Carlos Martínez García-Villarrubia

Conectadrive – Aplicació web de gestió per a empreses d'VTC.

TREBALL DE FI DE GRAU

dirigit per Marc Sánchez

Grau d'Enginyeria Informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

Agraïments

Vull expressar el meu profund agraïment a la meva família per haver-me donat suport en tot moment, fent possible la realització d'aquest projecte. Voldria també agrair l'oportunitat de participar en aquest projecte a Alejandro Bisbal, Nil Monfort, Roger Massana, Nèstor Urquiza, i altres membres de Conectadrive, sense els quals aquest projecte no hauria pogut tirar cap endavant. El rol de cada membre ha sigut clau, cadascun en el seu àmbit, i aquest projecte m'ha donat la oportunitat d'expandir les meves habilitats i participar en el disseny, direcció i execució del desenvolupament d'aquesta aplicació.

Resum.

Aquest projecte, anomenat Conectadrive, té com a finalitat desenvolupar un *SaaS*[1] que permeti a les empreses del sector del transport gestionar els seus serveis i reserves. Aquesta aplicació ofereix un mòdul que actua com a gestor de reserves i permet mantenir un control de les reserves de l'usuari, gestionar vehicles, etc. El segon mòdul consisteix en un algoritme de planificació que actua com a solució al *VRP*[2], bàsicament ofereix la millor distribució dels serveis entre els diferents vehicles dels que disposi l'empresa amb la finalitat d'abastir-los de la forma més eficient.

Per a desenvolupar aquest software s'ha utilitzat PostgreSQL com a base de dades relacional, Vue.js per al desenvolupament del frontend i Spring Boot per al backend. Tota l' arquitectura està dockeritzada i desplegada a un clúster de Kubernetes a *AWS*[3].

Resumen.

Este proyecto, llamado Conectadrive, tiene como objetivo desarrollar un *SaaS*[1] que permita a las empresas del sector del transporte gestionar sus servicios y reservas. Esta aplicación ofrece un módulo que actúa como gestor de reservas y permite mantener un control de las reservas de los usuarios, gestionar vehículos, etc. El segundo módulo consiste en un algoritmo de planificación que actúa como solución al *VRP*[2], ofreciendo básicamente la mejor distribución de los servicios entre los distintos vehículos disponibles de la empresa con el objetivo de abastecerlos de la manera más eficiente.

Para desarrollar este software se ha utilizado PostgreSQL como base de datos relacional, Vue.js para el desarrollo del frontend y Spring Boot para el backend. Toda la arquitectura está dockerizada y desplegada en un clúster de Kubernetes en *AWS*[3].

Abstract.

This project, named Conectadrive, aims to develop a *SaaS*[1] that allows transportation sector companies to manage their services and reservations. This application offers a module that acts as a reservation manager and allows maintaining control of user reservations, managing vehicles, etc. The second module consists of a scheduling algorithm that acts as a solution to the *VRP*[2], basically offering the best distribution of services among the different vehicles available to the company with the aim of supplying them in the most efficient way.

To develop this software, PostgreSQL has been used as the relational database, Vue.js for frontend development, and Spring Boot for the backend. The entire architecture is containerized and deployed on a Kubernetes cluster on *AWS*[3].

---

1 SaaS: Software as a Service

2 VRP: Vehicle Routing Problem

3 AWS: Amazon Web Services

Índex

Índex de taules

Índex de figures

1 Introducció

El sector del transport ha estat sempre clau dins de l'estructura social d'Espanya, facilitant als ciutadans la mobilitat sense la necessitat de fer reserves prèvies i a un cost que es considera relativament assequible. Aquest fet ha repercutit positivament en la dinàmica quotidiana dels ciutadans, ja que els ha permès un accés més fàcil i ràpid a diferents parts de la ciutat o del país sense grans complicacions logístiques o econòmiques.

No obstant, en els últims temps, el sector s'ha trobat amb desafiaments significatius, especialment amb l'aparició i consolidació de companyies com Uber, Cabify o altres empreses que operen sota el model de *VTC*[4]. Aquestes empreses han introduït una nova dinàmica de competència, caracteritzada per models de negoci innovadors que han capturat una porció considerable del mercat, complicant la situació per a les empreses tradicionals de transport.

1.1 Context

En el context de Mallorca, la situació del transport terrestre presenta particularitats que augmenten els reptes del sector. La major part del software utilitzat en la gestió de transports és notòriament obsolet, una realitat que limita la capacitat de les empreses locals per optimitzar els seus serveis i adaptar-se a les demandes canviants del mercat. Aquesta tecnologia antiquada es tradueix en una eficiència operativa reduïda i una falta de capacitats per a l'anàlisi de dades en temps real.

A més, la gestió dels VTC a l'illa no està ben regulada ni coordinada, el que provoca una competència deslleial i confusions entre els operadors. Això afecta negativament tant a empreses tradicionals com a noves entrades, generant un entorn de treball inestable i poc previsible.

Un altre punt crític és l'organització del transport en els punts neuràlgics com els aeroports, on la coordinació entre diferents mitjans de transport és sovint caòtica. Aquesta falta d'organització no només afecta la percepció del servei per part dels usuaris, sinó que també suposa un desafiament logístic significatiu, especialment en temporada alta turística quan l'afluència de passatgers és màxima. Les deficiències en la infraestructura i la planificació resulten en retards, confusió i una experiència generalment insatisfactòria per als viatgers que arriben o marxen de l'illa.

En aquest context, el treball que presentem aposta per un nou enfocament orientat a les empreses emergents de transport. L'objectiu és competir directament amb aquestes grans companyies mitjançant la implementació d'un sistema més adaptat a les necessitats actuals del mercat i a les expectatives dels consumidors moderns. Aquesta iniciativa va néixer després d'entrar en contacte amb Alejandro Bisbal, copropietari d'Avantcab i Conectadrive, qui em va contactar arran d'un anunci que vaig publicar a Fiverr. El projecte va començar amb una sèrie de modificacions a un programari antic del què disposava la seva empresa. Després d'analitzar la competència dins el sector a Mallorca, vam decidir procedir amb el desenvolupament d'aquest nou projecte, anomenat Conectadrive.

[4] VTC: Vehicles de transport amb conductor.

2 Descripció

Aquest projecte consistirà en el desenvolupament d'una aplicació web destinada a millorar la gestió del transport a Mallorca, particularment per a empreses que operen dins del model de VTC. Aquesta aplicació web serà desenvolupada utilitzant tecnologies modernes i eficients per garantir una experiència d'usuari optimitzada i una funcionalitat completa.

L'arquitectura de Conectadrive estarà basada en dos components principals: el frontend i el backend. El frontend de l'aplicació serà desenvolupat amb Primevue [1], una biblioteca de components de Vue.js dissenyada per oferir una interfície rica i adaptable per a aplicacions web. Aquesta elecció permetrà crear una experiència d'usuari agradable i intuïtiva, facilitant la visualització de reserves de clients, la gestió de col·laboracions empresarials, el seguiment dels vehicles en temps real, la gestió de personal, i la importació de dades de reserves d'agències externes.

D'altra banda, el backend funcionarà com una *API*[5] *REST*[6], que serà el motor central per a tota la lògica de negoci i manipulació de dades de l'aplicació. A més, aquest backend inclourà una sèrie de processos automatitzats (crons) programats per realitzar comprovacions regulars i gestionar l'enviament de dades al Ministeri de Transport, garantint el compliment de la normativa vigent.

Per a garantir operativitat i escalabilitat òptimes, desplegarem l'aplicació al núvol utilitzant *AWS*[7]. Utilitzarem un clúster de Kubernetes per a l'orquestració dels contenidors per a millorar la eficiència del desplegament i escalabilitat dels recursos.

A més, implementarem un pipeline de *CI*[8]/*CD*[9] per automatitzar els processos de desenvolupament, proves i desplegament de l'aplicació, que ens permetrà mantenir un cicle de desenvolupament àgil, on les actualitzacions i millores poden ser integrades i desplegades ràpidament i amb mínima intervenció manual.

A més dels seus mòduls principals, l'aplicació oferirà una versió mòbil desenvolupada amb Quasar [2], una eina que permet crear aplicacions lleugeres i eficients per a dispositius mòbils, que a més permet compilar el codi per a obtenir una aplicació nativa. Aquesta aplicació mòbil estarà dirigida als conductors, i oferirà únicament les funcionalitats necessàries pels conductors i així evitar una sobrecàrrega de contingut. Algunes de les funcionalitats podrien ser la visualització de properes reserves o bé la visualització d'itineraris.

[5] API: Interfície de programació d'aplicacions

[6] REST: REpresentational State Transfer

[7] AWS: Amazon Web Services

[8] CI: Integració contínua

[9] CD: Desplegament contínu

2.1 Motivacions

2.1.1 Modernització del Sector

Durant anys, el sector del transport a Mallorca ha mantingut una estructura força estàtica i poc innovadora. La dependència d'infraestructures antigues i la manca d'integració tecnològica han limitat considerablement la capacitat de resposta del sector davant l'increment constant de la demanda, especialment durant la temporada turística alta. ConectaDrive pretén introduir una solució tecnològica avançada per transformar i modernitzar aquest sector. Aquesta modernització no només vol millorar l'eficiència operativa dels serveis de VTC, sinó també elevar l'estàndard de servei, oferint als usuaris una experiència més ràpida, còmoda i personalitzada. La integració de sistemes de reserva en temps real i eines d'anàlisi de dades ajudarà les empreses a optimitzar rutes i recursos, reduint temps d'espera i millorant la planificació del transport.

2.1.2 Millora de la Competitivitat

Amb l'arribada i consolidació de grans plataformes internacionals de VTC, les empreses locals es troben en una lluita constant per mantenir la seva quota de mercat. ConectaDrive proporciona eines essencials per augmentar la competitivitat d'aquestes empreses, com ara la gestió eficient de la flota, l'optimització de rutes basades en l'anàlisi de dades en temps real i una millor gestió de la relació amb el client. Mitjançant una interfície fàcil d'usar i altament adaptable, el projecte permetrà a les empreses de VTC locals oferir un servei que pugui competir en qualitat i cost amb les opcions proporcionades per empreses globals.

2.1.3 Compliment Regulador

Els canvis constants en la regulació del transport a Espanya, especialment en regions amb alta afluència turística com Mallorca, requereixen una adaptació ràpida i eficient per part de les empreses de transport. És per això que aquest software inclourà funcionalitats que asseguren el compliment de les normatives locals i estatals, automatitzant processos com la generació d'informes i l'enviament dels serveis al ministeri de Transport. Aquesta capacitat no només simplifica les operacions diàries sinó que també redueix el risc de sancions i multes per incompliments de la normativa estatal.

2.1.4 Optimització de Recursos

Gestionar eficientment els recursos és crucial per a la rentabilitat en el sector del transport. ConectaDrive està dissenyat per optimitzar l'ús dels vehicles i del personal, reduint els períodes d'inactivitat i millorant la distribució de les càrregues de treball. Aquesta optimització passa per l'anàlisi de patrons de demanda, la previsió de necessitats i l'ajustament dinàmic de les assignacions de recursos, tot plegat amb l'objectiu de maximitzar l'eficiència sense sacrificar la qualitat del servei.

2.2 Objectius del treball

L'objectiu d'aquest treball de fi de grau és oferir una perspectiva tècnica sobre els avantatges de la implementació de tecnologies informàtiques en el sector del transport, un àmbit de creixent importància. El focus del meu estudi es centrarà en el desenvolupament del backend i els aspectes relacionats. En aquest projecte he col·laborat amb companys del

GEI[10], Nil Monfort i Roger Massana. Hem acordat dividir les tasques segons la nostra especialització; mentre ells es concentren en altres àrees, jo assumiré la responsabilitat del backend, desenvolupant i optimitzant els processos i la lògica darrera de l'aplicació ConectaDrive. Aquest enfocament permetrà que cadascun de nosaltres aprofundeixi en la seva especialitat i així poder tractar el 100% dels aspectes tècnics i teòrics que s'han vist involucrats durant el desenvolupament d'aquest programari.

No obstant això, també aportaré una visió general de l'arquitectura seguida i dels altres components que formen la aplicació per a poder tenir una idea del funcionament conjunt.

A més, aprofundiré en desenvolupaments concrets que considero destacables i claus per al funcionament de Conectadrive.

[10] GEI: Grau d'Enginyeria Informàtica

3 Programari utilitzat

3.1 Software previ

Per a poder entendre el perquè de Conectadrive, és necessari saber quin és el context actual de l'empresa i el seu trajecte. En primer cas tenim l'empresa arrel que s'anomena Avantcab, la qual és propietària dels vehicles i s'encarrega de dur a terme els serveis. Per a poder dur aquesta gestió, l'empresa porta 3 anys fent servir un software de gestió de serveis anomenat Gestrans. Aquest software és prou complet però té bastants inconvenients que esmentaré a continuació, que han sigut part dels motius que han portat al desenvolupament d'un software a mesura, Conectadrive.

3.1.1 DAAS

Aquest software de gestió de serveis, anomenat Gestrans, funciona amb el que es coneix com a *DAAS*[11], i en aquest apartat explicaré quines son les seves característiques, avantatges i inconvenients que comporta al ser aplicat en aquest sector.

3.1.1.1 Beneficis

Un dels principals beneficis de disposar d'un DAAS és òbviament la seguretat que això aporta, l'empresa que ofereix Gestrans s'encarrega d'emmagatzemar les dades i qualsevol connexió al seu aplicatiu s'ha de dur a terme mitjançant la virtualització, en aquest cas es fa servir una aplicació d'escriptori que per sota utilitza el protocol *RDP* [12]per a poder fer una espècie d'escriptori remot. A més, per a poder accedir a Gestrans s'ha de fer servir una VPN especial que garanteix que cap usuari extern té accés a la plataforma, a part de requerir un token que s'ha de renovar a través de la *CMD*[13] cada x hores. A més, l' escalabilitat és un altre punt que es beneficia d'aquesta modalitat, ja que es fa servir un hipervisor de tipus 2, del que s'esmentaran els beneficis a continuació. L'últim benefici però no menys important és el benefici econòmic que suposa auto-emmagatzemar les dades, ja que estalvia costos de proveïdors de tercers i permet tenir un millor control de la infraestructura.

3.1.1.2 Inconvenients

En quant als inconvenients, el primer de tots és el cost econòmic que comporta, tot i que la despesa inicial sigui inferior i no depenguem de proveïdors de tercers, no tenim la possibilitat i gran avantatge que el pay-as-you-go dona, i si volem tenir una infraestructura determinada no tindrem la oportunitat de fer up-scale o under-scale en moments determinats, com a molt podrem apagar els equips per a reduir el consum. A més, aquesta modalitat de software fa les integracions amb el client més complexes, obligant-lo a integrar la *VPN*[14] necessària per a treballar i adaptar la seva arquitectura actual al sistema en qüestió. No oblidem també que aquest sistema acostuma a tenir actualitzacions individualitzades per a cada client, fet que fa molt més complex el manteniment per part de l'empresa i ralenteix el

[11] DAAS: Destkop as a Service

[12] RDP: Remote Desktop Protocol

[13] CMD: Command prompt.

[14] Virtual Private Network

procés de correcció d'errors i noves funcionalitats. Per últim, tenim el handicap de la latència, al cap i a la fi estarem connectats a un altre ordinador de forma remota i és possible que l'experiència que això ofereixi no sigui la més recomanable, més encara en casos on la connexió a internet no sigui especialment favorable.

3.1.2 Hypervisor tipus 2

3.1.2.1 Beneficis

Un dels principals avantatges de l'ús de hipervisors de tipus 2 és la seva capacitat per oferir una gran flexibilitat i compatibilitat amb diversos sistemes operatius. Aquests, s'executen com una aplicació dins d'un sistema operatiu existent, cosa que els fa ideals per a entorns de desenvolupament i proves, on es poden configurar i desplegar ràpidament múltiples màquines virtuals amb diferents configuracions de sistema operatiu sense necessitat de múltiples peces de maquinari.

- **Facilitat d'instal·lació i configuració**: Els hipervisors de tipus 2 són fàcils d'instal·lar i configurar, semblant a qualsevol altra aplicació de programari, el que redueix significativament la complexitat tècnica per als usuaris finals.
- **Ideal per a entorns de desenvolupament i formació**: Permet als desenvolupadors i estudiants executar i provar diferents sistemes operatius i aplicacions en un entorn segur i aïllat sense afectar el sistema operatiu principal.
- **Menor necessitat d'inversió en hardware**: Ja que no requereixen maquinari especialitzat o múltiples màquines físiques, els usuaris poden estalviar en costos de hardware mentre mantenen la capacitat de provar i executar diverses configuracions.
- **Protecció i seguretat**: Encara que no tan robustos com els hipervisors de tipus 1, els de tipus 2 ofereixen una bona separació entre les màquines virtuals i el sistema operatiu amfitrió, permetent una millor gestió dels riscos de seguretat en comparació amb l'execució d'aplicacions directament en el sistema operatiu principal.

3.1.2.2 Inconvenients

Malgrat els seus avantatges, els hipervisors de tipus 2 també presenten alguns inconvenients que poden limitar la seva utilitat en certs escenaris, especialment en entorns de producció on el rendiment i la seguretat són crítics.

- **Rendiment inferior**: Com que operen sobre un sistema operatiu host, els hipervisors de tipus 2 poden experimentar una degradació del rendiment en comparació amb els de tipus 1. Això es deu a l'overhead addicional de gestionar la interacció entre la màquina virtual, el sistema operatiu host i el hardware.
- **Seguretat**: La dependència d'un sistema operatiu subjacent pot ser un punt de vulnerabilitat. Si el sistema operatiu host es veu compromès, totes les màquines virtuals que s'executen en ell podrien estar en risc.
- **Menys control sobre els recursos del sistema**: A diferència dels de tipus 1, els de tipus 2 no gestionen directament el hardware, el que pot resultar en una distribució menys eficient dels recursos del sistema entre múltiples màquines virtuals.

- **Dependència del sistema operatiu host**: El funcionament i la estabilitat del hipervisor de tipus 2 depenen en gran mesura del sistema operatiu en què s'executa. Qualsevol problema amb el sistema operatiu host pot afectar totes les màquines virtuals.

3.1.3 Software legacy – hub

A més d'aquest software anomenat Gestrans que feia servir l'empresa, trobem que fan ús d'un aplicatiu/portal que els permet enviar les reserves d'agències de tercers al software de Gestrans, ja que les possibilitats que ofereix són limitades. Aquest software consisteix de forma resumida en un aplicatiu desenvolupat amb Quasar, que és un framework de Vue.js i una API rest desenvolupada amb Spring Boot, amb una base de dades PostgreSQL. L' arquitectura és relativament senzilla i està tot allotjat a un *VPS*[15], que també suposa un cost mensual per a l'empresa. A més, no és gens escalable perquè només hi ha una instància desplegada, i tot i que el tràfic no sol ser molt elevat, ens impedeix escalar en cas de ser necessari.

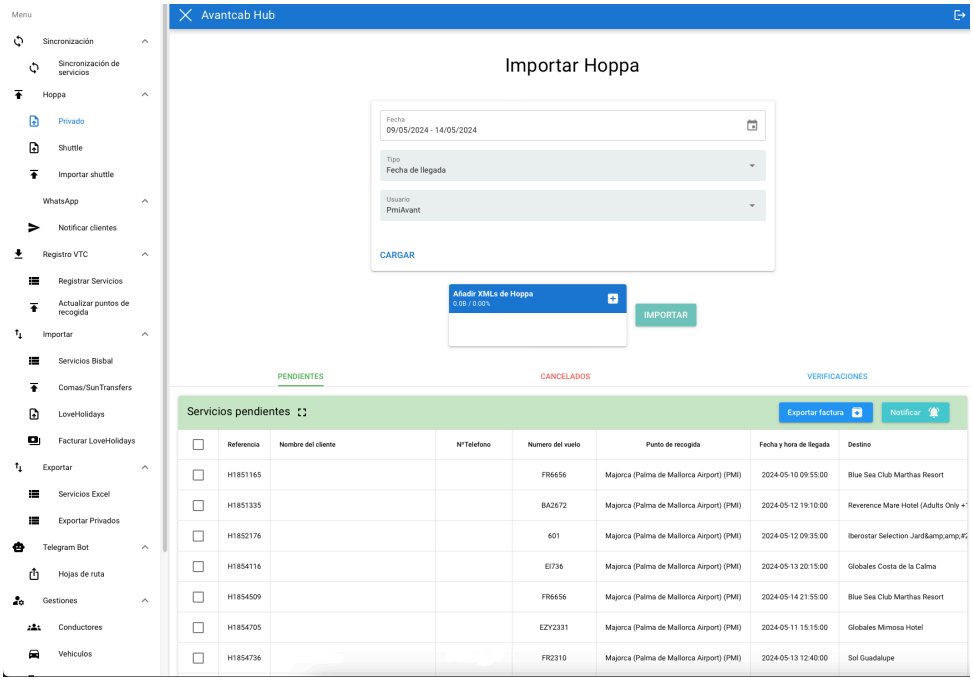

Figura 1. Pàgina web anterior

Com es pot apreciar, la interfície no està gens cuidada i tot i que hi hagi moltes pàgines al sidebar, moltes d'elles són duplicats amb lleugers canvis, fruit d'una mala programació i estructura de components. A més, la velocitat del servidor és bastant dolenta, només té una instància i al mateix servidor, que té 16GB de RAM, s'executen frontend, backend, base de dades PostgreSQL i una base de dades Redis per al cacheig d'informació.

[15] VPS: Virtual Private Server

Òbviament és una arquitectura senzilla i funcional per a empreses que no requereixen gran volum de treball, fet que abans es complia, però degut al gran creixement de l'empresa durant els últims 4 anys ha comportat valorar el desenvolupament de Conectadrive.

Un cop entesos els principals avantatges i inconvenients del *DAAS* i de l'hypervisor de tipus 2 que aquest utilitza i els altres aspectes esmentats, hem de sumar el fet que és un software considerablement car i poc personalitzable, que en molts cops ha obligat a l'empresa a buscar solucions i implementacions alternatives que cobreixin funcionalitats que gestrans no oferia. És per això que es va valorar la opció de considerar el desenvolupament propi d'un software que millori la gran majoria d'aquests aspectes. La idea bàsica seria fer una plataforma online que permeti a les empreses decidir entre auto-allotjar les seves dades o fer que conectadrive se n'encarregués i així despreocupar-se d'aquest fet, i dels altres aspectes que comporta com la ciberseguretat i escalabilitat.

3.2 Tecnologies Conectadrive

3.2.1 Backend

3.2.1.1 Java

Per al desenvolupament de la part backend del projecte ConectaDrive, hem triat utilitzar el llenguatge de programació Java, principalment per les seves capacitats en l'àmbit del desenvolupament de serveis d'aplicacions robustes, segures i altament escalables. Java és reconegut per la seva estabilitat i per oferir un rendiment consistent en ambients de producció, característiques que són essencials per a gestionar la complexitat de la gestió de reserves, empleats i vehicles en temps real que caracteritzen el nostre sistema.

Java proporciona una base sòlida per a aplicacions empresarials, com ConectaDrive, que requereixen una interacció constant amb bases de dades (en aquest cas, PostgreSQL) i un processament eficient de múltiples sol·licituds simultànies sense comprometre la seguretat.

3.2.1.2 Spring Boot

Hem optat per utilitzar Spring Boot, un framework basat en Spring. Spring Boot és reconegut per la seva capacitat de simplificar el desenvolupament d'aplicacions robustes i d'alt rendiment en Java, la qual cosa el converteix en una elecció ideal per al nostre sistema backend.

Aquest, està dissenyat per facilitar el procés de configuració i desplegament d'aplicacions Spring, proporcionant una configuració automàtica que redueix significativament la necessitat de definicions manuals.

Un dels avantatges més notables de Spring Boot és que inclou servidors web incrustats com Tomcat, eliminant la necessitat de desplegar l'aplicació en servidors d'aplicacions externs. Aquesta capacitat de "empaquetar i portar" simplifica el procés de desplegament i fa que les operacions siguin més àgils. A més, Spring Boot manega la gestió de dependències a través dels seus "starters", que agrupen les biblioteques comunes necessàries per a diverses funcionalitats, assegurant que les dependències siguin consistents i estiguin actualitzades.

En el context de ConectaDrive, on l'eficiència i la capacitat de resposta són crucials per gestionar reserves i operacions de vehicles en temps real, Spring Boot ofereix eines

robustes com Spring Data *JPA*[16] [3] per a la integració amb bases de dades i Spring Security [4] per a la gestió de l'autenticació i autorització. Aquests components ens permeten construir un sistema fiable i segur que pot escalar segons les necessitats dels usuaris i manegar eficaçment les transaccions complexes requerides pel servei.

3.2.1.3 Redis

Hem integrat Redis com a sistema de memòria cau per al nostre projecte ConectaDrive. Redis és un magatzem de dades en memòria, utilitzat com a base de dades, cau i intermediari de missatges. És conegut per la seva alta velocitat i eficiència en la lectura i escriptura de dades, cosa que el fa ideal per a aplicacions que requereixen un accés ràpid i freqüent a les dades.

L'utilitzem també per a cachejar les dades que s'accedeixen freqüentment [5], com informació de reserves, dades de vehicles i perfils d'empleats. Aquesta aproximació redueix la càrrega en la nostra base de dades principal, PostgreSQL, i millora significativament el rendiment de l'aplicació en reduir els temps de resposta. A més, Redis suporta estructures de dades complexes com strings, hash, llistes, conjunts i conjunts ordenats amb consultes que poden ser executades en temps constant. Això ens permet implementar funcionalitats avançades com la invalidació de memòria cau i la gestió de sessions de manera eficient i efectiva.

3.2.1.4 PostgreSQL

Com a sistema de gestió de base de dades hem seleccionat PostgreSQL per la seva robustesa i funcionalitats completes que són ideals per gestionar grans càrregues de treball. És particularment valuós per la seva capacitat per emmagatzemar de manera eficaç dades crítiques, que inclouen informacions detallades de reserves, dades d'usuaris i informació de vehicles.

Un dels principals avantatges és la seva extensió PostGIS [5], que permet la gestió avançada d'informació geo-espaials. A ConectaDrive, aquesta capacitat és crucial, ja que necessitem emmagatzemar i manipular geodades complexes com polígons, localitats i altres informacions geo-espaials relacionades amb les rutes i àrees de servei dels vehicles. PostGIS ofereix eines potents per a consultes espacials i anàlisis que ens permeten, per exemple, determinar quina zona geogràfica cobreix un vehicle específic o optimitzar les rutes basades en la proximitat i la disponibilitat.

Aquesta integració de Redis i PostgreSQL proporciona una arquitectura robusta per a ConectaDrive, on Redis manega accessos de dades ultra ràpids per reduir la càrrega de treballs repetitius, mentre que PostgreSQL, gestiona de manera segura i eficient les dades espacials i de persistència a llarg termini, garantint que les operacions de dades geo-espaials i les transaccions són gestionades amb alta precisió i eficàcia.

3.2.2 Frontend

3.2.2.1 Primevue

En el desenvolupament del frontend de ConectaDrive, hem optat per utilitzar PrimeVue. Aquesta elecció es basa en la riquesa dels seus components de UI que s'integren

[16] JPA: Java Persistence API

fàcilment amb els projectes Vue.js, oferint una àmplia gamma de widgets interactius i altament personalitzables, essencials per a una aplicació intensiva en dades com la nostra. PrimeVue no només destaca per la seva versatilitat en components com taules de dades, diàlegs modals i formularis, sinó també per la seva excel·lent documentació que facilita l'aprenentatge ràpid i l'adopció per part dels desenvolupadors.

3.2.2.2 TypeScript

Per augmentar la productivitat i millorar el manteniment del codi, hem adoptat TypeScript juntament amb l'ús de l'opció Script Setup en Vue 3, que simplifica la sintaxi dels components composable utilitzant la composition API. A més, aporta una capa addicional de seguretat al desenvolupament, permetent-nos capturar errors en temps de compilació abans que esdevinguin problemes en producció. Aquest llenguatge ofereix característiques avançades com la inferència de tipus, tipus genèrics i interfícies, millorant així la documentació del codi i facilitant la refactorització i l'escabilitat del codi.

3.2.2.3 ESLint

Per mantenir un alt estàndard en la qualitat del codi, utilitzem ESLint com a eina de linter. Aquest, ens ajuda a seguir unes normes de codi consistents i a evitar errors comuns en JavaScript i TypeScript. A través de la seva configuració personalitzable, podem definir regles específiques que s'adeqüen als nostres estàndards de codi, així com integrar millors pràctiques i estils de codi recomanats per la comunitat. L'ús d'ESLint assegura que tot el codi s'adhereixi a un format coherent, facilitant la col·laboració entre desenvolupadors i millorant la llegibilitat i mantenibilitat del codi.

3.2.2.4 Vite

Hem triat Vite com a eina de construcció per al nostre projecte frontend. Vite és una nova generació de constructor de front-end que ofereix una inicialització ràpida i un hot-reload extremadament ràpid, característiques que són possibles gràcies a l'ús eficaç de l'esborrat de JavaScript modern i el tractament dels mòduls ES. Això facilita un cicle de desenvolupament més ràpid i eficient, permetent als nostres desenvolupadors fer iteracions ràpides sense perdre temps en esperes de reconstrucció. Vite configura també de manera òptima el projecte per a la producció, assegurant-se que l'aplicació estigui optimitzada i llesta per a un rendiment elevat.

3.2.2.5 Tailwind CSS

Per al disseny de l'interfície d'usuari, hem incorporat Tailwind CSS [6], un framework CSS que utilitza classes d'utilitat per a un disseny ràpid i responsive. Tailwind simplifica el procés d'estilització sense necessitat d'escriure CSS personalitzat extensiu. Aquest enfocament "utility-first" permet als desenvolupadors construir dissenys complexos amb eficiència, ja que poden aplicar estils de manera modular directament en els elements del HTML. Tailwind és altament personalitzable, cosa que ens permet definir temes i estils que s'adeqüen a la imatge de marca de ConectaDrive, mentre mantenim una base de codi neta i fàcil de mantenir.

3.2.3 Repositori de codi – AWS CodeCommit

Hem escollit AWS CodeCommit per al control de versions del projecte ConectaDrive, principalment per la seva òptima integració amb els altres serveis d'Amazon Web Services que ja estem utilitzant.

3.2.4 Entorns de desenvolupament

3.2.4.1 IntelliJ IDEA Ultimate

En el desenvolupament del backend de ConectaDrive, hem optat per utilitzar IntelliJ IDEA Ultimate [7] com a entorn de desenvolupament integrat (*IDE*[17]). Aquesta elecció es basa en la robusta compatibilitat de IntelliJ amb Java i Spring Boot, que són fonamentals per al nostre projecte. Destaca, principalment, per la seva capacitat per simplificar la configuració de projectes, la gestió de dependències i proporcionar navegació automàtica cap a la configuració i els beans de Spring, facilitant considerablement el desenvolupament.

Un dels punts forts és la seva integració amb diferents sistemes de bases de dades. Aquesta funcionalitat ens permet als desenvolupadors connectar-nos, executar consultes i gestionar esquemes directament des de l'IDE, optimitzant els fluxos de treball amb JPA i altres frameworks *ORM*[18]. Les eines de depuració i anàlisi de codi d'IntelliJ són particularment potents, oferint depuració a nivell de línia i inspeccions de codi en temps real que ajuden a mantenir la qualitat del codi.

3.2.4.2 Visual Studio Code

Per al desenvolupament del frontend de ConectaDrive, hem triat Visual Studio Code (VS Code) [8] com a IDE. És conegut per la seva lleugeresa i eficiència, aspectes que juguen un paper crucial en el desenvolupament ràpid i eficient, especialment en entorns de frontend.

A més, suporta una àmplia gamma de llenguatges de programació i marcs, inclòs JavaScript, TypeScript, *CSS*[19], *HTML*[20] i frameworks moderns com Vue.js, que utilitzem a ConectaDrive. Aquesta compatibilitat, combinada amb la seva capacitat per integrar-se amb una gran varietat de plugins i extensions fa que VS Code sigui extremadament adaptable a les necessitats específiques del projecte.

Una de les característiques més valuoses de VS Code és el seu sistema d'extensions, que permet als desenvolupadors personalitzar l'entorn per adaptar-se millor a les seves necessitats. Extensions com ESLint per a l'anàlisi de codi JavaScript, i Prettier per a la formatació automàtica de codi, milloren significativament la productivitat i asseguren que el codi sigui net i conforme als estàndards del projecte.

3.2.4.3 PyCharm

Per al desenvolupament de scripts específics destinats a la manipulació i gestió de dades complexes a la nostra base de dades, hem optat per utilitzar PyCharm, que és un IDE específic per a Python, desenvolupat per JetBrains.

Dins el context de ConectaDrive, l'hem utilitzat per a desenvolupar i executar scripts d'inserció de dades que són fonamentals per al negoci. Per exemple, hem creat scripts per

[17] IDE: Integrated development environment

[18] ORM: Object relational mapping

[19] CSS: Cascading Style Sheets)

[20] HTML: HyperText Markup Language

afegir automàticament a la nostra base de dades els preus provinents d'agències com booking.com.

4 Requisits

Els requisits funcionals i no funcionals són dos tipus d'especificacions que s'utilitzen per descriure les característiques i criteris que un sistema, aplicació o producte ha de complir. Aquests requisits són essencials en el procés de desenvolupament de programari i sistemes, ja que guien als desenvolupadors, dissenyadors i tots els involucrats en la creació i avaluació del projecte.

4.1 Requisits funcionals

4.1.1 Gestió d'Usuaris

- RF1: Iniciar sessió. Els usuaris podran iniciar sessió introduint les seves credencials vàlides.
- RF2: Registrar-se. Els nous usuaris podran crear un compte omplint un formulari de registre i rebent una confirmació per correu electrònic.
- RF3: Sol·licitar enllaç de restabliment de contrasenya. Els usuaris podran sol·licitar un enllaç per restablir la seva contrasenya que serà enviat al seu correu electrònic.
- RF4: Restablir contrasenya. Els usuaris podran restablir la seva contrasenya seguint l'enllaç proporcionat.

4.1.2 Visualització i Gestió de Dades de Vehicles

- RF5: Visualitzar la ubicació dels vehicles en temps real. El sistema mostrarà un mapa centrat en les ubicacions actuals dels vehicles i permetrà visualitzar informació detallada de cada vehicle.
- RF6: Importar fitxers de proveïdors externs. El sistema permetrà importar dades de fitxers proporcionats per proveïdors externs per actualitzar o afegir informació al sistema.

4.1.3 Gestió de Reserves i Vehicles

- RF7: Visualitzar reserves. Els usuaris podran consultar totes les reserves amb la seva informació completa.
- RF8: Crear i modificar vehicles, reserves, conductors, empreses i tipus de servei. El sistema permetrà la creació i modificació d'aquests elements a través de formularis específics.
- RF9: Modificar usuaris. Usuaris amb aquest rol podran modificar qualsevol usuari en el sistema.
- RF10: Visualitzar i afegir localitats, municipis, garatges, punts de recollida i zones de treball. Els usuaris podran gestionar aquesta informació geogràfica i també dibuixar polígons per a les localitats en un mapa.

4.1.4 Integració i Automatització

- RF11: Emparellar punts de codis externs. El sistema permetrà emparellar codis de punts de recollida externs amb els corresponents interns.
- RF12: Enviar reserves a Fomento i Gestrans. El sistema inclourà un planificador en el backend per enviar reserves automàticament a aquests organismes segons necessitat.

- RF13: Processar reserves de correus electrònics. El sistema podrà processar automàticament reserves rebudes per correu electrònic de companyies com HolidayTaxi o SunTransfer.
- RF14: Ofereix preus a agències com Booking. Implementació de webhooks per oferir preus i disponibilitats en temps real a agències online.

4.2 Requisits no funcionals

4.2.1 Rendiment i Escalabilitat

- RNF1: Temps de resposta. El sistema ha de respondre a qualsevol acció de l'usuari en menys de 3 segons.
- RNF2: Suport de càrrega. El sistema ha de poder gestionar fins a 10,000 usuaris simultanis sense degradació significativa del rendiment.

4.2.2 Seguretat

- RNF3: Encriptació de dades. Totes les transaccions financeres i les dades personals han d'estar encriptades utilitzant estàndards com AES o SSL.
- RNF4: Autenticació segura. El sistema utilitzarà OAuth2 [9] i tokens basats en JWT per a la gestió de la sessió i autenticació.
- RNF5: Protecció contra atacs. El sistema ha d'implementar mesures per protegir-se contra atacs com SQL injection, *XSS*[21] i *CSRF*[22].

4.2.3 Usabilitat

- RNF6: Compatibilitat de navegador. El sistema ha de ser compatible amb les últimes versions de Chrome, Firefox, Safari i Edge.

4.2.4 Mantenibilitat

- RNF7: Logging i Monitorització. El sistema ha de registrar totes les accions crítiques i errors en un sistema de logging centralitzat per facilitar el seguiment i la resolució de problemes.
- RNF8: Actualitzacions del sistema. El sistema ha de suportar actualitzacions sense interrupció del servei, utilitzant estratègies com el desplegament blau-verd o canary releases.

4.2.5 Interoperabilitat

- RNF9: Integració amb sistemes externs. El sistema ha de proporcionar API RESTful per permetre la integració amb altres sistemes de tercers.
- RNF10: Exportació i importació de dades. El sistema ha de permetre la exportació i importació de dades en formats estàndard com CSV i JSON.

[21] XSS: Cross-site scripting

[22] CSRF: Cross-site request forgery

4.3 Casos d'ús

Un cas d'ús és una descripció detallada d'un procés específic que un usuari realitza amb un sistema per assolir un objectiu concret. Representa una interacció completada entre l'usuari i el sistema, que pot ser tan simple com introduir dades o tan complexa com processar una transacció. Els casos d'ús són fonamentals en el desenvolupament de programari per a definir clarament les funcionalitats que el sistema ha de proporcionar des del punt de vista dels usuaris finals.

En el context del desenvolupament d'un sistema per a la gestió de reserves, vehicles i conductors en una empresa com és Conectadrive, els casos d'ús permeten capturar i detallar les funcionalitats específiques que el sistema ha de proporcionar. De forma breu, un cas d'ús descriu una seqüència d'interaccions entre l'usuari i el sistema que resulta en un objectiu visible per a l'usuari, com ara gestionar una reserva o assignar un vehicle a un conductor.

Els casos d'ús són necessaris per a diversos aspectes del desenvolupament del projecte:

- Clarificació de funcionalitats: Ajuden a comprendre millor quines accions ha de poder realitzar el sistema i com aquestes accions serveixen les necessitats operacionals de l'empresa.
- Priorització del desenvolupament: Ens permeten organitzar les funcionalitats per importància i necessitat, assegurant que els recursos de desenvolupament es dediquen primer a les àrees més crítiques.
- Facilitació de les proves: Cada cas d'ús proporciona un marc clar per a les proves del sistema, assegurant que totes les funcions es comporten com s'espera abans de ser posades en producció.
- Millora de la comunicació: Els casos d'ús ofereixen una forma estandarditzada i entenedora per a tots els membres de l'equip de desenvolupament i els stakeholders de discutir les necessitats del sistema, millorant la comprensió general i reduint la possibilitat de malentesos.

Veure diagrama de classes a la Figura 2.

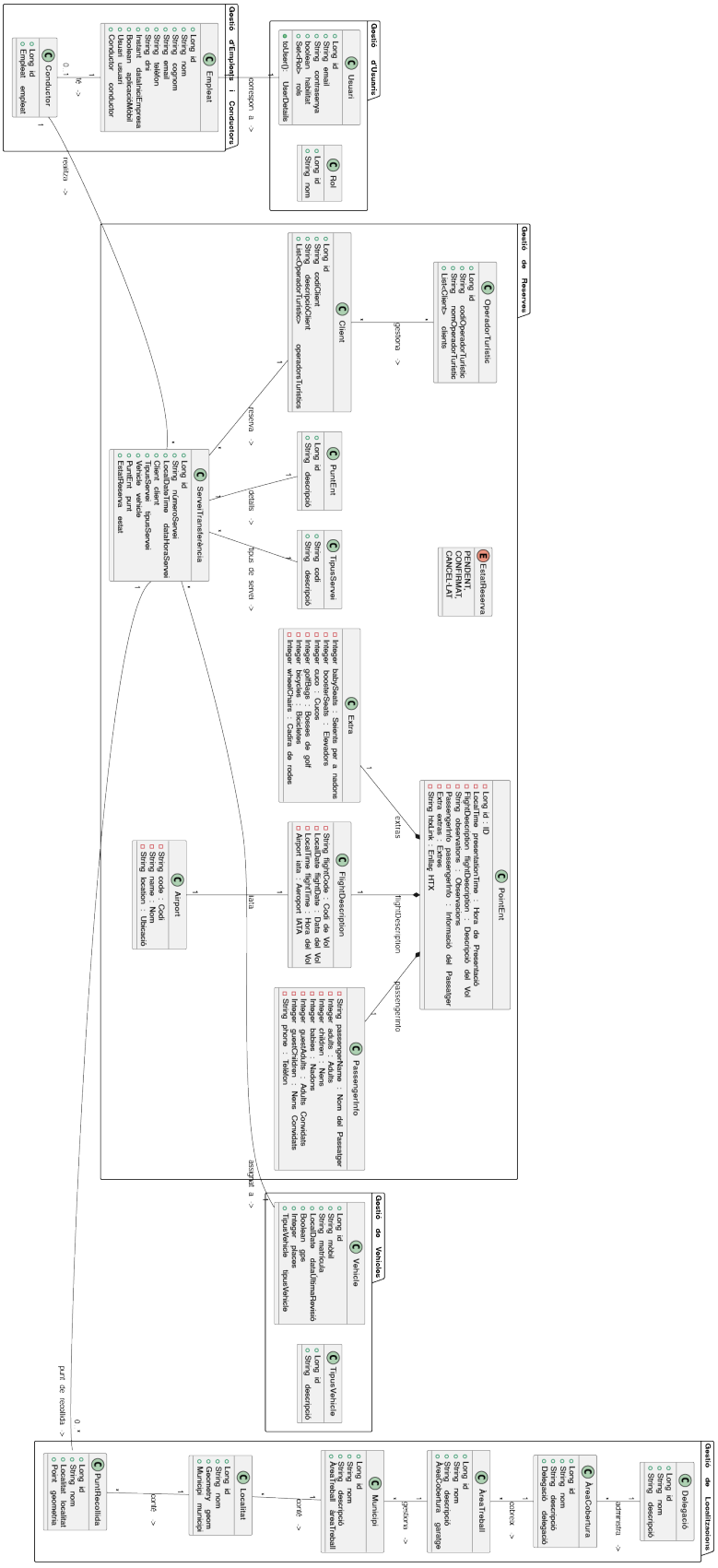

Figura 2 - Diagrama de classes

4.3.1 01. Iniciar sessió

Resum de la funcionalitat: Permetre als empleats de l'empresa existents accedir al sistema introduint les seves credencials vàlides.

Paràmetres d'entrada:

- Correu electrònic: L'adreça de correu electrònic registrada de l'empleat.
- Contrasenya: La contrasenya associada al compte de l'empleat.

Paràmetres de sortida: Cap, però en cas d'un inici de sessió reeixit, l'empleat serà redirigit a la seva pàgina d'inici.

Actors:

- Empleat (poden ser administradors, gestors de flota, etc., que treballen per a l'empresa).

Precondició:

- L'empleat ha de tenir un compte ja creat i confirmat en el sistema.
- L'usuari intentant accedir ha de ser un empleat de l'empresa.

Postcondició:

- Si les credencials són correctes i pertanyen a un empleat de l'empresa, l'usuari accedeix al seu compte i es redirigeix a la pàgina d'inici.
- Si les credencials no són correctes o no pertanyen a un empleat de l'empresa, l'usuari no podrà accedir i se li mostrarà un missatge d'error.

Procés normal principal:

1. L'empleat obre la pàgina de login del sistema.
2. El sistema presenta un formulari amb camps per al correu electrònic i la contrasenya.
3. L'empleat introdueix el seu correu electrònic i contrasenya en els camps proporcionats.
4. L'empleat fa clic al botó de "Iniciar sessió".
5. El sistema valida les credencials contra la base de dades i verifica que l'usuari sigui un empleat de l'empresa.
6. Si les credencials són correctes i l'usuari és un empleat confirmat, el sistema crea una sessió per a l'usuari i el redirigeix a la pàgina d'inici.
7. Si les credencials no són correctes o l'usuari no és un empleat de l'empresa, el sistema mostra un missatge d'error i permet a l'usuari re introduir les dades.

Alternatives de procés i excepcions:

3.a. Correu electrònic o contrasenya buits:

3a1. El sistema mostra un missatge que indica que els camps no poden estar buits.

6a. Credencials incorrectes o usuari no és un empleat:

6a1. El sistema mostra un missatge d'error que indica que el correu electrònic o la contrasenya són incorrectes o que l'usuari no és un empleat autoritzat.

6a2. L'usuari té l'opció de tornar a introduir les credencials.

Diagrama:

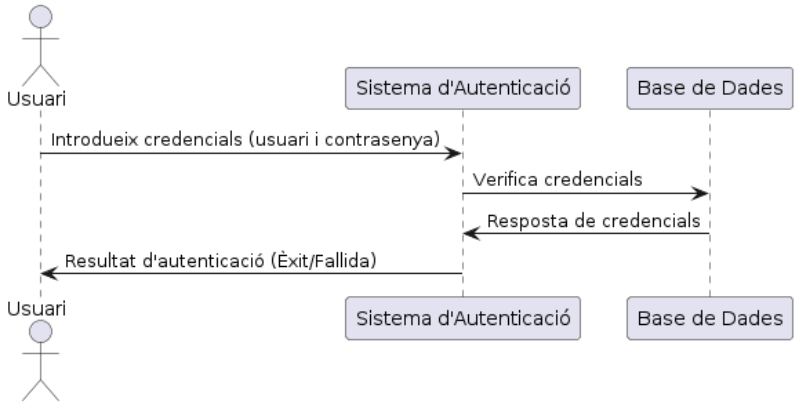

Figura 3 – Diagrama de seqüència CU 01

4.3.2 02. Registrar-se

Resum de la funcionalitat: Permetre als nous empleats de l'empresa crear un compte al sistema, omplint un formulari de registre i rebent una confirmació per correu electrònic.

Paràmetres d'entrada:

- Nom: Nom complet de l'empleat.
- Correu electrònic: Adreça de correu electrònic que l'empleat utilitzarà per al compte.
- Contrasenya: Contrasenya escollida per l'empleat.
- Confirmació de contrasenya: Repetició de la contrasenya per a verificació.

Paràmetres de sortida: Cap directament, però el sistema enviarà un correu electrònic de confirmació al nou usuari.

Actors:

- Empleat (nou usuari que es registra al sistema).

Precondició:

- L'empleat no deu tenir un compte existent en el sistema.

Postcondició:

- Un nou compte d'usuari és creat, i un correu electrònic de confirmació és enviat a l'adreça proporcionada.

Procés normal principal:

1. L'empleat accedeix a la pàgina de registre del sistema.
2. El sistema presenta un formulari amb els camps necessaris: nom, correu electrònic, contrasenya i confirmació de contrasenya.
3. L'empleat omple el formulari amb les seves dades i les introdueix al sistema.
4. L'empleat envia el formulari.
5. El sistema valida les dades introduïdes, assegurant-se que la contrasenya i la confirmació coincideixen i que el correu electrònic no està ja en ús.
6. Si la validació és correcta, el sistema crea el nou compte.

7. El sistema envia un correu electrònic de confirmació a l'adreça proporcionada amb un enllaç per activar el compte.
8. L'empleat clica l'enllaç rebut al seu correu per activar el compte.
9. El sistema activa el compte i mostra un missatge de benvinguda i confirmació de l'activació.

Alternatives de procés i excepcions:

5a. La contrasenya i la confirmació no coincideixen:

5a1. El sistema mostra un missatge d'error i sol·licita que les contrasenyes siguin re introduïdes.

5b. El correu electrònic ja està en ús:

5b1. El sistema informa a l'empleat que el correu electrònic ja està registrat i ofereix opcions per recuperar la contrasenya o iniciar sessió.

8a. El correu de confirmació no és clicat en un termini estipulat:

8a1. El compte roman inactiu fins que l'enllaç de confirmació sigui utilitzat, o el sistema pot requerir una nova sol·licitud de registre 15 minuts després d'haver emès el primer enllaç.

Diagrama:

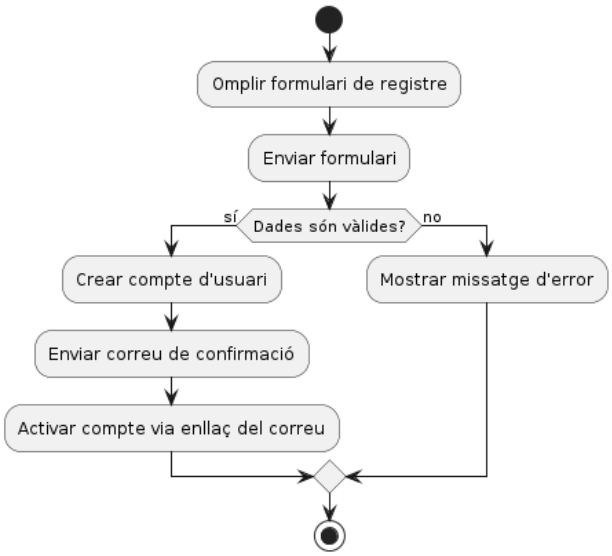

Figura 4 - Diagrama d'activitat CU 02

4.3.3 03. Sol·licitar enllaç de restabliment de contrasenya

Resum de la funcionalitat: Permet als empleats sol·licitar un enllaç per restablir la seva contrasenya, el qual serà enviat a la seva adreça de correu electrònic registrada si el compte existeix.

Paràmetres d'entrada:

- Correu electrònic: L'adreça de correu electrònic associada amb el compte de l'usuari.

Paràmetres de sortida: Cap directament, però el sistema informa a l'usuari que si el compte existeix, un correu electrònic amb l'enllaç de restabliment serà enviat.

Actors:

- Empleat (usuari que sol·licita el restabliment de la contrasenya).

Precondició:

- L'usuari ha d'accedir a la pàgina de restabliment de contrasenya.

Postcondició:

- Si l'adreça de correu electrònic correspon a un compte existent, un correu electrònic amb l'enllaç de restabliment és enviat. En tots els casos, es mostra el mateix missatge per protegir la privacitat dels usuaris.

Procés normal principal:

1. L'usuari accedeix a la pàgina de "Oblidat la contrasenya?" en el sistema.
2. El sistema presenta un formulari on l'usuari ha d'introduir la seva adreça de correu electrònic.
3. L'usuari introdueix el seu correu electrònic i fa clic al botó de "Sol·licitar restabliment".
4. El sistema processa la sol·licitud i, independentment de si l'adreça està associada amb un compte o no, mostra un missatge que diu: "Si el compte existeix amb aquest correu electrònic, rebràs un enllaç per restablir la teva contrasenya."
5. Si l'adreça està associada amb un compte existent, el sistema genera un enllaç de restabliment i l'envia al correu electrònic de l'usuari.

Alternatives de procés i excepcions:

5a. Error en l'enviament del correu electrònic:

5a1. Si sorgeix un problema tècnic que impedeix l'enviament del correu electrònic, el sistema intenta de nou automàticament. Si el problema persisteix, es registra internament per a revisió del suport tècnic.

Diagrama:

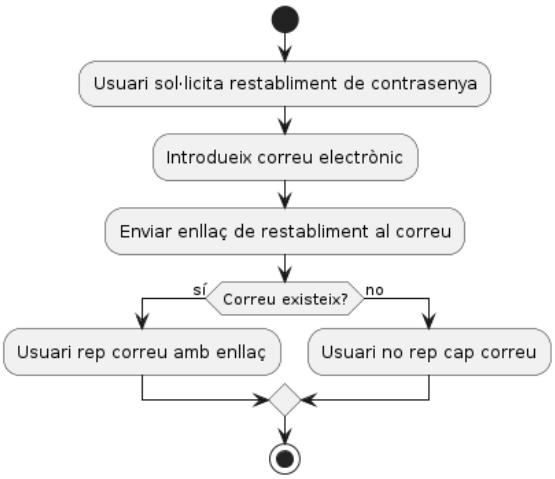

Figura 5 - Diagrama d'activitat CU 03

4.3.4 04. Restablir la contrasenya

Resum de la funcionalitat: Permet als usuaris restablir la seva contrasenya fent ús d'un enllaç que han rebut per correu electrònic, el qual conté un token de seguretat que expira en 15 minuts des de la seva emissió.

Paràmetres d'entrada:

- Enllaç de restabliment: Enllaç únic que conté un token de seguretat, enviat al correu electrònic de l'usuari.
- Nova contrasenya: La nova contrasenya que l'usuari vol establir.
- Confirmació de la nova contrasenya: Repetició de la nova contrasenya per a verificació.
- Paràmetres de sortida: Cap directament, però el sistema actualitzarà la contrasenya de l'usuari si el token és vàlid i no ha expirat.

Actors:

- Usuari (empleat que necessita restablir la seva contrasenya).

Precondició:

- L'usuari ha rebut un enllaç de restabliment per correu electrònic després de sol·licitar el restabliment de la contrasenya.
- L'usuari accedeix a l'enllaç dins de 15 minuts després de la seva emissió.

Postcondició:

- Si el token és vàlid i no ha expirat, la contrasenya de l'usuari és actualitzada amb èxit, i l'usuari pot iniciar sessió amb la nova contrasenya.
- Si el token ha expirat, l'usuari ha de sol·licitar un nou enllaç.

Procés normal principal:

1. L'usuari clica l'enllaç de restabliment que ha rebut al seu correu electrònic.
2. El sistema verifica que l'enllaç sigui vàlid i que el token no hagi expirat (menys de 15 minuts des de la seva emissió).
3. Si l'enllaç i el token són vàlids, el sistema redirigeix l'usuari a una pàgina segura on pot introduir la nova contrasenya.
4. L'usuari introdueix la nova contrasenya i la confirma en els camps proporcionats.
5. L'usuari fa clic al botó de "Restablir contrasenya".
6. El sistema valida que les dues contrasenyes introduïdes coincideixin i compleixin amb els criteris de seguretat establerts.
7. Si tot és correcte, el sistema actualitza la contrasenya de l'usuari amb la nova contrasenya.
8. El sistema mostra un missatge de confirmació que la contrasenya ha estat restablerta amb èxit i redirigeix l'usuari a la pàgina d'inici de sessió.

Alternatives de procés i excepcions:

2a. L'enllaç ha expirat o el token no és vàlid:

2a1. El sistema informa a l'usuari que l'enllaç no és vàlid o ha expirat i proporciona una opció per sol·licitar un nou enllaç de restabliment.

6a. Les contrasenyes no coincideixen o no compleixen els criteris:

6a1. El sistema mostra un missatge d'error i sol·licita que l'usuari reintrodueixi les contrasenyes.

Diagrama:

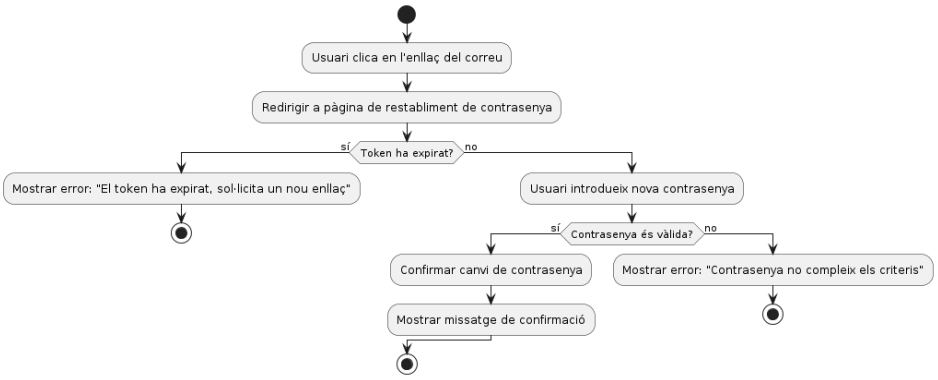

Figura 6 - Diagrama d'activitat CU 04

4.3.5 05 Visualitzar la ubicació dels vehicles en temps real

Resum de la funcionalitat: Permet als usuaris visualitzar la ubicació actual dels vehicles de l'empresa en un mapa interactiu, proporcionant informació detallada sobre cada vehicle.

Paràmetres d'entrada:

- Cap específic requerit per l'usuari, però el sistema necessita accés a les dades de localització dels vehicles en temps real.

Paràmetres de sortida:

- Mapa interactiu mostrant la ubicació actual de tots els vehicles actius.
- Informació detallada sobre cada vehicle quan es selecciona en el mapa.

Actors:

- Usuaris autoritzats (empleats de l'empresa, com gestors de flota, coordinadors de tràfic, etc.).

Precondició:

- Els usuaris han d'estar autenticats i autoritzats per accedir a aquesta funcionalitat.
- Els vehicles han d'estar equipats amb dispositius de seguiment GPS actius.

Postcondició:

- L'usuari pot veure la ubicació en temps real dels vehicles i accedir a informació detallada sobre cada un.

Procés normal principal:

1. L'usuari inicia sessió al sistema i accedeix a la funció de "Ubicació dels Vehicles".
2. El sistema consulta les dades del GPS dels vehicles a través del backend.
3. El sistema processa les dades i actualitza el mapa en temps real amb la ubicació actual dels vehicles.

4. El mapa es mostra a l'usuari amb marcadors indicant la ubicació de cada vehicle.
5. L'usuari pot clicar en qualsevol vehicle al mapa per obtenir informació detallada, com ara model del vehicle, estat, i l'última activitat.
6. L'usuari visualitza les dades desitjades i pot continuar monitoritzant o realitzar altres tasques dins del sistema.

Alternatives de procés i excepcions:

3a. Dades de GPS no disponibles temporalment:

3a1. El sistema mostra un missatge d'error indicant que la ubicació dels vehicles no està disponible.

3a2. L'usuari pot intentar actualitzar la pàgina o consultar més tard.

5a. Error en la càrrega de detalls del vehicle:

5a1. Si sorgeix un error en recuperar les dades detallades d'un vehicle, el sistema mostra un missatge d'error.

5a2. L'usuari pot provar de nou o reportar el problema al suport tècnic.

Diagrama:

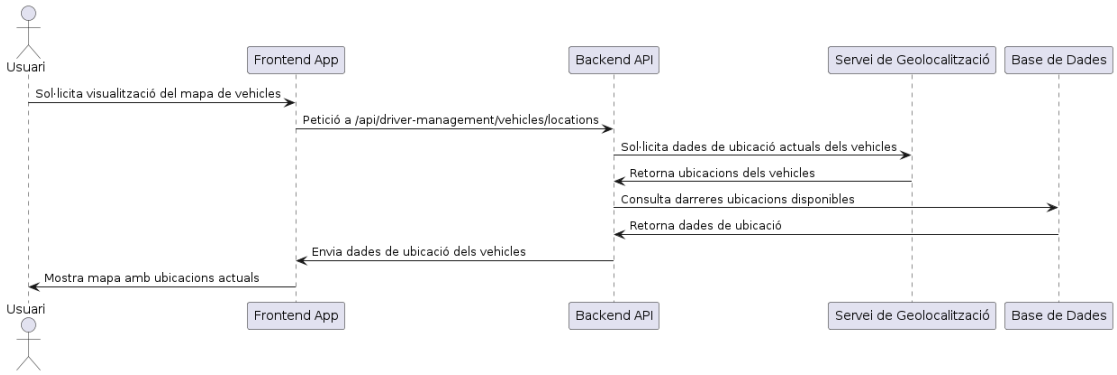

Figura 7 - Diagrama de seqüència CU 05

4.3.6 06 Importar fitxers de proveïdors externs

Resum de la funcionalitat: Permet als usuaris autoritzats importar dades des de fitxers proporcionats per proveïdors externs, amb l'objectiu d'actualitzar o afegir informació rellevant al sistema.

Paràmetres d'entrada:

- Fitxer de dades: Un fitxer (típicament en formats com CSV, XML o JSON) que conté les dades a importar.

Paràmetres de sortida:

- Cap directament, però el sistema actualitzarà la base de dades amb la informació importada.

Actors:

- Usuaris autoritzats (com gestors de dades o administradors).

Precondició:

- L'usuari ha d'estar autenticat i autoritzat per realitzar importacions de dades.
- El fitxer de dades ha de complir amb els formats acceptats pel sistema.

Postcondició:

- Les dades del fitxer són processades i integrades en la base de dades del sistema, actualitzant o afegint informació segons correspongui.

Procés normal principal:

1. L'usuari accedeix a la funció d'importació des de la interfície del sistema.
2. El sistema presenta una interfície per a seleccionar i carregar un fitxer de dades.
3. L'usuari selecciona el fitxer des del seu dispositiu i l'inicia la càrrega.
4. El sistema valida el format del fitxer i comprova que és compatible.
5. Si el format és correcte, el sistema processa les dades del fitxer.
6. El sistema importa les dades al sistema, realitzant les actualitzacions o afegint la informació necessària a la base de dades.
7. El sistema mostra un missatge de confirmació que les dades han estat importades correctament.
8. Si hi ha errors en les dades (p. ex., formats incorrectes dins del fitxer, dades que no compleixen les normatives de validació), el sistema reporta els errors a l'usuari per a la seva correcció.

Alternatives de procés i excepcions:

4a. Format de fitxer incorrecte:

4a1. El sistema alerta a l'usuari que el format del fitxer no és suportat.

4a2. L'usuari pot seleccionar un altre fitxer o convertir el fitxer actual al format adequat.

8a. Errors de validació de dades:

8a1. El sistema informa de quins errors s'han trobat durant el processament (p. ex., dades faltants, incompatibles o duplicades).

8a2. L'usuari pot corregir les dades al fitxer i tornar-lo a carregar, o descartar el procés d'importació.

Diagrama:

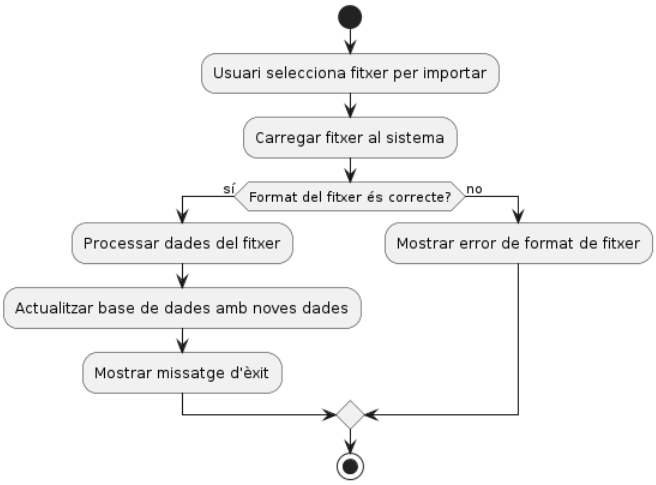

Figura 8 - Diagrama d'activitat CU 06

4.3.7 07 Visualitzar reserves

Resum de la funcionalitat: Permet als usuaris consultar i gestionar totes les reserves al sistema, utilitzant filtres avançats per columnes, accedint a detalls específics de cada reserva, i realitzant edicions quan necessari.

Paràmetres d'entrada:

- Filtres de columnes: Usuaris poden aplicar filtres a una o més columnes per a buscar en la base de dades.
- Ordenació: Usuaris poden ordenar les reserves per una columna específica.
- Referència de la reserva: Usuaris poden clicar en una referència específica de reserva per accedir a més detalls.

Paràmetres de sortida:

- Informació detallada de la reserva: Pantalla que mostra detalls complets de la reserva amb opcions per editar, veure historial de modificacions i guardar canvis.

Actors:

- Usuaris autoritzats (com administradors o gestors de reserves).

Precondició:

- L'usuari ha d'estar autenticat i autoritzat per accedir a la gestió de reserves.

Postcondició:

- Les modificacions realitzades sobre les reserves són guardades al sistema, amb un registre actualitzat accessible per futures consultes.

Procés normal principal:

1. L'usuari accedeix al mòdul de gestió de reserves en el sistema.
2. El sistema presenta una llista de reserves amb diverses columnes que inclouen informació clau.
3. Cada columna disposa d'opcions de filtre que l'usuari pot utilitzar per refinar la visualització de les reserves. Els usuaris poden aplicar filtres a múltiples columnes simultàniament.

4. El sistema permet a l'usuari seleccionar una columna per a ordenar les reserves.
5. L'usuari configura els filtres i selecciona l'ordenació desitjada.
6. El sistema consulta la base de dades amb els criteris especificats i actualitza la llista de reserves.
7. L'usuari fa clic sobre la referència de la reserva que desitja explorar més a fons.
8. El sistema obre una nova pantalla amb la informació detallada de la reserva seleccionada.
9. Dins d'aquesta pantalla, l'usuari pot:
    a. Editar camps de la reserva.
    b. Veure l'historial complet de modificacions de la reserva.
    c. Guardar qualsevol canvi realitzat.
10. L'usuari realitza les modificacions necessàries i guarda els canvis.
11. El sistema valida les dades introduïdes i actualitza la base de dades amb la informació modificada.
12. El sistema mostra un missatge de confirmació que els canvis han estat guardats correctament.

Alternatives de procés i excepcions:

3a. Filtres aplicats no retornen resultats:

3a1. El sistema mostra un missatge indicant que no hi ha resultats per als criteris especificats.

3a2. L'usuari pot ajustar els filtres i provar de nou.

11a. Errors de validació en les dades introduïdes:

11a1. El sistema informa de quins errors s'han trobat durant la validació (p. ex., formats de dades incorrectes).

11a2. L'usuari pot corregir les dades i intentar guardar de nou.

Diagrama:

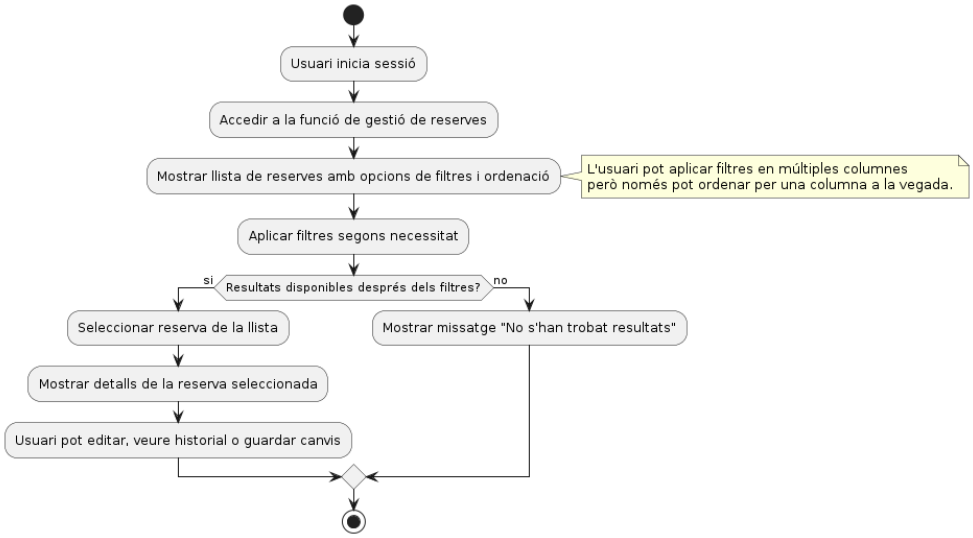

Figura 9 - Diagrama d'activitat CU 07

4.3.8 08 Crear, modificar vehicles, conductores, empreses...

Resum de la funcionalitat: Permet als usuaris autoritzats crear i modificar informació detallada sobre vehicles, conductors, empreses i tipus de servei utilitzant formularis específics dissenyats per cada tipus d'element.

Paràmetres d'entrada:

- Dades específiques de cada element: Com ara matrícula del vehicle, nom del conductor, dades de l'empresa, i detalls del tipus de servei.

Paràmetres de sortida:

- Confirmació de que les dades han estat creades o modificades correctament en la base de dades.

Actors:

- Administradors o usuaris amb permisos específics de gestió. (rol ALL_ALL_ADMIN).

Precondició:

- L'usuari ha d'estar autenticat i tenir els permisos necessaris per accedir a les funcions de creació i edició

Postcondició:

- Els elements creats o modificats són guardats en la base de dades amb les dades actualitzades proporcionades.

Procés normal principal:

1. L'usuari selecciona l'opció de crear o modificar dins del mòdul corresponent (vehicles, conductors, empreses, tipus de servei).
2. El sistema presenta un formulari específic basat en l'elecció de l'usuari (p. ex., un formulari per a vehicles inclourà camps com matrícula, model, capacitat, etc.).
3. L'usuari omple o modifica els camps necessaris en el formulari.
4. Una vegada completat, l'usuari envia el formulari.
5. El sistema valida les dades introduïdes per assegurar que compleixen amb tots els requisits i normatives.
6. Si les dades són vàlides, el sistema actualitza la base de dades amb la nova informació.
7. El sistema mostra un missatge de confirmació a l'usuari que les dades han estat guardades correctament.

Alternatives de procés i excepcions:

5a. Dades introduïdes no vàlides o incompletes:

5a1. El sistema alerta a l'usuari sobre els errors o camps obligatoris que falten.

5a2. L'usuari corregeix les dades i reenvia el formulari.

6a. Error en guardar les dades a la base de dades:

6a1. El sistema mostra un missatge d'error indicant que no s'ha pogut completar l'operació.

6a2. L'usuari pot intentar de nou o contactar amb suport tècnic.

Diagrama:

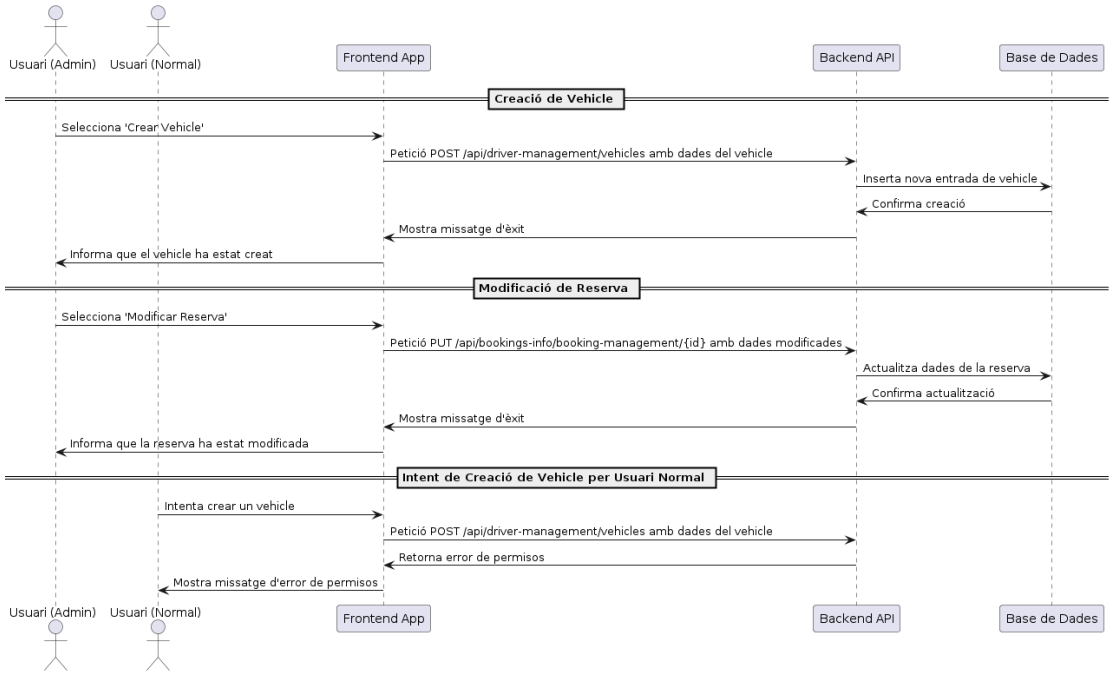

Figura 10 - Diagrama de seqüència CU 08

4.3.9 09 Modificar usuaris

Resum de la funcionalitat: Permet als usuaris amb rol específic de gestió (típicament administradors) modificar les dades dels usuaris registrats en el sistema.

Paràmetres d'entrada:

- Identificador de l'usuari: El ID o detalls específics de l'usuari que serà modificat.
- Dades de l'usuari a modificar: Inclou nom, correu electrònic, rol, estat de l'usuari (actiu/inactiu), entre altres dades personals o de configuració.

Paràmetres de sortida:

- Confirmació de les modificacions: Notificacions que confirmen que les dades de l'usuari s'han actualitzat correctament.

Actors:

- Administradors o usuaris amb permisos elevats de modificació d'usuaris.

Precondició:

- L'usuari modificador ha d'estar autenticat i ha de posseir els permisos necessaris per modificar altres usuaris.
- L'usuari a ser modificat ha d'estar registrat en el sistema.

Postcondició:

- Les dades de l'usuari seleccionat estan modificades en la base de dades segons les especificacions del modificador.

Procés normal principal:

1. L'usuari administrador accedeix al mòdul de gestió d'usuaris en el sistema.
2. El sistema presenta una llista de tots els usuaris registrats o permet la cerca d'un usuari específic.
3. L'administrador cerca i selecciona l'usuari a modificar de la llista.
4. El sistema mostra un formulari amb les dades actuals de l'usuari seleccionat.
5. L'administrador modifica els camps necessaris dins del formulari, que pot incloure canvis en el rol, estat de l'usuari, dades personals, etc.
6. Un cop les modificacions són completes, l'administrador envia el formulari.
7. El sistema valida les noves dades introduïdes per assegurar que compleixen amb tots els requisits i normatives.
8. Si les dades són vàlides, el sistema actualitza la base de dades amb la informació modificada.
9. El sistema mostra un missatge de confirmació a l'administrador que les dades de l'usuari han estat actualitzades correctament.

Alternatives de procés i excepcions:

7a. Dades introduïdes no vàlides o incompletes:

7a1. El sistema alerta l'administrador sobre els errors o els camps que falten.

7a2. L'administrador corregeix les dades i reenvia el formulari.

8a. Error en actualitzar les dades a la base de dades:

8a1. El sistema mostra un missatge d'error indicant que no s'ha pogut completar l'actualització.

8a2. L'administrador pot intentar de nou o contactar amb suport tècnic.

Diagrama:

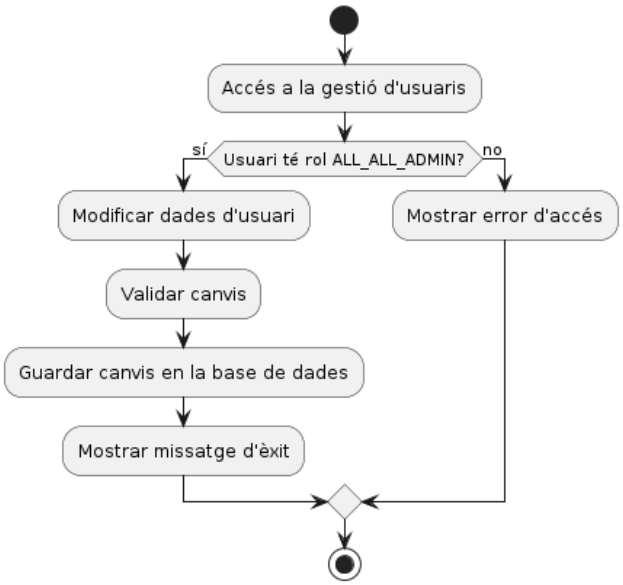

Figura 11 - Diagrama d'activitat CU 09

4.3.10 10 Visualitzar i afegir informació geogràfica

Resum de la funcionalitat: Permet als usuaris visualitzar, afegir i modificar dades geogràfiques de localitats, municipis, garatges, punts de recollida i zones de treball, incloent la capacitat de dibuixar i editar polígons representatius d'aquestes àrees en un mapa interactiu.

Paràmetres d'entrada:

- Dades geogràfiques: Informació detallada necessària per a cada nova entitat geogràfica, com nom, ubicació i dades específiques segons el tipus (p.ex., dimensions del garatge).
- Accions en el mapa: Interaccions per dibuixar o modificar polígons en el mapa.

Paràmetres de sortida:

- Visualització de dades actualitzades: Mostra la informació geogràfica actualitzada en el mapa.
- Confirmació de les modificacions: Notificacions que confirmen l'èxit o informen sobre qualsevol error en la manipulació de les dades.

Actors:

- Usuaris autoritzats com a administradors de sistema o gestors de logística amb permisos per gestionar informació geogràfica.

Precondició:

- L'usuari ha d'estar autenticat i tenir els permisos necessaris per interactuar amb la informació geogràfica.

Postcondició:

- Les noves localitats, municipis, garatges, punts de recollida i zones de treball són afegits al sistema i visualitzables en el mapa, amb polígons correctament definits.

Procés normal principal:

1. L'usuari accedeix al mòdul de gestió geogràfica en el sistema.
2. El sistema mostra un mapa interactiu juntament amb una llista o menú per seleccionar diferents entitats geogràfiques a visualitzar o modificar.
3. Per afegir una nova entitat:
4. L'usuari selecciona l'opció d'afegir i omple els detalls necessaris en un formulari específic.
5. L'usuari utilitza eines en el mapa per dibuixar el polígon que representa la nova entitat.
6. El sistema valida les dades i el polígon introduït.
7. Si les dades són vàlides, el sistema guarda la nova entitat i actualitza el mapa.
8. El sistema mostra un missatge de confirmació que l'entitat ha estat afegida i és visible en el mapa.
9. L'usuari pot continuar afegint més entitats o editar les existents.

Alternatives de procés i excepcions:

4a. Dades o polígon introduïts no vàlids:

4a1. El sistema mostra un missatge d'error detallant el problema.

4a2. L'usuari pot corregir els errors i re introduir les dades.

6a. Error en guardar les dades:

6a1. El sistema informa que no s'ha pogut guardar la informació.

6a2. L'usuari pot intentar de nou o contactar amb suport tècnic per a resolució de problemes.

Diagrama:

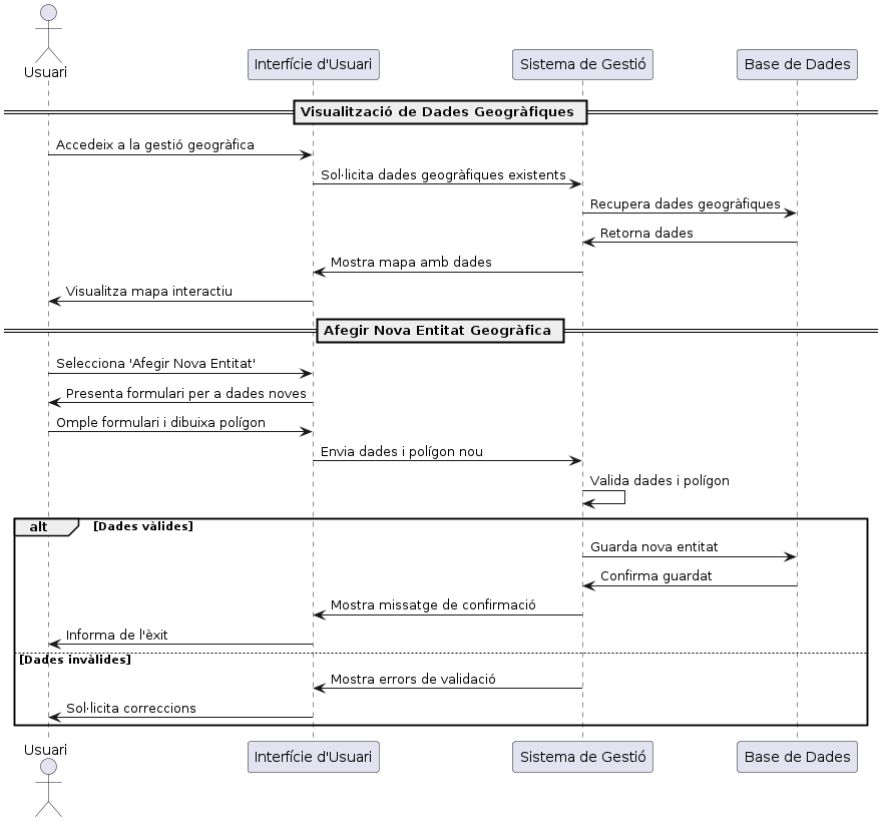

Figura 12 - Diagrama d'activitat CU 10

4.3.11 11 Emparellar punts de codis externs

Resum de la funcionalitat: Permet als usuaris emparellar codis de punts de recollida proporcionats per proveïdors externs amb punts de recollida interna del sistema, facilitant la integració i el maneig eficaç de les dades de recollida.

Paràmetres d'entrada:

- Codis de punts externs: Codis proporcionats pels proveïdors externs que necessiten ser associats.
- Selecció de punts interns: Els punts de recollida interna del sistema que corresponen als codis externs.

Paràmetres de sortida:

- Confirmació d'emparellament: Notificació que els codis han estat correctament associats.

Actors:

- Administradors de sistema o usuaris amb permisos específics per gestionar punts de recollida.

Precondició:

- Els codis externs han de ser pre-carregats o introduïts en el sistema.
- L'usuari ha d'estar autenticat i autoritzat per realitzar emparellaments.

Postcondició:

- Els codis de punts de recollida externs estan correctament vinculats als punts interns, facilitant operacions i seguiment futurs.

Procés normal principal:

1. L'usuari accedeix a la funció d'emparellament de codis en el sistema.
2. El sistema presenta una llista o forma d'introduir els codis de punts de recollida externs.
3. L'usuari introdueix o selecciona els codis externs a emparellar.
4. El sistema mostra una llista de punts de recollida interns disponibles.
5. L'usuari selecciona els punts interns que corresponen als codis externs introduïts.
6. El sistema valida l'associació entre els codis externs i els punts interns.
7. Si la validació és correcta, el sistema guarda l'emparellament en la base de dades.
8. El sistema mostra un missatge de confirmació que els codis han estat emparellats correctament.

Alternatives de procés i excepcions:

6a. Validació falla (p.ex., no hi ha correspondència clara o els codis són incorrectes):

6a1. El sistema informa a l'usuari dels errors trobats durant la validació.

6a2. L'usuari pot ajustar la selecció o corregir qualsevol error i re intentar l'emparellament.

Diagrama:

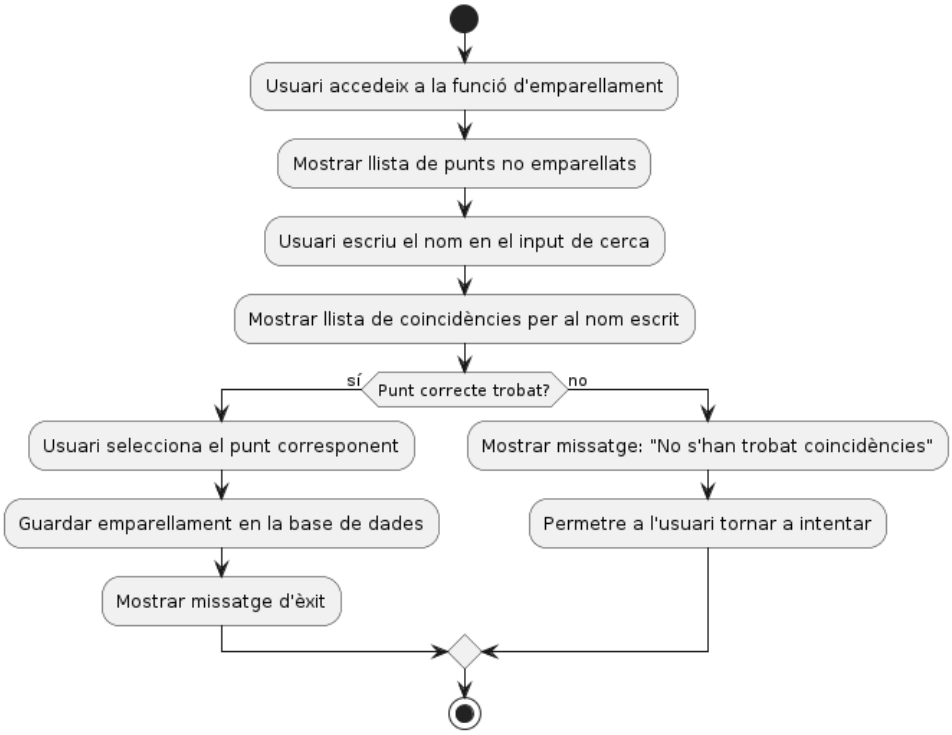

Figura 13 - Diagrama d'activitat CU 11

4.3.12 12 Enviar reserves a Gestrans i Fomento

Resum de la funcionalitat: Aquest cas d'ús descriu com el sistema planifica i executa de forma automàtica l'enviament de reserves als organismes de Fomento i Gestrans, assegurant que totes les reserves compleixin amb els requisits reglamentaris i siguin processades sense intervenció manual directa.

Paràmetres d'entrada:

- Reserves seleccionades per enviar: Determinades per la lògica de negoci implementada en el sistema basada en criteris específics com dates, estat de la reserva, i altres paràmetres rellevants.

Paràmetres de sortida:

- Confirmacions o errors d'enviament: Es registren en logs i es guarden en una base de dades per a la seva consulta posterior.

Actors:

- Sistema (planificador automatitzat): Responsable d'executar la tasca programada per enviar les reserves.

Precondició:

- Les reserves han d'estar validades i llestes per ser enviades segons els criteris establerts.
- Ha d'existir una configuració prèvia que determini la freqüència i les condicions sota les quals les reserves són enviades a Fomento i Gestrans.

Postcondició:

- Les reserves són enviades automàticament als organismes corresponents, i els resultats del procés són registrats.

Procés normal principal:

1. El planificador al backend s'activa segons la seva configuració (pot ser diari, setmanal, etc.).
2. El sistema selecciona automàticament les reserves que compleixen amb els criteris establerts per l'enviament.
3. El sistema intenta enviar les reserves a Fomento i Gestrans utilitzant serveis web o APIs proporcionades per aquests organismes.
4. Es capturen les respostes dels intents d'enviament:
5. Si l'enviament és exitós, es registra com a tal.
6. Si hi ha errors, es registren en els logs del sistema.
7. Els errors registrats s'emmagatzemen en una base de dades accessible des del frontend per a consultes i auditories.
8. S'actualitzen els estats de les reserves en el sistema basat en les respostes rebudes dels organismes.

Alternatives de procés i excepcions:

4a. Fallada en la connexió o en el servei web/API de Fomento o Gestrans:

4a1. Es re intentarà l'enviament segons la política de re intents configurada (p. ex., re intents cada x minuts/hores).

4a2. Es registraran detalls del fall per a revisió i correcció.

Diagrama:

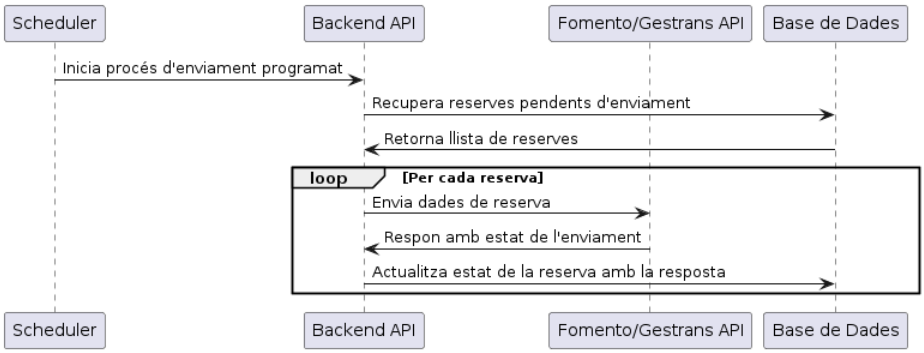

Figura 14 - Diagrama de seqüència CU 12

4.3.13 13 Processar reserves de correus electrònics

Resum de la funcionalitat: Aquest cas d'ús descriu com el sistema detecta i processa de forma automàtica les reserves rebudes a través de correus electrònics enviats per companyies externes, convertint aquestes dades en reserves gestionables dins del sistema.

Paràmetres d'entrada:

- Correus electrònics: Missatges rebuts que contenen dades de reserves procedents de companyies externes.

Paràmetres de sortida:

- Reserves processades: Dades de reserves que han estat extretes, validades i guardades en el sistema.

Actors:

- Sistema (Mòdul de processament de correu): Encarregat de la revisió i processament dels correus electrònics.

Precondició:

- El sistema ha de tenir accés a un compte de correu configurat per rebre aquests missatges.
- Deu existir una configuració prèvia per a identificar i validar les dades específiques de les reserves dins dels correus.

Postcondició:

- Les reserves procedents de correus electrònics estan correctament processades i incorporades com a reserves actives dins del sistema.

Procés normal principal:

1. El sistema monitoritza periòdicament la bústia de correu electrònic designada per a aquestes reserves.
2. Quan s'identifica un nou correu de les companyies especificades, el sistema l'analitza per extreure la informació de la reserva.
3. El sistema valida les dades de la reserva per assegurar-se que compleixen amb tots els requisits interns (p. ex., formats de data i hora, identificació del client).
4. Si les dades són vàlides, el sistema crea una nova reserva dins de la base de dades.
5. El sistema registra el resultat del processament, incloent qualsevol error o confirmació de l'èxit.
6. Els errors durant el processament són registrats i accessibles per a revisió futura.

Alternatives de procés i excepcions:

3a. Dades de reserva no vàlides:

3a1. El sistema registra l'error i pot notificar a un administrador per revisió manual.

3a2. Pot configurar-se per enviar una resposta automàtica al remitent indicant l'error i sol·licitant informació corregida.

4a. Error en la creació de la reserva en el sistema:

4a1. El sistema intenta de nou segons la configuració de re intents.

4a2. Si segueix fallant, el problema es registra per a una intervenció més detallada.

Diagrama:

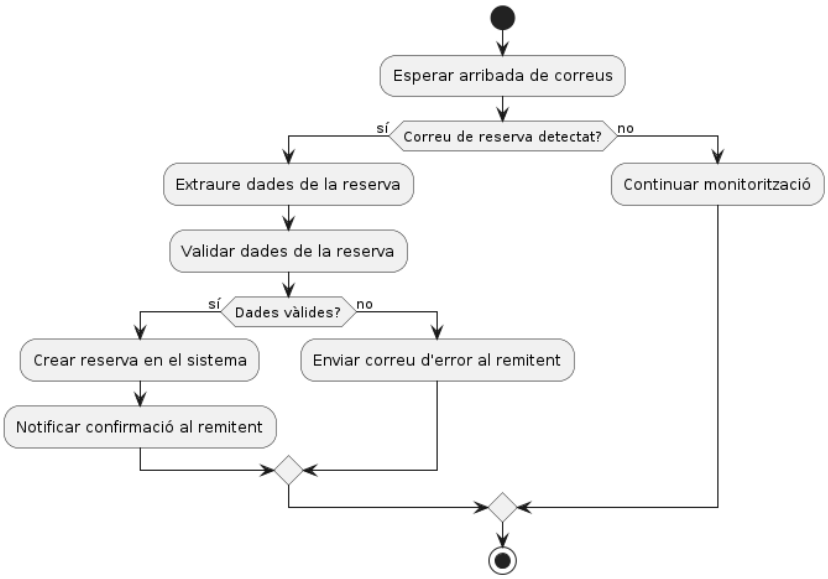

Figura 15 - Diagrama d'activitat CU 13

4.3.14 14 Oferir preus a agències

Resum de la funcionalitat: Aquest cas d'ús descriu el procés automàtic pel qual el sistema ofereix informació actualitzada sobre preus i disponibilitat de serveis a agències online com Booking.com, utilitzant webhooks per a una integració en temps real.

Paràmetres d'entrada:

- Sol·licituds de preus de les agències: Peticions rebudes via webhooks que demanen preus i disponibilitat.

Paràmetres de sortida:

- Respostes amb preus i disponibilitat: Informació enviada a les agències que reflecteix l'estat actual dels serveis oferts.

Actors:

- Sistema (API/Webhook handler): Component que gestiona les peticions de les agències i envia les respostes apropiades.
- Agències de reserves online (p. ex., Booking.com): Les plataformes que reben les dades de preus i disponibilitat.

Precondició:

- El sistema ha d'estar correctament integrat amb les APIs de les agències de reserves.
- Ha de tenir una base de dades actualitzada amb la informació de preus i disponibilitat.

Postcondició:

- Les agències reben informació precisa i actualitzada que els permet oferir reserves en temps real als seus clients.

Procés normal principal:

1. El sistema rep una sol·licitud de preus via webhook de una agència com Booking.
2. El sistema consulta la seva base de dades per determinar la disponibilitat actual i el preu dels serveis sol·licitats.
3. El sistema processa aquesta informació i genera una resposta formatada segons les especificacions de l'API de l'agència.
4. El sistema envia la resposta al webhook de l'agència, proporcionant els detalls requerits de preus i disponibilitat.
5. El sistema registra la transacció, inclòs qualsevol detall pertinent sobre la sol·licitud i la resposta.

Alternatives de procés i excepcions:

2a. Informació de disponibilitat o preus no actualitzada:

2a1. El sistema pot respondre amb un estat que indica la manca de disponibilitat o informació pendent d'actualització.

4a. Errors en l'enviament de la resposta:

4a1. Si hi ha un problema en el procés d'enviament de la resposta, el sistema ho intenta de nou segons una política de re intents prèviament definida.

4a2. Els errors durant l'enviament són registrats per a una posterior anàlisi i correcció.

Diagrama:

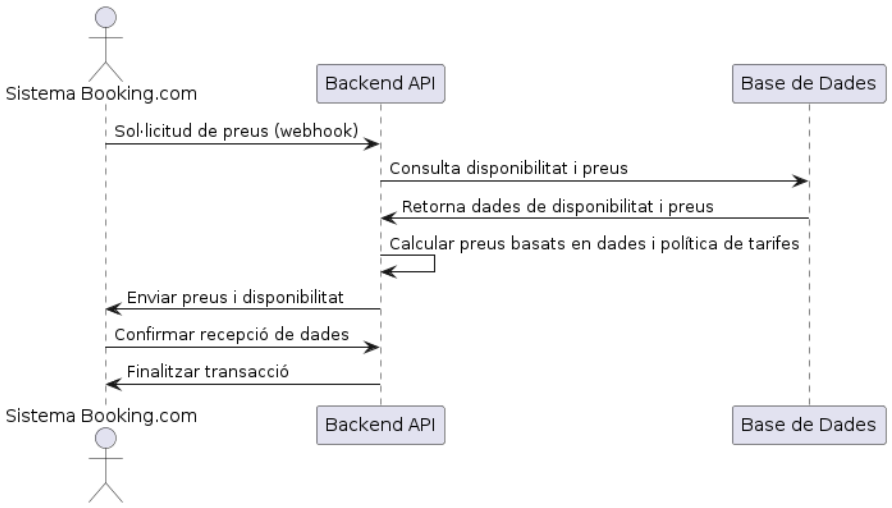

Figura 16 - Diagrama de seqüència CU 14

5 Disseny

El disseny d'una aplicació és un aspecte fonamental en el món del desenvolupament de software. No és suficient amb tenir un sistema que funcioni i es pugui fer servir, hem de garantir que el nostre sistema sigui robust i escalable. És per això que aspectes com un correcte disseny de la base de dades, o comunicació entre client i servidor, són fonamentals en aquest projecte. De forma breu, hem optat per un model relacional, per alguns dels aspectes que esmentaré a continuació, i va ser una decisió molt meditada durant la fase de preparació del projecte. En aquest apartat, faré una explicació de l'arquitectura general de l'aplicació i sobretot aprofundiré en el disseny del backend.

5.1 Model relacional

La decisió d'optar per un model relacional vers un model no-sql va ser un aspecte clau durant el disseny del sistema. Ficant els pros i contres de cadascun, vam arribar a la conclusió de que triar pel relacional ens donaria un millor rendiment i avantatge a l'hora d'executar queries complexes, ja que el nostre disseny involucra moltes taules i join, i la gran part de les queries no involucrarien només una entitat, sinó vàries relacionades entre si. A part, entre els avantatges, entrava el principi ACID i la garantia de poder executar transaccions, que és clau a l'hora de mantenir la integritat de les nostres dades.

En la següent Figura 17, extreta fent servir IntellijIDEA, es poden veure algunes de les taules principals, n'he fet selecció d'aquelles que he considerat més importants per a entendre tot el model de negoci, tot i que hi hagi bastants més al model relacional real:

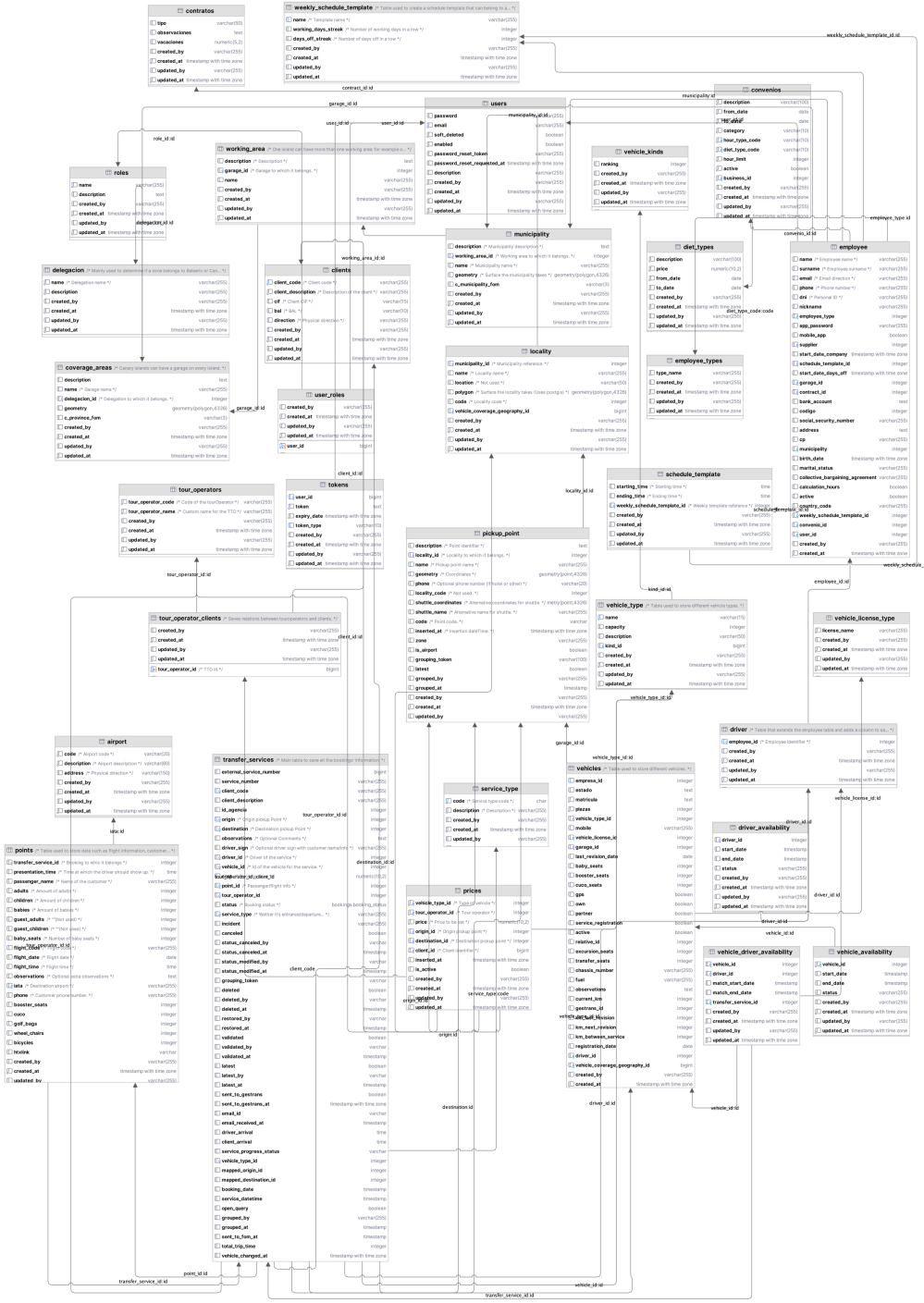

Figura 17 - Imatge model relacional

Tot aquest disseny el vam construir des de 0 i ha experimentat moltes modificacions des de que va començar el projecte el 2023, ja que s'han hagut d'afegir i modificar taules, a la vegada que esborrar-ne aquelles que no eren realment necessàries.

A més, hem implementat un enfocament rigorós d'auditoria per assegurar la traçabilitat de totes les modificacions en els registres. Cada taula del nostre sistema inclou columnes d'auditoria estandarditzades que capturen informació sobre quan i per qui es van fer canvis en les dades. Aquest enfocament no només compleix amb els requisits de seguretat, sinó que també facilita una gestió eficaç de la informació i ajuda a mantenir la integritat dels dades al llarg del temps.

Cada taula conté les següents columnes d'auditoria:

- **created_at**: Registra la data i hora exactes de la creació d'un registre.
- **created_by**: Emmagatzema informació sobre qui va crear el registre.
- **updated_at**: Registra la última data i hora en què el registre va ser modificat.
- **updated_by**: Indica qui va fer l'última modificació en el registre.

Columna	Descripció
created_at	Registra la data i hora exactes quan el registre es crea.
created_by	Indica quina persona o procés ha creat el registre.
updated_at	Registra la data i hora de l'última modificació del registre.
updated_by	Especifica qui ha realitzat l'última modificació en el registre.

Taula 1 - Columnes d'auditoria BD

A continuació explicaré només aquelles taules que considero imprescindibles i més importants per al negoci per a no allargar en excés aquesta part.

5.1.1 Transfer_services

Aquesta taula conté part de la lògica necessària per a guardar la informació d'una reserva. A la imatge no es pot apreciar gaire, però alguns dels camps més destacables són els següents:

- id
  - Descripció: Identificador únic per a cada reserva de servei. Es genera automàticament (serial).
  - Importància: Fonamental per rastrejar i gestionar individualment cada servei dins de la base de dades.
- service_number
  - Descripció: Número de servei donat per agències de tercers, permet poder obtenir modificacions i consumir APIs externes basant-se en al referència proporcionada per la agència en qüestió.
  - Importància: Clau per identificar serveis en comunicacions amb clients i dins d'operacions internes.
- client_code
  - Descripció: Codi de l'agència que ens ha proporcionat el servei, està vinculat a la taula bookings.clients mitjançant una clau forana.
  - Importància: Permet saber quina empresa ens ha derivat el servei, per a poder contactar en cas de ser necessari i poder analitzar el volum de treball amb cada empresa.
- origin i destination
  - Descripció: Claus foranes que apunten a locations.pickup_point, identificant els punts de recollida i destí de la reserva.
  - Importància: Crítics per a la logística del servei, permetent la planificació precisa de rutes i assignacions de recursos.

- observations
  - Descripció: Camp de text per a comentaris o notes especials sobre la reserva.
  - Importància: Proporciona flexibilitat per emmagatzemar informació addicional rellevant per al servei, com preferències del client o instruccions especials.
- cost
  - Descripció: Cost del servei, amb una restricció que assegura que sempre sigui un valor no negatiu.
  - Importància: Essencial per a la facturació i anàlisi financer dels serveis proporcionats.
- status
  - Descripció: Estat actual de la reserva, amb valors predeterminats com 'ACTIVE', i altres estats com 'CANCELLED', 'MODIFIED', etc.
  - Importància: Permet un seguiment eficaç del cicle de vida de cada reserva, crucial per a la gestió operativa i de servei al client.
- grouping_token, latest, grouped_by, i grouped_at
  - Descripció: Aquests camps donen suport a una implementació de versionat i agrupació sense una taula d'auditoria dedicada. grouping_token és un identificador que agrupa versions d'una mateixa reserva, latest indica si és la versió més recent, i grouped_by i grouped_at registren qui i quan es va realitzar l'agrupació.
  - Importància: Optimitza el rendiment evitant joins complexos amb taules d'auditoria, facilitant l'accés a versions anteriors d'una reserva per a auditories o revisions històriques.

5.1.2 Points

Aquesta taula està dissenyada per emmagatzemar informació detallada relacionada amb les reserves de serveis, incloent dades sobre els passatgers i els seus vols. Algunes de les columnes més importants:

- transfer_service_id
  - Descripció: Clau forana que vincula cada punt amb un servei específic de transferència registrat en la taula bookings.transfer_services.
  - Importància: Essencial per relacionar detalladament els punts amb els seus serveis de transferència corresponents, facilitant la gestió integrada de les operacions de reserva.

- presentation_time
  - Descripció: Hora programada per a la presentació del conductor.
  - Importància: Crítica per coordinar la logística del servei de transport, garantint la puntualitat en la recollida dels passatgers.
- passenger_name
  - Descripció: Nom del passatger principal associat amb la reserva.
  - Importància: Important per a la personalització del servei i per a verificacions durant el procés de recollida.
- adults, children, babies
  - Descripció: Quantitat d'adults, nens i nadons que participen en la reserva.

    - Importància: Crucials per assegurar que el vehicle assignat compleixi amb els requisits de capacitat i confort per a tots els passatgers.
- baby_seats, booster_seats, cuco, golf_bags, wheel_chairs, bicycles
    - Descripció: Detalla els requisits específics d'equipament com seients per a bebès, maletes de golf, cadires de rodes, entre d'altres.
    - Importància: Fonamental per preparar adequadament el vehicle segons les necessitats específiques dels clients, millorant la satisfacció del servei.
- flight_code, flight_date, flight_time
    - Descripció: Informació detallada del vol dels passatgers, inclouent el codi de vol, la data i l'hora.
    - Importància: Essencial per a la planificació precisa dels horaris de recollida i entrega en coordinació amb els horaris de vol.
- observations
    - Descripció: Notes addicionals o requeriments específics relacionats amb la reserva.
    - Importància: Proporciona un espai per registrar qualsevol informació addicional que pot ser rellevant per al servei.
- iata
    - Descripció: Codi IATA de l'aeroport de destinació, que enllaça a la taula hub.airport.
    - Importància: Clau per a la coordinació logística en les activitats aeroportuàries i la planificació de rutes.
- phone
    - Descripció: Número de telèfon del client.
    - Importància: Crucial per a la comunicació directa amb el client, permetent ajustaments i confirmacions en temps real.

5.1.3 Employees i divers

Les taules bookings.employee i bookings.driver estan dissenyades per funcionar en conjunció dins del sistema de gestió de la informació del personal d'una empresa, oferint una estructura robusta i flexible per a la gestió de dades relacionades amb tot tipus d'empleats, especialment els conductors.

5.1.3.1 Estructura i Funcionalitat Conjunta

La taula bookings.employee serveix com a nucli central on s'emmagatzemen les dades bàsiques de tots els empleats, incloent informació com el nom, cognom, email, telèfon, i altres dades personals i laborals. Aquesta taula està dissenyada per ser extensiva i adaptativa a les necessitats de diversos tipus d'empleats dins de l'organització.

Per als empleats que tenen rols específics, com els conductors, la taula bookings.driver actua com una extensió de la taula bookings.employee. Aquesta configuració utilitza el patró de disseny de composició, on la taula bookings.driver no només hereta la identitat de l'empleat a través de la clau forana employee_id, sinó que també afegeix detalls específics que són pertinents només per als conductors.

5.1.3.2 Composició i Extensió de Dades

- Composició: Cada entrada a la taula bookings.driver està directament relacionada amb una entrada a bookings.employee, indicant que tot conductor és primer un empleat. Aquesta relació es maneja mitjançant l'ús de la clau

forana employee_id, que vincula cada conductor amb el seu perfil d'empleat corresponent a la taula principal d'empleats.
- Extensió: La informació específica dels conductors, que podria incloure detalls com horaris de conducció, preferències de vehicles, o historials de llicències de conducció, s'emmagatzema dins de la taula bookings.driver. Aquest enfocament permet que les dades específiques del rol estiguin organitzades i accessibles només quan sigui necessari, mantenint la taula bookings.employee més neta i més orientada a la gestió general dels empleats.

5.1.4 Roles, users, user_roles i tokens

A continuació, es presenta la definició detallada dels camps de les taules d'autenticació segons el model proporcionat, que inclouen roles, users, tokens, i user_roles:

- id (aplica a totes les taules mencionades)
  - Descripció: Identificador únic generat automàticament per a cada entrada.
  - Importància: Clau per a la identificació única i referència a cada registre, essencial per mantenir la integritat de les dades.
- name (taula roles)
  - Descripció: Nom del rol assignat als usuaris.
  - Importància: Identifica el rol i els permisos associats, facilitant la gestió de l'accés a les funcionalitats del sistema.
- description (taula roles)
  - Descripció: Descripció breu del rol que ajuda a entendre les responsabilitats i els permisos concedits.
  - Importància: Ajuda a comprendre el propòsit del rol i guia en la configuració adequada dels permisos.
- email (taula users)
  - Descripció: Correu electrònic de l'usuari, utilitzat com a identificador per al login.
  - Importància: necessari per a l'autenticació dels usuaris i la comunicació principal amb ells.
- password (taula users)
  - Descripció: Contrasenya xifrada de l'usuari per accedir al sistema.
  - Importància: Crucial per a la seguretat de l'accés de l'usuari al sistema.
- soft_deleted i enabled (taula users)
  - Descripció: Indicadors que controlen l'estat de l'usuari dins del sistema, com si està habilitat o ha estat eliminat de manera suau (ocultat sense eliminar les dades).
  - Importància: Permet la gestió flexible dels comptes d'usuari sense perdre informació històrica.
- token (taula tokens)
  - Descripció: Token d'autenticació assignat a un usuari per gestionar les sessions.
  - Importància: Permet la validació de les sessions d'usuari i la gestió de l'accés a les funcionalitats del sistema de manera segura.
- expiry_date (taula tokens)
  - Descripció: Data i hora en què el token caducarà i deixarà de ser vàlid.

- Importància: assegurar que els tokens d'autenticació són temporals i incrementar la seguretat de la gestió de sessions.

- user_id i role_id (taula user_roles)
  - Descripció: Clau forana que vincula els usuaris amb els seus rols respectius.
  - Importància: Fonamental per definir l'accés d'un usuari a diferents parts del sistema basat en els rols assignats.

5.2 Arquitectura general de l'aplicació

Pel que fa a l'arquitectura del sistema, com he esmentat prèviament, fem servir un backend dissenyat amb Spring Boot i un frontend que utilitza Vue.js, basat en el patró Model-Vista-Controlador (MVC), que proporciona una estructura robusta i escalable per a desenvolupar aplicacions complexes. Aquesta arquitectura separa clarament les responsabilitats en diferents components, facilitant el manteniment i l'evolució del sistema.

En aquest apartat tractaré aquests aspectes previs i explicaré l'arquitectura que hem seguit a AWS i es pot veure a la Figura 18. Jo particularment, m'he enfocat en la part del backend, que correspon al requadre superior esquerre.

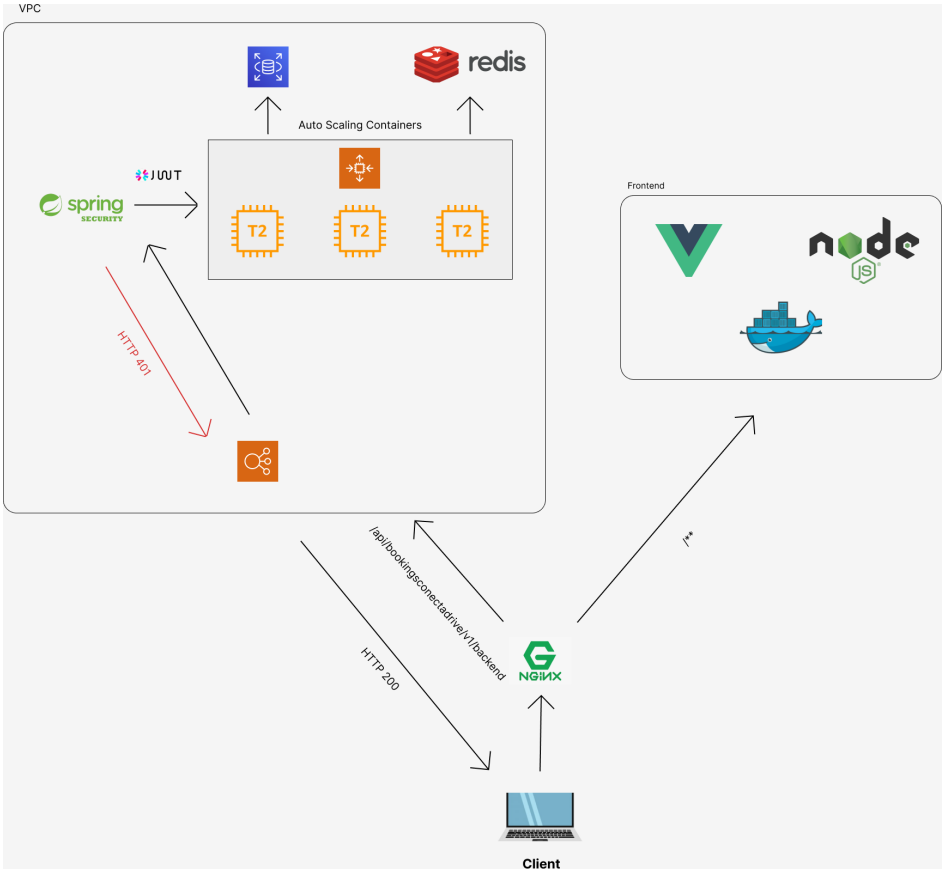

Figura 18 - Esquema de l'arquitectura

5.2.1 Components de l'Arquitectura MVC

- Model
  - Descripció: El Model representa les dades i la lògica de negoci de l'aplicació. En el nostre sistema, aquest component està format per les entitats JPA (Java Persistence API) que corresponen a les taules de la base de dades PostgreSQL.

    - Funció: Gestiona l'accés i la manipulació de les dades. A més, conté la lògica de negoci que defineix com es processen aquestes dades.
- Vista
    - Descripció: La Vista és responsable de presentar la informació a l'usuari final. En el nostre cas, utilitzem Vue.js per construir una interfície d'usuari interactiva i responsiva.
    - Funció: Mostra les dades del model a l'usuari i captura les interaccions d'aquest, que després es passen al controlador per al processament.
- Controlador
    - Descripció: El Controlador actua com a intermediari entre el Model i la Vista. En el backend, hem implementat controladors RESTful utilitzant Spring Boot.
    - Funció: Rep les peticions HTTP del client (Vista), invoca la lògica de negoci corresponent en el Model, i retorna les respostes adequades a la Vista.

5.2.2 Altres Aspectes Importants del Sistema

A més dels detalls generals de l'arquitectura MVC, hi ha diversos aspectes tècnics i decisions de disseny que val la pena destacar per mostrar la complexitat i la solidesa del sistema:

- **Lombok**: Hem utilitzat la llibreria Lombok per evitar el boilerplate code en les nostres classes Java. Lombok genera automàticament codi com constructors, getters, setters, i mètodes equals i hashCode, reduint així la quantitat de codi que cal escriure manualment i mantenir.
- **Generació de Stubs amb JAX-WS**: Per a la integració amb serveis SOAP, hem utilitzat JAX-WS per generar els stubs necessaris. Aquesta eina permet crear automàticament el codi client per consumir serveis web SOAP, facilitant la integració amb sistemes externs que utilitzen aquest protocol, com el del ministeri de Transport.
- **Spring Data JPA**: Utilitzem Spring Data JPA per simplificar l'accés a dades a través de repositoris, que gestionen les operacions *CRUD*[23] i altres consultes personalitzades. Això ens permet centrar-nos en la lògica de negoci sense preocupar-nos per la implementació dels detalls d'accés a la base de dades.
- **Monitorització i Logs**: Hem integrat SLF4J de Lombok per a la generació de logs a cada classe. Això permet tenir una traça detallada de les operacions sense necessitat de configurar un sistema de logs especialitzat.
- **Desplegament en el Núvol**: El sistema es desplega en entorns de testing i producció en el núvol, utilitzant un pipeline de CI/CD que automatitza el procés de desplegament després de cada push a la branca principal del repositori de codi. Això assegura que les noves funcionalitats i correccions es despleguin ràpidament i de manera consistent.

[23] CRUD: Create, read, update, delete

5.2.3 Patrons de Disseny Utilitzats

A més del patró MVC, hem implementat diversos patrons de disseny per millorar la cohesió i el desacoblament del codi:

- **Patró de Repositori**: Aquest patró s'utilitza per aïllar la lògica d'accés a les dades del codi de negoci. Els repositoris JPA gestionen les operacions CRUD (Crear, Llegir, Actualitzar, Esborrar) i permeten definir mètodes de consulta personalitzats.
- **Patró de Servei**: S'utilitza per encapsular la lògica de negoci. Els serveis són definits mitjançant interfícies que després es poden implementar. Això permet una abstracció que facilita la prova i el manteniment del codi.
- **Injecció de Dependències**: Spring Boot utilitza la injecció de dependències per gestionar les relacions entre els components. Això permet desacoblar el codi i facilita la substitució de components sense afectar altres parts del sistema.
- **Patró Strategy**: Hem utilitzat aquest patró per encapsular algorismes i comportaments intercanviables. Aquest patró permet que una família d'algorismes es puguin definir i encapsular, i que els algorismes es puguin intercanviar sense afectar el codi client que els utilitza. Un exemple és la implementació de diferents estratègies d'enviament d'informació al ministeri de Transport, per a enviar actualitzacions de reserves, esdeveniments del conductor, o cancel·lacions.
- **Patró Composition**: S'ha utilitzat per crear entitats compostes que encapsulen altres objectes per modelar relacions complexes. Per exemple, l'entitat TransferServiceEnt inclou altres entitats com PointEnt i Client, que permeten una major flexibilitat i reutilització del codi.

5.2.4 Integració i Seguretat

- **Autenticació i Autorització**: Amb Spring Security, el sistema autentica els usuaris mitjançant tokens. Els tokens són validats per cada petició per assegurar que l'usuari té els permisos adequats per accedir als recursos sol·licitats.
- **Gestió de Sessions**: Els tokens permeten gestionar les sessions de manera segura, ja que cada petició ha de contenir un token vàlid.
- **Roles Personalitzats**: A més dels rols d'usuari estàndard, hem definit rols personalitzats per a les agències amb les que treballem, permetent un control granular sobre les operacions que poden realitzar dins del sistema.

5.2.5 Arquitectura AWS

Pel que fa a l'entorn cloud on desplegar la nostra aplicació hem optat per AWS. Hem considerat que és un sistema complet, robust i fiable i permet escalar de forma gairebé automàtica.

Tot i haver optat per aquest proveïdor de IAAS, hem decidit dissenyar una arquitectura que sigui flexible i no estigui lligada als serveis de cada proveïdor, és per això que gran part dels desplegaments estan fets amb bash scripting i utilitzant kubernetes, fet que ens permetria migrar a qualsevol altre cloud provider, com GCP o Azure, i portar el nostre sistema de forma senzilla sense haver de fer grans canvis.

Aquesta decisió és essencial en termes de flexibilitat, i a continuació esmentaré de quina forma hem aconseguit dissenyar aquesta arquitectura.

5.2.5.1 Desplegament de bases de dades (RDS)

Per a desplegar les bases de dades hem seleccionat *RDS*[24], que és el servei que Amazon ofereix per a allotjar bases de dades PostgreSQL. En termes de portabilitat, les bases de dades no ens preocupen, ja que la eina psql permet fer un backup i després restaurar-ho al destí de forma senzilla. Pel que fa als recursos, vam escollir les diferents configuracions pels entorns de DEV, TEST i PROD:

Entorn	Tipus instància	VCPUs	RAM (GB)
DEV	db.t3.small	2	2
TEST	db.t3.small	2	2
PROD	db.t3.medium	2	4

Taula 2 - Recursos de les BD a AWS

Es pot apreciar que hi ha tres entorns de treball, la base de dades des DEV es farà servir pels desenvolupadors i poder realitzar proves sense afectar al contingut real. L'entorn de TEST es farà servir pel product owner, en aquest cas Alejandro, que s'encarregarà de validar les funcionalitats i crear els issues necessaris per a corregir funcionalitats o possibles bugs que es detectin. Finalment, l'entorn de producció serà, com bé indica el nom, l'anomenat PROD, i contindrà les dades i reserves reals dels clients i empleats.

5.2.5.2 Desplegament dels contenidors (EKS)

Pel que fa al desplegament d'ambdòs frontend i backend, s'ha optat per utilitzar kubernetes, amb la solució que AWS ofereix, anomenada EKS. Hem triat aquesta opció per a, en cas de voler migrar a un altre proveïdor cloud que ofereixi kubernetes, tenir aquesta opció i fer-ho de la forma més senzilla possible.

Primer de tot, destacar que tenim un clúster per a l'entorn de TEST i un altre per a l'entorn de PROD, d'aquesta forma aïllem les dades dels dos.

Pel que fa al número de pods, actualment només en tenim un per a cada aplicació, amb una política d'escalat definida, que crearà noves instàncies si el número de peticions a la api o clients al frontend creix sobtadament. Com aquesta aplicació no té un us públic actualment, no ens preocupa aquesta saturació del servei, ja que pel moment el número màxim de clients que hi ha connectats simultàniament son d'1 – 10. De tota manera, hem preparat el sistema d'aquesta forma amb kubernetes per a tenir la possibilitat d'escalar senzillament en el moment que sigui necessari.

Per a aconseguir aquest desplegament de kubernetes, hem optat per la CLI de helm, que difereix de la pròpia de kubectl. Aquest framework permet automatitzar el deploy a AWS sense haver de configurar-ho mitjançant la UI, i així poder desplegar múltiples entorns amb la mateixa configuració de forma molt simple.

A part d'aquest desplegament de contenidors, hem definit un Proxy revers amb nginx, que s'encarrega de dur a terme el Routing de les peticions al contenidor corresponent. Degut a que fem servir el nom de domini test.conectadrive.com, vam definir una política, mitjançant la qual les peticions que tinguessin el patró

[24] RDS: Relational database system

`/api/bookingsconectadrive/v1/**` anirien al backend. D'aquesta forma, totes les altres peticions al domini de test.conectadrive.com, resoldran a una vista del frontend.

5.2.5.3 Gestió dels dominis (Route 53)

Per a poder dur a terme la gestió dels diferents dominis que fem servir als nostres entorns, vam optar per utilitzar Route 53, que és el servei de gestió de DNS que ofereix Amazon. A més, per a poder allotjar aquests dominis a Amazon, vam haver d'obtenir un certificat vàlid, utilitzant en aquest cas Certificates Manager.

5.2.5.4 Emmagatzemament de les imatges

Per a poder allotjar les imatges construïdes amb docker, vam seleccionar el servei de *ECR*[25], que permet emmagatzemar múltiples imatges al fer el build per a després ser utilitzades al desplegar en la fase de deploy. A part, permet totes les funcionalitats necessàries per a dur a terme un versionat i anàlisi de seguretat de les imatges.

5.3 Disseny general del frontend

A continuació, mostraré de forma resumida el resultat d'algunes pàgines del frontend per a poder veure el resultat final. No aprofundiré per a centrar-me en la part del backend, però és una bona manera d'entendre de quina forma s'utilitzaran les funcionalitats que explicaré més endavant.

5.3.1 Pàgina de login

Component realitzat amb primeVue, podem veure que es dona la possibilitat de guardar els credencials, si es marca el checkbox es guardarà a les cookies.

En cas de no recordar la contrasenya, podrem clicar l'enllaç que es mostra i sol·licitar un enllaç, seguint el flux que he explicat prèviament als casos d'ús.

[25] ECR: Elastic container registry

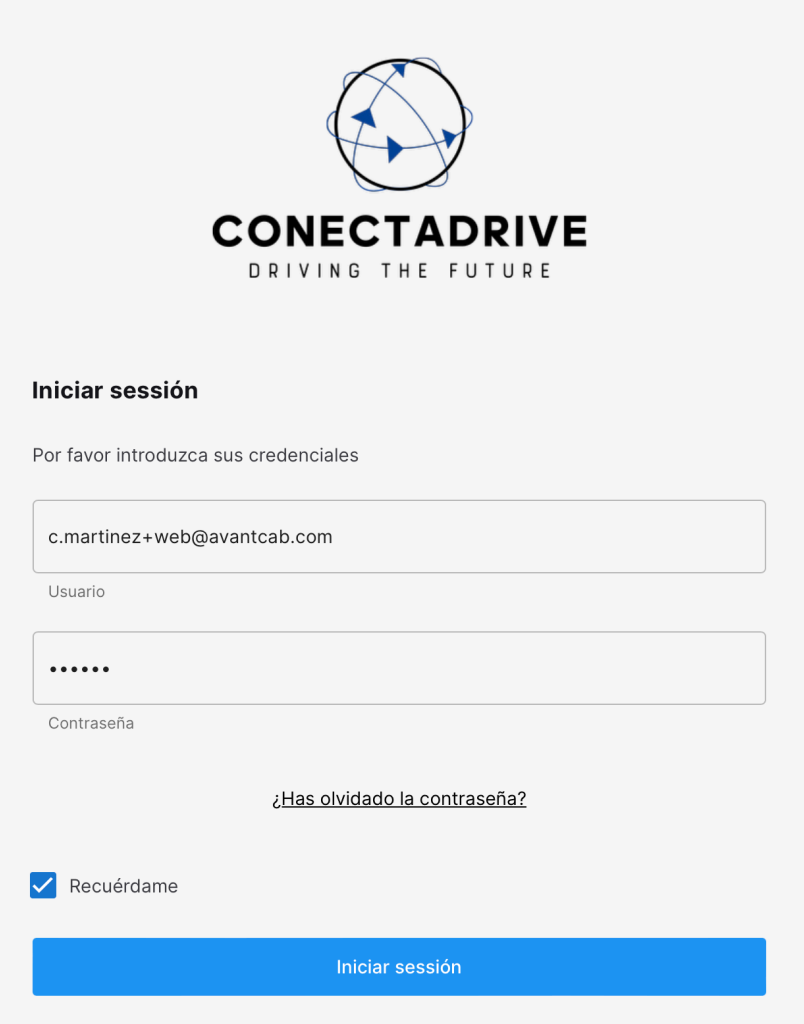

Figura 19 - Pàgina de login del frontend

5.3.2 Gestor de reserves

Podem veure que hi ha diverses columnes, cada una de les quals té un filtre que permet fer la cerca de les dades a la BD. Aquest filtratge està permet combinar diferents columnes i fent un filtratge amb JPA utilitzant especificacions, que explicaré més endavant. En quant a l'ordenació, només permet ordenar per una columna actualment.

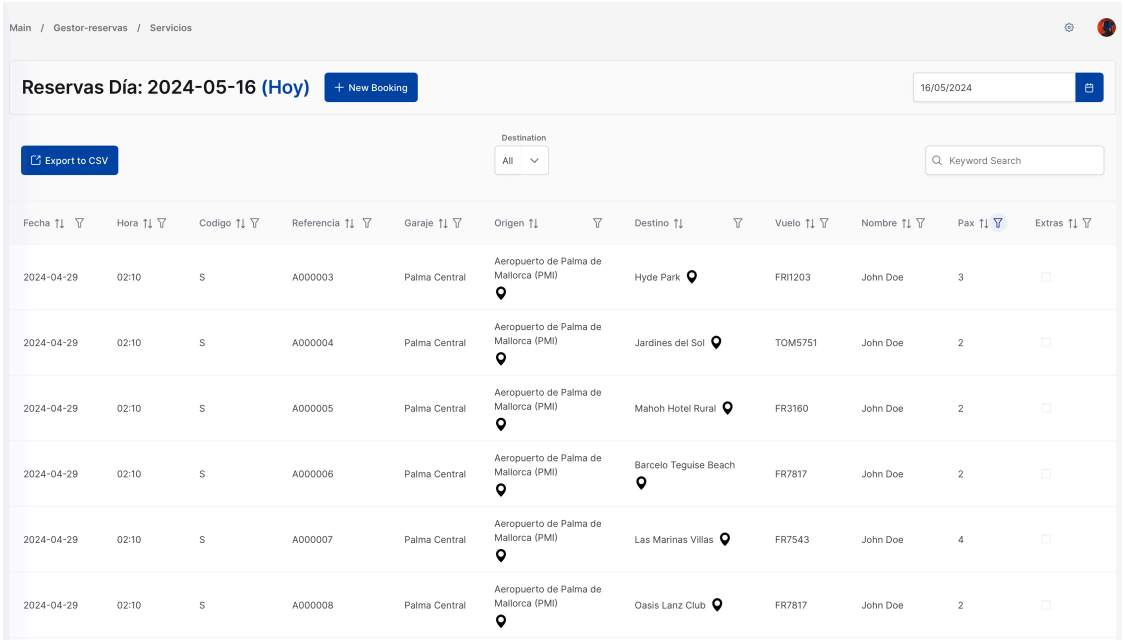

Figura 20 - Gestor de reserves del frontend

5.3.3 Importadors de fitxers d'agències

Aquesta pàgina admet fitxers XML, JSON i CSV. Com podem veure hi ha una targeta a la part superior que mostra els fitxers actualment pujats, i permet eliminar-los individualment o clicant el botó vermell a la cantonada superior esquerra.

Es poden pujar els tres formats alhora i el backend detectarà automàticament quin conversor ha d'utilitzar seguint una patró strategy. A més, la taula conté filtres a les columnes importants, que permeten buscar dades a les files que estan actualment carregades al component del DataTable.

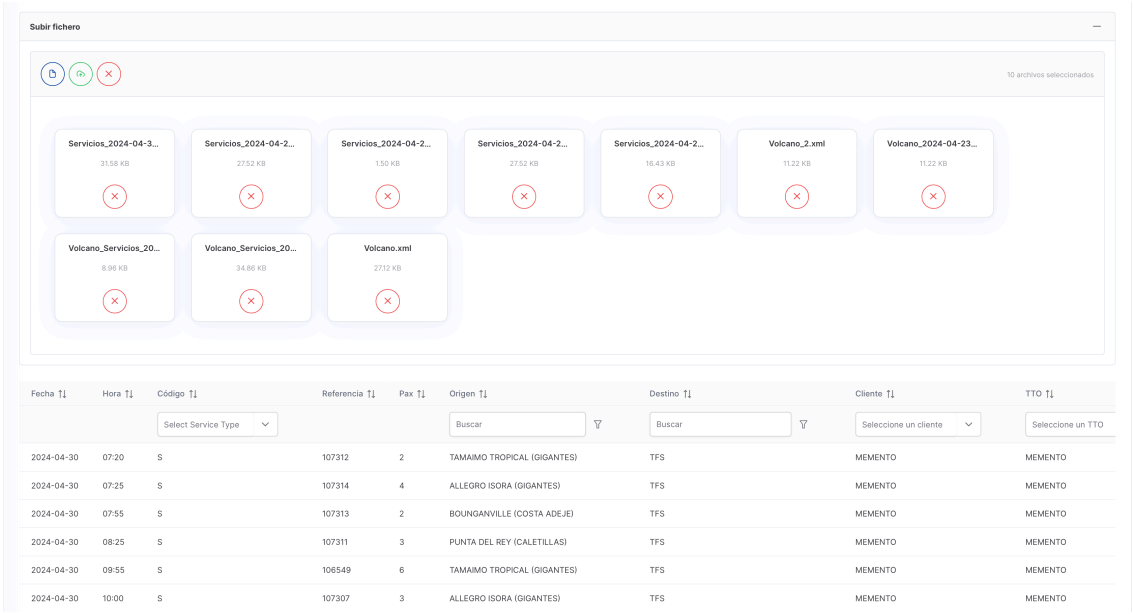

Figura 21 - Importador de fitxers del frontend

5.3.4 Localitats

Aquesta pàgina és l'encarregada de mostrar els polígons i permetre la seva edició i creació. Utilitzem el Mapbox GL JS [10] per a poder mostrar el mapa i crear diferents capes on mostrar els polígons i les línies que separen cada municipi. Al mapa hi ha icones personalitzades que permeten escollir un llapis per a dibuixar un polígon, que un cop guardar es persistirà a la base de dades utilitzant la llibreria PostGIS.

A més, a la dreta es mostren tots els polígons creats, i al fer clic a cadascun d'ells, es fa un zoom automàtic a la zona. Si canviem la illa que estem visualitzant amb el dropdown d'a dalt a la dreta, el mapa farà un zoom-out i zoom-in a la illa que haguem escollit.

Aquests polígons són necessaris per a poder establir diferents preus en funció de la zona on sigui el servei, i permeten definir les zones on un vehicle pot operar.

La pàgina es veu així:

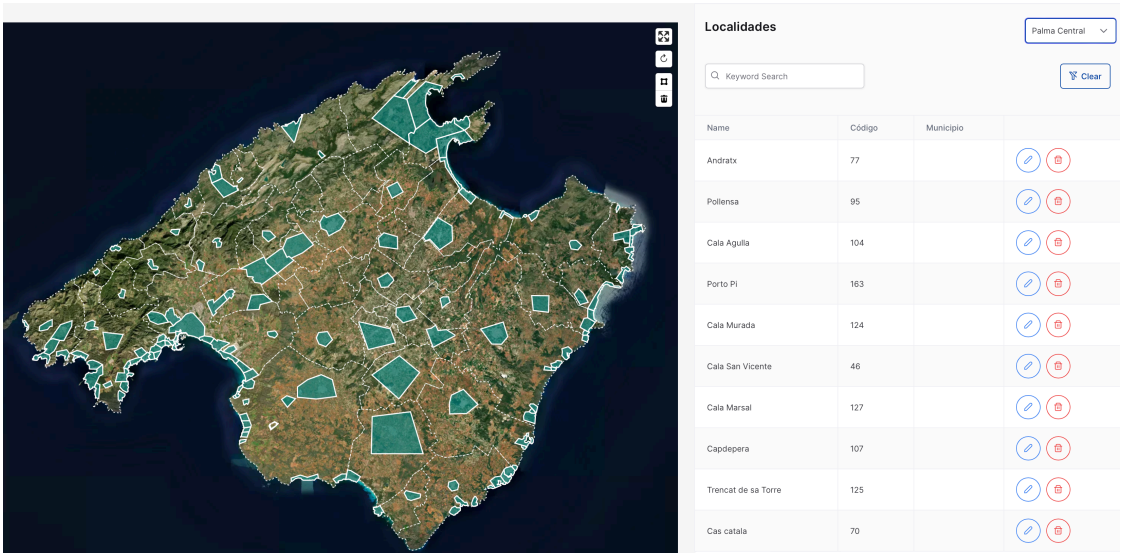

Figura 22 - Localitats del frontend

6 Implementació

En aquest apartat tractaré implementacions concretes que jo mateix he desenvolupat en el backend i considero interessants. Parlaré del disseny seguit per a generalitzar la creació de CRUD amb un repositori i entitat, la security filter chain implementada amb Spring Security, el processament de fitxers d'agències, entre d'altres.

A més, mencionar que la majoria de serveis eviten concrecions abstraient la lògica a una interfície que després es pot implementar 1 o N vegades. D'aquesta forma injectem la interfície i no la concreció amb la finalitat de poder tenir múltiples implementacions sense afectar al disseny de l'aplicació. Aquesta injecció de dependències permetria usar un @Qualifier per a indicar el nom del @Bean i injectar una concreció o una altra.

6.1 Implementació de CRUD genèrics

En el nostre projecte, hem utilitzat una implementació de CRUDs genèrics per gestionar les operacions bàsiques de diferents entitats de manera eficient i reutilitzable. Aquesta estratègia ens ha permès simplificar i estandarditzar la manipulació de dades en el backend. A continuació, es detallen les parts més rellevants de la implementació, incloent fragments de codi que il·lustren com es fa servir aquest enfocament.

6.1.1 Definició del contracte

Definim una interfície genèrica GenericInterface que declara els mètodes CRUD bàsics. Aquesta interfície actua com a contracte per a qualsevol servei que vulgui implementar operacions CRUD. Els mètodes inclouen findAll, findById, save, update, deleteById i existsById.

```
public interface GenericInterface<T, ID> {
    List<T> findAll();

    T findById(ID id);

    T save(T entity);

    T update(ID id, T entity);

    void deleteById(ID id);

    boolean existsById(ID id);
}
```

6.1.2 Implementació del servei genèric

La classe GenericService implementa GenericInterface utilitzant JpaRepository de Spring Data JPA per interactuar amb la base de dades. Aquesta classe abstracta defineix els mètodes CRUD bàsics i delega les operacions a JpaRepository.

```
@AllArgsConstructor
public class GenericService<T, ID, R extends JpaRepository<T, ID>>
implements GenericInterface<T, ID> {

    protected final R repository;
    protected final String entityClass;

    public List<T> findAll() {
        return repository.findAll();
    }

    public T findById(ID id) {
        return repository.findById(id).orElseThrow(() -> new
EntityNotFoundException(entityClass + " with id " + id + " not found"));
    }

    @Transactional
    public T save(T entity) {
        return repository.save(entity);
    }

    @Transactional
    public T update(ID id, T entity) {
        if (!repository.existsById(id)) {
            throw new EntityNotFoundException(entityClass + " with id " +
id + " not found");
        }
        return repository.save(entity);
    }
// Implementació dels altres mètodes de l'interfície
}
```

6.1.3 Implementació específica del servei

Per a una entitat específica com TourOperator, creem una classe de servei que estengui GenericService i implementi una interfície específica. Això ens permet personalitzar les operacions si cal.

Com podem veure, definim una interfície que estén la genèrica per si volem crear mètodes personalitzats per cada concreció.

```
@Service
public class TourOperatorServiceImpl extends GenericService<TourOperator,
Long, TourOperatorRepository> implements TourOperatorService {
    public TourOperatorServiceImpl(TourOperatorRepository repository) {
        super(repository, "Tour Operator");
    }
}

public interface TourOperatorService extends
GenericInterface<TourOperator, Long> {
    // Afegir els mètodes específics pel servei
}
```

6.1.4 Exemple d'ús en el controlador

Utilitzem el servei genèric en un controlador REST per exposar les operacions CRUD. Això permet que les operacions siguin accessibles mitjançant peticions HTTP.

Aquest enfocament permet evitar les concrecions, ja que el controlador no coneix la implementació concreta del servei, sinó que treballa amb la interfície. Això fa que el sistema sigui més flexible i fàcil de mantenir, ja que es poden canviar les implementacions dels serveis sense haver de modificar els controladors.

```
@RestController
@RequestMapping("/api/bookings-info/tour-operator")
@AllArgsConstructor
public class TourOperatorController {
    private final TourOperatorService tourOperatorService;

    @GetMapping
    public ResponseEntity<List<TourOperator>> getAll() {
        return ResponseEntity.ok(tourOperatorService.findAll());

    }

    @GetMapping("/{id}")
    public ResponseEntity<TourOperator> getById(@PathVariable Long id) {
        return ResponseEntity.ok(tourOperatorService.findById(id));
    }
    // La resta de mètodes són els mateixos que a GenericController.java
}
```

6.1.5 Beneficis de consumir serveis per interfícies

1. **Desacoblament**: En utilitzar interfícies, els controladors es desacoblen de les implementacions concretes dels serveis. Això permet canviar la implementació dels serveis sense afectar els controladors, millorant la mantenibilitat del codi.
2. **Flexibilitat**: Les interfícies permeten definir mètodes específics que no estan presents en la interfície genèrica. Això permet personalitzar els serveis segons les necessitats específiques de l'aplicació.
3. **Reutilització de Codi**: Amb l'ús de serveis genèrics, podem reutilitzar el codi comú per a operacions CRUD en múltiples entitats, reduint la duplicació de codi i facilitant la seva gestió.

6.2 Importació i conversió de fitxers externs

Treballar amb agències de tercers és una funcionalitat clau per a Conectadrive, però la interoperabilitat és una característica que no està present en aquesta comunicació i requereix adaptar-se als formats de cada empresa. Els formats dels fitxers amb que es pot treballar són CSV, JSON i XML actualment i la estratègia mostrada a continuació requereix tenir prèviament tenir creades les classes necessàries per a mapejar cada objecte donat un fitxer concret.

6.2.1 Definició del contracte

Aquesta interfície defineix els mètodes que qualsevol servei d'importació d'arxius ha d'implementar, garantint que es puguin gestionar diferents tipus d'arxius de manera flexible.

```
public interface FileService<S, T> {
    boolean canHandleFile(MultipartFile file);

    List<S> importFile(MultipartFile file);
```

```
    List<S> parseServices(MultipartFile file);

    List<S> convertServices(List<T> services);
}
```

6.2.2 Implementació del contracte

Aquesta classe abstracta proporciona la implementació base de la interfície FileService, definint mètodes comuns per als diferents formats d'arxius.

```
@AllArgsConstructor
@Service
@Slf4j
public abstract class FileServiceImpl<T> implements
FileService<TransferServiceEnt, T> {
    protected final JAXBConversionUtils jaxbConversionUtils;
    protected final Converter<List<T>, List<TransferServiceEnt>>
converter;
    protected final BookingManagementServiceInterface
bookingManagementService;
    protected List<String> supportedFileExtensions;

    /**Els altres mètodes**/

    @Override
    public List<TransferServiceEnt> convertServices(List<T> service) {
        return converter.convert(service);
    }

    protected abstract List<T> getServices(String xmlFile);

    @Override
    public boolean canHandleFile(MultipartFile file) {
        String fileName = file.getOriginalFilename();
        if (fileName == null) return false;

        String extension = fileName.substring(fileName.lastIndexOf('.') +
1).toLowerCase();
        return supportedFileExtensions.contains(extension) &&
additionalFileChecks(file);
    }

    protected abstract boolean additionalFileChecks(MultipartFile file);

}
```

6.2.3 Implementació específica del servei

Exemple de com s'especialitza FileServiceImpl per tractar un tipus específic d'arxiu XML.

Si parem atenció, podem veure que al construcció del super, s'indica una llista amb els formats de fitxers amb que pot treballar. Aquest disseny rep una llista degut a que un proveïdor pot tenir més d'un format de fitxer i això ofereix flexibilitat a l'hora de parsejar els serveis. A més, el mètode additionalFileChecks, permet rebre un fitxer i indicar si pot processar-lo utilitzant etiquetes clau que venen a cada fitxer i són úniques per aquell proveïdor.

```
@Service
@Slf4j
public class VolcanoFileServiceImpl extends
FileServiceImpl<VolcanoBooking> {
    public VolcanoFileServiceImpl(JAXBConversionUtils
jaxbConversionUtils,
                                  Converter<List<VolcanoBooking>,
List<TransferServiceEnt>> converter,
                                  BookingManagementServiceInterface
bookingManagementService) {
        super(jaxbConversionUtils, converter, bookingManagementService,
List.of("xml"));
    }

    @Override
    protected List<VolcanoBooking> getServices(String xmlFile) {
        return jaxbConversionUtils.convertStringXMLToObject(xmlFile,
Volcano.class).getBookings();
    }

    @Override
    protected boolean additionalFileChecks(MultipartFile file) {
        try {
            String content = new String(file.getBytes());
            return content.contains("<VOLCANO>");
        } catch (Exception e) {
            log.error("Error checking Volcano file");
            throw new RuntimeException(e);
        }
    }
}
```

6.2.4 Exemple d'ús del servei

Fragment que mostra com s'utilitzen les implementacions de FileService per processar els fitxers.

Podem observar que injectem la llista de FileService utilitzant Lombok. Aquest approach és molt més mantenible que injectar cada implementació de FileService de forma manual, i ens permet crear la implementació sense haver-ne de modificar l'ús a ImportPrivateService. D'aquesta forma, si haguéssim d'incorporar un nou format de fitxer, simplement hauríem d'estendre la classe en qüestió i Spring s'encarregaria d'injectar la dependència de forma automàtica. Comentar que haurem d'anotar la implementació amb un @Service o @Component per a que això sigui possible.

Per últim, podem veure que el mètode getFileConverter retorna la instancia que pot processar aquell fitxer, i fent ús del polimorfisme d'herència podem processar els fitxers amb utilitzant la implementació concreta que es requereixi en cada cas.

```
public class ImportPrivateService {
    private final List<FileService<TransferServiceEnt, ?>> fileServices;
    private final GestransInteractionServiceInterface
gestransInteractionService;

    public List<TransferServiceEnt> importFiles(List<MultipartFile>
files) {
        List<TransferServiceEnt> services = new ArrayList<>();
        for (MultipartFile file : files) {
            FileService<TransferServiceEnt, ?> fileService =
getFileConverter(file);
```

```
            services.addAll(fileService.importFile(file));
        }
        return services;
    }

    private FileService<TransferServiceEnt, ?>
getFileConverter(MultipartFile file) throws Exception {
        return fileServices.stream()
                        .filter(fileService ->
fileService.canHandleFile(file))
                        .findFirst()
                        .orElseThrow(() -> new Exception("File type not
supported"));
    }
}
```

6.2.5 Beneficis d'aquest enfocament

Aquest enfocament orientat a objectes millora l'escalabilitat i la flexibilitat del sistema. Permet incorporar ràpidament suport per a nous formats de fitxer afegint noves subclasses sense afectar el funcionament de les implementacions existents.

6.3 Estructura general d'Spring Security

Per a configurar la seguretat de la nostra aplicació, hem fet una implementació amb rols personalitzats d'Spring Security. Aquest enfocament ens permet definir scopes per a indicar si un usuari té accés a dades "Mobile", "web" o bé

ambdues. A més, hem configurat una política de cors i cadena de filtres per a aconseguir un logging exhaustiu i tenir una bona visibilitat al estar desplegat al cloud.

6.3.1 Configuració

Per a aquesta implementació, hem fet servir la versió 2.4.5 de Spring Security.

Primer de tot, necessitem anotar amb @Configuration i @EnableWebSecurity una classe que estengui WebSecurityConfigurerAdapter. Aquesta classe té un mètode concret del que n'hem de fer un override per a aplicar la nostra configuració, n'és el següent:

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
            .csrf().disable()
            .cors()
            .and()
.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELES
S)
            .and()
            .exceptionHandling()
            .authenticationEntryPoint(authenticationEntryPoint())
            .and()
            .authorizeRequests()
            .antMatchers(HttpMethod.PATCH,
getPrefixedPath("/api/booking/v2/bookings/{bookingReference}")).hasAnyAut
hority(getClientSpecificPlusApplicationRoles("BOOKING_UPDATE_AGENCY"))
    .antMatchers(SecurityConfigProperties.getAuthWhitelist()).permitAll()
            .anyRequest().hasAnyAuthority(getApplicationRoles())
            .and()
            .addFilterBefore(redirectionFilter(),
JwtAuthenticationTokenFilter.class);
}
```

En aquest mètode estem establim un política de seguretat stateless, a part d'habilitar la política de CORS i demanar que cada petició tingui un rol dels vàlids per a la aplicació. Podem veure també que al final s'estableix el filtre de redirecció com a el més prioritari, aquest és el que farem servir per a fer logging abans i després de cada petició.

Per a poder autenticar els usuaris utilitzant Spring Security, necessitarem crear una classe que implementi la interfície UserDetailsService i fer override del mètode "loadUserByUsername". Aquest mètode serà usat internament per Spring quan s'executi el AuthenticationManager.authenticate. La idea serà injectar el repositori d'usuaris en aquesta class i fer una cerca de l'usuari en BD. En cas que no existeixi, hauríem de llençar una excepció, que el @ControllerAdvice cachejarà per a retornar un error a l'usuari en el moment de fer el login.

6.3.2 Logging de les peticions

Per a poder tenir control sobre les peticions que es fan, vam definir un sistema de logging sofisticat, que no només fa logging de la URL a la que es fa la petició, si no també la IP d'origen, headers, temps de resposta, etc.

Aquest logging es fa estenent la classe OncePerRequestFilter, això permet fer logging abans de rebre la petició, processar la petició, i un cop acabada loggejar la resposta. És interessant la forma en que funciona aquest sistema, ja que si durant la validació del token o autenticació de l'usuari es produeix algun error, Spring marcarà la petició com a "commited", per a indicar que s'ha produït algun error i la resposta ja està sentenciada. Per a assegurar que aquest serà el primer filtre que s'executarà a cada aplicació, hem de configurar-ne l'ordre a la classe de configuració esmentada anteriorment i a part anotar el nostre logging-filter amb @Order(1).

6.3.3 Logging d'una petició:

En els casos de les peticions, fer logging dels headers és essencial, ens permeten veure el token proporcionat a cada petició i analitzar si es tracta d'un token caducat en cas de notar alguna anomalia en el sistema. Podem usar eines com jwt.io [14] per a veure els claims que conté el token i veure el user_id o data d'emissió del token i expiració.

```
    2024-05-16  18:07:01.667     INFO  85586  ---  [nio-8084-exec-6]
o.a.b.security.filter.RedirectionFilter         :    [Request]    POST
/api/bookingsconectadrive/v1/auth/login  from  IP:  0:0:0:0:0:0:0:1  with
Parameters:    Headers: host=[localhost:8084]; user-agent=[curl/8.4.0];
accept=[*/*]; content-type=[application/json]; content-length=[71];
```

6.3.4 Logging d'una resposta

6.3.4.1 Successful

```
    2024-05-16  18:08:44.754     INFO  85586  ---  [nio-8084-exec-8]
o.a.b.security.filter.RedirectionFilter      :   [Response   200]   POST
/api/bookingsconectadrive/v1/auth/login completed in 476 ms
```

6.3.4.2 Error

```
    2024-05-16  18:07:01.905   ERROR  85586  ---  [nio-8084-exec-6]
o.a.b.security.filter.RedirectionFilter      :   [Response   401]   POST
/api/bookingsconectadrive/v1/auth/login completed in 239 ms
```

6.4 Desplegament de l'aplicació en entorn de prova

Pel que respecta a l'arquitectura cloud, el projecte està desplegat a AWS, fet que tractarà més en profunditat el meu amic i company Roger Massana. És per això, que explicaré de quina forma hem desplegat el projecte en un entorn de proves, amb una base de dades nova sense informació sensible del client per a poder mostrar en aquest treball.

6.4.1 Configuració del domini i registres DNS

Per a poder fer el desplegament, vam aprofitar la llicència d'estudiant que ofereix Github, per a adquirir un domini gratuït a name.com. Com conectadrive.com és el nom de domini que fa servir el software oficial, vam optar per conectadrive.software.

Un cop adquirit el domini, vam haver de crear un certificat de propietat per a poder-ho registrar, vam fer servir la comanda:

```
openssl req -newkey rsa:2048 -nodes -keyout conectadrive.software.key
-out conectadrive.software.csr
```

Un cop registrat el domini i creat el certificat, vam procedir a instal·lar el certificat SSL gratuït amb Let's encrypt per a poder servir el nostre contingut utilitzant el protocol HTTPS.

Un cop instal·lat, vam establir les redireccions necessàries per a poder apuntar els dominis als dos proveïdors diferents que vam fer servir (Netlify [11] i DigitalOcean [12]). En resum, utilitzarem:

- conectadrive.software per al frontend
- api.conectadrive.software per al backend

Per a configurar les redireccions del DNS, primer vam configurar el desplegament a cada proveïdor, primer a Netlify i després a DigitalOcean com s'explica més abaix.

Les redireccions que vam fer van ser aquestes:

Tipus	Servidor	Resposta	Comentari
A	api.conectadrive.software	134.122.73.181	IPV4 backend
A	conectadrive.software	52.58.254.253	IPV4 netlify
AAAA	api.conectadrive.software	2a03:b0c0:3:d0::1b55:1001	IPV6 digitalOcean per al cert SSL.
CNAME	www.api.conectadrive.software	api.conectadrive.software	Redirecció de la API.
CNAME	www.conectadrive.software	conectadrive.software	Redirecció del frontend.

Taula 3 - Redireccions DNS

Com es pot veure a la taula anterior, redirigim les peticions de cada domini i sub-domini al proveïdor corresponent. També destacar que l'IPV6 és necessària per a poder emetre un certificat vàlid des del droplet del backend amb la eina "certbot". Això ens permetrà evitar problemes de cors i servir el nostre contingut del backend mitjançant HTTPS.

6.4.2 Desplegament del frontend

Per a desplegar el frontend, hem optat per triar Netlify, un cloud provider que permet servir pàgines web estàtiques de forma senzilla i escalable. Aquest proveïdor permet connectar el nostre compte de Github per a poder fer el deploy amb un dels repositoris del que siguem propietaris. Aquest procés va ser relativament senzill i straightforward, vam connectar el nostre compte, seleccionar el repositori, i canviar les variables d'entorn. Al ser una aplicació amb node el procés és bastant trivial i la pàgina estava corrent als 5 minuts d'haver començat el procés.

6.4.3 Desplegament del backend

En quant al desplegament del backend, el procés va ser més complex, ja que no es tracta d'una pàgina web estàtica, i la configuració no és tan trivial. Vam optar per la opció de DigitalOcean, ja que Github ofereix 200€ de crèdit.

Més concretament vam escollir la opció dels droplets, amb una imatge de docker basada en ubuntu 22.04, que té un kernel de Linux.

Concretament la instància té 2 vCPUs, 2 GB de RAM i 25 GB d'emmagatzematge. A més, té habilitat l'accés per SSH i té la opció d'escalar els recursos en cas que sigui necessari. De totes formes, per a l'entorn de proves és més que suficient aquesta configuració.

Un cop ja sabem on ho desplegarem, podem passar a explicar quin serà el funcionament dins d'aquest droplet. Al tenir una imatge custom de docker, podrem instal·lar i executar tots els contenidors que desitgem. La decisió ha sigut configurar un Proxy revers amb nginx [13], que permeti redirigir les peticions al droplet al contenidor corresponent. Tindrem un contenidor per al backend, que serà l'aplicació amb Spring Boot amb el port 8084 escoltant. Després tindrem un altre contenidor per a la BD PostgreSQL, que executarà una imatge personalitzada amb Postgis, ja que RDS de AWS si que suporta postgis out of the box, però el droplet no. Ambdós contenidors s'estaran executant a una network personalitzada per a permetre les peticions des del backend a la BD i configurarem una IP estàtica per a evitar problemes en cas que els contenidors es reiniciïn.

6.4.3.1 Dockerfile

El dockerfile que utilitzarem durant aquest procés serà el següent:

```
FROM amazoncorretto:17
WORKDIR /app

COPY target/*.jar /app/app.jar
COPY src/main/resources/* /app/resources/

EXPOSE 8084
ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```

Copiarem tot el contingut de la carpeta resources, ja que hi ha fitxers json necessàris per a la recepció de correus electrònics. Aquesta carpeta contindrà també el fitxer application.yml, tot i que sobreescriurem algunes de les configuracions en el moment d'aixecar el contenidor.

6.4.3.2 Docker build

Un cop definit el Dockerfile, podem procedir a construir la imatge, hi han més passos del normal degut a que el meu portàtil té un chip Apple Silicon M3 Pro:

```
#!/bin/bash
IMAGE_VERSION=$1

REPO_URL="registry.digitalocean.com/conectadrive"
IMAGE_NAME="backend"
PLATFORMS="linux/amd64,linux/arm64"
docker buildx create --name mybuilder
docker buildx use mybuilder

docker buildx build --platform $PLATFORMS -t
$REPO_URL/$IMAGE_NAME:$IMAGE_VERSION . --push
```

6.4.3.3 Docker start

Ara que tenim la imatge construïda i pujada al repositori, podem fer pull des del backend per a executar el contenidor, utilitzarem aquest script per a executar la nova instancia.

Com podem veure, sobreescriurem la configuració de la base de dades per a canviar els credencials i la IP que fem servir, la BD també s'executarà amb una IP estàtica.

```
#!/bin/bash
IMAGE_VERSION=$1

# Eliminar el contenidor si ja està en execució
docker rm backend -f

# Assegurar-se que la xarxa 'mynetwork' existeix, es crea si no està
present.
docker network create mynetwork || true

# Executar el contenidor amb una IP estàtica dins la xarxa creada.
docker run --name backend --network mynetwork --ip 172.18.0.3 -p
8084:8084 -d \
  -e SPRING_DATASOURCE_USERNAME=avantcab \
  -e SPRING_DATASOURCE_URL="jdbc:postgresql://172.18.0.2:5432/postgres" \
  -e SPRING_DATASOURCE_PASSWORD="" \
  -e GMAIL_CREDENTIALS_FILE-PATH="/app/resources/test.json" \
  registry.digitalocean.com/conectadrive/backend:$IMAGE_VERSION
```

6.4.3.4 Certbot

Un cop tenim els contenidors corrent, podem passar a configurar el certificat SSL amb certbot.

Com hem configurat prèviament la redirecció a la IPV6 del droplet, executar la següent comanda permetrà emetre el certificat SSL necessari per a configurar el nostre servidor amb Nginx:

```
certbot --nginx -d api.conectadrive.software
```

Aquesta comanda sol·licitarà la expedició d'un certificat vàlid i el guardarà al directori */etc/letsencrypt/renewal/api.conectadrive.software.conf*

6.4.3.5 Configuració Proxy Nginx

Com he esmentat prèviament, utilitzarem Nginx per a configurar un Proxy que permeti redirigir qualsevol petició a la IP del servidor, a la IP del contenidor corresponent.

Primer de tot, crearem un fitxer especialment per a la configuració del nostre subdomini, al path: */etc/nginx/sites-available/api.conectadrive.software*

6.4.3.5.1 Redirecció d'HTTP a HTTPS

Per assegurar que totes les sol·licituds es realitzin a través d'HTTPS, es configura un servidor que escolta al port 80 i redirigeix totes les peticions a HTTPS:

```
server {
  listen 80;
  server_name api.conectadrive.software;

  # Redireccionar totes les peticions a HTTPS
  return 301 https://$server_name$request_uri;
}
```

6.4.3.5.2 Configuració del servidor HTTPS

El servidor HTTPS està configurat per escoltar al port 443 i utilitzar certificats SSL per assegurar les connexions. Aquí es detalla la configuració:

```
server {
  listen 443 ssl;
  server_name api.conectadrive.software;

  # Especificar la ubicació dels certificats SSL
  ssl_certificate
/etc/letsencrypt/live/api.conectadrive.software/fullchain.pem;
  ssl_certificate_key
/etc/letsencrypt/live/api.conectadrive.software/privkey.pem;

  # Configuracions SSL recomanades
  include /etc/nginx/snippets/ssl-params.conf;

  # Configuració de la ubicació del proxy
  location / {
    proxy_pass http://172.18.0.3:8084;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
  }
}
```

6.4.3.5.2.1 Explicació

- **listen 443 ssl;**: Aquest bloc de configuració indica a Nginx que escolti al port 443 i utilitzi SSL per assegurar les connexions.
- **server_name api.conectadrive.software;**: Especifica el nom del servidor per al qual s'aplica aquesta configuració.
- **ssl_certificate i ssl_certificate_key**: Indiquen la ubicació dels certificats SSL emesos per Let's Encrypt per assegurar les connexions HTTPS.
- **include /etc/nginx/snippets/ssl-params.conf;**: Inclou configuracions SSL recomanades per millorar la seguretat.
- **location /**: Defineix com Nginx ha de gestionar les sol·licituds que coincideixen amb aquesta ubicació. En aquest cas, totes les sol·licituds es passen al contenidor Docker.
- **proxy_pass http://172.18.0.3:8084;**: Indica a Nginx que passi les sol·licituds al backend a l'adreça IP 172.18.0.3 i port 8084.

- **proxy_http_version 1.1;**: Utilitza HTTP/1.1 per a la comunicació amb el backend.
- **proxy_set_header Upgrade $http_upgrade;**: Configura la capçalera Upgrade per a WebSocket.
- **proxy_set_header Connection 'upgrade';**: Configura la capçalera Connection per permetre connexions WebSocket.
- **proxy_set_header Host $host;**: Reenvia la capçalera Host original.
- **proxy_cache_bypass $http_upgrade;**: Evita l'ús de la memòria cau per a les sol·licituds WebSocket.

7 Avaluació

La validació de qualsevol projecte és un punt clau i necessari per assegurar la qualitat i el correcte funcionament del sistema abans de qualsevol desplegament a producció. Per garantir-ho, hem implementat una sèrie de tests exhaustius, tant per al frontend com per al backend, utilitzant tecnologies i eines que proporcionen una cobertura completa i una execució eficient.

7.1 Tests End-to-End (E2E) amb Playwright

Per al frontend, utilitzem Playwright per realitzar tests end-to-end (E2E). Aquesta eina permet simular el comportament dels usuaris a la interfície gràfica i verificar que totes les funcionalitats treballen com s'espera. Els tests E2E es realitzen de la següent manera:

- **Execució Paral·lela**: Els tests s'executen en paral·lel per reduir el temps total de validació. Això permet identificar problemes ràpidament i millorar l'eficiència del procés de testing.
- **Execució en Background**: Per evitar l'obertura de múltiples finestres del navegador, els tests s'executen en background. Això no només millora el rendiment del sistema durant la validació, sinó que també redueix l'ús de recursos.
- **Integració en el Procés de Deploy**: Abans de fer un desplegament, s'executen automàticament els tests amb Playwright. Si algun test falla, el procés de build (npm run build) no es realitzarà i, per tant, no es desplegarà la nova versió.

7.2 Tests del Backend amb Mockito i JUnit4

Per al backend, hem implementat tests unitaris i d'integració utilitzant Mockito i JUnit4. Aquesta combinació permet validar tant la lògica de negoci individual com la integració de diferents components del sistema. Els tests del backend es realitzen de la següent manera:

- **Tests Unitaris**: Verifiquen la funcionalitat de components individuals del sistema de backend, assegurant que cada unitat funcioni correctament de manera aïllada.
- **Tests d'Integració**: Comproven la interacció entre múltiples components del sistema, assegurant que treballen junts correctament.
- **Testcontainers**: Utilitzem Testcontainers per crear entorns de prova efímers amb bases de dades i altres serveis necessaris. Això garanteix que els tests es realitzin en un entorn controlat i similar al de producció.
- **Integració en el Procés de Build**: Durant l'execució de de la comanda mvn clean install, s'executen tots els tests del backend. Si algun test falla, el procés d'instal·lació no es completarà i no es desplegarà la nova versió.

7.3 Pipeline de Deploy i Notificació Automàtica

El pipeline de deploy està dissenyat per ser flexible i no dependre d'una plataforma cloud específica. Això ens permet migrar a qualsevol altre proveïdor cloud amb facilitat. Hem implementat scripts de bash per gestionar tot el procés, incloent la validació i el deploy.

- **Scripts de Bash**: Els scripts automatitzen l'execució de tests, el build de l'aplicació i el desplegament. Això assegura que el procés sigui consistent i repetible.
- **Correu Automàtic**: Un cop completat el desplegament, si tot ha estat exitós, s'envia un correu automàtic amb la informació de la nova versió desplegada. Aquest correu inclou missatges com:
    - https://test.conectadrive.com/api/bookingsconectadrive/v1/info changed from 1.0.99 to 1.0.100

7.3.1 Pipeline per al Backend

El pipeline de deploy per al backend s'executa utilitzant un script de bash que segueix aquests passos:

1. **Autenticació amb Amazon ECR**: Login al repositori ECR d'Amazon.
2. **Clonació del Projecte**: Clona el projecte des d'un repositori especificat.
3. **Compilació i Test**: Executa mvn clean install per compilar el projecte i executar els tests.
4. **Construcció de la Imatge Docker**: Crea una imatge Docker del backend.
5. **Pujada de la Imatge Docker**: Puja la imatge Docker a Amazon ECR.
6. **Desplegament a Kubernetes**: Utilitza Helm per desplegar la imatge a un clúster de Kubernetes.

7.3.2 Pipeline per al Frontend

El pipeline de deploy per al frontend també s'executa utilitzant un script de bash amb els següents passos:

1. **Autenticació amb Amazon ECR**: Login al repositori ECR d'Amazon.
2. **Clonació del Projecte**: Clona el projecte des d'un repositori especificat.
3. **Construcció de la Imatge Docker**: Crea una imatge Docker del frontend utilitzant Node.js per construir l'aplicació i Nginx per servir-la.
4. **Pujada de la Imatge Docker**: Puja la imatge Docker a Amazon ECR.
5. **Desplegament a Kubernetes**: Utilitza Helm per desplegar la imatge a un clúster de Kubernetes.

Aquesta estratègia de validació i desplegament assegura que cada versió desplegada ha passat els quality checks, tant al frontend com al backend. Això no només permet detectar i solucionar problemes abans que arribin a producció, sinó que també proporciona la flexibilitat per adaptar-nos a diferents entorns cloud segons sigui necessari. La integració de Playwright per al frontend i Mockito juntament amb JUnit4 per al backend, combinada amb l'ús de Testcontainers i scripts de bash, forma una solució robusta i adaptable per a la gestió del procés de deploy.

7.4 Validació dels requisits no funcionals

- RNF1: Temps de resposta
  - Validació: Es van realitzar proves de càrrega utilitzant eines com JMeter per simular múltiples usuaris accedint al sistema simultàniament. Els resultats van demostrar que el temps de resposta es manté per sota dels 3 segons fins i tot sota càrrega elevada.
- RNF2: Suport de càrrega
  - Validació: Es van dur a terme proves d'estrès per assegurar que el sistema pot gestionar fins a 10,000 usuaris simultanis. Es va utilitzar un entorn de prova configurat amb Kubernetes per escalar automàticament els recursos segons la demanda, comprovant que el rendiment es manté estable sense degradació significativa.
- RNF3: Encriptació de dades
  - Validació: Totes les transaccions financeres i dades personals es transmeten utilitzant SSL/TLS per assegurar la protecció de les dades en trànsit. Es pot validar el correcte funcionament fent una crida amb postman. Veure figura Figura 23.
- RNF4: Autenticació segura
  - Validació: S'utilitza OAuth2 per a l'autenticació i gestió de sessions amb tokens JWT. Es van realitzar auditories de seguretat per verificar que el sistema compleix amb els estàndards de seguretat establerts.
- RNF5: Protecció contra atacs
  - Validació: Es van dur a terme proves de penetració per identificar i mitigar vulnerabilitats com SQL injection, XSS i CSRF. A més, es van configurar les cookies amb http-only per a l'entorn de producció. Veure Figura 24
- RNF6: Compatibilitat de navegador
  - Validació: Es van fer proves de compatibilitat en les últimes versions dels navegadors Chrome, Firefox, Safari i Edge per assegurar una experiència d'usuari consistent.
- RNF7: Logging i Monitorització
  - Validació: Es va implementar un sistema de logging centralitzat utilitzant SLF4J amb Lombok per registrar totes les accions crítiques i errors. Això facilita el seguiment i resolució de problemes. Veure Figura 25.
- RNF8: Actualitzacions del sistema
  - Validació: Es van utilitzar estratègies de desplegament blau-verd i canary releases per permetre actualitzacions sense interrupció del servei. Es van realitzar proves en un entorn de staging per validar que les actualitzacions es poden aplicar sense problemes.
- RNF9: Integració amb sistemes externs
  - Validació: El sistema proporciona API RESTful que permeten la integració amb altres sistemes de tercers. Es van realitzar proves d'integració amb sistemes externs per assegurar la correcta comunicació i intercanvi de dades.
- RNF10: Exportació i importació de dades
  - Validació: Es van implementar funcionalitats per a l'exportació i importació de dades en formats estàndard com CSV i JSON. Es van dur

a terme proves per validar que les dades es poden exportar i importar correctament sense pèrdua d'informació.

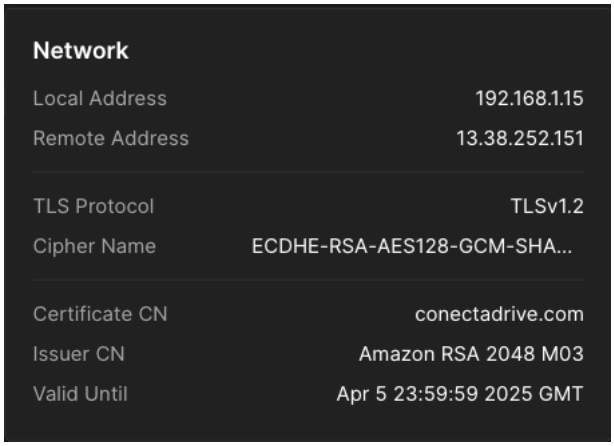

Figura 23 - Validació ús TLS RNF3

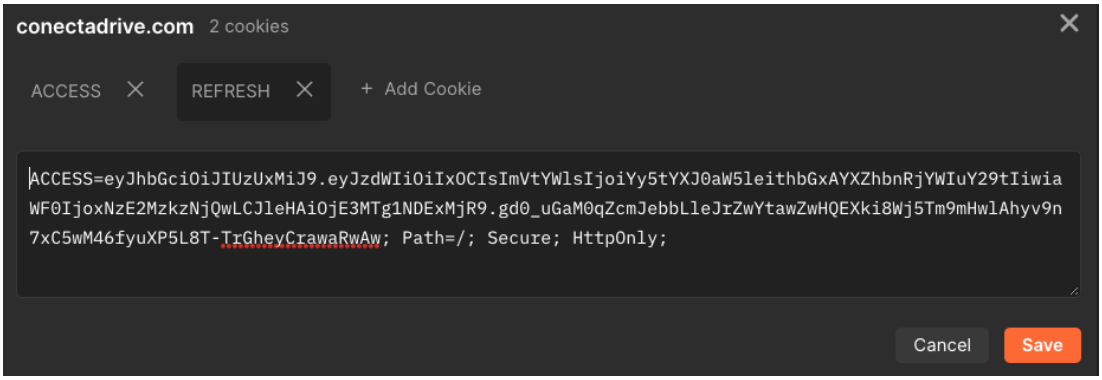

Figura 24 - Validació cookies RNF5

Figura 25 - Validació logging RNF7

8 Conclusions

A l'iniciar aquest projecte, la meva experiència en programació web era absolutament nul·la. La transformació tant a nivell personal com professional, ha estat significativa, i no només m'ha ajudat a adquirir coneixements de programació i del sector del transport, si no que també m'ha fet canviar les meves aspiracions i objectius com a enginyer en el futur.

Ha estat en endinsar-me en un projecte com aquest, on he programat frontend i backend, que m'he adonat que és al backend on em sento més còmode. M'agraden els reptes relacionats amb la escalabilitat i seguretat d'aplicacions, i he pogut aplicar-ho plenament durant el desenvolupament d'aquest projecte. A més, he tingut la oportunitat d'aplicar conceptes de patrons de disseny vistos a l'assignatura de TAP, conceptes de bases de dades apresos a l'assignatura de BD i conceptes de disseny de programari i estructures après a l'assignatura d'ADA. També he aplicat el disseny d'APIs REST que vaig aprendre a l'assignatura de sistemes oberts. En definitiva, aquest projecte ha estat una excel·lent oportunitat per posar en pràctica gran quantitat de conceptes apresos durant el grau i participar en un projecte real.

No només això, sinó que he pogut treballar amb un equip real de programadors, això m'ha fet haver d'adoptar unes bones pràctiques que al final són necessàries per a poder treballar amb altra gent i fer que les coses funcionin en conjunció. Això no involucra només definir regles de linting, o quants espais per a cada tab, sinó també portar un bon control de les versions i branques utilitzant GIT. Constantment he estat revisant les PR d'altres programadors, corregint errors de merge, fent cherry-picks i molts altres conceptes que vaig aprendre a computadors i he pogut aplicar aquí també.

Pel que fa a la part de sistemes, he posat en pràctica la creació de contenidors, xarxes Docker, creació de servidors i configuració de regles Firewall. Sense dubte el descobriment de Docker és essencial per a aquest desenvolupament, ja que tota la nostra arquitectura es basa en ell, des de la base de dades, passant pel frontend i arribant al backend.

A més, vaig encarregar-me de la configuració de Spring Security, implementant un sistema d'autenticació basat en JWT, o també configurant un estàndard universal com és oauth2. Segurament en el futur optaria per una solució gestionada, com pot ser AWS cognito o alguna pròpia d'altres cloud providers, però ha estat una bona oportunitat per veure la complexitat que hi ha darrere un sistema d'autenticació d'usuaris.

El principal objectiu del projecte era permetre a l'empresa gestionar les seves reserves internament, sense dependre de programari extern, garantint la seguretat i la integritat de les dades. Hem aconseguit aquest objectiu, i actualment l'empresa opera el sistema amb eficàcia, amb perspectives de comercialització a futur, pel que puc afirmar amb completa seguretat que aquest projecte ha sobrepassat qualsevol expectativa possible als inicis.

En conclusió, el desenvolupament de Conectadrive ha estat una experiència absolutament enriquidora que m'ha permès evolucionar des d'un coneixement bàsic fins a construir una aplicació empresarial completa. Ha sigut un any de molt treball i bastanta tensió en molts moments, però he gaudit i segueixo gaudint molt de poder seguir en aquest projecte. És per això que tinc molt clar que la tecnologia i els negocis són una combinació perfecta per a aquells que, com jo, busquem reptes constants.

9 Bibliografía

[1] Primevue. Documentació. https://primevue.org.

[2] Quasar. Documentació. https://quasar.dev.

[3] Spring. Queries de Spring Data JPA.https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html.

[4] Spring. Documentació de Spring Security. https://spring.io/projects/spring-security.

[5] Baeldung. Cacheig d'informació amb redis. https://www.baeldung.com/spring-boot-redis-cache.

[6] Github. Repositori de Postgis. https://github.com/postgis/postgis.

[7] Tailwind. Documentació https://tailwindcss.com.

[8] Jetbrains.Home page. https://www.jetbrains.com/idea/.

[9] Visual Studio Code. Home page. https://code.visualstudio.com/.

[10] Oauth. Documentació. https://oauth.net/2/.

[11] Mapbox. Documentació. https://docs.mapbox.com/mapbox-gl-js/guides/.

[12] Netlify. Documentació. https://docs.netlify.com.

[13] DigitalOcean. Documentació. https://www.digitalocean.com/community/tutorials.

[14] Nginx. Tutorial https://nginx.org/en/docs/beginners_guide.html.

[15] JWT IO. Home page. https://jwt.io.

ANNEX

L'entorn de testing desenvolupat es pot trobar a la URL d'abaix, en cas que no sigui funcional, podeu contactar amb mi a través del meu correu o perfil de Linkedin:

- Correu: c.martinezg@estudiants.urv.cat
- Linkedin: https://www.linkedin.com/in/carlos-martínez-garcía-villarrubia-887499215/

URL d'accés a la web:

- https://conectadrive.software/

Si es desitja realitzar una demo o veure les altres pàgines podeu escriure'm per a donar-vos accés amb un usuari de prova i així veure el resultat final.