però indicant els nivells de contaminació i meteorologia que hi ha en aquell moment, promovent així una vida saludable fent exercici i a més fent-ho en les condicions més adients possibles.

En aquest document es presenta la planificació seguida per a acomplir aquest projecte i el disseny i implementació de l'aplicació que compleixi amb les característiques comentades.

Abstract (Castellano)

Hoy en día, la esperanza de vida de las personas que viven en Cataluña es de las más largas del mundo. Sin embargo, el envejecimiento poblacional debe acompañarse de una buena calidad de vida. Por otro lado, a pesar de los avances tecnológicos y médicos, estas personas son más vulnerables a enfermedades e infecciones, y si juntamos esto con los grandes niveles de contaminación que hay actualmente en las ciudades y que se ha demostrado que tienen un impacto directo en las enfermedades, nos lleva a un factor a tener en cuenta para luchar por un envejecimiento saludable. Cuánto tiempo vivimos y qué proporción de esta vida se pasa con buena salud tienen importantes implicaciones para las personas y las sociedades.

Por tanto, en este proyecto se quiere aportar un granito de arena para promover un envejecimiento saludable a través del ejercicio. Concretamente, se implementa una aplicación móvil orientada principalmente a personas mayores, donde se mostrarán rutas para salir a andar y/o correr, pero indicando los niveles de contaminación y meteorología que hay en ese momento, así como al día siguiente, promoviendo así una vida saludable fomentando el ejercicio en las condiciones más adecuadas posibles.

En este documento se presenta la planificación seguida para realizar este proyecto y el diseño e implementación de la aplicación que cumple con las características comentadas.

Abstract (English)

Today, the life expectancy of people living in Catalonia is one of the longest in the world. Population aging must be accompanied by a good quality of life. Despite technological and medical advances, these people are more vulnerable to diseases and infections, and if we add this to the factor of the high levels of pollution that exist today that have been shown to have a direct impact on diseases, it gives a result to take into account to fight for healthy aging. How long we live and what proportion of this life is spent in good health have important implications for individuals and societies.

Therefore, in this project we want to try to contribute a grain of sand, to promote healthy aging through exercise. Specifically, it will deal with a mobile application aimed mainly at the elderly, where routes will be shown to go for a walk and/or run, but indicating the levels of pollution and meteorology that exist at that moment, thus promoting a healthy life by exercising and also doing it in the most appropriate conditions possible.

This document presents the planning followed to carry out this project and the design and implementation of the application that meets the characteristics mentioned.

1. Introducció

1.1. Context i Motivació:

Aquest projecte està emmarcat dins d'un estudi de recerca liderat per la Dra. Maria Grau de la Facultat de Medicina de la UB que té com a objectiu empoderar a les persones grans per aconseguir un procés d'envelliment saludable.

Quant de temps vivim i quina proporció d'aquesta vida es passa en bona salut tenen implicacions importants per a les persones i les societats.

Tradicionalment, la fragilitat de les societats envellides s'ha vist exposada per la càrrega social i econòmica de les malalties no transmissibles; tanmateix, hem après de la crisi de la COVID-19 sobre els efectes potencials –i ràpids– de les malalties infeccioses.
Tot i que les innovacions mèdiques modernes han augmentat la longevitat durant el segle passat, el canvi cap a una població més gran amb un major risc de malalties transmissibles i no transmissibles té el potencial de generar una crisi sanitària en els propers anys. Ara és el moment d'avançar i centrar-nos en un envelliment saludable i reeixit mitjançant una vida activa. De fet, la inactivitat física és la quarta causa de mort a tot el món, amb un cost aproximat de 67.500 milions de dòlars anuals als sistemes sanitaris a nivell mundial.
Tanmateix, el model clàssic dels determinants de la salut mostra que els estils de vida individuals estan incrustats en les normes i xarxes socials, i en les condicions de vida i de treball, que al seu torn estan relacionades amb l'entorn socioeconòmic i cultural més ampli. Per tant, el repte de l'envelliment actiu i saludable requereix reconèixer que aquesta qüestió depèn no només de l'individu sinó també de les xarxes socials i comunitàries i de les condicions generals socioeconòmiques, culturals i ambientals.

Per promoure una vida activa, és imprescindible entendre les vies a través de les quals els entorns físics i socials influeixen en la vida activa i, al seu torn, com la vida activa influeix en els resultats de salut. El grup creixent de persones de 55 anys i més pot aconseguir un envelliment saludable mitjançant la vida activa en el panorama actual d'una ciutat intel·ligent? El desenvolupament de la vida activa ha demostrat efectes beneficiosos per prevenir tant les malalties transmissibles com les no transmissibles. La nostra hipòtesi és que si oferim recomanacions personalitzades sobre l'activitat física segons les zones verdes del barri, el clima i la qualitat de l'aire, empoderem a la població per aconseguir un procés d'envelliment saludable.

Com comenta l'estudi, es vol tenir en compte les condicions ambientals sobre la població, ja que l'exposició a nivells alts de contaminació de l'aire pot causar una varietat de resultats adversos a la salut. Aquesta contaminació pot augmentar el risc d'infeccions respiratòries, malalties cardíaques, accidents cerebrovasculars i càncer de pulmó. Tant l'exposició a curt termini com a llarg termini als contaminants de l'aire s'ha associat amb impactes adversos a la salut. Els impactes més severs afecten les

persones que ja estan malaltes. Els nens, els ancians i els pobres són més susceptibles. Els contaminants més nocius per a la salut, estretament associats amb la mortalitat prematura excessiva, són partícules fines PM2,5 que penetren profundament als conductes pulmonars.

Si bé en general, la qualitat de l'aire als països d'alts ingressos ha millorat en les darreres dècades, els efectes adversos de la contaminació de l'aire ambiental exterior a la salut per partícules (PM per les sigles en anglès) continuen sent un problema mundial de salut pública, fins i tot a nivells relativament baixos.

De tot això, Barcelona no se n'escapa. Avui en dia és notícia la quantitat de contaminació que hi ha a tot el món i sobretot a les ciutats més grans com Barcelona i és per això que ha sorgit la necessitat de realitzar millores tecnològiques amb les quals ajudar a la població a reduir l'exposició, o com a mínim fer-los conscients dels efectes nocius de la contaminació.

Com és el cas d'aquesta aplicació, que té la intenció d'ajudar als usuaris a fer exercici caminant/corrents per zones més verdes i no tan verdes com el centre de la ciutat, però informant dels nivells de contaminació perquè les persones siguin conscients de quan és millor sortir a caminar. L'aplicació està orientada a gent gran, amb una experiència d'usuari el més simple possible, ja que la gent d'avançada edat sol tenir menys costum d'utilitzar les tecnologies. Per això, és una app amb indicacions clares i concises i sense un excés de vocabulari tècnic. Això no impedeix que pugui ser utilitzada per gent més jove sense problema.

A més de fer recomanacions sobre les rutes menys exposades a la contaminació, també s'inclouen avisos sobre les condicions meteorològiques. Es considera que un passeig per Barcelona en plena onada de calor pot ser més perjudicial que beneficiós. Així mateix, tampoc s'aconsella caminar si hi ha previsió de pluja.

Per últim, l'usuari podrà escollir entre diferents nivells de dificultat i quan vol fer-la. Les visualitzacions de les rutes en el mapa permeten escollir la ruta més adient per al moment desitjat.

1.2. Objectius:

El principal objectiu d'aquest treball és el de realitzar una aplicació mòbil en forma de MVP (Producte Mínim Viable). La qual permet als usuaris sortir a caminar i/o córrer mostrant rutes per la zona de Barcelona on està situat l'usuari (geolocalitzat) amb indicacions dels nivells de contaminació i també de la meteorologia. D'aquesta forma els usuaris seran conscients de la contaminació i condicions meteorològiques del moment en cada una de les rutes. Poden escollir d'aquesta forma les rutes més saludables o en el cas que totes les rutes tinguin una qualitat d'aire o condicions meteorològiques desfavorable podrà decidir esperar i sortir en un altre moment o dia a fer exercici.

D'aquesta forma, es podrà ajudar al benestar de la població evitant que surtin en moments de molta contaminació o condicions adverses.

En aquest treball, els objectius marcats eren els de crear la primera versió de l'app, mostrant diferents tipus de rutes tan saludables com no, adaptant-les a les necessitats dels diferents usuaris. A continuació es detallen les opcions de rutes utilitzades a l'app i els objectius principals que es van establir per fer en aquesta primera versió de l'app.

Les rutes utilitzades són:

- **Proposta inicial:** Rutes extretes de l'AllTrails*, distribuïdes per tota l'àrea de Barcelona.

- **Proposta verda:** Rutes extretes de l'OpenData BCN, amb tots els parcs d'entre 0,5 - 1 ha de l'àrea de Barcelona. Aquestes rutes fan anar a l'usuari pel que denominen zones verdes, ja que tenen arbres i estan més aïllades de la contaminació.

- **Començar aquí:** Proposta de ruta que es genera automàticament des del punt on està l'usuari fins al parc (zona verda) més proper segons la dificultat que hagi escollit.

Quan parlem de necessitats, ens referim a les característiques que pot tenir la ruta, és a dir, un usuari podrà indicar la dificultat que vol que tinguin les rutes per les quals caminarà. Els nivells de dificultat es defineixen a partir de la distància de la ruta i hi haurà 3 nivells; **Fàcil, Mitjà, Difícil**.

A més d'aquestes opcions, es va decidir que l'usuari pogués indicar quan volia fer la ruta, podent seleccionar entre 3 opcions: **Avui, Demà, Demà passat.** I indicant en quina hora del dia preferia fer-la. D'aquesta forma, l'usuari podrà buscar les rutes per "Demà" per exemple i veure els nivells de contaminació i decidir en quin moment vol realitzar-la.

Un altre dels objectius, va ser indicar a part de la contaminació, la predicció meteorològica pel moment en què l'usuari seleccioni per fer sortir a caminar. Per tant, tindrà pel dia i hora seleccionat, informació sobre la contaminació i meteorologia de Barcelona per decidir si sortir o no.

Per a cadascuna de les rutes se l'informarà de la contaminació actual de la ruta i la previsió meteorològica que hi ha i en casos de valors elevats es mostrarà una advertència que desaconsella sortir en aquell moment. D'aquesta forma l'usuari tindrà la suficient informació per a decidir si sortir o millor esperar i anar un altre dia a caminar.

En els següents apartats s'entra més a fons en el que s'ha fet per cada tipus de ruta i totes les possibilitats que ofereix l'aplicació.

2. Anàlisi previ

Com a primer pas de l'anàlisi es va fer una recerca sobre aplicacions ja existents que ofereixin funcionalitats semblants a les d'aquesta aplicació, per agafar idees i analitzar que s'havia de fer i què no. Tenint present tots els objectius es va donar pas a la recerca d'informació per estructurar el desenvolupament. Es van investigar les diferents eines amb què realitzar la part del frontend de l'aplicació i, per un altre costat, les del backend.

2.1. Aplicacions ja existents

La recerca feta ha servit per veure les aplicacions de les quals es pot extreure informació útil per aplicar a la nostra i les no tan útils que s'han descartat per al nostre cas.

1) **Caliope:** Aquesta app ofereix previsions amb 48 hores d'antelació, informant-nos de l'estat de l'aire a tot Europa. Es tracta d'un sistema que tracta de predir segons les dades recollides i l'estat del clima, la situació a què ens enfrontarem.
2) **Tiempo iOS:** App que ofereix Apple de forma gratuïta que mostra la previsió meteorològica per ciutat i també els nivells de contaminació amb recomanacions i alertes.
3) **AllTrails:** Aquesta aplicació no dona informació sobre contaminació o temps, però si ofereix rutes per anar a caminar/córrer. És segurament l'aplicació més coneguda pels "runners".

Cap d'aquestes aplicacions de forma individual ofereix el mateix que el que s'ha fet en aquest TFG, ja que és una combinació de funcionalitats que poden oferir diferents apps. Principalment, s'han obtingut les rutes que ofereix de forma gratuïta l'AllTrails, però a diferència d'ells també s'indica previsions de meteorologia i contaminació com ofereixen les altres dues apps mencionades.

2.2. Recerca de tecnologies frontend

Per realitzar l'app vaig investigar diferents tecnologies amb les quals dur a terme l'aplicació amb totes les funcionalitats requerides. Primer, al que vaig donar màxima prioritat era que l'app fos compatible tant per Android com per iOS sense haver de duplicar codi en diferents plataformes. Pel que, l'opció més coneguda que era Android Studio que havia utilitzat durant la carrera va quedar descartada des d'un inici. Investigant, les opcions que van sortir van ser Flutter, React Native i Ionic.

- **Flutter:**
  Flutter (2017), és un framework de codi obert creat per Google que utilitza el llenguatge de programació anomenat Dart (creat específicament per substituir Javascript) que s'empra per treballar des del client, per tant, sense la necessitat d'utilitzar un framework de suport extern. Té continguda tant la vista com la lògica en un mateix fitxer Dart, tractant-se, doncs, d'un framework bastant encapsulat on cada pàgina o component és independent.

El framework ens dona la possibilitat de córrer l'aplicació en les plataformes Android, IOS i Web.

En el cas d'Android és necessari tenir instal·lat al nostre ordinador Android Studio. Això ens permet la possibilitat d'execució directa des de la consola o interfície, sempre que hi hagi un emulador o dispositiu disponible, utilitzant la comanda *flutter run*. Pel que fa a l'IOS és necessari XCode, aquest programa està només disponible per als ordinadors que tenen com a sistema operatiu Macos i com en el cas anterior també ens permet la possibilitat d'execució mitjançant consola o interfície, sempre que hi hagi un emulador o dispositiu disponible, utilitzant la comanda *flutter run*.

- **React Native:**
  React Native (2015), és un framework de codi obert creat per Facebook que utilitza el llenguatge de programació Javascript i Npm associat a Node, és a dir, es tracta d'un framework que utilitza llenguatges bastant comuns de l'àmbit web, encara que intenta no allunyar-se d'un rendiment proper al d'una implementació nativa. Utilitza un sistema de components que marquen les diferents pàgines creades, aquestes, de manera similar a Flutter contenen tant la vista com la lògica en el mateix fitxer Javascript.

  El framework ens permet córrer l'aplicació en Android i IOS, en el cas d'emulació Web és necessària la instal·lació de paquets externs a l'aplicació. Per poder emular en dispositius mòbils es necessita Android Studio i XCode, a més a més en ambdós casos tenim la possibilitat d'execució via consola mitjançant la comanda *react-native run-android –simulator="device" i react-native run-ios –simulator="device"* sempre que hi hagi un emulador o dispositiu disponible

- **Ionic:**
  Ionic (2013) és un framework de codi obert que permet desenvolupar aplicacions híbrides utilitzant Html5, Css, Javascript i Cordova, a més a més de donar la possibilitat d'escollir entre react.js, angular.js o vue.js per gestionar les aplicacions. Cordova permet utilitzar les tecnologies abans esmentades, evitant el desenvolupament natiu de cada plataforma mòbil individualment, encara que brinda l'oportunitat d'accés a components natius dels dispositius mòbils que poden ser necessaris per a la implementació de certes funcionalitats de l'aplicació.

  Ionic utilitza Npm associat a node, per la vista fa servir Html5 i Css, i per la lògica utilitza Typescript per a la implementació d'angular.js, react.js o vue.js. Per tant, aplica un desenvolupament semblant a aplicacions web però adaptada a aplicacions mòbils.

  El framework ens dona la possibilitat de córrer l'aplicació en les plataformes Android, IOS i Web. Per córrer tant en Android i IOS és necessari, com en els casos anteriors, de la instal·lació d'Android Studio i XCode. Per tant, tenim la possibilitat d'execució mitjançant una comanda de consola, *ionic cordova run android* o *ionic cordova run ios*.

Finalment, vaig decidir desenvolupar l'aplicació utilitzant React Native per la part del front end.
La idea principal per la qual es va decidir d'utilitzar React Native per sobre d'altres opcions va ser, com ja he dit, per la facilitat que ofereix a l'hora de crear aplicacions natives tant per Android com per iOS. A més, utilitza Javascript, un llenguatge molt conegut amb el que ja he tingut experiència i facilitava la corba d'aprenentatge. També, per la facilitat que ofereix gràcies al sistema de components que ofereix React, és de codi obert i empreses com Facebook la fan servir. El que indica que és una eina popular i amb bones referències.

2.3. Recerca de tecnologies backend

Per la part del backend, es necessitava bàsicament una forma en què guardar dades i poder accedir a elles. Es va decidir fer una API REST, ja que d'aquesta forma fem que el servidor faci la major part de la feina. D'aquesta forma tindrem un servidor per un costat per al tractament de les dades.

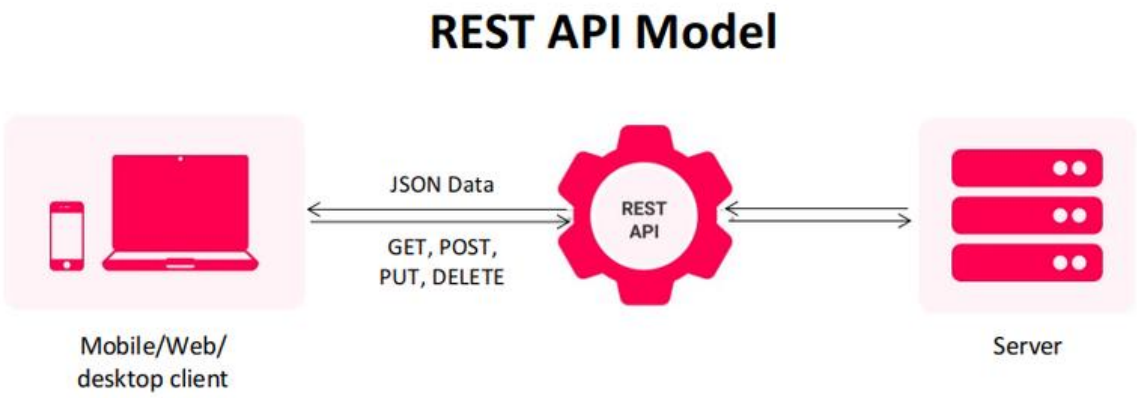


Figura 1: Model del funcionament d'una API REST

Un servei d'API REST, com veiem a la **Figura 1,** és un estil d'arquitectura de programari que es fa servir per descriure qualsevol interfície entre diferents sistemes que utilitzi HTTP per comunicar-se. REST significa Representational State Transfer (transferència d'estat representacional), cosa que vol dir que, entre dues trucades qualssevol, el servei no guarda les dades, aquesta responsabilitat queda delegada al backend, que serà el que emmagatzemarà, buscarà i actualitzarà les dades trobades a la base de dades. El client d'una API REST pot ser una aplicació Android o iOS o un navegador web, fins i tot pot ser qualsevol altre dispositiu o servei com ara televisions, Alexa o fins i tot un microones. D'aquesta manera amb el mateix servei d'API REST implementat es podria reutilitzar per implementar "Camina i Respira" a qualsevol altra plataforma.
En el meu cas s'ha utilitzat una base de dades relacional amb PostgreSQL utilitzant la plataforma d'Amazon Web Services, Relational Databases.

- **API REST:**
  En general, les API REST milloren les aplicacions web i les aplicacions mòbils que es distribueixen per Internet. Les aplicacions mòbils es tornen més escalables i

també és més fàcil modificar-les. En definitiva, l'aplicació serà més fiable, portàtil, visible i simplificada amb una API RESTful.

El servidor pot ajudar els usuaris a mantenir baixos els costos de les seves dades de xarxa i també a estalviar la durada de la bateria dels seus dispositius.

Així l'aplicació mòbil es beneficiarà si s'emmagatzemen les dades de manera remota en un servidor o eliminem algunes tasques difícils fent-les en un dispositiu remot. Connectar-se a una API remota mantindrà totes les dades segures quan s'emmagatzemen.

A més, permetre que el servidor faci la major part del treball, finalment, també beneficiarà els desenvolupadors. Els estalviarà temps i els donarà l'oportunitat de consolidar part de la codificació.

Un cop decidida l'API, es va decidir implementar-la en Python, un llenguatge molt conegut, que he utilitzat bastant durant la carrera i que proveeix bones eines per realitzar APIs REST. Concretament amb el framework Flask, que també havia utilitzat una mica. L'altra opció era utilitzar Django però la veritat que tots dos oferien coses molt semblants així que em vaig decantar pel framework que dominava una mica més.

El procés de desenvolupament tant del backend com del frontend s'explica amb més detall en l'apartat 6. Disseny i Implementació.

2.4. Visualització dels mapes

Un cop decidit el framework amb el que treballar, el següent pas va ser decidir quina eina s'utilitza per visualitzar les rutes per les quals l'usuari podrà caminar. Principalment, hi havien dues opcions, la més coneguda, Google Maps o Mapbox. Tots dos són proveïdors de mapes en línia personalitzables.

Finalment, com a principal tecnologia es va decidir utilitzar Mapbox per a la visualització de mapes i rutes. Ja que era gratuïta a diferència d'altres opcions com Google Maps que tenia un límit de peticions a la seva API de forma gratuïta.

Mapbox és avui en dia una de les plataformes de mapes de codi obert més importants del món. Ofereix una gran quantitat de productes i serveis als desenvolupadors per dissenyar mapes personalitzables i crear aplicacions basades en les seves eines.
Concretament, es farà servir el seu producte "Maps" que proporciona les eines de disseny i biblioteques necessàries per crear mapes dinàmics, eficaços i personalitzables.
També, es farà ús d'una de les seves APIs anomenada "Directions" per mostrar les diferents rutes a l'usuari.

3. Anàlisi

També va ser necessari un treball d'anàlisi per veure com organitzar el projecte per així satisfer tots els objectius plantejats.

3.1. Casos d'ús

Perquè s'entengui millor la navegació de l'aplicació que pot fer l'usuari, a continuació a la Figura 33 es mostra els casos d'ús amb totes les funcionalitats que es poden fer:

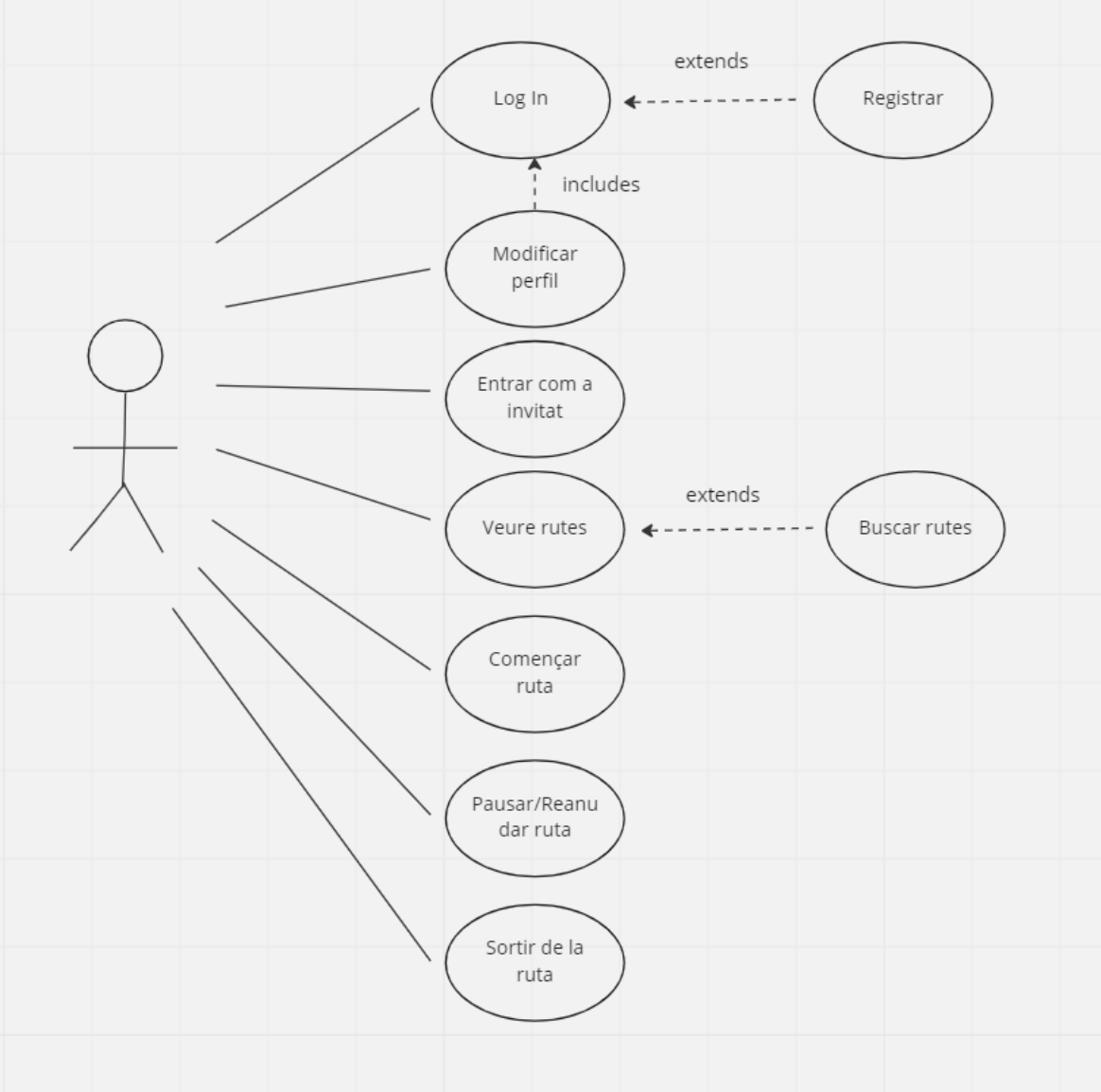


Figura 33: Diagrama de casos d'ús

3.2. User Stories

Una història d'usuari és la unitat de treball més petita en un marc àgil. És un objectiu final, no una funció, expressat des de la perspectiva de l'usuari del programari. Les històries d'usuari són unes poques frases en llenguatge senzill que descriuen el resultat desitjat. No entren en detalls, ja que els requisits s'hi afegeixen més tard.

Una història d'usuari és una explicació general i informal d'una funció de programari escrita des de la perspectiva de l'usuari final. El seu propòsit és articular com proporcionarà una funció de programari valor al client. Es componen de 3 parts:

- **Nom o títol:** Curt i concís.
- **Breu descripció:** Una user story sol desenvolupar-se d'acord amb una estructura fixa que descriu la funcionalitat a fer per l'usuari. Aquesta estructura sol ser la següent: Tipus d'usuari -> Acció -> Objectiu o "**Com** a, **vull** <objectiu/desitjos> **per** <finalitat>".
- **Criteris d'acceptació:** Els detalls del requisit s'exposen breument. Aquests criteris són una manera de determinar si s'han complert els requisits.

D'aquesta manera, la part interessada pot formular o negociar amb precisió els requisits d'un programa àgil.

Es va crear un Product Backlog a l'eina Trello amb les següents User Stories (US) mostrades a la Figura 2:

Títol	Descripció	Acceptance Criteria
US1 Sign Up	**Com** Usuari sense registrar y sense loguejar **Vull** entrar a l'aplicació i registrar-me a través del botó Sign Up **Per** tenir un compte a l'aplicació	Creació de compte utilitzant un correu electrònic personal, una contrasenya secreta i indicant Nom, Cognoms i Edat.
US2 Log In	**Com** usuari sense loguejar **Vull** iniciar sessió a l'aplicació **Per** tenir el meu compte connectat	Inici de sessió utilitzant correu electrònic i contrasenya per tenir accés a la interfície pròpia de l'usuari i que pugui guardar progressos

US3 Entrar com invitat	**Com** usuari sense loguejar **Vull** entrar a l'aplicació sense iniciar sessió **Per** accedir a les rutes	Accés al mapa i a les rutes sense necessitat d'iniciar sessió a través del botó "Entrar com Invitat"
US4 Log Out	**Com** usuari loguejat **Vull** tancar sessió a l'aplicació **Per** tenir el meu compte desconnectat	Un cop la sessió està iniciada, en tornar a l'Inici amb el botó de "Tancar Sessió" sortim del compte. També amb la sessió iniciada, des de la pantalla del mapa, fent click al botó superior dret accedim al menú lateral i es pot "Tancar sessió".
US5 Modificar perfil	**Com** usuari loguejat **Vull** modificar el meu perfil **Per** tenir el meu compte actualitzat	Amb sessió iniciada i des de la pantalla principal amb el mapa, fent click al botó superior dret accedim al menú lateral i tindrem l'opció "Modificar perfil". Allà podrem modificar el nom, correu, edat i contrasenya del compte.
US6 Visualitzar mapa sense rutes	**Com** Usuari registrat i loguejat (o invitat) **Vull** accedir al mapa **Per** visualitzar el mapa i la meva ubicació	Passem la pantalla d'inici i accedim al mapa on veure'm la geolocalització de l'usuari i la càmera es posicionarà sobre ell. Ens podrem moure pel mapa lliscant amb els dits. Però encara sense mostrar cap ruta.
US7 Filtrar rutes	**Com** Usuari registrat i loguejat (o invitat) **Vull** poder seleccionar el QUAN, DIFICULTAT i TIPUS DE RUTA **Per** filtrar les rutes	En la pantalla del mapa, hi ha un filtre a la part superior on es pot seleccionar 3 opcions per filtrar rutes: **QUAN ->** Avui, Demà, Demà passat **DIFICULTAT ->** Fàcil, Mitjà, Difícil **RUTA ->** Proposta inicial,

		Proposta verda, Començar aquí
US8 Veure rutes per defecte	**Com** Usuari registrat i loguejat (o invitat) **Vull** no filtrar les rutes **Per** veure les opcions per defecte	Si l'usuari no selecciona cap opció de filtre, en buscar rutes li sortiran en el mapa totes les rutes que hi ha, de tots els nivells i tipus i per al dia actual "Avui".
US9 Veure rutes PROPOSTA INICIAL	**Com** Usuari registrat i loguejat (o invitat) **Vull** filtrar per Ruta -> Proposta inicial **Per** veure les opcions de ruta que ofereix	Es mostraran rutes de la proposta inicial, que no estan catalogades com a "zones verdes" pel que normalment no passen per parcs. Es mostraran del nivell que esculli o de tots si no escull cap i per al dia seleccionat.
US10 Veure rutes PROPOSTA VERDA	**Com** Usuari registrat i loguejat (o invitat) **Vull** filtrar per Ruta -> Proposta verda **Per** veure les opcions de ruta que ofereix	Es mostraran rutes de la proposta verda, que normalment seran més sanes, ja que van per zones verdes com parcs. Es mostraran del nivell que esculli o de tots si no escull cap i per al dia seleccionat.
US11 Veure rutes COMENÇAR AQUÍ	**Com** Usuari registrat i loguejat (o invitat) **Vull** filtrar per Ruta -> Proposta verda **Per** veure les opcions de ruta que ofereix	Es crearà una única ruta automàtica des del punt on està l'usuari fins a una zona verda i tornarà fins al punt d'inici. La zona verda a la qual anirà dependrà del nivell de dificultat que seleccioni l'usuari. Si escull fàcil serà una propera, si va augmentant el nivell també la distància fins a la zona verda.
US12 Color ruta	**Com** Usuari registrat i loguejat (o invitat) **Vull** veure el color de la ruta	Les rutes podrà tenir 4 colors; Verd, Groc, Taronja, Vermell. On cadascun indica un nivell:

	al mapa **Per** saber el nivell de contaminació i previsió meteorològica que hi ha	**Verd ->** Si la ruta té contaminació < 50 ICA i no plou **Groc ->** Si la ruta té contaminació entre 50 i 100 ICA i no plou **Taronja->** Si la ruta té contaminació entre 100 i 150 ICA i/o si plou moderat **Vermell ->** Si la ruta té contaminació > 150 ICA i/o plou molt
US13 Veure "Informació de la ruta"	**Com** Usuari registrat i loguejat (o invitat) **Vull** iniciar sessió a l'aplicació **Per** tenir el meu compte connecta	Modal on es veu la **distància** que té la ruta, el **nivell de contaminació** ICA i la **previsió meteorològica** per al dia i hora seleccionat. A més, el color del requadre seguirà la mateixa lògica que el color de les rutes: **Verd ->** Si la ruta té contaminació < 50 ICA i no plou **Groc ->** Si la ruta té contaminació entre 50 i 100 ICA i no plou **Taronja->** Si la ruta té contaminació entre 100 i 150 ICA i/o si plou moderat **Vermell ->** Si la ruta té contaminació > 150 ICA i/o plou molt
US14 Tancar "Informació de la ruta"	**Com** Usuari registrat i loguejat (o invitat) **Vull** tancar la informació de ruta **Per** tornar a la selecció de	Un cop l'usuari ha fet click sobre una ruta i ha vist la informació, podrà tancar-ho fent click a la creueta superior dreta del modal o fent click sobre el mapa.

	rutes	
US15 Veure tutorial	**Com** usuari registrat **Vull** veure el tutorial **Per** entendre el funcionament de l'app	L'usuari s'acaba de registrar i li apareix una pantalla "tutorial" on explica que fa l'aplicació i com buscar rutes. Un cop llegit, prem en acceptar i surt del tutorial.
US16 Visualitzar advertències	**Com** Usuari registrat i loguejat (o invitat) **Vull** visualitzar les advertències **Per** ser conscient dels nivells de contaminació i meteorologia perillosos	L'usuari ha fet click sobre una ruta que té una contaminació elevada i/o previsió meteorològica desfavorable. Li apareix un requadre al mig de la pantalla desaconsellant sortir a caminar en aquell moment.
US17 Començar ruta	**Com** Usuari registrat i loguejat (o invitat) **Vull** començar la ruta seleccionada **Per** començar a caminar	L'usuari té clara quina ruta vol fer, fa click sobre ella i dins d'"Informació de la ruta" tindrà un botó per "Començar ruta", el qual li redirigeix a la navegació de la ruta per ja caminar. Tindrà un cronòmetre per veure el temps que porta caminant, un botó per parar la ruta i un altre per sortir.
US18 Pausar ruta	**Com** Usuari registrat i loguejat (o invitat) **Vull** parar la ruta **Per** descansar un moment abans de continuar	Mentre l'usuari està caminant, pot fer click al botó inferior dret per parar la ruta si necessita descansar o qualsevol altre motiu. El temps es pararà i li apareixerà un botó al mig de la pantalla en forma de "Play" perquè pugui reprendre quan vulgui. Igual que el botó inferior dret canviarà el text a **"Reanudar ruta"**.
US19 Continuar ruta	**Com** Usuari registrat i	Fent click al botó **"Play"** o al botó **"Continuar ruta"** el cronòmetre es tornarà a

	loguejat (o invitat) **Vull** reanudar la ruta **Per** poder continuar en el punt on o havia parat	posar en marxa, el **"Play"** desapareix i torna a canviar el text a **"Pausar ruta"**. D'aquesta forma l'usuari podrà seguir caminant.
US20 Sortir de la ruta	**Com** Usuari registrat i loguejat (o invitat) **Vull** sortir de la ruta **Perquè** ja he acabat de caminar per la ruta	Un cop l'usuari decideix que no vol caminar més farà click al botó inferior esquerre "Sortir de la ruta" i veurà un modal preguntant-li si està segur que vol sortir de la ruta. En cas que accepti, tornarà a la vista principal del mapa. Si prem cancel·lar, continuarà la ruta actual.
US21 Mostrar discalimer	**Com** usuari sense loguejar **Vull** veure un avís **Per** saber de quines coses l'app no es fa responsable sobre les rutes	A l'usuari se li mostrarà un **"discalimer**" avisant que l'aplicació no es fa càrrec de l'estat de les rutes ni dels inconvenients que en elles pugui haver-hi. Un cop entri a l'app li apareix el modal amb aquesta informació, on haurà de pulsar **"Ok"** indicant que ha llegit l'avís.

Figura 2: User stories creades

Totes aquestes US estan marcades amb prioritat de l'1 al 5. De tal manera que es prioritza les que es considerava que tenen un major impacte en el desenvolupament inicial de l'aplicació.

4. Planificació

Per tal de portar un procés de desenvolupament ordenat i on totes les parts tinguin clar el que es faria, es van crear User Stories amb tots els requisits que havia de complir l'aplicació. D'aquesta forma, tant el desenvolupador com el client (equip de la Dra. Maria Grau) estàvem alineats en els objectius plantejats i com es duien a terme.

4.1. Metodologia àgil Scrum

Durant el desenvolupament del projecte s'ha utilitzat la metodologia àgil Scrum que utilitza un sistema de Sprints. Aquest sistema està plantejat amb l'objectiu de poder organitzar i dividir les tasques que s'han d'implementar en el projecte. En el meu cas, cada Sprint tenia una durada de 2-3 o 4 setmanes on al final s'organitzava una reunió amb les tutores on ensenyava tot el que havia avançat i quines coses faltaven per fer. Aquestes reunions servien per anar veient de forma més visual com estava avançant l'aplicació i si feia falta modificar alguna de les idees inicials.

Per fer-ho de forma ordenada es va separar el temps de desenvolupament en diferents Sprints i al final de cada un s'havien d'intentar tenir acabades les US parlades a l'inici amb les tutores. Per tant, l'organització en Sprints va quedar de la següent forma (Figura 3):

Nº Sprint	Data inici	Data final
1	5/10/22	19/10/22
2	19/10/22	2/11/22
3	2/11/22	23/11/22
4	23/11/22	12/12/22
5	12/12/22	28/12/22

Figura 3: Nombre de Sprints fets i dates

- Sprint 1:
  En aquest primer sprint sobretot el que es va fer va ser deixar clar tots els requisits que havia de complir l'aplicació i fer un disseny de l'app amb totes les pantalles i funcionalitats visuals que havia de tenir. Tot i que cap al final del desenvolupament es va decidir de canviar una mica el format de l'app.
  Per tant, primer de tot es va presentar un disseny fet amb l'eina Figma per a veure de forma visual com es duria a terme l'aplicació i que totes parts tinguéssim present la mateixa idea, tal com es veu a les figures 4 i 5.

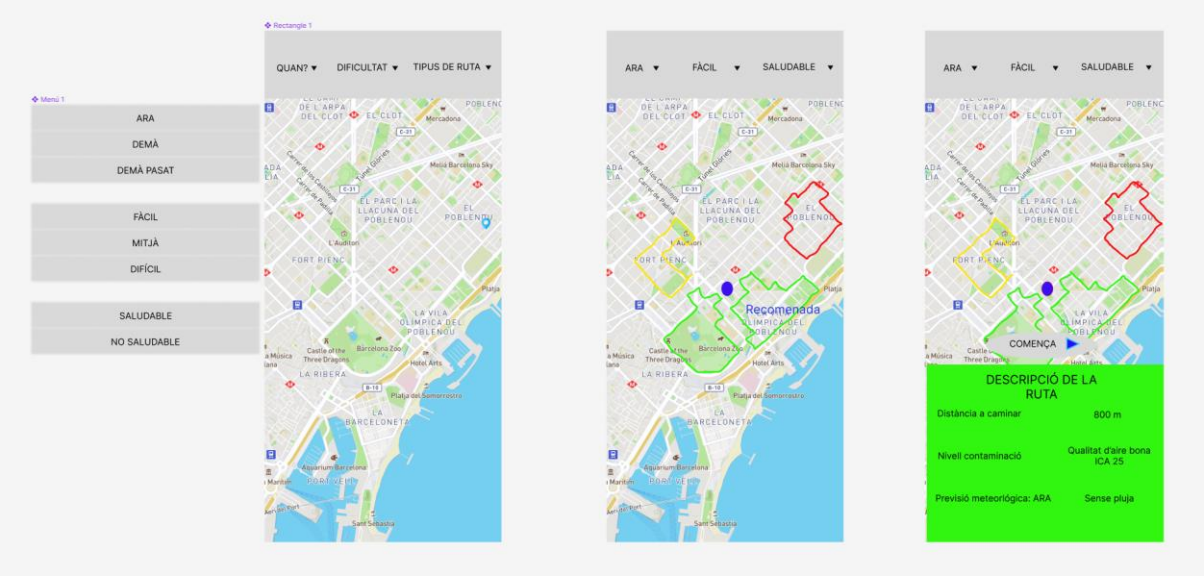


Figura 4: Disseny inicial del mapa i ruta sense contaminació

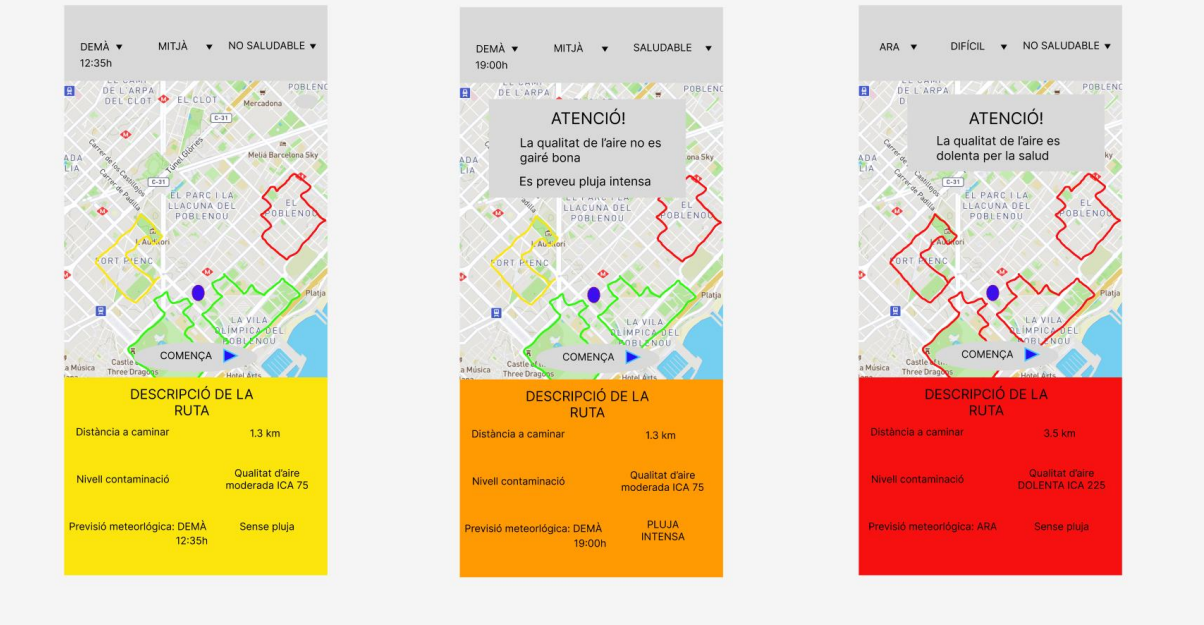


Figura 5: Disseny inicial del mapa i rutes amb nivells de contaminació

A continuació, es va preparar l'entorn de treball per realitzar l'aplicació amb React Native com ja he comentat i començar amb les següents User Stories que es consideraven més prioritàries a l'inici del desenvolupament (Figura 6):

Nº User Story	Títol
US1	Sign Up
US2	Log In

US3	Entrar com invitat
US6	Visualitzar mapes sense rutes

Figura 6: User stories del Sprint 1

Tot i que no es van poder acabar del tot la US1, US2. Ja que inicialment el que es va intentar va ser tenir tota la navegació de l'aplicació entre pantalles, però sense funcionalitats, i per això el Sign Up i Log In estaven falsejats, ja que faltava tota la part del backend per fer el control d'usuaris i dades. Només es van crear aquests botons i pantalles de forma visual.
La US6 sí que es va completar, permetent la visualització del mapa i geolocalització de l'usuari.

- Sprint 2:

  Ja en el segon sprint es van acabar les US que venien de l'anterior sprint (Figura 7). Aquest sprint va estar més centrat a preparar l'entorn de treball del backend amb Flask i en la creació de la base de dades. Per tant, es van acabar la US1 i US2 ara sí amb dades reals i fent un control de les dades introduïdes.

Nº User Story	**Títol**
US1	Sign Up
US2	Log In
US3	Entrar com invitat
US4	Log Out
US5	Modificar perfil
US7	Filtrar rutes

Figura 7: User stories del Sprint 2

Amb la base de dades ja creada i els endpoints de l'API REST també encaminats, es van poder afegir las US4 i US5 per la gestió d'usuaris.
També es va fer la US7 afegint els dropdowns pel filtratge de rutes a mode visual.

- Sprint 3:

  En aquest sprint es va començar a treballar en la visualització de les rutes en el mapa. Primer es va començar per la de "Proposta inicial" i "per defecte", ja que són les més bàsiques.
  També es va treballar en la part d'"Informació de la ruta" un cop les rutes es van fer seleccionables. En fer aquesta part, es va començar a treballar en l'obtenció de les dades de contaminació i meteorologia que no es va poder acabar fins al següent sprint. En la figura 8 es veuen les US fetes:

Nº User Story	Títol
US8	Veure rutes per defecte
US9	Veure rutes PROPOSTA INICIAL
US12	Color ruta
US13	Veure “Informació de la ruta”
US14	Tancar “Informació de la ruta”

Figura 8: User stories del Sprint 3

- Sprint 4:

Aquest sprint era una mica més llarg que els anteriors, i per això és on més funcionalitats es van afegir a l’aplicació. Es van implementar la resta de tipus de rutes que faltaven i es va afegir la UI de la ruta un cop has començat a caminar. Es va acabar la part d'“Informació de la ruta” ara sí obtenint totes les dades de contaminació i meteorologia.

Nº User Story	Títol
US10	Veure rutes PROPOSTA VERDA
US11	Veure rutes COMENÇAR AQUÍ
US13	Veure “Informació de la ruta”
US16	Visualitzar advertències
US17	Començar ruta
US18	Pausar ruta
US19	Continuar ruta
US20	Sortir de la ruta

Figura 8: User stories del Sprint 4

- Sprint 5:

En aquest últim sprint, es van acabar de fer les User stories per veure el tutorial i el disclaimer i es va acabar la de visualització d'advertències quan els nivells de contaminació són elevats (Figura 9).

Nº User Story	Títol
US15	Veure tutorial
US16	Visualitzar advertències

US21	Mostrar disclaimer

Figura 9: User stories del Sprint 5

Però, principalment aquest sprint es va dedicar a acabar d'arreglar alguns bugs que van sortir en el moment de fer proves i deixar llesta la part pràctica d'aquest Treball final de grau.

Aquests sprints corresponen al temps de desenvolupament del codi, però també es va invertir molt de temps en l'estudi de les tecnologies, sobretot amb React Native una plataforma que no havia utilitzat i a la que m'havia d'adaptar. Pel que durant les 2 setmanes anteriors a començar els sprints i durant una gran part del primer sprint es va dedicar a l'estudi i recopilació d'informació de les tecnologies que s'anaven a utilitzar.
A posteriori, un cop acabats els sprints i totes les User Stories plantejades a l'inici del projecte es va dedicar el temps restant a l'escriptura d'aquesta memòria, recopilant punt per punt tot el que s'havia fet en l'etapa de desenvolupament.

4.2. Kanban

També es va utilitzar la metodologia Kanban durant el desenvolupament. El que es va fer va ser dividir cada User Story en diferents tasques més individualitzades organitzades en Tauler Kanban al repositori de Github del treball. Aquesta metodologia permet visualitzar i fer un seguiment del progrés dels sprints d'una forma més detallada i ajudava a limitar el treball en progrés. Ja que a vegades una User story era molt extensa i durava més d'un sprint.

El tauler Kanban tenia les següents columnes tal com es pot veure a la Figura 10:

- **To Do:** Llistat de tasques que queden per fer.
- **In Progress:** Llistat de tasques que s'estan fent actualment
- **MR - DEV:** Tasques acabades i fusionades a l'entorn de DEV, on és testejant amb la resta de codi que ja estava acabat.
- **MR - STG:** Tasques fetes i fusionades a l'entorn de STG, on estan totes les tasques ja provades i la versió de codi està llesta per ensenyar a les tutores.
- **Deployed:** Tasques que estan en l'entorn final PRO i han estat desplegades després del vist bo de les tutores.

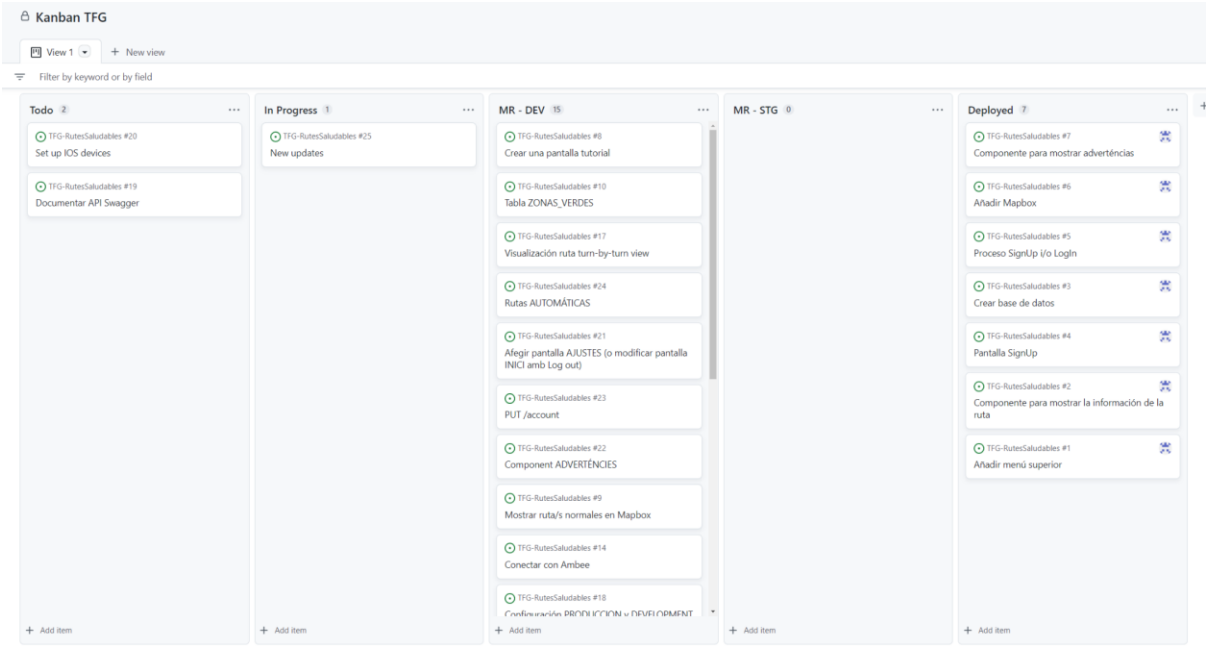


Figura 10: Exemple del meu tauler kanban ja cap al final del projecte

4.3. Diagrama de Gantt

Un diagrama de Gantt és una eina útil per planificar projectes. Proporciona una vista general de les tasques programades, totes les parts implicades sabran quines tasques s'han de completar i en quina data.

En la següent figura 11 es veu el diagrama format per totes les User Stories fetes a cada Sprint per tal de veure millor el temps que s'ha utilitzat per cadascuna d'elles.

User Story	Data inici	Duració (en días)	Data final
US1	5/10/2022	28	2/11/2022
US2	5/10/2022	28	2/11/2022
US3	5/10/2022	28	2/11/2022
US4	19/10/2022	14	2/11/2022
US5	19/10/2022	14	2/11/2022
US6	5/10/2022	14	19/10/2022
US7	19/10/2022	14	2/11/2022
US8	2/11/2022	21	23/11/2022
US9	2/11/2022	21	23/11/2022
US10	23/11/2022	19	12/12/2022
US11	23/11/2022	19	12/12/2022
US12	2/11/2022	21	23/11/2022
US13	2/11/2022	21	12/12/2022
US14	2/11/2022	21	23/11/2022
US15	12/12/2022	16	28/12/2022
US16	23/11/2022	35	28/12/2022
US17	23/11/2022	19	12/12/2022
US18	23/11/2022	19	12/12/2022
US19	23/11/2022	19	12/12/2022
US20	23/11/2022	19	12/12/2022
US21	12/12/2022	16	28/12/2022

Inici projecte	5/10/2022
Final Projecte	28/12/2022

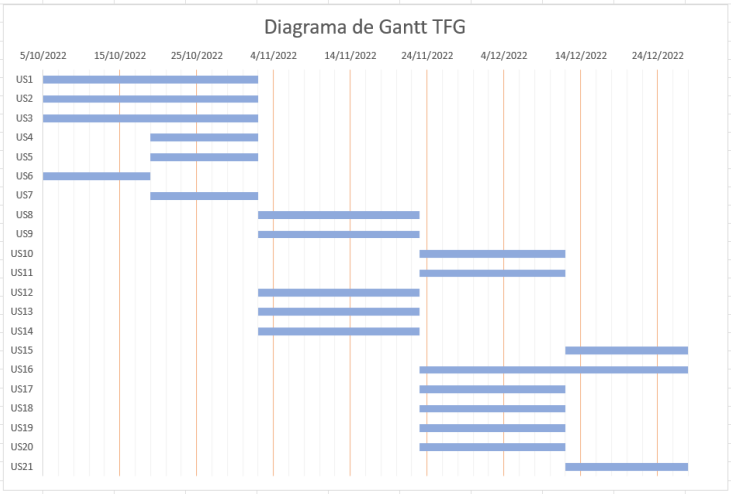


Figura 11: Diagrama de Gantt

5. Cost del projecte

En aquest apartat es realitza una aproximació dels costos econòmics que suposaria fer aquest treball a nivell empresarial.

Durant el desenvolupament de l'aplicació totes les tecnologies han sigut amb plans de dades reduïts, i per això el cost és molt menor, però en el moment de voler posar en producció aquesta aplicació els plans haurien de millorar-se, ja que s'esperen moltes més peticions.

- Development:

    - **Base de dades RDS Amazon ->** S'ha utilitzat una instància petita per guardar les dades de les rutes i usuaris. Actualment, té un cost de **0 € al mes**.

    - **Host de l'API a Heroku ->** Pla **"Eco dynos"** per desenvolupament i proves, té un cost de **5 € al mes**.

    - **Mapbox ->** Servei gratuït.

    - **API contaminació (Ambee) ->** Límit de **500 peticions** amb el pla gratuït de 30 dies.

    - **API meteorologia (Meteocat) ->** Es va sol·licitar accés al ***Meteocat*** per a l'accés com a estudiant. Em van donar un límit de **100 peticions al mes**.

    Pel que el procés de desenvolupament té un cost de 5€ al mes, a condició que es compleixin els límits d'ús de les APIs. Que en entorns de prova poden estar en "mock" per així no gastar peticions de més.

- Production:
    Això dependrà de l'ús que se li doni a l'aplicació. Els càlculs són una aproximació imaginant un ús de l'app d'unes 500 persones al dia.

    - **Base de dades RDS Amazon ->** Per la posada en producció, el cost seria molt semblant, ja que les dades de les rutes són les mateixes i és el volum principal de dades, tot i que es poden anar afegint algunes més. El que canviaria serien els usuaris que tampoc ocupen molt, per la qual cosa aproximadament per 500 tindria un cost de **20 € al mes**.

    - **Host de l'API a Heroku ->** S'hauria d'augmentar al pla de dades **"Production"** amb un cost de **25 € al mes**.

    - **Mapbox ->** Servei gratuït.

- **API contaminació (Ambee) ->** La web de l'API no proporciona informació del cost, s'ha de contactar amb ells per explicar la intenció de l'aplicació i que et facin un pressupost. Aproximadament de més de **500 € al mes**, ja que es una API privada i s'ha de consultar múltiples vegades per al càlcul de la contaminació de les rutes.

- **API meteorologia (meteocat) ->** Sempre que es consideri un projecte estudiantil, l'accés es permet gratuïtament. Però si es fa servir de forma professional té un cost de **79 € al mes** per a un límit de 100.000 peticions.

Finalment, ens quedaria un preu de 854 € al mes per mantenir els serveis que utilitza l'aplicació en producció per a uns 500 usuaris.

A més, s'hauria de sumar el preu per pujar l'aplicació al Google Play, on s'ha de tenir un compte de desenvolupador que té un cost d'uns 25 € que només s'ha de pagar un cop. Per altra part, en el cas d'IOS mitjançant l'App Store, es necessita un compte de desenvolupador que té un cost de 99 € anuals.

I per últim, el cost humà que comporta realitzar aquest projecte, tenint en compte que, tot i que com a treballador seria de Junior (8 € l'hora), el treball fet implicaria a més tipus de treballadors, per tant, estimem uns 15 euros l'hora aproximadament. Tenint en compte que el treball està estimat en 14 setmanes (es va començar a finals d'octubre) i sense comptar el temps per la memòria, amb un treball diari de 6 hores en 3.75 dies, el cost total humà que hi ha hagut en el desenvolupament és d'uns 4050 euros.

A continuació la Figura 12 amb una taula resum de tots els costos:

Tipus de despesa	Preu
Base de dades	20 €/mes
APIs	604 €/mes
Deploys	115 € primer any, 99 € la resta d'anys -> 9,6 €/mes
Cost humà	4050 € en total -> 337,5 €/mes
Total	**638,6 €/mes**

Figura 12: Cost del deploy de l'aplicació.

6. Disseny i implementació

Com s'ha esmentat en el punt 2.1 es va decidir utilitzar la tecnologia React Native. Això, com veurem ara, determina l'estructura que tindrà el nostre projecte. En aquest apartat s'explicarà l'arquitectura del projecte, amb els models de la base de dades, l'estructura de les dades, els diagrames de classes i els patrons utilitzats.

6.1. Arquitectura frontend

Per tal d'estructurar el codi a React Native per fer la part del frontend de l'aplicació, s'ha seguit l'arquitectura recomanada generalment dins del desenvolupament mòbil amb React Native. Aquesta organització consisteix a separar els fitxers en tres capes (Figura 13):

1. **UI o Capa de presentació:** Representa tots els components o elements de la interfície d'usuari amb els quals interactua l'usuari, com ara botons, finestres emergents, text, etc.

2. **Capa lògica:** Responsable de mantenir la lògica de l'aplicació. També és responsable de tots els esdeveniments i de gestionar les interaccions amb la capa de presentació.
3. **Capa API:** Responsable de totes les interaccions back-end. Aquí és on l'aplicació fa trucades d'API al nostre backend i servidor de bases de dades i als serveis web externs per la informació meteorològica i contaminació.

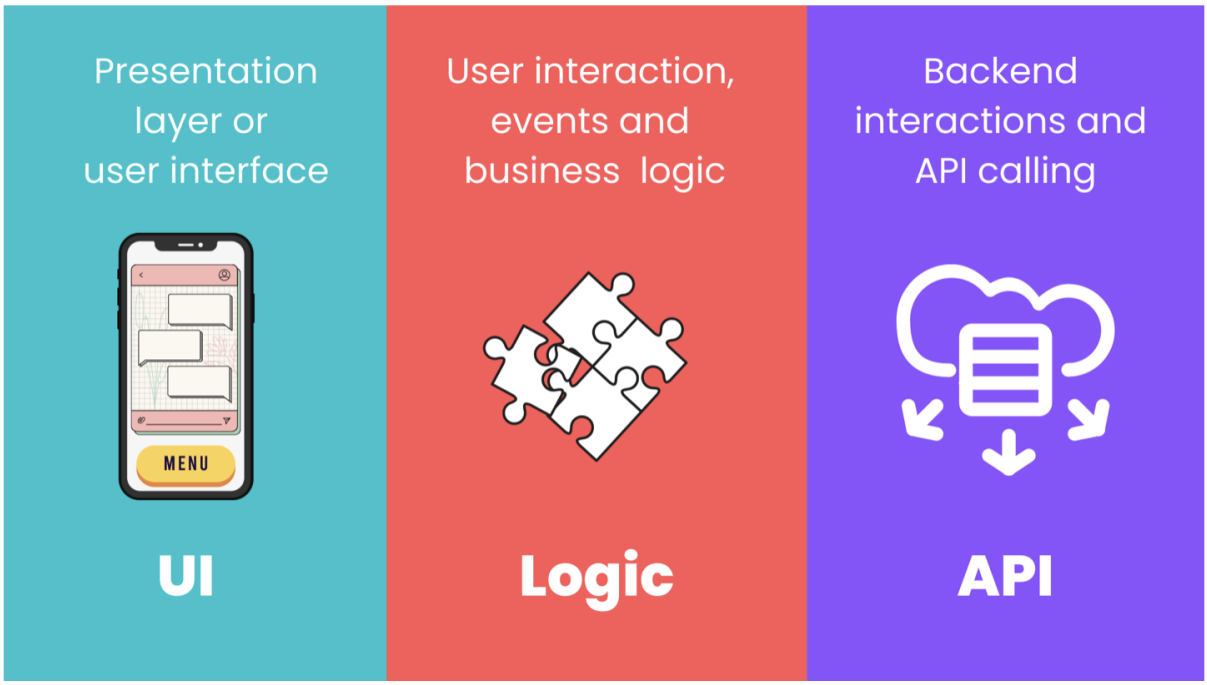


Figura 13: Patró de programació seguit per la part del frontend

Per realitzar aquesta separació en tres capes, s'ha utilitzat la següent organització de les carpetes del codi (Figura 14):

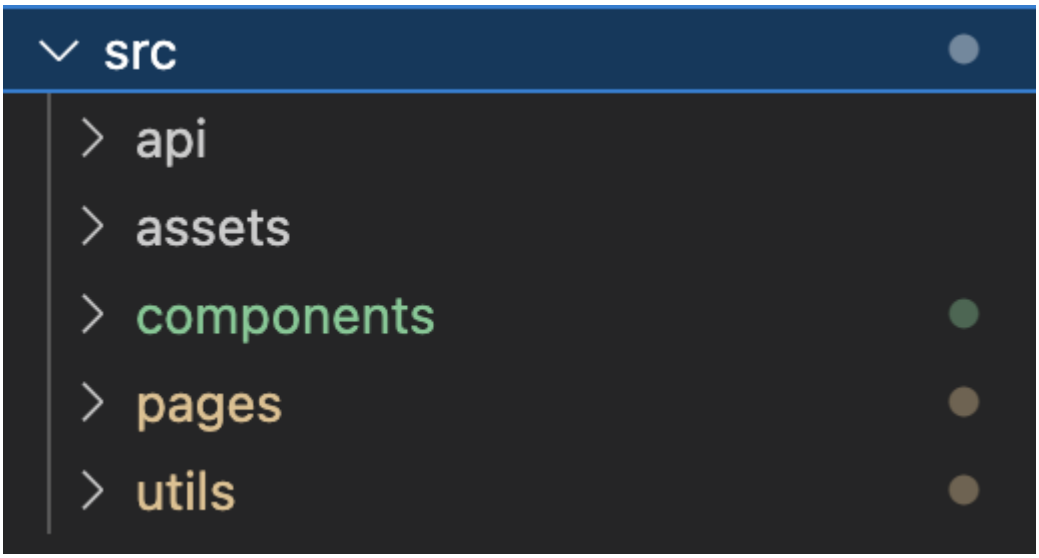


Figura 14: Estructura de carpetas a React Native

1. **Components:** Conté tots els fitxers i carpetes per a components reutilitzables.

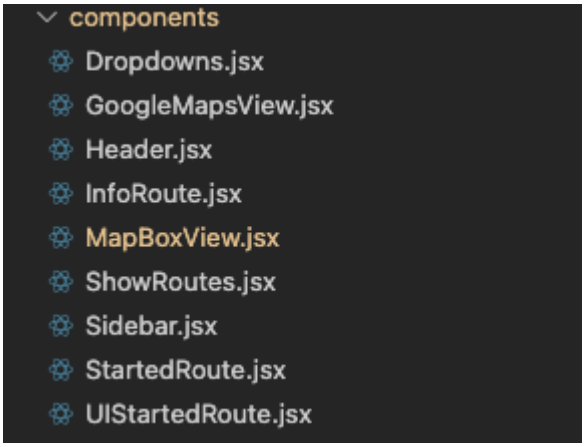


Figura 15: Components creats

2. **Assets:** Carpeta on guardar arxius estàtics com imatges i/o icones.
3. **Pages/Screens:** Conté les pàgines o pantalles de l'aplicació com ara HomeScreen, LoginScreen, etc.

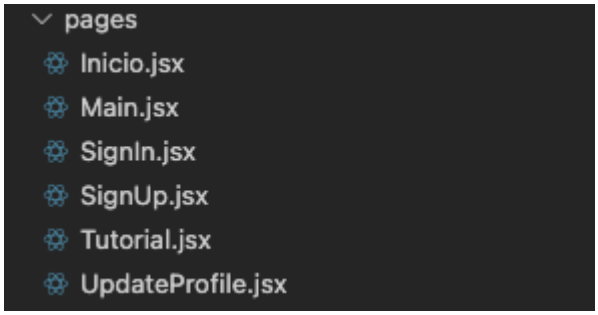


Figura 16: Pàgines creades

Com es veu a la imatge, aquí guardo les diferents pantalles per on l'usuari pot navegar i que conjuntament amb els components de l'altra carpeta acaben de formar tota la capa de presentació de l'aplicació.

4. **Api:** On estan implementades les trucades d'API relacionades amb les funcionalitats d'inici de sessió, registre, dades de contaminació i meteorologia, etc.

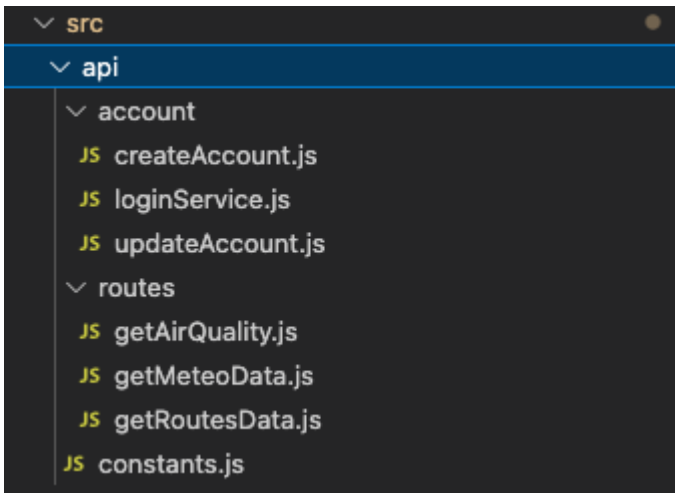


Figura 17: Fitxers encarregats de la comunicació amb l'API

Aquesta carpeta és l'encarregada d'implementar la Capa d'API. Com es veu he separat les trucades als diferents serveis per funcionalitat. Una carpeta s'encarrega de tot el relacionat amb la informació de l'usuari i l'altre en tot el relacionat amb les rutes, com l'obtenció de les coordenades per mostrar les rutes al mapa o l'obtenció de la contaminació i meteorologia.

A més, d'un fitxer "constants.js" on guardar les constants que s'utilitzen com les BASE_URLs a les que apunten els serveis i els API_KEY que demanen alguns dels endpoints per a poder retornar la informació sol·licitada.

5. **Utils:** Carpeta on estan tots els fitxers "helpers" que s'encarreguen de la lògica d'aplicació i connecten la capa de presentació o UI amb la capa d'API.

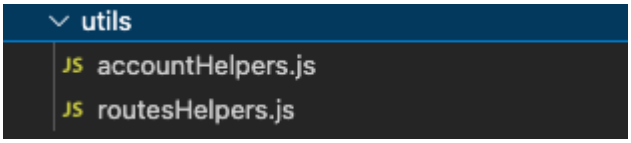


Figura 18: Carpeta utils

Concretament, tinc un fitxer accountHelpers.js per la lògica de dades relacionada amb l'usuari i la connexió amb els serveis de createAccount, loginService i updateAccount que es veu en la figura anterior. I un routeHelpers.js que s'encarrega de la lògica per calcular tot allò que necessiten els components de Mapbox per mostrar les rutes i preparar els paràmetres a enviar als diferents endpoints per les dades de les rutes.,

En resum, un dels grans avantatges de l'estructura del projecte seguida és la reutilització. Sempre es pot traduir aquesta estructura a qualsevol altra aplicació de frontend. Es podria reutilitzar directament alguns dels fitxers i carpetes creats i al ser un projecte que està pensat per fer en diferents fases i amb diferents alumnes crec que ha sigut l'opció més correcta.

Les constants, útils i carpetes API es poden utilitzar directament en qualsevol altre projecte similar que es volgués crear. A més, hi ha més abstracció entre les diferents capes de l'aplicació. Això permet separar les preocupacions mentre es crea l'aplicació i tenir una major abstracció i independència de les diferents parts del codi el que millora la resiliència del codi.

6.1.1. Programació amb React Native

Tal com he explicat en els anteriors apartats React Native és un framework Javascript per crear aplicacions reals natives per a iOS i Android, basat en la llibreria de JavaScript React per a la creació de components visuals, canviant el propòsit dels mateixos per, en lloc de ser executats en navegador, córrer directament sobre les plataformes mòbils natives, en aquest cas iOS i Andorid. És a dir, en lloc de desenvolupar una aplicació web híbrida o en HTML5, el que obtens al final com a resultat és una aplicació real nativa, indistingible de la qual es pot desenvolupar en codi natiu amb Objective-C o Java.

React Native utilitza el mateix paradigma fonamental de construcció de blocs d'UI (components visuals amb què interacciona l'usuari) que les aplicacions natives reals d'Android i iOS, però gestiona la interacció entre aquests utilitzant les capacitats de JavaScript i React. Però, com s'utilitzen i programen aquests blocs o components?

6.1.1.1. Programació amb components

React utilitza tots els elements proporcionats per l'especificació HTML, com , , etc., mentre que React Native segueix la mateixa lògica per la vista, però introdueix nous components per utilitzar-los com <view> i <text>. És a dir, l'estructura és molt semblant a la del desenvolupament web, però utilitzant una sèrie

de components predissenyats que no es poden trobar en un disseny web tradicional. Hi ha components base de la llibreria pròpia de React Native, però també es poden utilitzar components de tercers instal·lant prèviament les llibreries amb npm. Aquest son els components més utilitzats i considerats “Components base”, sobre els que es fonamenta qualsevol desenvolupament:

- **StyleSheet*:*** aquest és el component on afegirem els estils del component. Ho podríem assimilar als fitxers CSS del desenvolupament web.
- **View:** aquest és el component més bàsic sobre el qual hereten la majoria dels altres components. Ho podem equiparar a un HTML .
- **Text:** aquest component és on mostrarem textos.
- **Image:** aquest component serveix per mostrar imatges
- **TextInput:** la funció d'aquest component és proporcionar un espai a l'usuari perquè pugui introduir text mitjançant el teclat del dispositiu.
- **ScrollView:** aquest component permet establir un contenidor on es podran emmagatzemar diversos components que es poden anar desplaçant a la pantalla.

Per tant, un exemple de programació bàsic utilitzant aquests components quedaria de la següent manera:

```
<ScrollView>
    <View>
        <Text>Exemple components</Text>
        <TextInput label=”Introdueix el nom”/>
        <TextInput label=”Introdueix contrasenya” />

        <Button color=”#fff” title=”Iniciar sessió” onPress={() =>
                navigation.navigate(“MainPage”)} />
    </View>
        </ScrollView>
```

Com es veu, el component View és l’element principal per a l’organització de la vista, però s’afegeix per sobre el component ScrollView per al cas que sigui necessari fer scroll a la pantalla del mòbil, ja que no hi cap tot. Després, dins del View tenim els components per crear un formulari bàsic d’inici de sessió introduint nom i contrasenya i per últim un botó que s’encarrega de redirigir a la següent pàgina.

Tot això és un exemple molt bàsic, però serveix per veure l’estructura dels components amb React Native i com funcionen. Una cosa molt important a conèixer són els ***“props”,*** que són els atributs que tenen els components. Cadascun pot tenir diferents props que ofereixen diferents funcionalitats, per exemple, el TextInput té com a prop “***label***” on s'indica el placeholder que hi haurà dins l’element, en canvi,

Button té com a props "***color***" on indicar el color del botó, "***title***" on es posa el text que hi haurà dins del botó i "***onPress***" on s'indica que es vol fer quan es faci click sobre el botó.

Com dic, aquests són exemples bàsics i cada component pot tenir molts més props que s'han d'anar consultant a la documentació oficial de React Native o a la documentació de tercers si s'utilitzen components externs.

6.1.1.2. React Native Caché

En una aplicació que consumeix dades d'una API externa, és probable que es vulgui emmagatzemar algunes d'aquestes dades a la memòria cau al dispositiu de l'usuari, almenys temporalment. Un dels motius principals per fer-ho és reduir les sol·licituds de xarxa addicionals a l'API i oferir un cert grau de suport fora de línia als usuaris.

El funcionament de la memòria cau o caché consisteix en guardar les respostes que dona l'API després d'haver fet una primera consulta i així en la pròxima petició en comptes llançar la petició a l'API, es recupera de la memòria cau del dispositiu. D'aquesta forma es millora el rendiment de l'aplicació, ja que la memòria cau sempre serà més ràpida que les consultes a les APIs i ofereix a l'usuari la capacitat d'utilitzar certes parts de l'aplicació que necessitaven internet sense connexió.
Per al meu cas, s'ha implementat en les APIs externes que fan les consultes sobre meteorologia i contaminació de l'aire, ja que com he explicat són plans de dades gratuïts en el que tinc peticions limitades i gràcies a aquest recurs puc estalviar moltes peticions diàries i d'aquesta forma fer més proves amb l'entorn real i no amb dades falsejades per a no gastar aquest límit.

Per realitzar aquesta implementació en React Native, s'ha utilitzat el sistema d'emmagatzematge AsyncStorage de valor-clau no xifrat, asíncron, persistent i global per a l'aplicació. En la implementació dels serveis s'ha afegit la lògica per guardar a caché la resposta de l'API durant 24 hores, si aquest temps passa es torna a obtenir la dada d'API i s'actualitza la memòria cau.
La dada es guarda utilitzant el mètode ***.setItem('key', 'value')*** i es recupera amb ***.getItem('key')*** des de qualsevol punt del codi on es vulgui utilitzar. A continuació es pot veure en la Figura 18 com s'ha fet per al servei del meteocat que recupera les dades de meteorologia.

```
/**
 * Get forecast data from external API Meteocat. 3 Days forecast.
 *
 * @param {*} time day and hour
 * @returns rain value forecast
 */
export const getForecastData = async (time) => {
    const cacheIntervalInHours = 24
    const cacheExpiryTime = new Date()
    cacheExpiryTime.setHours(cacheExpiryTime.getHours() + cacheIntervalInHours)
    const lastRequest = await AsyncStorage.getItem('meteoData')

    if (lastRequest == null || lastRequest > cacheExpiryTime) {
        var rainValue = 0;
        const options = {
            method: 'GET',
            url: URL_METEO_DATA,
            headers: {'X-Api-Key': meteoToken, 'Content-type': 'application/json'}
        };
        await axios.request(options).then(async function (response) {
            rainValue = response.data.dies[time.day].variables.precipitacio.valor[time.hour].valor;
            await AsyncStorage.setItem('meteoData', rainValue)
        }).catch(function (error) {
            console.error(error.message);
        });
        console.log('Forecast: ' + rainValue)
        return rainValue;
    }

    return lastRequest;

}
```

Figura 18: Implementació servei Meteocat amb memòria cau

6.1.2. Diagrama de components

Per tant, tota aquesta arquitectura seguida per a la correcta organització del codi a React Native, deixa el següent diagrama de components de la Figura 19 on es veu de forma més gráfica la relació entre totes les parts implicades en la interfície d'usuari:

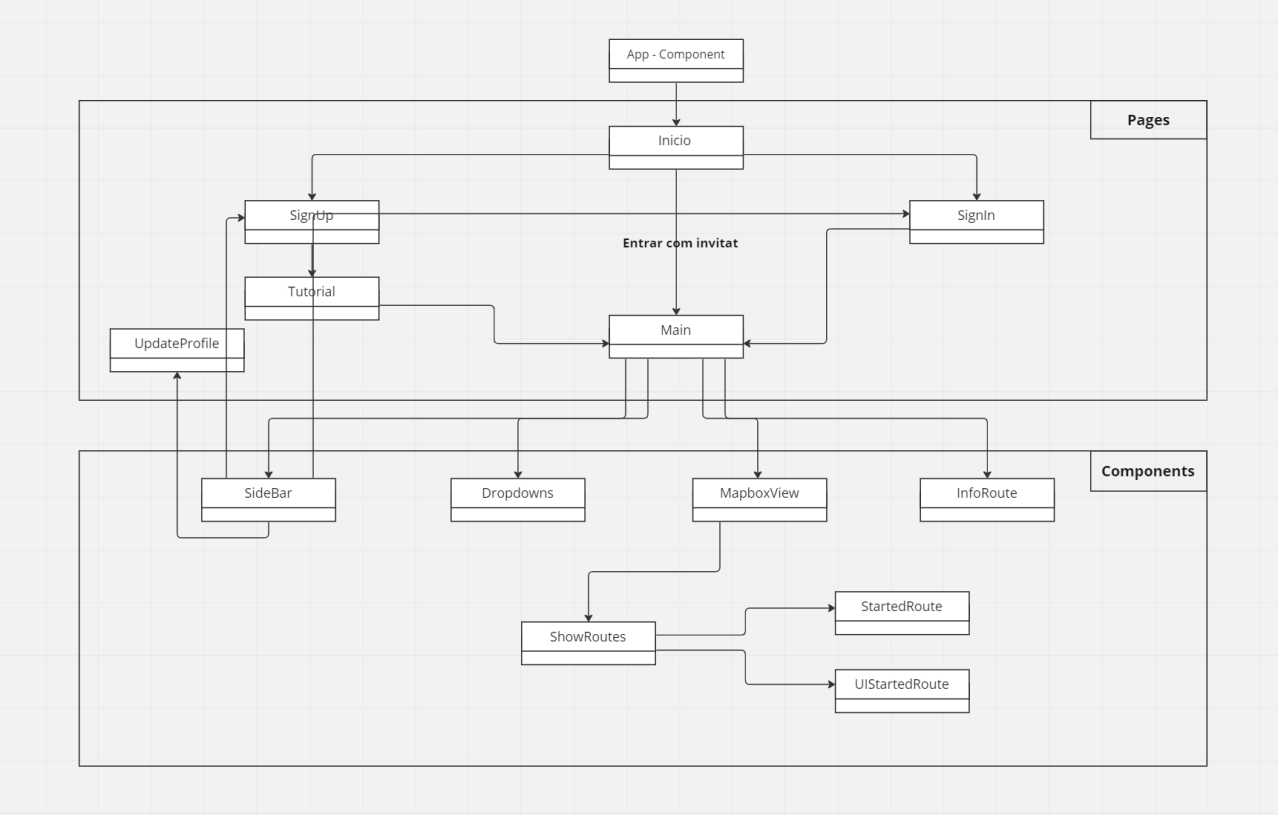


Figura 19: Diagrama de components de React Native

Tenim la part de les "Pages" que conté totes les pantalles per les quals l'usuari pot navegar:

1) **Inici:** Pantalla principal al entrar a l'app, amb les opcions de Iniciar sessió, registrarse o entrar com invitat

2) **SignUp:** Pantalla on l'usuari podrà realitzar el registre del seu compte

3) **SignIn:** Pantalla on iniciar sessió un cop l'usuari ja té la compte creada
4) **Tutorial:** Quan acaba el registre, es mostra una pantalla de tutorial on s'explica com funciona l'aplicació

5) **UpdateProfile:** Pantalla on l'usuari podrá modificar las seves dades

6) **Main:** Pantalla principal de l'aplicació que s'accedeix un cop acabat el tutorial, després d'iniciar sessió o directament des de l'inici entrant com a invitat.

A continuació, tenim tots els “components” que ajuden a visualitzar l’aplicació, pero que no son pantalles com a tal. Amb el conjunt de components s’acaben de montar las pantalles:

1) **SideBar:** Menú lateral on l’usuari podrà modificar el seu perfil, iniciar sessió o registrarse si encara no ho ha fet.

2) **Dropdowns:** Component amb el qual l’usuari podrà seleccionar les opcions per filtrar les rutes a buscar.

3) **MapboxView:** Component on es visualitza el mapa utilitzant Mapbox i sobre el que es pintaran les rutes.

4) **InfoRoute:** Quan se selecciona una ruta, aquest component és l’encarregat de mostrar la informació del recorregut.

5) **ShowRoutes:** Component que mostra sobre la vista del mapa totes les rutes disponibles amb un codi de colors segons contaminació i meteorologia.

6) **StartedRoute:** Visualització del mapa quan l’usuari comença el recorregut.

7) **UIStartedRoute:** Juntament amb el component anterior, quan la ruta ha començat, aquest component s’encarrega de proporcionar a l'usuari una interfície amb la qual interactuar per pausar la ruta, sortir o veure el temps que porta.

En aquest diagrama tenim la distribució dels components de les pàgines i l’ordre en el qual s’instancien. S’ha de tenir en compte que totes les relacions són 1..1, ja que aquest sistema utilitza les pàgines com un únic objecte. En canvi, els components si es poden instanciar diverses vegades, tot i que en el meu cas només hi ha una relació 1..n entre MapboxView i el ShowRoutes, el qual pot pintar diverses rutes sobre el mapa.

Com es pot veure el App-Component està fora de la carpeta de pàgines, el qual s’executa i mostra per defecte a React Native i en aquest, trobem el menú de l’aplicació que ens permet la navegació entre pàgines. La ruta inicial està assignada a la pàgina d'Inici des de la que després podrem anar a la resta de pantalles.

En l’apartat de resultats es mostrarà amb imatges la navegació completa entre pàgines i com interactuen aquests components.

6.2 Arquitectura backend

Per la part del backend, s’ha seguit una organització de codi vista durant el grau utilitzant Flask. Aquesta arquitectura consisteix a separar per un costat els models de dades encarregats de definir l’estructura dels elements amb els que treballa l'API, en el meu cas rutes i usuaris per guardar-los a la base de dades. Per un altre costat, hi ha les “routes” o “resources” encarregats de definir els endpoints que proporciona la meva API i implementar els mètodes HTTP que aquests utilitzen i que interactuen amb els models.

6.2.1. Estructura del codi

Per tant, l'estructuració del codi quedaria com es veu a la Figura 20:

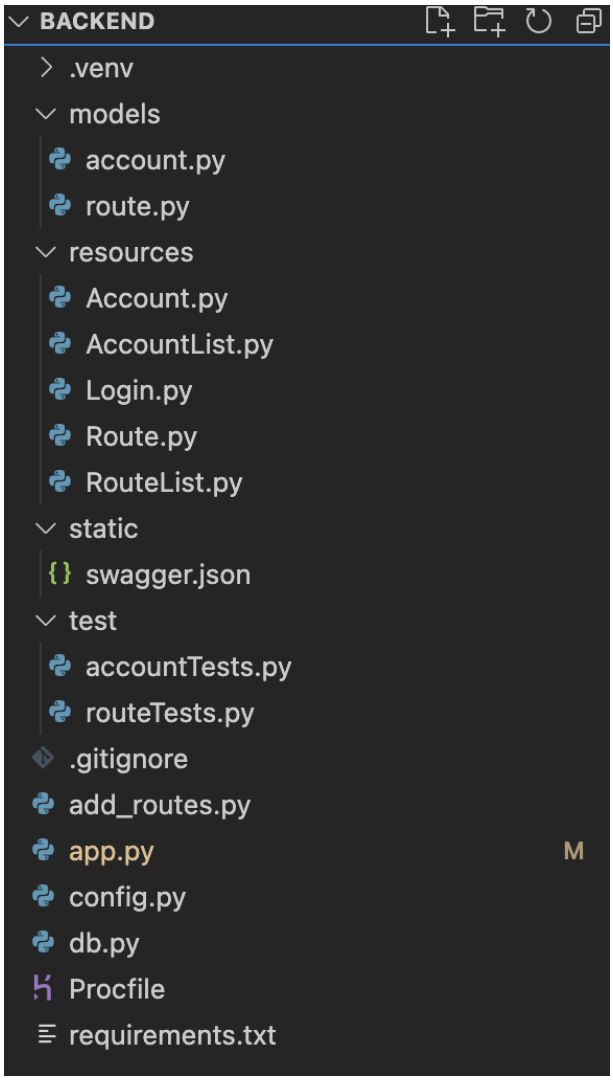


Figura 20: Estructura de fitxers del backend

Tal com explicava, primerament tenim el que correspon a tota la lògica i funcionament dels endpoints de l'API:

1) **Models:** Un model ens dona l'estructura sota la qual operarà la lògica més gran de l'aplicació, per tant, aquí és on donem un sentit a les diferents entitats amb què hem de comptar per poder exercir les nostres accions. Les dades que guardem a la nostra base de dades seran representades per una col·lecció de classes que són referides com a models de base de dades. En el meu cas, l'"account" o compte dels usuaris i les rutes.

   A continuació, es mostra el model definit (Figura 21) pels comptes dels usuaris, on guardem:

   - **id:** Identificador únic de cada usuari
   - **name:** Nom i cognoms de l'usuari
   - **email:** Correu de l'usuari. També és únic.
   - **edat:** Anys que té l'usuari

- **password:** Contrasenya amb què l'usuari s'ha registrat. Es guarda encriptada a causa de la protecció de dades.

```python
class AccountsModel(db.Model):

    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    email = db.Column(db.String(100))
    edat = db.Column(db.Integer)
    password = db.Column(db.String(400))

    def __init__(self, name, email, edat, password):
        self.name = name
        self.email = email
        self.edat = edat
        self.password = password
```

Figura 21: Model de dades dels "users"

Per últim, hi ha el model definit per les rutes del mapa (Figura 22), on guardem:

- **id:** Identificador únic de cada ruta
- **name:** Nom de la ruta
- **level:** Nivell de dificultat que té la ruta
- **activity:** Tipus d'activitat per la que está destinada la ruta. Passeig i/o córrer.
- **distance**: Distància en km del recorregut.
- **type**: Indica la forma de la ruta. Pot ser circular o d'anada i tornada.
- **coordinates**: Llistat de totes les coordenades que formen el recorregut.
- **elevation**: Camp on s'indica l'elevació de les rutes si n'hi ha.
- **health_type**: Tipus de ruta. "Normal" per les de proposta inicial i "Zones verdes" per les de Proposta verda.

```python
class RoutesModel(db.Model):

    __tablename__ = 'routes'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    level = db.Column(db.String(100))
    activity = db.Column(db.String(100))
    distance = db.Column(db.Float)
    type = db.Column(db.String(100))
    coordinates = db.Column(db.Text())
    elevation = db.Column(db.Float)
    health_type = db.Column(db.String(100))

    def __init__(self, name, level, activity, distance, type, coordinates, elevation, health_type):
        self.name = name
        self.level = level
        self.activity = activity
        self.distance = distance
        self.type = type
        self.coordinates = coordinates
        self.elevation = elevation
        self.health_type = health_type
```

Figura 22: Model de dades de les “routes”

En l’apartat 5.2.3 es pot veure el model relacional de tota la base de dades.

2) **Resources:** Els resources són els encarregats de definir els mètodes HTTP GET, POST, PUT, DELETE i com aquests interactuen amb la base de dades i retornen respostes a les sol·licituds entrants.

   Pel meu desenvolupament, en tenir els models “account” i “routes” es van crear els resources “Account” l'obtenció, creació i actualització individual dels usuaris. Per obtenir, un llistat de tots els usuaris hi ha el “AccountList” on el mètode GET retorna un json amb tots els usuaris i la seva informació. Per l’apartat dels usuaris, es va crear un últim resource “Login” encarregat de controlar l’inici de sessió dels usuaris a l’aplicació.
   La mateixa lògica es va seguir per les rutes, hi ha un “Route” per a l’obtenció i creació individual de rutes i un “RouteList” per a l’obtenció del llistat complet de rutes amb tota la seva informació.

Però, a més tenim altres carpetes que acaben de completar tota la part del backend i ajuden a complementar les parts principals:

3) **Static:** Guarda un fitxer swagger.json on s’han definit tots els endpoints i així facilitar als usuaris l’enteniment de l’API. En l’apartat 5.2.4 s’entra en detall del swagger.

4) **Test:** S’han creat tests unitaris per comprovar el correcte funcionament de l'API durant el desenvolupament. Els tests s’encarreguen de comprovar que els endpoints que ofereixen serveis als usuaris tant creant, recuperant, actualitzant com eliminant segueixen funcionant i d'igual forma recuperant les rutes de la base de dades. Cada cop que es feien tasques relacionades amb el backend, s’executaven per veure que tot segueix funcionant sense problemes.

Per fer-ho s'ha utilitzat el paquet de python ***unittest*** que ajuda a la implementació dels tests unitaris i el paquet ***requests*** per a reproduir les peticions a la meva pròpia API. S'executen els tests i veiem si hi ha algun problema amb algun endpoint o per al contrari tot ha anat correctament com a la Figura 23.

```
(.venv) albertperezcosta@MacBook-Air-de-Albert-2 backend % python3 -m unittest
..........
----------------------------------------------------------------------
Ran 10 tests in 1.614s

OK
```

Figura 23: Resultat OK dels tests unitaris sobre l'API

5) **App.py:** Fitxer principal, encarregat de començar l'execució de l'API i on es defineixen tots els endpoints. S'utilitza com a controlador entre els models i les resources.

6.2.2. Algoritmes

- **Rutes proposta inicial:**

També es va crear un fitxer "*add_routes.py"* que s'encarrega d'omplir la base de dades amb totes les rutes que es mostraran a l'aplicació. Aquestes rutes van ser extretes de l'AllTrails i de l'OpenDataBCN en formats JSON i GPX. Segons el lloc d'on estaven extrets els fitxers tenen un format de les dades diferent, i per això aquests scripts s'encarreguen de recórrer aquests fitxers i tractar les dades per guardar-les en el mateix format que jo necessitava dins de la base de dades.

```python
#RUTAS NORMALES
def db_circular_routes(app, db):

    path = "/Users/albertperezcosta/Desktop/Rutas"
    dirct = os.listdir(path)

    for filename in dirct:
        if filename.endswith('.js'):
            f = open(path + '/' + filename)
            data = json.load(f)
            coordinates, elevation = format_coordinates(data['data']['trackData'][0])
            distance = calculate_distance(data['data']['trackData'][0])

            if (distance <= 2.5):
                new_route = RoutesModel(filename.split('.')[0], "FACIL", "PASEO", distance, "CIRCULAR",
                coordinates, elevation, "NORMAL")
            if (distance > 2.5 and distance <= 4.5):
                new_route = RoutesModel(filename.split('.')[0], "MITJA", "PASEO", distance, "CIRCULAR",
                coordinates, elevation, "NORMAL")
            if (distance > 4.5):
                new_route = RoutesModel(filename.split('.')[0], "DIFICIL", "PASEO", distance, "CIRCULAR",
                coordinates, elevation, "NORMAL")

            with app.app_context():
                if not RoutesModel.find_by_name(new_route.name): #Para no añadir rutas repetidas
                    db.session.add(new_route)
                    db.session.commit()
                    db.session.close()
```

Figura 24: Script per la càrrega de rutes "Proposta inicial"

En la Figura 24 es veu com es llegeix d'una carpeta de l'ordinador tots els fitxers que contenen rutes i es formateja la informació de forma que em quedo només amb les coordenades com a mi m'interessa (x,y) i creo una llista amb totes elles. També es fa el càlcul de la distància entre el llistat de coordenades i es guarda la ruta en base de dades indicant la dificultat segons la distància que té cada ruta.

- **Rutes proposta verda:**

De la mateixa forma es fa per les rutes de "Proposta verda" (Figura 25), que es van extreure de l'OpenDataBCN amb format .gpx, per la qual cosa vaig formatejar les coordenades per guardar les de la mateixa forma que les altres rutes. Igual que amb les anteriors, es va definir la dificultat de la ruta segons la distància que tenen.
A més, per ambdós casos, es va tenir en compte que els serveis de mapbox només deixen dibuixar rutes al mapa que tinguin 25 waypoints. Els waypoints, són les coordenades per on passa la ruta, pel que cadascuna ha de tenir un límit de 25 parells de coordenades. Per solucionar aquest límit i que la ruta es vegi completa, es van agafant les coordenades de la ruta cada "x" valors depenent del nombre de coordenades que hi ha. Per exemple per una ruta amb 100 coordenades, es guardarà a la base de dades cada 4 coordenades, deixant així la ruta en 25 waypoints.

```python
#RUTAS ZONAS VERDES
def db_zonas_verdas_routes(app, db):
    path = "/Users/albertperezcosta/Desktop/Rutas/ZONAS VERDES/pev_parcs_od_1.gpx"
    gpx_file = open(path, 'r')
    gpx = gpxpy.parse(gpx_file)

    count = 0
    for track in gpx.tracks:
        points = ''
        name = ''
        distance = 0
        waypoints = 0
        last_point = []
        first_point = []
        for extensions in track.extensions:
            if extensions.tag == '{http://osgeo.org/gdal}nom':
                name = extensions.text
        for segment in track.segments:
            for point in segment.points:
                waypoints += 1
                #Guardamos el punto inicial para cuando detectemos que llevamos 24 waypoints, ponerlo el ultimo para hacer que la ruta sea circular
                if not first_point:
                    first_point = point
                # Guardamos cada vez el punto anterior para poder hacer el calculo de la distancia
                if last_point and waypoints < 27:
                    distance += geopy.distance.geodesic(last_point , [point.longitude, point.latitude]).km
                if waypoints < 25:
                    points += '[' + str(point.longitude) + ',' + str(point.latitude) + '],' #Para mantener el formato con las rutas normales guardamos las coordenadas en fromato string
                    last_point = [point.longitude, point.latitude]
                elif waypoints == 25:
                    points += '[' + str(first_point.longitude) + ',' + str(first_point.latitude) + ']'  #Añadimos el primer punto al final y queda la ruta circular y terminamos
                    last_point = [point.longitude, point.latitude]

        #Acabamos de formatear las coordenadas
        coordinates = '[' + points + ']'
        coordinates = coordinates.replace(",]", "]")

        #Para no añadir demasidas rutas cortas, añadiremos solo las que tengan > 1.5km
        if (distance >= 1 and distance <= 1.5):
            new_route = RoutesModel(name, "FACIL", "PASEO", distance, "CIRCULAR",
            coordinates, 0, "ZONES VERDES")
            print(name)
            print(distance)
            with app.app_context():
                if not RoutesModel.find_by_name(new_route.name): #Para no añadir rutas repetidas
                    db.session.add(new_route)
                    db.session.commit()
                    db.session.close()

        if (distance > 1.5 and distance <= 2): #Para no añadir demasidas rutas cortas, añadiremos solo las que tengan > 1.5km
            new_route = RoutesModel(name, "MITJA", "PASEO", distance, "CIRCULAR",
            coordinates, 0, "ZONES VERDES")
            print(name)
            print(distance)
            with app.app_context():
                if not RoutesModel.find_by_name(new_route.name): #Para no añadir rutas repetidas
                    db.session.add(new_route)
                    db.session.commit()
                    db.session.close()

        if (distance > 2 ): #Para no añadir demasidas rutas cortas, añadiremos solo las que tengan > 1.5km
            new_route = RoutesModel(name, "DIFICIL", "PASEO", distance, "CIRCULAR",
            coordinates, 0, "ZONES VERDES")
            print(name)
            print(distance)
            with app.app_context():
                if not RoutesModel.find_by_name(new_route.name): #Para no añadir rutas repetidas
                    db.session.add(new_route)
                    db.session.commit()
                    db.session.close()

    print("Guardando rutas ZONAS VERDES")
```

Figura 25: Script per la càrrega de rutes “Proposta verda”

- **Rutes proposta automàtica:**

També, tenir en compte que l’usuari té l’opció de generar una ruta automàtica des del punt on està. Aquestes rutes no es guarden a base de dades ja segons la posició de l’usuari pot canviar cada cop que busqui. Pel que hi ha un script a part, encarregat de definir aquesta ruta automàtica. Consisteix a crear una ruta on la primera i última coordenada sigui la posició de l’usuari i el fa passar per una de les zones verdes que sí que tenim guardades. Segons la dificultat seleccionada l’aplicació el farà anar a una zona verda més propera o més llunyana, sempre mantenint la resta de funcionalitats de l’app com són la predicció meteorològica i els nivells de contaminació.

Per últim, hi ha diversos arxius principalment de configuració de l'entorn que són necessaris per al deployment de l'api a Heroku. El Procfile que indica a Heroku quin procés s'utilitza, en el meu cas "web" i quin script executar, en el meu cas "app", ja que és el fitxer principal. Després tenim el requirements.txt on s'indiquen tots els paquets de python que utilitza l'API i que són necessaris per al correcte funcionament.

6.2.3. Base de dades

Com ja he explicat en l'apartat dels models, s'han definit dos models de dades per guardar informació sobre els usuaris i les rutes. Per la part dels usuaris estava bastant clar la informació a guardar a partir dels requisits que tenia l'aplicació. Amb un model bàsic on guardar les dades principals dels usuaris era suficient, per això es va optar per guardar id, name, email, edat i password.

En canvi, pel cas de les rutes hi va haver més indecisió, ja que les dades que obtenia de l'Alltrails i l'OpenDataBCN eren limitades i amb un format molt concret. Mentre que les rutes de l'Alltraills indiquen les coordenades i elevació, les de l'OpenDataBCN només tenien coordenades. A més, considerava necessari tenir més informació de cadascuna de les rutes per mostrar-li als usuaris, així que tota la resta de camps s'havien de calcular d'alguna forma com per exemple s'ha fet amb la distància. Els altres camps de forma manual s'havien d'especificar, era el cas del tipus de ruta, ja que totes les rutes descarregades eren de forma circular, pel que sempre es guarden així, tot i que està preparat per si en algun moment es vol guardar d'un altre forma. El nivell de cada ruta s'estableix a partir de la distància de cadascuna i el health_type s'estableix manualment a cadascun dels scripts encarregats de carregar les dades, com es veu a les figures 22 i 23.
També es va pensar de separar les rutes en dues taules, una per les que venen de l'Alltrails i l'altre per les de l'OpenDataBCN que corresponen a les zones verdes i així tenir una tercera taula "ZONES_VERDES", però finalment vaig preferir tenir-ho tot en una sola taula i diferenciar rutes per un atribut.

Amb tot això va quedar el model de base de dades de la Figura 26, és un model molt senzill, ja que tampoc havia de guardar molta informació pel que no hi ha relacions ni claus foranes, només les dues taules independents ja comentades.

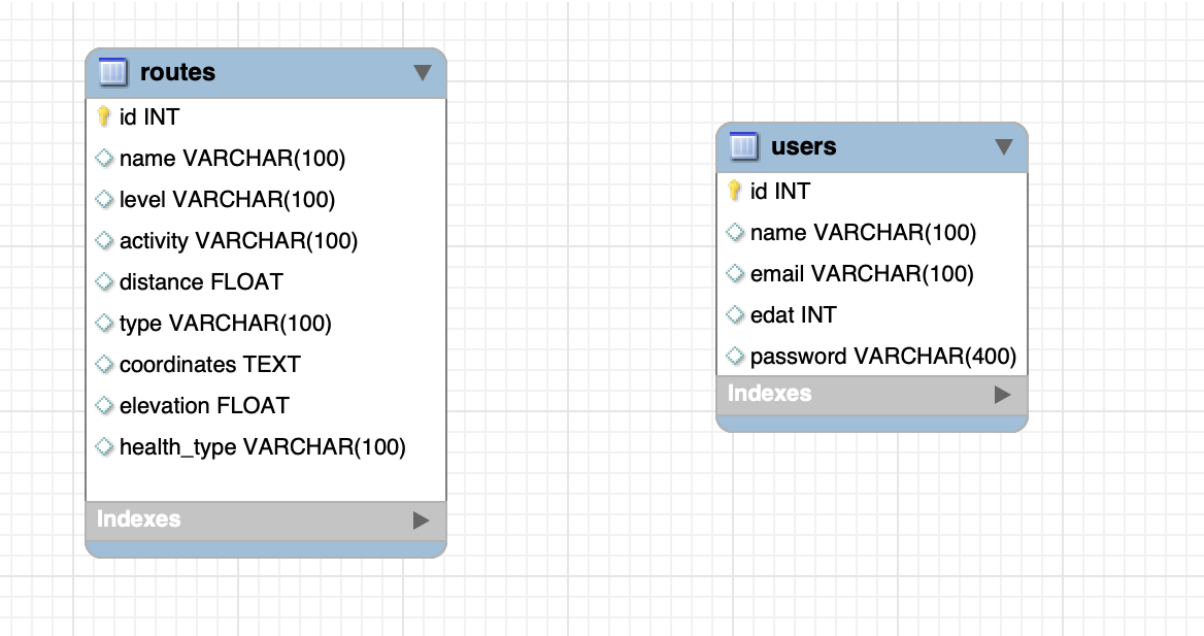


Figura 26: Diagrama relacional de la base de dades

A continuació, una part de les dades guardades a la base de dades, on es pot veure l'estructura comentada durant aquest apartat:

1	Paseo por Glóries	MITJA	PASEO	4.33	CIRCULAR	[[2.18448,41.40001],[2.18443,41.40023],[2.182...	2.28272	NORMAL
2	Funicular de Montjuic - Mirador de l'Antic Teatre...	FACIL	PASEO	2.36	CIRCULAR	[[2.1638,41.36665],[2.16364,41.36708],[2.1635...	17.5241	NORMAL
3	Parc Güell	MITJA	PASEO	2.58	CIRCULAR	[[2.1506,41.41389],[2.15009,41.41312],[2.1511,...	15.6777	NORMAL
4	La Munia	FACIL	PASEO	0.28	CIRCULAR	[[1.61788,41.32433],[1.61823,41.32464],[1.617...	141.4	NORMAL
5	Funicular de Montjuic - Esferic Barcelona - Cast...	DIFICIL	PASEO	4.99	CIRCULAR	[[2.16366,41.36657],[2.16326,41.36722],[2.163...	12.2561	NORMAL
6	Port Vell Barcelona	DIFICIL	PASEO	11.16	CIRCULAR	[[2.17874,41.37861],[2.17968,41.378],[2.17879,...	0.526316	NORMAL
7	Parc Via Trajana	FACIL	PASEO	1.11908	CIRCULAR	[[2.204361411587768,41.42855598822156],[2.2...	0	ZONES VERDES
8	Plaça de les Arts	MITJA	PASEO	1.61209	CIRCULAR	[[2.18541985481719,41.40096870296308],[2.1...	0	ZONES VERDES
9	Jardins Carrer del Cinca	FACIL	PASEO	1.04821	CIRCULAR	[[2.192784481842371,41.44117702715821],[2.1...	0	ZONES VERDES
10	Jardins del Dr.Pla i Armengol	FACIL	PASEO	1.07303	CIRCULAR	[[2.170694356293783,41.41409306685953],[2....	0	ZONES VERDES
11	Parc del Riu	DIFICIL	PASEO	2.4253	CIRCULAR	[[2.107254369516955,41.32795932902285],[2...	0	ZONES VERDES
12	Jardins Mossèn Costa i Llobera	FACIL	PASEO	1.37119	CIRCULAR	[[2.17019453549424,41.36475018336461],[2.1...	0	ZONES VERDES
13	Mirador del Poble Sec	FACIL	PASEO	1.09239	CIRCULAR	[[2.172426721723349,41.37130528581072],[2....	0	ZONES VERDES
14	Jardins de la Rambla de Sants	MITJA	PASEO	1.68226	CIRCULAR	[[2.135030162538386,41.37535817951119],[2.1...	0	ZONES VERDES

Figura 27: Exemple de consulta a la taula “routes”

id	name	email	edat	password
7	admin	admin@admin.com	24	6rounds=656000$fVv0ydygujeR/wrF$8FFsK...
26	Albert Perez	albertperezcosta99@gmail.com	20	6rounds=656000$rNUm8oArvqA4tope$DzOk...
27	Montse Costa	montse@example.com	58	6rounds=656000$/qAFoYGDyTrRLI7E$RzD...
28	Albert Perez	albert2@example.com	25	6rounds=656000$Zv6EYtn1uMtJrQjl$HWAy...
29	Toni Perez	antgasperez@hotmail.com	72	6rounds=656000$5u7NhxyzHZR4sQ3x$eFd...
32	Antonio Perez	antgasperez1@hotmail.co	69	6rounds=656000$zp2uXDhVsL7tUl49$m8OJ...
34	Antonio Perez	antgasperez1@hotmail.com	70	6rounds=656000$LCNFoekYuNw3Mz2y$WP...
36	Montse Costa	montse@gmail.com	59	6rounds=656000$rHjBozfn5ZCKsum9$5jUe2...
42	Montse2 Costa	montse2@example.com	65	6rounds=656000$kGyj953mTnWuvoH3$6rR...
43	Albert Perez Costa	albert@example.com	26	6rounds=656000$KQY9c9Ze.lcp4Tvd$3Ge9...
44	example example	a@a.com	50	6rounds=656000$T8j3yccv5ZzVSFzz$Ht3QA...

Figura 28: Exemple de consulta a la taula “users”

6.2.4. Endpoints

Amb tota l’estructura del backend ja explicada i vist el funcionament de las APIs RESTful, només em falta per comentar com han quedat finalment els endpoints que l’aplicació consumirà pel tractament de dades dels usuaris i la visualització de les rutes.

Hi ha un total de 7 endpoints (Figura 29) on tenint en conta que el host s'ha fet amb heroku i el domini es diu https://tfg-rutes-saludables-api.herokuapp.com/ queden de la següent forma:

1) **POST /account:** Endpoint encarregat destinat a afegir usuaris nous a la base de dades. S'ha implementat amb un POST on s'envia al body de la petició totes les dades necessàries pels usuaris.

2) **GET o PUT /account/<string:email>:** Aquest segueix sent el mateix endpoint que abans, però s'utilitza pels mètodes HTTP GET i PUT. Aquest s'encarrega a partir de rebre per paràmetre el correu d'un usuari ja creat fer un GET per recuperar tota la informació de l'usuari. O també es pot realitzar un PUT per a 'actualització de l'usuari que s'indiqui, enviant la informació a actualitzar al body de la petició.

3) **GET /accounts:** Endpoint que retorna el llistat complet de les rutes que hi ha a la base de dades amb tota la informació de cadascuna d'elles.

4) **POST /login:** Endpoint per crear el login de l'usuari a partir de l'email i la contrasenya. A partir d'aquestes dades es verifica que l'usuari existeix a la base de dades i es retorna la informació de l'usuari.

5) **POST /route:** Endpoint encarregat d'afegir rutes a la base de dades si així es necessita, a partir d'un POST amb tota la informació necessària per a les rutes. Actualment, no s'utilitza, ja que com he explicat, totes les rutes s'afegeixen amb l'script add_routes.py. Però s'ha deixat creat per si és necessari en el futur.

6) **GET /route/<string:id>:** Endpoint que retorna la ruta a partir de l'identificador que té cadascuna.

7) **GET /routes:** L'últim endpoint i pot ser el més important, és l'encarregat de retornar un llistat de rutes. Aquest llistat es pot filtrar segons les necessitats de l'usuari, a partir de tres paràmetres d'entrada; "health_type", "level" i "coords".

    a) **health_type:** Tal com el nom indica, correspon amb el camp de la base de dades i s'utilitza per sol·licitar a l'API que retorni les rutes filtrades pel tipus de ruta (proposta inicial, proposta verda o automàtica).
    b) **level:** El segon també correspon amb el camp level de la base de dades i fa que retorni rutes segons el nivell (fàcil, mitjà, difícil).
    c) **coords:** Aquest camp s'utilitza conjuntament amb el health_type quan té el valor "auto" indicant que l'usuari vol obtenir una ruta automàtica. Llavors, en aquest camp s'envia les coordenades actuals de l'usuari per retornar la ruta a partir d'aquest punt. Per la resta de combinacions no s'utilitza.

    Exceptuant el paràmetre "coords" que ja he comentat com i quan s'utilitza, els altres dos es poden utilitzar de forma individual o combinada. És a dir, si s'envia només health_type l'API retornarà las rutes filtrades per aquest camp i igual amb "level", però si s'envien tots dos es retorna les rutes que compleixen totes dues condicions.

```python
@app.route("/")
def home():
    return "Hello, TFG!"

api.add_resource(Accounts, '/account/<string:email>', '/account')
api.add_resource(AccountsList, '/accounts')
api.add_resource(Login, '/login')
api.add_resource(Routes, '/route/<string:id>', '/route')
api.add_resource(RoutesList, '/routes')
```

Figura 29: Endpoints creats per consumir l'API

Per facilitar, a futurs consumidors de l'API per les següents versions de l'aplicació he creat un swagger on es veu de forma clara i amb exemples com queden totes els endpoints i com funcionen (Figura 30). El swagger permet reproduir totes les peticions a la API existents per així fer una idea de com funciona i com implementar els serveis per consumir-la:

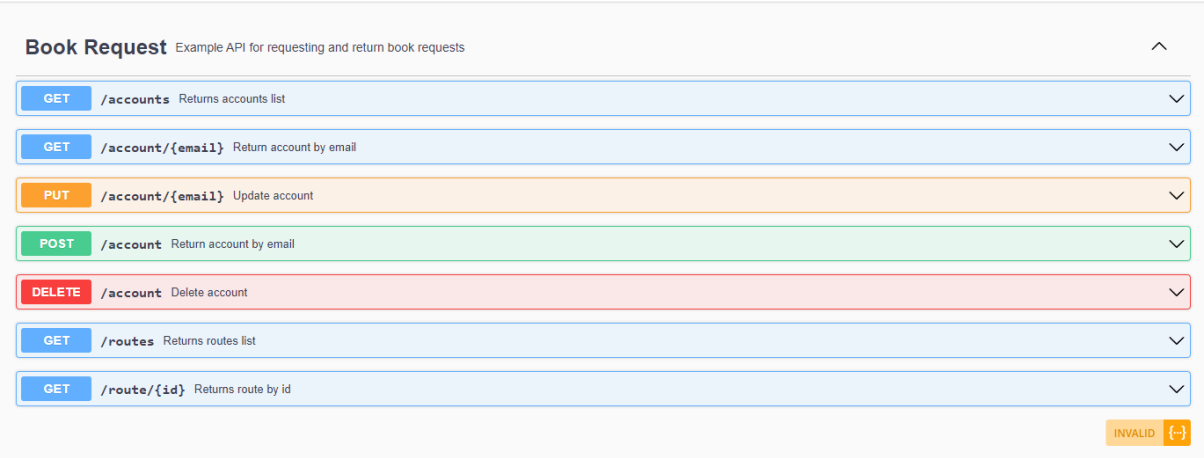


Figura 30: Swagger creat a https://tfg-rutes-saludables-api.herokuapp.com/swagger/

Com aquest domini és públic, s'ha afegit seguretat a alguns dels endpoints que tenen informació privada dels usuaris, com són els noms, correus i inclús contrasenya tot i estar encriptada. Per això, només les persones que tinguin les credencials d'administrador podran consultar tota la informació relativa als usuaris. La resta d'endpoints dedicats a les rutes són públics ja que de la mateixa forma que jo he obtingut aquestes rutes de plataformes públiques jo deixo aquesta informació també oberta a tothom.

Per tant, en el moment d'intentar executar els endpoints des de el swagger demanarà la informació de la Figura 31:

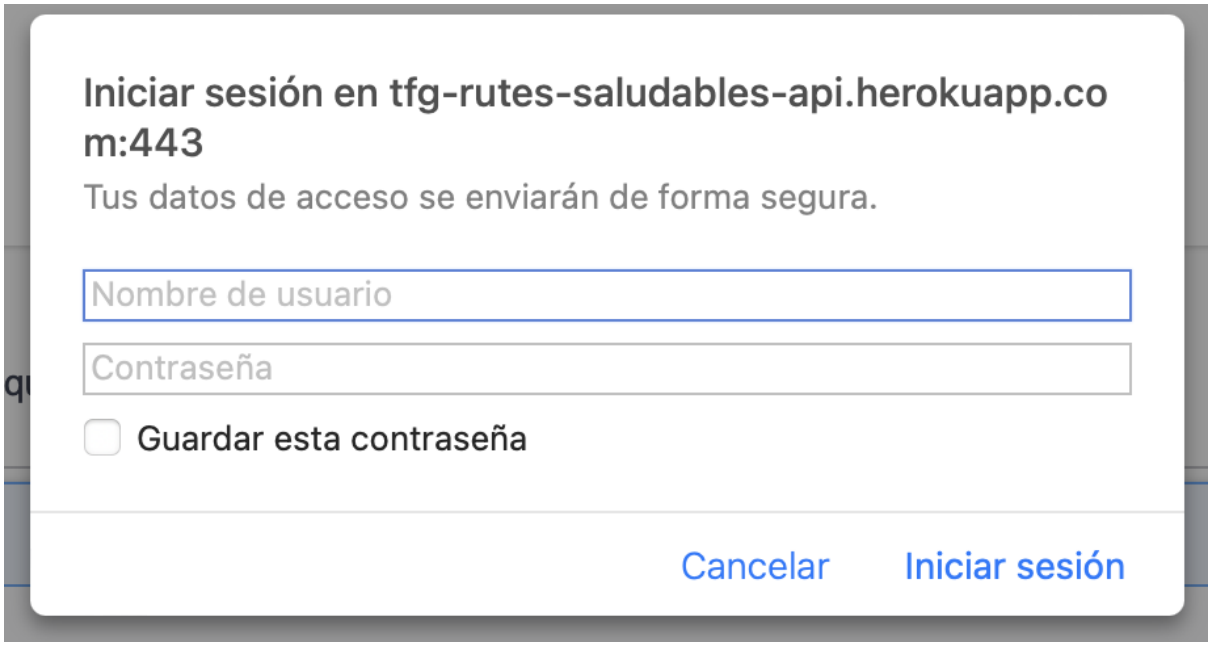


Figura 31: Credencials demanades per veure informació dels usuaris

Si es vol fer la consulta com a l'exemple de la Figura 32 a Postman o en el cas de voler trucar a aquest endpoint des del front de l'aplicació s'hauria d'indicar com a Header el camp "Authorization" amb valor "Basic + "nom usuari + contrasenya" codificat en base64, com es veu a la següent imatge d'exemple.

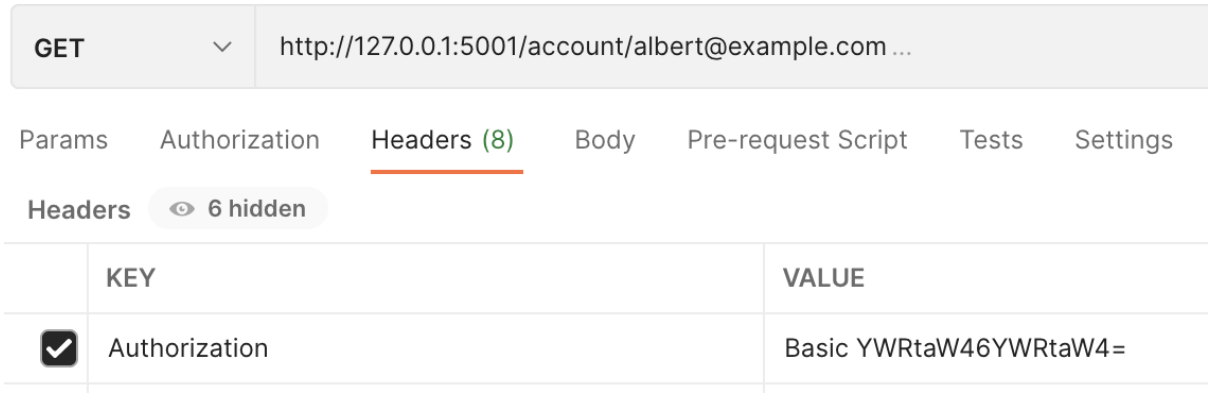


Figura 32: Exemple d'ús de credencials des de Postman

7. Resultats

En aquest apartat veurem els diferents resultats obtinguts després de la implementació de l'aplicació. Trobarem la interfície final i els resultats dels tests obtinguts durant tot el desenvolupament de l'aplicació.

7.1. Navegació

Aquí es mostra el resultat final de les pàgines implementades a la interfície d'usuari. Veient el flux complet que segueix l'usuari per a utilitzar totes les funcionalitats de l'aplicació.

- **Inici:** A continuació, es mostra l'inici de l'aplicació. Quan entres per primer cop et sol·licita permís per accedir a la teva localització (surt en anglès perquè és un emulador) com es veu a la Figura 33. Tot seguit, des de la pantalla d'inici de la Figura 34 es pot accedir al registre, inici de sessió o directament entrar com a invitat.

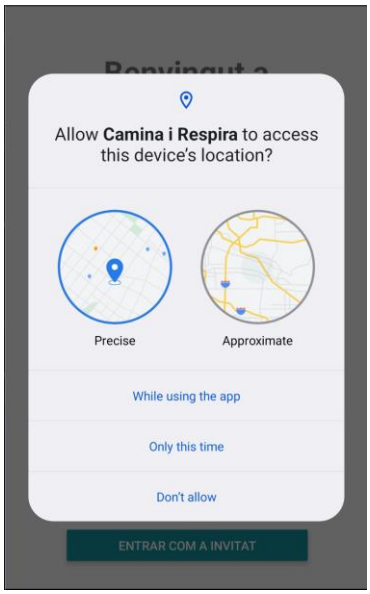


Figura 33: Sol·licitud localització

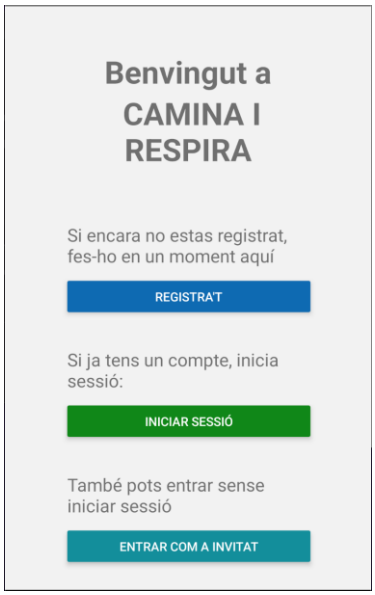


Figura 34: Pàgina d'inici

- **Registre:** En les figures 35 i 36 es veu el formulari de registre amb les validacions en cas que hi hagi un error. Un cop registrat accedim al Tutorial (Figura 37) on l'usuari llegirà com funciona l'aplicació.

Figura 35: Pàgina registre

Figura 36: Pàgina registre amb errors

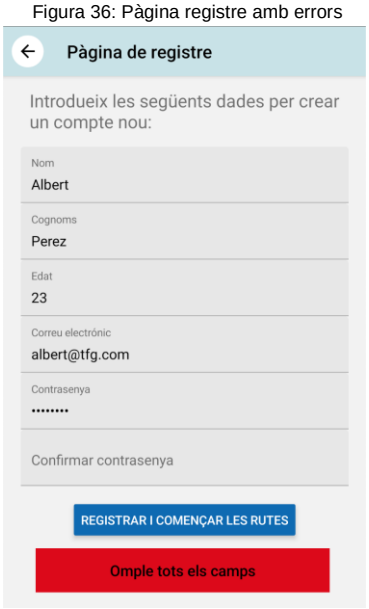


Figura 37: Tutorial

- **Inici de sessió:** Un cop accedim des de l'inici, en la pantalla de la figura 38 podrem iniciar sessió si ja tenim un compte registrat. En la figura 39 es veu el missatge d'error en cas que les validacions no siguin satisfactòries:

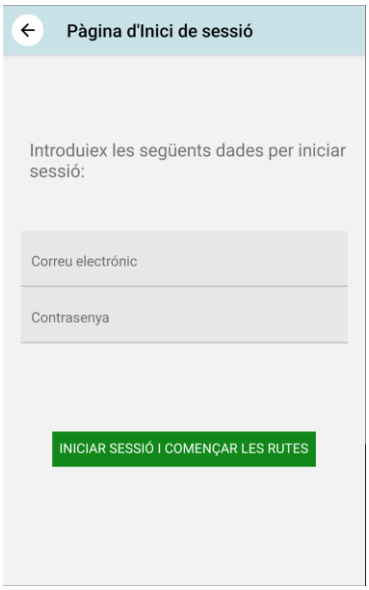


Figura 38: Pàgina inici de sessió

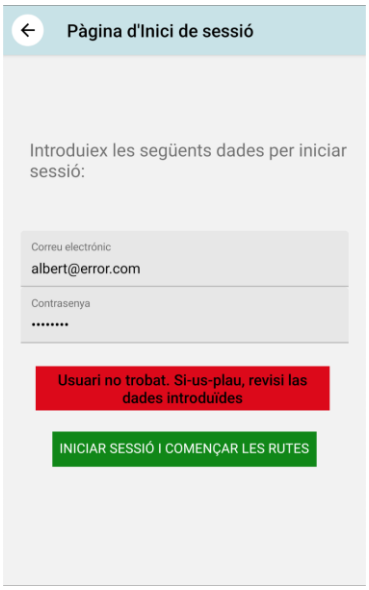


Figura 39: Pàgina inici de sessió amb errors

- **Main:** Un cop ens hem loguejat o entrat com a convidat accedim al Main de l'aplicació. Primer es mostra el disclaimer d'avís que no ens fem responsables de l'estat de les rutes (Figura 40) i a continuació es veu el mapa amb la ubicació i la interfície d'usuari per filtrar de la Figura 41:

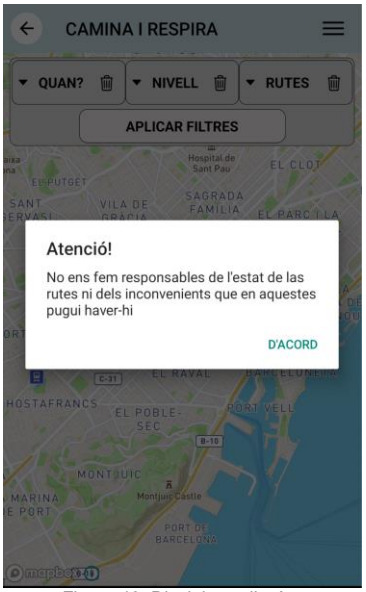


Figura 40: Disclaimer d'avís

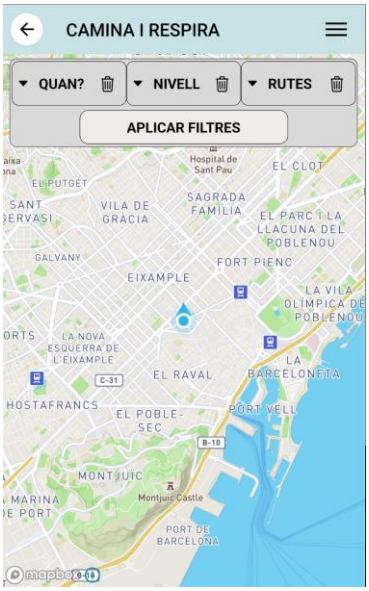


Figura 41: Pàgina principal Main

En el cas que s'intenta tornar enrere amb el botó de dalt a l'esquerra o amb els botons propis del mòbil anirem a l'inici però si s'ha iniciat sessió prèviament tindrà l'aspecte que es mostra a la Figura 42:

Figura 42: Página inici amb usuari loguejat

- **Menú lateral:** La figura 43 mostra com es veu el menú lateral al qual s'accedeix des del botó superior dret (marcat en vermell) quan estàs loguejat i et deixa anar a les pantalles d'iniciar sessió o registrar-se si encara no ho has fet. En canvi, en la figura 44 quan l'usuari té la sessió iniciada permetrà anar a la pantalla de modificar perfil o tancar la sessió.

Figura 43: Menú lateral sense loguejar

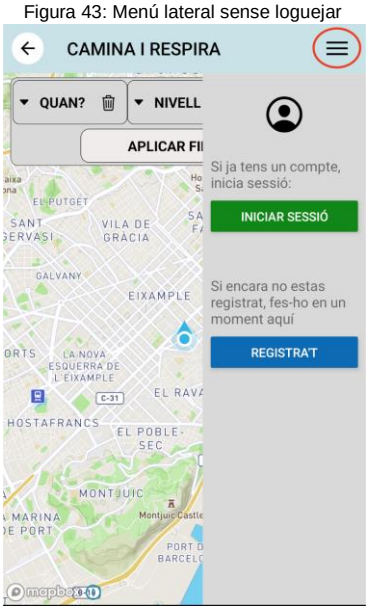
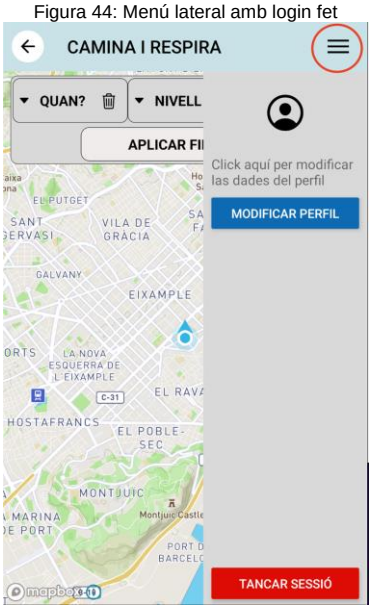


Figura 44: Menú lateral amb login fet



- **Modificar perfil:** Accedint des del menú lateral, podem modificar les dades del compte amb el formulari tal com es veu a la Figura 45. En cas d'errors es mostren com en la Figura 46. Si tot ha anat bé es mostra l'avís de la Figura 47.

Figura 45: Modificar perfil

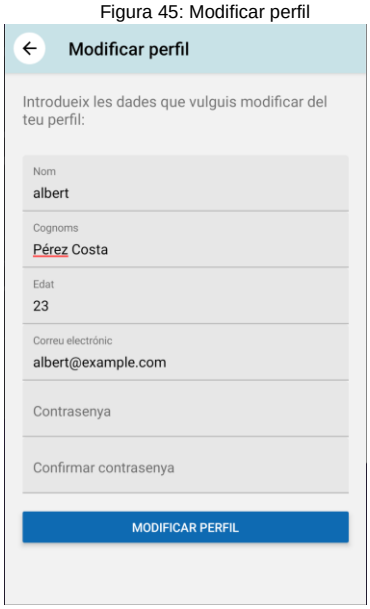


Figura 46: Modificar perfil amb errors

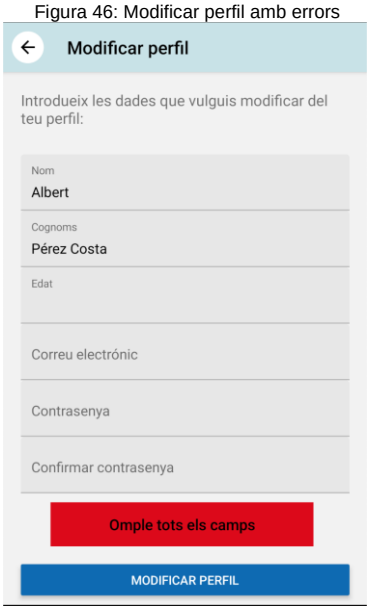
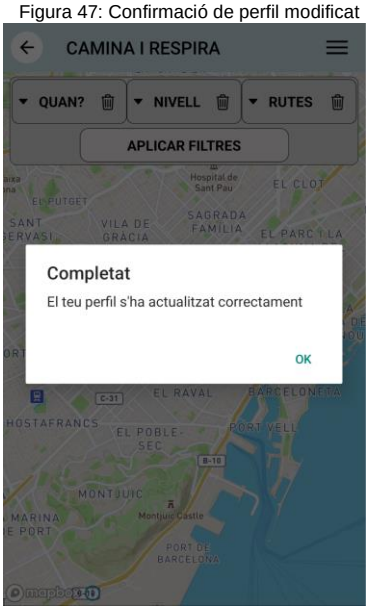


Figura 47: Confirmació de perfil modificat



- **Opcions de filtre:** Per començar a buscar rutes, es poden establir els filtres que es mostren a la Figura 48 per seleccionar el dia, els de la Figura 49 per seleccionar la dificultat de la ruta i els de la Figura 50 per al tipus de ruta que es vol fer. En la Figura 51 es veu com seleccionar l'hora en què es desitja fer la ruta per veure les dades de meteorologia que hi haurà.

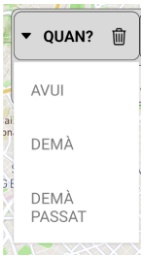


Figura 48: Filtre 1

Figura 49: Filtre 2

Figura 50: Filtre 3

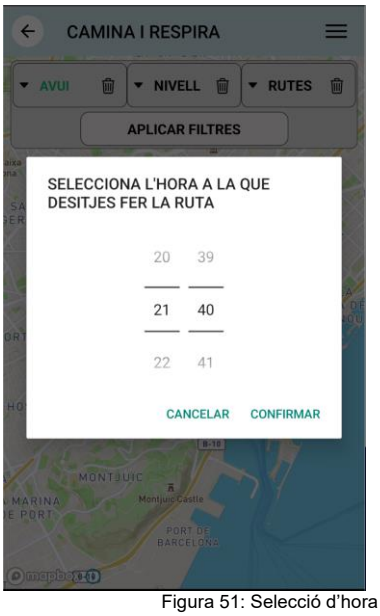


Figura 51: Selecció d'hora

- **Mostrem rutes:** Un cop filtrat per les necessitats de l'usuari, es pinten les rutes al mapa. A la figura 52 es mostren rutes per zones verdes i en la 53 una ruta automàtica des del punt on està l'usuari fins a una zona verda. Totes dues mostren rutes de color verd el que indica que no tenen contaminació. En canvi, en la Figura 54 es mostren grogues ja que la contaminació comença a ser més alta. Els altres dos casos de la Figura 55 i 56 indiquen que encara hi ha més contaminació i/o inclús possibilitat de pluja, per això els colors taronja i vermells.

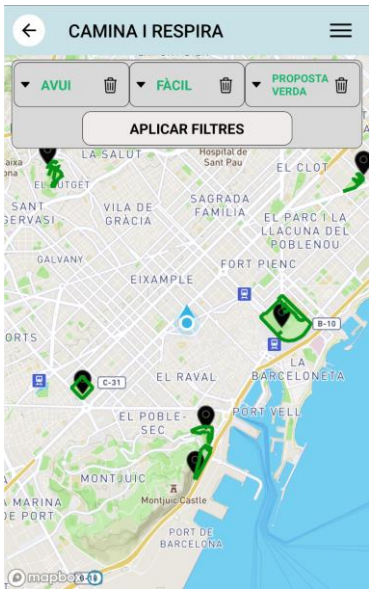


Figura 52: Rutes verdes sense contaminació

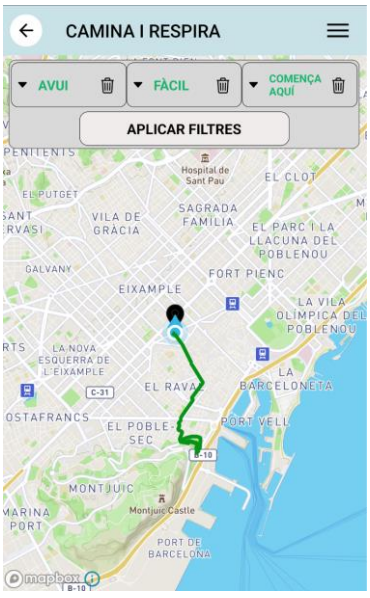


Figura 53: Ruta automàtica sense contaminació

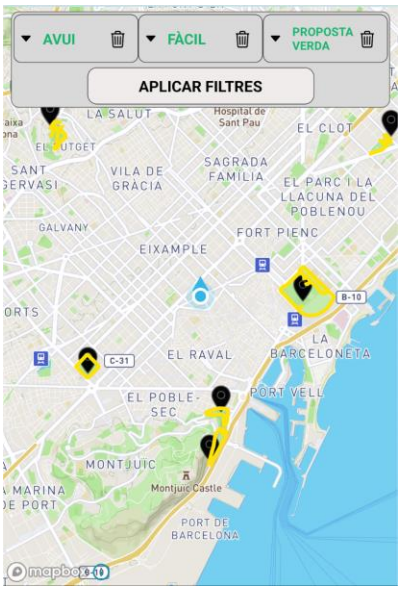


Figura 54: Rutes amb poca contaminació

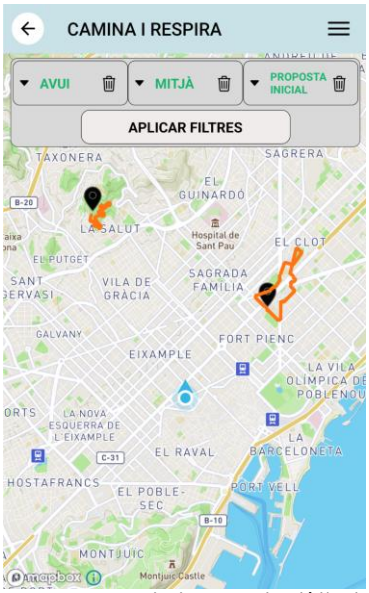


Figura 55: Rutes amb alta contaminació i/o pluja

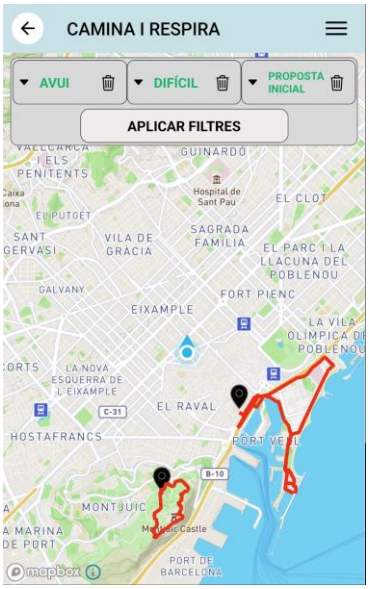


Figura 56: Rutes amb molta contaminació i/o pluja intensa

- **Selecció de rutes:** Quan es seleccionen les rutes s'obre un modal mostrant la descripció de la ruta, el qual tindrà el mateix color que es veia en la ruta segons las dades de contaminació i/o meteorologia. En el cas de la Figura 57, no tenim contaminació ni pluja pel que es veu verd. En la figura 58, ja es comença a tenir algo de contaminació però sense mostrar advertències. En canvi, en la Figura 59 ja hi ha una contaminació bastant elevada pel que es mostra un cartell d'advertència. Igual succeeix en la Figura 60, però aquí a més a més tenim previsió de pluja pel que també s'avisa. Ja en la figura 61 es mostra el modal en vermell i advertències pel fet que els nivells de contaminació són molt alts i també es preveu pluja intensa.

Figura 57: Info ruta sense contaminació contaminació

Figura 58: Info ruta amb poca contaminació

Figura 59: Info ruta amb alta

Figura 60: Info ruta amb alta contaminació i pluja

Figura 61: Info ruta amb molta contaminació i pluja intensa

- **Ruta començada:** Un cop s'ha decidit començar la ruta, es mostra la següent interfície d'usuari amb la ubicació actual i la ruta a seguir. En la Figura 62 es pot veure un cronòmetre per indicar a l'usuari el temps que porta caminant i dos botons a la part inferior, un per sortir i acabar la ruta i l'altre per pausarla. En la Figura 63 es veu quan es decideix parar la ruta que el botó canvia a "Reanudar ruta" i es para el cronòmetre, a més, es mostra un botó al mig de la pantalla per poder reanudar la ruta. Un cop es decideix sortir o s'intenti tornar enrere amb els botons del mòbil es mostra primer l'avís de la Figura 64 i en acceptar es redirigeix al Main de l'aplicació un altre cop.

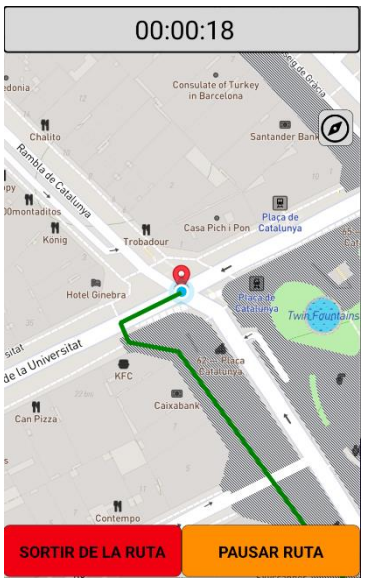


Figura 62: Ruta començada

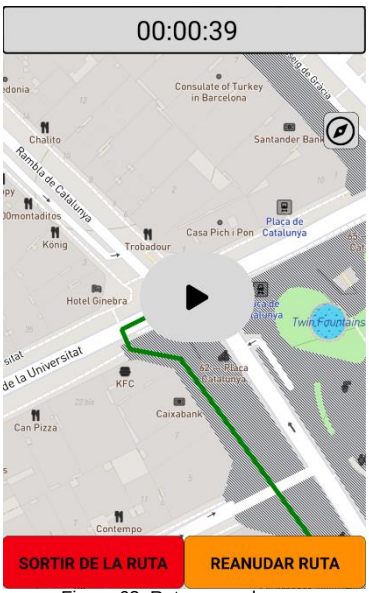


Figura 63: Ruta pausada

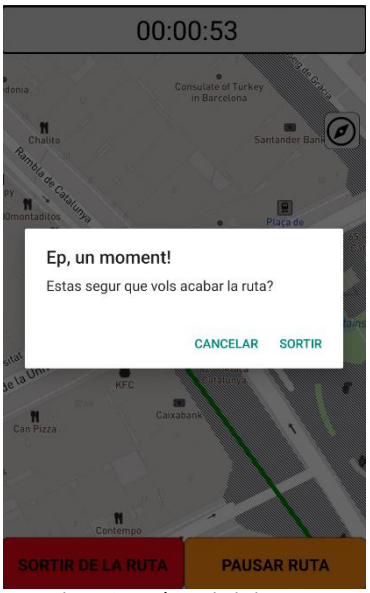


Figura 64: Avís sortir de la ruta

7.2. Accessibilitat

En ser una aplicació orientada a gent gran, és bastant important tenir en compte que la informació ha de ser clara i senzilla. Per això, es van seguir alguns criteris bàsics d'accessibilitat. Principalment, es van aplicar els següents conceptes:

- **Claredat (reducció cognitiva):** Mostrem informació bàsica a l'usuari, sense saturar amb tecnicismes ni excés d'informació que pot induir a confusions.

- **Contrast (pèrdua visió):** A causa de la pèrdua de visió que es pot tenir amb l'edat és sempre recomanable realitzar les aplicacions amb un cert contrast. On els elements primaris com els botons tinguin colors més propers al negre i els elements del fons utilitzin colors més propers al blanc, d'aquesta forma es produeix un contrast que ofereix una millor visibilitat i enteniment de la informació. Per a comprovar que això es complia es va utilitzar l'aplicació "Colour Contrast Analyser" que indica que els diferents components de l'app compleix amb els requisits bàsics d'accessibilitat.

- **Àrees d'activació força grans i separades entre si (motricitat fina no gaire bona):** També s'havia de tenir en compte que amb l'edat la motricitat no és tan bona igual que la vista, i per això es recomana utilitzar botons, textos i components tan grans com sigui possible perquè sigui més fàcil de veure i clickar per a l'usuari. Per això es pot veure com en l'aplicació s'ha intentat aplicar el concepte fent botons i textos grans.

7.3. Proves en usuaris

Com que ens trobem en el desenvolupament d'una aplicació mòbil la usabilitat d'aquesta és molt important. Per tant, s'han fet proves en usuaris finals per saber si la navegació de l'aplicació és intuïtiva i còmoda, i conèixer millor l'efecte que causaria l'aplicació als usuaris finals.

Per fer les proves s'ha definit un perfil de voluntaris d'entre els 55 i 70 anys. Seran persones grans amb coneixements bàsics de l'ús d'aplicacions al mòbil, però que no tenen tanta facilitat com pot tenir una persona més jove avui dia. Es faran les proves en un total de 6 voluntaris que són del meu cercle personal a les que he demanat participar.

7.3.1 Metodología

El tipus de prova dut a terme serà de validació, es cercarà que el producte compleix amb els objectius del disseny.
Les tasques es faran en un espai obert on es disposi d'una connexió estable. Les proves es realitzaran de forma moderada i se li explicarà al subjecte la idea de l'aplicació, però en cap moment se li explicarà com funciona, únicament ha de conèixer el concepte del projecte i realitzarà una sèrie definida de tasques a realitzar a l'aplicació que es faran de manera seqüencial. Es comptaran el nombre de passos

que ha realitzat per arribar a realitzar la tasca satisfactòriament i es compararan amb el nombre ideal d'accions que es requereixen per dur-les a terme. El llistat de tasques a realitzar és el següent:

1) **Registre**
2) **Log Out**
3) **Log In**
4) **Modificar perfil**
5) **Cerca de rutes “avui” “fàcil” “normals”**
6) **Cerca de rutes “avui” “difícil” “verdes”**
7) **Començar ruta**
8) **Sortir de la ruta**

Finalment, es faran una sèrie de preguntes, demanant observacions i millores que els subjectes pensin que poden augmentar la qualitat de l'aplicació. El llistat de preguntes que es faran seran les següents:

- **Creus que ha estat fàcil trobar els elements de l'aplicació que s'han demanat?**
- **Com valoraries estèticament parlant, l'aplicació?**
- **L'aplicació respon ràpidament?**
- **Què creus que podria millorar la qualitat de l'aplicació, o què creus que li manca?**

Un cop realitzades les proves, tenint en compte que cada acció començava a contar després d’haver realitzat els passos de l’anterior, és a dir, per realitzar la cerca de rutes no es compten els passos inicials de registre i/o login per accedir a la pàgina principal amb el mapa, sinó que directament partim des d’allà. Els resultats obtinguts van ser els següents:

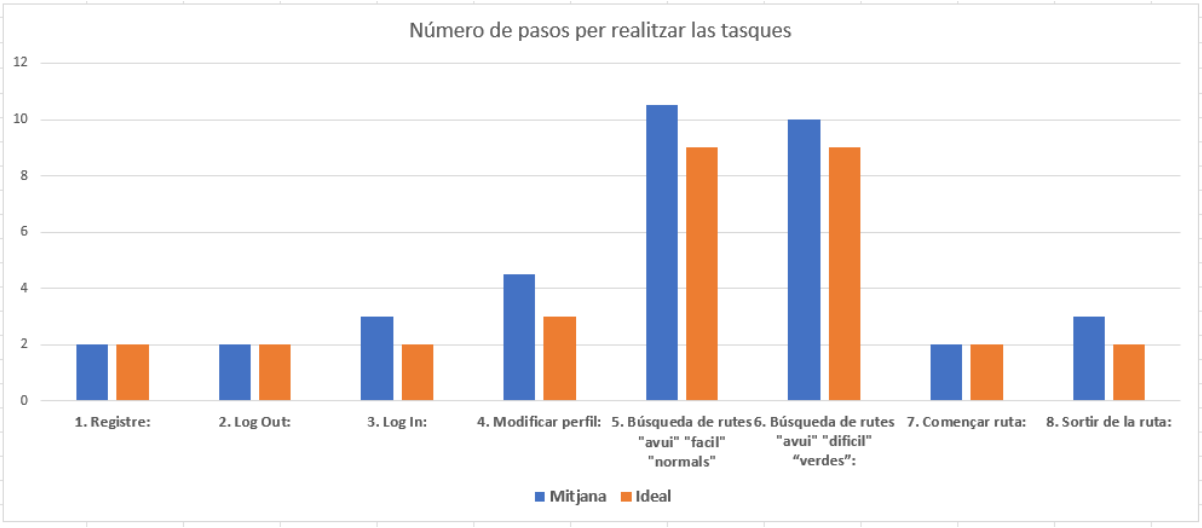


Figura 33: Resultats de les tasques demanades als voluntaris

A la gràfica podem veure el resultat de les accions que ha pres als usuaris realitzar les tasques en comparació amb un valor ideal representat amb el color taronja, aquest valor ideal és el nombre mínim d'accions per acomplir la tasca. Les tasques han estat realment satisfactòries en aquest aspecte, els usuaris han trobat els elements per dur a terme les tasques de manera molt fàcil i intuïtiva. A més, se n'ha pres nota d'algunes observacions a destacar observades durant la realització d'aquestes tasques:

- A la prova de Log In, alguns usuaris feien click a "Entrar com a invitat" primer hi havien de tornar enrere per realitzar el log in correctament.

- A la prova de Modificar perfil, alguns usuaris feien click enrere per buscar el modificar perfil a l'inici, ja que no s'havien fixat inicialment en el botó superior dret on s'obre el menú lateral per realitzar la modificació.

- Pel que fa a les cerques de rutes, tot i ser la funcionalitat que podem dir és més complexa de tota l'app, ha anat força bé. Alguns donaven a buscar directament, sense filtrar primer amb els valors indicats. Però en general ha anat bé.
- Quant a la tasca de Sortir de la ruta, alguns primer pausaven i després sortien de la ruta, el final era el mateix, però amb un pas extra, pel que tampoc és crític.
- En general, no hi han hagut complicacions en les tasques, alguns usuaris els hi costava més i utilitzaven algun pas de més, però acabaven arribant al final.

I referent a les preguntes formulades, els usuaris van contestar el següent:

➔ *Creus que ha estat fàcil trobar els elements de l'aplicació que s'han demanat?*

En general tots els usuaris han manifestat que els ha resultat fàcil fer les tasques, han comentat que és intuïtiva i fàcil d'entendre. A més, en quant a usabilitat els hi ha agradat la mida de la lletra que és llegible i els botons són prou grans per fer click.

➔ *Com valoraries estèticament parlant, l'aplicació?*

Tots els usuaris han respost de manera positiva a aquest aspecte. Tot i que alguns no els hi ha agradat com es veu un cop comences la ruta, principalment la part del cronòmetre.

➔ *L'aplicació respon ràpidament?*

Comenten que a vegades en buscar rutes és una mica lent, però agraeixen que el "spinner" estigui indicant que està carregant i han d'esperar. La resta de funcionalitats no hi ha hagut queixa alguna.

➔ *Què creus que podria millorar la qualitat de l'aplicació, o què creus que li manca?*

Comenten que estaria molt bé, una altra forma d'indicar les rutes disponibles i així no haver de fer click sobre el mapa, sobretot pel cas en què hi ha vàries rutes juntes i s'ha d'ampliar el mapa per fer click sobre la que vols.
Un proposa fer una espècie de llistat amb les rutes disponibles i fer click sobre aquest llistat que apareixeria sobre el mapa.

Dos dels sis voluntaris, van comentar que un cop començada la ruta estaria bé anar indicant quan l'usuari ha de girar, com la típica navegació de Google Maps.

En conclusió, les proves han demostrat que l'aplicació és funcional i usable per usuaris d'edats entre 55 i 70 anys que era l'objectiu que teníem i que majoritàriament no han tingut problemes per utilitzar-la. De les opinions dels voluntaris hem pogut extreure idees que potser es podrien aplicar en treballs futurs per les següents fases d'aquest estudi, com el que he comentat de fer un llistat per seleccionar les rutes i d'aquesta forma millorar l'experiència d'usuari. També s'ha pogut veure que la usabilitat en tema d'eficiència era bona i en tema de colors, mides de lletres i components era la correcta per a usuaris d'avançada edat, tot i que a partir dels tests es va decidir canviar els colors en lletres de botons que no s'acabaven de veure del tot bé a la pàgina de registre i inici de sessió.

8. Treballs futurs

Com ja s'ha comentat a l'inici del treball, aquest projecte forma part d'un estudi més gran i el desenvolupament de l'aplicació es va pensar en diverses fases on realitzaran més avenços. Els objectius dels treballs futurs que estan planificats per millorar l'app i proporcionar dades valuoses a aquest estudi són els següents:

1) **Personalització:** Afegir funcionalitats noves que ofereixin a l'usuari més participació i ajudin a detectar les seves preferències.
    - Definir reptes per motivar-lo a tenir una vida més saludable.
    - Avaluació de les rutes per part de l'usuari i que pugui compartir-ho.
    - Es podria guardar les rutes que l'usuari desitgi en una llista personalitzada on es guarden les rutes que més li agraden.
    - Afegir opció "De quant de temps disposes?", si té temps se li podria proposar que agafi el transport públic per anar a un altre barri on hi ha rutes saludables aquest dia.
    - Els usuaris poden pujar fotografies i comentaris per il·lustrar barreres i incentius per fer activitat física (Perspectiva de ciència ciutadana)
    - Incloure les dades d'activitat física recollides mitjançant el mòbil o polsera de l'usuari en la història clínica de l'usuari.

2) **Ciència de dades:** Algunes de les idees que també es van plantejar per al futur consisteixen a fer un recomanador entre usuaris. És a dir, a partir de rutes puntuades positivament, recomanar-les a altres usuaris:
    - Avaluació de realització de la ruta per a perfeccionar les recomanacions
    - Emmagatzemar històric. Guardar quan l'ha fet per tal de guardar l'estat meteorològic i contaminació del moment de quan l'ha realitzat i en un futur poder extreure conclusions sobre com influeixen els nivells de contaminació sobre les persones.
    - Crear alguna espècie de Chatbot amb recomanacions segons el temps. Com per exemple, "Fes exercicis a casa quan hi hagi episodi d'alta contaminació".

3) **Noves rutes:** Altres millores que es poden afegir, són les de tenir noves rutes disponibles, amb unes característiques diferents de les que s'han fet en la primera fase:
    - Una idea és la de proposar rutes noves utilitzant el catàleg d'arbres de tota Barcelona, el qual indica la posició de tots els arbres de la ciutat. D'aquesta forma es podrien generar rutes els més saludables possibles passant per carrers amb el nombre més gran d'arbres possible.
    - Els mateixos usuaris puguin crear rutes per la ciutat al seu gust.

9. Conclusions

En el primer apartat d'aquest treball estan definits els objectius a assolir al final del desenvolupament d'aquest projecte. En particular, es volia crear una primera aplicació MVP per:

"Permetre als usuaris sortir a caminar i/o córrer mostrant rutes per la zona de Barcelona on està situat l'usuari (geolocalitzat) amb indicacions dels nivells de contaminació i també de la meteorologia. D'aquesta forma els usuaris seran conscients de la contaminació i condicions meteorològiques del moment en cada una de les rutes. Poden escollir d'aquesta forma les rutes més saludables o en el cas que totes les rutes tinguin una qualitat d'aire o condicions meteorològiques desfavorable podrà decidir esperar i sortir en un altre moment o dia a fer exercici."

Un cop finalitzat el treball, es pot dir que l'objectiu principal s'ha aconseguit, ja que l'aplicació és totalment funcional perquè les persones puguin sortir a caminar i/o córrer per la ciutat de Barcelona i indicant els nivells de contaminació i meteorologia. Tot i que l'objectiu principal s'ha aconseguit, van sorgir alguns problemes que van provocar que l'app no fos tan completa com es volia. El problema de l'obtenció de nivells de contaminació per a dies futurs impacta una mica sobre la funcionalitat principal de l'aplicació, tot i que per al dia actual no hi ha problemes, quan l'usuari vol mirar la contaminació a dies futurs no és possible a causa de l'API que s'utilitza que no permet donar prediccions amb plans de dades gratuïts. Pel que és un punt amb el qual no quedo del tot satisfet tot i que en ser un projecte estudiantil on no es volia gastar diners no es podia fer res més. Un punt que també es va plantejar a l'inici i que els usuaris també van trobar a faltar quan es van fer els tests, va ser una interfície més còmoda quan s'inicia una ruta. La idea era mostrar una interfície que indiqui a cada gir cap on s'ha d'anar com quan es posa el Google Maps, però va haver-hi problemes amb el que oferia Mapbox, ja que només es podia utilitzar aquesta eina quan es vol anar d'un punt a un altre, en canvi, pel nostre cas s'anava per 25 waypoints fins a completar la ruta i va fer impossible fer-ho.

En conclusió, tot i aquests dos punts, l'aplicació en general compleix amb tot el demanat i els usuaris que la van provar van estar satisfets amb la part estètica i amb les funcionalitats que ofereix. A més, brinda un punt d'inici a l'estudi de la Dra. Maria Grau des del que poder continuar amb noves fases per afegir noves funcionalitats a l'aplicació i arribar al seu objectiu final de millorar l'envelliment saludable de les persones.

10. Bibliografia

1. “API contaminació” https://docs.ambeedata.com/#aq-history-geospatial

2. “Rutes OpenData per caminar a barcelona” https://opendata-ajuntament.barcelona.cat/data/ca/dataset/np-circuits-caminar-correr

3. “Rutes per caminar a barcelona” https://ajuntament.barcelona.cat/esports/es/deporte-en-el-espacio-publico/correr-circuitos-deportivos

4. “Rutes allTrails” https://www.alltrails.com/es/

5. “API Meteocat” https://apidocs.meteocat.gencat.cat/documentacio/

6. “Documentació Mapbox” https://docs.mapbox.com/android/maps/guides/

7. “React native set up” https://reactnative.dev/docs/environment-setup

8. “Documentació React Native” https://reactnative.dev/docs/components-and-apis

9. “Guies Flask” https://flask.palletsprojects.com/

10. “Set up Flask” https://code.visualstudio.com/docs/python/tutorial-flask

11. “Hosting amb Heroku” https://www.heroku.com

12. “AWS Relational Database” https://eu-west-1.console.aws.amazon.com/rds/home

13. “Repositori amb tot el codi” https://github.com/

11. Annex

A continuació, es mostra la informació necessària per al traspàs del projecte als següents desenvolupadors que vulguin continuar el projecte.

Hosting:
Han de tenir en compte que per a fer el deploy del backend necessitaran un compte nou de Heroku o qualsevol altre servei de Hosting que es desitgi, ja que el meu deixarà de funcionar un cop finalitzi el treball.

APIs:
S'haurà de crear comptes per accedir a les APIs de contaminació Ambee per a obtenir una nova API KEY per versions gratuïtes o utilitzar qualsevol altre servei que es vulgui. Però un cop acabat el treball les dades de contaminació deixaran de funcionar.

Per l'API del Meteocat, també s'haurà de demanar un nou accés, ja que la meva API KEY té un ús de 3 mesos.

Per Mapbox, la KEY és gratuïta i en principi sense caducitat, però es recomana crear un compte propi, per si un cas.

Totes les KEYs i links a canviar es poden trobar al fitxer ***api/constants.js***.

Base de dades:
La base de dades, feta amb AWS també deixarà de funcionar un cop acabat el treball, per la qual cosa s'haurà d'utilitzar una altra conta d'AWS o qualsevol altre base de dades que es prefereixi. Es poden canviar les dades a les quals apunta el Backend des del fitxer ***config.py***.