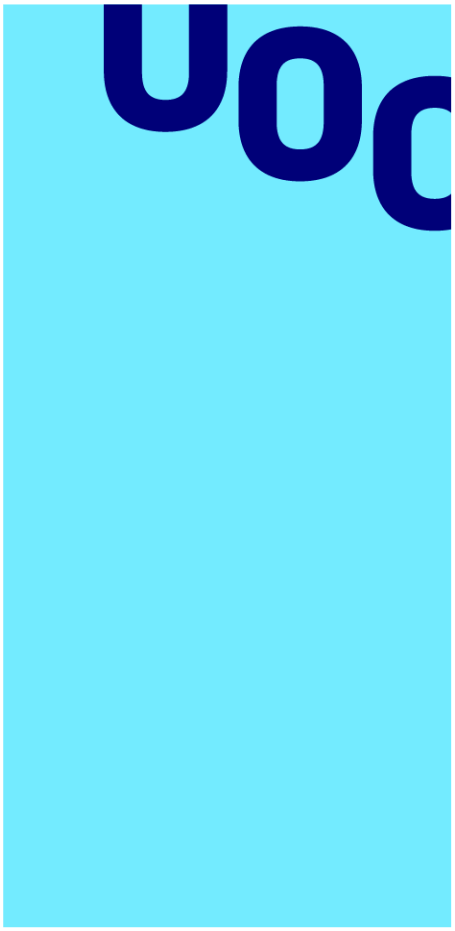
Gestor de cites

eAgenda



Universitat Oberta
de Catalunya

Víctor Simón Aguilera

Grau d'Enginyeria Informàtica
Java EE

Tutor/a de TF
Albert Grau Perisé
Professor/a responsable de l'assignatura
Santi Caballe Llobet

Data Lliurament: 19/06/2024

Llicències alternatives (triar alguna de les següents i substituir la de la pàgina anterior)

A) Creative Commons:

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-SenseObraDerivada 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement-CompartirIgual 3.0 Espanya de Creative Commons

Aquesta obra està subjecta a una llicència de Reconeixement 3.0 Espanya de Creative Commons

B) GNU Free Documentation License (GNU FDL)

C) Copyright

FITXA DEL TREBALL FINAL

Títol del treball:	*Gestor de cites*
Nom de l'autor:	*Víctor Simón Aguilera*
Nom del consultor/a:	*Albert Grau Perisé*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*06/2024*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Java EE*
Idioma del treball:	*Català*
Paraules clau	*Java EE, Spring Boot, ReactJS*

Resum del Treball

Aquest projecte, dins de l'àrea Java EE, del grau d'enginyeria informàtica, consisteix en el desenvolupament d'una prova de concepte d'una aplicació web que permet gestionar el calendari de cites d'una persona o d'una empresa. El projecte ofereix a les persones interessades les dates disponibles per programar una reunió entre dues parts o per a reservar una cita d'un servei. Per consegüent, el que aporta aquest projecte, és evitar la usual llarga cadena de comunicació fins a arribar a coordinar una cita a través de correus electrònics, missatges o trucades de telèfon.

La gestió i desenvolupament del projecte s'ha fet seguint el mètode en cascada. Les etapes per les quals ha passat el projecte han sigut les etapes de definició, d'anàlisi i disseny, la implementació i finalment la memòria i presentació. El projecte ha servit per a posar en pràctica els coneixements adquirits al grau, en especial els adquirits a l'itinerari d'enginyeria del programari.

Les conclusions extretes a destacar són, per una banda, que la corba d'aprenentatge d'un projecte que necessita de frontend, backend i infrastructura requereix molt temps per adquirir el coneixement i entendre les diferents parts i, per una altra banda, que la a gestió d'un projecte amb el mètode de cascada és complicada de complir fil per randa, ja que sorgeixen problemes o requisits enmig de la implementació del projecte que no s'havien tingut en compte a l'anàlisi.

Abstract

This project, within the Java EE area, for the computer engineering degree, consists of developing a proof of concept for a web application that allows managing the appointment calendar of a person or a company. The project offers interested parties the available dates to schedule a meeting between two parties or to book an appointment for a service. Consequently, what this project contributes is avoiding the usual long chain of communication required to coordinate an appointment through emails, messages, or phone calls.

The management and development of the project were carried out following the waterfall method. The stages the project went through were the definition, analysis and design, implementation, and finally the report and presentation. The project served to put into practice the knowledge acquired in the degree, especially those acquired in the software engineering track.

The main conclusions drawn are, on the one hand, that the learning curve for a project that requires frontend, backend, and infrastructure takes a lot of time to acquire the knowledge and understand the different parts; and, on the other hand, that managing a project with the waterfall method is difficult to follow to the letter, as problems or requirements arise during the implementation of the project that were not considered in the analysis.

Índex

Llista de figures

1. Introducció

1.1. Context i justificació del Treball

Avui dia, per a concertar una cita o una reunió entre dues parts, és comú tenir una llarga cadena de missatges per correu electrònic, a través de missatgeria o haver de fer més d'una trucada fins a aconseguir fer una reserva. Per exemple, aquesta situació és habitual per concertar una entrevista de treball, ja sigui presencial, una trucada per telèfon o a través d'una videotrucada. El mateix problema, o molt similar també es dona quan un usuari vol reservar cita a un servei que té disponibilitat limitada, com per exemple a un servei de fisioteràpia, odontòlegs, perruqueria, etc.

En tots els casos presentats, l'usuari del servei s'ha de posar en contacte amb l'empresa o persona amb la qual vol fer la reserva, i entre les dues parts, han de trobar un moment que als dos els vagi bé. Tot i que normalment s'aconsegueix programar una cita, i després dels intercanvis de comunicacions, és possible que la data triada per la reserva no sigui més convenient per alguna de les dues parts, a causa de no poder veure la disponibilitat d'alguna de les dues parts a cap lloc.

Per tal d'evitar aquest problema aquest projecte proposa la creació d'una prova de concepte, basada en una aplicació web que mostrarà la disponibilitat d'una de les dues parts, específicament de la part que ofereix el seu servei. La disponibilitat es mostrarà en un calendari dinàmic, que permet a l'usuari interessat a programar la reunió seleccionant la data que més li convingui entre totes les disponibles. Una vegada es confirma la cita, les dues parts reben un correu electrònic per confirmant que la cita s'ha programat.

La utilització d'aquesta aplicació simplifica la gestió de les cites, ja que el calendari es compartirà en un únic missatge o, fins i tot, pot estar disponible a la pàgina web o al perfil de xarxa social. D'aquesta manera s'evitarà l'allargament en el temps i l'intercanvi de missatges o trucades fins aconseguir programar una cita.

1.2. Objectius del Treball

Els objectius marcats en aquest projecte són els següents:

- Aplicar les habilitats i competències adquirides durant el grau d'enginyeria informàtica i en especial en l'itinerari d'enginyeria del programari.
- Aplicar els coneixements adquirits sobre sistemes distribuïts.
- Desenvolupament d'una aplicació basada en microserveis, seguint una estructura basada en arquitectura hexagonal.
- Aprofundir coneixements de JavaEE i Spring Boot.
- Aprendre a crear aplicacions frontend amb ReactJS.

- Desenvolupament d'una web que mostra el calendari dinàmic per a cada usuari, que serà accessible a través d'un enllaç i on es mostra la disponibilitat de cites.
- Creació d'un panell d'administració per gestionar la disponibilitat, i així crear el calendari dinàmic de cites.
- Aprendre i implementar l'autenticació a través d'OAuth2.
- Aprendre a empaquetar les aplicacions en contenidors amb docker i aprendre a gestionar les dependències locals amb docker compose.
- Aprendre a desplegar els microserveis a la plataforma cloud amazon, AWS.

1.3. Impacte en sostenibilitat, ètic-social i de diversitat

Respecte a l'impacte sobre el medi ambient i sostenibilitat, per una banda, la petjada d'aquest projecte afecta el punt ODS 9 (Industry, innovation and infrastructure). El projecte està desplegat a Amazon Web Services i és àmpliament conegut que els centre de dades necessiten molta energia i recursos per ser construïts i per mantenir-los en funcionament. Per tant, aquest servei es podria dir que té un petit impacte negatiu sobre el medi ambient.

Per una altra banda, l'objectiu d'aquest projecte és que s'hagi de fer menys ús d'altres plataformes (correu electrònic, missatgeria, telèfon...) i, en conseqüència, aquesta petjada es veuria compensada fent menys ús de les altres plataformes.

Respecte a l'impacte ético-social, i de diversitat, l'impacte principal està relacionat amb el marc legislatiu (ODS 16 - Peace, justice and strong institutions) debut a què aquest projecte ha de complir les lleis de privacitat i seguretat per a mantenir de les dades personals dels usuaris.

1.4. Enfocament i mètode seguit

L'estratègia per la qual s'ha optat en aquest treball ha sigut la creació d'un producte nou, desenvolupant una prova de concepte d'una aplicació web que servirà per crear un calendari de cites dinàmic, que permet gestionar les cites d'una manera senzilla i eficient. Seguint l'esquema proposat d'avaluació contínua de l'assignatura, per la gestió i desenvolupament del projecte s'ha utilitzat el mètode de cascada.

Tot i que en el món laboral cada dia s'utilitzen més metodologies àgils com SCRUM, l'ús del mètode de cascada té també els seus avantatges i té tot el sentit continuar aplicant-ho per a petits projectes tancats o proves de conceptes, com en el cas d'aquest projecte. No obstant això, en cas que el producte es continuï desenvolupant, seria convenient canviar cap a metodologies àgils i fer les millores del producte per iteracions.

Les diferents etapes en les quals s'ha dividit el projecte, com es veurà en detall a l'apartat de planificació, han sigut les següents:

- Pla de treball
- Anàlisi i disseny
- Implementació
- Memòria i presentació

Tot i que no està especificat en cap lloc, paral·lelament a l'etapa del pla de treball, l'etapa d'anàlisi i disseny i a l'etapa d'implementació, m'he format en les tecnologies amb les quals no tenia experiència o que necessitava més coneixement.

A més, per ajudar a fer un seguiment de les tasques que s'havien de desenvolupar durant la fase d'implementació, he fet ús de Trello i, per a tenir control dels canvis del codi, he fet servir git amb repositoris de codi a Github.

1.5. Planificació del Treball

Per a la planificació del projecte s'ha seguit la planificació proporciona al diagrama de Gantt de la Figura 1. Es pot observar que la planificació gira entorn la pauta proporcionada per les entregues de les PAC i al voltant de les diferents les diferents etapes exposades al punt anterior. Es pot observar que la part d'implementació és la part més important en termes de temps i esforç, destacant les següents fases:

1. Configuració de les eines
2. Implementació dels microserveis backend
3. Implementació del frontend
4. Fase de proves

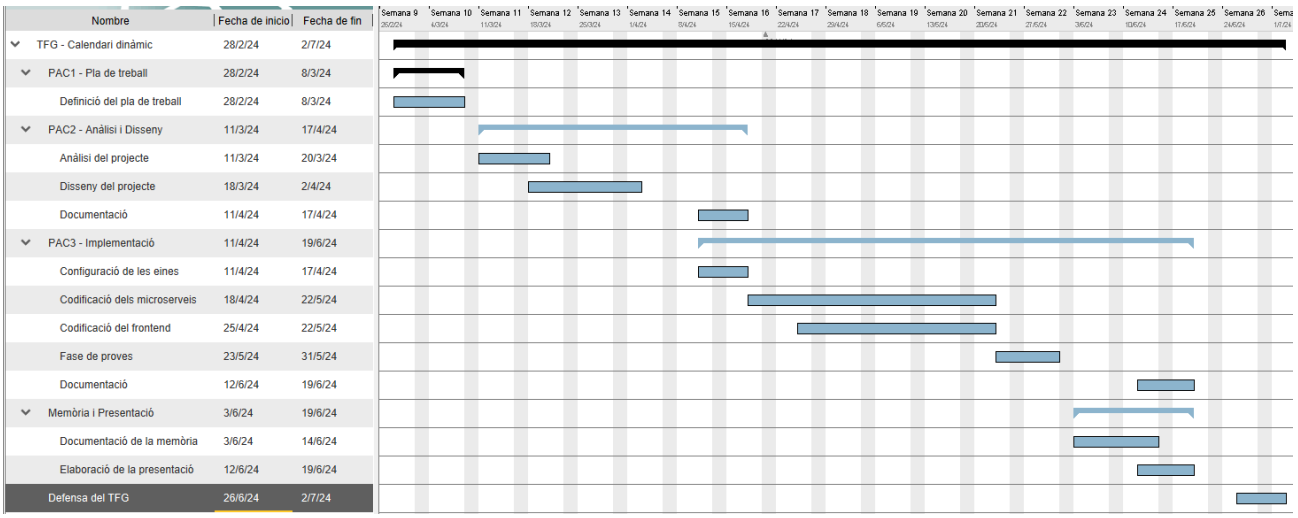


Figura 1: Planificació temporal

Per l'autoorganització de les tasques s'ha fet servir l'eina Trello durant totes les fases d'implementació i també durant la confecció de la memòria i presentació. L'esquema que s'ha seguit es mostra a la Figura 2. Es pot observar que tauler és un tauler bàsic amb tres columnes que mostren l'estat de les tasques.

A la primera columna tenim les tasques pendent de fer (TODO), a la segona columna les tasques amb què s'està avançant (DOING) i finalment a la tercera columna, les tasques que ja estan acabades (DONE). En la Figura 2, concretament es poden veure les tasques de memòria i presentació.

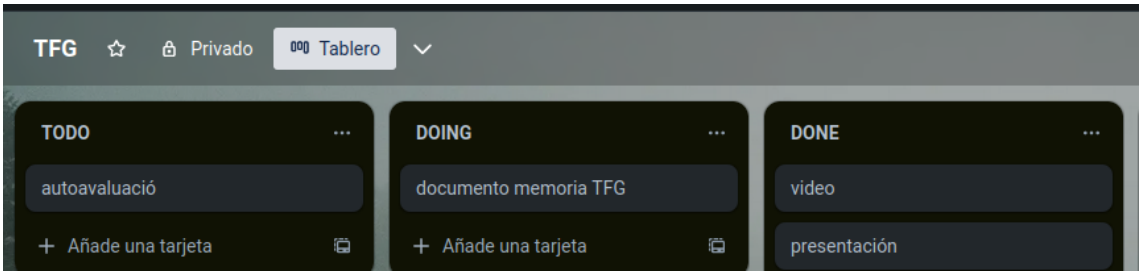


Figura 2: Planificació amb trello

1.6. Breu sumari de productes obtinguts

Els productes obtinguts en el desenvolupament del projecte, han sigut tres microserveis backend i dues aplicacions frontend, és a dir, s'han obtingut un total de cinc productes.

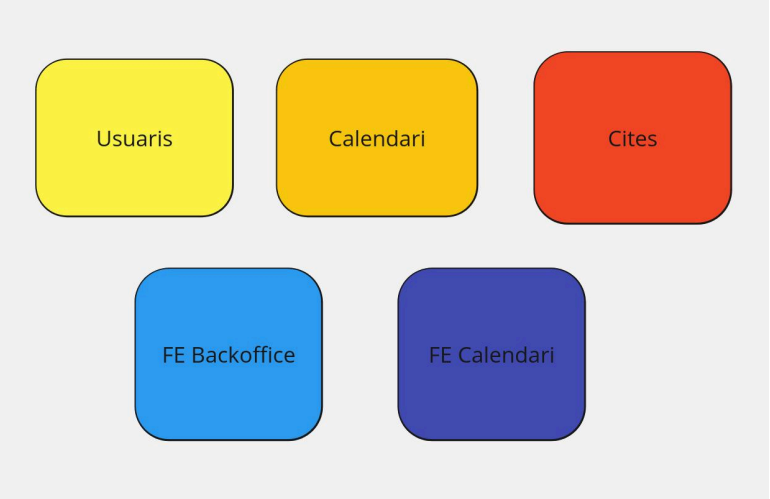


Figura 3: Resum dels productes

Com es mostra a la Figura 3, per la part dels microserveis backend tenim el microservei de gestió d'usuaris, el de gestió del calendari i el de gestió de les cites.

Per la part de les aplicacions frontend, s'han dividit les funcionalitats en les de la part pública, on es mostren els calendaris dinàmics de disponibilitat que

serviran per a concertar les cites, i la part l'administració dels calendaris, on els usuaris i empreses gestionaran la disponibilitat que volen mostrar als usuaris.

Els repositoris privats del codi són els següents:

- https://github.com/VktorSimon/eagenda-users
- https://github.com/VktorSimon/eagenda-calendar
- https://github.com/VktorSimon/eagenda-bookings
- https://github.com/VktorSimon/egenda-website
- https://github.com/VktorSimon/eagenda-backoffice-fe

1.7. Breu descripció dels altres capítols de la memòria

Els capítols en què s'ha desgranat la memòria són "Anàlisi i disseny", "Implementació" i "Conclusions".

A l'apartat d'anàlisi i disseny es mostren els detalls i requisits que es van recollir a l'inici del projecte, després de la fase de definició del pla del projecte. Aquesta secció s'ha dividit en els següents subcapítols:

- **Actors**: En aquest subapartat es mostren els diferents rols que faran ús de l'aplicació.
- **Casos d'ús - Visió genera**l: Una vegada definits els rols, es mostra una visió general de les diferents accions que tindrà disponible cada actor.
- **Casos d'ús - Fitxes**: En aquest subapartat es fa una definició detallada dels requisits i funcionalitats de cadascun dels casos d'ús.
- **Prototips**: En aquest subapartat es mostren prototips de les diferents pantalles requerides que es van detectar a la fase d'anàlisi.
- **Model de classes**: En aquest subapartat es mostra el model de classes proposat per l'aplicació durant la fase d'anàlisi.
- **Model de base de dades**: A partir de les classes definides en el subapartat anterior, es va fer el modelatge requerit per la base de dades.
- **Arquitectura**: En aquest subapartat es defineix els requisits d'arquitectura plantejats per l'aplicació.

A l'apartat d'implementació es mostra les diferents eines utilitzades durant la fase d'implementació i també es mostren els detalls més importants de l'aplicació una vegada acabada la implementació. En concret, aquesta secció s'ha dividit en els següents subcapítols:

- **Eines i frameworks**: Recopilació de les diferents eines utilitzades.
- **Revisió de les funcionalitats principals**: En aquest subapartat es mostra les principals funcionalitats per a cadascun dels productes obtinguts.
- **Arquitectura a AWS**: Un breu resum de l'arquitectura de l'aplicació a desplegada a Amazon Web Services.

A l'apartat de conclusions es mostra les conclusions extretes una vegada finalitzat el projecte i també es mostra una visió del futur de l'aplicació. Específicament, aquesta secció s'ha dividit en els següents subcapítols:

- **Conclusions**: Explicació de les conclusions de la gestió i implementació del projecte.
- **Funcionalitats que falten**: Breu resum de les funcionalitats que s'han quedat fora de l'abast del projecte i que s'havien definit a la part d'anàlisi i disseny.
- **Evolucions de l'aplicació**: Explicació de les següents passes a desenvolupar en cas de continuar desenvolupant el projecte.

També s'ha inclòs un apartat d'annexos, per afegir documentació sobre el projecte que no tenen cabuda en la resta d'apartats. Els subapartats en què està dividit són:

- **Instal·lació de l'aplicació**: En aquest subapartat es mostra les diferents passes per posar en marxa l'aplicació.
- **Github Actions**: Es mostra un exemple d'execució de la *pipeline* amb Github Actions
- **AWS**: Es mostren els diferents component de l'aplicació a la plataforma Amazon Web Services
- **Cloudflare**: Es mostra una part de la configuració del domini a Cloudflare.

2. Anàlisi i disseny

En aquest apartat es mostra l'anàlisi del projecte tant a escala funcional com nivell d'arquitectura. Es defineixen les funcionalitats que tindrà l'aplicació, desgranant les funcionalitats i agrupant-les per l'àmbit on apliquen i també es defineix fins on arribarà la implementació del projecte.

En primer lloc, en la part funcional, es mostren els actors implicats al sistema, els casos d'ús en què està implicat cada actor i s'especifiquen en detall de cadascun d'aquests casos d'ús. Seguidament, es presenten les diferents pantalles d'usuari que tindrà l'aplicació, per a tenir un exemple visual de les especificacions dels casos d'ús.

En segon lloc, en la part tècnica, es mostra el disseny de les classes principals del projecte, el disseny de la base de dades i una la definició de l'arquitectura del projecte.

2.1 Actors del projecte

Hi ha dos actors que hem de tenir en compte per aquest projecte. Per una banda, tenim la persona o empresa que vol oferir el calendari i, per una altra banda, la persona interessada a fer una reserva. A la Figura 4 podem veure una representació visual dels actors.

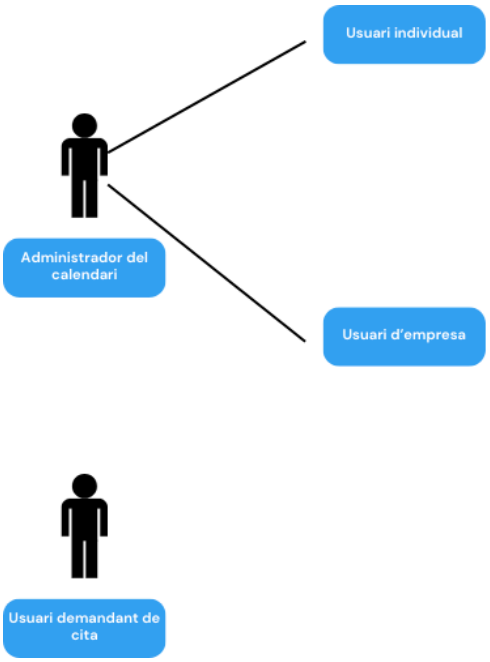


Figura 4: Actors del projecte

- Usuari administrador del calendari: És l'usuari que configura i comparteix la disponibilitat del seu calendari per tal d'oferir un servei. Aquest d'usuari es divideix en dos tipus diferents, un usuari individual que representarà a ell mateix o un usuari que representa una empresa.

- Usuari client: És l'usuari interessat a fer una reserva per tal d'obtenir el servei associat en la data seleccionada per ell.

2.2 Casos d'ús - Visió general

Una vegada definits els actors del projecte, veiem els diferents casos d'ús que requereix cadascun dels rols per a fer ús de l'aplicació. Per l'usuari administrador del calendari, s'especifiquen els casos d'ús d'autenticació, d'administració del calendari i d'administració de les reserves. Per l'usuari que demana la cita, s'especifiquen els casos d'ús relacionats amb la reserva com a client.

Com que els dos usuaris administradors del calendari tindran les mateixes funcionalitats, en el diagrama dels casos d'ús es mostra un usuari administrador genèric, que representa tant l'usuari d'empresa com el que es representa a ell mateix. A la Figura 5 es mostren els casos d'ús del projecte, segregats per rol.

Figura 5: Casos d'ús

2.3 Casos d'ús - Fitxes

En aquesta secció es presenten els detalls, requisits i funcionalitats dels casos d'ús presentats en la Figura 5. Per tal de facilitar el seguiment de la figura anterior i veure fàcilment la relació amb les fitxes dels casos d'ús s'ha seguit el mateix ordre, de dalt a baix.

Nom	Autenticació d'usuari
Actor	*Administrador del calendari*
Descripció	L'usuari accedeix a la pantalla d'autenticació de l'aplicació i es podrà autenticar o registrar a través del botó corresponent al correu electrònic a través de la implementació amb OAuth2
Postcondició	Si l'usuari no existia en el sistema,

	s'haurà creat
Alternativa	L'usuari es pot donar de baixa en el sistema.

Nom	Configuració de les reserves
Actor	*Administrador del calendari*
Descripció	L'usuari accedirà a la pantalla de configuració de les reserves i veurà un formulari on podrà indicar la configuració general del seu calendari. Indicarà la zona horària, el temps de durada per reserva, el màxim temps seguit que un usuari pot reservar, el nombre de cites concurrents per diferents usuaris i la descripció del correu electrònic de confirmació que rebrà l'usuari. Opcionalment, també es podrà configurar la inclusió d'un enllaç per a fer videotrucades
Precondició	L'usuari ha d'estar autenticat.
Postcondició	Si és la primera vegada que es fa la configuració, es desbloqueja la configuració de disponibilitat.

Nom	Configuració de la disponibilitat
Actor	*Administrador del calendari*
Descripció	L'usuari accedirà a la pantalla de configuració de disponibilitat. Podrà indicar la disponibilitat recurrent, manualment i bloquejar dies que no estarà disponible.
Precondició	L'usuari ha d'estar autenticat.
Postcondició	Si és la primera vegada que es configura el calendari, es mostrarà l'enllaç per a compartir la seva disponibilitat i que els interessats puguin fer reserves.

Nom	Visualització de les reserves
Actor	*Administrador del calendari*
Descripció	L'usuari accedirà a la pantalla de reserves i veurà el calendari amb totes les reserves fetes. També tindrà l'opció d'exportar el calendari.
Precondició	L'usuari ha d'estar autenticat.
Alternativa	L'usuari podrà cancel·lar una reserva o enviar una proposta de canvi de dates.

Nom	Fer una reserva
Actor	*Usuari demandant de cita*
Descripció	L'usuari accedirà al calendari de disponibilitat a través d'un enllaç que haurà rebut. Seleccionarà el dia i hora que més li convingui entre totes les disponibles i farà clic al botó de confirmació.
Postcondició	Tant l'usuari demandant de cita com l'usuari administrador del calendari, rebran un correu electrònic amb la confirmació de la reserva. L'interval de temps seleccionat deixarà d'estar disponible si no hi ha més disponibilitat.

Nom	Gestió d'una reserva
Actor	*Usuari demandant de cita*
Descripció	L'usuari podrà cancel·lar o modificar la data d'una reserva a través d'un enllaç a la descripció de l'esdeveniment al calendari.
Postcondició	Tant l'usuari demandant de cita com l'usuari administrador del calendari, rebran un correu electrònic amb el

	canvi de la reserva. L'interval de temps canviat seleccionat tornarà a estar disponible i si hi ha nova reserva, el nou interval deixarà d'estar disponible si no hi ha més disponibilitat.

2.4 Prototips

Ara veiem una representació visual de les diferents pantalles i apartats del projecte. Les figures següents mostren els esbossos de com s'havien especificat les diferents pantalles que es mostren als usuaris, associades als casos d'ús especificats en l'apartat anterior. En el disseny final aquestes pantalles varien, fent ús dels components visuals disponibles. Comencem mostrant el panell d'administració, on l'usuari administrador configura el seu compte i disponibilitat.

Pantalla de configuració del calendari

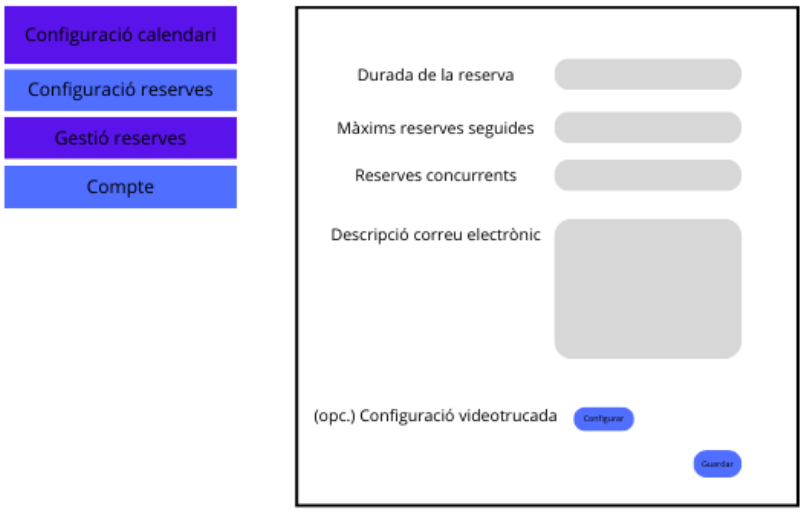


Figura 6: Pantalla de configuració del calendari

Pantalla de configuració de la disponibilitat

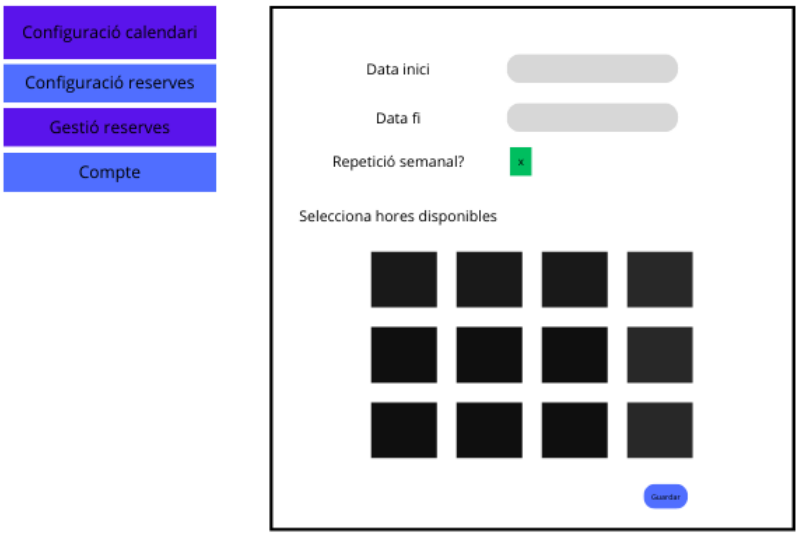


Figura 7: Pantalla de configuració de la disponibilitat

Pantalla de visualització de les reserves

Figura 8: Pantalla de visualització de les reserves

I a continuació es mostra l'esboç de la pantalla que veurà el client

Pantalla de reserva de cita

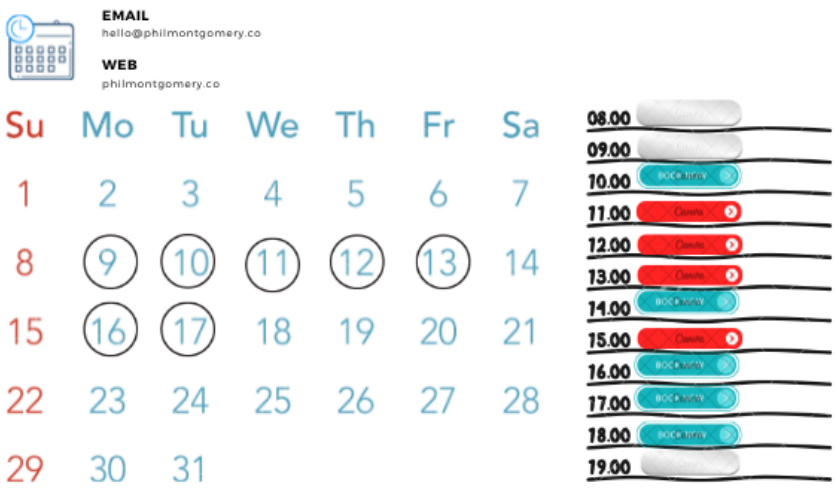


Figura 9: Pantalla de reserva de cita

2.5 Model de classes

En el model de classes de la Figura 10, podem veure la representació de l'usuari administrador d'un calendari que té associat un calendari de disponibilitat. Aquest calendari té una configuració de calendari i té associats uns intervals de disponibilitat concrets. Finalment, la reserva estarà associada a un usuari i a un interval de dates concret, també es guardarà l'email de l'usuari que ha fet la reserva.

Tot i que a l'hora de representar la vista es mostrarà aquest model de dades, la disponibilitat del calendari estarà representada internament per tres elements: la definició de disponibilitat de dates recurrent, la disponibilitat de dates manual i el bloqueig de dies específics (per exemple, festius o vacances).

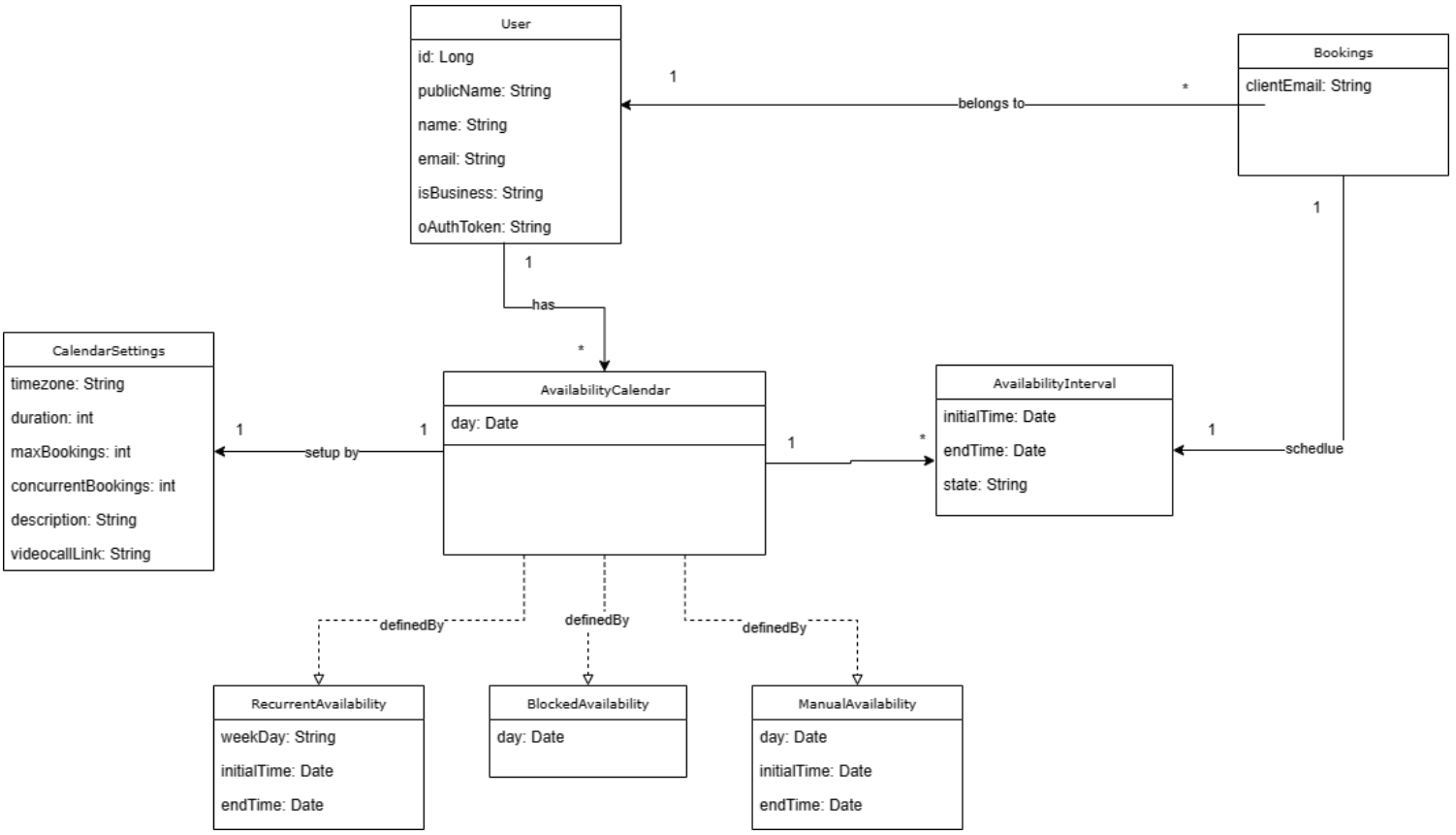


Figura 10: Model de classes

2.6 Model de base de dades

La representació del model de base de dades es pot veure a la Figura 11. Es pot observar que per a representar el model de dades anterior a la base de dades, s'ha omès la classe `AvailabilityCalendar`. Aquest model és una simplificació del model de classes. Tenim doncs les taules d'usuari, la de configuració del calendari, la de definició de dades disponibles o bloquejades i la de les reserves.

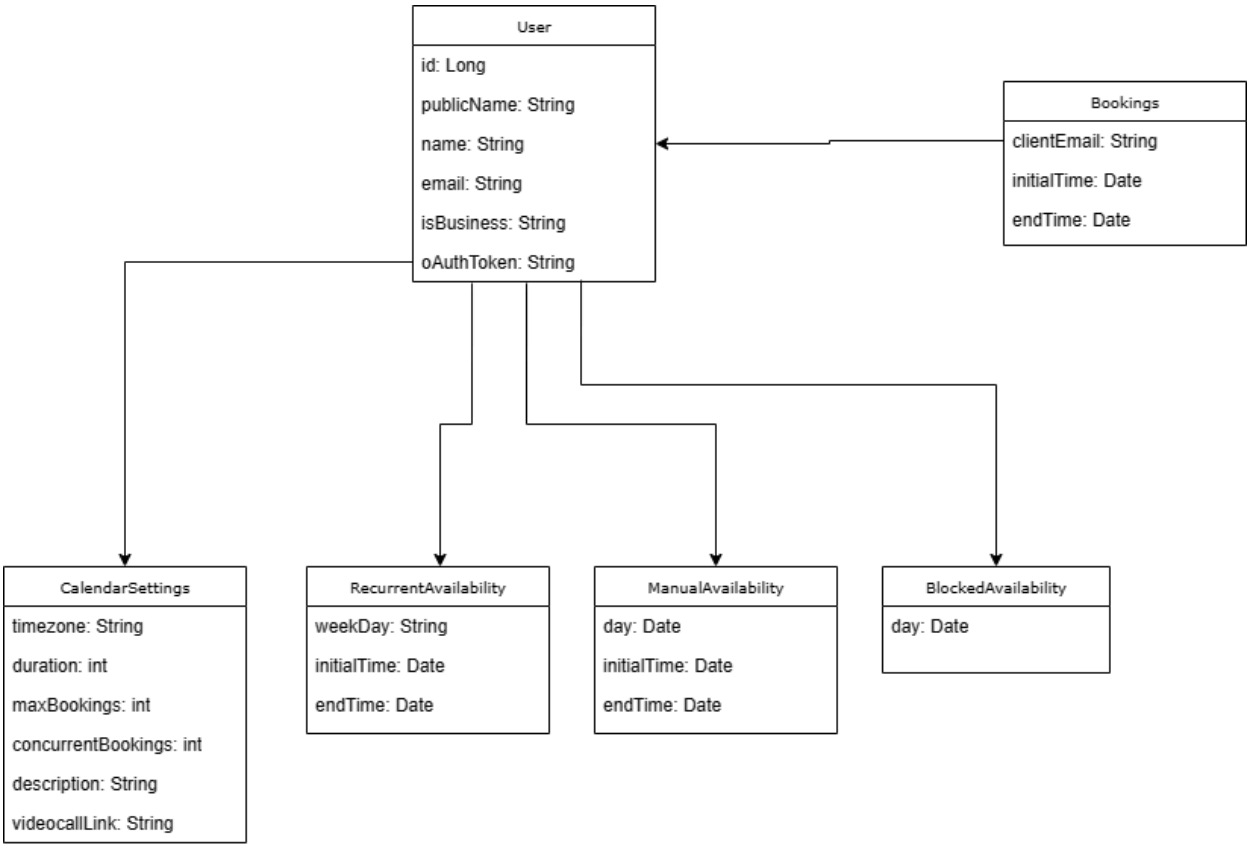


Figura 11: Model de base de dades

2.7 Arquitectura

El projecte, com s'ha comentat en els apartats anteriors està compost per tres microserveis backend: el de gestió d'usuaris, el de gestió del calendari i el de gestió de les cites. Les tecnologies principals per desenvolupar els microserveis són Java EE i Spring Boot. Els microserveis es comuniquen amb la base de dades relacional PostgreSQL.

Per a mostrar les interfícies d'usuari tenim dos productes, un per la web pública i un altre pel panell d'administració. La tecnologia principal emprada és ReactJS. Aquestes aplicacions es comuniquen amb els microserveis per a fer ús de les seves funcionalitats i per a obtenir dades a través de peticions HTTP amb la llibreria axios.

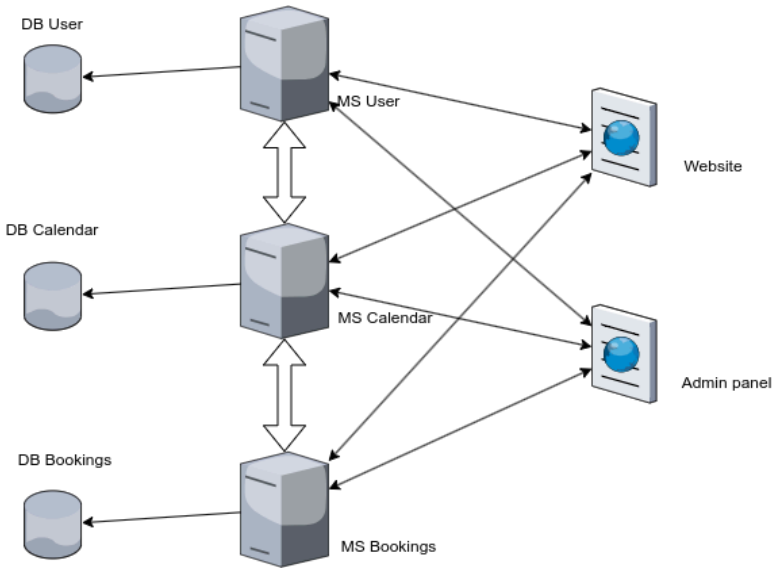


Figura 12: Arquitectura aplicació

Els microserveis de backend, ofereixen les seves dades i funcionalitats en una API REST i la documentació de la API REST està documentada amb swagger. L'estructura utilitzada dins dels microserveis està basada en l'arquitectura hexagonal. D'aquesta manera els microserveis estan dividits en tres capes.

La capa de domini: En aquesta capa trobarem els models de l'aplicació i la lògica principal.
La capa d'aplicació: En aquesta capa trobarem els casos d'ús de l'aplicació.
La capa d'infraestructura: En aquesta capa trobarem els elements que actuen com a punt d'entrada o sortida de la nostra aplicació, en el nostre cas els controladors d'una API Rest, les bases de dades i els enviaments d'emails.

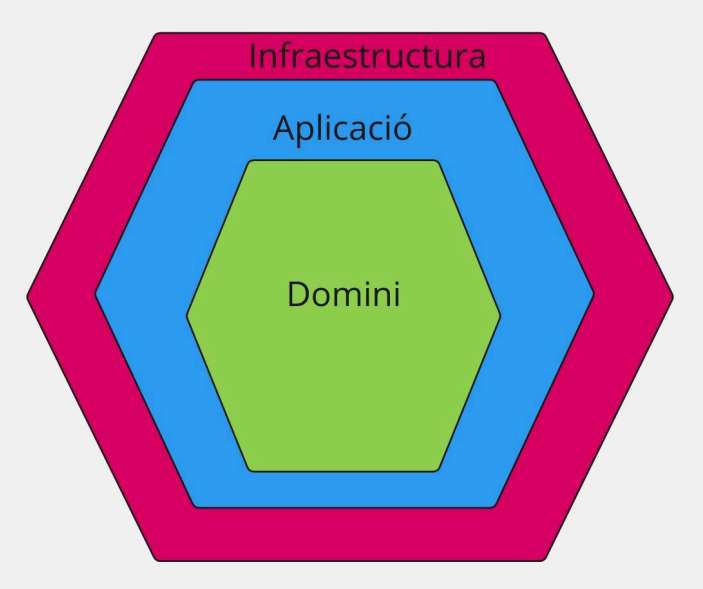


Figura 13: Arquitectura hexagonal

En la implementació del projecte, pel backend s'ha fet servir el framework Spring Boot, juntament amb les llibreries Spring Data JPA, Spring Security, Spring Actuator, Lombok, Hibernate, Spring Starter Mail. També s'ha fet servir JUnit i Mockito per a fer els tests dels projectes.

A més a més, durant la fase de desenvolupament del projecte, també s'han configurat les Github Actions per a tenir integració contínua. Gràcies a la integració continua, es verifica que no hi ha regressió de funcionalitats a través dels tests.

A la fase final del projecte, també s'ha afegit la configuració necessària per desplegar l'aplicació a Amazon Web Service, utilitzant la capa gratuïta proporcionada per aquest servei.

3. Implementació

En aquest apartat es mostren les diferents eines que s'han utilitzat durant la implementació del projecte, es veuran les funcionalitats principals per a cada part del projecte i també es comenta breument l'arquitectura de l'aplicació a desplegada a AWS.

3.1 Eines i frameworks

S'ha fet servir una gran varietat d'eines, tecnologies, frameworks i llibreries per la implementació del projecte. Les dividirem en quatre seccions, les utilitzades a la part del backend, les utilitzades pel frontend, les utilitzades al núvol i eines transversals o suplementàries.

Backend:

- **Jakarta EE** (anteriorment Java EE): És la tecnologia principal que s'utilitzarà al projecte. S'usarà en la creació dels microserveis i és on residirà la gran part de la lògica del projecte. Es denomina com Jakarta EE a les especificacions que estenen Java pel context professional.
- **Spring Boot**: És un dels frameworks més trajectòria i un dels més emprats que proporciona les eines més comunes pel desenvolupament de microserveis amb Java. A més a més, s'han utilitzat les següents llibreries:
    - Spring Data JPA
    - Spring Security
    - Spring Actuator
    - Lombok
    - Swagger
    - JUnit
    - Mockito
- **Gradle**: Sistema d'automatització per a la compilació del projecte.
- **PostgreSQL**: Sistema gestor de base de dades, on es guardaran les dades del projecte, com la configuració dels calendaris i les reserves.
- **IntelliJ**: Entorn de desenvolupament per a aplicacions Java.
- **Docker compose**: S'ha fet servir per a gestionar les dependències d'infraestructura dels microserveis.

Frontend:

- **React**: Framework de JavaScript que es farà servir per a la part de visualització del projecte. L'avantatge d'aquest framework és la creació de components reutilitzables que reaccionen als diferents esdeveniments generats pels usuaris. A més a més, s'han utilitzat les següents llibreries:
    - Axios
    - Material UI
- **Visual Studio Code:** IDE que s'ha utilitzat pel desenvolupament de les aplicacions front-end.

Núvol:

- **AWS:** En la fase final del projecte, s'ha desplegat el projecte en l'entorn d'Amazon Web Services, utilitzant la capa gratuïta. Els serveis específics que s'han fet servir són:
    - EC2
    - S3
    - RDS
    - Api Gateway
- **Cloudflare**: S'ha configurat el domini eagenda.live amb cloudflare i, per tant, també les redireccions del domini cap a AWS.

Transversals:

- **Github**: Eina per tenir un control de canvis i de versions del projecte. També s'ha fet servir les Github Actions per tenir integració contínua al projecte.
- **Docker**: Permetrà la creació de contenidors dels diferents microserveis i de la capa de presentació.
- **Trello**: Eina per organitzar les tasques pendents, les fetes i les que estaven en progrés.

3.2 Revisió de les funcionalitats principals

Comencem revisant el backends, on les funcionalitats estan més segregades i finalitzarem amb els frontends que tenen una funcionalitat més transversal.

3.2.1 Microservei User

eagenda-users

La funcionalitat principal d'aquest microservei és el registre i login dels usuaris amb oAuth2. En aquesta primera versió només s'ha integrat el login amb google, però si es continuarà avançant amb el desenvolupament de l'aplicació es poden afegir altres proveïdors.

A part del registre i login d'usuaris, aquest microservei també conté funcionalitats per a obtenir i actualitzar informació dels usuaris. A la Figura 14 es poden veure els endpoints disponibles a aquest microservei.

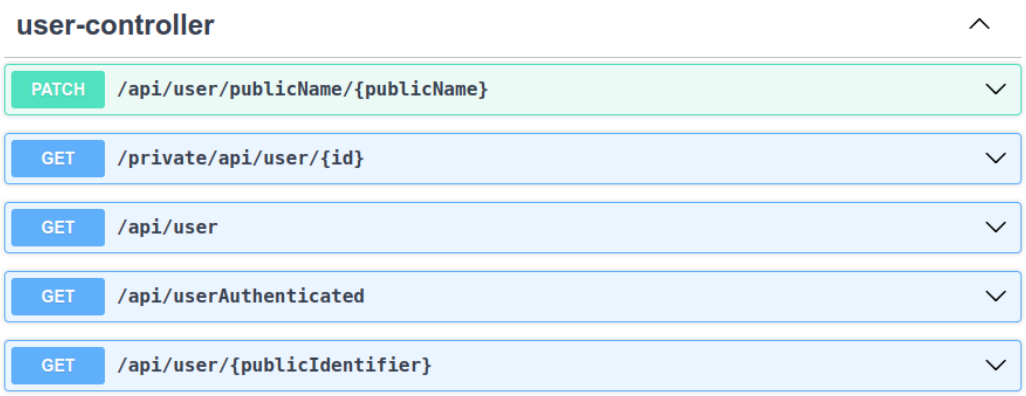


Figura 14: Endpoints microservei usuari

3.2.2 Microservei Calendar

eagenda-calendar

Les funcionalitats principals d'aquest microservei tenen a veure amb la configuració general del calendari i la configuració de disponibilitat dels usuaris administradors. També es proveeixen endpoints per a obtenir la informació del calendari, una vegada que està guardada a la base de dades. Aquest microservei es comunica amb els altres microserveis per a recuperar dades d'usuari i de reserves. A la Figura 15 es poden veure els endpoints disponibles a aquest microservei.

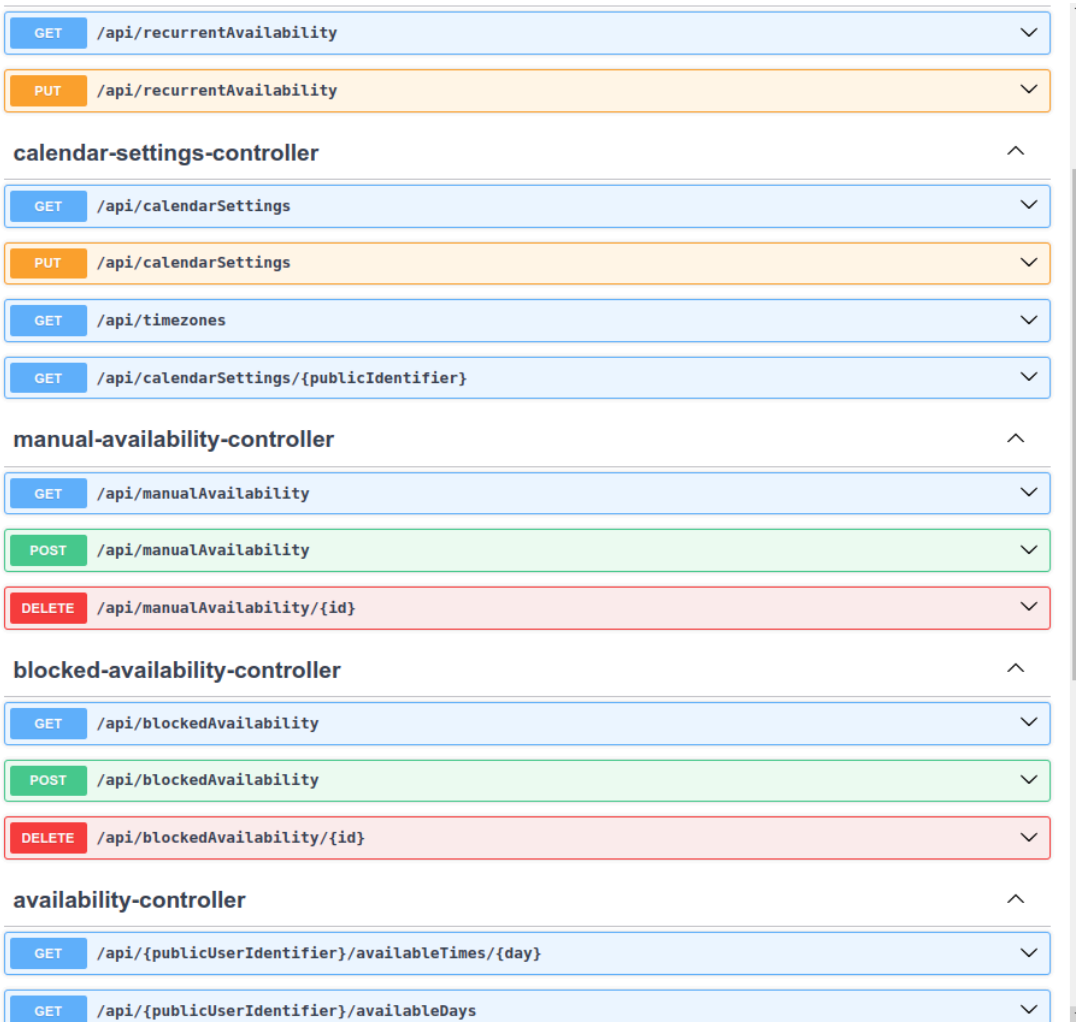


Figura 15: Endpoints microservei calendar

3.2.3 Microservei Bookings

[eagenda-bookings]

La funcionalitat principal d'aquest microservei és la gestió de les reserves, donats un usuari i la seva disponibilitat. Degut a aquest motiu, aquest microservei es comunica amb els altres dos per a obtenir algunes dades. A la Figura 16 es poden veure els endpoints disponibles a aquest microservei.

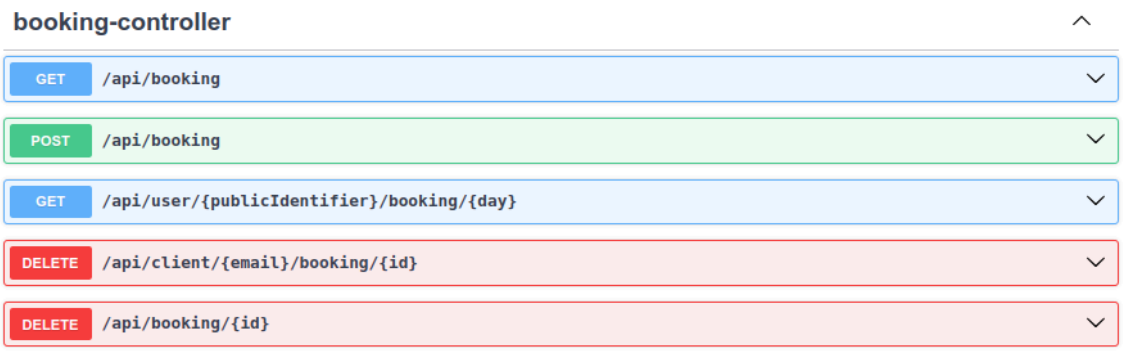


Figura 16: Endpoints microservei bookings

3.2.4 Panell d'administració

[eagenda-backoffice-fe]

Una vegada registrat i loguejat, la part més important del panell d'administració, és la configuració de disponibilitat recurrent:

Figura 17: Configuració de disponibilitat

Amb aquesta configuració, tot i que és ideal revisar i configurar tota la resta de configuracions, ja tindrem disponible el calendari de reserves. Anem al perfil per a veure el link del calendari:

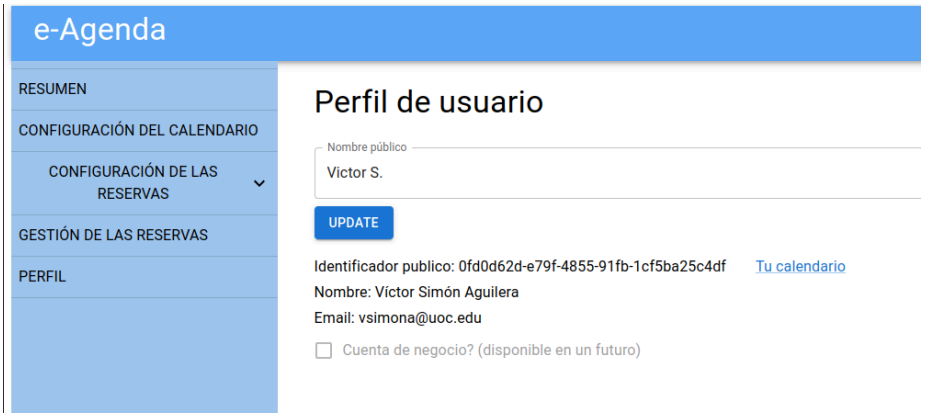


Figura 18: Configuració de perfil d'usuari

Des de la gestió de reserves podem veure les reserves que ens han fet:

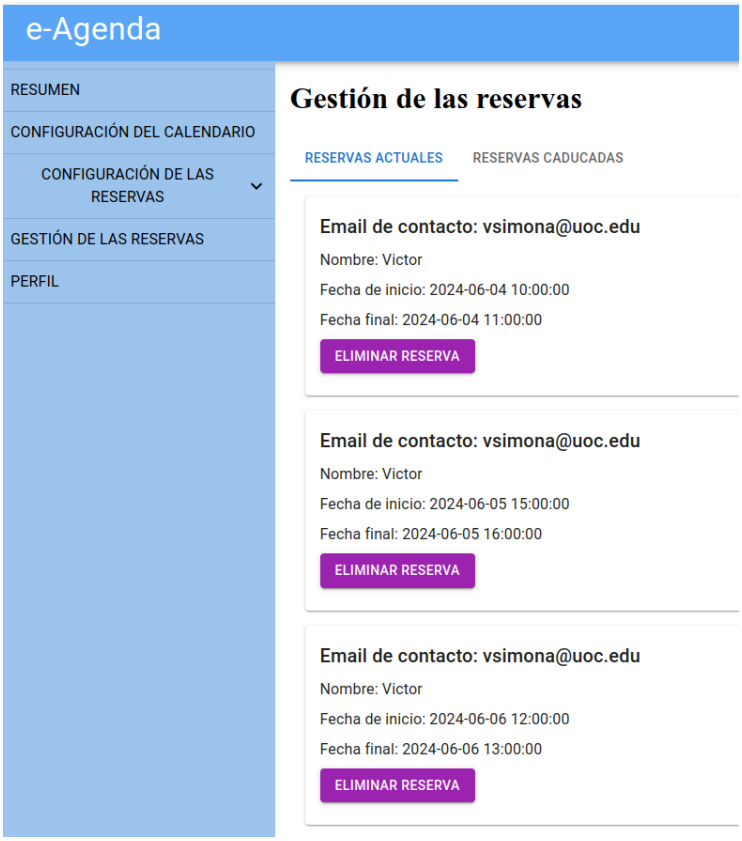


Figura 19: Gestió de les reserves

Hi ha altres funcionalitats al panell d'administració, però les mostrades són les funcionalitats principals.

3.2.5 Pàgina web

egenda-website

La funcionalitat principal de la web és mostrar la disponibilitat del calendari generat i poder fer les reserves.

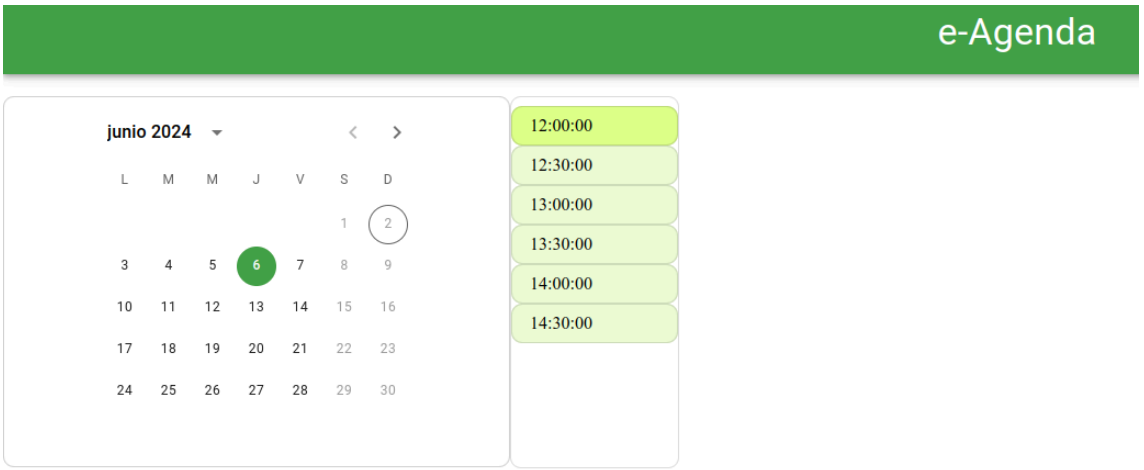


Figura 20: Calendari dinàmic

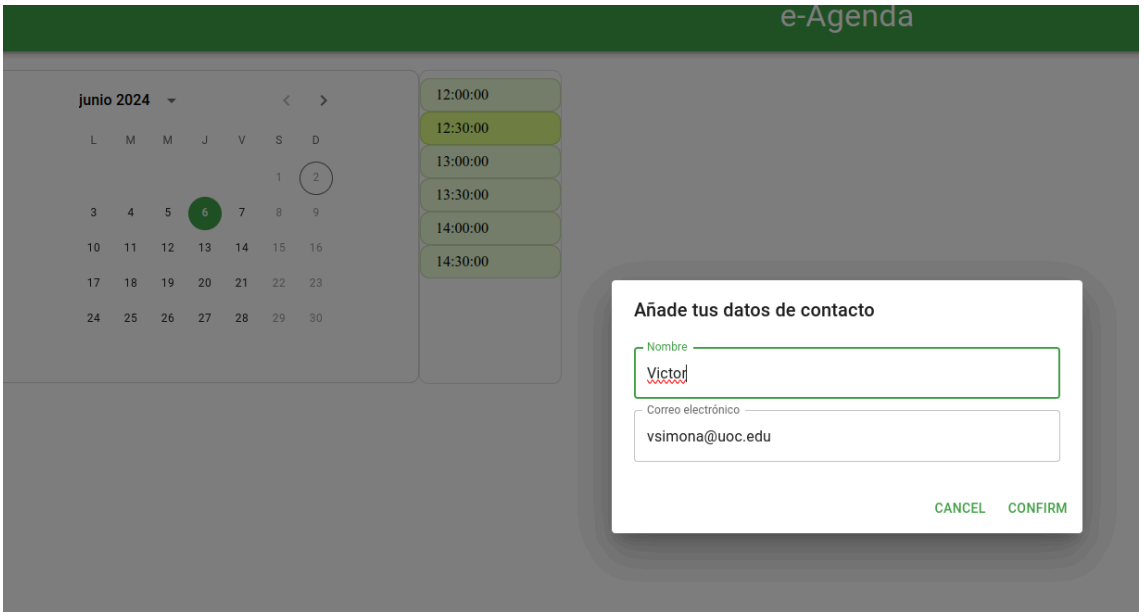


Figura 21: Reserva de cita

3.4 Arquitectura AWS

El desplegament a la plataforma d'Amazon, Amazon Web Services s'ha fet seguint l'esquema de la Figura 22.

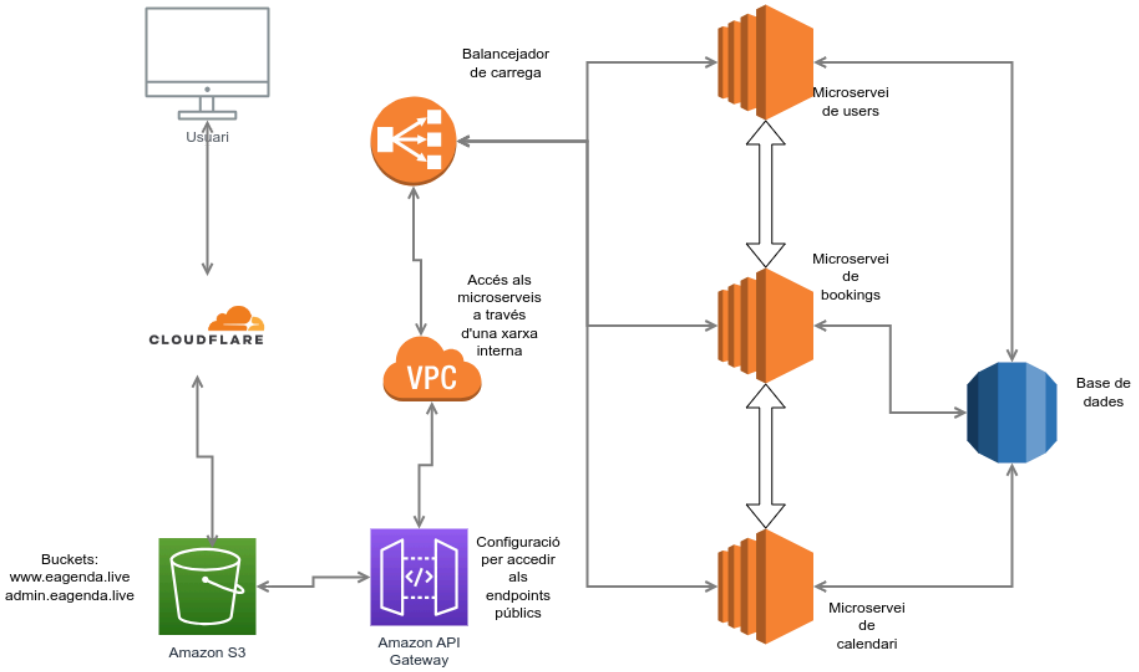


Figura 22: Arquitectura AWS

Les aplicacions frontend estan desplegades al servei S3, un bucket per a cada aplicació. Des de Cloudflare, es redirigeix al bucket corresponent depenent del subdomini. El subdomini `www` redirigeix a la pàgina web pública i el subdomini `admin` redireix al panell d'administració. Les aplicacions frontend, per a fer les peticions als microserveix ho fan a través de l'API Gateway. A l'API Gateway només estan disponibles els recursos públics. L'API Gateway fa les peticons als microserveis passant per una xarxa privada (VPC) i per un balancejador de càrrega (ELBLes imatges de docker es poden trobar a dockerhub: https://hub.docker.com/u/vsimonauoc).

Els microserveis backend estan desplegats a EC2 i es comuniquen entre ells i amb la base de dades. El desplegament d'aquesta primera versió s'ha fet de forma manual, utilitzant les imatges els contenidors de Docker en les instàncies EC2 i pujant el codi generat pels frontals a S3.

Les imatges de docker es poden trobar a dockerhub (https://hub.docker.com/u/vsimonauoc). En futures versions els desplegaments s'haurien de fer idealment amb Github Actions.

4. Conclusions i treballs futurs

4.1 Conclusions

Durant la implementació del projecte s'ha intentat seguir la planificació que es va plantejar durant el pla del projecte, però han sorgit problemes per arribar a implementar totes les funcionalitats que es volien fer, detallades a l'anàlisi del projecte. L'estimació de temps no havia sigut la correcta, pensant que algunes parts serien més senzilles i no s'havia considerat suficient temps per l'autoformació. Tot i això, considero que els resultats són els esperats respecte als objectius, ja que les funcionalitats que s'han quedat fora no són les essencials del projecte.

Repassem els objectius i veiem que s'han complert tots:

- Aplicar les habilitats i competències adquirides durant el grau d'enginyeria informàtica i en especial en l'itinerari d'enginyeria del programari. ✔
- Aplicar els coneixements adquirits sobre sistemes distribuïts. ✔
- Desenvolupament d'una aplicació basada en microserveis, seguint una estructura basada en arquitectura hexagonal. ✔
- Aprofundir coneixements de Java EE i Spring Boot. ✔
- Aprendre a crear aplicacions frontend amb ReactJS. ✔
- Desenvolupament d'una web que mostra el calendari dinàmic per a cada usuari, que serà accessible a través d'un enllaç i on es mostra la disponibilitat de cites. ✔
- Creació d'un panell d'administració per gestionar la disponibilitat, i així crear el calendari dinàmic de cites. ✔
- Aprendre i implementar l'autenticació a través d'OAuth2. ✔
- Aprendre a empaquetar les aplicacions en contenidors amb docker i aprendre a gestionar les dependències local amb docker compose. ✔
- Aprendre a desplegar els microserveis a la plataforma cloud amazon, AWS. ✔

En finalitzar el projecte, les conclusions extretes han sigut, en primer lloc, que el projecte ha servit per a posar en pràctica els coneixements adquirits al grau, en especial l'adquirit a l'itinerari d'enginyeria del programari.

En segon lloc, que la corba d'aprenentatge d'un projecte amb frontend, backend i infraestructura requereix molt temps per adquirir els coneixements i entendre les diferents parts.

En tercer lloc, que la gestió d'un projecte amb el mètode de cascada és complicada de complir, ja que sorgeixen problemes o requisits enmig de la implementació del projecte que no s'havien tingut en compte a l'anàlisi.

Finalment, també s'ha vist que la capa gratuïta d'AWS no sembla prou potent per a aplicacions Java.

4.2 Funcionalitats que falten

No hi ha hagut temps suficient per a implementar totes les funcionalitats detectades a la fase d'anàlisi i disseny. Separem les funcionalitats que falten en funcionalitats aplicació, funcionalitats d'usabilitat i funcionalitats d'implementació.

Funcionalitats d'aplicació

- Configuració de cites per empreses (reserva concurrents, reserves seguides)
- Baixa d'usuari
- Exportar calendari
- Canvi de cita per un altre data

Funcionalitats d'usabilitat

- Dies que apareixen com a disponibles al calendari quan han estat reservades totes les franges horàries.

Funcionalitats d'implementació

- Implementació dels tests, només s'han implementat una part dels tests.
- Enviar els correus de forma asíncrona amb la utilització de cues.
- Millorar les consultes SQL.
- Millorar la gestió d'errors.

4.3 Evolucions de l'aplicació

En cas de que segueixi el desenvolupament de l'aplicació estaria bé continuar amb les següents tasques:

- Implementació de les funcionalitats descrites en l'apartat anterior
- Afegir més proveïdors pel registre i login amb oAuth2.
- Traducció dels textos
- Disseny professional per l'aplicació web, creació de logos, …
- Disseny mobile friendly.
- Afegir cache pels endpoints on la informació no sigui tan dinàmica, com l'endpoint d'informació d'usuari quan es vol utilitzar per obtenir el correu electrònic.
- Observabilitat, monitoratge i logging de l'aplicació amb eines com NewRelic, Datadog, Dynatrace, ...
- Utilització d'un proveïdor per enviar els emails, com mailgun, mailchimp, …
- Definir un model de subscripció a l'eina.

5. Glossari

IDE (Integrated Development Environment): Entorn de desenvolupament integrat que proporciona eines per a la programació com editor de codi, depurador i compilador.

IntelliJ: IDE per al desenvolupament d'aplicacions Java.

Visual Studio Code: IDE per al desenvolupament d'aplicacions web i altres tecnologies.

Microservei: Unitat funcional petita i independent que forma part d'una arquitectura més gran, comunicant-se amb altres microserveis.

Jakarta EE (anteriorment Java EE): Especificacions que estenen Java per a aplicacions empresarials.

Spring Boot: Framework que simplifica la creació de microserveis i aplicacions amb Java.

Spring Data JPA: Llibreria de Spring Boot per treballar amb bases de dades mitjançant JPA (Java Persistence API).

Spring Security: Llibreria de Spring Boot per gestionar la seguretat de les aplicacions.

Spring Actuator: Llibreria de Spring Boot per monitoritzar i gestionar aplicacions en producció.

Lombok: Llibreria per reduir el codi cerimonial en Java mitjançant anotacions.

Swagger: Eina per dissenyar, construir, documentar i consumir APIs REST.

JUnit: Framework per a la creació i execució de tests unitàries en Java.

Mockito: Llibreria per crear tests de components individuals amb mocks en Java.

Gradle: Sistema d'automatització per a la construcció, prova i desplegament de projectes.

PostgreSQL: Sistema gestor de bases de dades relacionals.

Docker: Eina per crear, desplegar i executar aplicacions dins de contenidors.

Docker Compose: Eina per definir i gestionar aplicacions multi-contenidor amb Docker.

React: Llibreria de JavaScript per crear interfícies d'usuari basades en components.

Axios: Llibreria de JavaScript per fer peticions HTTP des del navegador i Node.js.

Material UI: Llibreria de components React per construir interfícies d'usuari amb estil.

Github: Eina per a control de versions i col·laboració en projectes de programari.

Pipeline: Seqüència automatitzada de passos per construir, provar i desplegar programari.

Github Actions: Plataforma per automatitzar fluxos de treball de desenvolupament directament des de GitHub.

Cloudflare: Plataforma que proporciona serveis de CDN, seguretat i domini per a webs.

AWS (Amazon Web Services): Plataforma de serveis de computació al núvol proporcionada per Amazon.

S3 (Simple Storage Service): Servei d'emmagatzematge d'objectes escalable d'AWS.

EC2 (Elastic Compute Cloud): Servei d'AWS que ofereix capacitat de computació escalable al núvol.

RDS (Relational Database Service): Servei d'AWS per gestionar bases de dades relacionals.

API Gateway: Servei que actua com a porta d'entrada per a API, gestionant el trànsit d'API de manera segura i escalable.

VPC (Virtual Private Cloud): Xarxa virtual dedicada dins d'AWS per desplegar recursos de forma segura.

ELB (Elastic Load Balancing): Servei que distribueix automàticament el trànsit d'aplicacions entrant entre múltiples instàncies EC2.

6. Bibliografia

[1]. Project Lombok annotations -
https://projectlombok.org/features/

[2]. Spring Data JPA -
https://spring.io/guides/gs/accessing-data-jpa

[3]. Spring Security Authorization -
https://developer.okta.com/blog/2019/06/20/spring-preauthorize

[4]. Spring Security Cors -
https://docs.spring.io/spring-security/reference/reactive/integrations/cors.html

[5]. Spring Security OAuth2 login -
https://docs.spring.io/spring-security/reference/reactive/oauth2/login/index.html

[6]. Spring Security OAuth2 client -
https://docs.spring.io/spring-security/reference/reactive/oauth2/client/index.html

[7]. Spring Security OAuth2 resource server -
https://docs.spring.io/spring-security/reference/reactive/oauth2/resource-server/index.html

[8]. Spring Boot Actuators -
https://www.baeldung.com/spring-boot-actuators

[9]. ReactJS tutorial -
https://www.youtube.com/watch?v=7iobxzd_2wY&list=PLUofhDIg_38q4D0xNWp7FEHOTcZhjWJ29

[11] ViteJS environment variables -
https://vitejs.dev/guide/env-and-mode

[12]. Material UI learning documentation -
https://mui.com/material-ui/getting-started/learn/

[13]. React Deployment in AWS -
https://docs.aws.amazon.com/es_es/prescriptive-guidance/latest/patterns/deploy-a-react-based-single-page-application-to-amazon-s3-and-cloudfront.html

[14]. GitHub Actions quickstart -
https://docs.github.com/en/actions/quickstart

[15]. GitHub Actions tutorial -
https://www.youtube.com/watch?v=R8_veQiYBjI

[16]. Cloudflare domain configuration - https://developers.cloudflare.com/fundamentals/setup/manage-domains/add-site/

[17]. Cloudflare to S3 configuration - https://developers.cloudflare.com/support/third-party-software/others/configuring-an-amazon-web-services-static-site-to-use-cloudflare/

7. Annexos

Annex 1: Inicialització de l'aplicació

Com que l'aplicació está dividida en microserveis backend i aplicacions frontend, per a fer funcionar l'aplicació aixecarem tots els components. Com s'explica en el readme dels diferents repositories, hem de tenir les diferents eines instal·lades per a aixecar els projectes:

- Frontend: nodejs i npm
- Backend: docker, docker-compose, java 21

Primerament aixecarem els microserveis backend. Per la gestió de les dependències de infraestructura, en aquest cas la base de dades ho fem amb *docker-compose*. Aixequem la base de dades amb la següent comanda als repostories *eagenda-users, eagenda-calendar, eagenda-booking*:

```
docker compose up -d
```

A continuació aixequem l'aplicació. En aixecar-se es crearà l'equema de la base de dades si no existia previament. Això ho fa tenint en compte les entitats definides en les aplicacions.

```
./gradlew bootRun
```

Amb aquestes passes, ja tindrem els microserveis backend escoltant a *localhost:8080, localhost:8081 i localhost:8083.*

Ara aixequem les aplicacions frontend. Primerament ens assegurem de tenir instal·lades les llibreries i a continuació posem en marxa els frontend.

```
npm install
node run dev
```

Amb això ja hauriem de tenir l'aplicació funcionant. Els frontals escolten a *localhost:3000* (aplicació web) i *localhost:3030* (panell d'administració).

Hem de tenir en compte que les bases de dades estará buida, per tant és interessant començar a utilitzar l'aplicació des del panell d'administració, creant un compte i configurant la disponibilitat.

Annex 2: Github Actions

Per a tenir una forma de comprovar que no hi ha regressions en el codi, tenim implementats els tests als projectes de backend. Tot i això, només tenint els tests no ens assegurem que s'estiguin comprovant els resultats d'aquests.

Per assegurar-nos que els verifiquen que tots els tests estan passat correctament, s'ha configurat les GitHub Actions. A la Figura 23 es pot veure una pipeline de GitHub actions, que es genera en crear un Pull Request contra la branca main.

En aquesta pipeline s'executen els tests configurats al projecte i també es comproven que les dependències del projecte estan actualitzades, gràcies a la funcionalitat proveïda per GitHub per a comprovar-les.

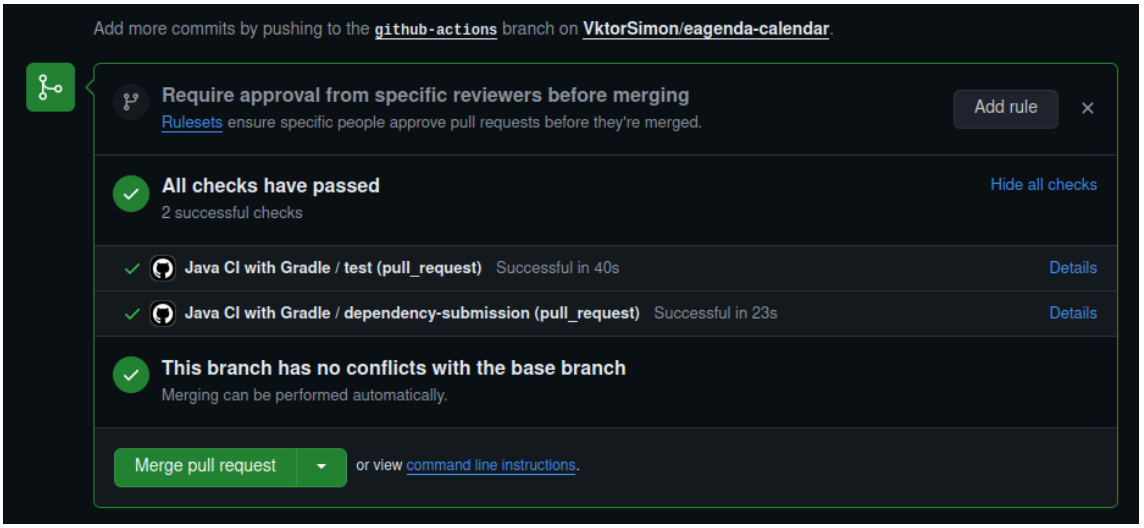


Figura 23: GitHub Actions

Annex 3: Cloudflare

A Cloudflare s'ha configurat el domini `eagenda.live`, per a rebre les peticions que es fan a l'aplicació. A part de les configuracions pròpies del domini, s'han configurat tres redireccions CNAME per als subdominis `admin`, `www` i `api`. Els dos primers redirigeixen el tràfic cap a les aplicacions frontends

que estan desplegades als buckets de S3. El subdomini `api` ha estat configurat per a rebre el tràfic configurat a l'API Gateway.

Type	Name	Content	Proxy status	TTL	Actions
CNAME	admin	admin.eagenda.live.s3-website.eu-north-...	Proxied	Auto	Edit
CNAME	api	d-boqir5a5ph.execute-api.eu-north-1.am...	DNS only	Auto	Edit
CNAME	www	www.eagenda.live.s3-website.eu-north-1....	Proxied	Auto	Edit

Figura 24: Cloudflare

Annex 4: AWS

A continuació es mostra un resum dels components creats a Amazon Web Services.

Instàncies a EC2:

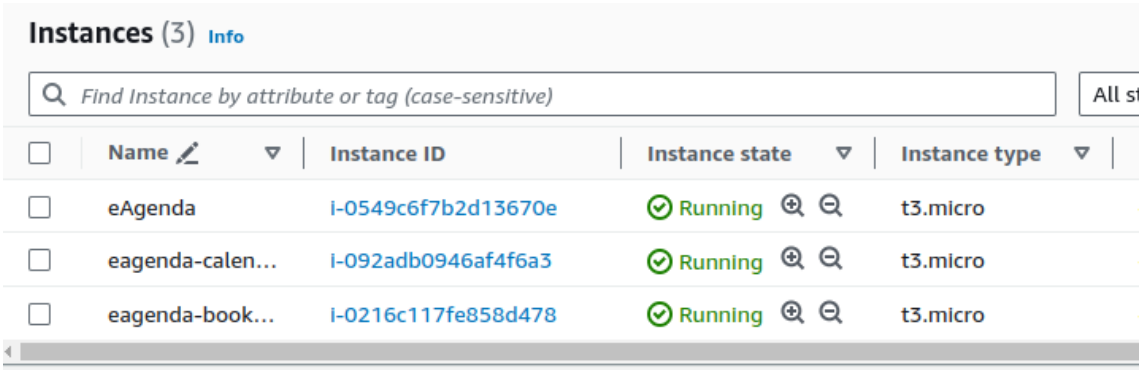


Figura 25: AWS S3

Balancejador de càrrega a ELB:

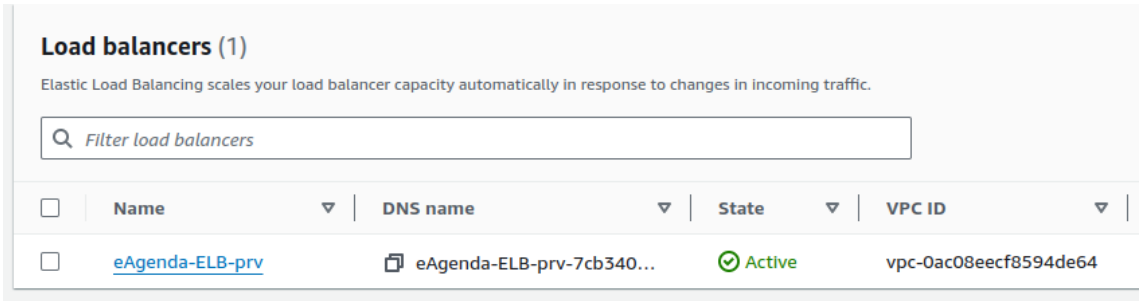


Figura 26: AWS ELB

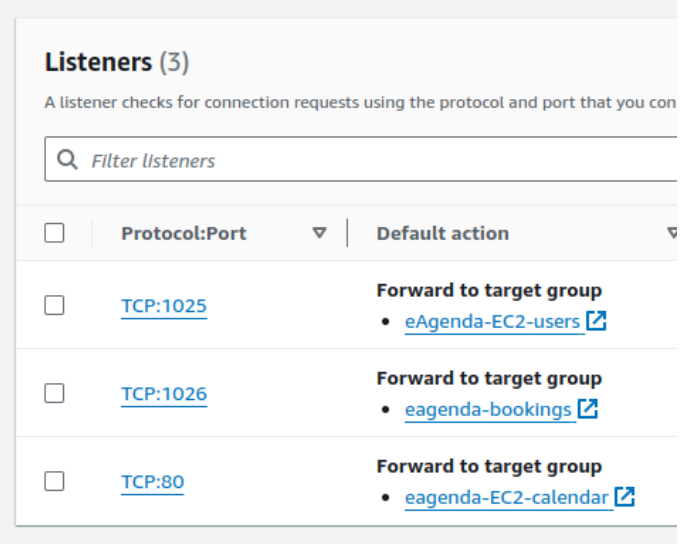


Figura 27: AWS ELB Listeners

Xarxa privada VPC:

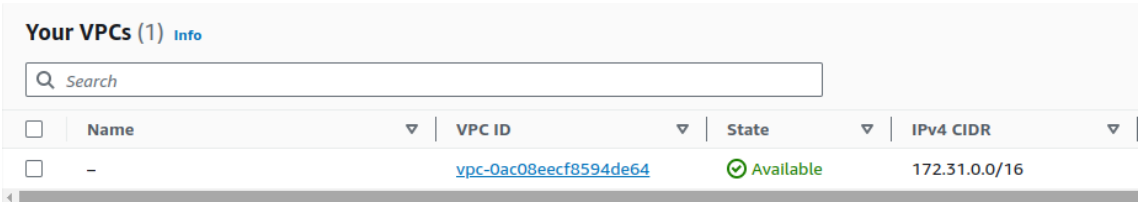


Figura 28: AWS VPC

Base de dades RDS:

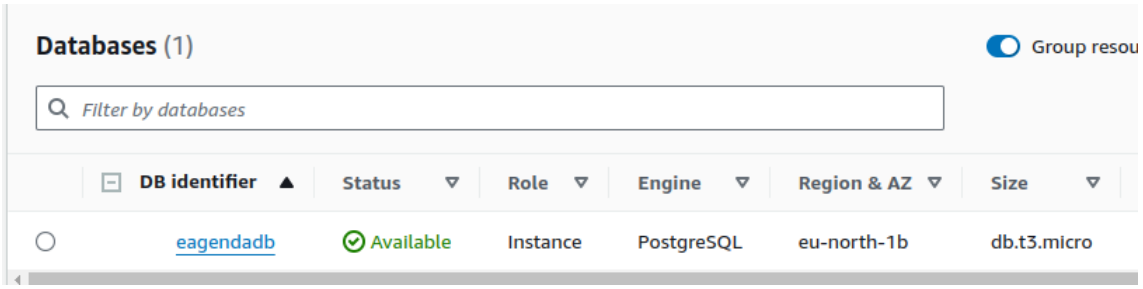


Figura 29: AWS RDS

API Gateway:

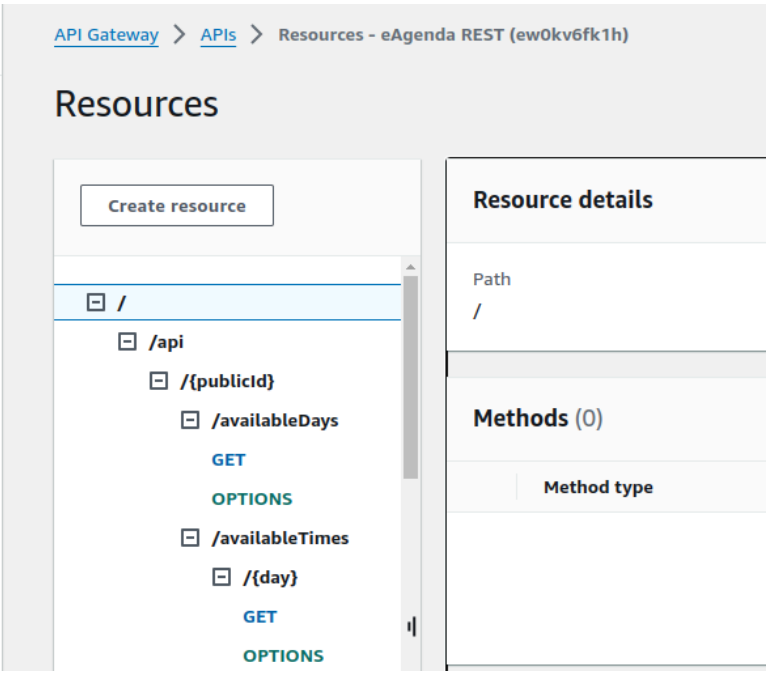


Figura 30: AWS API Gateway

Buckets S3:

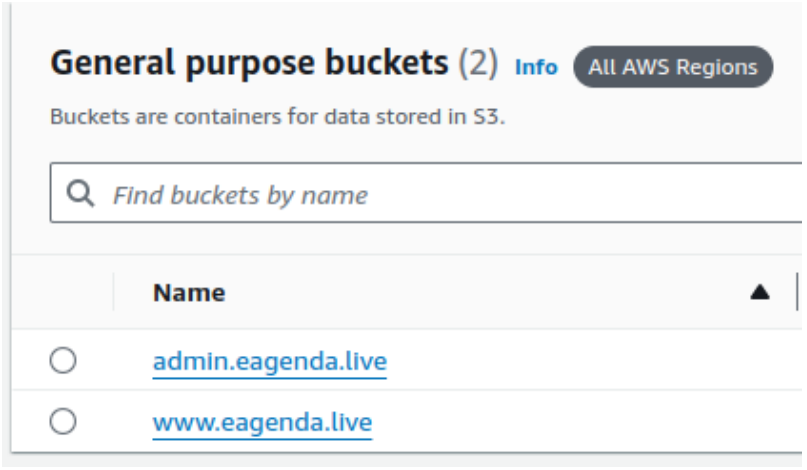


Figura 31: AWS S3