Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

NEARLY: APLICACIÓ DE DESCOBRIMENT DE NEGOCIS DE PROXIMITAT

Alejandro Cano Calvente

Director: Josep Vañó Chic
Realitzat a: Departament de
Matemàtiques i Informàtica
Barcelona, 20 de juny de 2021

Resum

El projecte desenvolupat tracta sobre l'anàlisi, disseny i desenvolupament d'una aplicació web per a registrar i descobrir negocis de proximitat.

Per a la implementació, s'han utilitzat les eines que proporciona Google Cloud, com ara Google Cloud Sotrage, Google Maps APIs, la base de dades en el núvol MongoDB Atlas i el llenguatge de programació Javascript amb els frameworks ReactJs i ExpressJs.

L'aplicació permet, de manera senzilla, trobar els negocis registrats més propers a l'usuari, així com mostrar la informació rellevant d'aquests: direcció, descripció, imatges, contacte, etc. A més, permet filtrar aquests segons la distància a l'usuari que els visualitza, les categories assignades, etc.

El projecte consta de la implementació dels conceptes vistos durant el grau en assignatures com Enginyeria del Software, Software Distribuït o Computació Orientada al Web.

Resumen

El proyecto desarrollado trata sobre el análisis, diseño y desarrollo, de una aplicación web para registrar y descubrir negocios de proximidad.

Para la implementación, se ha utilizado las herramientas que proporciona Google Cloud, como son Google Cloud Storage, Google Maps APIs, la base de datos en la nube MongoDB Atlas y el lenguaje de programación Javascript con los frameworks ReactJs y ExpressJs.

La aplicación permite de manera sencilla encontrar los negocios registrados mas cercanos a el usuario, así como mostrar la información relevante de estos: dirección, descripción, imágenes, contacto, etc. Además, permite filtrar estos según la distancia al usuario que los visualiza, filtrar por las categorías asignadas, etc.

El proyecto consta de la implementación de los conceptos vistos durante el grado en asignaturas como Enginyeria del Software, Software Distribuït o Computación Orientada al Web.

Abstract

This project consists of the analysis, design and development of a web application for registering and discover proximity businesses.

For the implementation, the tools given by Google Cloud have been used, such as, Google Cloud Storage, Google Cloud APIs, the cloud database system MongoDB Atlas and the programming language Javascript, with the frameworks, ReactJs and ExpressJs.

The application lets the user find the nearest registered businesses in the platform, as well as showing the most relevant information about them: the address, a description, images, contact information, etc. It lets the user filter the business shown by the distance between the user and the business, as well as filter by the categories assigned to each of them.

The project is built with practical and theorical concepts seen during the grade in courses like Enginyeria del Software, Software Distribuït or Computació Orientada al Web.

Agraïments

Als meus pares per ser-hi sempre, per recolzar-me i donar-me ànims en aquesta muntanya russa que ha sigut el grau. Al meu germà Jose Manuel, company i pilar de la meva vida per fer-me veure que tot es possible. I, per últim, a en Ferran, per ser el meu company d'aventures, amic i la llum que em guia. Gràcies per tot el suport i confiança que m'has demostrat durant aquest últim treball del grau.

Índex

Índex de figures

1. Introducció

1.1. Motivacions

Avui dia la gran varietat de comerços a cada municipi proporciona a cada comunitat un ventall de possibilitats on escollir on fer la compra, ja siguin grans cadenes de supermercats, petits comerços independents o comerços internacionals de gran varietat.

La gran majoria de comerços disposen avui dia de presència en el món digital, ja sigui a través de xarxes socials, pàgina web, aparicions en cerques de Google Maps, etc. Però hi ha un nombre reduït de comerços, molts són els de tota la vida, regentats per persones d'edat avançada i sense gaire coneixement tecnològic/web que el seu impacte a internet és mínim. Són aquests comerços precisament els que més perjudicats s'han vist durant aquesta pandèmia, on el comerç online ha sigut clau per a sobreviure.

Nearly neix de la idea d'enaltir i proporcionar reconeixement a aquests comerços, que són de proximitat. Tenir un lloc web on poder descriure el negoci, història i rellevància dels productes.

1.2. Descripció del Projecte

El context d'aquest projecte es crear una aplicació web que es comuniqui amb una base de dades per tal de poder presentar els diferents comerços que ja estiguin registrats prèviament per tal que els usuaris puguin conèixer informació rellevant d'aquests (història, xarxes socials, fotografies i ubicació) així com la distancia a la que cadascun d'aquests està situat de l'usuari.

Per registrar un negoci caldrà estar registrat a l'aplicació prèviament, i aquests no apareixeran a la graella de negocis disponibles fins que no s'hagin revisat prèviament per un administrador del sistema.

L'aplicació web serà desenvolupada completament en Javascript, tant per al *frontend* com per al *backend*. Tot això vindrà completat amb la base de dades NoSQL MongoDB i amb les APIs de Google Maps.

1.3. Objectius

1.3.1. Objectiu General

L'objectiu principal del projecte és dissenyar i implementar una aplicació web amb diferents funcionalitats per poder conèixer quins són els comerços de proximitat a prop de l'usuari. Aquestes funcionalitats es basaran a poder registrar negocis (descripció, ubicació, galeria d'imatges, etc.) així com poder filtrar per categories, cerca per nom i distancia a l'usuari.

1.3.2 Objectius Específics

L'objectiu general és aprofundir en diferents coneixements relacionats amb el desenvolupament d'aplicacions web. Algunes de les tecnologies que es tractaran són les següents:

- Llenguatge Javascript amb els *frameworks* ExpressJs i ReactJs.
- Base de dades NoSql MongoDB allotjada en un clúster *free tier* distribuït de MongoDB Atlas.
- Entorn de desenvolupament amb Docker per minimitzar dependències al sistema.
- APIs de Google Maps per geolocalitzar cada negoci i obtenir informació rellevant (distància, puntuació mitjana a les *reviews* de Google). Aquestes són:
    - Geocoding API
    - Places API
    - Distance Matrix API
    - Local Context API
- Deploy de cost gratuït a les plataformes Netlify (frontend) i Heroku (backend)

1.4. Justificació de la metodologia

La metodologia més adient per aquest projecte, tenint en compte el temps i el desenvolupament que hi ha, era emprar una metodologia *agile*, en aquest cas Kanban. Aquesta metodologia es caracteritza per subdividir les diferents etapes del procés de desenvolupament en tasques petites, ja sigui el disseny d'un component, la implementació, realització de casos d'ús, etc.

Ha sigut un procés iteratiu que m'ha permès desenvolupar diferents parts de l'aplicació independentment les unes de les altres i anar fent pujades al servidor (producció) amb un producte estable fins arriba a tenir un MVP (minimmum viable product).

2. Requisits

2.1 Funcionals

Els requisits funcionals són els que satisfan les funcionalitats principals de l'aplicació ha d'oferir als usuaris.

Autentificació d'usuaris:

- Els usuaris s'han de poder registrar amb un e-mail i poder recuperar la contrasenya si no recorden aquesta.
- La sessió d'usuari en un dispositiu s'ha de mantenir durant un període de temps il·limitat mentre que l'usuari no tanqui la sessió per tal de mantenir continuïtat a l'aplicació.

Negocis:

- Els negocis registrats han d'oferir un seguit d'informació mínima per poder identificar-los i fer que siguin atractius de cara a l'usuari final. És per això que hauran d'incloure un seguit d'imatges, enllaços a les xarxes socials de les quals disposin, així com una descripció del negoci per poder treure a relluir les bondats d'aquest.
- La visualització dels negocis registrats ha de permetre filtrar els resultats d'aquests que apareixen en funció de la distància a l'usuari, un seguit de categories o per un camp de text.
- Les imatges dels negocis han d'estar presents sempre que un negoci estigui referenciat per tal que els usuaris els identifiquin més ràpid.
- La ubicació dels negocis ha de ser la real, es per això que s'utilitzarà les API de Google Maps per obtenir la informació adient de cada negoci.

2.2 No funcionals

Els requisits no funcionals són aquells que ha de tenir present l'aplicació, per tal d'aconseguir un correcte funcionament, tant a nivell visual com a nivell tècnic. És per això que aquesta aplicació web es basa en requeriments de disseny de la interfície.

- L'aplicació per tal de mostrar un funcionament correcte, informarà de diferents esdeveniments que succeeixen a l'aplicació. Farà validació de tots els camps obligatoris als formularis abans d'executar qualsevol acció.
- Tots els desplegables de informació, hauran d'estar ordenats alfabèticament i algun d'ells dependrà de privilegis segons l'organització.
- Les icones de les diferents accions de l'aplicació hauran de ser intuïtius, fent una correspondència correcta entre la icona i la descripció d'aquella acció.

2.3 Casos d'ús

2.3.1. Diagrama de casos d'ús

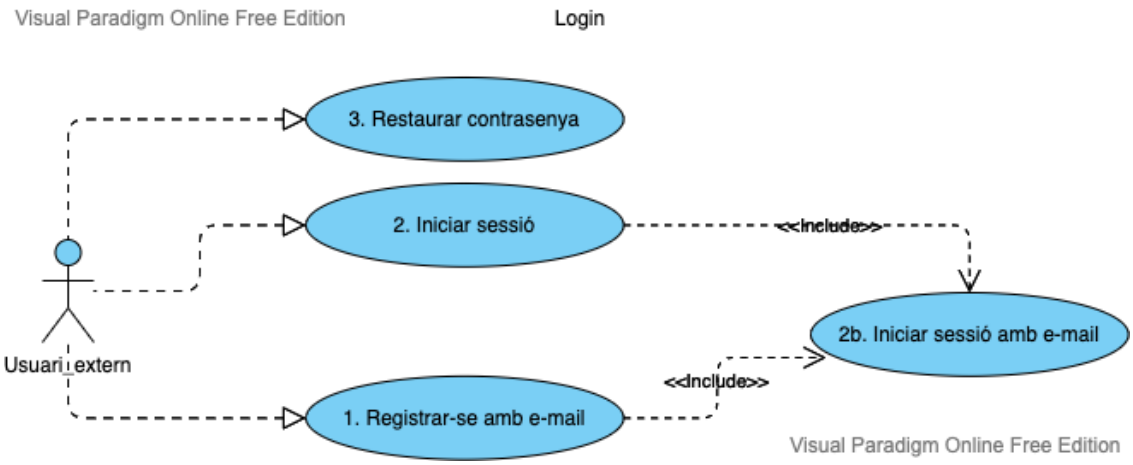

Figura 1: Diagrama de casos d'ús a l'estat de login

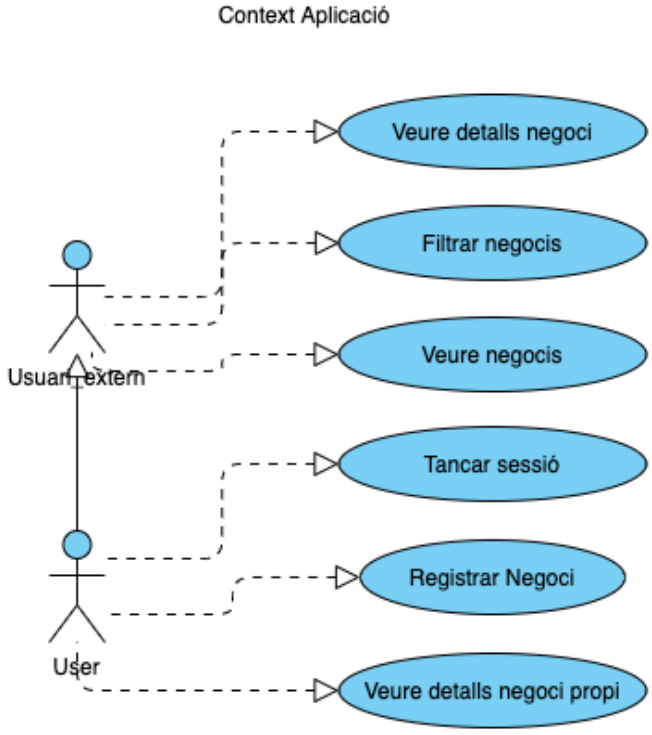

Figura 2: Diagrama de casos d'us contextuals de l'aplicació

2.3.2 Casos d'ús textuals

Els casos d'ús textuals, descriuen els casos d'ús amb els requeriments i accions necessàries per passar d'un estat a un altre. A continuació, es mostren els casos d'ús més rellevants de l'aplicació.

Descripció:	**UC01-Registre amb e-mail**
Actors:	Usuari Extern
Precondicions:	L'usuari ha de disposar d'un compte d'e-mail. L'usuari no ha de tenir sessió iniciada.
Flux bàsic:	1. Usuari: Selecciona l'opció "Registra't" a la navbar. 2. Sistema: Redirigeix a l'usuari a una altra pantalla amb un formulari corresponent al nom, e-mail i contrasenya. 3. U: Emplena el formulari introduint el nom, e-mail, contrasenya i confirmació de contrasenya. Finalment prem "Registrar-me". 4. S: Registra l'usuari i inicia sessió. Redirigeix a l'usuari a la pantalla principal.
Flux Alternatiu:	4a. Si els camps no són vàlids, no permet registrar a l'usuari. 4b. Si ja existeix un usuari amb el mateix e-mail registrat es mostra un missatge d'error.
Postcondicions:	L'usuari es registra i inicia sessió en l'aplicació

Figura 3: Cas d'ús textual: Registre amb e-mail.

Descripció:	**UC02-Iniciar sessió amb e-mail**
Actors:	Usuari Extern
Precondicions:	L'usuari ha d'estar registrat a l'aplicació. L'usuari no ha de tenir sessió iniciada.
Flux bàsic:	1. Usuari: Selecciona l'opció "Entra" a la navbar. 2. Sistema: Redirigeix a l'usuari a una altra pantalla amb un formulari corresponent a l'e-mail i contrasenya. 3. U: Emplena el formulari introduint e-mail i contrasenya Finalment prem "Entra". 4. S: Comprova les dades introduïdes i inicia sessió. Redirigeix a la pantalla principal.
Flux Alternatiu:	4a. Si els camps no son vàlids, no permet iniciar sessió. 4b. Si el camp e-mail no s'ha registrat anteriorment, o si la contrasenya és invàlida, mostra un missatge d'error.
Postcondicions:	L'usuari inicia sessió en l'aplicació.

Figura 4: Cas d'ús textual: Iniciar sessió amb e-mail.

Descripció:	UC03-Registrar negoci
Actors:	Usuari logat
Precondicions:	L'usuari ha d'estar registrat a l'aplicació. L'usuari ha de tenir sessió iniciada.
Flux bàsic:	1. Usuari: Selecciona l'opció "Registrar Negoci" en el desplegable que surt al fer clic sobre la icona d'usuari a la navbar. 2. Frontend: Redirigeix a l'usuari a una altra pantalla amb un formulari corresponent als camps escrits d'un negoci (Nom, Descripció, Ubicació, etc.). 3. U: Emplena el formulari introduint les dades. Finalment prem "Entra". 4. F: Comprova les dades introduïdes i les envia al Backend. Redirigeix a la pantalla de registre d'imatges. 4b. Backend: Geolocalitza el negoci i guarda les coordenades. 5. U: Puja les imatges que desitgi i representin el negoci i prem el botó "Pujar Imatges". 6. F: Redirigeix a l'usuari a la pàgina per veure el detalls del negoci que acaba de crear. 6b. B: Puja les imatges a Cloud Storage i guarda les url en referencia al negoci.
Flux Alternatiu:	4c. Si els camps no son vàlids, no permet i crear el negoci
Postcondicions:	El negoci queda registrat en l'aplicació

Figura 5: Cas d'ús textual: Registrar negoci

Descripció:	UC04-Veure detalls de negoci (propi)
Actors:	Usuari logat
Precondicions:	L'usuari ha d'estar registrat a l'aplicació. L'usuari ha de tenir sessió iniciada.
Flux bàsic:	1. Usuari: Selecciona l'opció "El meu negoci" en el desplegable que surt en fer clic sobre la icona d'usuari a la navbar. 2. Frontend: Redirigeix a l'usuari a una altre pantalla amb tota la informació del negoci.
Flux Alternatiu:	
Postcondicions:	

Figura 6: Cas d'us textual: Veure detalls negoci (propi)

Descripció:	UC05-Visualitzar negocis registrats
Actors:	Usuari extern, Usuari logat
Precondicions:	
Flux bàsic:	1. Usuari: Selecciona l'opció "Som-hi" a la pàgina principal. 2. Frontend: Redirigeix a l'usuari a la graella de negocis.
Flux Alternatiu:	
Postcondicions:	L'usuari visualitza correctament els negocis registrats i aprovats a la plataforma

Figura 7: Cas d'ús textual: Visualitzar negocis registrats

3. Disseny

3.1. Arquitectura

En aquest projecte desenvolupat en Javascript utilitzant ReactJs i ExpressJs, s'estructurà amb una *single page application* (SPA) i una API REST. La API REST que exposa el backend son un conjunt d'*endpoints* fonamentats en una base Model-Controlador. Els models, són els que defineixen com ha de ser les nostres dades o entitats. Sobre aquetss models es realitza un CRUD (create, read, update, delete) mitjançant un servei de domini per a cada model. Aquests serveis són els que cada controlador utilitza per realitzar accions sobre els models. Cada controlador està assignat a un conjunt de rutes (endpoints) que fan de punt d'entrada. A més, cada ruta utilitza serveis entremitjos (middlewares) per realitzar accions comunes com validació de dades d'entrada o control de permisos d'usuari.

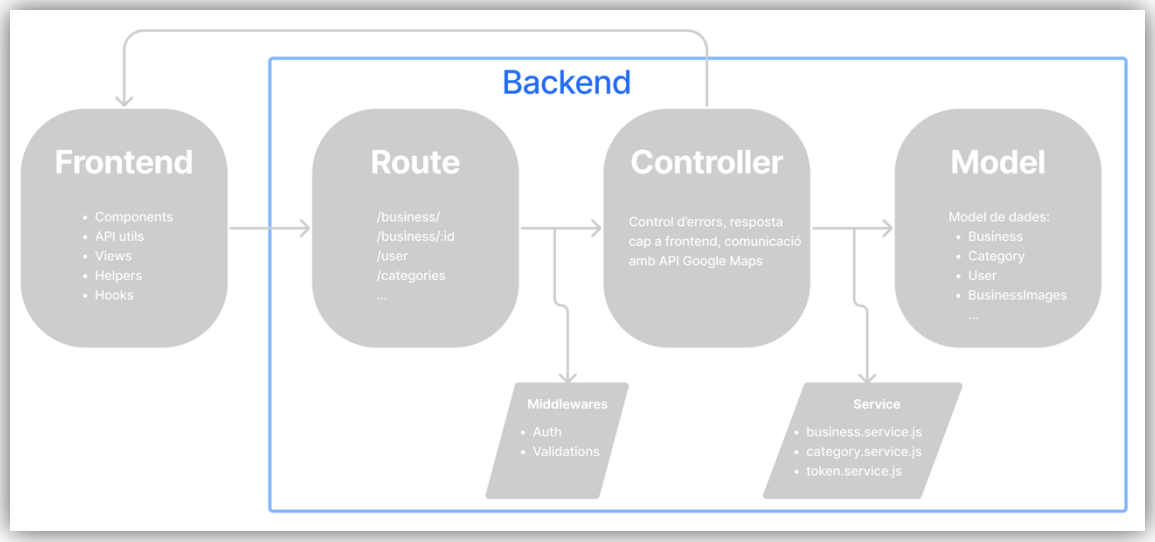

Figura 8: Diagrama arquitectura.

3.2. Estructuració d'entitats

L'entitat principal de l'aplicació és el Business. Aquesta entitat està relacionada amb tres entitats més: BusinessImages, Category i User. Les relacions entre elles són:

- Business 1 a 1 amb BusinessImages.
- Business 1-n amb Category
- Business 1 a 1 amb User.

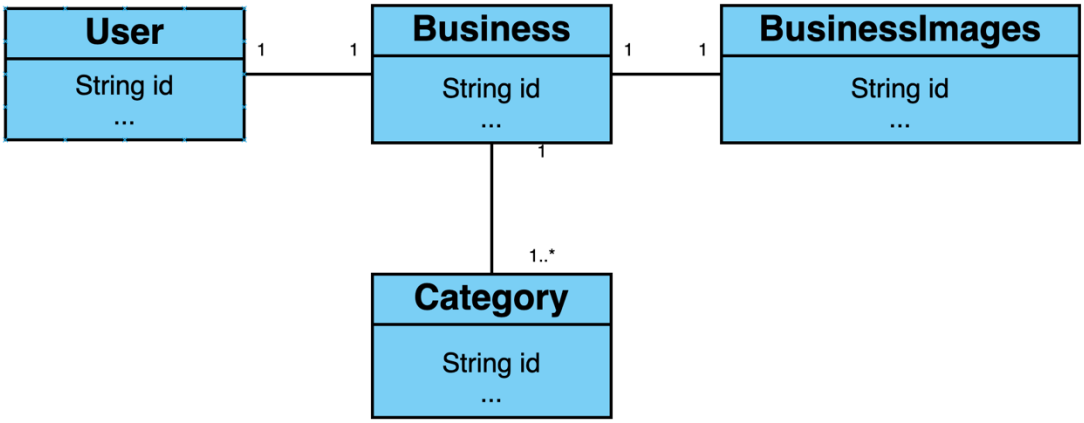

Figura 9: Diagrama relacions d'entitats.

A continuació, es mostren representades les entitats de la plataforma amb les seves propietats i relacions totals:

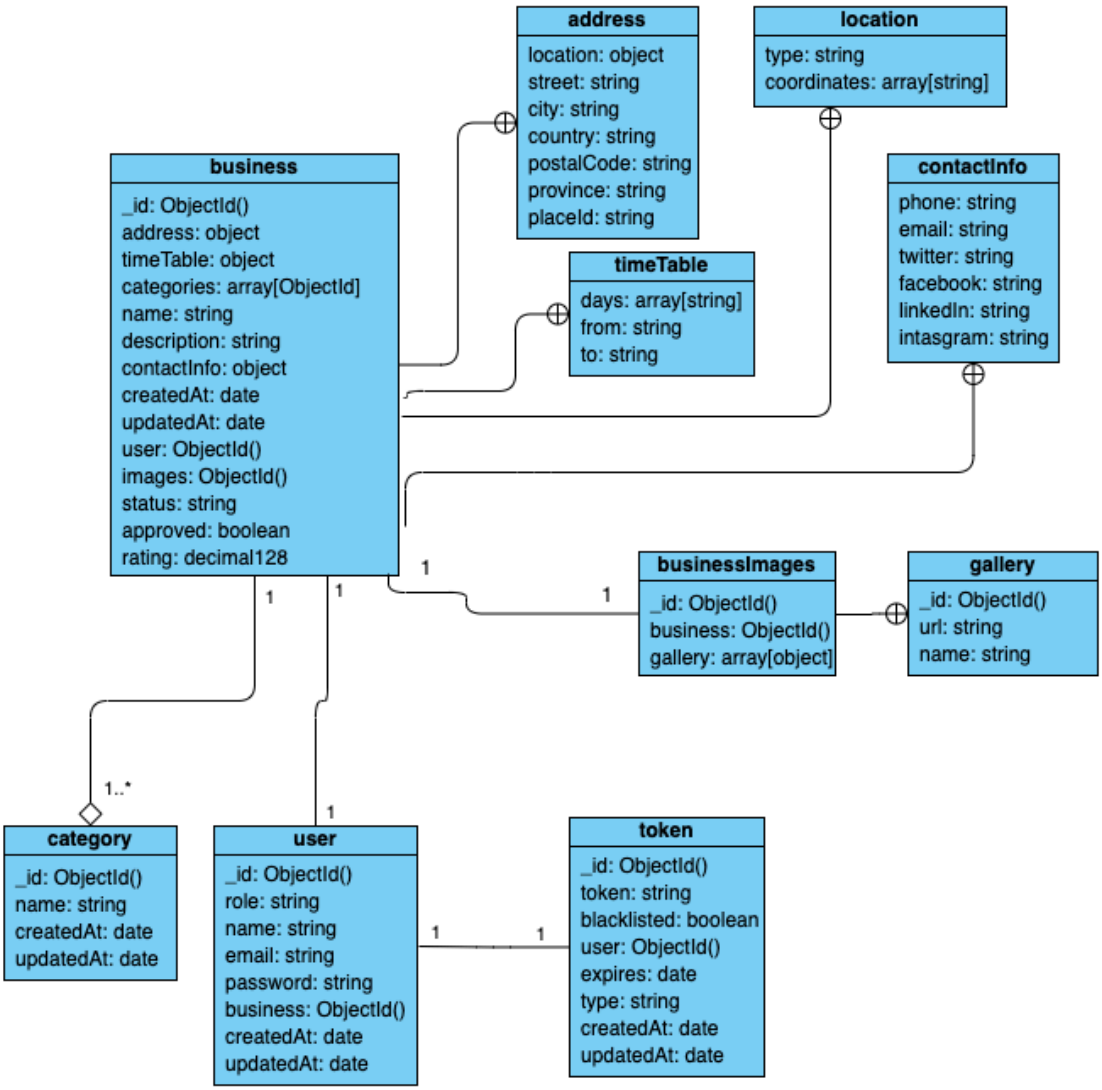

Figura 10: Diagrama d'entitats.

3.3. Pantalles

Les següents pantalles de l'aplicació són les que permetran mostrar el diferent contingut de l'aplicació a l'usuari. Aquestes pantalles mostraran els components necessaris per a cada funcionalitat de l'aplicació.

A continuació es mostra una taula amb les diferents pantalles de l'aplicació, una descripció de la pantalla i els rols que s'apliquen a aquestes:

Pantalla	Descripció	Rols que apliquen
Login	És la pantalla que mostra el formulari per iniciar sessió	Usuari_extern
Registre de compte amb e-mail:	És la pantalla, necessària per iniciar sessió per als usuaris. Aquesta pantalla recull un formulari amb les dades necessàries, perquè l'usuari pugui iniciar sessió a l'aplicació.	Usuari_extern
Registrar Negoci	Aquesta pantalla, mostra el formulari amb els camps necessaris per identificar un negoci. L'usuari ha de registrar tots els camps requerits per poder enviar el formulari.	Usuari_logat
Registrar Imatges Negoci	Aquesta pantalla, mostra un formulari per afegir imatges al negoci creat prèviament. L'usuari pot pujar un total de 10 imatges.	Usuari_logat
Detalls Negoci Propi	És la pantalla on l'usuari registrat pot veure la informació que ha emplenat prèviament sobre el seu negoci junt amb les imatges.	Usuari_logat
Home	Aquesta és la pantalla principal de l'aplicació la qual mostra informació sobre l'aplicació i un enllaç per anar a la pantalla de visualització de negocis	Usuari_logat Usuari_extern
Graella Negocis	Aquesta pantalla mostra tots els negocis registrats en l'aplicació i que hi hagin estan aprovats prèviament per l'administrador. L'usuari que visualitza la pantalla pot filtrar els negocis que apareixen per text o per categories	Usuari_logat Usuari_extern
Detall Negoci Registrat	Aquesta pantalla és on l'usuari (independentment dels permisos) pot veure la informació relacionada d'un negoci així com la galeria d'imatges d'aquest.	Usuari_logat Usuari_extern

Figura 11: Definició pantalles.

Els colors de l'aplicació són senzills. En aquest cas el blanc és el color predominant en menús i pantalles, mentre que s'utilitza un gradient de 2 colors com accents. Aquests 2 gradients s'utilitzen en els botons de l'aplicació. Els gradients són inversos l'un de l'altre, un determina una acció principal i l'altre determina accions secundàries.

4. Implementació

En aquest apartat veurem tot el desenvolupament que s'ha donat per poder realitzar l'aplicació amb els requisits generals i específics. Per això, comentarem la configuració del projecte, estructuració de projecte i cadascuna de les implementacions de les pantalles.

4.1. Configuració del projecte

Per tal de poder aplicar tots els requisits en el projecte, s'han utilitzat eines essencials en el desenvolupament. Com a IDE s'ha fet ús de Visual Studio Code el qual ens proporciona eines de desenvolupament en Javascript com és el debugger en NodeJS i el debugger en navegadors (Google Chrome, Mozzilla Firefox, Opera o Safari). A part, el número d'extensions que es poden afegir a l'IDE el fan una eina essencial. Com que el backend en l'entorn de desenvolupament corre en un contenidor de Docker, l'extensió de Docker per a VSC ha sigut imprescindible i de molta utilitat.

La infraestructura del servidor (backend) en l'entorn de desenvolupament està orquestrada en contenidors de Docker. Docker és un projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors, proporcionant així una capa d'abstracció i virtualització d'aplicacions entre diferents sistemes operatius. D'aquesta manera en l'entorn de desenvolupament local no ha fet falta instal·lar cap de les dependències que té el backend (NodeJs i les seves llibreries o MongoDb) sinó que s'han utilitzat imatges ja generades per la comunitat. Per poder orquestrar la posada en marxa dels contenidors s'ha fet ús de Docker Compose, el qual ens permet gestionar l'execució de tots els contenidors en un ordre determinat i controlar les dependències entre ells (dades, xarxa interna, ports de comunicació, etc.).

La base de dades escollida pel projecte ha sigut MongoDB, ja que aquesta treballa amb BSON els quals són una implementació de JSON (JavaScript Object Notation). Gràcies a això la transformació de dades entre base de dades,

backend i frontend és gairebé nul·la i per tant són homogènies. La base de dades per al servidor de producció està allotjada en un free tier clúster de MongoDB Atlas, per tant s'ha hagut de configurar.

A nivell de backend, el framework de Javascript ExpressJs, ens aporta un seguit d'eines com middlewares, routers o eines per treballar amb el body de les requests. Això, però, no és suficient pels requisits del projecte. Per tant s'han hagut d'instal·lar un seguit de llibreries externes. A continuació es descriuen un breu conjunt i la seva utilitat:

- **Mongoose**. És una llibreria ODM (Object Data Modeling) la qual ens permet gestionar les relacions entre les dades, ens dona una validació d'schemas i s'utilitza per traduir els objectes emmagatzemats a la base de dades en codi.
- **Joi**. És una llibreria per validar les dades que arriben a cadascun dels endpoints de l'API Rest.
- **Google Cloud Storage**. Aquesta llibreria ens permet interactuar amb el servei d'emmagatzematge de fitxers al núvol de Google per tal de guardar les imatges dels negocis registrats.
- **Google Mapis Services**. Llibreria que conté la implementació de les APIs de Google Maps disponibles (Geocoding, Geolocate, Distance Matrix, etc.).
- **Bcrypt**. Llibreria per poder codificar i generar un hash segur per les contrasenyes dels usuaris.
- **Jsonwebtoken**. Aquesta llibreria ens permet generar tokens d'autenticació, segurs i fiables, amb límits de temps per a cada usuari registrat en la plataforma.

El frontend s'ha desenvolupat amb ReactJs utilitzant el boilerplate que ens proporciona la comanda de npm: create-react-app. Com passa amb el framework ExpressJs, el conjunt d'utilitats que ens proporciona ReactJs és limitat i per tant s'han hagut d'afegir llibreries externes. A continuació es descriu un breu conjunt i la seva utilitat:

- **Chakra UI.** Aquesta llibreria ens proporciona un seguit de components ja definits i útils. Des de components per gestionar la distribució del contingut de les vistes, components per generar formularis, botons, icones, etc.
- **Filepond.** Aquesta llibreria ens proporciona un component en esteroides per poder gestionar la càrrega d'imatges en la plataforma.
- **React Google Places Autocomplete.** Aquesta llibreria ens proporciona un component i un seguit de "hooks" per poder cercar direccions o negocis gràcies a l'API de Google Maps.

4.2. Estructuració de directoris del projecte

4.2.1. Backend

El codi del projecte s'ha estructurat en diferents directoris per tal de tenir les responsabilitats separades, com es mostra a la figura XX.

El projecte està estructurat en 9 directoris:

config:

- Aquest directori conté els fitxers de configuració del projecte així com d'algunes de les llibreries externes instal·lades.

controllers:

- Aquest directori conté els diferents controladors amb les funcions que executaran el CRUD de cada entitat del nostre projecte, així com un fitxer index.js que serveix d'entrypoint al directori.

docs:

- En aquest directori trobem la representació en el fitxer components.yml de la documentació relacionada sobre la nostra API.

middlewares:

- Aquest directori conté els diferents serveis entremitjos que s'utilitzen a les routes del nostre projecte (validacions, autenticació, errors i temps màxim de resposta).

models:

- Cada fitxer d'aquest directori conté la definició de l'schema de Mongoose de cada entitat del sistema que després es tradueix a l'hora d'interactuar amb la base de dades MongoDB.

routes:

- A cada directori dintre de routes es definiran els endpoints de cada recurs de l'API REST. Aquests directoris serviran per nomenar les diferents versions de l'API.

services:

- Conté els diferents serveis de cada entitat que realitzen un CRUD sobre elles. Són els serveis els que s'encarregaran de comunicar-se amb la base de dades mitjançant mètodes de Mongoose.

utils:

- Conté eines que realitzen accions dins del context del projecte, fora de la lògica de negoci. En aquest cas, les funcions de geolocalització i mètodes per controlar la sincronitat de les Promise de Javascript.

validations:

- Conté els schemas de validació de dades d'entrada amb Joi per als endpoints de cada recurs.

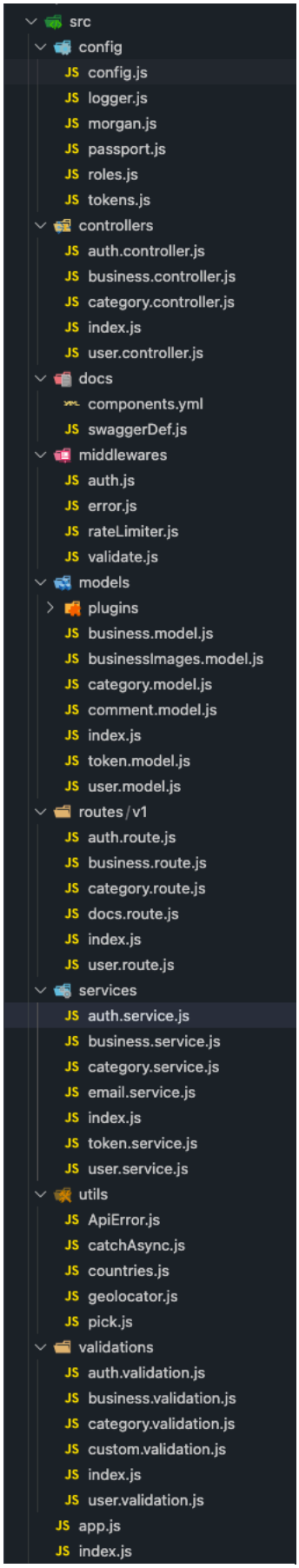

Figura 12: Estructura directoris backend.

4.2.2. Frontend

El codi del projecte en frontend s'ha estructurat en directoris agrupant els components independents i reutilitzables per una banda i les vistes de cada pantalla per una altra:

components:

- Conté els components de React creats pel context de l'aplicació els quals contenen lògica de negoci.

helpers:

- Conté les eines i serveis necessàries per a lògica de negoci. En aquest cas, la definició dels mètodes que serveixen per comunicar-se amb el backend.

hooks:

- Els hooks en React són funcions de Javascript amb context que contenen lògica de l'aplicació, els quals poden ser invocats des de qualsevol component del projecte mantenint les dades entre un i altres. Serveixen per mantenir "l'state".

utils:

- Conté eines que realitzen accions dins del context del projecte, fora de la lògica de negoci. En aquest cas, les funcions de càlcul de distància entre negoci i usuari.

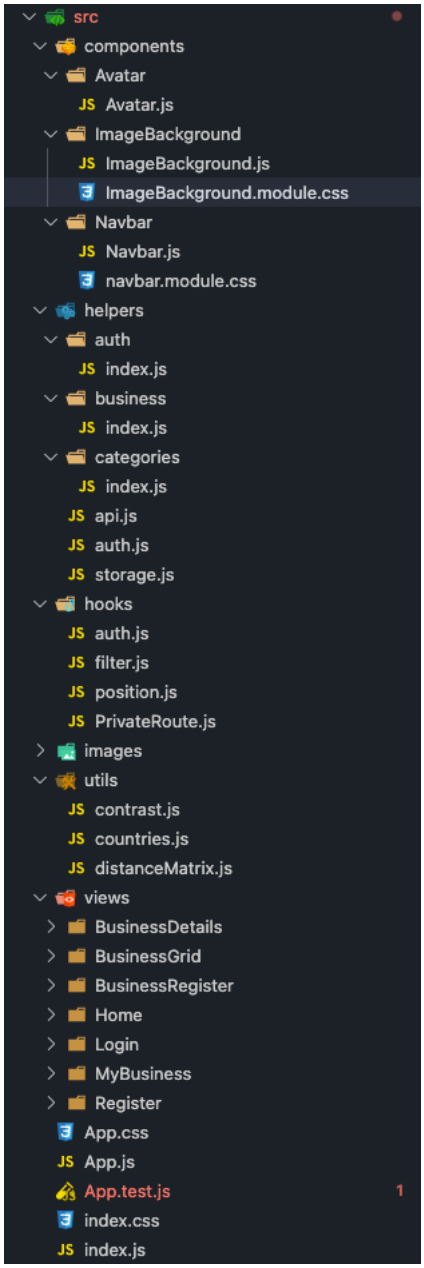

Figura 13: Estructura directoris frontend.

views:

- Cada subdirectori conté els components exclusius de cada vista de l'aplicació. Aquests components són els que mantenen cadascú el seu estat i el comparteixen entre ells mitjançant hooks i la jerarquia parent-children de React.

4.3. Autenticació

Per a poder tenir un registre de l'usuari a l'aplicació, s'ha creat un sistema d'autenticació basant en JSON Web Token (JWT). El JWT és un estàndard obert (és públic, però amb certs drets associats) basat en JSON per a la creació de tokens d'accés que permeten la propagació de la identitat de l'usuari i els privilegis d'aquest relacionats amb el rol establert dins del context de l'aplicació. Aquest estan signats per una clau privada del servidor que els genera (la clau ha de ser coneguda pel client) i d'aquesta manera els dos poden verificar la veracitat d'aquests.

4.3.1. Registre

El registre és el mètode en el qual un usuari es registre a la plataforma amb un correu electrònic i una contrasenya.

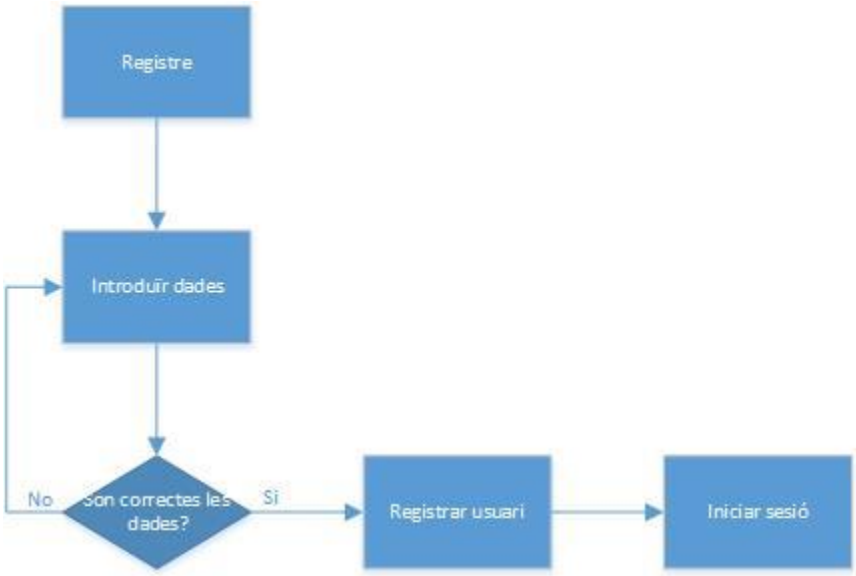

Figura 14: Diagrama de flux: Registre d'un usuari amb correu electrònic/contrasenya.

La pantalla per tant consta de 4 camps, l'e-mail, el nom de l'usuari, contrasenya i la confirmació per fer una doble verificació. Un usuari no es podrà registrar si el compte de correu electrònic ja hi és registrat prèviament. A part d'aquesta comprovació es duen a terme altres comprovacions. El correu electrònic es valida, de tal forma que el format del correu sigui vàlid. Igualment amb la contrasenya, aquesta ha de tenir una llargada de 8 caràcters i els dos camps han de coincidir en contingut per poder enviar el formulari.

Un cop el servidor rep les dades, comprova que no existeixi cap usuari registrat amb el mateix e-mail i encripta la contrasenya (sempre es guarda el *hash* de la contrasenya, mai el seu valor original).

Un cop l'usuari es registra correctament, es genera una nova entrada a MongoDb on l'usuari queda identificat per un ObjectId. Aquest ObjectId és generat per MongoDb i és un valor de 12 bytes que consisteix en:

- 4 bytes on el seu valor és la data en segons de creació de l'objecte mesurada des de la creació de l'Unix epoch.
- 5 bytes amb un valor aleatori.
- 3 bytes amb valor d'un comptador incremental inicialitzat en un valor aleatori.

Juntament amb l'objecte de l'usuari a la base de dades, es genera un segon objecte dins la *collection* tokens on es referencia el'ObjectId de l'usuari creat.

Aquest objecte conté la informació del JWT generat per l'usuari:

- Valor del token
- Tipus
- Data de validesa del token
- Data de creació
- Data de modificació

El valor del token i la data de validesa es modifiquen cada cop que l'usuari fa login a l'aplicació.

La informació de l'usuari i el token es retornen al frontend el qual els emmagatzema i canvia l'estat de *logged out* a *logged in*.

4.3.2. Inici de sessió

El funcionament de l'inici de sessió és senzill doncs s'ha d'emplenar un formulari de 2 camps: e-mail i contrasenya. El formulari primer de tot valida que el format de l'e-mail sigui vàlid abans d'enviar les dades al servidor. El servidor llavors comprova que l'e-mail introduït pertanyi a un usuari registrat a la plataforma, si és correcte, s'encripta la contrasenya enviada per l'usuari i es compara la firma generada per aquesta amb la firma de la contrasenya associada a l'usuari amb l'e-mail a base de dades. Si aquesta és correcte es genera un nou token que s'envia junt amb la informació de l'usuari al frontend.

4.4. Registre d'un negoci

El registre de negoci és la part amb mes lògica dins l'aplicació per l'ús de les eines de Google Cloud: Storage i les APIs de Google Maps.

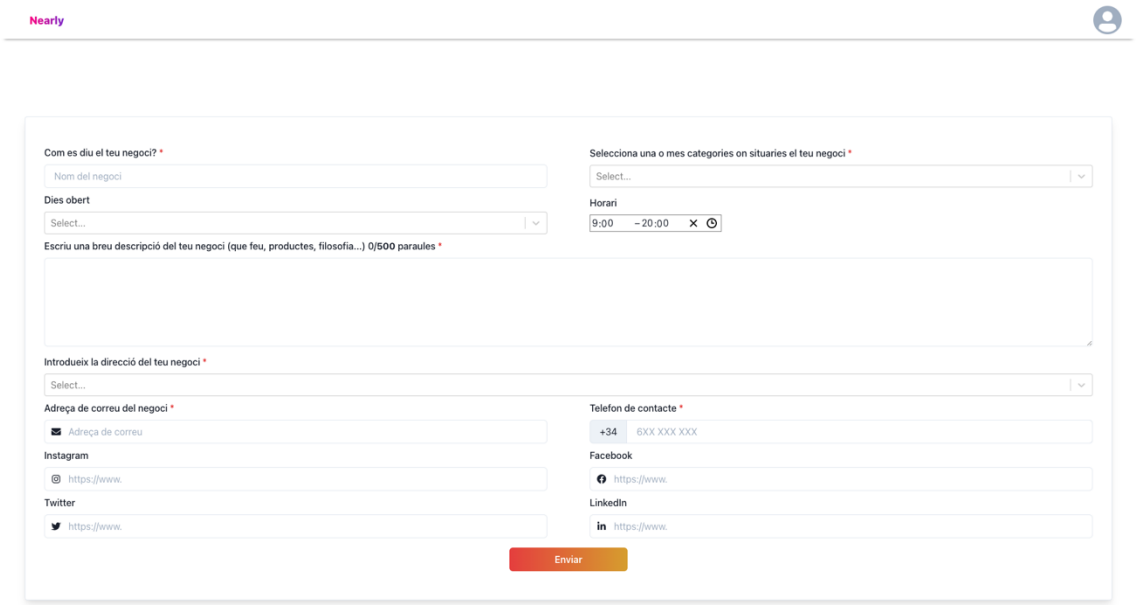

Figura 15: Formulari de registre d'un negoci.

L'usuari ha d'emplenar el formulari de creació, on els camps marcats amb un asterisc són obligatoris. Els camps, telèfon, e-mail i xarxes socials tenen validacions de formats. L'apartat més important d'aquest formulari es el camp per introduir l'adreça física del negoci.

Aquest component està construït sobre el component Select Autocomplete de Google Maps. El que ens permet és a mesura que s'escriu, buscar l'adreça tal com està en els servidors de Google Maps. Ens serveix tant per buscar una adreça completa utilitzant el nom del carrer, ciutat, país, província i codi postal, o bé introduir el nom d'un negoci ja existent. El component internament retornarà el place_id de l'adreça. Aquest place_id és la referència única que té Google Maps per referenciar un parell de coordenades (latitud i longitud) en el mapa. Aquest component no ens retorna les coordenades en si.

Quan s'envia el formulari emplenat al backend, aquest s'encarrega de recollir tota la informació referent a la ubicació del negoci utilitzant la API Geocode de Google Maps. Per fer-ho, necessitem passar-li el place_id que hi ha a la request. La API ens retornarà un objecte JSON amb la informació que Google Maps té sobre el place_id:

```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "1600",
               "short_name" : "1600",
               "types" : [ "street_number" ]
            },
            {
               "long_name" : "Amphitheatre Pkwy",
               "short_name" : "Amphitheatre Pkwy",
               "types" : [ "route" ]
            },
            {
               "long_name" : "Mountain View",
               "short_name" : "Mountain View",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Santa Clara County",
               "short_name" : "Santa Clara County",
               "types" : [ "administrative_area_level_2",
"political" ]
            },
            {
               "long_name" : "California",
               "short_name" : "CA",
               "types" : [ "administrative_area_level_1",
"political" ]
            },
            {
               "long_name" : "United States",
               "short_name" : "US",
               "types" : [ "country", "political" ]
            },
            {
               "long_name" : "94043",
               "short_name" : "94043",
               "types" : [ "postal_code" ]
            }
         ],
         "formatted_address" : "1600 Amphitheatre Parkway,
Mountain View, CA 94043, USA",
         "geometry" : {
            "location" : {
               "lat" : 37.4224764,
               "lng" : -122.0842499
            },
            "location_type" : "ROOFTOP",
            "viewport" : {
               "northeast" : {
                  "lat" : 37.4238253802915,
                  "lng" : -122.0829009197085
               },
               "southwest" : {
                  "lat" : 37.4211274197085,
                  "lng" : -122.0855988802915
               }
            }
         },
         "place_id" : "ChIJ2eUgeAK6j4ARbn5u_wAGqWA",
         "plus_code": {
            "compound_code": "CWC8+W5 Mountain View,
California, United States",
            "global_code": "849VCWC8+W5"
         },
         "types" : [ "street_address" ]
      }
   ],
   "status" : "OK"
}
```


Figura 16: JSON resposta Geocoding API Google Maps

D'aquest JSON recollirem la informació relacionada amb l'adreça física del negoci, així com les seves coordenades.

Seguidament s'utilitza l'API Places per obtenir les valoracions i puntuacions que els usuaris han donat al negoci a Google Maps. D'aquesta manera l'usuari de la nostra plataforma pot conèixer la valoració general d'aquest negoci. Per últim es referencia l'ObjectId generat al guardar el negoci a base de dades en l'objecte de l'usuari que ha registrat el negoci i viceversa.

El negoci, però, no està registrat del tot encara i el seu estat és "pendingImages". El frontend un cop enviat el formulari redirigeix l'usuari a la pàgina per poder pujar imatges del negoci:

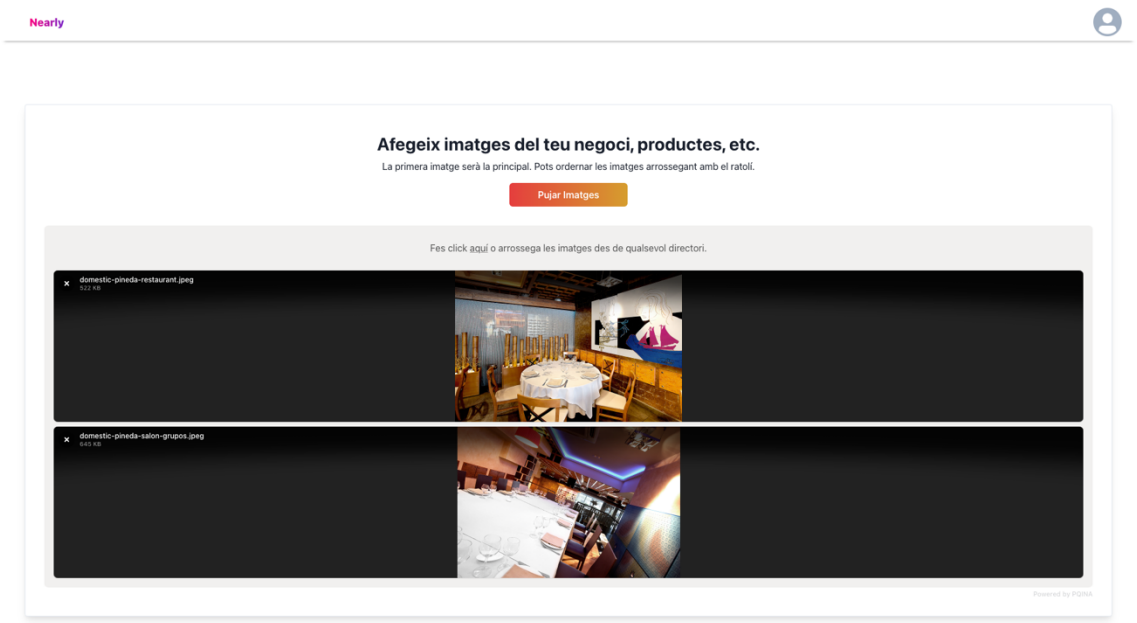

Figura 17: Formulari registre d'imatges

L'usuari pot pujar fins a un màxim de 10 imatges les quals s'enviaran al backend. Per no sobrecarregar la base de dades amb el pes total que pot arribar a tenir cada imatge, es va decidir pujar les imatges al servidor d'emmagatzematge al núvol de Google, Google Cloud Storage. Quan s'envia el formulari, el backend puja cadascuna de les imatges a un directori dins d'un bucket de Google Cloud Storage, guardant-se les url de cadascuna de les imatges pujades. Després genera un objecte BusinessImages on es guarda una referencia a el ObjectId del negoci al qual pertanyen les imatges i una llista amb les url de les imatges. Quan s'hagin de mostrar les imatges, es el frontend l'encarregat de renderitzar-les mitjançant la url de cadascuna.

5. Conclusions

En aquest Treball de Fi de Grau, s'ha dissenyat i implementat, una aplicació web per a descobrir negocis a prop de l'usuari. Aquest treball s'ha assolit gràcies els diferents objectius proposats en l'aplicació i coneixements teòrics del grau, mitjançant les diferents tecnologies aplicades en el projecte, com són el llenguatge Javascript i les bases de dades NoSQL com MongoDb, junt amb les API de Google Maps per poder gestionar la informació geoespacial d'un punt en el mapa. Un dels objectius que havia de complir l'aplicació és que fos intuïtiva i de disseny senzill, per tal que no calguessin uns grans coneixements tecnològics per poder interactuar amb ella.

El disseny s'ha treballat de tal manera que sigui senzill, intuïtiu i accessible per a qualsevol usuari. S'ha treballat cadascuna de les pantalles curosament, utilitzant components especifiques per a cada tipus de dada, intentat que l'experiència d'usuari sigui la més fluida possible. Altres components, totes les pantalles, segueixen la mateixa línia de patró de disseny i colors.

És per això, que el procés de disseny s'ha fet amb cura i deixant-ho obert a futures implementacions. A més, qualsevol usuari tindrà major accessibilitat a la informació de la aplicació, ja que permet inicis de sessió simultanis per un mateix compte a diferents dispositius.

Pel que fa a els objectius generals, l'aplicació compleix amb els requeriments, un usuari, pot crear un negoci, pot visualitzar els negocis registrats a més de filtrar en funció de la distància i categories de cadascun.

En relació al codi, s'ha treballat de tal manera que sigui accessible i intel·ligible per a qualsevol usuari. A més està estructurat de tal forma que per a futures ampliacions, l'estructura sigui flexible, sense haver de modificar molt de codi.

5.1 Treball futur

Pel que fa al treball futur, es pretén implementar funcionalitats noves i adaptar el software a plataformes mòbils (Android i iOS) per maximitzar el nombre d'usuaris. Dins de les funcionalitats a implementar es troba que l'aplicació estigui disponible en diferents idiomes per fer-la més accessible a un nombre encara més gran d'usuaris.

La funcionalitat més important que es podria afegir a la plataforma és afegir un market place on cada negoci pugui posar a la venda els seus serveis i productes. Això seria beneficiós per aquells negocis que no disposen d'una infraestructura web o una plataforma e-commerce on vendre els seus productes.

Respecte als usuaris que existeixen dins de la plataforma, en tenim 2, els externs i els registrats. Idealment podríem afegir un segon tipus d'usuari registrat el qual no hagi de registrar un negoci, sinó interactuar amb la plataforma, deixar comentaris i puntuacions als negocis que vulgui, així com interactuar amb el market place a futur.

Bibliografia

Chakra UI. (2017). Chakra UI Docs. Recuperat el 17 de Abril de 2021 de https://chakra-ui.com/docs/

Docker. (2013). Docker Documentation. Recuperat el 17 de Abril de 2021 de https://docs.docker.com/

Express. (2010). Express API Reference. Recuperat el 17 de Abril de 2021 de https://expressjs.com/es/4x/api.html

Google. (2005). Google Cloud Storage. Recuperat el 17 de Abril de 2021 de https://cloud.google.com/storage/docs/

Google. (2005). *Google Distance Matrix*. Recuperat el 21 de Abril de 2021 de https://developers.google.com/places/

Google. (2005). *Google Maps*. Recuperat el 20 de Abril de 2021 de https://developers.google.com/maps/

Google. (2005). *Google Places*. Recuperat el 21 de Abril de 2021 de https://developers.google.com/places/

Heroku. (2007). Heroku Documentation. Recuperat el 28 de Maig de 2021 de https://devcenter.heroku.com/

Netlify. (2014). Netlify Documentation. Recuperat el 28 de Maig de 2021 de https://docs.netlify.com/

MongoDB. (2009). MongoDB Atlas. Recuperat el 16 de Abril de 2021 de https://docs.atlas.mongodb.com/mongo/

Mongoose. (2011). Mongoose Documentation. Recuperat el 16 de Abril de 2021 de https://mongoosejs.com/docs/guide.html

React. (2013). React Documentation. Recuperat el 19 de Abril de 2021 de https://es.reactjs.org/docs/getting-started.html