Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Proyecto 1

Citas-App

31-Octubre-2024

Jhony Roel Fuentes Lopez 202031288
Estuardo Israel Ramos Gomez 201830358

Índice

Introducción

Este manual técnico proporciona una visión integral de la aplicación de comercio electrónico Citas-app, abarcando todos los aspectos necesarios para su comprensión, implementación y mantenimiento. Ofrece una guía detallada sobre la estructura del proyecto, que incluye tanto el frontend desarrollado con Angular como el backend en Spring-Boot, así como la gestión de bases de datos con Mysql. Se profundiza en las tecnologías clave utilizadas, la implementación de autenticación de dos factores (2FA) para garantizar la seguridad. Además, se explica el flujo de trabajo para el control de versiones mediante Git y GitHub,destacando la estructura de ramas del repositorio y las prácticas de integración continua utilizando Jenkins. La gestión del proyecto y las historias de usuario así como los criterios de aceptación a través de Jira. Finalmente, el manual incluye instrucciones para la instalación y configuración del entorno de desarrollo, así como estrategias para el mantenimiento y actualización modular del sistema, asegurando una administración eficiente y adaptable a futuras mejoras.

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Objetivos

Generales

- Desarrollar una plataforma que permita a los usuarios administrar las citas de su negocio, así como a los clientes poder tener una herramienta para poder planificar sus citas y tener una mejor interacción con el negocio.

Específicos

- Implementar una API REST que gestione usuarios,citas,reportes utilizando Spring-Boot.
- Desarrollar una interfaz de usuario moderna en Angular.
- Configurar un sistema seguro de autenticación con JWT y 2FA para la protección de datos.
- Utilizar Git y GitHub para control de versiones del proyecto utilizando ramas.
- Utilizar Jira para poder manejar las historias de usuario y aplicar metodología spring.
- Realizar pruebas unitarias.

Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

Nombre Del Proyecto

Citas-App

Descripción General de la aplicación

La aplicación es una plataforma de gestión de citas que permite a los usuarios registrar una cuenta, navegar por los servicios disponibles, programar citas, ver sus citas próximas, y gestionar su perfil. Los administradores pueden gestionar los servicios ofrecidos, horarios, permisos de los usuarios y datos de los clientes. Se implementa un backend en Spring Boot, utilizando autenticación con JWT y autenticación de dos factores (2FA) para generar códigos de autenticación. La base de datos es MySQL, y las imágenes se almacenan en Dropbox. Además, se aplican buenas prácticas de Git para el control de versiones, gestionado a través de GitHub y Jira para el control de tareas en todo el ciclo del desarrollo.

Responsables del proyecto

Jhony Roel Fuentes Lopez
Estuardo Israel Ramos Gomez

PALABRAS CLAVE

- **Spring-Boot:** Entorno de ejecución para el backend.
- **Mysql**: Base de datos relacional.
- **JWT**: Sistema de autenticación basado en tokens.
- **2FA:** Autenticación de dos factores para mayor seguridad.
- **DROPBOX:** Almacenamiento de imágenes en la nube.
- **Angular:** Framework para el frontend.
- **Tailwind CSS:** Framework CSS para diseño responsivo.
- **Flowbite:** Biblioteca de componentes UI.
- **Git:** Sistema de control de versiones.
- **GitHub:** Plataforma para alojar repositorios de Git.

Requerimientos Técnicos

Procesador: core i5 gen o superior
Memoria RAM: 2 Gb
Espacio de almacenamiento: 10 Gb
Sistema Operativo
Otros: Tener instaladas las siguientes herramientas.

Java

Version: openjdk version "21.0.2" 2024-01-16 LTS

Lenguaje de programación versátil y orientado a objetos que se utiliza para desarrollar aplicaciones robustas y escalables.

Instalación en Windows:

1. Visita el sitio web oficial de Java: oracle.com/java/technologies/javase-downloads.html.
2. Descarga el instalador de Java JDK.
3. Ejecuta el instalador y sigue las instrucciones en pantalla.

Instalación en Linux:

Abre una terminal y ejecuta los siguientes comandos:

```
sudo apt update
```

```
sudo apt install openjdk-11-jdk
```

Spring Boot

Version: 3.3.5

Framework de desarrollo que simplifica la creación de aplicaciones Java basadas en Spring.

Instalación en Windows:

1. Descargar Spring Boot: Ve a Spring Boot Downloads y descarga el archivo ZIP del proyecto o utiliza Spring Initializr para generar un proyecto.
2. Configurar tu IDE: Si usas un IDE como IntelliJ IDEA o Eclipse, abre el proyecto descargado.

Mysql
Versión: 9.0.1
Base de datos relacional para la gestión de datos como usuarios, productos y pedidos.

Instalación en Windows: Descarga e instala desde postgresql.org.

Instalación en Linux:

sudo apt update
sudo apt install mysql-server

Angular
Versión 16
Framework para el desarrollo de frontend en aplicaciones web.

Instalación en Windows y Linux: Necesitas tener Node.js instalado, luego ejecuta:

npm install -g @angular/cli

Tailwind CSS
Framework CSS utilitario para crear interfaces de usuario responsivas.

Instalación en Windows y Linux:
npm install -D tailwind css
npx tailwindcss init

Git
Sistema de control de versiones para gestionar el código.

Instalación en Windows:
Descarga desde git-scm.com.
Instalación en Linux:
sudo apt update
sudo apt install git

Postman
Plataforma popular utilizada por desarrolladores para probar y depurar APIs.

Instalación en windows:
Visita el sitio web oficial de Postman: Postman.
Descarga el instalador para Windows (archivo .exe).

Instalación en Linux:
sudo snap install postman

Aquí tienes el formato de instalación para **Jenkins**, siguiendo el estilo que has proporcionado:

Jenkins
Herramienta de integración continua que automatiza el proceso de construcción, prueba y despliegue de aplicaciones.

Instalación en Windows:

1. Visita el sitio web oficial de Jenkins: jenkins.io.
2. Descarga el instalador para Windows (archivo .msi).
3. Ejecuta el instalador y sigue las instrucciones en pantalla.

Instalación en Linux:

Abre una terminal y ejecuta los siguientes comandos:
bash

```
sudo apt update
sudo apt install openjdk-11-jdk
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo
apt-key add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins
```

Inicia Jenkins:

```
sudo systemctl start jenkins
```

1. Accede a Jenkins desde tu navegador en: `http://localhost:8080`.

Spring Boot

- **Dependencias adicionales**:

JUnit: Para pruebas unitarias.

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.7.1</version>
    <scope>test</scope>
```


```
</dependency>
```

JaCoCo: Para la cobertura de código.

```
<dependency>
    <groupId>org.jacoco</groupId>
    <artifactId>org.jacoco.agent</artifactId>
    <version>0.8.6</version>
    <scope>runtime</scope>
</dependency>
```

-

Mockito: Para pruebas simuladas.

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>3.11.2</version>
    <scope>test</scope>
</dependency>
```

Herramientas y lenguajes utilizados para el desarrollo.

Java
Lenguaje de programación versátil y orientado a objetos que se utiliza para desarrollar aplicaciones robustas y escalables en el backend. Se integra con frameworks como Spring Boot para el desarrollo de aplicaciones empresariales.

Spring Boot
Spring Boot es un marco de trabajo para desarrollar aplicaciones Java que permite la creación rápida de aplicaciones backend. Proporciona una configuración automática y una estructura predefinida, lo que facilita el desarrollo de microservicios y aplicaciones web robustas. Su enfoque en la simplicidad y la productividad permite a los desarrolladores centrarse en la lógica de negocio en lugar de en la configuración.

Incluye herramientas de pruebas como:

- **JUnit**: Para realizar pruebas unitarias de manera eficiente.
- **Mockito**: Para realizar pruebas simuladas de objetos en el código.
- **JaCoCo**: Para medir la cobertura de código y asegurarse de que se están realizando pruebas adecuadas.

MySQL
MySQL es un sistema de gestión de bases de datos relacional de código abierto que se caracteriza por su rapidez y facilidad de uso. En este proyecto, se utiliza para almacenar y gestionar la información relacionada con usuarios, citas y servicios, garantizando integridad de datos y soporte para consultas complejas.

Angular
Angular es un framework de JavaScript mantenido por Google que se utiliza para el desarrollo de aplicaciones frontend. Permite crear aplicaciones web de una sola página (SPA) con una arquitectura basada en componentes. Angular ofrece una experiencia de usuario dinámica y una estructura organizada para el desarrollo del frontend.

JWT (JSON Web Tokens)
JWT es un estándar abierto para la autenticación segura de usuarios mediante tokens. Se utiliza para enviar información entre partes de forma compacta y segura. En este proyecto, se utiliza JWT para gestionar la autenticación de los usuarios, permitiendo que inicien

sesión y mantengan sesiones seguras sin necesidad de almacenar credenciales en cada solicitud.

2FA (Two-Factor Authentication)
La autenticación de dos factores (2FA) es un método de seguridad que agrega una capa adicional de protección, solicitando a los usuarios dos formas de verificación antes de acceder a una cuenta. En este proyecto, se implementa con tecnologías como Verify FA y Enable FA, mejorando la seguridad del acceso de los usuarios.

Tailwind CSS
Tailwind CSS es un framework CSS utilitario que facilita el diseño de interfaces web. A diferencia de otros frameworks, Tailwind no ofrece componentes predefinidos, sino que proporciona clases utilitarias que permiten personalizar el diseño con gran flexibilidad y precisión. Se utiliza para desarrollar una interfaz de usuario moderna y responsiva.

Flowbite
Flowbite es un conjunto de componentes de interfaz de usuario diseñados específicamente para integrarse con Tailwind CSS. Estos componentes permiten crear rápidamente elementos comunes de UI como botones, tarjetas, formularios, y mucho más, acelerando el proceso de desarrollo frontend.

Git
Git es un sistema de control de versiones distribuido que permite a los equipos de desarrollo gestionar los cambios en el código de forma eficiente y colaborativa. En este proyecto, Git se utiliza para llevar un historial completo de los cambios realizados en el código y para facilitar el trabajo colaborativo entre los desarrolladores.

Postman
Es una herramienta de desarrollo de APIs que facilita la creación, prueba y gestión de solicitudes HTTP. Permite a los desarrolladores enviar solicitudes a servidores web y recibir respuestas, facilitando la prueba de servicios web y APIs. Con Postman, se pueden realizar solicitudes GET, POST, PUT, DELETE, entre otras, y analizar las respuestas en un entorno intuitivo. Además, Postman soporta la automatización de pruebas y la generación de documentación para APIs.

GitHub
GitHub es una plataforma basada en la web que utiliza Git para alojar repositorios de código. Proporciona herramientas para la colaboración, como pull requests, issues, y la integración continua. En este proyecto, GitHub permite a los equipos trabajar de manera remota, gestionar versiones del código, y automatizar pruebas y despliegues.

Jira
Jira es una herramienta de gestión de tareas que utiliza tableros, listas y tarjetas para organizar y priorizar el trabajo. Cada miembro del equipo puede asignarse tareas, seguir el progreso y actualizar el estado de cada tarea en tiempo real, facilitando la gestión del ciclo de desarrollo del proyecto.

Jenkins
Herramienta de integración continua que automatiza el proceso de construcción, prueba y despliegue de aplicaciones. Permite a los equipos de desarrollo implementar un ciclo de desarrollo más ágil y eficiente mediante la automatización de tareas repetitivas.

Diagramas

Casos de Uso

Registrar Usuario CU01

- **Descripción:** Permite al usuario registrarse en el Sitio
- **Actores Involucrados:** Usuario
- **Flujo Principal:**
    - El usuario accede al formulario de registro.
    - El usuario completa la información requerida (nombre, apellido, email, contraseña, dirección, nit).
    - El sistema valida la información.
    - El sistema crea la cuenta para el usuario y lo redirige a la página de inicio.

Iniciar Sesión (Autenticación 2FA) CU02

- **Descripción:** Permite al usuario iniciar sesión en el sistema con autenticación de dos factores (2FA).
- **Actores involucrados:** Usuario, Comprador.
- **Flujo Principal:**
    - El usuario accede al formulario de inicio de sesión.
    - El usuario ingresa sus credenciales (correo electrónico y contraseña).
    - El sistema valida las credenciales.
    - El sistema solicita un código de verificación adicional (enviado por SMS o correo electrónico).
    - El usuario ingresa el código de verificación.
    - El sistema verifica el código 2FA.
    - El sistema autentica al usuario y lo redirige a su área de página principal.

Recuperar Contraseña CU03

- **Descripción**: Permite al usuario recuperar su contraseña en caso de olvido.
- **Actores involucrados:** Usuario.
- **Flujo Principal:**
    - El usuario selecciona la opción de "Recuperar contraseña".
    - El usuario ingresa su correo electrónico registrado.
    - El sistema envía un enlace de recuperación de contraseña al correo electrónico.
    - El usuario accede al enlace y restablecer su contraseña.
    - El sistema valida y confirma el restablecimiento de la contraseña.
    - El usuario puede iniciar sesión con la nueva contraseña.

Agendar Cita CU04

- **Descripción**: Permite al usuario agendar una cita.
- **Actores involucrados**: Usuario, Comprador.
- **Flujo Principal:**
    - El usuario accede a la página de agendar citas.
    - El usuario selecciona fecha,el servicio y el servidor(opcional)
    - El sistema muestra los resultados con horarios disponibles
    - El usuario selecciona un horario.
    - El usuario agenda la cita.

Asignar roles a un Usuario CU05

- **Descripción**: Permite al administrador agregar roles a un usuario.
- **Actores involucrados:** Usuario, Administrador.
- **Flujo Principal:**
    - El administrador selecciona el usuario a quien desee agregar el rol.
    - El administrador selecciona el rol.
    - El sistema agrega el rol al usuario.

Gestión de Citas CU06

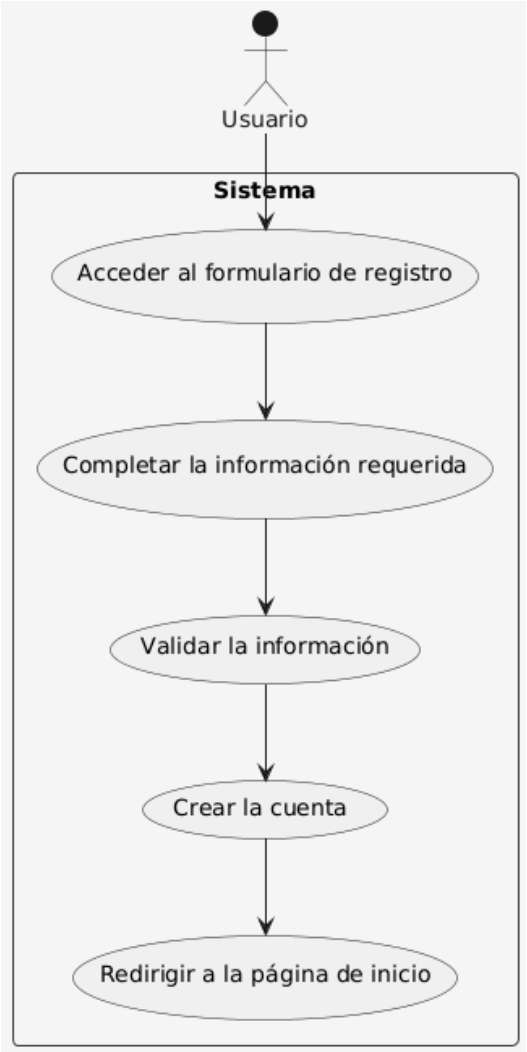
- **Descripción**: Permite a los administradores y/o ayudantes gestionar las citas de la empresa
- **Actores involucrados**: Administrador, Ayudante.
- **Flujo Principal:**
    - El administrador o ayudante accede a la sección de gestión de citas.
    - El sistema muestra las citas por estados.
    - El administrador o ayudante cambia el estado de las citas.
    - El sistema guarda los cambios y actualiza las citas.

Generar Reportes CU07

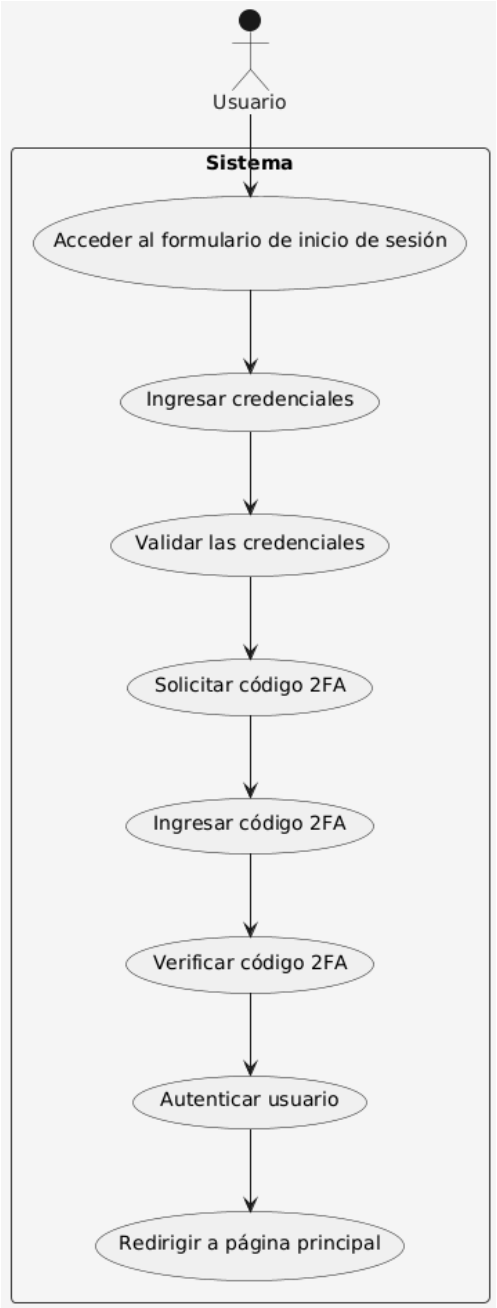
- **Descripción**: Permite al administrador generar reportes.
- **Actores involucrados**: Administrador.
- **Flujo Principal**:
    - El administrador accede al dashboard de administración.
    - El administrador selecciona la opción para generar reportes.
    - El sistema presenta opciones de filtros para los reportes (fecha,Servidor,servicio, etc.).
    - El administrador genera el reporte.
    - El sistema exporta el reporte en el formato solicitado

Diagramas de Casos de Uso

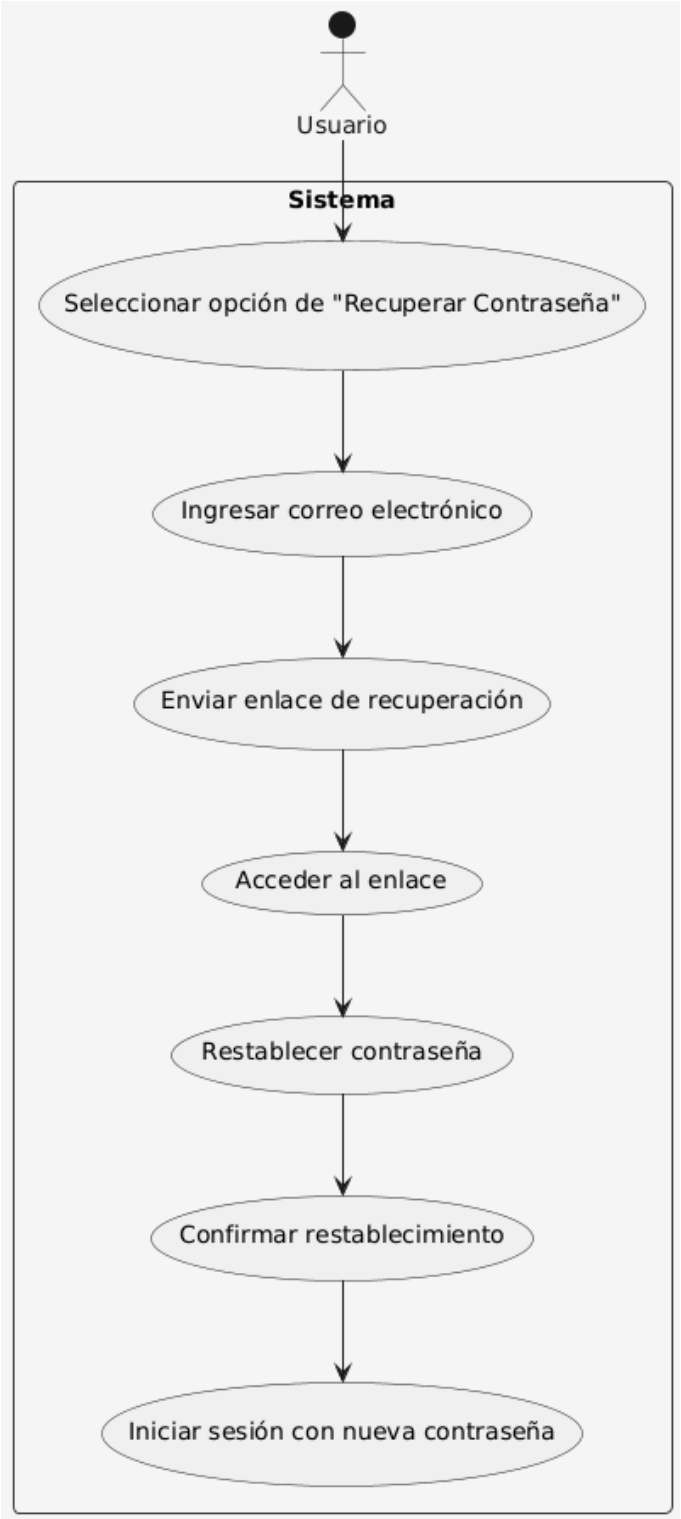
CU01



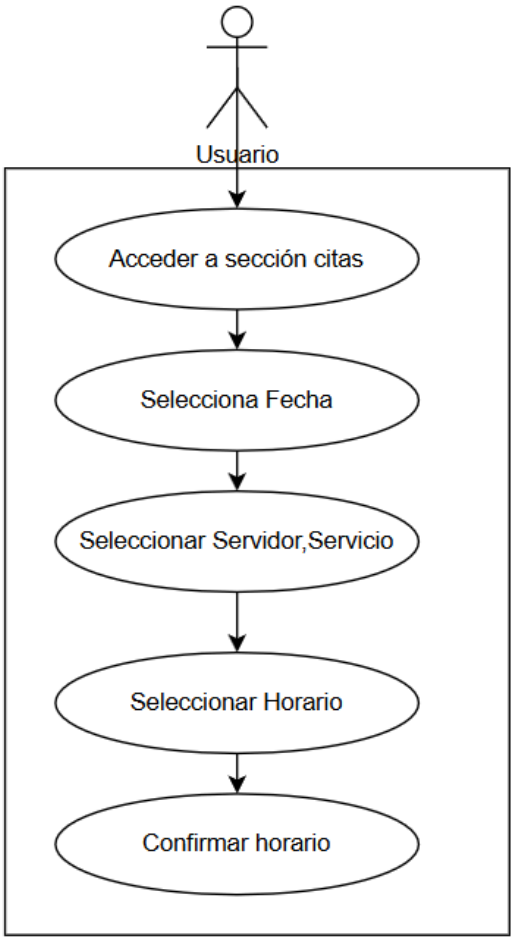
CU02


Usuario
Sistema
Acceder al formulario de inicio de sesión
Ingresar credenciales
Validar las credenciales
Solicitar código 2FA
Ingresar código 2FA
Verificar código 2FA
Autenticar usuario
Redirigir a página principal

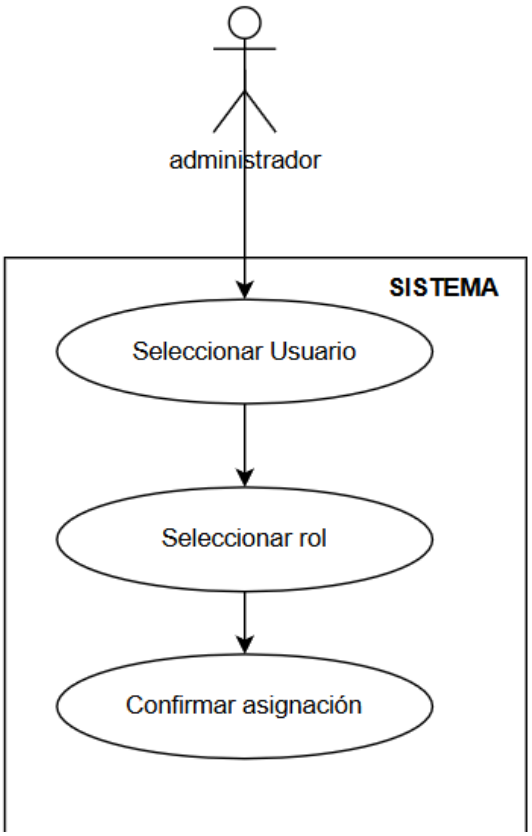
CU03


Usuario
Sistema
Seleccionar opción de "Recuperar Contraseña"
Ingresar correo electrónico
Enviar enlace de recuperación
Acceder al enlace
Restablecer contraseña
Confirmar restablecimiento
Iniciar sesión con nueva contraseña

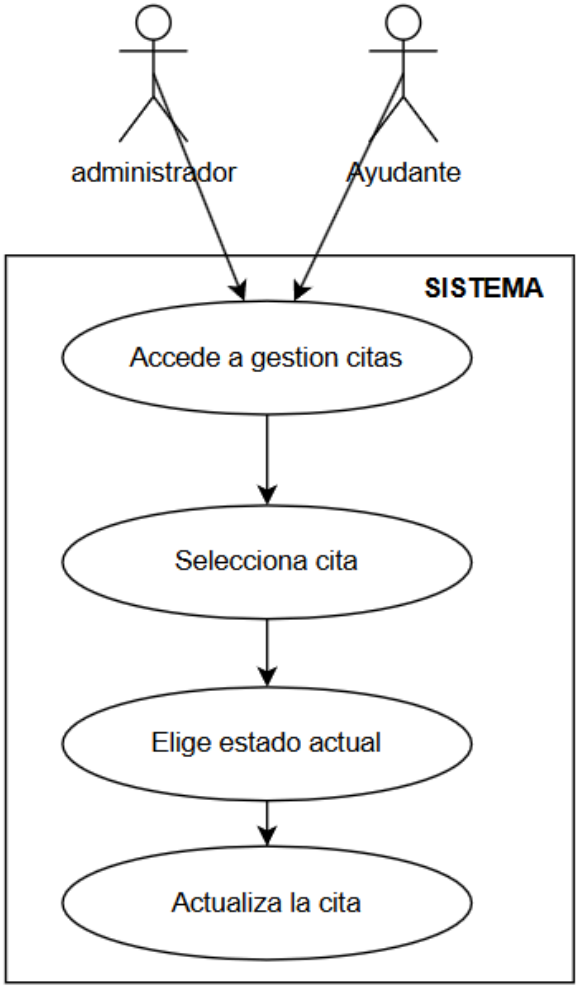
CU04


Usuario
Acceder a sección citas
Selecciona Fecha
Seleccionar Servidor,Servicio
Seleccionar Horario
Confirmar horario

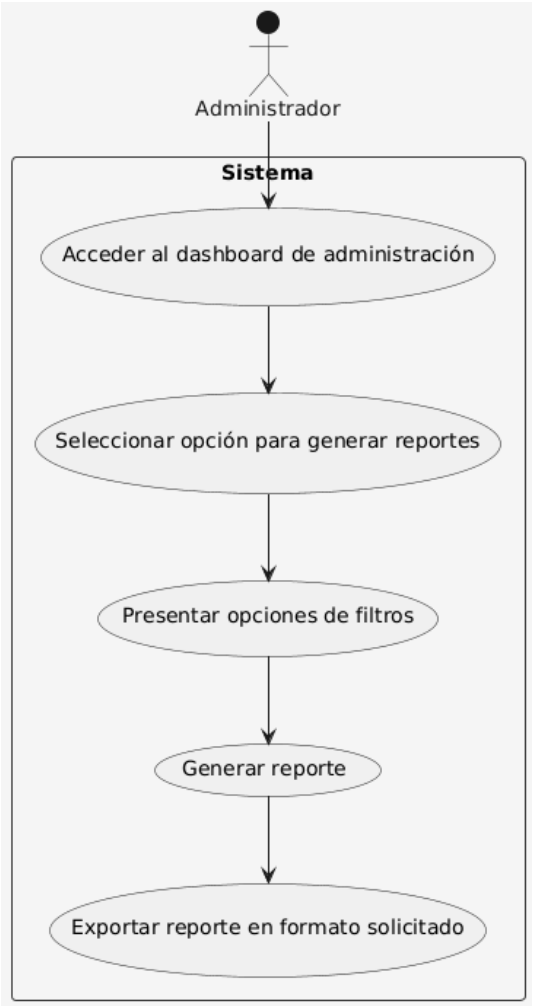
CU05


administrador
SISTEMA
Seleccionar Usuario
Seleccionar rol
Confirmar asignación

CU06

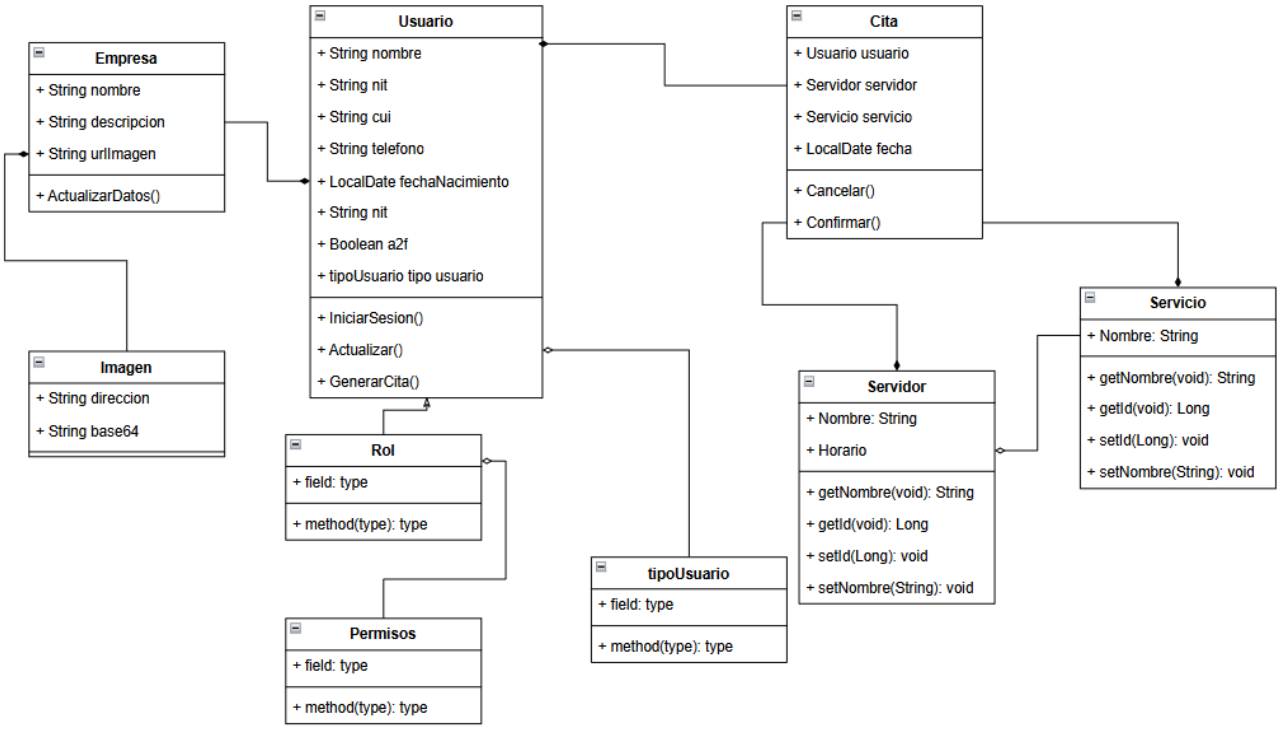

administrador
Ayudante
SISTEMA
Accede a gestion citas
Selecciona cita
Elige estado actual
Actualiza la cita

CU07


Administrador
Sistema
Acceder al dashboard de administración
Seleccionar opción para generar reportes
Presentar opciones de filtros
Generar reporte
Exportar reporte en formato solicitado

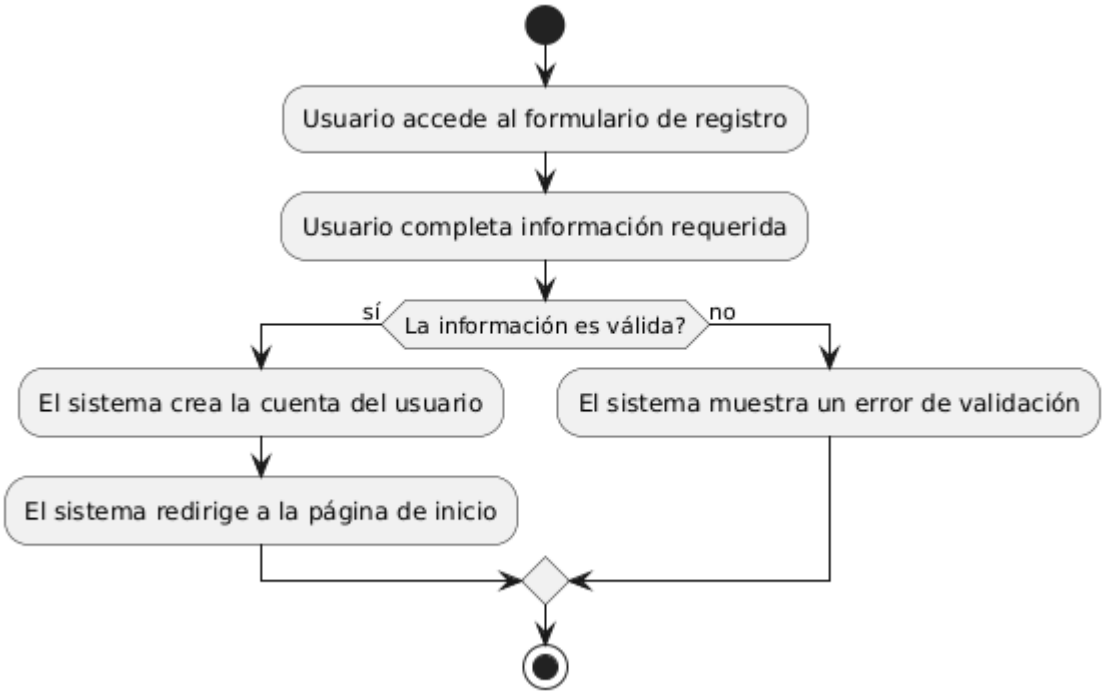
Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

Clases

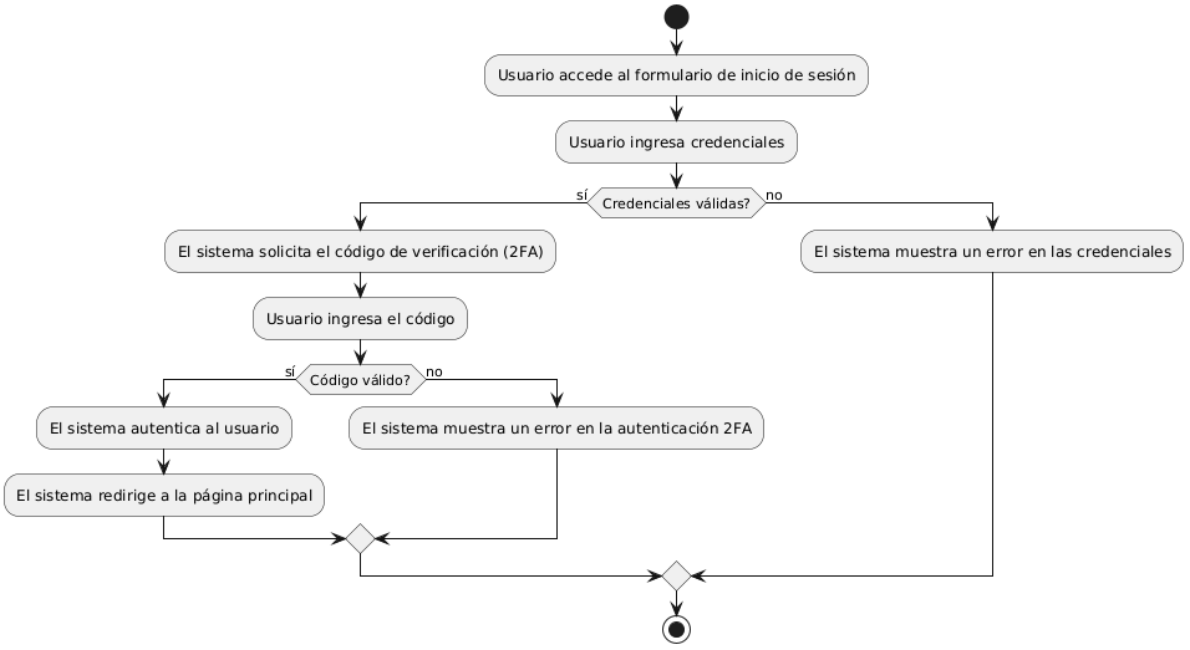


Actividades

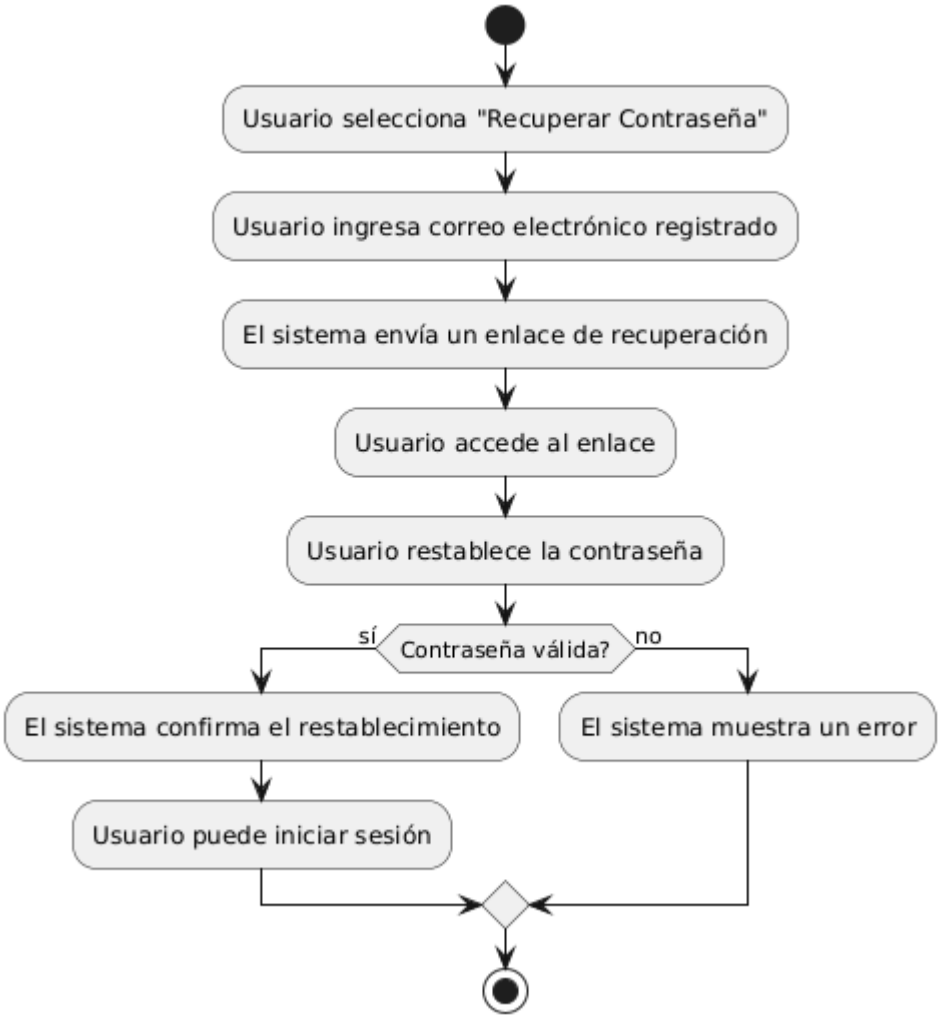
CU01



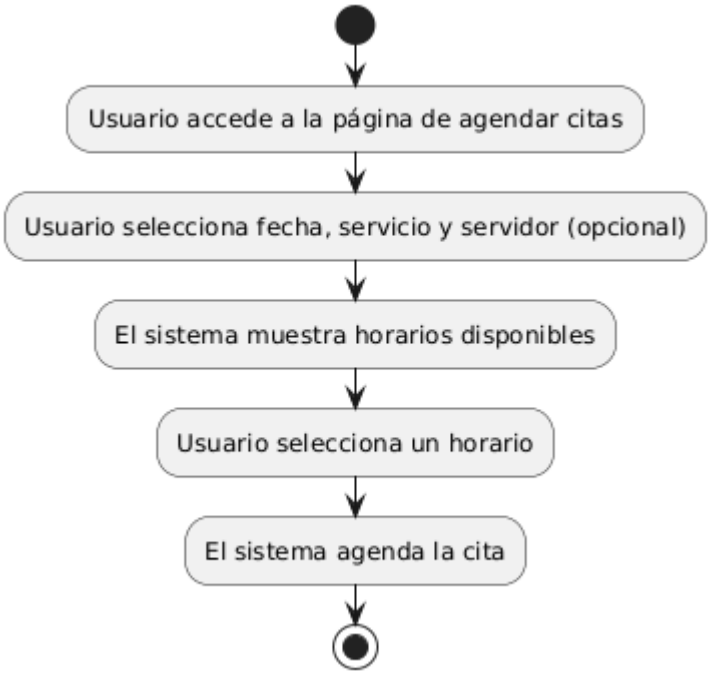
CU02



CU03


Usuario selecciona "Recuperar Contraseña"
Usuario ingresa correo electrónico registrado
El sistema envía un enlace de recuperación
Usuario accede al enlace
Usuario restablece la contraseña
Contraseña válida?
sí
no
El sistema confirma el restablecimiento
El sistema muestra un error
Usuario puede iniciar sesión

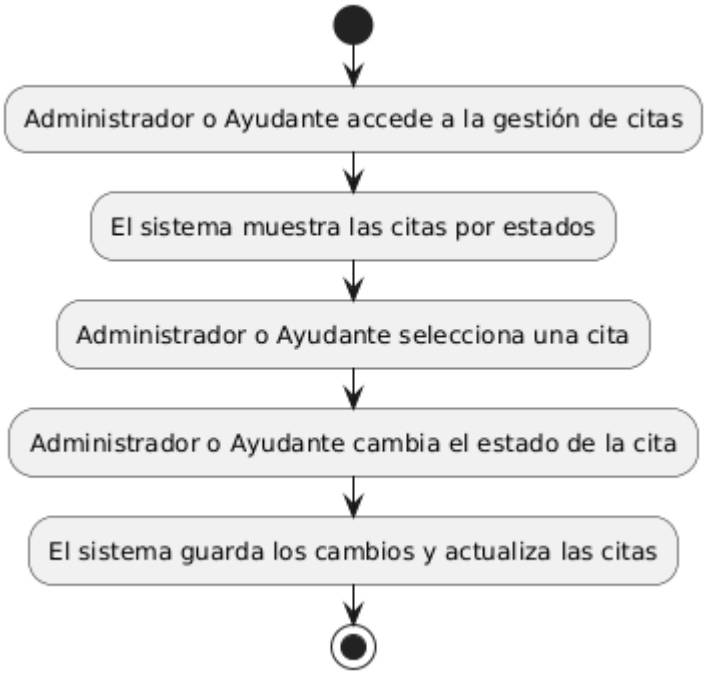
CU04


Usuario accede a la página de agendar citas
Usuario selecciona fecha, servicio y servidor (opcional)
El sistema muestra horarios disponibles
Usuario selecciona un horario
El sistema agenda la cita

CU05


Administrador selecciona el usuario
Administrador selecciona el rol
El sistema agrega el rol al usuario

CU06


Administrador o Ayudante accede a la gestión de citas
El sistema muestra las citas por estados
Administrador o Ayudante selecciona una cita
Administrador o Ayudante cambia el estado de la cita
El sistema guarda los cambios y actualiza las citas

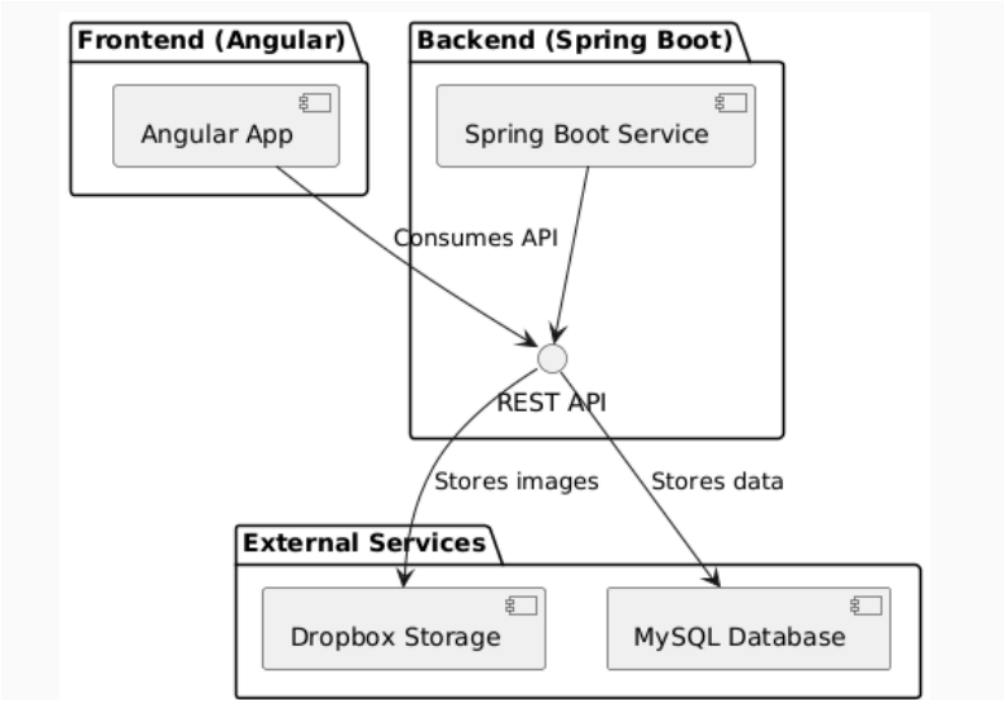
CU07

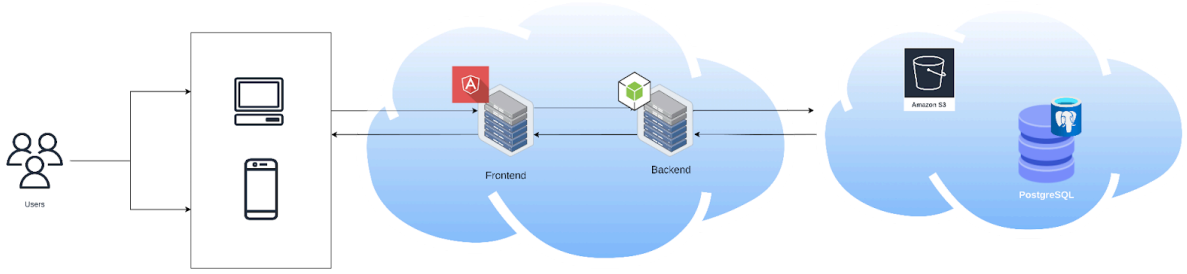

Administrador accede al dashboard de administración
Administrador selecciona la opción para generar reportes
El sistema muestra filtros de reporte (fecha, servidor, servicio, etc.)
Administrador aplica filtros y genera el reporte
El sistema exporta el reporte en el formato solicitado

CU08

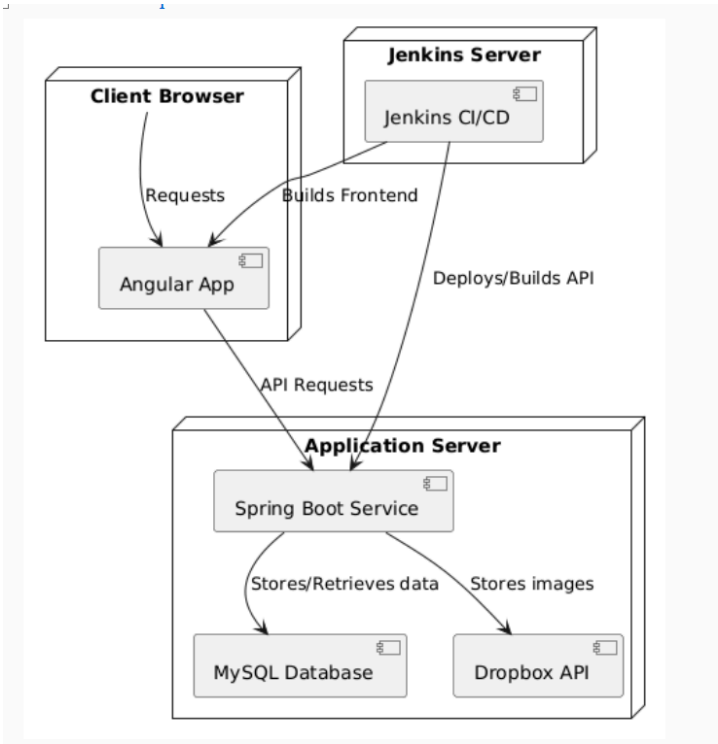

Administrador accede al dashboard
Administrador selecciona la opción de generar reportes
El sistema presenta opciones de filtros
Administrador genera el reporte
El sistema exporta el reporte

Arquitectura

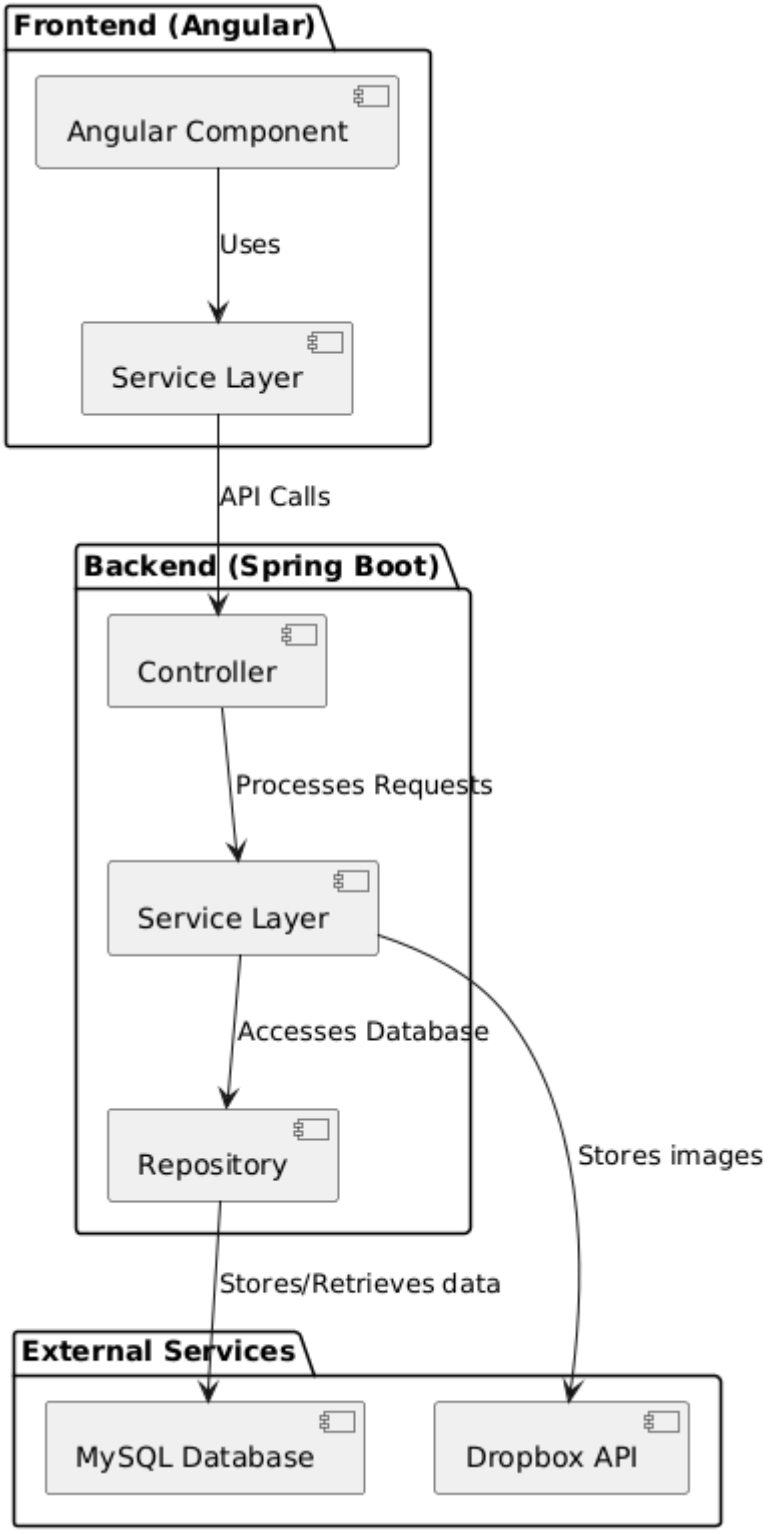




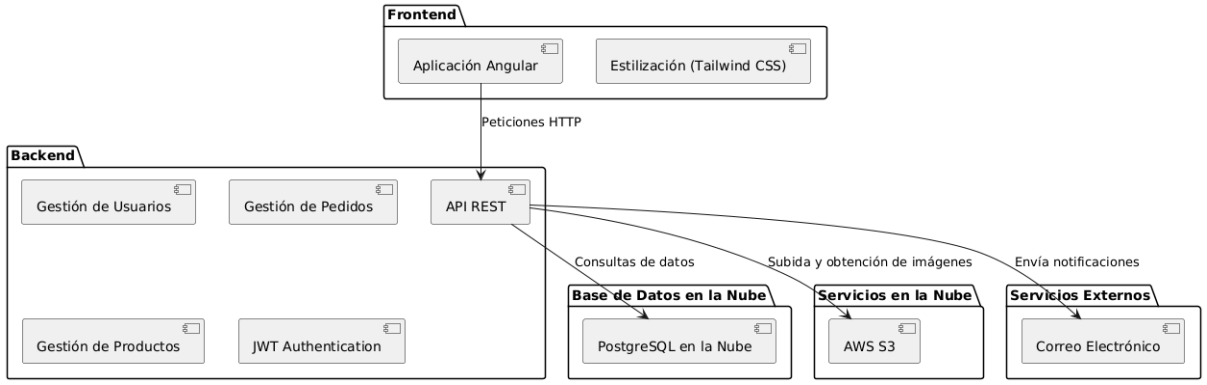
Despliegue


Jenkins Server
Client Browser
Jenkins CI/CD
Requests
Builds Frontend
Angular App
Deploys/Builds API
API Requests
Application Server
Spring Boot Service
Stores/Retrieves data
Stores images
MySQL Database
Dropbox API

Componentes



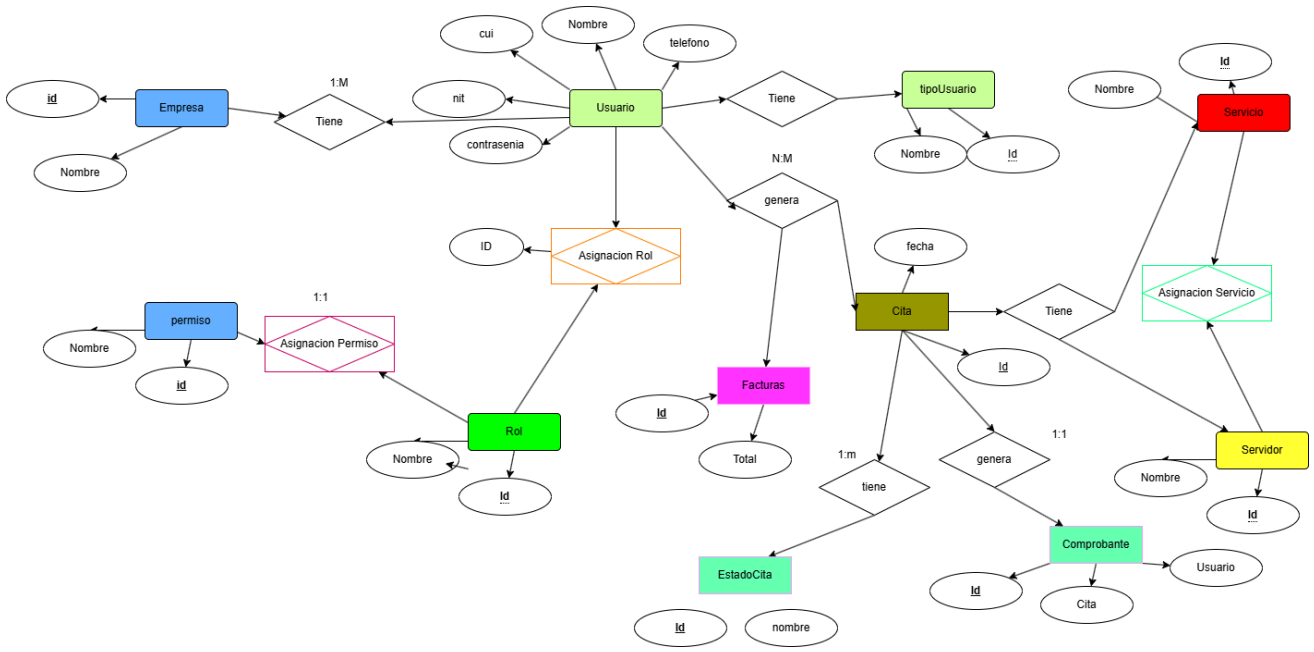
Paquetes


Frontend
Aplicación Angular
Estilización (Tailwind CSS)
Peticiones HTTP
Backend
Gestión de Usuarios
Gestión de Pedidos
API REST
Gestión de Productos
JWT Authentication
Consultas de datos
Subida y obtención de imágenes
Envía notificaciones
Base de Datos en la Nube
PostgreSQL en la Nube
Servicios en la Nube
AWS S3
Servicios Externos
Correo Electrónico

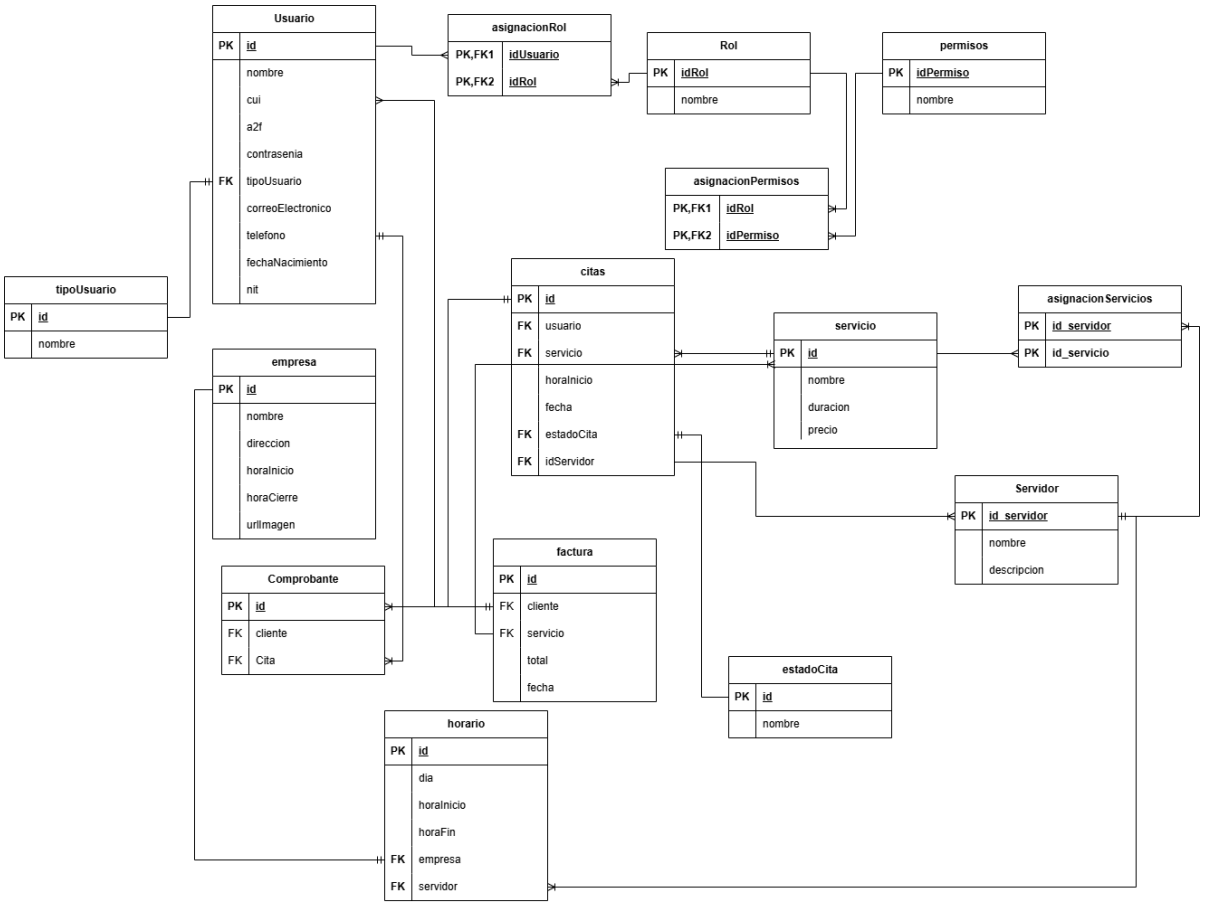
Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

Diagrama Base de datos

Diagrama ER



Relacional


Usuario
PK id
nombre
cui
a2f
contrasenia
FK tipoUsuario
correoElectronico
telefono
fechaNacimiento
nit
asignacionRol
PK,FK1 idUsuario
PK,FK2 idRol
Rol
PK idRol
nombre
permisos
PK idPermiso
nombre
asignacionPermisos
PK,FK1 idRol
PK,FK2 idPermiso
tipoUsuario
PK id
nombre
citas
PK id
FK usuario
FK servicio
horaInicio
fecha
FK estadoCita
FK idServidor
servicio
PK id
nombre
duracion
precio
asignacionServicios
PK id_servidor
PK id_servicio
empresa
PK id
nombre
direccion
horaInicio
horaCierre
urlImagen
Servidor
PK id_servidor
nombre
descripcion
Comprobante
PK id
FK cliente
FK Cita
factura
PK id
FK cliente
FK servicio
total
fecha
estadoCita
PK id
nombre
horario
PK id
dia
horaInicio
horaFin
FK empresa
FK servidor

USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Instalación Del Sistema

Clonar el repositorio desde Github

git clone https://github.com/EstuardoRamos/App-citas

Instalar dependencias para Backend

cd Backend
npm install

Instalar dependencias para Frontend

cd Frontend
npm install

Configurar el archivo .env

Es necesario crear un archivo .env en la raíz del proyecto con las siguiente variables configuradas

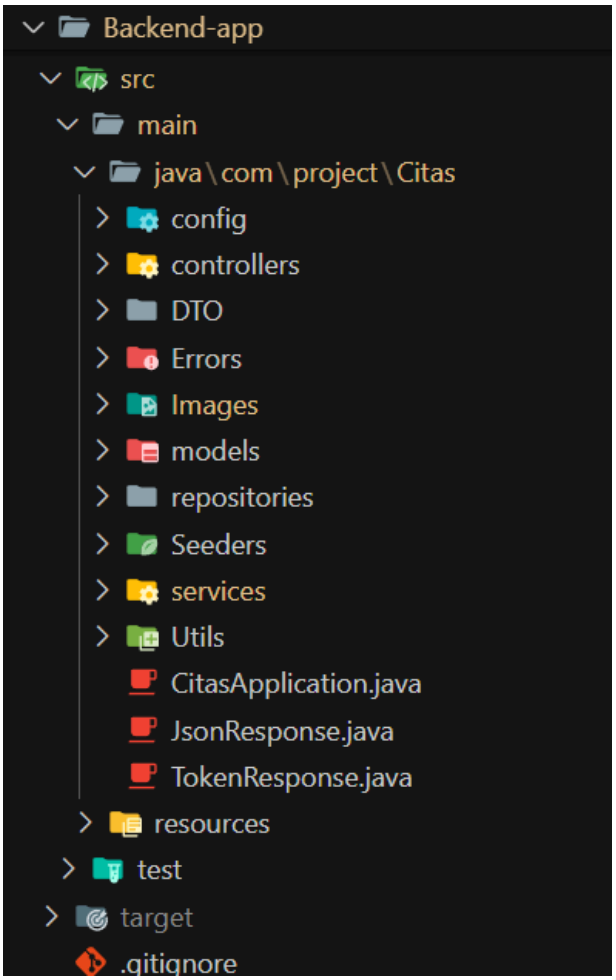
```
# Variables para el backend
APP_PORT=
DATABASE_NAME=
DATABASE_USER=
DATABASE_PASSWORD=
DATABASE_HOST=
DATABASE_PORT=
AWS_BUCKET_NAME=
AWS_BUCKET_REGION=
AWS_PUBLIC_KEY=
AWS_SECRET_KEY=
JWT_KEY=
PASSWORD=
CORREO=
ORIGIN=
```

Estructura del proyecto

Estructura del Backend

```
/backend
│
├── src/main/java/com/project/Citas
│   ├── configs/
│   │   ├── Securityconfig.java        # Configuración para la integración con AWS S3
|   |
│   ├── controllers/
│   │   ├── adminController.java              # Controlador de administración
│   │   ├── userController.java               # Controlador para manejar usuario
│   │   ├── citasController.java              # Controlador para gestionar de Citas
│   │   ├── empresaController.java            # Controlador para gestionar de Empresa
│   │   ├── reportes.controller.java          # Controlador para generar reportes
│   │   └── clienteController.java            # Controlador para la gestión de clientes
│   │
│   ├── DTO/                      # Archivos relacionados con los DTO de solicitudes
│   │
│   ├── Errors/
│   │   ├── ErrorResponse.java                    # Mensaje errores
│   │   ├── GlobalExceptionHandler.java           # Excepciones globales
│   │   ├── ModelAlreadyExistException.java       # Excepciones modelo existentes
│   │   └── ModelNotFoundException.java           # Excepciones modelos no encontrados
│   ├
│   ├──Images/                    # Archivos relacionados con los DTO de solicitudes
│   │   ├── DropBoxImage.java             # Servicio para guardar imágenes en dropbox.
│   │   ├── ImageService.java             # Herramientas para manejo de imágenes
│   │
│   ├──models/
│   │ ├── Citas/
│   ├── asignacionServicios.java           # Modelo para la asignación de servicios
│   ├── citasModel.java                    #Modelo para las citas
│   ├── estadoCitaModel.java               # Modelo para el estado de las citas
│   ├── ServicioModel.java                 # Modelo para los servicios
│   └── ServidorModel.java                 # Modelo para los servidores
│   │ ├── Empresa/
│   └── diasAtencionModel.java       # Modelo para los días de atención de la empres
│   │ ├── primariasCompuestas/
│   ├── rolPermisoId.java            # Clase de clave compuesta para rol y permiso
│   ├── rolUsuarioId.java            # Clase de clave compuesta para rol y usuario
│   └── servidorServicioId.java      # Clase de clave compuesta para servidor y servicio
```

```
│   │   └── Roles/
    ├── asignacionPermisosModel.java       # Modelo para la asignación de permisos
    ├── asignacionRolModel.java            # Modelo para la asignación de roles
    ├── EmpresaModel.java                  # Modelo para la información de la empresa
    ├── horarioModel.java                  # Modelo para los horarios
    ├── permisosModel.java                 # Modelo para los permisos
    └── rolModel.java                      # Modelo para los roles de usuario
│   │
│   ├── services/
│   │   ├── administracionService.js       # Lógica para administración
│   │   ├── citasService.java              # Lógica para manejo de Citas
│   │   ├── clienteService.java            # Lógica para el manejo de clientes
│   │   ├── empresaService.java            # Lógica para el manejo de Servicios
│   │   ├── JwtService.java                # Lógica para generar Token
│   │   ├── reporteService.java            # Lógica para el manejo de reportes
│   │   └── userService.java               # Login para el manejo de usuarios
│   ├── repositories/
│   │   ├── asignacionPermisosRepository.java    # Repositorio para la asignación de
permisos
│   │   ├── asignacionRolRepository.java         # Repositorio para la asignación de roles
│   │   ├── asignacionServiciosRepository.java   # Repositorio para la asignación de
servicios
│   │   ├── citasRepository.java             # Repositorio para el manejo de citas
│   │   ├── diasAtencionRepository.java        # Repositorio para los días de atención
│   │   ├── EmpresaRepository.java           # Repositorio para la información de empresas
│   │   ├── estadoCitaRepository.java        # Repositorio para el estado de las citas
│   │   ├── horarioRepository.java           # Repositorio para los horarios
│   │   ├── permisosRepository.java          # Repositorio para los permisos
│   │   ├── rolRepository.java               # Repositorio para los roles de usuario
│   │   ├── servicioRepository.java          # Repositorio para los servicios
│   │   ├── servidorRepository.java          # Repositorio para servidores
│   │   ├── tipoUsuarioRepository.java         # Repositorio para tipos de usuario
│   │   └── userRepository.java              # Repositorio para la gestión de usuarios
│   │
│   ├──Seeders/                              # Inserts iniciales en la base de datos
│   │   ├── DataBaseSeeder.java              # Inserciones iniciales en la base de datos
│
│   ├──Utils/                          # Otras herramientas
│   │   ├── Verificaciona2f.java       # cache para guardar codigo y correo electrónico
│
├── test/java/com/project/Citas       #Archivos para tests.
│
├── .gitignore                        # Archivos a ignorar por Git
```



Descripción de las carpetas principales
configs/: Contiene las configuraciones de seguridad del sistema como la configuración de cors y el método para encriptación de contraseñas.

Services/: Aquí se encuentran los servicios, que contienen la lógica de negocio de cada entidad (usuarios, productos, compras, etc.). Los controladores reciben las solicitudes de las rutas y se encargan de la interacción con los repositorios para gestionar los datos.

Models/: En esta carpeta se definen los modelos de la base de datos usando JPA.

Controllers: Contiene las rutas que conectan las solicitudes HTTP con los controladores. Cada archivo define los puntos de acceso (endpoints) para interactuar con los diferentes módulos de la aplicación.

Repositories/: Contiene las interfaces que gestionan las operaciones de acceso a la base de datos. Estas interfaces son parte del **Data Access Layer** (capa de acceso a datos) y se usan para interactuar con las entidades de la base de datos a través de consultas, actualizaciones, eliminaciones y búsquedas.

Seeders/: Contiene las inserciones iniciales necesarias en la base de datos para que la aplicación pueda funcionar.

DTO: Contiene los archivos (Data transfer object) que se utilizan para transportar datos sin incluir lógica adicional.

Errors: Contiene las clases que se encargan de manejar los errores y devolver el mensaje al cliente.

Test: Este directorio contiene clases de prueba diseñadas para manejar y capturar errores en la funcionalidad de las citas. Estas pruebas garantizan que, ante posibles fallos o condiciones excepcionales, el sistema devuelve mensajes claros y adecuados al cliente. Las clases también verifican la lógica de validación y aseguran que los mensajes de error sean precisos y estén alineados con los requisitos de usuario, ayudando a mejorar la robustez y la experiencia del cliente al interactuar con los servicios de citas.

Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

Despliegue y Mantenimiento

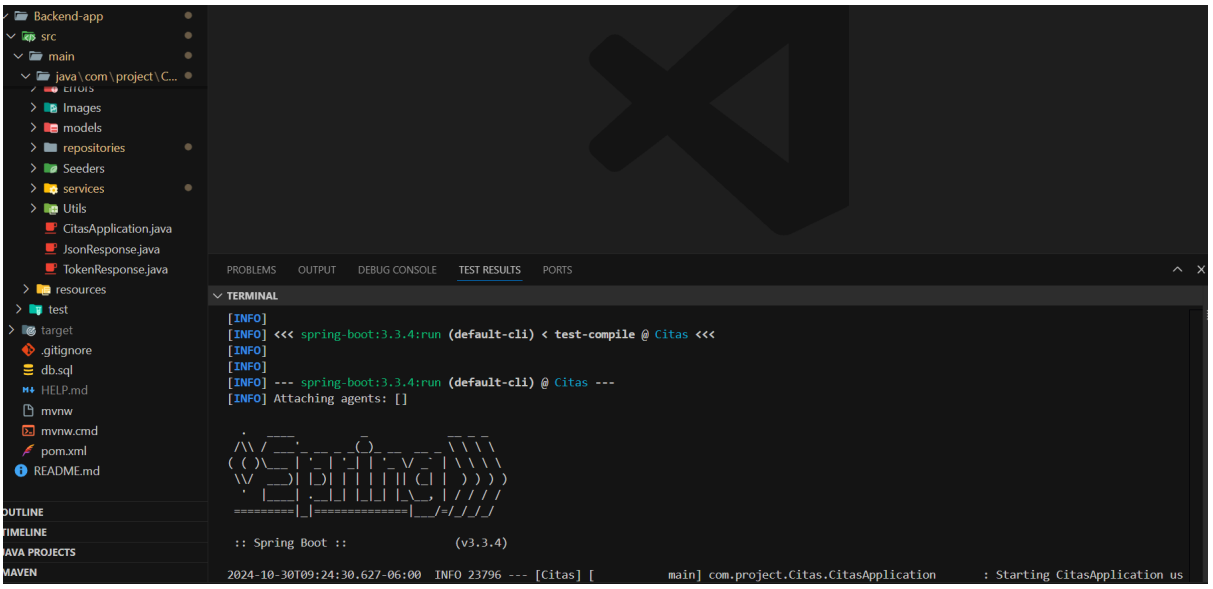
Despliegue

Para el despliegue de la aplicación abra en visual studio code donde clonó el repositorio del proyecto y siga las siguientes instrucciones

Backend

instalar las dependencias con maven
mvn clean install
Correr proyecto:

.\mvnw spring-boot:run

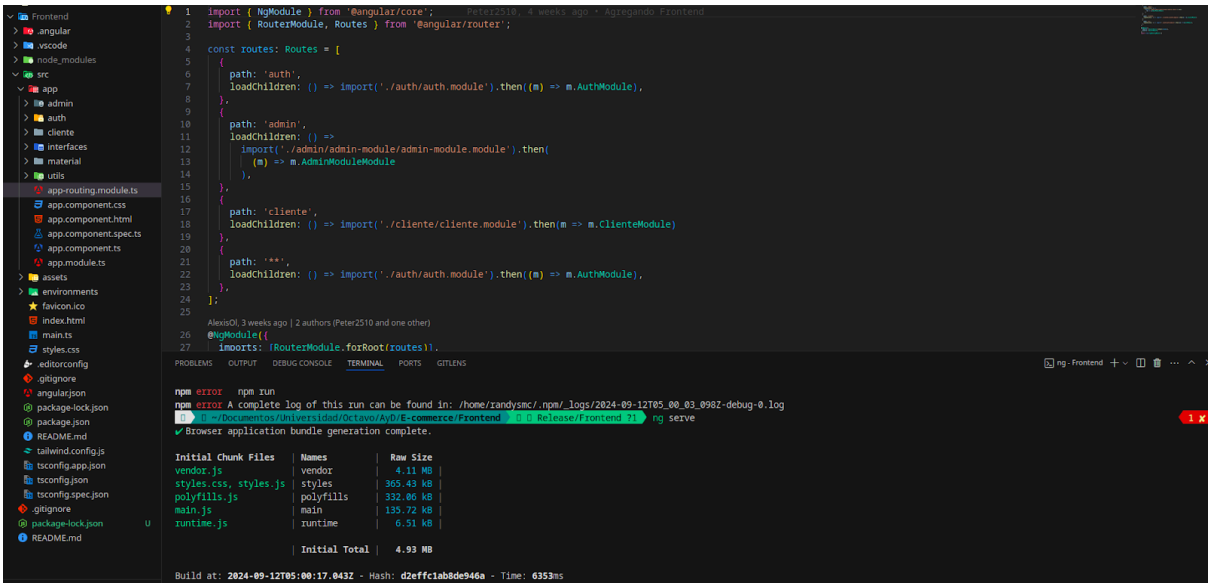


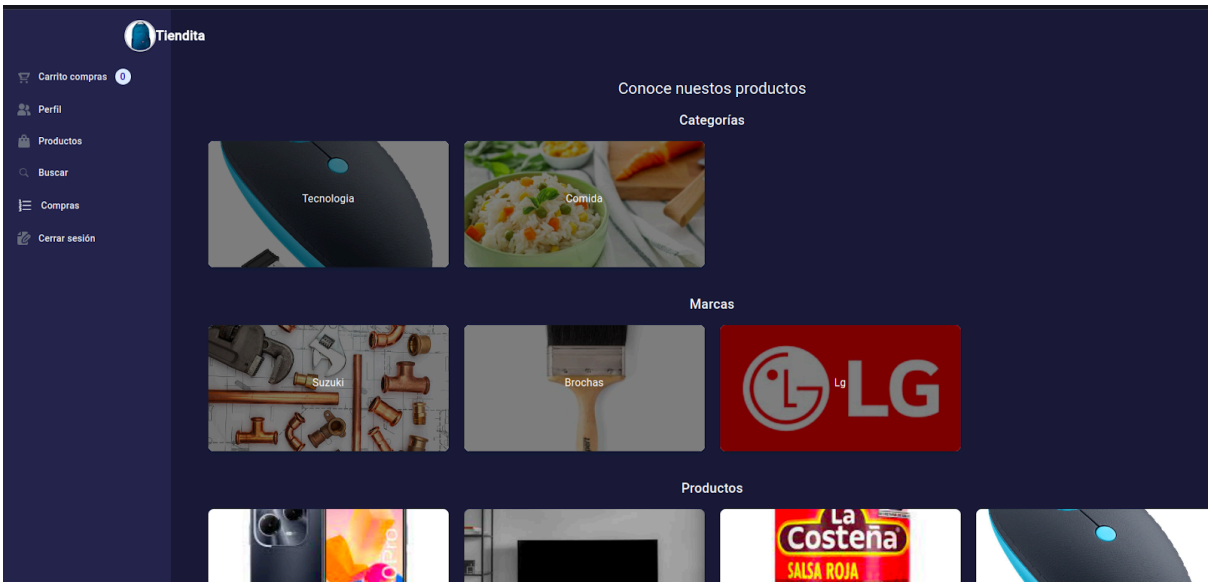
para verificar que está funcionando correctamente puede realizar solicitudes desde postman

Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ingeniería
Ingeniería en Ciencias y Sistemas
Análisis y Diseño de Sistemas 1

Frontend

instalar las dependencias con npm i
ng serve





Se desplegará la aplicación en el http://localhost:4200/


USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Mantenimiento

El sistema está diseñado con una arquitectura modular para facilitar su mantenimiento y futuras actualizaciones. Esta modularidad permite realizar modificaciones y mejoras en componentes individuales sin afectar al resto del sistema. A continuación, se detallan las prácticas recomendadas para el mantenimiento de las diferentes áreas del sistema:

Backend
Controladores, Servicios y modelos: Los controladores y servicios están organizados en módulos específicos, lo que permite actualizarlos o reemplazarlos de manera independiente. Para realizar mantenimiento, se recomienda revisar las dependencias y los servicios asociados para asegurar la compatibilidad.

Rutas y configuración: Las rutas y la configuración del backend se encuentran en módulos dedicados, lo que facilita la modificación de las rutas sin afectar otros componentes. Se debe verificar regularmente la configuración para adaptarse a nuevas necesidades o cambios en la estructura del sistema.

Frontend
Componentes y vistas: El frontend está compuesto por componentes y vistas modulares que pueden ser actualizados o reemplazados sin impactar el funcionamiento global de la aplicación. Para el mantenimiento, es crucial revisar las dependencias de cada componente y asegurar que las actualizaciones se integren correctamente.
Estilos y Recursos: Los estilos y recursos, como archivos CSS y de imagen, están organizados de manera estructurada. Se debe gestionar cuidadosamente cualquier cambio en estos archivos para evitar conflictos con el diseño y la funcionalidad de la aplicación.

Para un mantenimiento eficiente, se recomienda seguir las mejores prácticas de desarrollo, el uso de control de versiones, y la realización de pruebas tras cualquier actualización. De esta manera, se asegura que el sistema continúe funcionando de manera óptima y se puedan integrar nuevas características sin problemas.

Jira

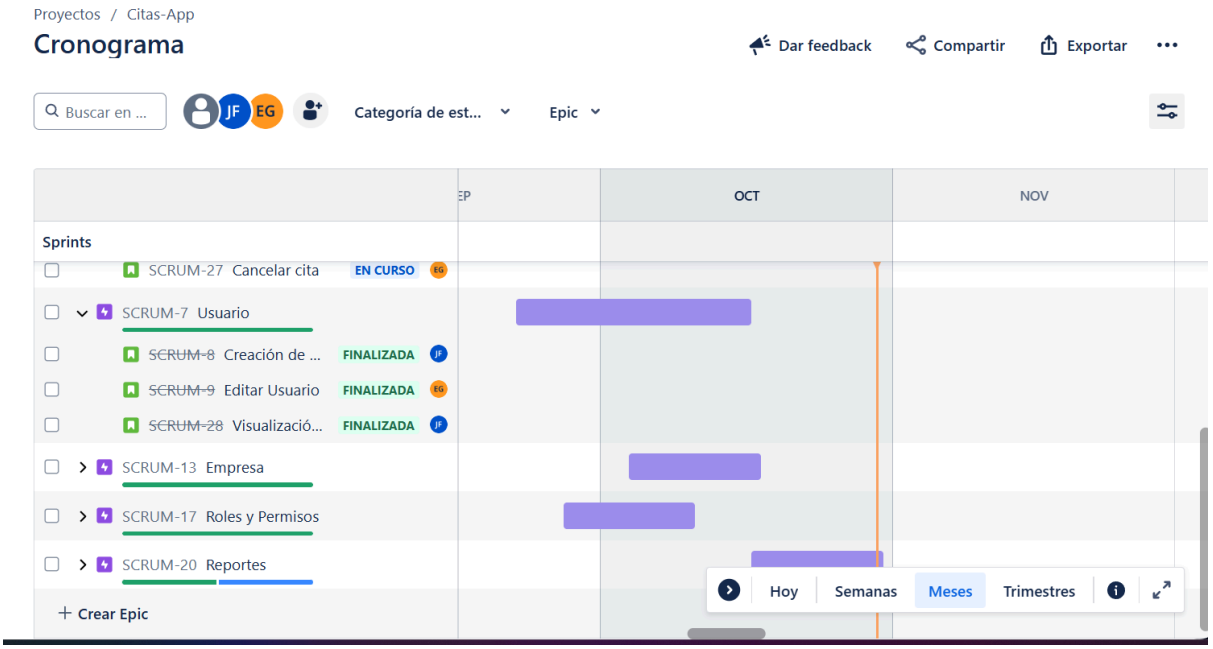
Para organizar y coordinar el trabajo del equipo de desarrollo, se utilizó la aplicación **Jira**. Esta herramienta de gestión de proyectos colaborativa facilitó el seguimiento y la actualización del estado de las tareas asignadas a cada miembro del equipo.

Actualización y Seguimiento:

Cada miembro del equipo actualiza sus historias de usuario y tareas en Jira de manera regular, moviéndolas entre estados según el progreso (por ejemplo, "Por Hacer", "En Progreso", "Hecho"). Esta actualización permite a todos los miembros del equipo ver el estado actual de las tareas y facilita la colaboración en tiempo real.

Visibilidad:

El tablero de Jira proporciona una visión clara y actualizada del progreso del proyecto, permitiendo a todos los miembros del equipo estar al tanto de las tareas y el estado general del proyecto. Las historias de usuario, junto con sus requerimientos de aceptación, garantizan que todos los criterios se cumplan antes de que una tarea se considere completa, mejorando así la calidad del desarrollo.



Git y github

Todo el código fuente del proyecto está alojado en un repositorio en Github, permitiendo la colaboración entre los miembros del equipo. Las ramas siguen el siguiente flujo en Git

Main
Develop
devEstuardo frontend
devJhony backend

Flujo de Trabajo con Ramas
En el repositorio, se empleó una estrategia de ramas para gestionar el desarrollo y mantenimiento del proyecto de manera eficiente.
Se detalla el flujo de trabajo utilizado por el equipo:

Ramas de Desarrollo Individuales (devNombre):

Cada miembro del equipo trabaja en su propia rama de desarrollo, denominada devNombre, para agregar nuevas funciones o realizar modificaciones.

Los miembros del equipo hacen commits y actualizaciones en sus ramas individuales.

Rama de Desarrollo (develop):

La rama develop actúa como la rama de pre-producción donde se integran los cambios de devJhony y devEstuardo una vez que ambos se consideran funcionales al 100%.

Después de realizar las pruebas necesarias en las ramas de liberación y confirmar que todo funciona correctamente, se realiza un merge de Release/Backend y Release/Frontend a la rama develop. Esta rama sirve para la preparación final antes de la publicación en producción.

Rama Principal (main):

La rama main representa la versión estable y en producción del proyecto.
Una vez que los cambios en develop han sido probados y se considera que están listos para producción, se realiza un merge de develop a main. Esta rama refleja la última versión estable del sistema.
Este flujo de trabajo asegura una gestión organizada de las funcionalidades y correcciones, minimizando el riesgo de introducir errores en el entorno de producción y facilitando una colaboración efectiva entre los miembros del equipo.