Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: Desenvolupament d'una aplicació de guies de turisme

Document: Memòria

Alumne:
Dan Spoiala
Hamza Saddouki

Tutor: Ignacio Martín
Departament: Departament d'Informàtica,
Matemàtica Aplicada i Estadística (IMAE)
Àrea: Llenguatges i sistemes informàtcs

Convocatòria (mes/any): Setembre 2022

Universitat de Girona
Escola Politècnica Superior

PROJECTE FI DE GRAU

Desenvolupament d'una aplicació de guies de turisme

Autor:
Dan SPOIALA
Hamza SADDOUKI

Setembre 2022

Grau en Enginyeria Informàtica

Tutors:
Ignacio MARTÍN

Resum

Avui en dia qualsevol cerca per internet per una recomanació porta cap a un portal web d'alguna agència de viatges o bàsicament a qualsevol mena de publicitat sense aportar gaire informació sobre el lloc on un usuari vol viatjar.

L'interès de molts usuaris és trobar informació accessible i centralitzada sobre un país o ciutat sobre la qual tenen interès per viatjar en un futur pròxim o llunyà, molts dels usuaris solen cercar molts vídeos o experiències per xarxes socials, però poques vegades les xarxes socials ens donen un enfocament objectiu i pròxim.

Amb aquest projecte tenim la finalitat de facilitar als usuaris un entorn on podran publicar i cercar rutes, opinions i recomanacions d'altres usuaris sobre països i ciutats a les quals tinguin un interès per viatjar, d'aquesta manera l'enfocament serà més cap als usuaris i no tan enfocat en un interès econòmic dirigit a les agències de viatges o a algun tipus de promoció especial.

Agraïments

Per començar volem agrair molt especialment a la universitat per aportar-nos aquesta oportunitat per crear un projecte en el qual es posa a prova els nostres coneixements adquirits al llarg de la nostra carrera com enginyers informàtics.

També volem agrair als nostres professors que ens han tingut paciència i ens han ajudat a portar a terme els nostres objectius.

Finalment, volem agrair a les nostres famílies per aguantar aquesta temporada de TFG i donar suport en tot moment perquè ens vagi del millor possible.

Índex

Capítol 1

Introducció

Quan una persona vol viatjar a un país o ciutat a la qual no pertany amb un objectiu de fer turisme sol tenir molts dubtes dels llocs a visitar, preus, restaurants, on trobar un bon habitatge temporal, etc. Com a norma la solució és començar a cercar informació al respecte, però aquesta no està centralitzada sinó que està escampada per diversos llocs web, llavors l'usuari es pot arribar a confondre o desistir del viatge, ja que no té clar a fer com a resultat de la cerca de diverses pàgines web on la majoria són d'agències de viatge oferint serveis i publicitant llocs per treure un benefici.

El nostre propòsit és oferir un espai centralitzat en forma d'aplicació mòbil on les persones puguin compartir les seves opinions, guies i llocs recomanats d'una zona (una ciutat, regió o país). Aquestes publicacions seran puntuables per donar a conèixer les millors per zona, també volem afegir altres filtres que permetran seleccionar un rang de preu o una puntuació mínima. Tenim la finalitat de facilitar la vida a l'usuari i com a tal afegirem diverses seccions on es podran visualitzar les publicacions preferides o un mapa on es pugui seleccionar de forma interactiva una zona per a veure les guies disponibles, entre altres.

Entenem que no tots els usuaris tindran un comportament correcte i seguiran les normes i restriccions sobre el contingut a publicar, per això hem decidit afegir una administració de contingut. L'objectiu principal de l'administració serà poder decidir quins continguts publicar i quins no quan un usuari guardi la seva publicació, bloquejar el compte d'un usuari, etc. En aquest cas oferim una web en la qual de manera interactiva es pugui dur a terme les tasques d'administració.

1.1 Motivació

Durant el grau universitari toquem molts àmbits de la informàtica des de programar a molt baix nivell, estudiar i dissenyar xarxes, manipular robots, etc. Però amb el que més ens hem sentit identificats com a pròxims enginyers informàtics és en el disseny i implementació d'aplicacions. A assignatures com Multimèdia i interfícies d'usuari i Projecte de desenvolupament de software ja entrem en aquest apartat, però donem sobretot les bases perquè ens sigui fàcil després desenvolupar-nos en aquests coneixements.

Un cop ja teníem les bases volíem saber més sobre aquest tema i realment fer un producte que utilitzes les tecnologies i entorns que faria servir una empresa actual, ja que en l'actualitat hi ha moltíssims entorns per escollir. Això ens ha motivat a plantejar-ho com a TFG, però el problema era com enfocar-ho, pel fet que necessitàvem un tema sobre el qual treballar.

D'estar parlant ambdós un dia a classe a la universitat sobre viatges i on volíem anar el pròxim estiu vam iniciar una discussió sobre les Apps de viatges, de tota la publicitat que et mostren, com t'ofereixen llocs que els hi interessa, tota la informació que està en diferents aplicacions, etc. Així que ens vam plantejar que era una bona oportunitat per a desenvolupar una App que permeti als usuaris crear guies i rutes de turisme, perquè així altres usuaris en facin ús.

1.2 Objectius

L'objectiu principal és aprendre com més millor sobre el desenvolupament d'aplicacions en el món laboral de l'actualitat, com ho fan, quines són les tecnologies més usades, quina és la seva forma de treballar, etc.

Un cop tinguem decidits els entorns de desenvolupament haurem d'aprendre com instal·lar aquests i més important com es fan servir de manera correcta, seguint al màxim possible les bones pràctiques dels diferents llenguatges.

Un punt clau serà aprendre com es connecten les diferents entitats de forma eficient i segura per a protegir els usuaris i assegurar una utilització fluida de les aplicacions.

Per mantenir tot organitzat i sota control haurem d'aprendre a treballar en equip seguint una metodologia.

No ens podem oblidar de la llei de protecció de dades i és que si publiquem l'APP al públic serà molt important complir amb tot i afegir el necessari a les aplicacions.

Aconseguir uns productes que compleixin amb els requisits i que siguin una bona base per a desenvolupar a posteriori sobre aquests.

Capítol 2

Estudi de viabilitat

Pel fet que és un projecte per compte propi l'hem plantejat a cost 0 o el més pròxim a aquest, sempre hem intentat utilitzar programari lliure o gratuït, però que funciones bé, cosa que en uns inicis ens ha fet passar un cert temps informant-nos sobre alternatives.

El que teníem des d'un inici són dos ordinadors personals prou potents per a programar i mantenir en funcionament qualsevol servei que fes falta i també som dos programadors disposats a dedicar-hi hores així que ara només quedava elegir bé el programari utilitzar, tecnologies per a desenvolupar i a quin o quins serveis hi guardaríem les dades.

El principal dubte a resoldre eren els mapes i és que la nostra aplicació requereix sobretot un servei per mostrar mapes i rutes sobre aquests mapes, el proveïdor més accessible i més fàcil que ens aporta aquests requisits és Google, el primer punt que hem de tenir en compte és que Google ens permetrà mostrar els mapes tant a Android com iOS de forma gratuïta sense cap cost ni cap limitació utilitzant unes credencials que podem demanar, però el servei de mostrar rutes sí que té un cost, se'ns permet utilitzar aquest servei de forma gratuïta cada mes fins a les 100.000 peticions, a partir de les 100.000 peticions existeix un cost de 3,73€ per cada 1000 peticions, encara que aquest segon servei té un cost el podem utilitzar gratuïtament per començar, ja que és poc probable que arribem a aquest nombre de peticions en llançar l'aplicació a producció, en qualsevol cas si arribéssim a aquest volum d'usuaris podríem monetitzar l'aplicació i es podria amortitzar.

Respecte a tecnologies per desenvolupar les que més ens han cridat l'atenció i disposaven d'un accés gratuït i sense cap cost per programar han sigut: per a l'API NodeJS, pel Backoffice utilitzarem Angular i finalment utilitzarem React Native per l'aplicació mòbil, tots aquests frameworks tot-hi ser gratuïts disposen de molta documentació i solucions per internet per possibles futurs problemes, a part que podem utilitzar milers de llibreries útils a partir del servei npm de forma total gratuïta.

Per publicar els nostres programes hem seguit la temàtica de mínim cost, primer

per publicar l'API i la seva corresponent base de dades hem pensat utilitzar el servei Heroku, el qual ens permetrà tenir dos servidors on tenir-ho de forma pública sense cap cost, el Backoffice podem utilitzar Github i el seu servei Github Pages que ens atorga la possibilitat de posar-hi una pàgina web amb un domini sota github sense cap cost, finalment les aplicacions mòbils sí que tindrien un cost, si volem publicar l'aplicació en Android, hem de crear un compte de desenvolupador al Google PlayStore i fer un pagament de 23,32€ un sol cop. Per altra banda, si volem publicar en iOS hem de crear un compte de desenvolupador a l'AppStore i a més si volem publicar existeix un cost de 92,35€ anuals, a més de les restriccions que té Apple que per a publicar necessites un dispositiu amb MacOS en aquest cas ens podríem replantejar si publicar inicialment a iOS o esperar a obtenir algun benefici abans.

2.1 Pressupost inicial

Encara que no hem generat cap benefici amb el projecte sí que hem vist interessant fer un petit pressupost molt general de què hagués costat el desenvolupament del mateix tenint en compte que aquest es queda en un estat de preproducció i que si volguéssim publicar-lo hauríem d'afegir els costos mencionats anteriorment de les tendes d'apps a més de contractar servidors més potents per a mantenir les dades i satisfer les necessitats dels clients, entre altres costos que podríem detectar a posteriori. Per fer una aproximació i el càlcul ens hem basat en la web **https://www.calculadorafreelance.com/** que ens dona informació més en detall del que hauríem de cobrar si volguéssim fer d'aquest un projecte rendible real.

2.1.1 Costos directes

Aquests són els costos del projecte que tenen relació directa amb el volum de feina i la seva dificultat. Aquests costos els hem calculat de manera aproximada extraient la informació de diferents pàgines d'internet i sempre hem intentat agafar preus tirant a baixos, ja que hem considerat que els dos som programadors de perfil junior llavors és normal que tardem més a fer certes tasques que el que tardaria un programador sénior. En aquest camp han sortit un total de quatre apartats a considerar i la definició del que engloba cada apartat seria la següent:

- Documentació: Documentar el codi desenvolupat però sobretot la memòria del projecte.

- Anàlisis i Disseny: Tasques de disseny funcional d'APP, BackOffice i API però també tasques de disseny d'interfície d'APP i BackOffice.
- Desenvolupament APP, API, BackOffice: Com indica el seu nom les tasques a complir i programar per a crear l'APP, l'API i el BackOffice.
- Testing APP, API i BackOffice: Com indica el seu nom les tasques de test d'usabilitat de l'APP, API i el BackOffice.

Si calculem un salari brut mitjà mensual de 2000€ d'un programador junior, per un any hauríem de guanyar 24000€ en brut cada un de nosaltres. Això fa un preu/hora brut de 16,48€ que podem obtenir fent 24.000 €/hores de feina anuals = 16,48€. Les hores de feina anuals les calculem tenint en compte una mitja de 4 hores al dia durant els 7 dies de la setmana i durant 52 setmanes de l'any.

Recapitulant si hem de cobrar en brut 16,48€ cada un, hem de complir 15 crèdits ECTS del treball i cada crèdit són 25 hores de feina tenim un resultat de 12.360€ en brut que hauríem de rebre si contemplem els costos directes.

2.1.2 Costos indirectes

Aquests són els costos que tenen relació amb les eines i espais utilitzats per dur a terme el projecte. En aquests costos no hem tingut en compte els ordinadors i materials dels quals ja disposàvem abans de començar el projecte. Hem fet el recompte de què ens hagués costat llogar un lloc de treball per fer el desenvolupament durant aquests mesos de feina amb les seves despeses com l'internet i llum. El total de costos indirectes ens surt a 7800 €

2.1.3 Costos totals

Observem que ens queda un cost total de 20.160€ pel desenvolupament del projecte. Això és molt molt baix, ja que si fem servir la web ens recomana un preu aproximat hora de 35,05€ per a treure rendibilitat, però nosaltres només hem fet servir el preu/hora brut inicial sense tenir en compte les vacances ni les hores no treballades.

CAPÍTOL 3

Metodologia

La metodologia sempre va íntimament lligada a l'estructura i qualitat final d'un projecte, ja que una bona metodologia ens assegura un producte més refinat i ben estructurat. En el nostre cas en ser un treball en grup encara aquest apartat cobra més importància perquè les dues parts al final havien de treballar com si fos una única.

Durant el grau en Enginyeria Informàtica hem tingut pinzellades de diverses metodologies per a desenvolupar projectes, sent les més conegudes en l'àmbit empresarial la Waterfall (Cascada) i Agile (Àgil). Com que els dos integrants del grup ja som part d'empreses i hem vist com aquests treballen actualment hem decidit que el que més ens convenia era el mètode Agile. Aquest mètode ens beneficia, ja que consisteix a anar iterant sobre el producte per anar afegint funcionalitats, però assegurant-se que tot el que hi havia anteriorment funciona correctament, això fa que els problemes es detecten i corregeixin durant el desenvolupament a mesura que anem incrementant el producte i li anem afegint capes, per altra banda, si haguéssim triat Waterfall el test es fa just al final i és quan hauríem detectat els problemes i haguéssim dedicat temps a corregir-los.

Aquesta metodologia té quatre parts bàsiques per a funcionar, primer de tot es fa una reunió de planificació (Planning) on es decideix que es farà en el pròxim període de treball (Sprint), quan tots els membres estan d'acord es tanca aquesta reunió i es passa a la següent part que és la implementació, durant aquesta part és molt important la comunicació entre membres de l'equip per a assegurar un progrés uniforme, normalment se sol fer una reunió diària de pocs minuts on cada membre informe al grup referent al seu avanç en les tasques que li pertoquen. Un cop acabat el període de treball es fa una revisió (Review) on es fa un resum de quines tanques s'han acabat o quines queden pendents, aquesta revisió va seguida d'una altra reunió anomenada retrospectiva (Retrospective) en el que no es busca tant quins objectius s'han complit i quins no sinó que es busca més la raó per la qual no s'han pogut acabar les tasques per així per a la pròxima reunió de planificació i període de treball tenir-ho en compte. Finalment, aquestes quatre parts es van repetint fins a acabar totes les tasques del projecte.

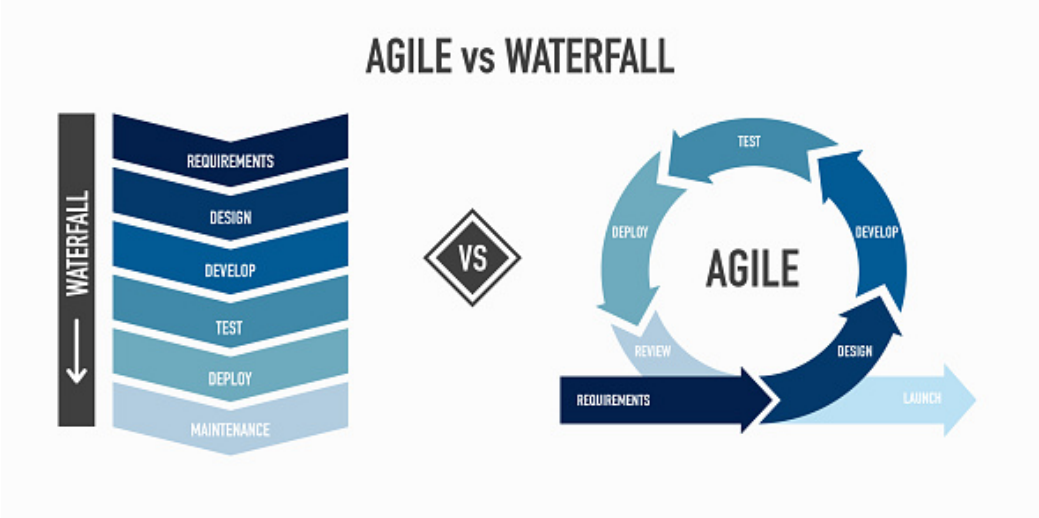

Figura 3.1: Gràfic on es compara les dues metodologies mencionades.

En el nostre cas cada cap de setmana fèiem un Planning per a organitzar la feina de cada Sprint que per nosaltres era una setmana, durant la setmana intentàvem que almenys cada dos-tres dies l'altre company sàpiga l'estat de les nostres tasques per així saber ell amb quines seguir en el pròxim període. Quan tornàvem a arribar al cap de setmana fèiem de manera breu la Review i Retrospective i tornàvem a fer un Planning fins a acabar el projecte. D'aquesta forma manteníem una bona comunicació i anàvem fent intercanvi d'opinions respecte a quines tasques fer primer i quines deixar per més endavant.

Com era d'esperar no tot ha anat perfecte, ja que una setmana és poc temps per certes tasques que hem agafat massa grans i que hagués sigut molt millor dividir en subtasques o, per altra banda, tasques que a priori semblaven fàcils, però al final han sigut un mal de cap important. Un exemple d'això és la integració dels mapes que semblava fàcil per APP, ja que era fer servir una llibreria externa que et dona quasi tot fet i només cal integrar i més complicat per l'API per guardar aquestes, perquè no sabíem ni com fer-ho, però al final la integració a l'APP va donar més problemes de conflictes entre llibreries i que no acabava de funcionar que no pas a l'API per a guardar-les. Coses com aquestes va fer que una tasca es fes en varis Sprints i no només en un com estava planejat.

CAPÍTOL 4

Planificació

En el capítol anterior hem comentat la metodologia que volíem continuar durant el treball, la metodologia Agile, però ens falta definir quines tasques necessitem fer per a assolir l'objectiu del projecte, ja que seran les que anirem posant dins els Sprints a mesura que anem fent les iteracions de la metodologia. A priori els dos partim de saber que volem que es pugui fer en l'aplicació: crear rutes i guies, compartir mapes, veure les rutes i guies i puntuar aquestes, etc. Però ens fa falta organitzar i ordenar aquestes tasques de manera que sapiguem quines seran més prioritàries, quines han d'anar abans que altres, si ens hem deixat alguna o si hem pensat de mes. Un primer pas per posar ordre és descompondre el contingut del projecte en paquets de treball, aquests paquets els hem plantejat com els pilars del projecte i cada un rep un títol que agruparà les tasques relacionades.

4.1 Paquets de treball

Com podem observar en la figura 4.1 hem dividit el projecte en cinc paquets de treball, un primer de documentació, un segon de planificació o estudi, un tercer de desenvolupament del projecte, un quart de control i test i un últim paquet de tancament o anàlisis. Aquests paquets són només la capçalera, ja que després els hem dividit en subpaquets més específics.

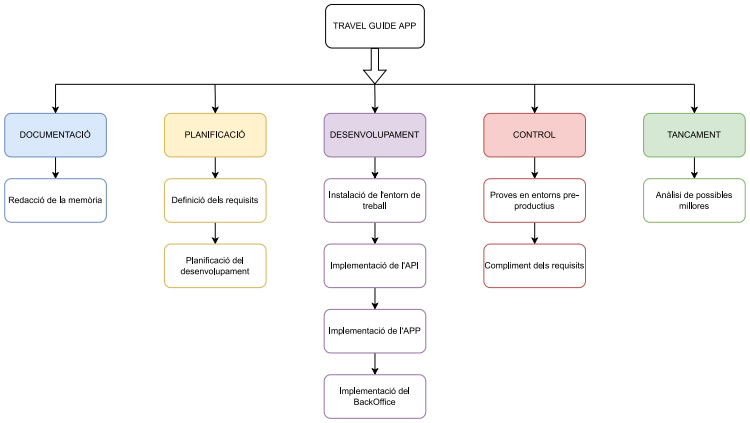

Figura 4.1: Proposta de paquets de treball pel projecte.

4.1.1 Descripció dels paquets de treball

A continuació hem fet unes taules de mostra de cada paquet amb les tasques a realitzar per a donar per fet aquell paquet junt amb una breu descripció d'aquest, quins lliuraments esperem aconseguir un cop acabat, el temps aproximat que pensem que trigarem a fer-lo i finalment el responsable del paquet.

NOM DEL PAQUET:	DOCUMENTACIÓ
DESCRIPCIÓ:	Recol·lecta de la informació i tasques fetes durant el projecte.
TASQUES A REALITZAR:	Redacció de la memòria del projecte.
LLIURAMENTS:	Memòria del projecte.
TEMPS:	14 setmanes.
RESPONSABLE:	Dan Spoiala i Hamza Saddouki

Taula 4.1: Paquet de Documentació.

NOM DEL PAQUET:	PLANIFICACIÓ
DESCRIPCIÓ:	Anàlisis de requisits i planificar desenvolupament
TASQUES A REALITZAR:	->Definir els requisits d'APP i API ->Planificar el desenvolupament ->Disseny de la base de Dades
LLIURAMENTS:	Document de requisits
TEMPS:	1 setmana
RESPONSABLE:	Dan Spoiala i Hamza Saddouki

Taula 4.2: Paquet de Planificació.

NOM DEL PAQUET:	DESENVOLUPAMENT
1:	Instal·lació de l'entorn de treball
2:	Implementació de l'API
3:	Implementació de l'APP
4:	Implementació del BackOffice

Taula 4.3: Taula resum dels paquets de desenvolupament.

NOM DEL PAQUET:	Instal·lació de l'entorn de treball
DESCRIPCIÓ:	Instal·lar i configurar els entorns de treball
TASQUES A REALITZAR:	Instal·lar, configurar i preparar els entors necessaris per el funcionament correcte dels serveis
LLIURAMENTS:	Entorn preparat tant per ús local com públic a internet
TEMPS:	1 setmana
RESPONSABLE:	Dan Spoiala i Hamza Saddouki

Taula 4.4: Paquet de desenvolupament 1, instalació dels entorns.

NOM DEL PAQUET:	Implementació de l'API
DESCRIPCIÓ:	Implementar totes les crides que necessiti l'APP
TASQUES A REALITZAR:	Progamar un API que serveixi a l'APP amb les crides per realitzar les diferents accions CRUD Disseny de la base de dades
LLIURAMENTS:	Documentació de les diferents crides disponibles amb els parametres necessari
TEMPS:	4 setmanes
RESPONSABLE:	Hamza Saddouki

Taula 4.5: Paquet de desenvolupament 2, desenvolupament del'API

NOM DEL PAQUET:	Implementació de l'APP
DESCRIPCIÓ:	Crear totes les pantalles i funcionalitats de l'APP
TASQUES A REALITZAR:	Crear pantalles sobre les quals ha de navegar l'usuari Crear funcionalitats que s'executarán al prémer un botó o realitzar una acció
LLIURAMENTS:	Aplicació funcional amb tots els requisits plantejats
TEMPS:	6 setmanes
RESPONSABLE:	Dan Spoiala

Taula 4.6: Paquet de desenvolupament 3, desenvolupament de l'APP

NOM DEL PAQUET:	Implementació del BackOffice
DESCRIPCIÓ:	Crear totes les pantalles i funcionalitats del BackOffice
TASQUES A REALITZAR:	Crear pantalles sobre les quals ha de navegar l'administrador Crear funcionalitats que s'executarán al prémer un botó o realitzar una acció
LLIURAMENTS:	Aplicació funcional amb tots els requisits plantejats
TEMPS:	2 setmanes
RESPONSABLE:	Hamza Saddouki

Taula 4.7: Paquet de desenvolupament 4, desenvolupament del BackOffice

NOM DEL PAQUET:	CONTROL
DESCRIPCIÓ:	Anàlisis de requisits sadisfets i provar el funcionament del producte final
TASQUES A REALITZAR:	->Analitzar compliments requisits ->Fer proves en entorns pre-productius. ->Arreglar errors percebuts durant les proves.
LLIURAMENTS:	Document d'implementació, resultats i conclusions.
TEMPS:	1 setmana
RESPONSABLE:	Dan Spoiala i Hamza Saddouki

Taula 4.8: Paquet de control, anàlisis i test

NOM DEL PAQUET:	TANCAMENT
DESCRIPCIÓ:	Anàlisis de millores futures del projecte i conclusions.
TASQUES A REALITZAR:	Anàlisis de millores.
LLIURAMENTS:	Document conclusions i millores.
TEMPS:	1 setmana.
RESPONSABLE:	Dan Spoiala i Hamza Saddouki

Taula 4.9: Paquet de tancament, anàlisis i millores

4.1.2 Matriu de traçabilitat

Un cop tenim definits els paquets de treball juntament amb la descripció de cada un d'ell el que fem és una matriu de traçabilitat per comprovar que realment hem tingut en compte tots els requisits. Com podem veure en la Taula 4.10 tots els nostres requisits són satisfets per l'APP, API o BackOffice.

	DESENVOLUPAMENT ENTORN DE TREBALL	DESENVOLUPAMENT API	DESENVOLUPAMENT APP	DESENVOLUPAMENT BACKOFFICE
RF1		X	X	
RF2		X	X	X
RF3		X	X	
RF4		X	X	
RF5		X	X	
RF6		X		X
RF7		X		X
RF8		X		X
RF9		X		X
RF10		X	X	
RF11		X	X	
RF12		X	X	
RNF1			X	
RNF2		X		
RNF3		X		
RNF4			X	
RD1	X			
RD2	X			
RD3	X			

Taula 4.10: Matriu de traçabilitat.

4.2 Temporalització

Per poder organitzar millor la feina a fer i per a fer un càlcul estimat del temps que trigaríem a acabar-la hem fet uns diagrames. Aquests diagrames són seqüencials i ens mostren una idea de les tasques distribuïdes en el temps.

4.2.1 Diagrama d'activitats

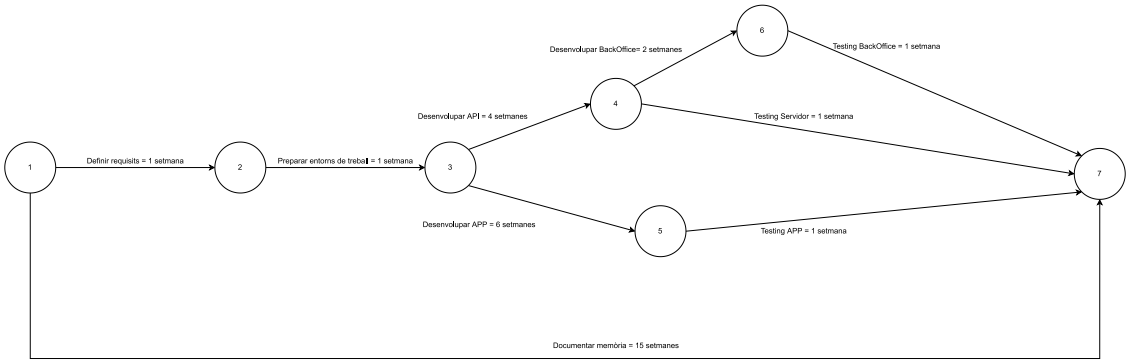

Figura 4.2: Proposta de diagrama d'activitats.

4.2.2 Diagrama de Gantt

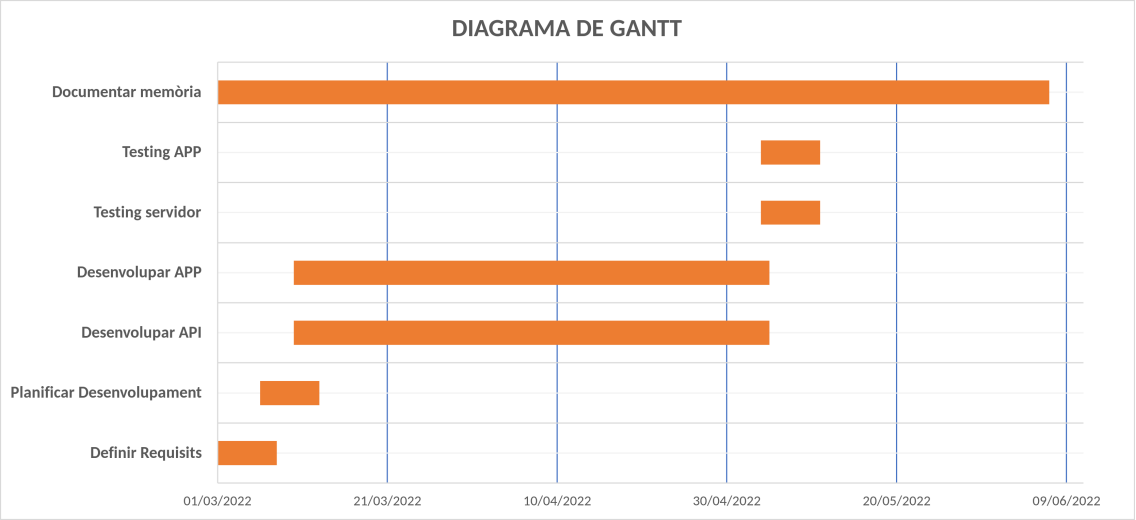

Figura 4.3: Proposta de diagrama de Gantt.

Enllaç del diagrama de Gantt per visualitzar-ho en alta resolució:
https://i.ibb.co/wN9C4q8/diagrama-Gantt.png

CAPÍTOL 5

Marc de treball i conceptes previ

Abans d'endinsar-nos en el projecte comentarem conceptes breus necessaris per entendre millor el treball i la feina que s'ha fet.

5.1 Aplicacions de turisme

Avui en dia la majoria hem passat per una aplicació de turisme com per exemple Booking, TripAdvisor, Kayak, etc. Però encara que totes tenen alguna característica extra que les fa diferenciar en el negoci respecte a les altres, en la base totes fan aproximadament el mateix. Si volem visitar una ciutat o país seleccionem aquest en el buscador i ens retorna un llistat de llocs a visitar, si volem reservar un hotel, vol o cotxe seleccionem unes dates en el buscador i ens retorna un llistat de vols o establiments on podem efectuar la reserva en aquelles dates. El que destaquen en aquests llistats que ens retornen és que els primers resultats solen ser els més famosos i moltes vegades no és que sigui perquè són els millors en el que fan sinó perquè hi ha acords entre companyes. Altrament, també de mentre naveguem per les webs ens trobem multitud d'anuncis de llocs que ens podrien interessar amb un descompte fictici per fer que hi entrem.

Amb aquest concepte si el que volem és organitzar un viatge hauríem de fer un seguit de passos:

- 1: Entrar en una de les aplicacions i buscar els llocs que volem visitar i si escau reservar entrada per aquests.
- 2: Entrar en una de les aplicacions i buscar vol i/o cotxe de lloguer.
- 3: Entrar en una de les aplicacions i buscar hotel.
- 4: Quan ho tinguem tot, comprovar les dates i veure que tot està disponible en les que teníem pensades.

Això fa que d'entrada ja detectem un problema afegit i és que fàcilment en el primer pas ens podríem encallar, ja que hauríem de fer una cerca a part sobre quins llocs són els més interessants per visitar en el país o ciutat i també planificar la ruta que els uneix per arribar a tots durant el nostre viatge. Llavors el que

havia de ser un viatge de vacances i descans es converteix en un procés laboriós de decisions per veure on anar i com anar.

5.2 Tecnologies i Entorns de desenvolupament

En l'actualitat tenim multitud de formes de desenvolupar aplicacions, podem fer aplicacions web amb entorns com Angular, React, WordPress que després s'adaptin a les pantalles de dispositius mòbils, podem fer una aplicació per Android o iOS fent servir l'entorn que ens proporciona cadascú dels sistemes operatius com és el cas d'Android amb Android Studio, iOS amb XCode o podem buscar una alternativa conjunta com és React Native o Flutter.

Pels servidors podem escollir diferents maneres d'implementar-los fent servir APIs (application programming interface), aquestes poden ser SOAP, RPC, WebSocket, REST, etc. són les encarregades de comunicar l'aplicació amb la base de dades i és on s'implementa gran part de la lògica de negoci. Per altra banda, les bases de dades també donen molt de joc, ja que poden ser SQL o noSQL, dins d'aquestes tenim multitud d'implementacions com Postgress, SQLServer, Oracle, MySQL, Elastic, etc.

Això fa a què l'hora d'elegir una opció no tinguem tant clar quina fer servir, però les empreses cada cop més busquen ser més eficients i treure més rendiment en aquest aspecte, és a dir si podem tenir un programador que ho faci tot per un sou X no pagarem cinc programadors diferents cadascú per un sou X. Per altra banda, els programadors cada cop més veuen la monotonia de certes aplicacions i la facilitat que els hi dona si creen un entorn on tinguin la major part feta i així implementar només les parts diferents. A partir d'aquesta idea es van creant els entorns de desenvolupament actuals, eines que ens donin la màxima facilitat i eficiència en implementar codi, però alhora que mantinguin també un alt grau de seguretat i eficiència en funcionar per aconseguir la millor experiència d'usuari.

A continuació farem una breu pinzellada més en profunditat dels entorns que hem fet servir en aquest projecte.

5.2.1 React Native

Learn once, write anywhere. Amb aquest eslògan ens presenten aquest entorn de desenvolupament, aquesta tecnologia ens vol resoldre un problema molt comú avui en dia i és concentrar la multitud d'entorns de desenvolupament necessaris per desenvolupar aplicacions en diferents sistemes operatius.

Com a exemple pràctic en el nostre cas l'aplicació mòbil hauria de ser compatible amb iOS i Android això faria que haguéssim de desenvolupar en com a mínim dos llenguatges de programació diferents i dos entorns diferents. Per part d'Android es fa servir Java o Kotlin i, per altra banda, iOS utilitza Project-C o Swift.

En comparació React Native ens ofereix programar en llenguatges com JavaScript o TypeScript al costat de CSS pels estils i després el compilador intern s'encarrega d'exportar el nostre codi a Kotlin per a Android i Swift en iOS. Això per les empreses és un benefici molt gran, ja que passen de tenir dos programadors, un per cada plataforma, a tenir un sol programador que doni suport a les dues.

Cada cop se sent més a parlar de React Native, punts clau de la seva fama sent el suport de Meta darrere del seu desenvolupament, però també una molt gran comunitat de programadors que aporten funcionalitats noves en cada versió com també multitud de llibreries externes que es poden instal·lar molt fàcilment mitjançant el sistema NPM. I és que tot el codi de React Native es troba a GitHub una plataforma que ens permet compartir codi entre desenvolupadors on qualsevol persona pot fer una petició d'aportació de millora o novetat al codi ja existent de React Native.

Tot-hi ser una eina molt potent React Native no aconsegueix unificar la totalitat de les dues plataformes, ja que cada una d'elles al final interpreta certes dades de forma diferent i de mentre codifiquem alguna vegada hem de dividir el codi entre les dues plataformes per a no tenir problema, això, però la gent de React Native ho té molt present i ens dona tot d'eines de selecció per poder agrupar el codi el màxim i només fer els canvis en les línies necessàries. Un exemple d'això és el comportament que té el "Notch", zona on es troba la càmera frontal del mòbil que cada cop és més popular fer-ho el més petit possible per a guanyar més pantalla, en aquest aspecte Android des del mateix sistema operatiu gestiona que el que mostrem a l'aplicació no quedi per sota la pantalla de mentre que en iOS ha de ser la mateixa aplicació que tingui en compte aquest aspecte i no fer les pantalles massa grans perquè ens quedin dades que no es vegin. Aquest aspecte, però ja cada cop està més afinat i ara existeix una vista anomenada "SafeAreaView"dins la llibreria de React Native que ens ajuda a tenir-ho sota control de manera molt fàcil.

5.2.2 NodeJS

Node.js és un entorn de temps d'execució de JavaScript, aquest entorn de temps d'execució en temps real inclou tot el que es necessita per executar un programa escrit en JavaScript en qualsevol plataforma.

Node.js ha sigut molt popular per diverses raons que a part de fer-ho molt accessible el converteix en una opció molt vàlida, a l'estar construït sobre la biblioteca de JavaScript de Google Chrome el converteix un dels entorns més ràpids, uns dels avantatges que atorga és la facilitat per crear mètodes asíncrons i gestionarlos de forma correcta.

A part de les característiques esmentades Node.js té el mòdul npm, un mòdul que permet instal·lar tota mena de llibreries, mòduls, aplicacions i línies de comandes que ens faciliten molt la feina i ens evita reinventar la roda i reaprofitar funcionalitats que ja estan definides per altres programadors.

Un altre dels avantatges que ens atorga el NodeJs és que al ser molt popular significa que també molts programadors han fet codi amb aquest entorn i com a tal és més fàcil trobar una solució a qualsevol problema que ens pot sorgir durant el desenvolupament de les nostres aplicacions.

5.2.3 Angular

Angular és un framework de JavaScript de codi obert el qual està desenvolupat amb TypeScript, és un dels frameworks per frontend més coneguts i utilitzats en el món de la programació. Aquest framework està dissenyat de forma que tot és modular de tal manera que crear qualsevol component, servei o mòdul que necessitem per la nostra web sigui una tasca molt senzilla.

Angular ens permet crear aplicacions de tipus Progressive Web Apps o Single Page Application amb una gran facilitat, això implica que és un framework amb alta escalibilitat.

Angular és un framework que ens atorga moltes facilitats a l'hora de crear aplicacions que funcionin bé tant per navegadors webs com aplicacions dins dels telèfons mòbils.

5.2.4 PostgreSQL

PostgreSQL és un gestor de bases de dades relacional que utilitza el llenguatge SQL, aquest gestor combina moltes característiques faciliten el tractament i emmagatzematge de dades de forma fàcil i segura.

PostgreSQL és un gestor gratuït, open source i altament escalable, et permet definir els teus propis tipus de dades, construir funcions personalitzades, i utilitzar les bases de dades amb altres llenguatges de programació sense recompilar la base de dades.

Ens interessa molt utilitzar aquest gestor sobretot per la versatilitat que ofereix per tractar dades geoespacials, ja que la nostra aplicació tracta principalment de tractar rutes i mapes ens interessa molt tenir una manera molt fàcil d'emmagatzemar i tractar aquestes dades, i en aquest cas PostgreSQL és un gestor preparat per aquestes necessitats, sobretot ens serveix molt per futures funcionalitats que es poden afegir.

CAPÍTOL 6

Requisits del sistema

Desenvolupadors: Hamza Saddouki i Dan Spoiala

6.1 Actors

Podem identificar dos actors principals, un seria l'usuari, el qual es registrarà a l'aplicació mòbil i interactuarà amb el sistema des d'ella, l'altre actor seria l'administrador el qual accedeix a la web des de la qual gestiona els usuaris i publicacions d'aquests.

6.2 Requisits

Els requisits són dividits en tres categories: [Alta],[Mitja],[Baixa] depenent de la seva importància en el producte final, sent els de categoria alta els obligatoris, els de categoria mitjana necessaris, però que si no hi són no hi ha massa problema i finalment els de categoria baixa que són complements als anteriors.

6.2.1 Requisits funcionals

- RF1: L'usuari ha de poder registrar-se en l'aplicació.[Alta]
- RF2: L'usuari ha de poder iniciar sessió en l'aplicació.[Alta]
- RF3: L'usuari ha de poder donar-se de baixa en l'aplicació.[Alta]
- RF4: L'usuari ha de poder publicar guies, opinions i rutes.[Alta]
- RF5: L'usuari ha de poder filtrar les publicacions per puntuació, zona i usuaris.[Alta]
- RF6: L'administrador ha de poder editar i/o eliminar una publicació.[Alta]
- RF7: L'administrador ha de poder afegir, editar o eliminar zones.[Alta]
- RF8: L'administrador ha de poder bloquejar o eliminar un usuari.[Alta]

- RF9: L'administrador ha de poder aprobar o rebutjar la publicació d'un contingut.[Alta]
- RF10: L'usuari ha de poder puntuar les publicacions d'altres usuaris.[Mitja]
- RF11: L'usuari ha de poder seleccionar publicacions com a favorites i visualitzar-les en un apartat diferent.[Mitja]
- RF12: L'usuari ha de poder seguir el perfil d'un usuari concret.[Baixa]

6.2.2 Requisits no funcionals

- RNF1: Un usuari nou sabrà usar totes les funcionalitats del sistema després de dues hores d'entrenament cometent menys de 5 errors.[Alta]
- RNF2: Quan hi hagi 20 usuaris accedint simultaniament al sistema el temps de resposta serà inferior a un segon.[Alta]
- RNF3: Davant un error del software el sistema no tardarà més de 5 minuts en restaruar les dades del sistema i tornar-se a posar en marxa.[Alta]
- RNF4: L'app pot ser executada en Android i iOS i conté la totalitat de codi compartit.[Alta]

6.2.3 Requisits de domini

- RD1: L'app vindra programada en React Native versió 0.67 [Alta]
- RD2: El backend vindra programat amb Express juntament amb NodeJs versió 16.14 [Alta]
- RD3: El backend tindrà un SwaggerHub desde el qual es podran veure les consultes disponibles i la documentació de cada una. [Alta]

6.3 Matriu de requisits

La matriu de requisits és una eina més que ens ajuda molt per organitzar-nos, ja que en ella podem veure les dependències entre els diferents requisits del nostre sistema.

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10	RF11	RF12
RF1												
RF2	X											
RF3	X	X										
RF4	X	X										
RF5												
RF6				X								
RF7												
RF8												
RF9				X								
RF10	X	X										
RF11	X	X										
RF12	X	X										

CAPÍTOL 7

Estudi i decisions

De forma resumida aquest treball dona lloc a tres productes que interactuen entre si, un primer producte és l'aplicació de mòbil que va destinada als usuaris, amb aquesta aplicació es podrà a part de les tasques més comunes com donar-se d'alta, iniciar sessió o modificar el perfil també crear publicacions de rutes i veure publicacions d'altres persones.

Per altra banda, tenim un BackOffice que no és més que una pàgina web destinada als clients del producte i des de la qual es pot controlar el bon funcionament d'aquest, ja que ens dona gestió directa sobre els usuaris i publicacions.

Finalment, però no menys important tenim el servei (API), aquest és l'encarregat de consultar la bases de dades i donar la informació a l'aplicació mòbil i el BackOffice per a poder funcionar.

Per a cada un dels productes s'ha escollit una tecnologia diferent per a fer el desenvolupament. A l'aplicació mòbil fem servir React Native, a l'API fem servir NodeJS i el BackOffice està fet en Angular. A la figura 7.1 podem veure un esquema de com està muntat el sistema, podem observar que la comunicació entre API, BackOffice i APP es fa a través de missatges que segueixen l'estàndard actual denominat HTTPs, això fa que segons ens indica l'estàndard els nostres missatges es transmetran de forma segura. Per altra banda, l'API es comunica a través de missatges SQL amb la base de dades central a on hi tenim totes les dades necessàries per a funcionar.

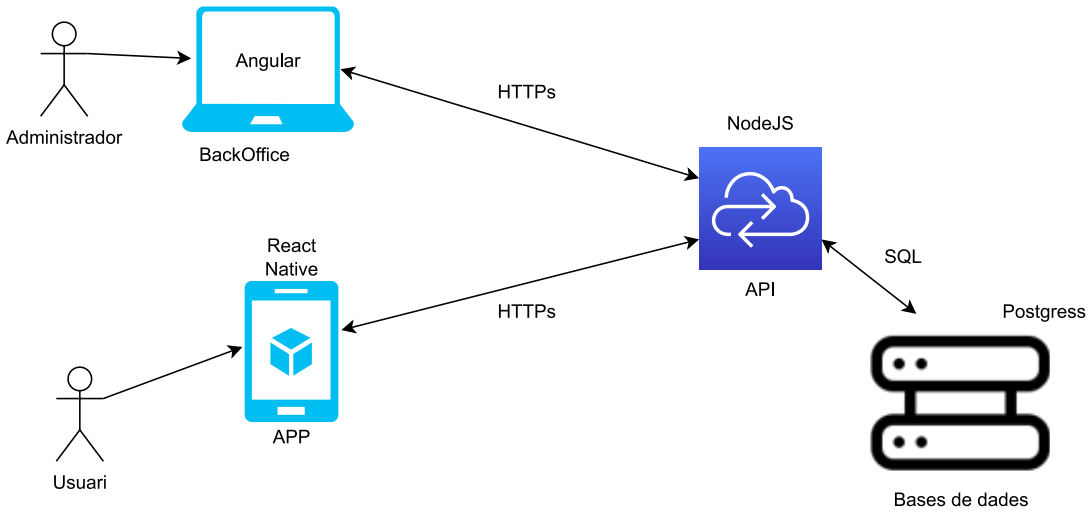

Figura 7.1: Esquema on es veu com esta construit el sistema.

7.1 API

- **Express**: Aquesta llibreria és essencial per tractar els diferents tipus de peticions que ens farà un client, **express** ens permet tractar les peticions i dades que rebem d'una forma accessible i senzilla, a part que ens permet definir validadors per les peticions i definir les respostes exactes a retornar.

- **Sequelize**: Necessitàvem alguna llibreria per definir els models i fer operacions CRUD sobre la base de dades, aquesta llibreria és una de les opcions que teníem per tractar la base de dades, vàrem decidir utilitzar aquest a causa de la quantitat de documentació i facilitat de definir els diferents models, tipus de dades i la facilitat que ens atorga per fer les diferents operacions sobre la base de dades sense utilitzar llenguatge SQL.

- **Dotenv**: Ja que tenim un repositori públic i volem evitar que altres usuaris tinguin accés a les nostres claus API dels possibles serveis externs que utilitzem llavors volem amagar aquestes claus, per suplir aquesta carència hem decidit definir variables d'entorn del repositori i al servidor i per poder accedir a aquestes variables és necessària aquesta llibreria.

- **Jsonwebtoken**: Volem integrar autentificació d'usuaris i la manera més accessible de controlar si un usuari està autentificat per l'api és utilitzant tokens, en aquest cas JWT a causa de totes les possibilitats que ens aporta, la llibreria que hem decidit triar és la que té més documentació al respecte.

- **Typescript**: A vegades deduir el tipus de dades que té una variable en JavaScript és molt ambigu, a causa d'aquest problema hem decidit utilitzar TypeScript el qual ens aporta un llenguatge tipat i ens facilita la lectura del nostre codi.

- **Bycryptjs**: Hem considerat molt necessària aquesta llibreria per guardar les contrasenyes dels usuaris de forma encriptada en la base de dades, ja que sempre és millor tenir més seguretat.

- **nodemailer**: Una de les coses que hem volgut implementar és que els usuaris puguin crear una nova contrasenya en cas que s'oblidin de la seva actual, per això és necessari que rebin un correu electrònic amb un enllaç a la pàgina per definir la seva nova contrasenya, per això hem instal·lat aquest mòdul el qual permet enviar correus electrònics als usuaris.

- **nodemailer-sendgrid-transport**: Aquest mòdul és una extensió del mòdul nodemailer, ens permet enviar mails amb el servei Sendgrid el qual és el que utilitzem nosaltres per enviar correus electrònics, aquest mòdul no és obligatori i es pot substituir per qualsevol altre servei.

- **winston**: Aquest mòdul és molt útil per generar logs, és important tenir en compte els possibles errors i situacions que poden succeir dins la nostra api sense que ens adonem, per això hem decidit afegir logs a l'api per tenir un control de les accions i sobretot per si ocorre qualsevol error tenir-ho en compte.

7.2 APP

- **Navegació**: Per a poder navegar dins les diferents pantalles de l'app hem fet servir diverses llibreries, ja que React Native no té navegació integrada, però sí que hi ha llibreries com les següents que tenen suport oficial.
  - react-navigation/native : Llibreria principal de navegació.
  - react-navigation/native-stack : Llibreria addicional que ens permet agrupar pantalles.
  - react-navigation/bottom-tabs: Llibreria addicional que ens permet també agrupar pantalles, però afegint-hi la barra de navegació a la part d'abaix de la pantalla per tenir accessos directes.
  - react-native-screens: Llibreria addicional que gestiona les pantalles de l'app i com aquestes es guarden en memòria del telèfon mòbil.

  Aquestes llibreries són molt fàcils d'utilitzar i estan molt ben documentades en les pàgines web de cada una, per integrar-les a l'aplicació només ha fet falta importar-les i crear un fitxer on hem donat un identificador a cada pantalla i les hem agrupat segons ens interessava que l'usuari accedís d'una forma o altra.

- **Mapes i geolocalització**: Per a poder mostrar els mapes dins l'app i que l'usuari pogués visualitzar-los i seleccionar rutes i punts dins d'aquests hem hagut d'importar diverses llibreries.
  - **react-native-map** : Mostra mapes de google.
  - **react-native-maps-directions**: Ens permet crear rutes i seleccionar punts dins dels mapes.
  - **react-native-community/geolocation**: Ens permet obtenir la localització de l'usuari.

  Aquestes han sigut les més difícils d'integrar, ja que han donat molts conflictes i necessitaven una bona configuració. Però un cop implementades el seu ús és molt fàcil i permet adaptar-les molt fàcilment a cada pantalla i funcionalitat que es vol. En aquest cas hem elegit aquestes, perquè són les més utilitzades i les que més suport reben de la comunitat.

- **Idiomes**: Pels idiomes hem fet servir la llibreria **I18n-js**, aquesta és molt simple de fer servir, ja que has de crear tants fitxers com idiomes tens amb el format: (clau: valor), on la clau serà l'identificador que fas servir i el valor la traducció en l'idioma respectiu, després es configura en iniciar l'app perquè carregui un idioma si ja s'ha escollit o sinó un per defecte i en les diferents pantalles es crida la llibreria amb la clau. En aquest cas tot-hi haver-hi moltes alternatives la més utilitzada és aquesta i tot-hi ser la més simple ja ens ha funcionat molt bé.

- **Emmagatzematge**: Aquestes són llibreries que ens donen accés a la memòria del telèfon quan necessitem guardar o recopilar alguna informació.

  - **react-native-secure-key-store**: En alguns casos necessitem guardar informació de l'usuari a la memòria, però per respectar la seva privacitat necessitàvem d'una manera segura per fer-ho, aquesta llibreria és molt lleugera molt fàcil de fer servir i encripta les dades dins el telèfon, llavors compleix amb les nostres necessitats, funciona com un mapa on l'indiques una clau i un valor que pots crear o pots obtenir de la memòria. Un dels seus inconvenients, però és que només pot guardar text.
  - **react-native-image-picker**: Com que amb la llibreria anterior no fèiem prou, ja que l'app necessitava gestionar imatges també hem afegit aquesta llibreria que ens dona accés a la galeria del mòbil d'on l'usuari pot seleccionar les imatges a pujar.

7.3 Backoffice

- **Bootstrap**: Aquesta és una llibreria que ens atorga la biblioteca completa de Bootstrap, això ens permet crear un disseny responsive a part de tots els components que ens proporciona, utilitzar aquesta llibreria ens evita haver de definir els estils de 0 i ens estalvia molta feina a l'hora de realitzar el disseny de la pàgina web.

- **Notiflix**: Aquesta llibreria ens proporciona un notificador pels usuaris, la considerem necessària pel fet que l'usuari no pot estar segur de si les seves accions dins el backoffice s'han realitzat degudament o ha sorgit un error sense cap notificació que ho indiqui.

- **FontAwesome**: Aquesta llibreria ens atorga una gran biblioteca d'icones gratuïtes, la considerem necessària perquè en afegir icones als botons o a certs llocs ajuda molt pel fet que proporciona un disseny més accessible per l'usuari, també és més fàcil identificar els diferents elements de la pàgina web.

CAPÍTOL 8

Anàlisi i disseny del sistema

Un cop teníem els requisits clars el primer pas era fer una anàlisi dels mateixos per veure quines solucions podríem plantejar per a complir amb aquests en cada un dels productes que hem desenvolupat durant el treball.

8.1 Bases de dades i API

Primer de tot hem començat per la base de dades, ja que vam pensar que tenir clar com guardaríem les coses ens ajudaria més endavant en el BackOffice i APP. El que teníem clar de bon principi és que tindríem Usuaris, Publicacions i que aquestes dues entitats havien de tenir interacció entre elles. També sabíem que les Publicacions havien de guardar la Ruta per mostrar en el mapa i un conjunt de textos i imatges que mostraríem com a contingut d'aquestes.

Un cop plantejades les entitats principals vam començar a desenvolupar les mateixes i pensar en què necessitàvem guardar de cada una d'elles i quina mena de relació tindrien.

De cada usuari a part de les dades bàsiques com el nom o el correu també necessitem guardar la foto de perfil, la puntuació que té a partir de les publicacions, el rol (Si és administrador) i finalment si està bloquejat, ja que volem bloquejar a un usuari en cas que no compleixi les regles de l'aplicació. De les publicacions ens interessa saber la puntuació, a quin país està dirigida la publicació i si algú l'ha reportat el nombre de reports que té la publicació. Dins de cada publicació volem guardar els diferents tipus de contingut que guardem com text, imatges o les rutes. A les rutes necessitem saber on comença i acaba la ruta i altres trajectes en mig d'aquests dos punts.

L'apartat de guardar rutes és el que més ens ha costat, ja que inicialment pensàvem que amb un conjunt de punts en el mapa seria fàcil representar un camí, però per com estaven implementats els mapes en l'app vam haver de donar una volta més i remodelar el disseny de la base de dades més endavant. Finalment, la base de dades utilitzada fa ús d'un disseny segons veiem en el Diagrama de la figura 8.1

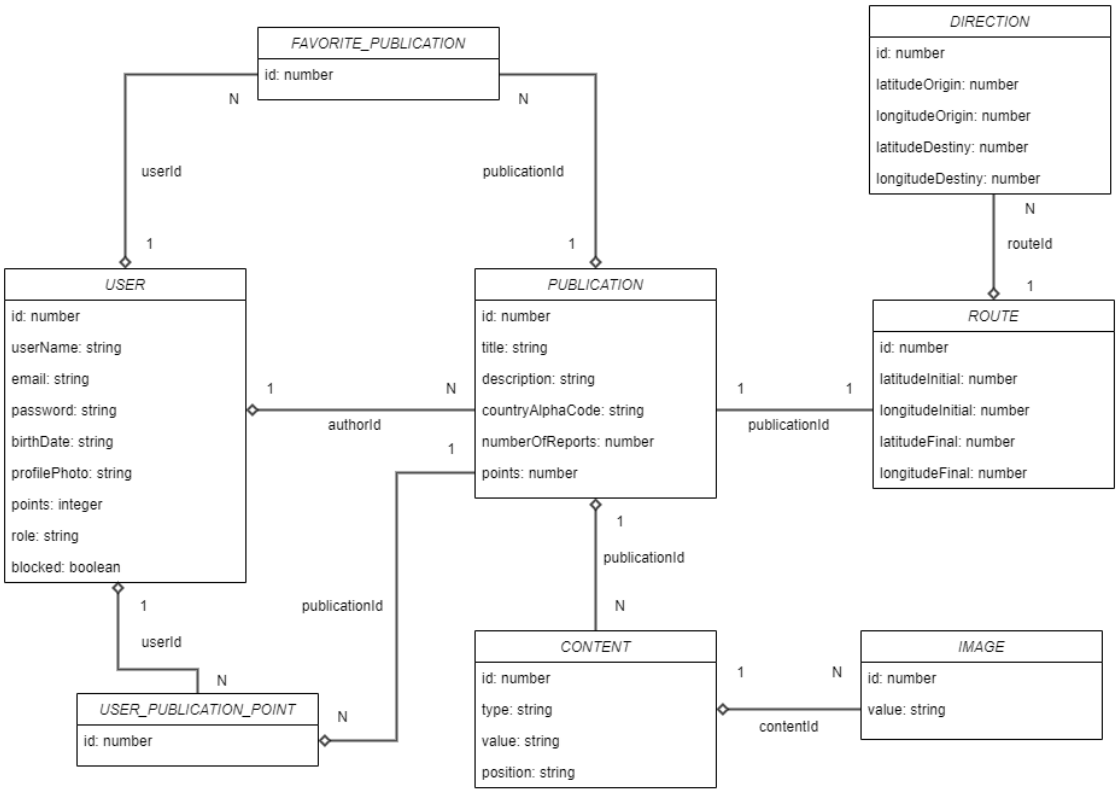

Figura 8.1: Diagrama UML de la base de dades

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/4gCtTS8/Travel-Guide-UML.png

Un cop tenim clar de manera lògica com s'estructura la base de dades el següent pas és pensar on guardaríem tota la informació i com hi accediríem des de fora d'aquesta. La resposta a com accedir des de fora és molt fàcil, ja que tothom avui en dia el que fa és una API (Application Programming Interface), que no és més que un servei que rep les peticions del client (en el nostre cas el BackOffice o l'APP), gestiona aquestes peticions aplicant lògica i consultant la bases de dades si fes falta i retorna una resposta que en cas d'èxit pot ser un correcte o la informació demanada o altrament en cas de succeir algun problema retorna un error amb el missatge associat. Per a resoldre la següent qüestió d'on guardaríem les dades necessitàvem algun servei que ens dones espai i accés públic al mateix i per això l'eina escollida és Heroku, ja que és un servei gratuït que tot-hi oferir poc espai inicial a menys que vulguis gaudir d'alguna subscripció per l'entorn de proves ens era més que suficient així que finalment tenim l'API a un servidor d'Heroku des d'on podem fer crides sense cap problema a un endpoint públic, a part de l'api també necessitem on guardar la base de dades i des d'on es connectarà l'API per obtenir totes les dades necessàries, en aquest cas també tenim la base de dades guardada a Heroku.

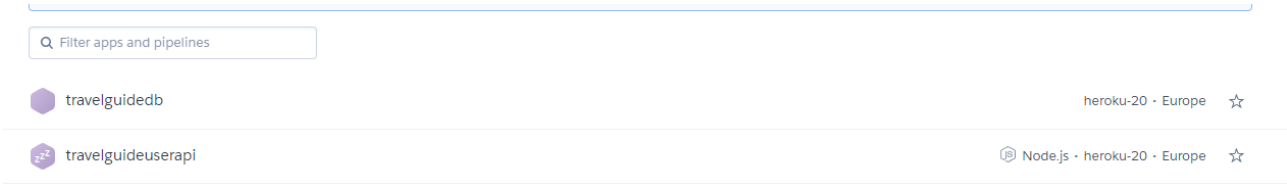

Figura 8.2: Servidors d'API i base de dades a heroku

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/h8z1b16/APPS-HEROKU.png

8.2 APP

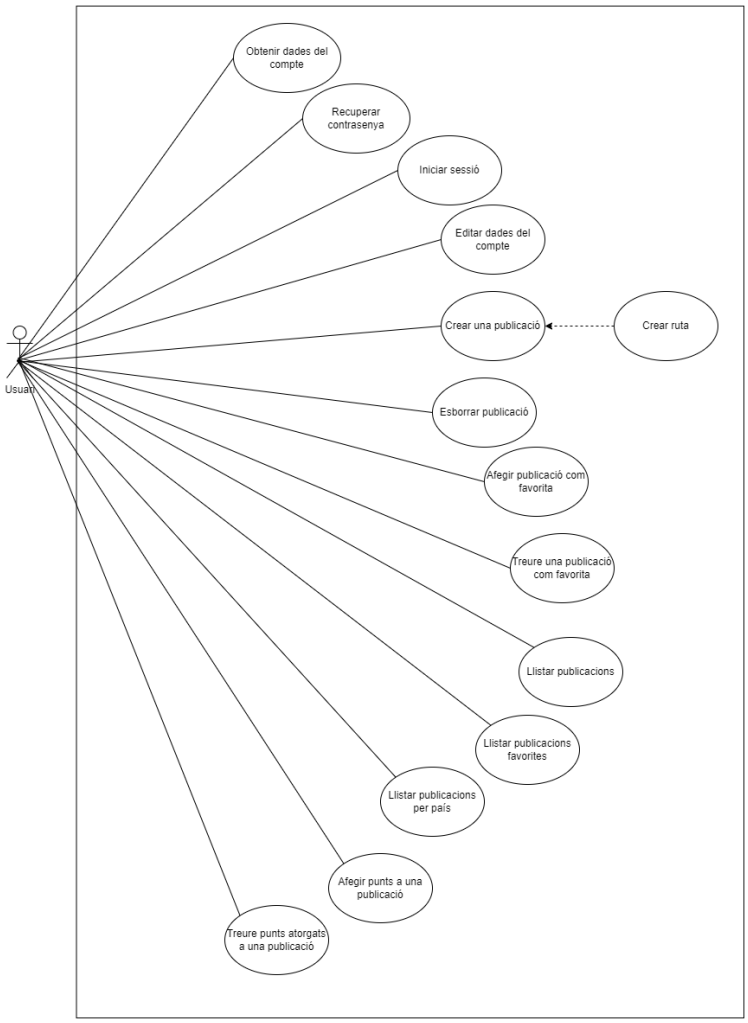

Figura 8.3: Casos d'us per l'app

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/pbTcC3v/Casos-Us-App.png

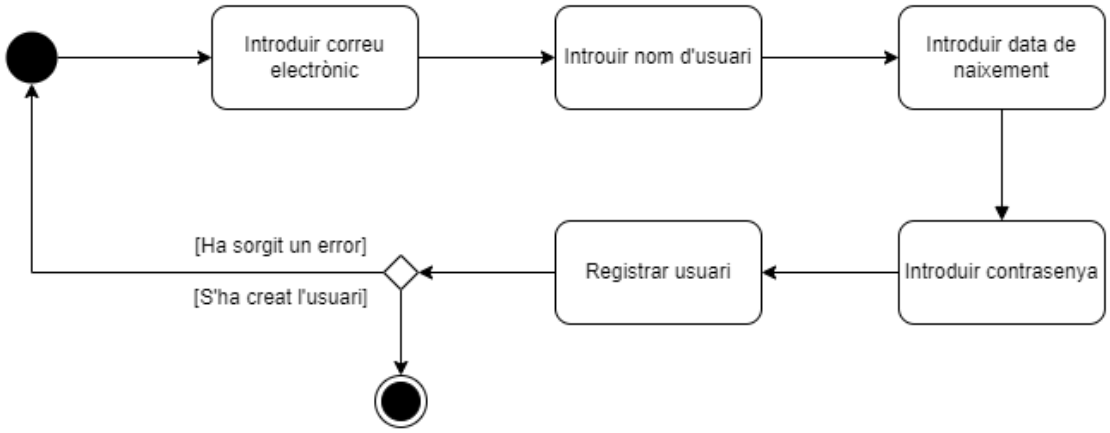

Figura 8.4: Diagrama d'activitats pel Registre d'un usuari

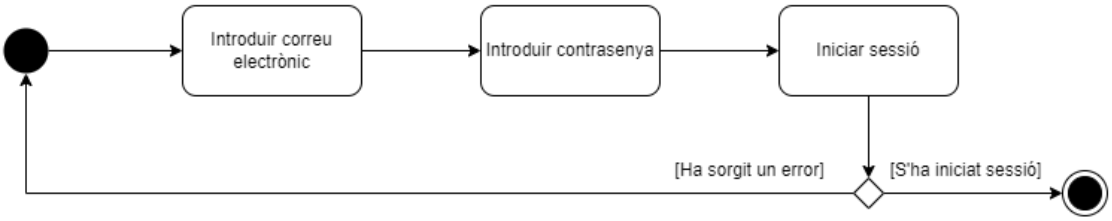

Figura 8.5: Diagrama d'activitats per l'Autenteificació d'un usuari

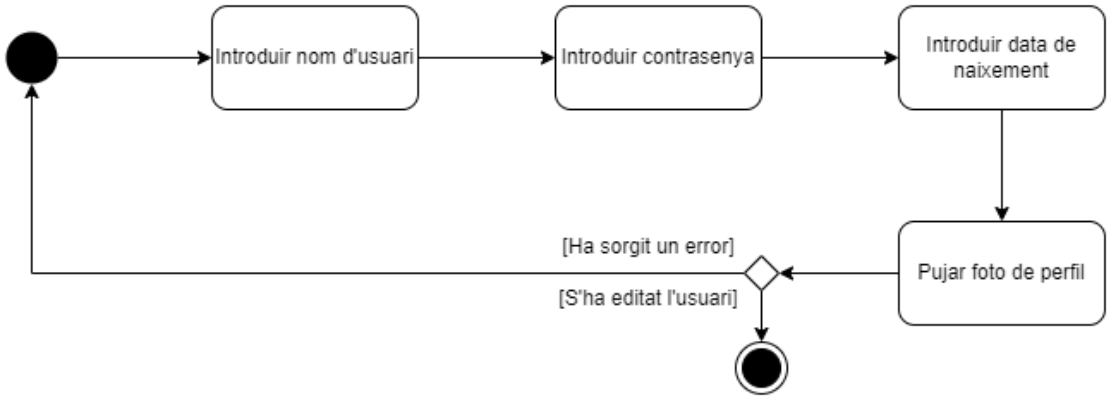

Figura 8.6: Diagrama d'activitats per l'Edició de les dades d'un usuari

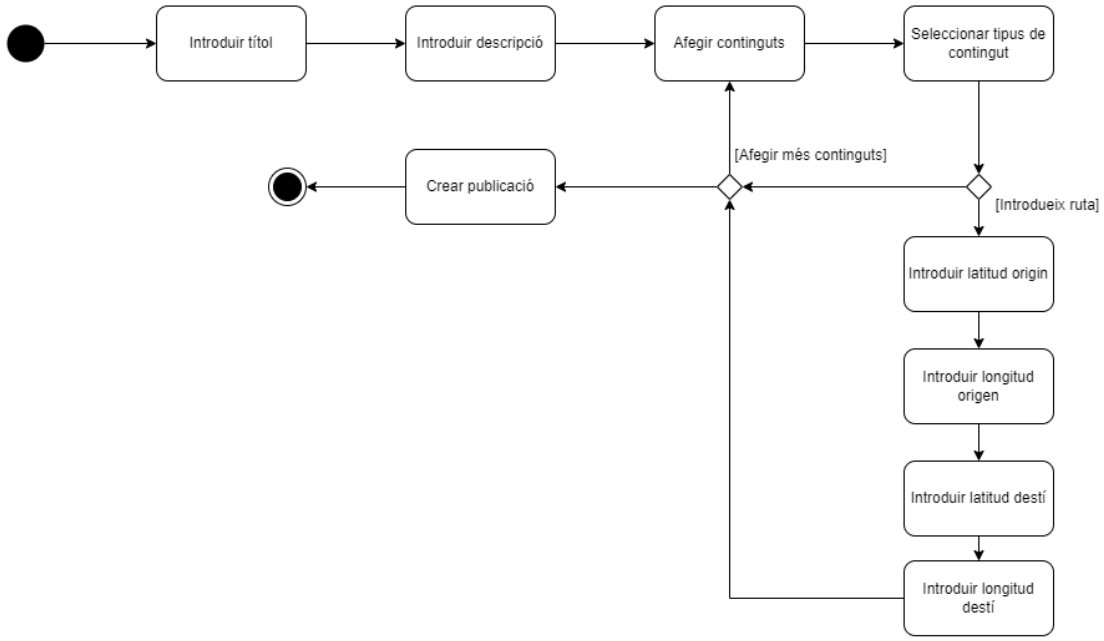

Figura 8.7: Diagrama d'activitats per la Creació d'una publicació

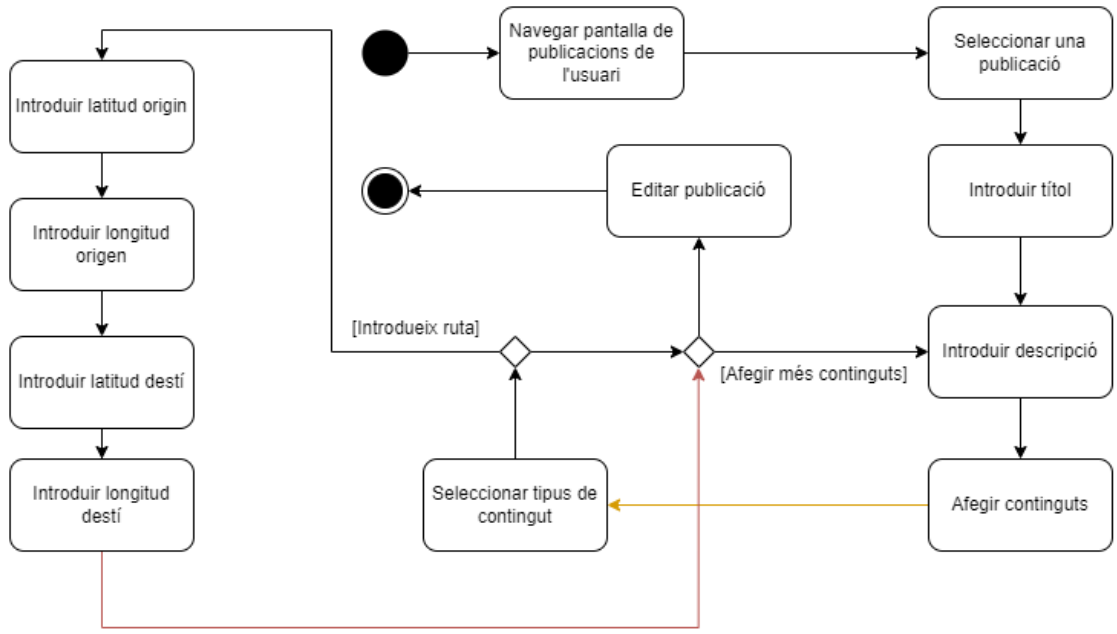

Figura 8.8: Diagrama d'activitats per l'Edició d'una publicació

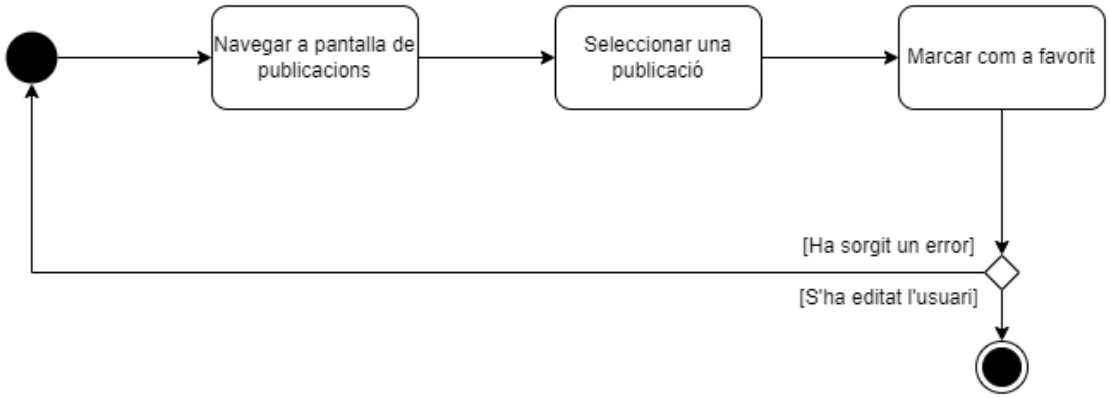

Figura 8.9: Diagrama d'activitats per Afegir una publicació en favorits

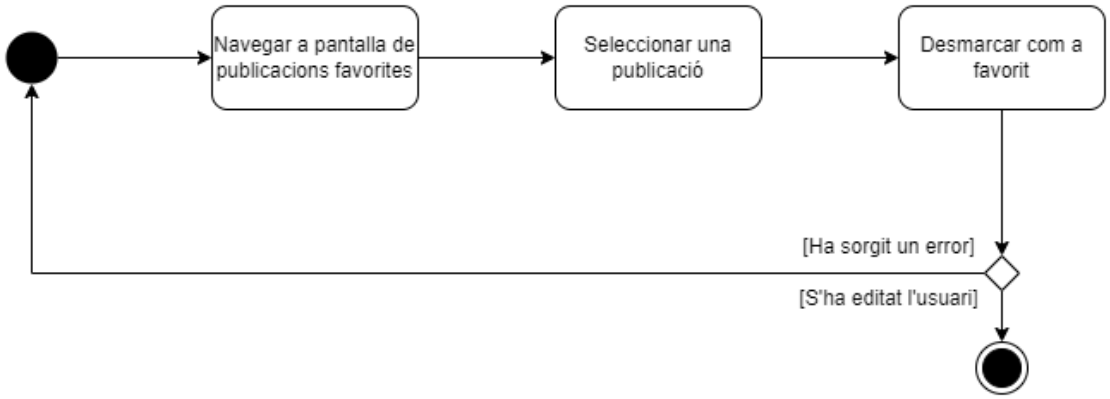

Figura 8.10: Diagrama d'activitats per Treure una publicació de favorits

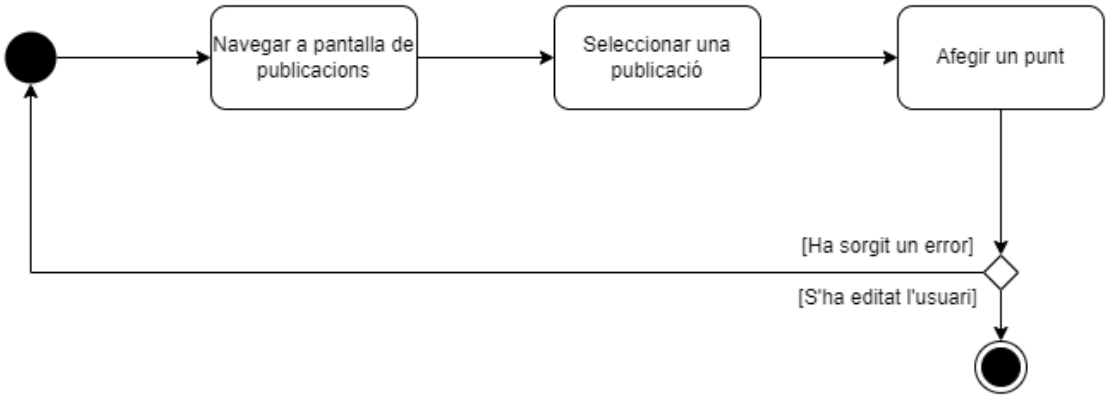

Figura 8.11: Diagrama d'activitats per Afegir punts a una publicació

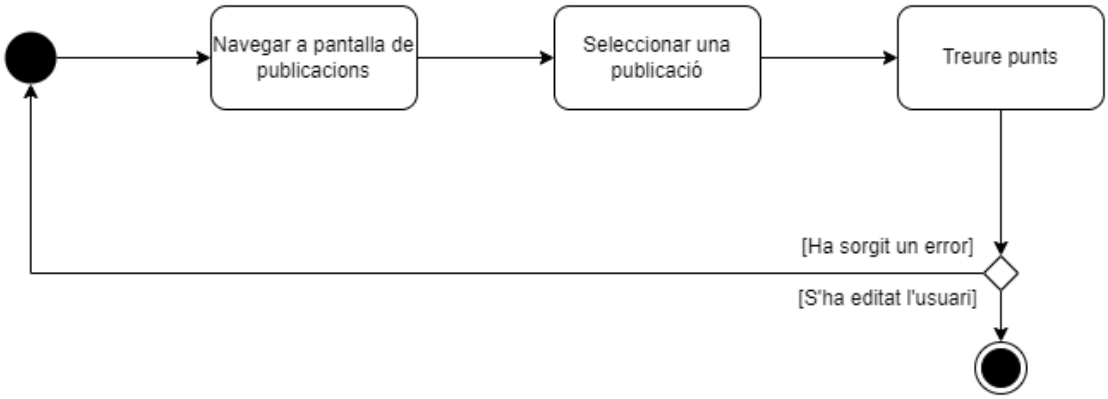

Figura 8.12: Diagrama d'activitats per Treure punts a una publicació

Per a l'APP abans de posar-nos a treballar en el projecte primer havíem de crear un partint de les directives de React Native i entenent qui és l'esquema pel qual es regeix el mateix.

Per crear un projecte de React Native un cop tenim tot l'entorn necessari instal·lat tal com mostra la guia és tan fàcil com executar la comanda **npx react-native init [NOM PROJECTE]** i això ens genera una carpeta amb totes les dependències bàsiques per a desenvolupar. Aquest projecte, però és molt bàsic i el primer que hem de fer si volem mostrar més d'una pantalla és instal·lar totes les llibreries de navegació que hem mencionat anteriorment, ja que és el que ens permetrà tenir més pantalles i que l'usuari pugui canviar entre elles.

Aquestes pantalles que hem mencionat es construeixen a partir de components tan directament de React Native com components nostres propietaris o de llibreries que afegim, llavors l'esquema que ens queda d'esquelet inicial de l'APP és el que veiem a la figura 8.3.

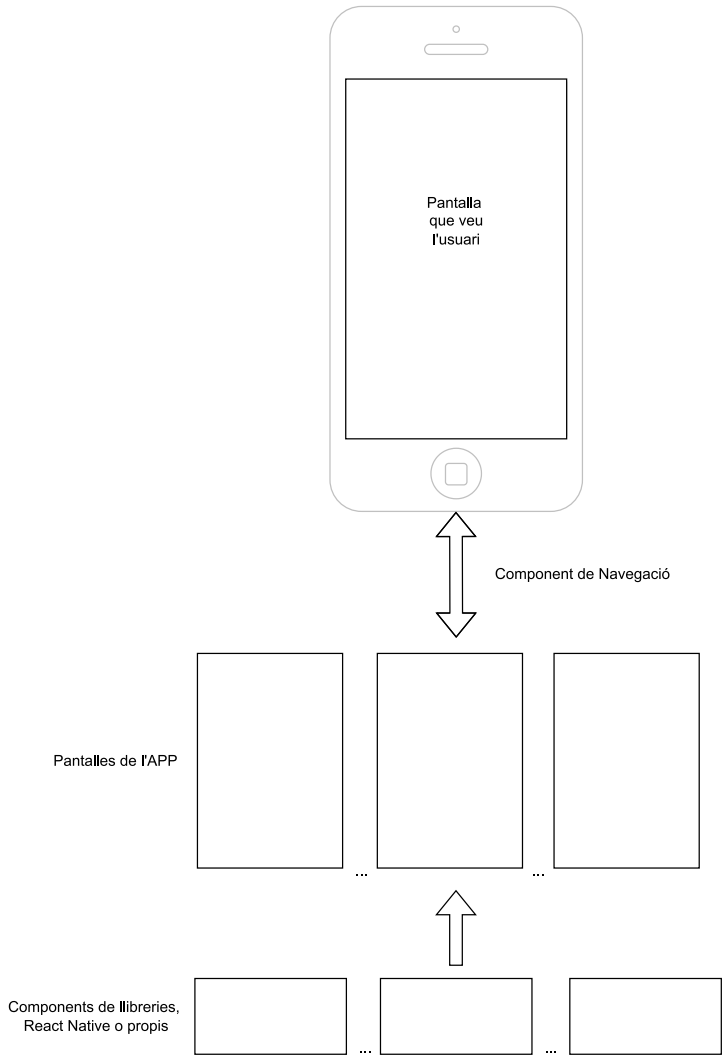

Figura 8.13: Esquema bàsic de React Native.

Un cop tenim plantejada la base ja podem definir les interfícies de les pantalles de l'APP, com cap dels dos components del grup sap massa de disseny el que hem fet és veure com solucionen altres apps del mercat els problemes que ens anàvem plantejant. Per tant, de forma molt esquemàtica així quedarien les primeres interfícies:

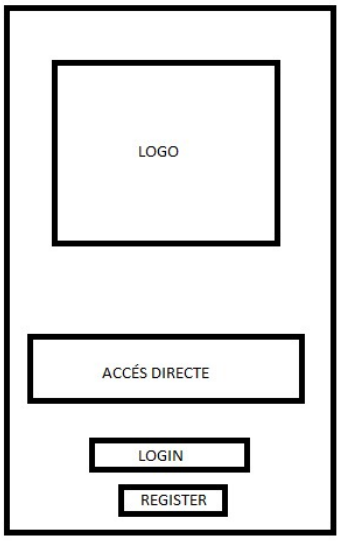

Figura 8.14: Esquema de la Home.

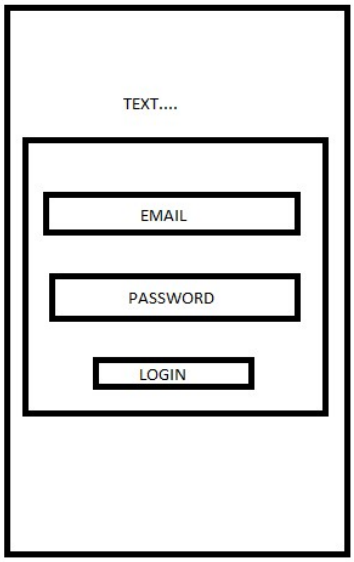

Figura 8.15: Esquema del Login.

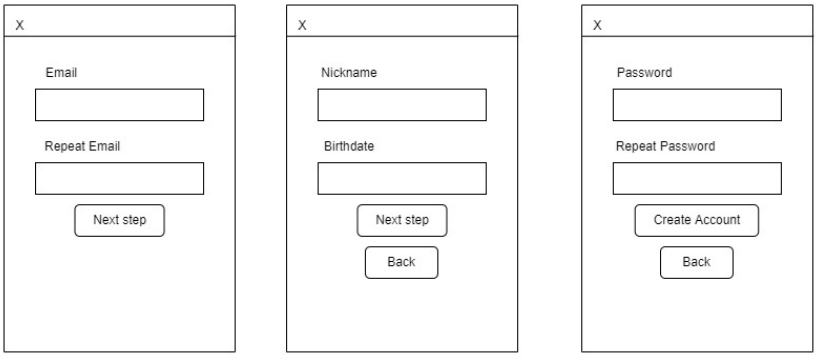

Figura 8.16: Esquema del procés de Registre.

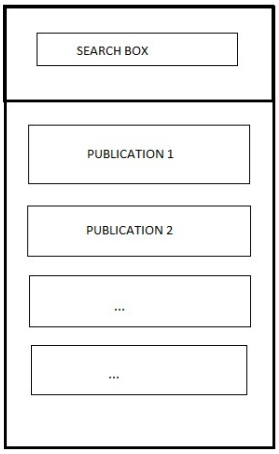

Figura 8.17: Esquema del llistat de publicacions.

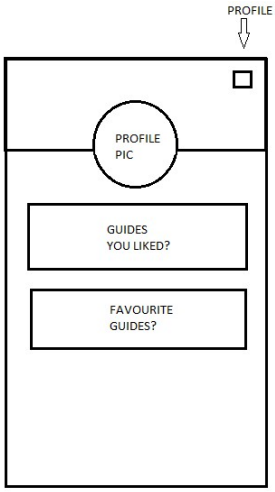

Figura 8.18: Esquema del Perfil.

Més endavant veurem com aquestes interfícies inicials al final es van veure alterades al moment d'implementar, ja que vam detectar informació que ens faltava per mostrar o que el disseny plantejat no acabava d'encaixar amb la interfície, al mateix temps que hem afegit pantalles que hem fet durant el projecte i no hem fet interfície anterior a implementar-les, pel fet que la persona que feia l'APP era una sola i anava implementant i dissenyant al mateix temps.

CAPÍTOL 9

Implementació i proves

9.1 API

El nostre plantejament sobre l'api ha sigut d'un api sobre la qual es puguin realitzar accions tant d'usuari client com usuari administrador, per gestionar aquesta diferència de rols hem integrat un sistema d'autentificació amb tokens jwt, aquest sistema permet afegir a un token generat informació addicional, un exemple seria quin usuari és el que està fent l'acció, d'aquesta manera podem fer una cerca a la base de dades i deduir quin rol està utilitzant, en el nostre cas hem utilitzat un altre mètode que és més eficaç, el qual és dintre el token introduir la informació del rol d'usuari i, per tant, no fa falta fer cap consulta a la base de dades i afegeix un plus d'eficàcia a l'api.

Hem pensat que seria una bona idea afegir registres al nostre api per quan passi algun error ens puguem assabentar que aquest error existeix i com s'ha produït, per així nosaltres poder reproduir el mateix error i localitzar des d'on pot provenir i corregir-ho l'abans possible, en aquest cas hem trobat que una bona manera seria implementar un sistema ja preparat com el servei Papertrail.

Per realitzar les proves corresponents a l'API hem utilitzat el programa de Postman, el qual ens permet provar les diferents crides que tenim definides, en cas d'error mostrem l'error per consola i el servidor d'Heroku ens permet accedir als logs i revisar que és el que ha fallat exactament, això ens ha permès identificar amb molta rapidesa quin era l'error i on es trobava el problema, a part això ens permet revisar diferents casuístiques definides que tenim i revisar si el comportament que hem definit està funcionant correctament i els usuaris no tenen permès realitzar accions no contemplades.

Una de les parts que considerem importants és que guardem les contrasenyes de forma encriptada en la base de dades amb el mòdul byCrypt.

```
const salt = await genSalt(10);
const hashedPassword = await hash(body.password, salt);

const newUser = {
    userName: body.userName,
    email: body.email,
    password: hashedPassword,
    birthDate: body.birthDate
};

const user: any = await UserModel.create(newUser);
user.save();

const token = sign({
    name: user.userName,
    id: user.id
}, process.env.TOKEN_SECRET);

res.status(200).json({
    user: user,
    token: token
});
```

Figura 9.1: Codi per generar contrasenya encriptada

Una de les coses recurrents és que necessitem saber quin usuari està fent la petició, per tal d'evitar repetir codi hem creat una funció per obtenir l'usuari loguejat a partir del token que ens envia.

```
export const getCurrentUserByToken = async( req = request, _ = response ): Promise<JWTUser> => {
    let token = '';
    const bearerHeader: string = req.body?.token || req.query?.token || req.headers["authorization"];

    if (bearerHeader.includes('Bearer')) {
        const bearer = bearerHeader.split(' ');
        token = bearer[1];
    } else {
        token = bearerHeader;
    }

    let splitToken = token.split('.');
    let toPayload = splitToken[1];
    const payload = atob(toPayload);
    const user: JWTUser = JSON.parse(payload);
    return user;
}
```

Figura 9.2: Codi per obtenir l'usuari a partir del token

Com que ens interessa saber si el client que ha realtizat una petició està autoritzat a fer-la hem de comprovar el token que ens envia i revisar si és vàlid, per no tenir que repetir la comprovació hem creat un mètode que fa de middleware per les peticions.

```
export const verifyToken = ( req = request, res = response, next ) => {
    const bearerHeader: string = req.body?.token || req.query?.token || req.headers["authorization"];


    if (!bearerHeader) {
        return res.status(401).json({
            message: "És necessari loguejar-se per realitzar aquestes operacions"
        });
    }

    let token = '';

    if (bearerHeader.includes('Bearer')) {
        const bearer = bearerHeader.split(' ');
        token = bearer[1];
    } else {
        token = bearerHeader;
    }

    try {
        const decoded: string | JwtPayload = decode(token);
        if ((typeof decoded !== "string") && Date.now() >= decoded.exp * 1000) {
            return res.status(401).json({
                message: "El token ha caducat"
            });
        }

        const verified = verify(token, process.env.TOKEN_SECRET);
        if (!verified) {
            return res.status(401).json({
                message: "Aquest token és invalid"
            });
        }

        const signed = sign(token, process.env.TOKEN_SECRET);
        if (!signed) {
            return res.status(401).json({
                message: "Aquest token és invalid"
            });
        }
    } catch (err) {
        return res.status(401).json({
            message: "Aquest token és invalid"
        });
    }
    return next();
```

Figura 9.3: Codi per comprovar l'autentificació del client

9.2 APP

Des d'un inici ja teníem una breu idea del que era React Native, ja que n'havíem sentit a parlar i havíem vist alguns exemples, però quan ens vam posar a codificar l'APP ens vam adonar que realment no era tan fàcil d'utilitzar com semblava. Així que el primer dels problemes va ser familiaritzar-se amb l'entorn, cosa que va provocar que les primeres tasques anessin molt lentes i que més endavant es repassés l'APP per ajustar codi inicial no tan ben fet. Aquest punt, però no va ser massa difícil de resoldre, ja que les nocions bàsiques per a començar a fer pantalles i components són molt fàcils d'entendre.

Un altre dels percals inicials és la guia que has de seguir per a instal·lar l'entorn, ja que inicialment semblen pocs passos, però tots són importants fer-los bé si no més endavant surten errors inesperats. Vam haver de repetir algun pas de la instal·lació diverses vegades perquè l'ordinador no agafava la configuració adequada.

Quan teníem tot preparat per desenvolupar vam començar afegint la navegació que codificar aquesta ens va semblar molt senzill inicialment, però més endavant teníem problemes en navegar entre pantalles, ja que ens costava entendre el concepte d'un stack container o un tab container, i és que el component de navegació et permet agrupar diferents pantalles en contenidors i així aïllar aquestes de les altres. Això ho hem fet servir en el nostre benefici per a aïllar certes pantalles on té accés un usuari amb sessió iniciada de les pantalles on té accés qualsevol usuari.

Un dels altres problemes que vam tenir és que per exemple en fer crides a l'API no enteníem per què a vegades funcionava i altres no i ens vam adonar mirant tutorials que era perquè React Native té funcions que són síncrones, és a dir executen el codi tal com l'hem escrit nosaltres línia darrera línia a mesura que es van completant i d'altres que són asíncrones, és a dir executen el codi sense esperar cap resultat. Per resoldre això vam descobrir les Promises, una eina que ens dona React Native per dir-li que ha d'esperar el resultat abans de continuar executant.

El problema més gran que hem tingut amb l'APP és el fet de mostrar mapes i rutes, vàrem cercar diferents llibreries i opcions, una de les llibreries que vàrem trobar va ser la d'OpenLayers, el cas és que aquesta llibreria ens va donar molts de problemes, ja que té molts components que són exclusius per navegadors i no per aplicacions de mòbil, després de cercar molt vàrem trobar la llibreria de

react-native-maps i react-native-maps-directions, aquestes semblaven més prometedores, perquè disposaven de més documentació, en instal·lar-les no vam tenir massa problemes, però després vam haver de seguir un tutorial de configuració perquè funcionessis correctament, aquí és on vam dedicar temps també en buscar totes les dades que ens feien falta. Un cop teníem tot compilant i funcionant també vam haver de donar voltes al disseny, ja que a vegades no es veia bé el mapa, altres no ens sortien els marcadors a sobre, etc. Finalment, a força de fer proves vam aconseguir que tot funciones.

Amb els mapes vam aprendre que la idea d'instal·lar massa llibreries externes amb el servei NPM no era molt bona idea, ja que ens podíem trobar en situacions similars. Per això vam decidir fer servir el mínim de mòduls externs i que tot el que es pogués desenvolupar de forma pròpia es fes.

Un codi que ens va costar de treure és com gestionar les imatges i text perquè sempre mantinguin el mateix ordre, ja que la bases de dades no assegura cap ordre encara que ho enviem ordenat en guardar. Per fer-ho hem implementat la posició (figura 9.14). El que fem és guardar juntament amb l'ítem la seva posició i cada cop que guardem anem incrementant. Això fa que quan tornem a rebre el llistat ens sigui fàcil ordenar les dades segons la seva posició.

```
const onSave = (
    _image: { uri: string; base64: string },
    _text: string | null,
) => {
    let _position = position;
    if (_image) {
        content.push({
            type: 'image',
            value: null,
            position: _position,
            image: {
                uri: _image.uri,
                value: _image.base64,
            },
        });
        _position++;
    }
    if (_text) {
        content.push({
            type: 'text',
            value: _text,
            position: _position,
            image: null,
        });
        _position++;
    }
    setPosition(_position);
    setShowAddModal(false);
};
```

Figura 9.4: Codi que guarda l'imatge i text carregats de forma ordenada.

Un altre codi interessant (figura 9.15) és com obtenim les imatges, ja que inicialment ens va semblar difícil, però un cop entès com funciona la llibreria va ser un moment. El que fem és cridar la llibreria amb un conjunt de paràmetres on li diem que volem carregar un sol element, de tipus foto (això fa que no permetem carregar gifs per exemple) i quan aquesta te el resultat ens el retorna, nosaltres per mantenir-nos a l'espera fem servir el "await", un cop tenim el resultat ja el podem guardar, en aquest cas ens interessa el format Base64 una codificació que es fa servir molt en l'actualitat per a comprimir ítems.

```
const addImage = async () => {
    try {
        const result = await launchImageLibrary({
            selectionLimit: 1,
            mediaType: 'photo',
            includeBase64: true,
        });
        if (result.assets) {
            setImage(result.assets[0]);
        }
    } catch (err) {}
};
```

Figura 9.5: Codi que guarda l'imatge

Els mapes van ser els més difícils d'implementar, però un cop funcionant el seu us és molt fàcil, per exemple per guardar la posició del mapa a on fa click l'usuari només és necessari estar escoltat un esdeveniment d'aquest. En aquest cas el mapa ens ocuparia tota la pantalla per l'estil que li donem i a més s'obriria en la regió inicial que li passem per paràmetre, en aquest cas són coordenades d'estats units.

```
<MapView
    provider={PROVIDER_GOOGLE}
    style={{
        width: Dimensions.get('window').width,
        height: Dimensions.get('window').height,
    }}
    initialRegion={{
        latitude: 37.78825,
        longitude: -122.4324,
        latitudeDelta: 0.0922,
        longitudeDelta: 0.0421,
    }}
    onPress={onLocationSelect}
>
    {marker && <Marker coordinate={marker} />}
</MapView>
```

Figura 9.6: Vista del Mapa que mostrem amb la funció onPress que ens retorna un esdeveniment.

```
const onLocationSelect = (event: MapEvent) => {
    setMarker(event.nativeEvent.coordinate);
};
```

Figura 9.7: Codi que guarda internament cada punt per si el fem servir més endavant.

```
<Button
    title={I18n.t('save')}
    disabled={!marker}
    onPress={() => {
        saveMarker(marker);
    }}
    color={colors.primaryOrange}
/>
```

Figura 9.8: Botó que fa click l'usuari i que retorna el punt que tenim seleccionat actualment a la pantalla 3 de crear nova ruta.

9.3 Backoffice

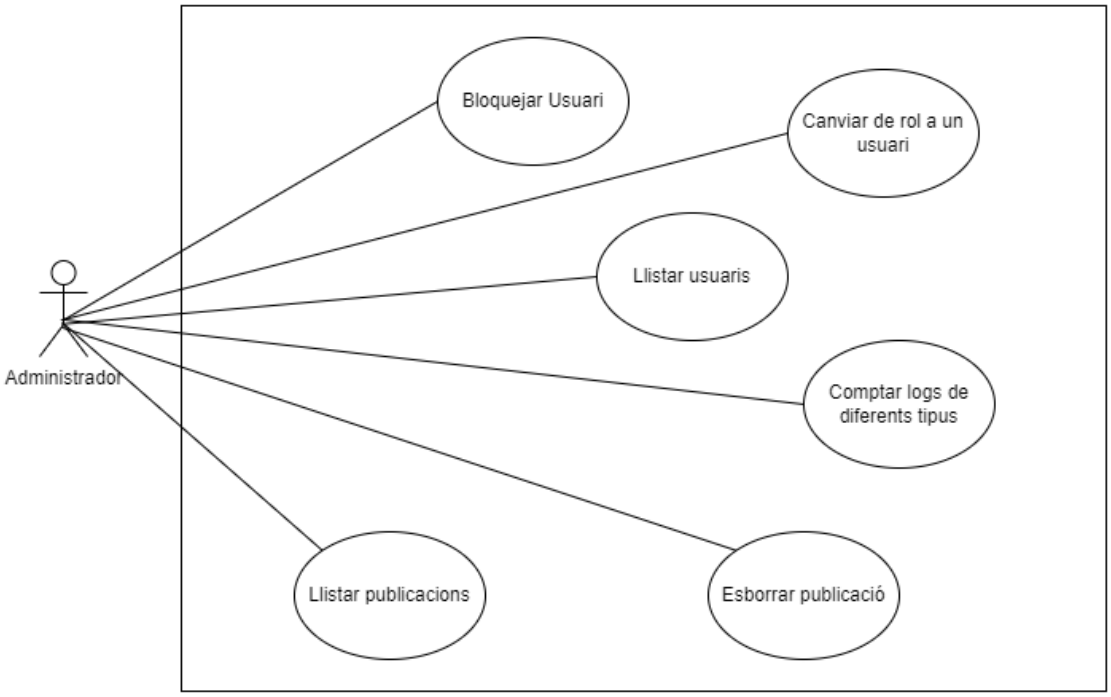

Figura 9.9: Casos d'us pel backoffice

L'objectiu d'implementar un backoffice ha sigut el de poder gestionar tant els usuaris com les publicacions que generen, considerem que tota aplicació on els usuaris puguin crear contingut s'ha de controlar i posar certs límits, sobretot quan és una aplicació que té un objectiu i un enfocament tan específic.

Principalment, hem tingut la idea de mostrar un llistat d'usuaris i també un llistat de publicacions, en el llistat d'usuaris principalment s'han de poder efectuar accions com bloquejar-los perquè no puguin accedir més a l'aplicació, també una altra acció és la de desbloquejar-los en cas que l'usuari ADMIN s'hagi equivocat o revoqui la seva decisió, i finalment en cas de voler tenir més d'un usuari administrador que pugui fer gestions es pot canviar el rol de qualsevol usuari a administrador o a usuari client.

Les proves del backoffice han tractat majoritàriament sobre revisar que només els usuaris amb permisos d'administrador puguin accedir-hi i realitzar les diferents accions, i que totes les accions que es realitzen no tinguin cap error i es puguin realitzar sense cap problema.

Hem decidit utilitzar la funcionalitat de GitHub anomenada Github Pages, la qual et permet a partir d'una organització allotjar una web de forma gratuita.

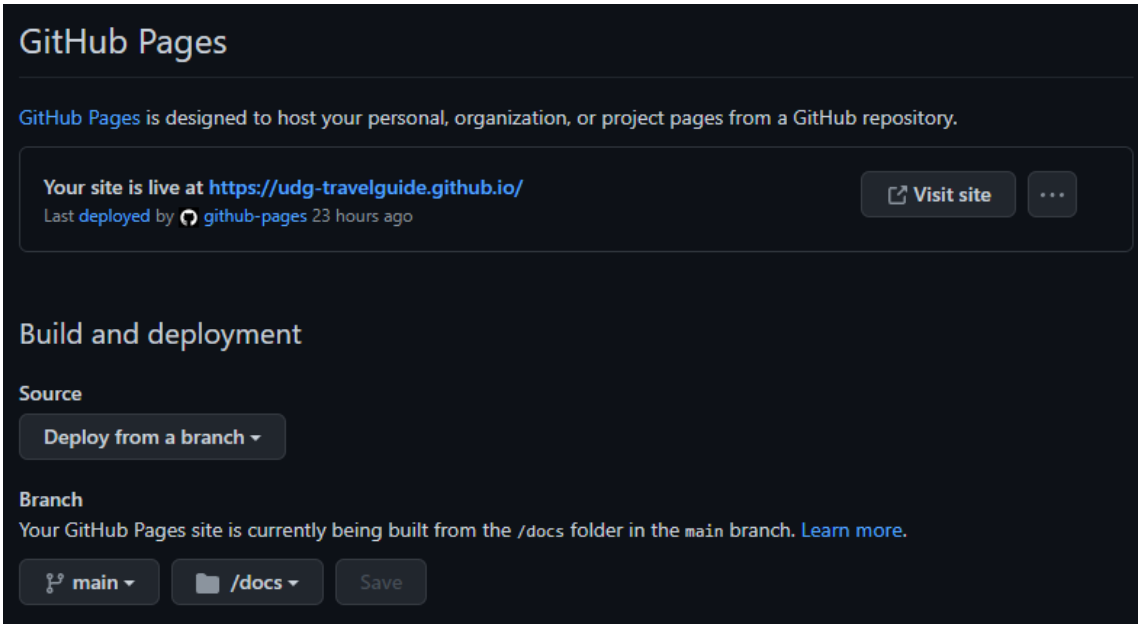

Figura 9.10: Configuració del repositori per allotjar la web

Per facilitar-nos el treball de estar compilant i desplegant l'api i el backoffice cada cop que fèiem un canvi significatiu hem decidit utilitzar el sistema Continuous Integration, tant el Github com Heroku ens facilita molt la vida a l'hora de fer aquest procés, en el cas de l'API hem integrat el repositori de github amb el servidor de heroku fent que cada cop que es fa un push i es compila correctament es fa automàticament un deploy a heroku.

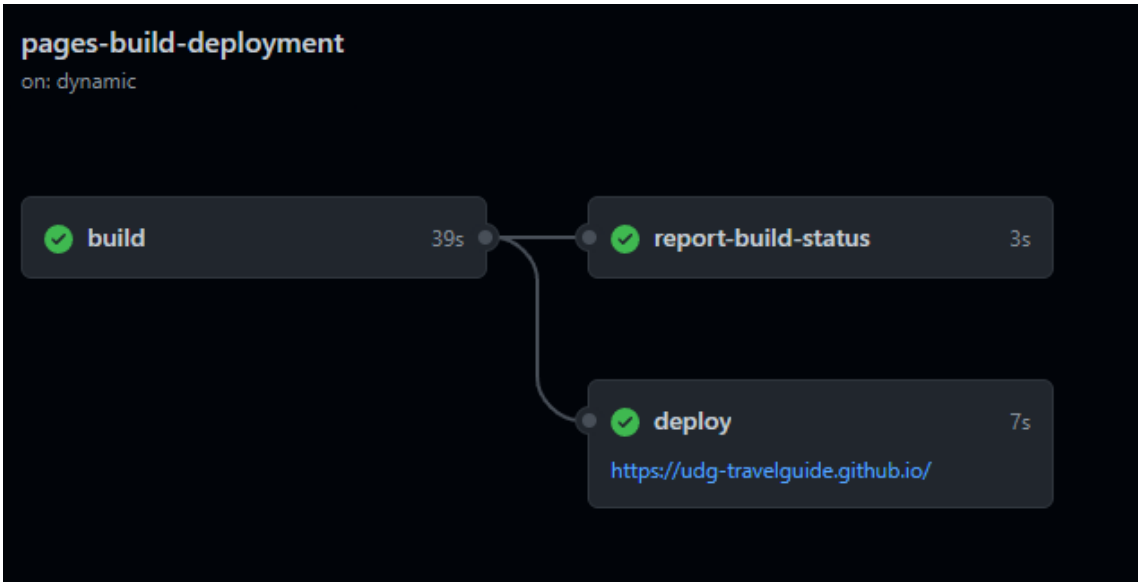

Figura 9.11: Fluxe automatitzat de backoffice

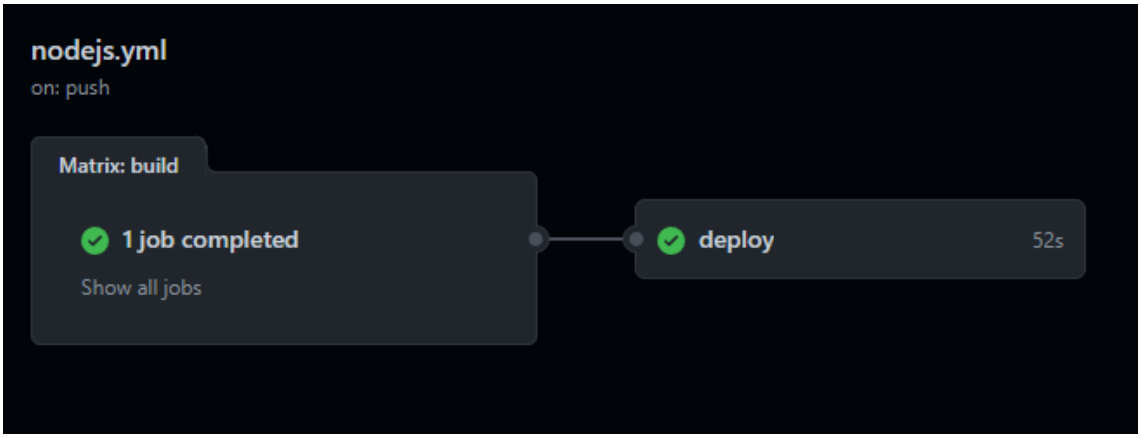

Figura 9.12: Fluxe automatitzat de l'API

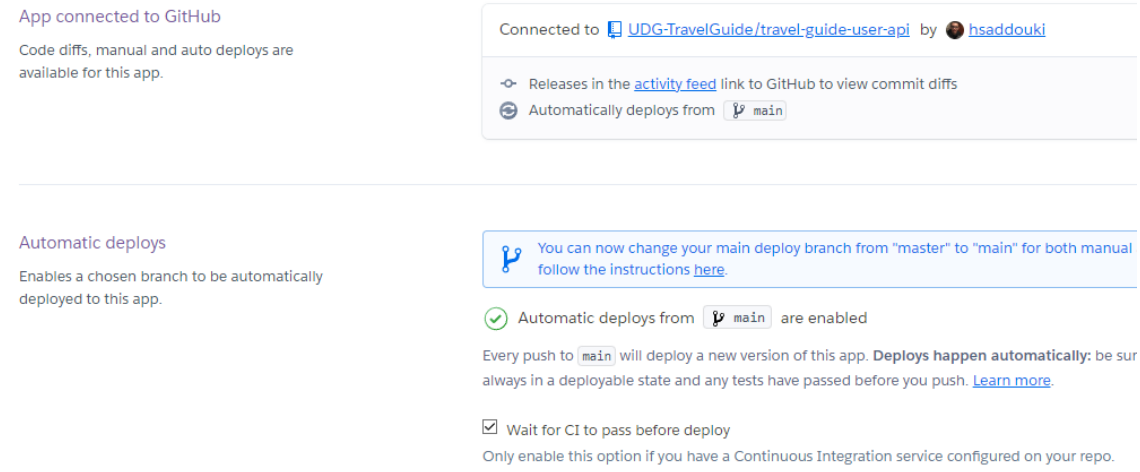

Figura 9.13: Configuració per permetre desplegaments automàtics

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/4StQzSt/Heroku-Deploy.png

CAPÍTOL 10

Implantació i resultats

10.1 Backoffice

A l'hora de programar el backoffice hem hagut de plantejar-nos principalment que volíem poder gestionar com a administradors i quines accions s'havien de poder dur a terme, el procés que s'ha dut a terme és principalment preparar un entorn sobre el qual es visualitzaran els resultats, en el nostre cas hem trobat una manera gratuïta que és utilitzant el servei de Github Page, a continuació principalment hem hagut de consultar molt la documentació d'Angular i moltes publicacions de StackOverflow per saber com es realitzaven moltes accions concretes.

Després de preparar l'entorn i haver decidit utilitzar el framework d'Angular vam preparar-nos i cercar documentació de com limitar l'accés a pàgines tenint en compte que s'ha d'estar loguejat, a part d'això també el com validar que l'usuari que s'està logujeant és un usuari amb un rol en concret i que això no es pugui manipular.

Un cop hem tingut clar com afrontar aquests dos problemes la resta del procés ha sigut més senzill a causa de la facilitat que atorga Angular per crear diferents components i definir la navegació entre les diferents pantalles, a part en utilitzar el mòdul HttpClient que implementa Angular ens ha permès realitzar crides als endpoints d'una manera més accessible.

Per poder accedir al backoffice tenim la pantalla de login.

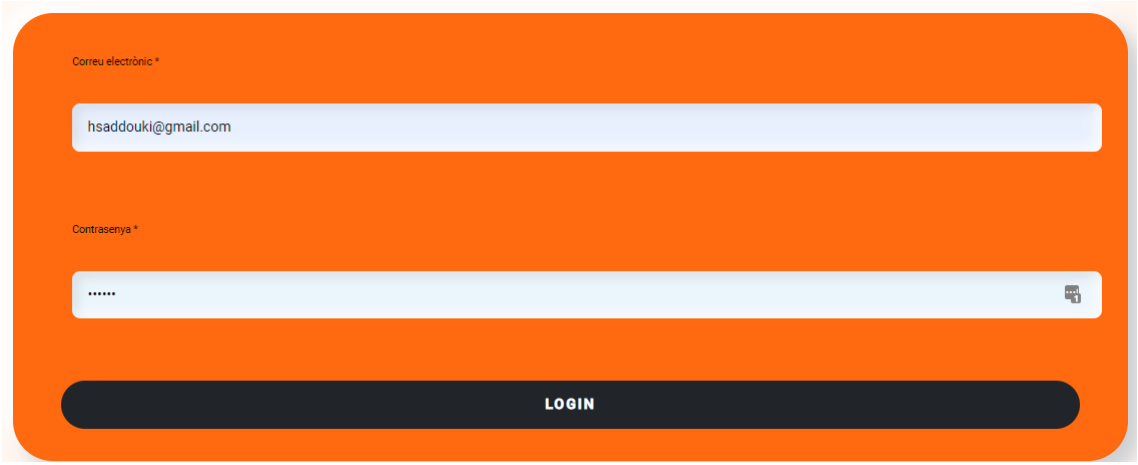

Figura 10.1: Pantalla de login del backoffice

A l'accedir al backoffice el primer que se'ns mostrarà és la pantalla principal amb diferents botons que porten a diferents llocs del backoffice.

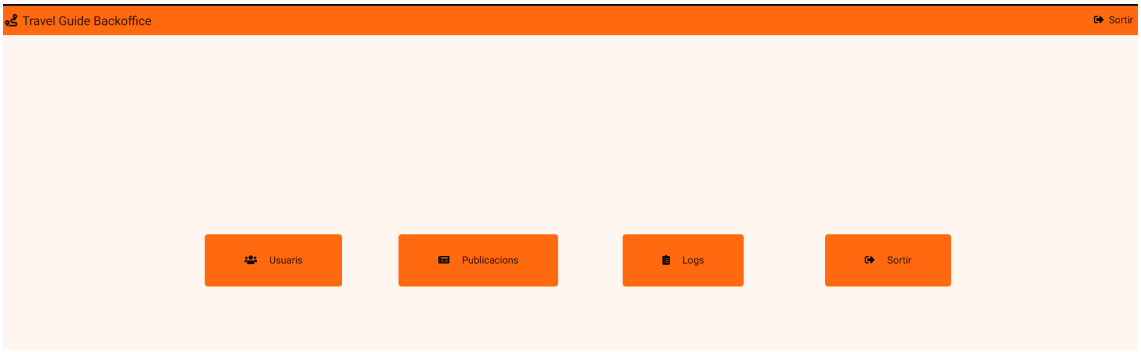

Figura 10.2: Pantalla principal del backoffice

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/GvsDtdW/HOME-BO.png

Des de la pantalla es pot accedir a diferents apartats, un dels apartats és la pàgina per gestionar els usuaris, des d'on podem visualitzar els diferents usuaris i aplicar diferents accions sobre ells.

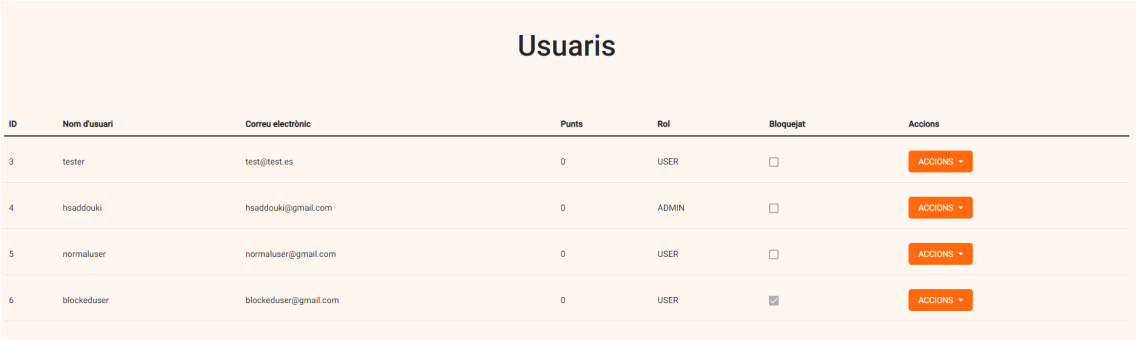

Figura 10.3: Pantalla per gestionar els usuaris des del backoffice

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/gTkNCkg/USERS-BO.png

Un altre apartat seria la pàgina per gestionar les diferents publicacions generades pels usuaris, en aquest cas ocorre el mateix que amb la pantalla d'usuaris, és un llistat de publicacions sobre el qual es pot executar diferents accions.

Figura 10.4: Pantalla per gestionar les publicacions

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/ZNTkDP8/BO-6.png

Un altre apartat és l'apartat de Logs, des d'aquesta pantalla podrem observar diferents comptadors indicant quin tipus de logs s'han anat generant en l'API.

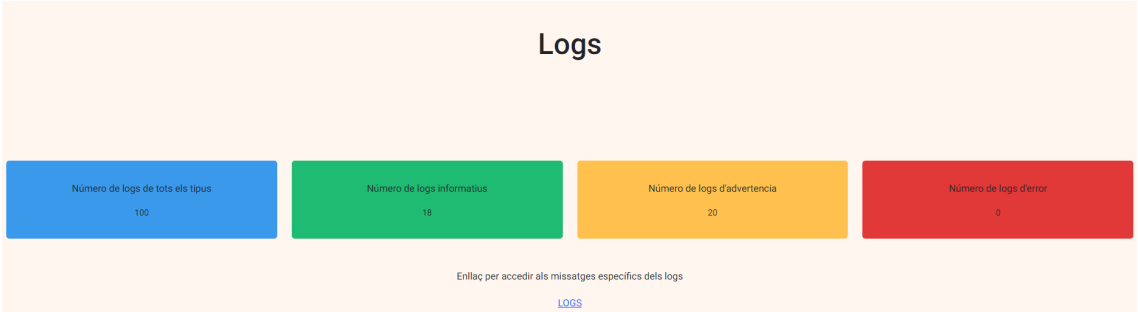

Figura 10.5: Pantalla per visualitzar els diferents LOGS

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/NxLXT0X/LOGS-TEST.png

Per visualitzar específicament els missatges generats pels logs tenim una integració amb els serveis de Papertrail i Slack per visualitzar-los d'una manera més senzilla. Dins de Slack tenim quatre canals, un per cada tipus de logs igual que al backoffice.

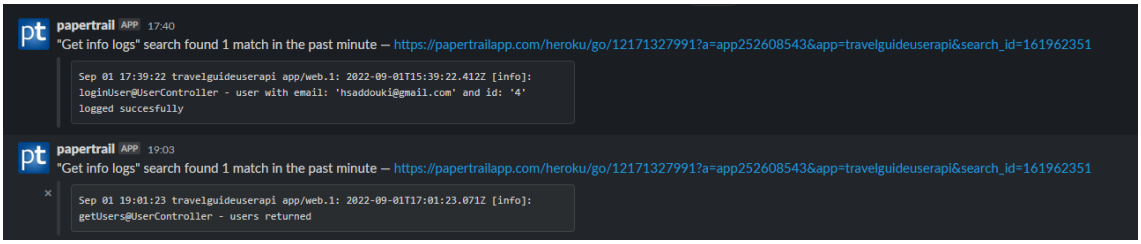

Figura 10.6: Exemple de logs de tipus informatiu

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/qdzwgjn/LOGS-INFO.png

Figura 10.7: Exemple de logs de tipus d'advertencia

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/hVM81sq/LOGS-WARN.png

10.2 APP

Com ja hem introduït en l'estudi de viabilitat del projecte el primer pas per a poder publicar la nostra APP és fer-nos un compte de desenvolupador al Google PlayStore a l'AppStore o en ambdues, tot pagant la taxa que això comporta. Un cop estem donats d'alta haurem de seguir un parell de passos independents de cada plataforma, però que al final acaben coincidint en pujar l'arxiu a la store des d'on tenim un control de versions i podem editar des del nom o descripció de l'APP a tenir tot un conjunt d'analítiques que ens diuen que tan bé funciona l'APP, quina quantitat d'usuaris tenim, quin és el grau d'adquisició d'usuaris, etc.

10.2.1 Publicar Android

Android és un sistema operatiu obert això ens permet compartir la nostra APP fins i tot sense publicar en cap store, per fer-ho simplement necessitem executar la comanda: **npx react-native run-android –variant=release**
això ens generarà una apk (comprimit) amb tot el nostre codi compilat per a Android que podrem enviar a qualsevol persona perquè pugui l'instal·lar al seu smartphone i la poguí provar.

Això, però faria que haguéssim de compartir nosaltres manualment l'arxiu apk o un enllaç amb la gent que volem que tinguin l'APP, per fer-ho realment de forma pública i que tothom hi tingui accés hem de publicar al Google Play Store.

El primer pas és configurar el projecte, per fer-ho necessitem una clau especial que s'encarregarà de confirmar que som nosaltres qui publiquem, per fer-ho hem d'executar la següent comanda:

keytool -genkeypair -v -storetype PKCS12 -keystore [nom clau].keystore -alias [alias nostre] -keyalg RSA -keysize 2048 -validity 10000

Aquesta ens retornarà un fitxer amb una clau associada de 10000 dies de durada, és important no compartir-la amb gent que no volem que publiquin en el nostre nom.

Un cop tenim el fitxer .keystore és important deixar-lo guardat en el directori **android/app**, ja que a continuació configurarem el Gradle, un servei que s'encarregarà d'empaquetar la totalitat de la nostra APP per a poder publicar i aquest l'anirà a buscar en el directori mencionat per poder signar-la.

Al directori **/.gradle/gradle.properties** obrim el fitxer i hem d'introduir les següents línies:
MYAPP_UPLOAD_STORE_FILE=[nom clau].keystore
MYAPP_UPLOAD_KEY_ALIAS=[alias nostre]
MYAPP_UPLOAD_STORE_PASSWORD=*****
MYAPP_UPLOAD_KEY_PASSWORD=*****

Un cop ho tenim hem d'anar al següent directori **android/app/build.gradle** on deixarem el fitxer d'aquesta manera:

```
...
android {
    ...
    defaultConfig { ... }
    signingConfigs {
        release {
            if (project.hasProperty('MYAPP_UPLOAD_STORE_FILE')) {
                storeFile file(MYAPP_UPLOAD_STORE_FILE)
                storePassword MYAPP_UPLOAD_STORE_PASSWORD
                keyAlias MYAPP_UPLOAD_KEY_ALIAS
                keyPassword MYAPP_UPLOAD_KEY_PASSWORD
            }
        }
    }
    buildTypes {
        release {
            ...
            signingConfig signingConfigs.release
        }
    }
}
...
```

Figura 10.8: Fitxer de la ruta android/app/build.gradle

Finalment, dins del directori **/android** haurem d'executar la comanda **./gradlew bundleRelease** i si ho hem fet tot bé obtindrem un arxiu AAB (Android App Bundle) que a continuació podrem pujar a la Google Play Store per publicar.

10.2.2 Publicar iOS

Per publicar en iOS necessitem algun ordinador amb MacOS com a sistema operatiu ja s'ha de fer servir Xcode, una eina d'Apple per a poder programar.

Per fer-ho és tan simple com tenir el certificat que ens dona Apple com a desenvolupadors un cop estem donats d'alta a la AppStore i anar al Xcode amb el projecte de l'APP carregat a l'apartat Producte -> Esquemes -> Editar Esquema i a la pantalla que ens surt seleccionem que volem una release, no hem de modificar massa d'aquí, ja que per defecte ve prou complet. Un cop tenim l'esquema hem de tornar a la barra superior a l'opció Producte -> Arxiu i ell sol ens generarà i agruparà la informació necessària. Un cop acabat ens dona diverses opcions, la que ens interessa és Distribuir App -> App Store Connect -> Pujar -> Signar Automàticament -> Pujar. Amb aquest procés haurem fet que l'APP estigui disponible a l'App Store Connect des d'on com en el cas de la Play Store podem acabar d'editar algunes dades i publicar.

Com en iOS no tenim la mateixa facilitat com en Android per a compartir Apks en el procés de publicar podem triar si la versió la pugem al TestFlight, un servei d'Apple que ens permetrà fer proves abans de sortir a la totalitat del públic de la nostra app, aquest servei ens permet tant posar personal intern com persones seleccionades dintre els usuaris que ja fan servir la nostra aplicació.

10.2.3 Resultats APP

En l'aplicació mòbil hi ha pantalles en les quals tens accés depenent de si has iniciat sessió o no, al mateix temps que algunes pantalles canvien en funció d'això. Algunes pantalles tenen una petita pantalla de càrrega, ja que hem detectat que tardaven a carregar totes les dades i així ens evitem errors per clicks, altrament quan passa algun error l'APP informa l'usuari amb un missatge per pantalla que desapareix al cap d'uns segons.

La pantalla d'inici (figures 10.9 i 10.10) és la primera que veu l'usuari, abans de mostrar-la ja determinem si l'usuari ha iniciat sessió prèviament així que aquí ja arribaria amb la sessió oberta si ja ho ha fet un cop. Al mateix temps que si té seleccionat un altre llenguatge que no sigui anglès l'APP ja estaria en aquell llenguatge. La pantalla ofereix dreceres a les parts més diferencials de l'APP, la pantalla de rutes, el mapa i un últim accés a més serveis una pantalla que manté les configuracions. Si tenim sessió se'ns mostrarà un text, en canvi, si no en tenim se'ns mostrarà dreceres a login i registre. El logotip té una animació per fer que aquesta pantalla sigui més curiosa.

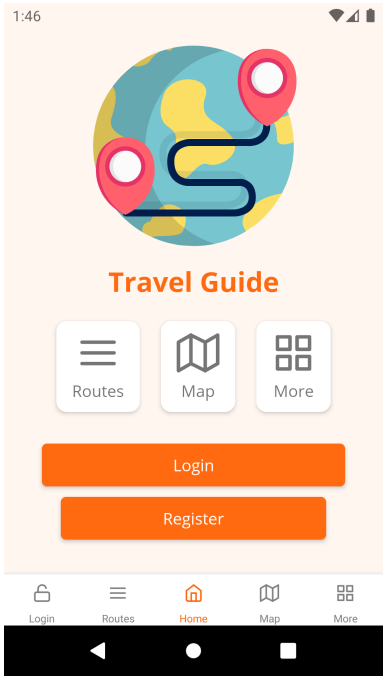

Figura 10.9: Pantalla d'inici sense sessió iniciada.

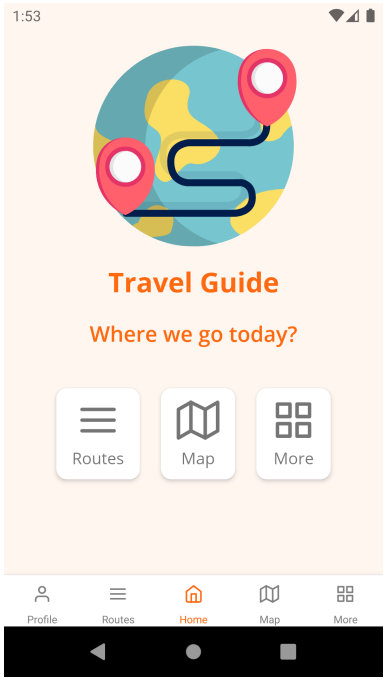

Figura 10.10: Pantalla d'inici sense amb sessió iniciada.

Si volem iniciar sessió podem navegar amb la barra inferior o des de la pantalla, inici. Veiem a la figura 10.11 que aquesta pantalla té un text, els camps per introduir email i contrasenya i seguidament uns textos que són dreceres a la funcionalitat de recuperar contrasenya i al registre. En cas de voler iniciar sessió i de cometre algun error se'ns mostraria un missatge per pantalla com veiem a la figura 10.12.

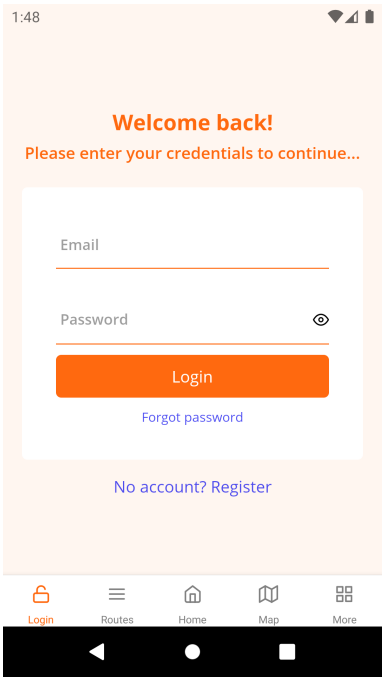

Figura 10.11: Pantalla d'inici sessió.

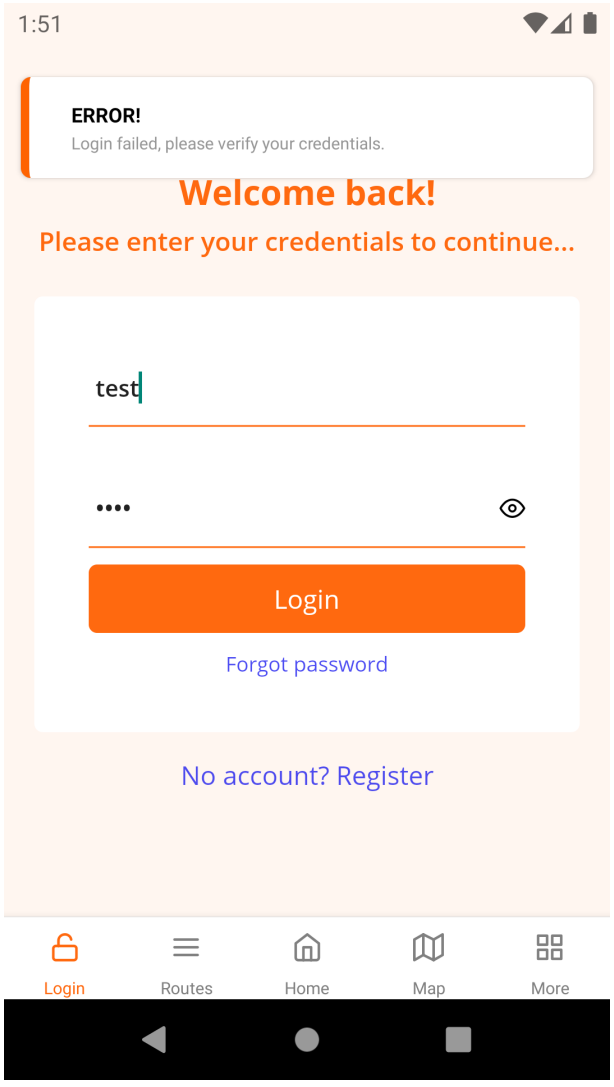

Figura 10.12: Pantalla d'inici sessió amb un error.

Si no tenim compte i ens volem donar d'alta hem de seguir el procés de registre (figures 10.13, 10.14, 10.15) en ell se'ns requereix correu electrònic, nickname, data de naixement i contrasenya. En el cas del correu electrònic internament verifiquem que aquest com a mínim segueixi el format d'un. La contrasenya l'hem deixat a lliure elecció. No acceptem els camps invàlids o buits així que s'han d'omplir tots per continuar.

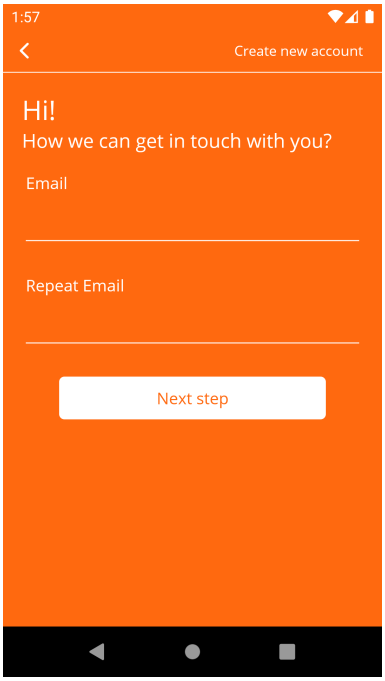

Figura 10.13: Pantalla registre pas 1.

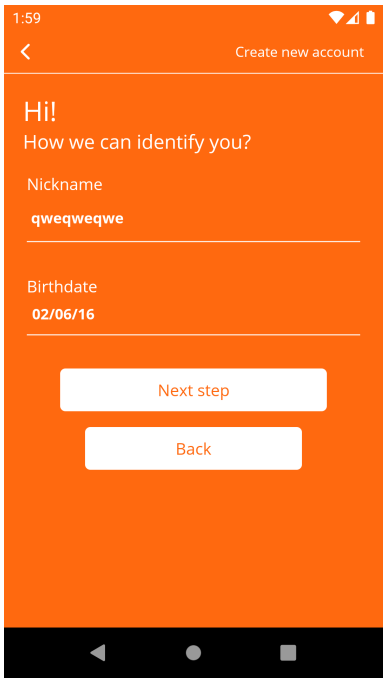

Figura 10.14: Pantalla registre pas 2.

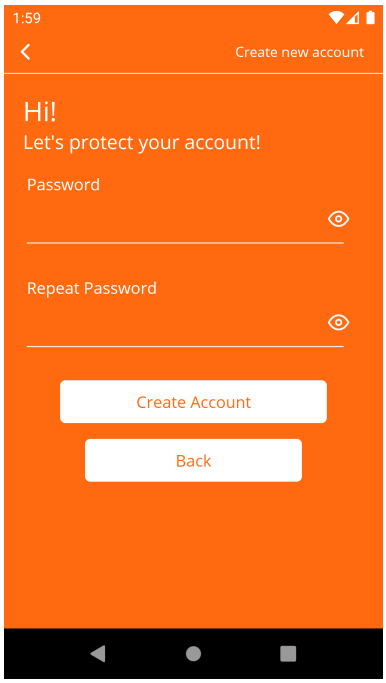

Figura 10.15: Pantalla registre pas 3.

Un cop tenim compte podem accedir a la pantalla del perfil (figura 10.16), aquesta ens ofereix un resum de les rutes i guies que hem publicat al mateix temps que les que tenim agregades a favorits. També ens dona accés a tancar sessió i a editar el perfil (figura 10.17).

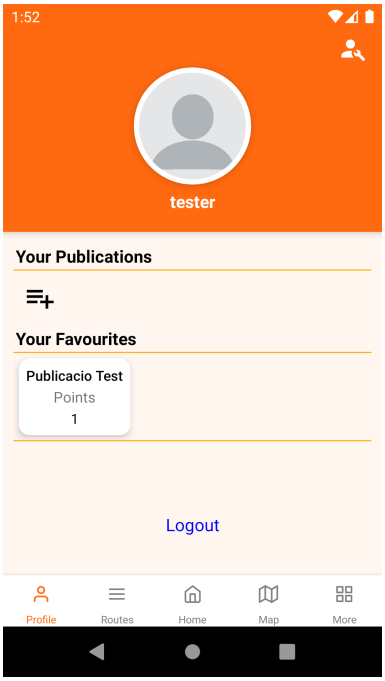

Figura 10.16: Pantalla perfil.

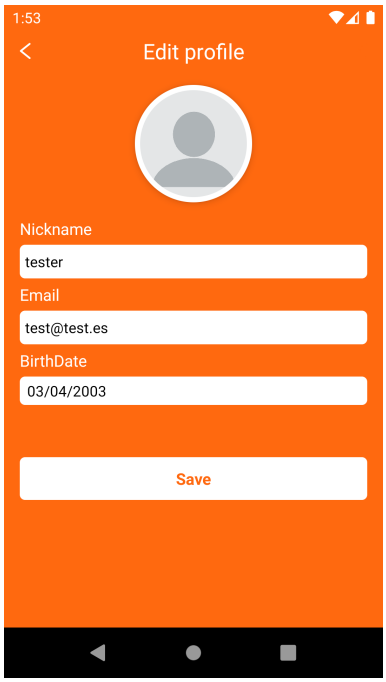

Figura 10.17: Pantalla d'edició perfil.

El llistat de rutes (figures 10.18 i 10.19) ens permet veure totes les rutes de forma paginada, és a dir el servidor no ens torna totes les rutes de cop sinó que l'APP va demanant més rutes a mesura que l'usuari va baixant pel llistat. Aquest llistat també dona l'opció de filtrar per país (figura 10.20) i crear una ruta nova. Anar al detall de cada ruta és tan fàcil com fer click sobre la carta de presentació. Aquesta carta ens diu informació d'interès com el títol, una breu descripció i la puntuació de la ruta.

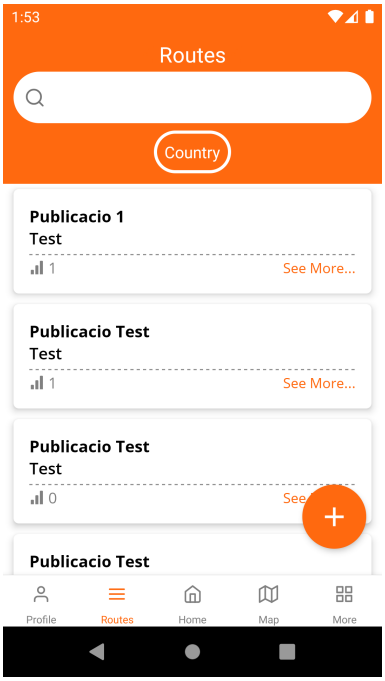

Figura 10.18: Pantalla del llistat de rutes versió amb inici de sessió.

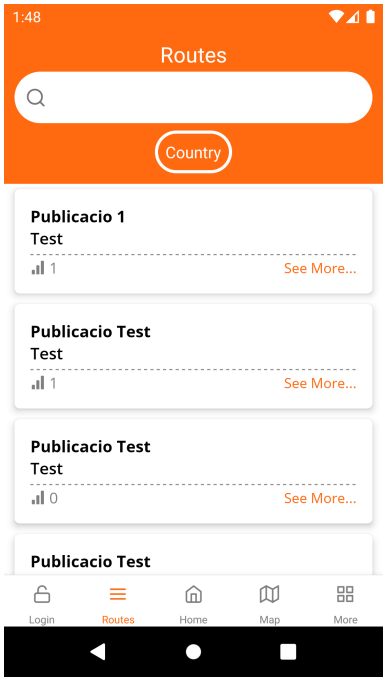

Figura 10.19: Pantalla del llistat de rutes versió sense inici de sessió, perdem el botó d'afegir ruta.

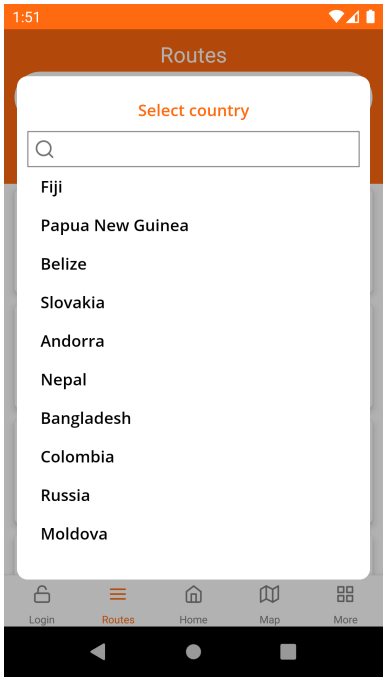

Figura 10.20: Modal del llistat de rutes per seleccionar país.

El detall d'una ruta o guia (figures 10.21 i 10.22) ens repeteix la informació de la carta de presentació i ens mostra el contingut. Des de la icona ruta podem veure un mapa amb la ruta i si anem baixant veiem el contingut de la guia. Si tenim sessió iniciada se'ns mostren unes icones per puntuar i per afegir a favorits.

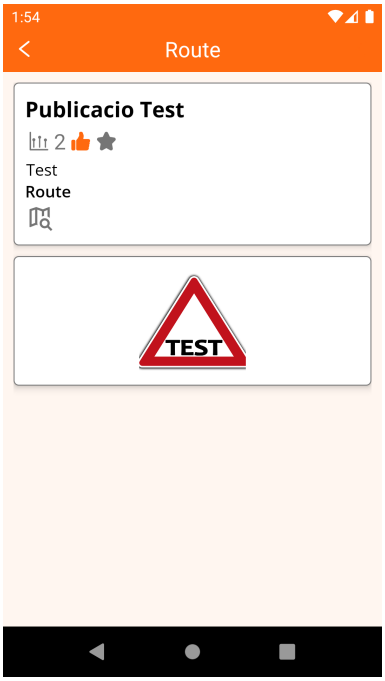

Figura 10.21: Pantalla del detall de ruta o guia versió amb inici de sessió.

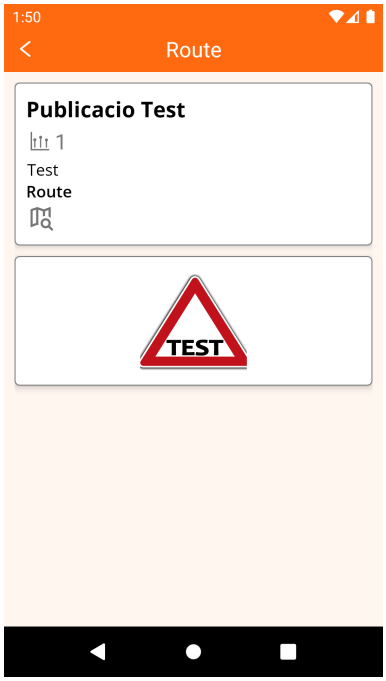

Figura 10.22: Pantalla del detall de ruta o guia versió sense inici de sessió, perdem la posibilitat de puntuar i afegir a favorits.

El mapa (figura 10.23) és un punt de vista diferent de les rutes, aquest en entrar ens demana permisos d'ubicació i si els deneguem ens mostra una pantalla d'error (figura 10.24) des de la qual podem demanar permís un altre cop., altrament se centra en la nostra ubicació en el mapa, però ens deixa navegar lliurement. En fer click sobre la marca d'una ruta ens mostra el títol sobre la marca i la carta de presentació d'aquesta que si fem click ens porta al detall de la ruta. També sobre la carta tenim una creu per tancar-la i que no ens molesti en navegar pel mapa. Addicionalment, des de la icona de la part de dalt ens permet tornar a centrar el mapa a on estem ubicats físicament.

Figura 10.23: Pantalla del mapa.

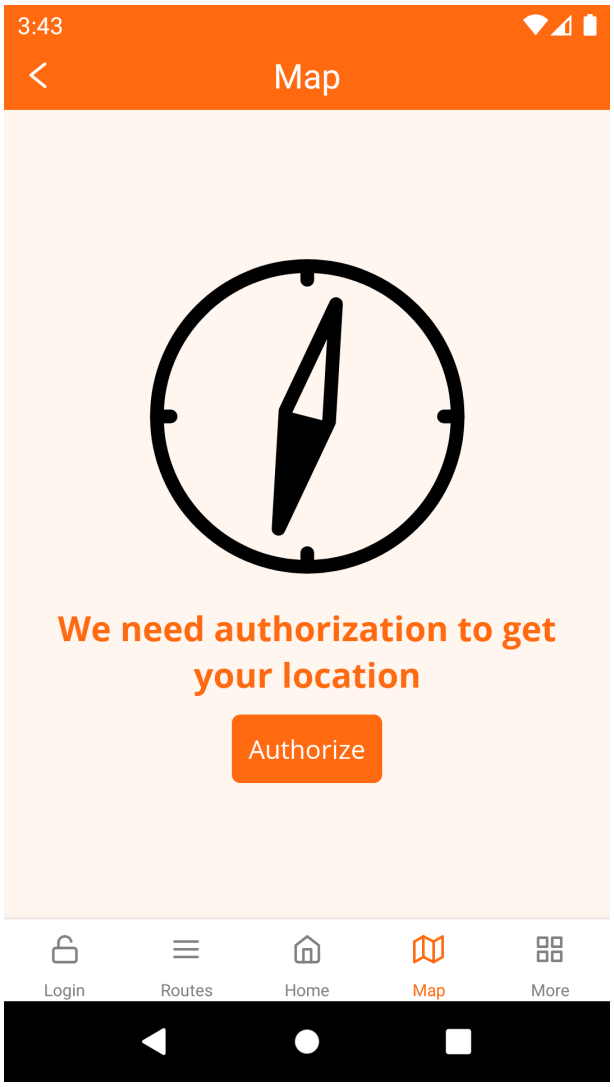

Figura 10.24: Pantalla d'error del mapa.

Si volem publicar hem d'entrar des del llistat de rutes i seguirem un conjunt de passos semblant a la forma d'actuar del registre. El primer (figura 10.25) és introduir el títol, descripció i país de la ruta o guia, un cop tinguem els camps omplerts passem al següent pas. En aquest (figura 10.26 i 10.28) des del botó superior ens mostrarà un modal (figura 10.27) on podem carregar una imatge i/o escriure text. Per carregar una imatge en obra la galeria del mòbil des de la qual seleccionem la que volem carregar. Aquest pas el podem anar iterant tot el que volem per omplir el contingut de la ruta o guia, quan estem premem a següent (figura 10.29) i ens demanarà l'origen de la ruta, el destí i les direccions. Les direccions són punts pels quals volem que passi la ruta, ja que altrament els mapes de Google ens dibuixaran la ruta més directe entre origen i destí. En fer click ens obre un mapa que en pressionar un punt ens mostra un botó per a guardar-lo (figura 10.30). Quan estem de tot premem el botó de crear ruta.

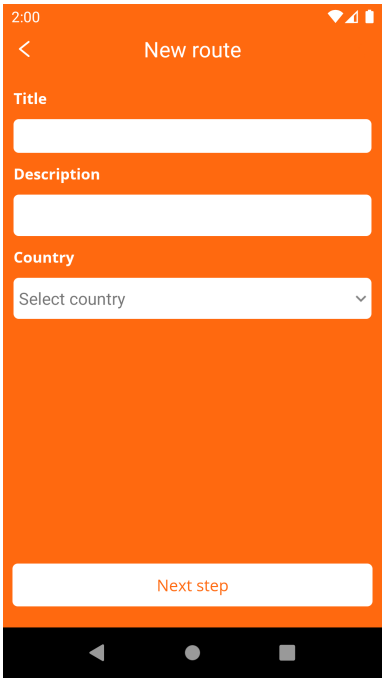

Figura 10.25: Pantalla de creació de ruta pas 1.

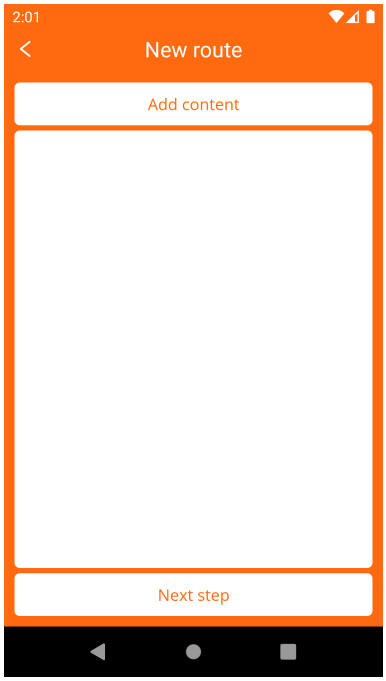

Figura 10.26: Pantalla de creació de ruta pas 2.

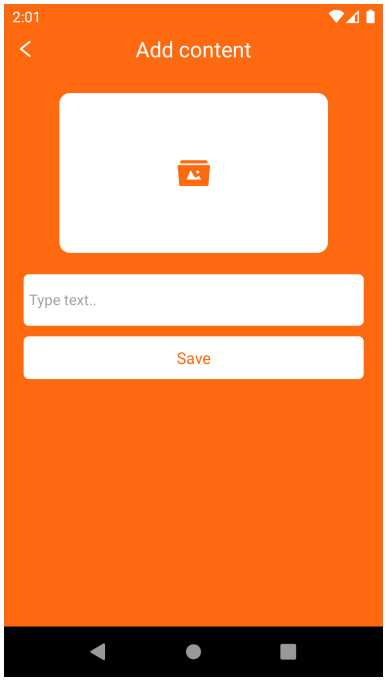

Figura 10.27: Pantalla de creació de ruta pas 2 amb el modal per afegir contingut.

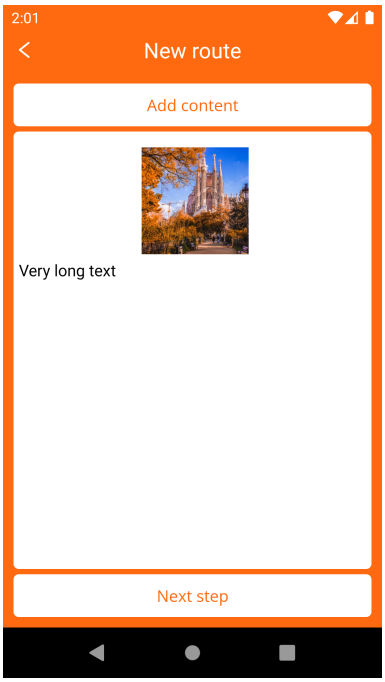

Figura 10.28: Pantalla de creació de ruta pas 2 amb contingut.

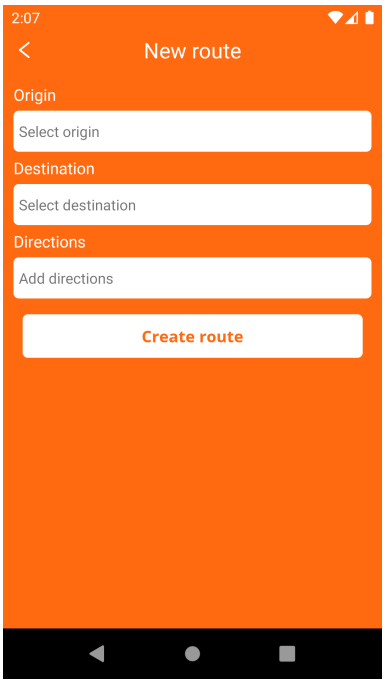

Figura 10.29: Pantalla de creació de ruta pas 3.

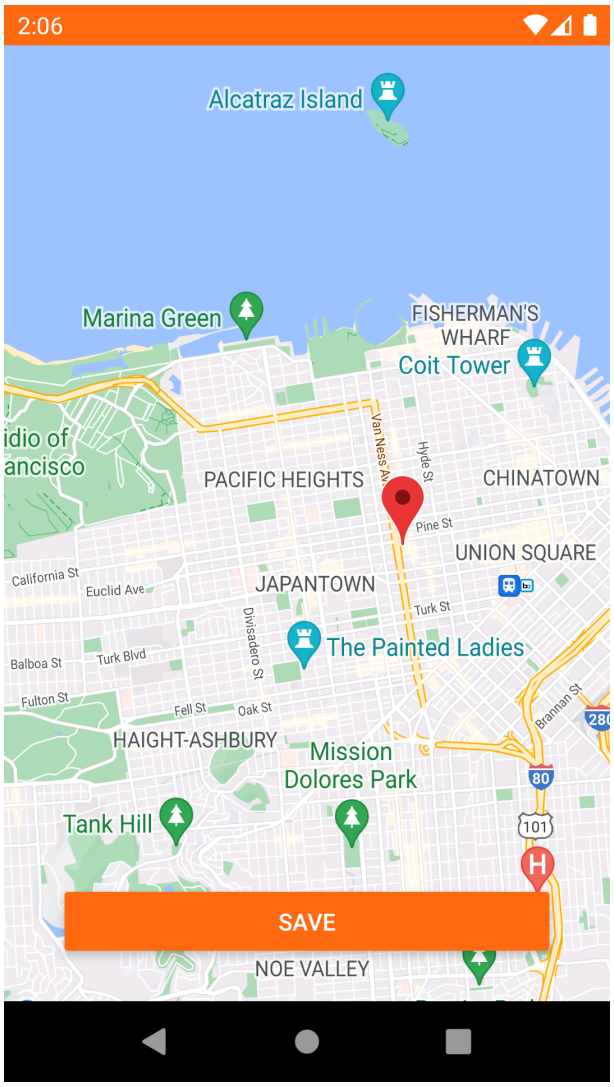

Figura 10.30: Pantalla de creació de ruta pas 3 amb modal per afegir punt.

Finalment, tenim la pantalla de més serveis (figura 10.31), aquí tenim el selector d'idioma de l'APP, però de cara a publicar s'afegiria un punt de menú informatiu sobre la llei RGPD.

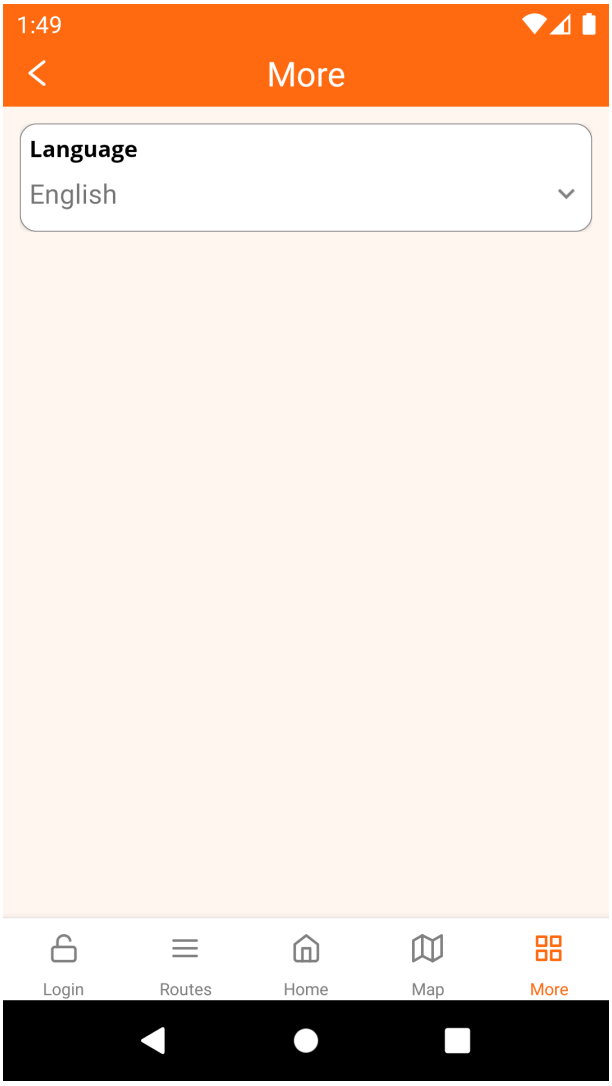

Figura 10.31: Pantalla de més serveis.

Capítol 11

Conclusions

Per assolir els objectius ens hem separat les tasques per cadascú, programar l'API i el BackOffice ha sigut la prioritat d'en Hamza Saddouki i la programació de l'APP ha sigut la prioritat d'en Dan Spoiala, hem anat fent reunions setmanals per conèixer els requisits a prioritzar de part de l'APP i els canvis que serien necessaris a l'API per adaptar els comportaments i resultats per facilitar el desenvolupament, d'aquesta manera hem anat assolint quasi tots els requisits que hem definit, els requisits que no hem assolit han sigut per decidir al final que no eren tan necessaris o que la manera de plantejar-ho no era la correcta, el primer és el de L·L'usuari ha de poder seguir el perfil d'un usuari concret", en aquest cas vàrem decidir no assolir-lo per la baixa prioritat i que al final ens ha semblat que no era necessari pel fet que tenim una altra funcionalitat similar, la qual és seleccionar publicacions favorites i la funcionalitat d'obtenir publicacions per autor.

L'altre requisit no complet ha sigut el de "Els usuaris poden posar opinions sobre una publicació", en aquest cas hem decidit que era suficient amb el requisit de "Els usuaris poden puntuar una publicació", a part que tenim una altra funcionalitat de poder reportar una publicació.

En termes d'APP i Backoffice ens han agradat bastant les tecnologies amb les quals hem treballat i les facilitats que ens aporten, per altra banda, treballar amb NodeJS utilitzant una base de dades relacional no ha sigut tan senzill com amb altres tecnologies, en aquest cas hem trobat diverses mancances i moltes coses que es podien haver fet en menys temps si haguéssim triat una altra tecnologia com Spring o Net Core, en aquest cas hem arribat a la conclusió que NodeJS pot ser una molt bona opció per base de dades no relacionals, però per un projecte així no és la millor opció a escollir.

El funcionament de l'APP és correcte tot-hi que abans de llançar-la a producció caldria acabar d'afinar-la, en alguns aspectes encara pot donar algun error que no ens podem permetre que surti a producció. Per altra banda, no estem del tots contents amb el disseny en comparació a altres aplicacions que veiem avui en dia al mercat, a la nostra li faltaria donar una volta més per a fer que sigui més diferencial a les altres.

Finalment, ha sigut un projecte molt interessant que ens ha fet aprendre bastant sobre el desenvolupament d'aplicacions de l'actualitat, hem vist punts en comú amb el que ja sabíem de classe, però també hem vist maneres de treballar noves. Hem treballat des del disseny i anàlisis fins al desenvolupament i implementació, hem fet testos per veure que tot funciones i hem après a escollir les solucions més eficients als nostres problemes durant el desenvolupament dins de la multitud que trobem per internet.

Capítol 12

Treball futur

Com a treball futur ens interessaria sobretot implementar unes quantes funcionalitats que aportarien molt als usuaris i ajudaria a popularitzar més l'aplicació, un exemple és una funcionalitat de poder fer un seguiment d'un altre usuari i poder veure les publicacions que va creant, d'aquesta manera tenir una pantalla específicament per fer un seguiment de les publicacions de tots els usuaris als quals ens interessa tenir en compte en l'aplicació.

Una altra cosa a tenir present seria millorar molt el servei posant l'api a un servidor de pagament i no un gratuït com està actualment, Heroku és un bon servei que aporta molt als programadors i permet començar un projecte sense tenir en compte gaire l'economia, però a la llarga es pot convertir més en un problema si arribem a tenir un volum considerable d'usuaris, en aquest cas és millor tenir l'api i la base de dades en un servidor ja preparat per això.

Parlant sobre l'equip seria interessant afegir un dissenyador per revisar les interfícies i ajustar el disseny perquè sigui uniforme, modern i funcional. I com l'equip creix també elegir un responsable que ens ajudi a ajustar els sprints, afegir importància a les tasques i organitzar-les de manera que el desenvolupament pels programadors ens sigui més fàcil.

Com a punt final ens interessaria molt també ampliar el nostre mercat i apuntar també als usuaris que utilitzen el sistema operatiu iOS, i en aquest cas si l'ampliació va bé també seria interessant afegir una web com a zona d'usuaris i que també es pugui utilitzar el servei des d'un navegador.

CAPÍTOL 13

Manual d'usuari i/o instal·lació

13.1 API

Per instal·lar l'api és un requisit tenir instal·lat la versió 16 de NodeJs, es pot obtenir des de https://nodejs.org/dist/v16.17.0/

El primer pas és descarregar el repositori que es troba a https://github.com/UDG-TravelGuide/travel-guide-user-api o clonar-lo amb la comanda **git clone https://github.com/UDG-TravelGuide/travel-guide-user-api.git**.

A continuació s'ha d'executar la comanda **npm install** dins el directori del repositori descarregat.

És important executar les comandes necessàries a la base de dades per crear les taules i camps que es faran servir, per tal de facilitar la feina hi ha un arxiu anomenat scritps.sql dins el repositori el qual si s'executa crearà totes les taules i camps necessaris.

Finalment, hem de definir les variables d'entorn necessàries pel correcte funcionament de l'api, per fer-ho hem de crear un arxiu anomenat .env dins el directori **/dist**, dins aquest arxiu definirem les següents variables.

- `DATABASE_URI` És la URI de connexió per la base de dades, aquí tenim un exemple dels diferents formats d'una URI https://stackoverflow.com/questions/3582552/what-is-the-format-for-the-postgresql-connection-string-url
- `OAUTH2_CLIENT_ID` És la id de client que permet a l'api crear una autentificació per google, es pot obtenir la id des de aquest apartat https://console.cloud.google.com/apis/credentials
- `PSWD_RCVR_URL` És l'enllaç que indica la pàgina per introduir la nova contrasenya d'un usuari al demanar una recuperació de contrasenya, es pot substituir la part **host-backoffice**"del següent enllaç pel domini on es troba el backoffice **https://host-backoffice/recover.html**

- `TOKEN_SECRET` És el codi per generar token d'autentificació, pot ser qualsevol codi, serveix basicament com una l·llavor"per la generació de tokens.
- `ADMIN_EMAIL_API_KEY` L'api per defecte envia mails amb el servei de **Twilio SendGrid** i aqui definim l'API KEY per utilitzar aquest servei, en cas de que no s'utilitzi el servei s'ha de canviar el codi ubicat al arxiu **AuthController.ts** en el mètode **recoverPassword**.
- `MAIL_USER` És l'usuari de correu electrónic des del qual volem enviar els correus als usuaris.

Aquest seria un exemple de com definir les variables d'entorn i els seus valors al arxiu .env:

```
DATABASE_URI=postgress://user:password@host:port/database
OAUTH2_CLIENT_ID=loremipsum.google.com
PSWD_RCVR_URL=https://backoffice.com/recover.html
TOKEN_SECRET=tokencodesecret
ADMIN_EMAIL_API_KEY=SG.code.example
MAIL_USER=user@host.com
```

Finalment per executar l'api podem fer servir la comanda **npm start**

13.2 APP

13.2.1 Instal·lació

Abans de carregar les dades del projecte primer hem de tenir l'entorn de desenvolupament preparat per això hem de seguir la guia que ens dona React Native a https://reactnative.dev/docs/environment-setup

El primer pas és instal·lar chocolatey, un gestor de paquets que ens facilitarà el procés, un cop el tenim instal·lat des de la seva web https://chocolatey.org/ podem executar la comanda **choco install -y nodejs-lts openjdk11** aquesta s'encarregarà d'instal·lar al nostre ordinador el NodeJS juntament amb l'OpenJDK de Java.

A continuació hem de descarregar l'Android Studio, ja que necessitem les dependències de SDK i SDK Platform que conté. Podem descarregar-lo des de https://developer.android.com/studio. Les SDK que hem de seleccionar són totes aquelles per les quals volem testejar l'APP, pel fet que són eines que ens permeten desenvolupar en la versió relacionada, per exemple els SDK 31 ens permeten desenvolupar en la versió d'Android 12.

Un cop tenim les SDK el que hem de fer és configurar les variables d'entorn de ANDROID_HOME, SDKs i Platform, per fer això hem d'anar a la cerca del sistema operatiu i buscar on es troben, en el cas de Windows en el panell de control.

Finalment, ja podem copiar el projecte de l'APP des del GitHub amb:
git clone https://github.com/UDG-TravelGuide/travel-guide-app.git
utilitzant l'eina Git que ja hauríem de tenir instal·lada a la nostra màquina. L'últim pas és fer un **npm install** per instal·lar les dependències i ja podríem treballar amb el nostre projecte. Per iniciar-lo en mode de treball és tan fàcil com escriure en la consola **npx react-native run-android** en el cas de treballar amb android o **npx react-native run-ios** en el cas d'iOS.

13.2.2 Manual d'usuari

Un cop tinguem l'APP instal·lada segons alguna de les formes que hem explicat anteriorment ja podrem utilitzar-la com un usuari més. Primer de tot cal familiaritzar-se amb el sistema de navegació, una barra amb icones a la part inferior de la pantalla, com veiem a la figura 13.1 aquesta eina ens ofereix accessos directes a diverses pantalles des del login i el llistat de rutes al mapa de rutes i més serveis.

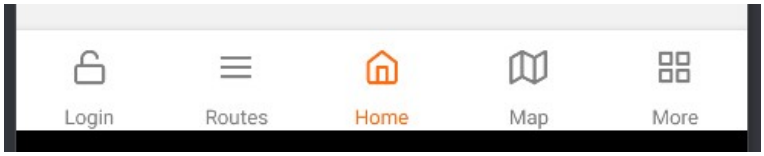

Figura 13.1: Navegació de l'APP

Podem navegar per l'APP sense necessitat de fer login ja que la majoria de funcionalitats estan habilitades, però si ens interessa publicar rutes, puntuar rutes, guardar rutes a favorites i tenir un perfil personal haurem de donar nos d'alta o iniciar sessió.
Com podem veure a la figura 13.2 la pantalla Login ens permet iniciar sessió si ja estem donats d'alta, recuperar la contrasenya en cas que l'hem oblidat o registrar-nos si no tenim compte.

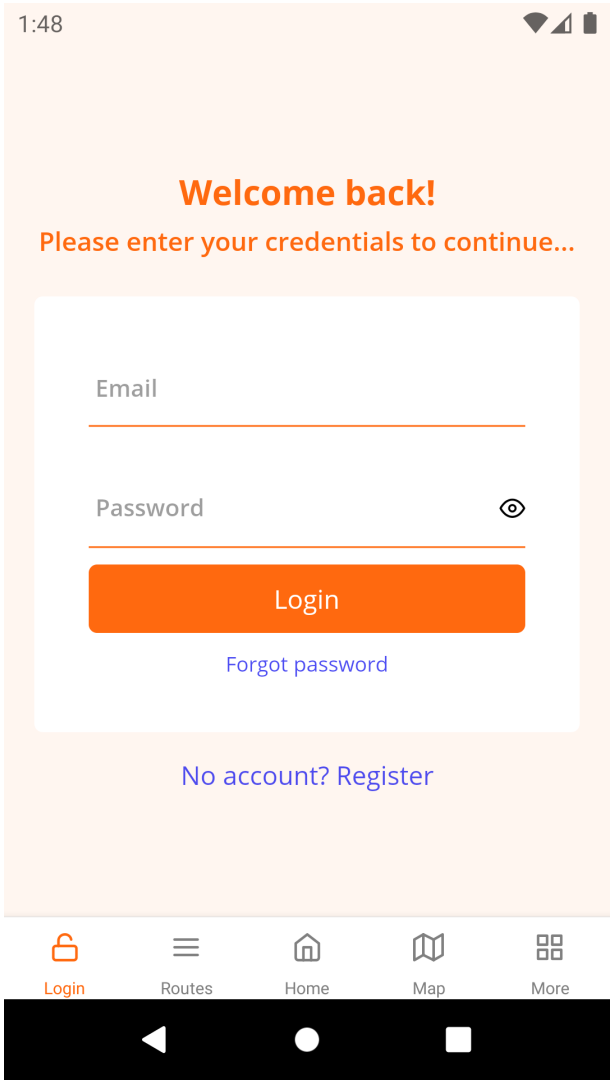

Figura 13.2: Pantalla login APP

Per facilitar l'ús de l'aplicació aquesta disposa de pantalles de càrrega que mostrem de mentre carreguem la informació i també missatges que es mostren a l'usuari perquè sàpiga si alguna cosa no ha funcionat del tot bé, per exemple si fem un login malament l'APP ens enviarà un missatge com aquest:

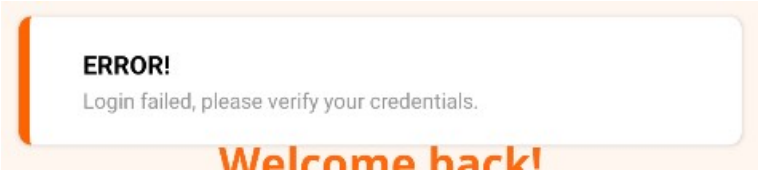

Figura 13.3: Pantalla login error APP

Si fem login a l'APP tindrem accés al perfil de l'usuari, des del perfil l'usuari pot veure les seves rutes, les rutes que té a favorits, pot tancar sessió i pot editar el perfil. També la barra de navegació canvia el Login pel Profile.

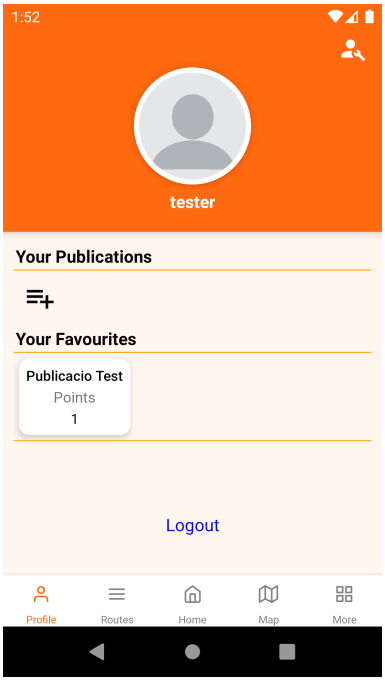

Figura 13.4: Pantalla perfil APP

Si fem click a la icona d'editar el perfil l'aplicació ens porta a la següent finestra:

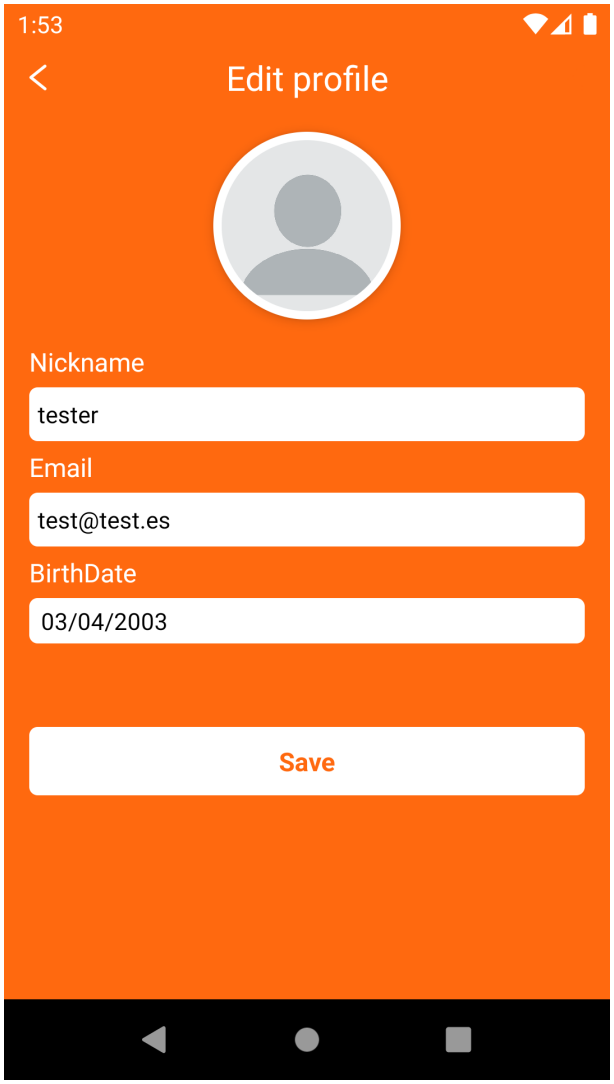

Figura 13.5: Pantalla perfil APP

En aquest espai deixem editar a l'usuari l'imatge, el nom públic, el correu i la data de naixement.

Si visitem el llistat de rutes, la segona icona de la navegació ens trobarem el següent:

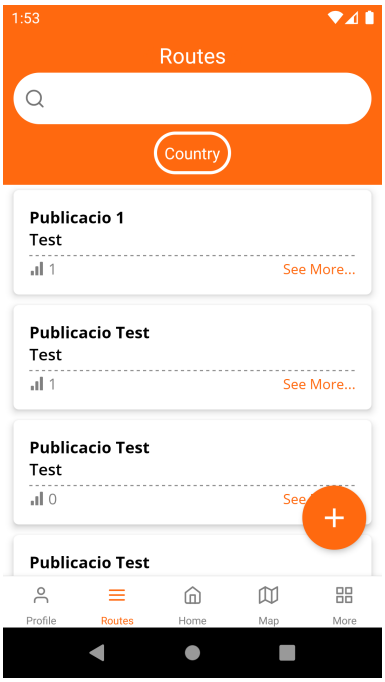

Figura 13.6: Pantalla llistat rutes APP

Podem observar que aquesta pantalla ens deixa cercar rutes per nom, filtrar per país i afegir rutes des del botó "+".

Si el que volem es crear un usuari nou, accedim al registre i seguim els passos que ens marca:

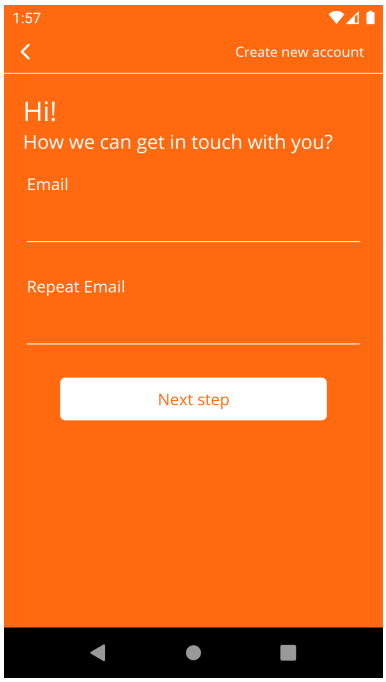

Figura 13.7: Primera pantalla registre APP

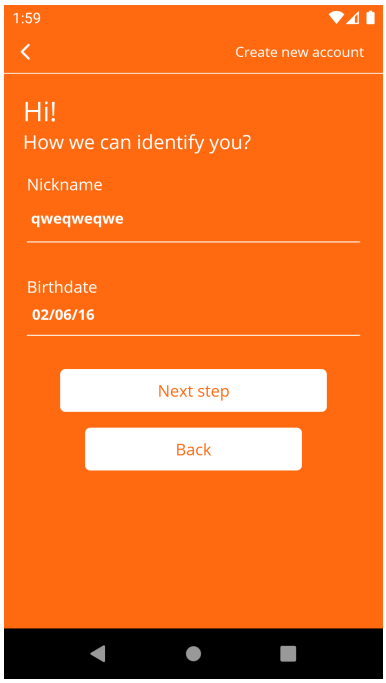

Figura 13.8: Segona pantalla registre APP

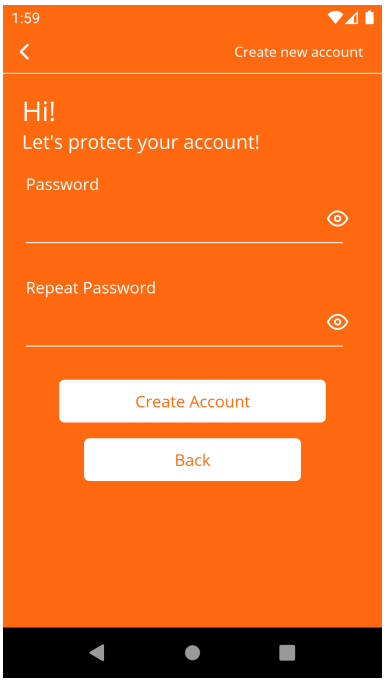

Figura 13.9: Tercera pantalla registre APP

Per altra banda, si volem crear una ruta hem de seguir els passos indicats: Primer introduïm un títol, una breu descripció i el país a on estarà ubicada la ruta.

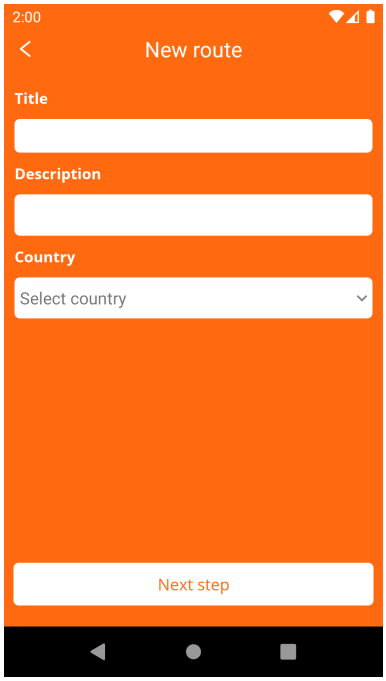

Figura 13.10: Primera pantalla rutes

Després passem a una pantalla on anirem iterant entre la figura 13.11 i la 13.12 cada cop que fem click a "add content"a la figura 13.11 ens mostrarà la figura 13.12 des de la qual podem carregar una imatge o escriure text.

Finalment, a la tercera pantalla hem de seleccionar un origen, destí i direccions, per fer-ho anirem iterant sobre la figura 13.13 i 13.14, ja que un cop cliquem en la 13.13 se'ns obrirà la 13.14 per marcar el punt.

Quan tinguem tot fet premem crear ruta i es demana a l'API la creació.

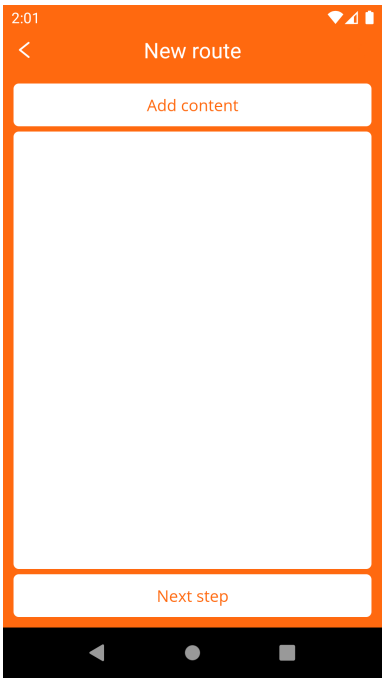

Figura 13.11: Segona pantalla rutes

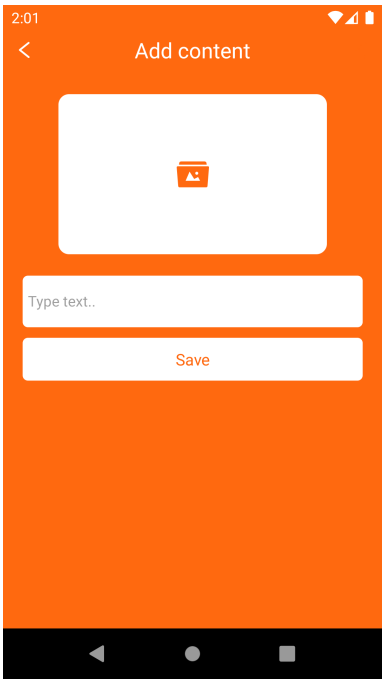

Figura 13.12: Modal segona pantalla rutes

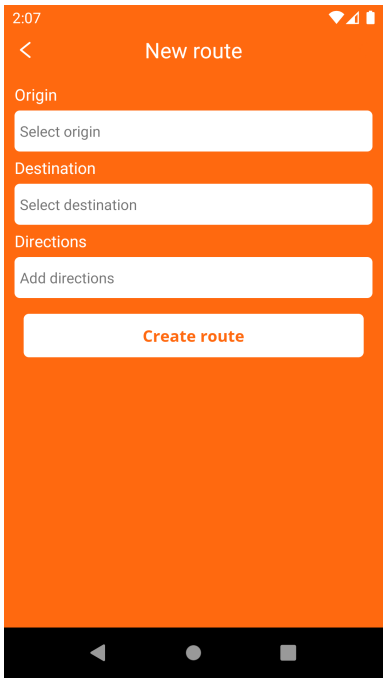

Figura 13.13: Tercera pantalla rutes

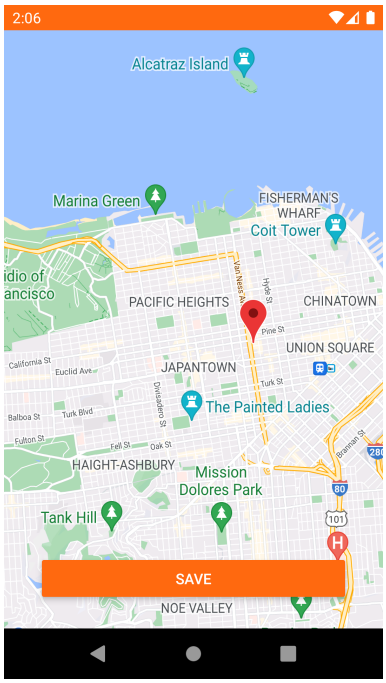

Figura 13.14: Modal tercera pantalla rutes

Finalment, la pantalla "More"que ens serveix per agrupar opcions de configuració de l'APP ens permet canviar l'idioma.

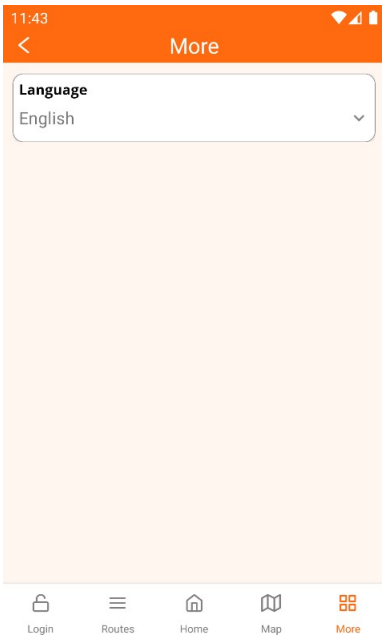

Figura 13.15: Pantalla More de l'APP

13.3 Backoffice

13.3.1 Instal·lació

Per instal·lar el backoffice és un requisit tenir instal·lat NodeJs (recomanat tenir l'última versió), es pot obtenir des de https://nodejs.org

És necessari també instal·lar la línia de comandes d'Angular amb la comanda **npm install -g @angular/cli**

A continuació s'ha de descarregar el repositori des de https://github.com/UDG-TravelGuide/UDG-TravelGuide.github.io o amb la comanda git clone **https://github.com/UDG-TravelGuide/UDG-TravelGuide.github.io.git**

Després podem executar en localhost el backoffice amb la comanda ng serve o generar la versió desplegable amb la comanda **ng build –output-path=directori**

Si hem generat la versió desplegable l'últim pas seria copiar els arxius del directori de sortida i posar-los al servidor per via ftp.

13.3.2 Manual d'usuari

El primer pas és accedir al backoffice a partir del seguent enllaç: https://udg-travelguide.github.io

Hem d'introduir un usuari amb permisos de ADMIN per poder accedir al backoffice, un exemple d'usuari seria **tfg@backoffice.com** amb contrasenya **tfgbackoffice**.

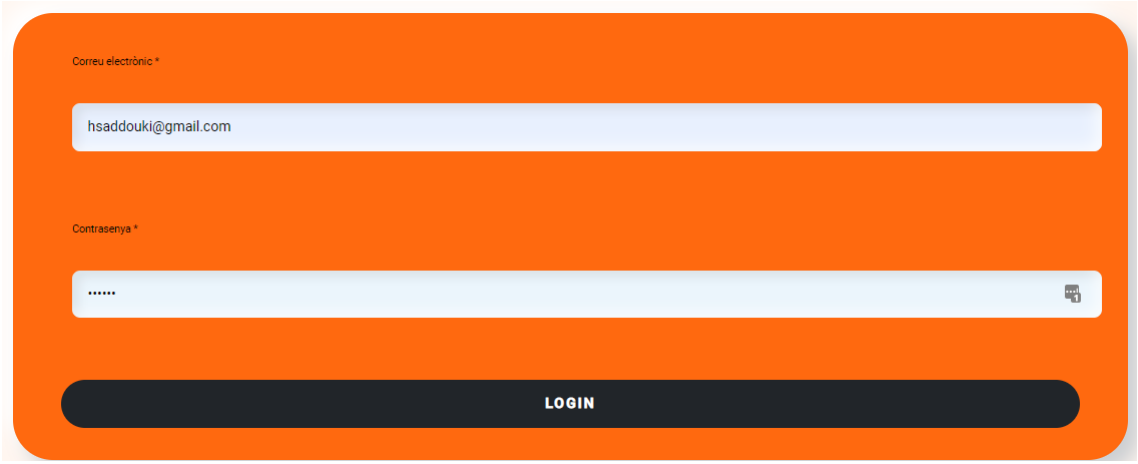

Figura 13.16: Formulari per autentificar-se al backoffice

Un cop ens hem autentificat visualitzarem la pantalla principal del backoffice on ens mostraran les diferents pantalles a les quals podem accedir.

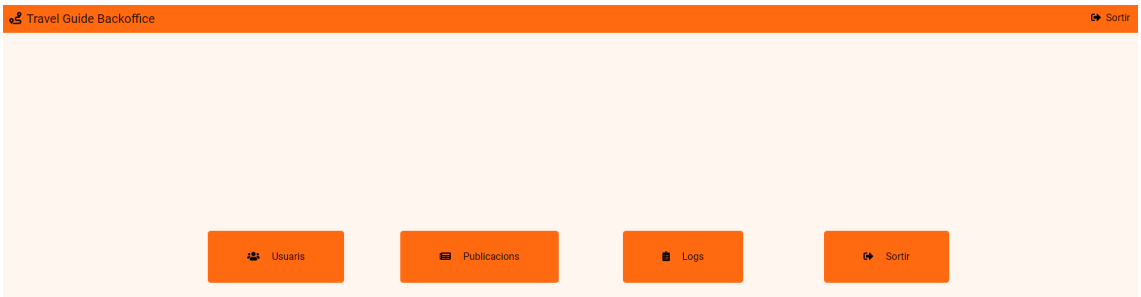

Figura 13.17: Pantalla principal del backoffice

Enllaç de l'imatge en alta resolució:
https://i.ibb.co/GvsDtdW/HOME-BO.png

Qualsevol de les opcions que es mostra en la pantalla principal és un botó que ens porta a un apartat del backoffice.

Per accedir al panell per gestionar els usuaris hem de prèmer el botó d'Usuaris.

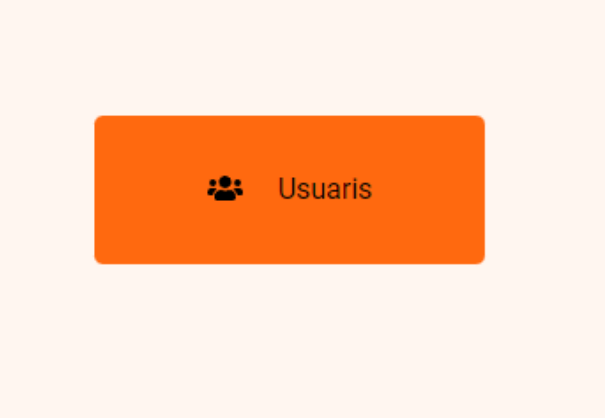

Figura 13.18: Botó per accedir a la pantalla de gestió d'usuaris

Dins la pantalla de gestió d'usuaris podrem veure una taula amb tots els usuaris amb la seva informació corresponent com el nom d'usuari, correu electrònic, nombre de punts atorgats, rol (pot ser administrador o un usuari client), i finalment si és un usuari que està bloquejat.

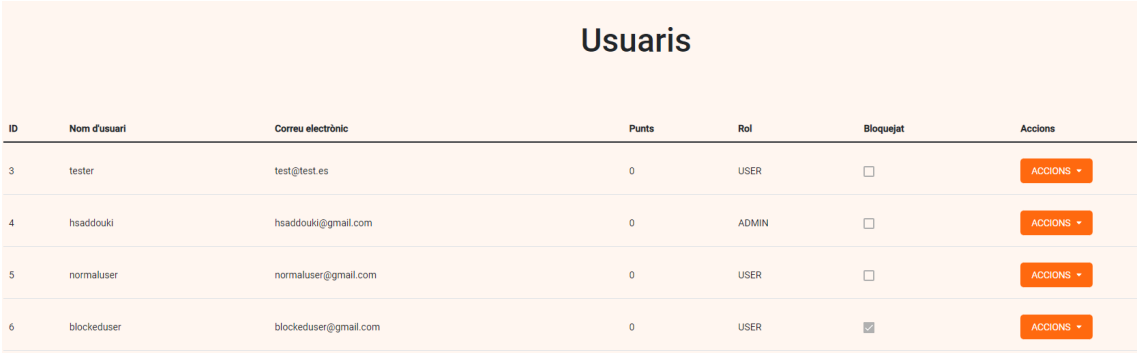

Figura 13.19: Taula amb el llistat d'usuaris

Enllaç de l'imatge en alta resolució:
https://i.ibb.co/gTkNCkg/USERS-BO.png

Com usuaris administradors del backoffice podem realitzar diferents accions sobre els usuaris prement el botó d'Accions que es troba al costat dret de cada usuari.

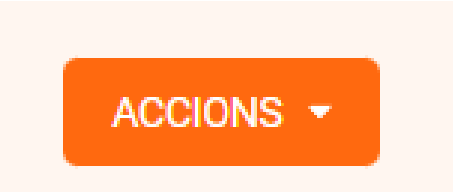

Figura 13.20: Botó per desplegar les diferents accions disponbiles a executar

Com a administradors podem executar tres accions diferents sobre un usuari

- **Bloquejar** Ens permet bloquejar un usuari en concret, aquesta acció fa que l'usuari no tingui permisos per accedir ni a l'app ni al backoffice
- **Desbloquejar** Ens permet desbloquejar un usuari en concret, aquesta acció fa que l'usuari torni a tenir accés ja sigui al backoffice i/o a l'app.
- **Canviar rol** Ens permet alternar el rol d'un usuari, per defecte tenim un sol usuari administrador a la base de dades al realitzar la instal·lació de l'API, però podem fer que un usuari nou tingui aquest permís a través d'aquesta acció, també ens permet canviar un usuari administrador per un rol d'usuari client.

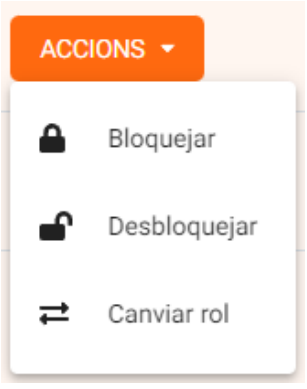

Figura 13.21: Llistat d'accions que es poden executar sobre un usuari

En cas d'executar l'acció de Bloquejar o Desbloquejar un usuari ens sortirà un popup per confirmar l'acció en cas que l'usuari s'hagi equivocat al prémer el botó.

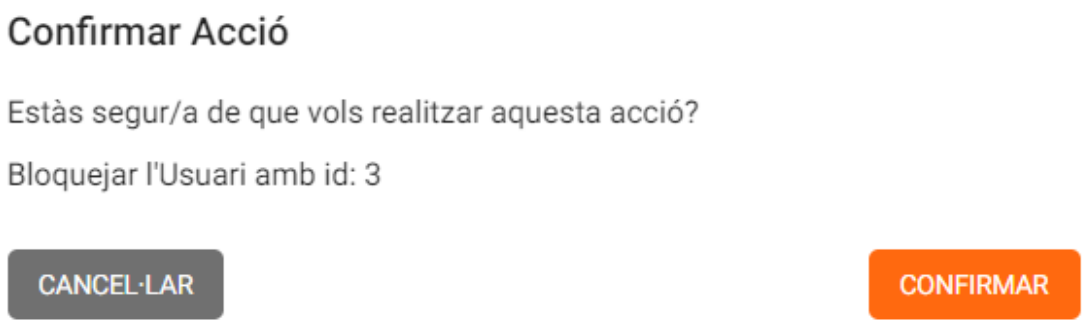

Figura 13.22: Popup per confirmar l'acció de bloquejar un usuari

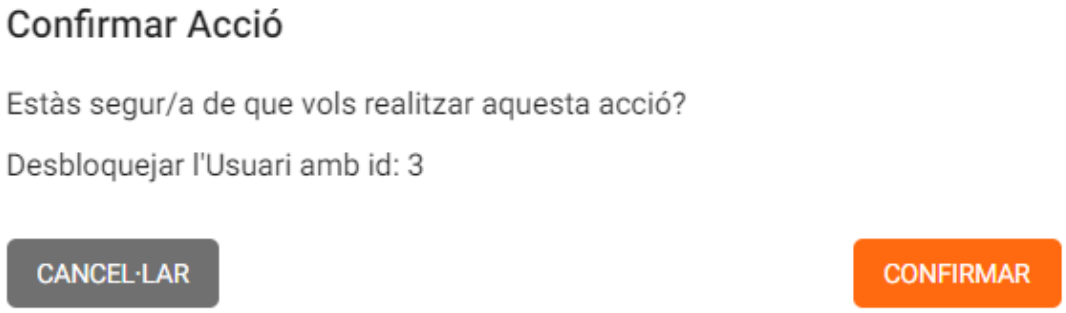

Figura 13.23: Popup per confirmar l'acció de desbloquejar un usuari

En cas d'executar l'acció de Canviar rol ens sortirà un popup amb un desplegable amb les diferents opcions de rols a escollir.

Figura 13.24: Popup per canviar de rol a un usuari

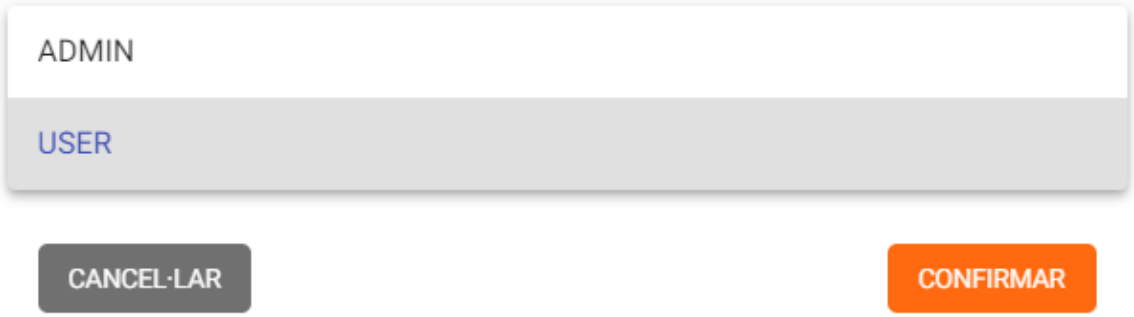

Figura 13.25: Popup amb el desplegable mostrant els diferents rols a escollir

Per accedir a la pantalla principal per visualitzar tots els apartats del backoffice podem fer servir la barra de navegació superior prement sobre el Logo del backoffice.

Figura 13.26: Logo del backoffice ubicat a la part superior esquerra de la pàgina web

Per accedir al panell per gestionar les publicacions hem de prémer el botó de Publicacions des de la pantalla principal.

Figura 13.27: Botó per accedir a la pantalla de gestió de publicacions

Dins la pantalla de gestió de publicacions podrem veure una taula amb totes les publicacions amb la seva informació corresponent com el títol, l'id de l'autor, a quin país està dirigida la publicació, el nombre de reports que ha obtingut i el nombre de punts atorgats.

Figura 13.28: Taula amb el llistat de publicacions

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/ZNTkDP8/BO-6.png

Com usuaris administradors del backoffice podem realitzar diferents accions sobre les publicacions prement el botó d'Accions que es troba al costat dret de cada publicació.

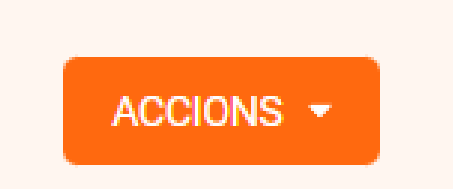

Figura 13.29: Botó per desplegar les diferents accions a executar sobre una publicació

Com a administradors podem executar tres accions diferents sobre una publicació

- **Veure publicació** Obre un popup mostrant-nos tot el contingut que pot ser il·lícit o no permés, com per exemple el text o imatge introduïts a la publicació.
- **Esborrar** Ens permet esborrar per complet una publicació de la base de dades.
- **Bloquejar autor** Ens permet bloquejar l'autor de la publicació sense tenir la necessitat d'anar a la pantalla de gestió d'usuaris a cercar l'usuari en concret.

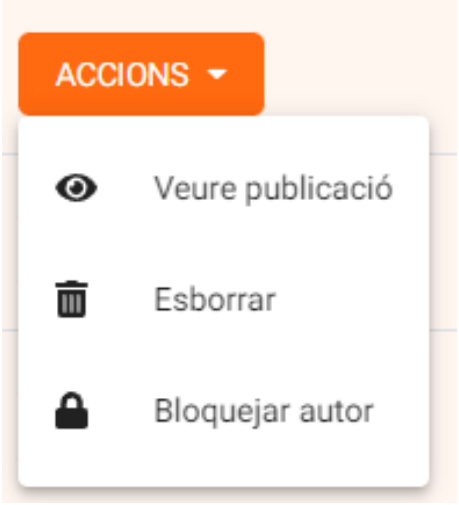

Figura 13.30: Llistat d'accions que es poden exaecutar sobre una publicació

En cas d'executar l'acció de Veure publicació es mostra un popup amb tota la informació rellevant de la publicació i en quin ordre estan posats els continguts i de quin tipus són.

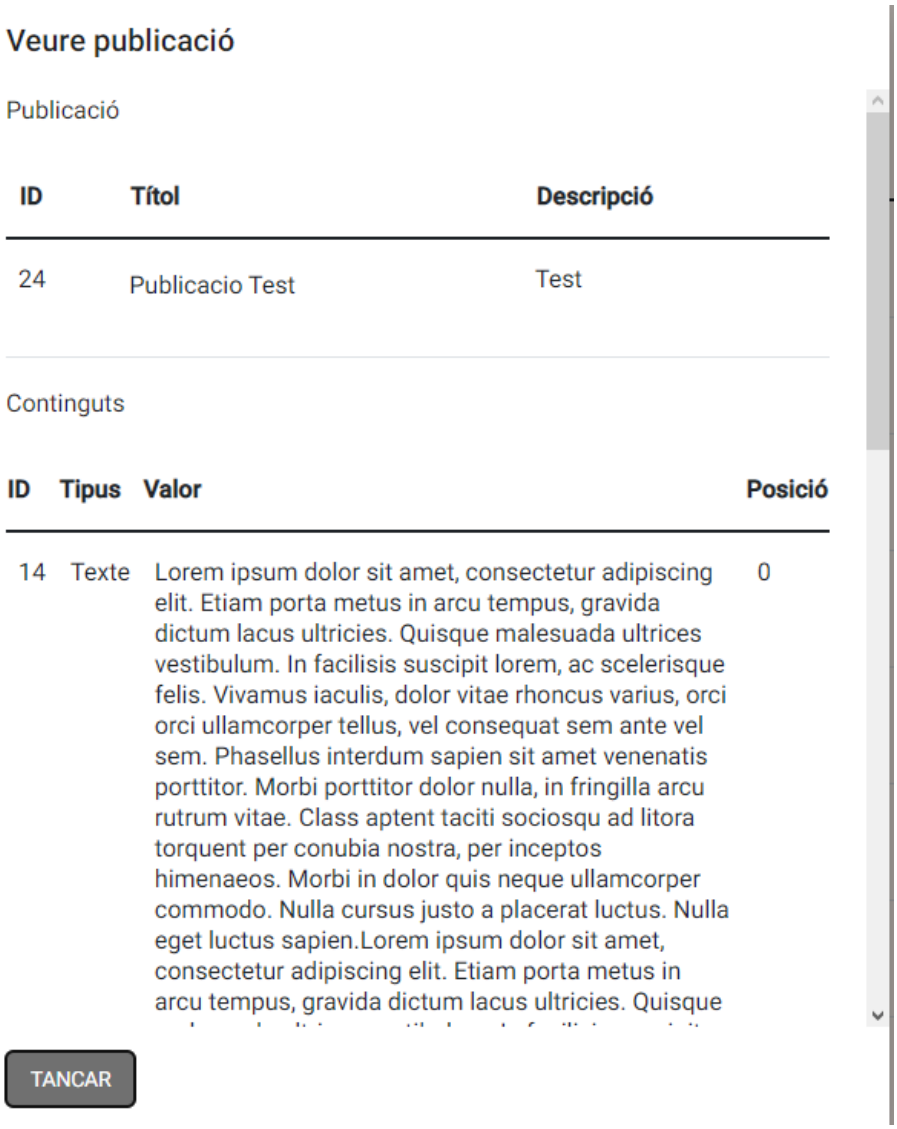

Figura 13.31: Exemple de visualització dels continguts d'una publicació

En cas d'executar l'acció d'Esborrar ens sortirà un popup per confirmar l'acció, el mateix passa amb l'acció de bloquejar autor, surt el mateix popup que el que s'ha mostrat en la Figura 13.7

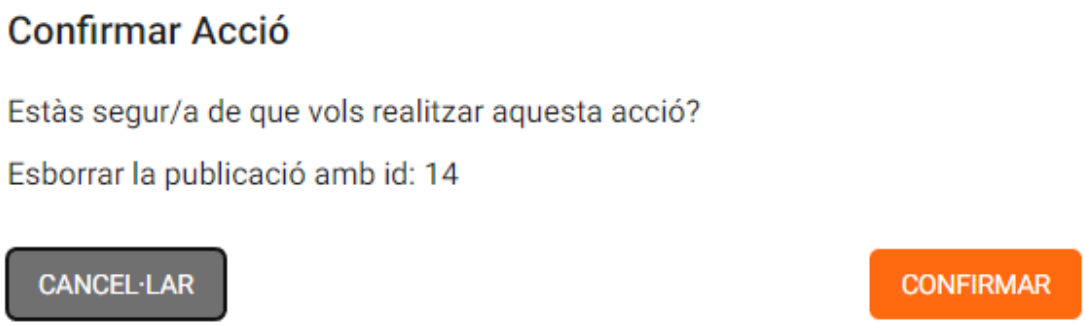

Figura 13.32: Popup per confirmar l'eliminació d'una publicació

Per accedir a la pantalla dels Logs primer hem de navegar a la pantalla principal prement el Logo del backoffice igual que s'ha indicat a la Figura 13.11
A continuació s'ha de prémer el botó de Logs ubicat a la pantalla principal

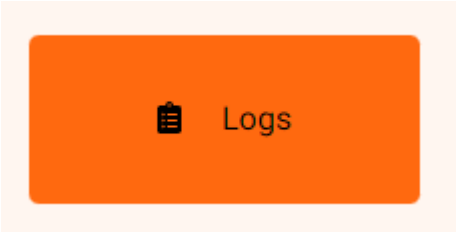

Figura 13.33: Botó per accedir a la pantalla de logs

A l'accedir a la pantalla de logs podrem visualitzar quatre comptadors diferents els quals indiquen diferents coses.

- **Número de logs de tots els tipus** Aquest comptador ens indica el número de tots els logs que genera l'API, en aquest cas es compta tots els logs i també les consultes SQL que es realitzen a la base de dades, és un indicatiu del nombre d'accions que s'han generat a l'API
- **Número de logs informatius** Aquest comptador ens indica el número de logs informatius, en aquest cas són els logs que informen que una acció s'ha dut a terme de forma correcta, com per exemple editar una publicació o bloquejar un usuari, és un indicatiu del nombre d'accions que s'han dut a terme sense cap problema
- **Número de logs d'advertència** Aquest comptador ens indica el número de logs d'advertència, en aquest cas són els logs per les accions que per alguna raó no han acabat bé (consultar una id incorrecte, una acció no permesa...), no és alarmant, però va bé tenir en compte el nombre d'accions que no s'han pogut dur a terme.
- **Número de logs d'error** Aquest comptador ens indica el número de logs d'error, en aquest cas són els logs pels errors generats per l'API, en aquest cas és un indicatiu que existeixen bugs que es troben a l'api i que s'han de solucionar.

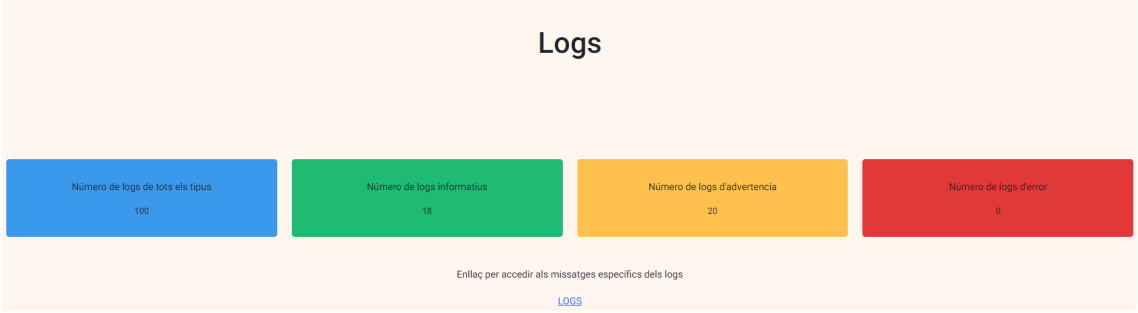

Figura 13.34: Comptadors dels diferents tipus de logs que es generen

Enllaç a l'imatge en alta resolució:
https://i.ibb.co/BCQydBj/LOGS-BO.png

En el nostre cas tenim una integració de Slack amb Papertrail per obtenir els logs en missatges en diferents canals de Slack i des d'alla podem observar en detall cada log generat, l'enllaç indicat en la captura de pantalla porta a l'aplicació de Slack amb els diferents canals esmentats.

En cas que vulguem sortir de la sessió podem fer-ho de dues maneres.
La primera és accedint a la pantalla principal i des d'allà prémer el botó de Sortir

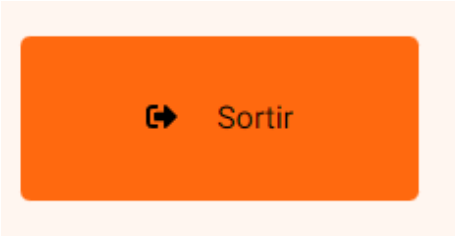

Figura 13.35: Botó per tancar la sessió ubicat en la pantalla principal

També tenim l'opció de fer logout des del botó de Sortir ubicat en el costat superior dret de la pàgina web.

Figura 13.36: Botó per tancar la sessió ubicat en la part superior dreta de la pàgina web

Bibliografia

[1] *Article amb indicacions del cost de contractar com a programador freelance.* https://www.freelancermap.com/blog/es/c-mo-calcular-el-precio-por-tus-servicios-como-freelancer.

[2] *Article orientatiu sobre el preu per hora d'un programador* https://kaira.es/cual-es-el-precio-hora-de-un-programador.

[3] *Article orientatiu sobre el cost que té un dissenyador gràfic* https://www.zaask.es/cuanto-cuesta/disenador-grafico.

[4] *Pàgina oficial sobre el mòdul Sequelize* https://sequelize.org/docs/v6/getting-started/.

[5] *Pàgina oficial sobre el gestor de base de dades Postgress* https://www.postgresql.org/docs/14/index.html.

[6] *Pàgina per consultar tota classe de llibreries sobre qualsevol mena de projecte* https://github.com.

[7] *Pàgina per consultar llibreria de node sobre tots els mòduls disponibles* https://www.npmjs.com.

[8] *Guía per configurar React Native* https://reactnative.dev/docs/next/environment-setup.

[9] *Guía dels components de React Native* https://reactnative.dev/docs/next/components-and-apis.

[10] *Guía dels components de React Native* https://reactnative.dev/docs/next/components-and-apis.

[11] *Guía del com compilar una aplicació React per Android* https://reactnative.dev/docs/signed-apk-android.

[12] *Guía del com compilar una aplicació React per iOS* https://reactnative.dev/docs/publishing-to-app-store.