Centro Universitario de Occidente

Análisis y Diseño 2

Ing. Moises Granados

Segundo Semestre 2024

“Manual Técnico
App-Magazine”

201930693 - Leví Isaac Hernández Sapón
201930699 - Erick Daniel Morales Xicará
201830498 - Oscar Antonio de León Urizar

Índice

MANUAL TÉCNICO

Requisitos del Sistema

El software puede ejecutarse en cualquier sistema operativo, ya que es un proyecto web. Está desarrollado utilizando TypeScript en el frontend y Java (Spring Boot) en el backend. A continuación, se listan los requisitos principales:

- **Frontend**: Angular 17.x+
- **Backend**: Spring Boot 3.x+ con microservicios
- **Base de datos**: MySQL 8.x+
- **Herramientas de desarrollo**: IntelliJ IDEA, Visual Studio Code
- **Navegadores soportados**: Google Chrome, Mozilla Firefox, Microsoft Edge (últimas versiones)

Descripción del Proyecto

Este software es una aplicación web para la **gestión de citas y reservas** en negocios como clínicas, salones de belleza o canchas deportivas. El sistema permite a los negocios organizar su tiempo de atención con los clientes y gestionar su personal. Entre sus principales características se incluyen:

- **Gestión de citas**: Los usuarios pueden ver la disponibilidad de horarios y hacer reservas.
- **Personalización del negocio**: Cada negocio puede configurar su nombre y logo en la interfaz de usuario y facturas.

- **Gestión de roles y permisos**: Diferentes roles para administradores y clientes, con acceso a funcionalidades personalizadas.
- **Asignación de personal**: El sistema permite seleccionar manualmente o asignar de forma automática los empleados o recursos (p. ej., canchas).

Tecnologías Utilizadas

- **Frontend**: Angular 17.x, utilizando TypeScript.
- **Backend**: Spring Boot 3.x con arquitectura de microservicios.
- **Base de datos**: MySQL, con migraciones y consultas optimizadas.
- **Control de versiones**: Git para el control de versiones y colaboración.
- **CI/CD**: Jenkins para la integración y entrega continua, incluyendo pruebas automáticas con JaCoCo para garantizar al menos un 80% de cobertura de código.

Funcionalidades Clave

- **Gestión de citas**: Visualización y gestión de horarios de disponibilidad para agendar citas o reservas.
- **Gestión de usuarios**: Administración de cuentas y roles para clientes y administradores.
- **Personalización del negocio**: Configuración del nombre y logo del negocio, así como la cantidad de empleados o recursos disponibles.
- **Asignación de personal**: El sistema distribuye automáticamente los recursos para balancear la carga de trabajo o permite que el cliente elija el empleado o recurso.
- **Procesamiento de pagos**: Integración con servicios de pago seguros para la gestión de transacciones en línea.

- **Notificaciones**: Actualizaciones en tiempo real sobre el estado de las reservas y modificaciones en la disponibilidad.

Instalación

Frontend (Angular)

Clonar el repositorio del proyecto:

```
git clone https://github.com/9601dani/Revista-AyD2-GrupoA
```

1.

Instalar las dependencias:

```
cd Revista-AyD2-GrupoA/app-frontend
```

```
npm install
```

2.

Levantar el servidor de desarrollo:

```
ng serve
```

3.

Backend (Spring Boot)

Clonar el repositorio del backend:

bash

Copy code

```
git clone https://github.com/9601dani/Revista-AyD2-GrupoA
```

1.

Instalar las dependencias:

```
cd Revista-AyD2-GrupoA/app-backend
```

```
mvn clean install
```

2. Configurar las variables de entorno en un archivo `.env` para conectar la base de datos MySQL, configurando parámetros como `host`, `usuario` y `contraseña`.

Levantar el servidor:

```
mvn spring-boot:run
```

Base de Datos (MySQL)

Instalar MySQL:

```
sudo apt install mysql-server
```

```
sudo systemctl start mysql
```

1. Configurar la base de datos localmente o acceder a la base de datos en la nube con las credenciales proporcionadas.

Seguridad

- **Encriptación de contraseñas**: Todas las contraseñas serán encriptadas usando algoritmos seguros.
- **Autenticación segura**: Se implementará autenticación y recuperación de contraseñas mediante correo electrónico para evitar accesos no autorizados.

Pruebas y CI/CD

- **Pruebas unitarias**: Se utilizarán JUnit y Mockito para pruebas en el backend.
- **Cobertura de código**: Utilizar JaCoCo para asegurar que al menos el 80% del código esté cubierto por pruebas.
- **Integración continua**: Jenkins gestionará el pipeline de CI/CD, ejecutando pruebas y asegurando la entrega continua en cada commit.

Estructura del Proyecto

Backend:

El backend maneja la lógica del negocio y la interacción con la base de datos, implementado con Spring Boot, algunas de las principales carpetas son:

1. Ms-Upload

- **clients**
  - *EmailRestClient*: controla la lógica para la interacción con otro microservicio
- **GsonConfig**:
  - *createGsonWithLocalDateTimeAdapter:* Método estático que crea y devuelve una instancia de Gson personalizada.
- **SecurityConfig**:
  - *securityFilterChain:* Configura la cadena de filtros de seguridad, deshabilitando la protección CSRF, permitiendo el acceso a todas las solicitudes y utilizando un manejo de sesiones sin estado (`STATELESS`), adecuado para aplicaciones basadas en tokens como JWT.
  - *getEncoder*: Proporciona una implementación de `PasswordEncoder` basada en PBKDF2, utilizando una clave secreta obtenida desde la base de datos. Si la clave de configuración no se encuentra, lanza una excepción `SettingNotFoundException`.
- **AuthController**:
  - **helloWorld**: Endpoint GET que responde con un mensaje de prueba para verificar el funcionamiento del controlador.

- **register**: Endpoint POST que permite registrar un nuevo usuario. Recibe los datos de registro en el cuerpo de la solicitud y devuelve una respuesta de autenticación tras la creación del usuario.
- **login**: Endpoint POST para iniciar sesión. Recibe las credenciales de inicio de sesión y devuelve un `AuthResponse` si el inicio de sesión es exitoso.
- **verifyEmail**: Endpoint PUT para verificar la dirección de correo electrónico de un usuario. Recibe un token como parámetro en la URL y devuelve un mensaje de verificación.
- **sendEmailVerification**: Endpoint PUT que envía un correo electrónico de verificación al usuario. Recibe el nombre de usuario o correo electrónico como parámetro.
- **send2FA**: Endpoint POST para enviar un código de verificación de 2FA (autenticación de dos factores) al usuario identificado por su ID.
- **verify2FA**: Endpoint PUT para verificar el código de autenticación de dos factores (2FA) proporcionado por el usuario.
- **sendRecoveryPassword**: Endpoint POST para enviar un correo electrónico con un enlace para la recuperación de la contraseña del usuario.
- **resetPassword**: Endpoint PUT para restablecer la contraseña del usuario. Recibe un token de recuperación y la nueva contraseña.

- **DTO**
  - **Request**
    - AuthRequest
    - Email Request
    - LoginRequest
  - **Response**

        - AuthResponse

- **Exceptions**
    - **EmailVerificationExpiredException**: Se lanza cuando el token de verificación de correo electrónico ha expirado, impidiendo que el usuario complete la verificación.
    - **SettingNotFoundException**: Se lanza cuando no se encuentra una configuración específica en la base de datos, generalmente utilizada en casos de configuración del sistema como claves secretas.
    - **UserNotAllowedException**: Se lanza cuando un usuario intenta realizar una acción para la cual no tiene los permisos necesarios.
    - **UserNotCreatedException**: Se lanza cuando ocurre un error durante la creación de un nuevo usuario, indicando que el proceso de registro falló.
    - **UserNotFoundException**: Se lanza cuando no se puede encontrar un usuario en el sistema, generalmente en el contexto de búsqueda o autenticación.
    - **UserNotVerifiedException**: Se lanza cuando un usuario no ha completado el proceso de verificación (como la verificación de correo electrónico) y no puede realizar ciertas acciones que requieren una cuenta verificada.
- **Models**
    - **CompanySetting**: Representa las configuraciones de la compañía, como claves o ajustes personalizados, que son almacenados en la base de datos y utilizados por la aplicación para modificar comportamientos específicos.
    - **EmailVerification**: Modela la verificación de correo electrónico de los usuarios, incluyendo el token y el estado de verificación. Es utilizado para gestionar el proceso de verificación tras el registro.

- **Role**: Define los diferentes roles que pueden ser asignados a los usuarios del sistema (por ejemplo, administrador, usuario regular), determinando sus permisos y acceso a ciertas funcionalidades.
- **SettingType**: Enumera los diferentes tipos de configuraciones que se pueden almacenar en el sistema. Ayuda a categorizar y estructurar la configuración de la compañía.
- **User**: Representa a un usuario del sistema, conteniendo información básica como nombre, correo electrónico, y datos de autenticación.
- **User2FA**: Gestiona los datos relacionados con la autenticación de dos factores (2FA) para los usuarios, almacenando códigos y estados necesarios para verificar la identidad adicionalmente.
- **UserInformation**: Contiene información adicional del perfil del usuario, como detalles de contacto o datos personales adicionales que pueden no estar directamente relacionados con la autenticación.
- **UserRole**: Modela la relación entre los usuarios y sus roles, permitiendo asignar múltiples roles a un solo usuario, facilitando la administración de permisos.
- **ValueType**: Enumera los diferentes tipos de valores que pueden ser utilizados en las configuraciones, lo que permite una gestión tipada de las propiedades almacenadas en la base de datos.

- **Repositories**
  - **AuthRepository**: Proporciona métodos para interactuar con la base de datos y gestionar la autenticación de los usuarios. Incluye funcionalidades relacionadas con la búsqueda y manipulación de datos de autenticación.

- **CompanySettingRepository**: Permite acceder y modificar las configuraciones de la empresa almacenadas en la base de datos. Utilizado para gestionar ajustes específicos como claves secretas u otros valores importantes de la aplicación.
- **EmailVerificationRepository**: Gestiona las operaciones relacionadas con la verificación de correos electrónicos. Proporciona métodos para buscar y manipular tokens de verificación de correo, así como para comprobar el estado de la verificación.
- **RoleRepository**: Maneja la interacción con la base de datos para las operaciones relacionadas con los roles de usuario. Permite buscar, crear y modificar los diferentes roles asignados a los usuarios.
- **User2FARepository**: Gestiona los datos relacionados con la autenticación de dos factores (2FA) de los usuarios. Proporciona métodos para almacenar y verificar los códigos de 2FA.
- **UserHasRoleRepository**: Maneja la relación entre usuarios y roles, permitiendo gestionar y acceder a los roles asignados a un usuario específico.
- **UserInformationRepository**: Proporciona acceso a la información adicional del usuario que no está relacionada directamente con la autenticación, como los datos personales adicionales que completan el perfil del usuario.

SERVICES

- **AuthService**: Gestiona la lógica de autenticación y registro de usuarios,
  - **register**: Registra un nuevo usuario en el sistema, validando que no exista previamente, generando un hash de la contraseña y asignándole un rol. Además, envía un correo electrónico de verificación al usuario.

    - **login**: Permite a un usuario autenticarse con su nombre de usuario o correo electrónico y contraseña. Valida la verificación y activación del usuario antes de otorgarle acceso.
    - **verifyEmail**: Verifica un token de correo electrónico para activar la cuenta de un usuario.
    - **sendVerification**: Envía un correo electrónico para la verificación de la cuenta.
    - **send2FA**: Envía un código de autenticación de dos factores (2FA) al correo electrónico del usuario.
    - **verify2FA**: Verifica el código de 2FA proporcionado por el usuario para completar la autenticación.
    - **sendRecoveryPassword**: Envía un correo electrónico con un enlace de recuperación de contraseña al usuario.
    - **resetPassword**: Permite al usuario restablecer su contraseña utilizando un token de recuperación.
- **TokenService:**
    - **getToken**: Genera un token JWT para un usuario autenticado, utilizando configuraciones almacenadas como el secreto JWT y el tiempo de expiración.
    - **getInstant**: Calcula la fecha y hora de expiración del token en función de la zona horaria y el tiempo de vida configurado.

2. Ms-Email

- **config**
    - **GsonConfig**: Proporciona la configuración de Gson para serialización y deserialización de objetos JSON, con soporte para tipos personalizados como `LocalDateTime`.
    - **MailSenderConfig**: Configura el servicio de envío de correos electrónicos utilizando `JavaMailSender`, especificando las propiedades necesarias para la conexión al servidor de correo.
- **controllers**
    - **EmailController**: Controlador que expone endpoints relacionados con el envío de correos electrónicos, manejando las solicitudes HTTP para enviar correos con o sin adjuntos generados a partir de plantillas HTML.
- **DTO**
    - **EmailRequest**: Clase de transferencia de datos que encapsula la información requerida para enviar un correo electrónico. Contiene los campos como destinatario, asunto, contenido, y una opción para generar un archivo PDF adjunto.
- **exceptions**
    - **CompanySettingsNotFoundException**: Se lanza cuando no se encuentran las configuraciones de la compañía necesarias para el envío de correos.
    - **EmailNotSendException**: Se lanza cuando ocurre un error al intentar enviar un correo electrónico utilizando el servicio de correo
- **models**

    - **CompanySetting**: Representa una configuración clave-valor relacionada con la compañía, como la plantilla de correo o configuraciones de envío.
    - **SettingType**: Enumera los diferentes tipos de configuraciones posibles, categorizando los valores almacenados en `CompanySetting`.
    - **ValueType**: Enumera los tipos de valores posibles para las configuraciones, como `STRING`, `INTEGER`, o `BOOLEAN`.
- **repositories**
    - **CompanySettingRepository**: Gestiona la interacción con la base de datos para acceder y modificar las configuraciones de la compañía, como las plantillas de correos y otros valores necesarios para la personalización de los correos electrónicos.
- **services**
    - **sendEmail**: Envía un correo electrónico basado en la solicitud `EmailRequest`, utilizando el servicio `JavaMailSender`. Si la solicitud lo requiere, genera un archivo PDF a partir del contenido HTML y lo adjunta al correo.
    - **generatePdfFromHtml**: Método auxiliar que toma contenido HTML y genera un archivo PDF a partir de él utilizando la biblioteca **iText**.

3. **Ms-Img**

- **config**
    - **GoogleCloudStorageConfig**: Proporciona la configuración para integrar el servicio de almacenamiento en la nube de Google Cloud Storage.
- **Controller**
    - **helloWorld**: Endpoint GET que responde con un mensaje de prueba para verificar el funcionamiento del controlador.
    - **uploadImage**: Endpoint POST para subir una sola imagen al almacenamiento en la nube. Recibe la imagen como un archivo `MultipartFile` y devuelve el nombre del objeto subido.
    - **uploadMultipleImages**: Endpoint POST para subir múltiples imágenes al almacenamiento en la nube. Recibe un arreglo de archivos `MultipartFile` y devuelve una lista con los nombres de los objetos subidos.
    - **uploadProfileImage**: Endpoint PUT para subir o actualizar una imagen de perfil. Recibe el archivo de imagen y el nombre de la imagen anterior (si existe) para eliminarla antes de subir la nueva.
    - **uploadResourceImage**: Endpoint POST para subir una imagen asociada a un recurso. Recibe el archivo de imagen y devuelve el nombre del objeto subido.
    - **updateResourceImage**: Endpoint PUT para actualizar una imagen de recurso. Elimina la imagen anterior y sube una nueva en su lugar.
- **DTO**

    - **ResponseString**: Clase de respuesta utilizada para devolver cadenas simples como el nombre del archivo de imagen subido o actualizado.

- **Exceptions**
    - **FileNotCreatedException**: Se lanza cuando ocurre un error durante la creación o almacenamiento de un archivo en Google Cloud Storage.
- **Services**
    - **uploadImage**: Sube una imagen a Google Cloud Storage y genera un nombre único para el archivo. Devuelve el nombre del objeto subido o lanza una excepción si falla.
    - **uploadProfileImage**: Sube o actualiza una imagen de perfil. Si se proporciona un nombre de imagen anterior, la elimina antes de subir la nueva.
    - **deleteImage**: Método auxiliar para eliminar una imagen existente en Google Cloud Storage, lanzando una excepción si no se encuentra o no se puede eliminar.
    - **uploadResourceImage**: Sube una imagen relacionada con un recurso a Google Cloud Storage. Devuelve el nombre del objeto subido o lanza una excepción en caso de error.
    - **updateResourceImage**: Actualiza una imagen de recurso, eliminando la imagen anterior y subiendo una nueva.

4.Ms-User

- **clients**
    - **AuthRestClient**: Controla la lógica para interactuar con el microservicio de autenticación.
    - **EmailRestClient**: Gestiona la lógica para interactuar con el microservicio de envío de correos electrónicos.
- **config**
    - **GsonConfig**: Configura la serialización y deserialización de objetos JSON, permitiendo el manejo de tipos personalizados, como fechas y horas.
- **controllers**
    - **AppointmentController**: Controlador que gestiona las citas para empleados y recursos.
        - **saveAppointment**: Guarda una nueva cita en el sistema.
        - **findByResourceOrEmployee**: Busca citas asociadas a un recurso o empleado en particular.
        - **findAll**: Recupera todas las citas almacenadas en el sistema.
        - **getBill**: Genera y retorna un reporte de facturación de las citas.

      CommentController: Gestiona los comentarios de los usuarios.

        - **getComments**: Recupera todos los comentarios almacenados.
        - **saveComment**: Guarda un nuevo comentario en el sistema.

      CompanySettingController: Controlador que maneja las configuraciones de la compañía.

- **findByKeyName**: Recupera una configuración específica por su nombre clave.
- **findBySettingName**: Busca todas las configuraciones por tipo de ajuste.
- **update**: Actualiza una lista de configuraciones de la compañía.
- **findAllSettingTypes**: Recupera todos los tipos de configuraciones existentes.

EmployeeController: Gestiona los empleados de la compañía.

- **getAllEmployees**: Recupera todos los empleados registrados.
- **createEmployee**: Crea un nuevo empleado con la información proporcionada.

ResourcesController: Controlador para la gestión de recursos de la compañía.

- **getAttributes**: Recupera los atributos de los recursos.
- **createAttribute**: Crea un nuevo atributo de recurso.
- **createResource**: Crea un nuevo recurso en el sistema.
- **getResources**: Recupera todos los recursos disponibles.
- **getResourceById**: Busca un recurso específico por su ID.
- **updateResource**: Actualiza un recurso existente con nueva información.

RoleController: Controlador que gestiona los roles de usuario.

- **getRoles**: Recupera todos los roles existentes en el sistema.
- **saveRole**: Guarda un nuevo rol en el sistema.

ServiceController: Controla la creación y gestión de servicios.

- **getServices**: Recupera todos los servicios ofrecidos por la compañía.
- **getServiceById**: Recupera un servicio específico por su ID.
- **createService**: Crea un nuevo servicio.
- **updateService**: Actualiza la información de un servicio existente.
- **findResourcesByServicesId**: Recupera los recursos disponibles para un servicio específico.
- **findEmployeesByServicesId**: Recupera los empleados asociados a un servicio.

UserController: Gestiona los usuarios y sus perfiles.

- **getPages**: Recupera las páginas asignadas a un usuario específico.
- **getById**: Recupera la información completa de un usuario por su ID.
- **updateProfile**: Actualiza el perfil de un usuario.
- **getInfo**: Obtiene la información de la imagen de perfil de un usuario.
- **updateImageProfile**: Actualiza la imagen de perfil de un usuario.
- **set2fa**: Activa la autenticación de dos factores (2FA) para un usuario.
- **getMyAppointments**: Recupera todas las citas asociadas a un usuario.
- **getPopularity**: Genera un reporte de popularidad basado en las interacciones con los servicios.
- **getUsersByRole**: Recupera un reporte de usuarios agrupados por su rol.
- **getPopularityResources**: Genera un reporte de popularidad de los recursos disponibles.

- **dto / request**

    - **AppointmentRequest**: Define la estructura de la solicitud para crear o actualizar una cita.
    - **AuthRequest**: Estructura de la solicitud de autenticación.
    - **CommentRequest**: Define la estructura de la solicitud para agregar comentarios.
    - **CompanySettingRequest**: Solicitud para actualizar las configuraciones de la compañía.
    - **CreateEmployeeRequest**: Solicitud para crear un nuevo empleado.
    - **ResourceRequest**: Define la estructura para crear o actualizar un recurso.
    - **ServiceRequest**: Solicitud para crear o modificar un servicio.
    - **UserAllRequest**: Estructura de la solicitud para actualizar los datos de un usuario.
- **dto / response**
    - **AppointmentResponse**: Respuesta que encapsula los detalles de una cita.
    - **AuthResponse**: Respuesta de la autenticación del usuario.
    - **BillReportResponse**: Respuesta con detalles del reporte de facturación.
    - **CommentResponse**: Respuesta que encapsula los detalles de un comentario.
    - **CompanySettingResponse**: Estructura que devuelve los detalles de una configuración de la compañía.
    - **CreateEmployeeResponse**: Respuesta con los detalles de un empleado creado.
    - **EmployeeResponse**: Respuesta que contiene la información de un empleado.
    - **ModuleResponse**: Define la estructura de la respuesta para los módulos de la aplicación.

    - **PageResponse**: Respuesta que contiene la información de una página del sistema.
    - **PopularityResponse**: Estructura de la respuesta que devuelve detalles sobre la popularidad de servicios y recursos.
    - **ResourceResponse**: Estructura que contiene los detalles de un recurso específico.
    - **ServiceResponse**: Respuesta con los detalles de un servicio.
    - **UserAllResponse**: Respuesta que contiene la información completa de un usuario.
- **Exceptions**
    - **CompanySettingNotFoundException**: Se lanza cuando no se encuentra una configuración específica de la compañía.
    - **EmployeeNotFoundException**: Excepción lanzada cuando no se encuentra un empleado.
    - **ResourceNotFoundException**: Se lanza cuando no se encuentra un recurso solicitado.
    - **ServiceNotSaveException**: Excepción lanzada cuando no se puede guardar o actualizar un servicio.
    - **UserNotFoundException**: Se lanza cuando no se encuentra un usuario en el sistema.
- **Models**
    - **Appointment**: Representa una cita en el sistema, vinculada a un usuario, recurso y/o empleado, con detalles como tiempo de inicio, fin, y costo total.
    - **AppointmentHasService**: Relaciona las citas con los servicios ofrecidos, almacenando detalles como el precio y tiempo estimado.

- **Attribute**: Define un atributo asociado a un recurso en el sistema.
- **Bill**: Representa una factura generada para una cita, conteniendo detalles de los servicios proporcionados, el usuario, y los montos.
- **Comment**: Modela los comentarios dejados por los usuarios sobre citas o servicios.
- **CompanySetting**: Representa configuraciones clave-valor de la empresa, como el logotipo, dirección, o moneda utilizada en las facturas.
- **Employee**: Modela un empleado de la compañía, con información como nombre, fecha de nacimiento, y asociación con un usuario del sistema.
- **Module**: Define los módulos que un usuario puede acceder en el sistema, relacionados con su rol.
- **Page**: Representa las páginas accesibles dentro del sistema, cada una vinculada a un módulo.
- **Resource**: Modela un recurso dentro del sistema (por ejemplo, una sala o equipo), asociado a atributos y disponible para reservas en citas.
- **Role**: Define los roles que un usuario puede tener dentro del sistema, determinando sus permisos.
- **Service**: Representa un servicio que la compañía ofrece, con detalles como precio, duración aproximada y disponibilidad.
- **SettingType**: Enumera los tipos de configuraciones disponibles en el sistema.
- **User**: Representa un usuario en el sistema, con detalles de autenticación y perfil.
- **UserInformation**: Almacena información adicional sobre el usuario, como imagen de perfil, NIT, DPI y número de teléfono.

- **UserRole**: Relaciona usuarios con roles dentro del sistema, permitiendo que un usuario tenga múltiples roles.
- **ValueType**: Define los tipos de valores permitidos en las configuraciones de la compañía, como cadenas, números, o booleanos.

- **Repositories**
  - **AppointmentRepository**: Proporciona acceso y métodos para gestionar las citas en la base de datos.
  - **AppointmentServiceRepository**: Gestiona la relación entre citas y servicios, permitiendo almacenar y recuperar los servicios asociados a una cita.
  - **AttributeRepository**: Proporciona métodos para gestionar los atributos de los recursos en la base de datos.
  - **BillRepository**: Administra el acceso y las operaciones relacionadas con las facturas generadas en el sistema.
  - **CommentRepository**: Gestiona los comentarios de los usuarios, permitiendo su almacenamiento y recuperación.
  - **CompanySettingRepository**: Proporciona acceso a las configuraciones de la compañía en la base de datos, como plantillas de correos o configuraciones de facturación.
  - **EmployeeHasServiceRepository**: Administra la relación entre empleados y los servicios que pueden proporcionar.
  - **EmployeeRepository**: Gestiona el acceso y las operaciones relacionadas con los empleados de la compañía.
  - **ModuleRepository**: Gestiona el acceso a los módulos dentro del sistema, proporcionando información sobre las páginas disponibles para los usuarios.

- **ResourceHasAttributeRepository**: Administra la relación entre recursos y atributos, permitiendo asociar múltiples atributos a un recurso.
- **ResourceHasServiceRepository**: Gestiona la relación entre recursos y servicios, permitiendo asignar servicios a recursos.
- **ResourceRepository**: Proporciona métodos para almacenar y recuperar recursos disponibles en la compañía.
- **RoleRepository**: Gestiona los roles de los usuarios, permitiendo su almacenamiento y asignación a los usuarios.
- **ServiceRepository**: Administra los servicios ofrecidos por la compañía, almacenando y recuperando detalles sobre ellos.
- **SettingTypeRepository**: Proporciona acceso a los tipos de configuraciones disponibles en el sistema.
- **UserHasRoleRepository**: Gestiona la relación entre usuarios y roles, permitiendo asignar roles a los usuarios.
- **UserInformationRepository**: Almacena y gestiona la información adicional de los usuarios.
- **UserRepository**: Proporciona acceso a los datos de los usuarios en la base de datos, incluyendo detalles de autenticación y perfil.

- **Services**
  - **AppointmentService**: Gestiona la lógica relacionada con las citas, desde su creación hasta la generación de facturas. Proporciona métodos para encontrar citas por recurso o empleado, y envía correos electrónicos con las facturas.
  - **CommentService**: Proporciona métodos para recuperar y almacenar comentarios en el sistema, incluyendo la creación de nuevos comentarios.

- **CompanySettingService**: Gestiona las configuraciones de la compañía, permitiendo su actualización y búsqueda por nombre de clave o tipo de configuración.
- **EmployeeService**: Gestiona la creación, actualización y recuperación de empleados, incluyendo la asignación de roles a los empleados creados.
- **ResourceService**: Administra los recursos del sistema, proporcionando métodos para crear, actualizar y recuperar recursos, así como asignar atributos a ellos.
- **RoleService**: Proporciona métodos para gestionar los roles del sistema, permitiendo la creación de nuevos roles y la recuperación de los existentes.
- **ServiceService**: Gestiona la creación y actualización de los servicios ofrecidos por la compañía, permitiendo la asignación de empleados y recursos a cada servicio.
- **UserService**: Proporciona métodos para gestionar el perfil del usuario, incluyendo la actualización de información personal, imagen de perfil y la activación del 2FA (autenticación de dos factores). También permite la recuperación de citas y reportes de popularidad.

Frontend

El frontend está implementado con **Angular 17**, ofreciendo una interfaz visual responsiva para los usuarios. A continuación se describen las principales carpetas y rutas del proyecto.

Componentes

Los componentes están organizados dentro de src/app/components, cada carpeta representa un componente individual o un conjunto de componentes relacionados. Algunos de los componentes más importantes son:

admin/

Esta carpeta contiene los componentes relacionados con la administración de recursos, empleados, servicios y la generación de reportes:

- **add-employee**: Componente para agregar nuevos empleados a la base de datos.
- **add-resource**: Componente para añadir recursos disponibles para la reserva.
- **add-service**: Componente para la creación de nuevos servicios en el sistema.
- **custom-reports**: Componente para generar reportes personalizados.
- **report-bill**: Muestra las facturas generadas para los servicios.
- **show-reports**: Componente que lista los reportes disponibles para su visualización.
- **view-employee**: Permite ver la información de los empleados.
- **view-resource**: Muestra los recursos registrados.
- **view-service**: Lista los servicios disponibles.

appointments/reserve

- **reserve**: Componente que permite reservar citas o servicios.

commons/

Carpeta que agrupa componentes reutilizables o relacionados con la navegación y vistas comunes:

- **home**: Página principal de la aplicación.
- **login**: Componente de autenticación de usuarios.
- **navbar**: Barra de navegación para usuarios autenticados.
- **not-found**: Componente que se muestra cuando no se encuentra una página (Error 404).
- **recovery-password**: Componente para la recuperación de contraseña.
- **reset-password**: Componente para restablecer la contraseña de un usuario.
- **verification-email**: Componente para verificar la dirección de correo electrónico.
- **verification2fa**: Componente que permite verificar el segundo factor de autenticación (2FA).

options/company-settings

- **company-settings**: Componente para la configuración de la empresa, donde se ajustan valores clave como moneda, logotipo, etc.

user/

- **my-appointments**: Componente para que los usuarios visualicen sus citas reservadas.
- **profile**: Componente que permite a los usuarios editar la información de su perfil.

Servicios (services/)

Los servicios manejan la lógica de negocio e interacciones con el backend y otras funcionalidades. Algunos de los servicios más importantes son:

- **auth.service.ts**: Gestiona la autenticación de usuarios, incluyendo el inicio de sesión, registro y recuperación de contraseñas.
- **img.service.ts**: Maneja la lógica para la subida y recuperación de imágenes (por ejemplo, perfiles de usuario o imágenes de recursos).
- **local-storage.service.ts**: Administra el almacenamiento de información en el navegador, como los tokens de autenticación.
- **user.service.ts**: Gestiona la información de los usuarios, como la edición de perfiles y la recuperación de citas.

Directivas

Las directivas permiten aplicar comportamientos personalizados a los elementos del DOM:

- **auth-logo.directive.ts**: Directiva que gestiona el logotipo según el estado de autenticación.
- **not-profile.directive.ts**: Directiva para gestionar la imagen de perfil cuando no se ha establecido una.

Modelos (models/)

Los modelos son representaciones de los datos que se manejan en la aplicación:

- **CompanySetting.model.ts**: Define los parámetros de configuración de la empresa.

- **Module.model.ts**: Modela los módulos a los que los usuarios tienen acceso.
- **Page.model.ts**: Modela las páginas disponibles en la aplicación.

Rutas

El archivo de rutas define la navegación entre los diferentes componentes de la aplicación. Aquí algunas de las rutas más importantes:

- **/home**: Página principal.
- **/login**: Página de inicio de sesión.
- **/verify-email/**

  : Verificación de correo electrónico mediante un token.
- **/recovery-password**: Página para recuperar la contraseña.
- **/reset-password/**

  : Restablecimiento de la contraseña mediante un token.
- **/verify-2fa**: Verificación del segundo factor de autenticación (2FA).

Dentro de las rutas protegidas, tenemos las siguientes secciones:

Opciones

- **/options/company-settings**: Configuración de la empresa.

Edición de perfil

- **/edit/profile**: Edición del perfil del usuario.

Servicios

- **/services/add-service**: Añadir un nuevo servicio.

- **/services/edit-service/**

  : Editar un servicio existente.

- **/services/show-services**: Mostrar todos los servicios.

Empleados

- **/employees/add-employee**: Añadir un nuevo empleado.
- **/employees/show-employee**: Mostrar todos los empleados registrados.

Recursos

- **/resources/add-resource**: Añadir un nuevo recurso.
- **/resources/show-resources**: Mostrar todos los recursos disponibles.
- **/resources/add-resource/**

  : Editar un recurso existente.

Citas

- **/appointments/reserve**: Reservar una cita.
- **/appointments/my-appointments**: Mostrar todas las citas del usuario.

Reportes

- **/reports/custom-reports**: Generar reportes personalizados.
- **/reports/show-reports**: Mostrar los reportes generados.
- **/reports/report-bill**: Mostrar la facturación de servicios.

Cualquier ruta no encontrada redirige a la página **NotFoundComponent**.

Base de Datos (MariaDB)

Descripción General

El sistema utiliza una base de datos relacional en **MariaDB** para almacenar la información necesaria para la gestión de usuarios, recursos, servicios, roles y otros aspectos clave del sistema. El modelo sigue un diseño relacional con varias tablas interconectadas que representan entidades como usuarios, servicios, empleados y más.

Tablas Principales

1. user

Descripción: Almacena la información de los usuarios registrados en el sistema.

- **id**: Identificador único del usuario.
- **email**: Correo electrónico del usuario.
- **username**: Nombre de usuario.
- **password**: Contraseña encriptada.
- **auth_token**: Token de autenticación utilizado para la gestión de sesiones.
- **is_activated**: Indica si la cuenta está activada.
- **is_verified**: Indica si el correo electrónico ha sido verificado.
- **is_2FA**: Indica si el usuario tiene activada la autenticación de dos factores.

Relaciones:

- Relacionado con **user_information**, **user_2fa**, **employee**, **appointment**, **comment** y otras.
- Relacionado con **role** a través de la tabla **user_has_role**.

2. appointment

Descripción: Registra las citas que los usuarios han reservado.

- **id**: Identificador único de la cita.
- **FK_User**: Clave foránea que conecta con la tabla **user**.
- **FK_Resource**: Clave foránea que conecta con la tabla **resource** (si se utiliza un recurso para la cita).
- **FK_Employee**: Clave foránea que conecta con la tabla **employee** (si un empleado participa en la cita).
- **total**: Precio total de la cita.
- **start_time**: Hora de inicio de la cita.
- **end_time**: Hora de fin de la cita.
- **state**: Estado de la cita (PENDING, CONFIRMED, CANCELED).

Relaciones:

- Relacionado con **employee**, **resource**, **service** y **bill**.

3. bill

Descripción: Representa una factura generada para una cita.

- **id**: Identificador único de la factura.
- **appointment_id**: Clave foránea que conecta con la tabla **appointment**.
- **nit**: NIT del cliente para la factura.
- **name**: Nombre del cliente en la factura.
- **address**: Dirección del cliente.
- **description**: Descripción de los servicios facturados.
- **price**: Total sin impuestos.
- **tax**: Impuesto aplicado a la factura.
- **advancement**: Adelanto realizado para la cita.

Relaciones:

- Relacionado con **appointment**.

4. employee

Descripción: Almacena los datos de los empleados del sistema.

- **id**: Identificador único del empleado.
- **first_name**: Nombre del empleado.
- **last_name**: Apellidos del empleado.
- **date_of_birth**: Fecha de nacimiento.
- **FK_User**: Clave foránea que conecta con la tabla **user**.

Relaciones:

- Relacionado con **appointment** para citas asignadas a empleados.
- Relacionado con **service** a través de **employee_has_service**.

5. service

Descripción: Registra los servicios ofrecidos por la empresa.

- **id**: Identificador único del servicio.
- **name**: Nombre del servicio.
- **description**: Descripción del servicio.
- **price**: Precio del servicio.
- **time_aprox**: Tiempo aproximado del servicio.
- **is_available**: Indica si el servicio está disponible.

Relaciones:

- Relacionado con **appointment** a través de **appointment_has_service**.

- Relacionado con **resource** a través de **resource_has_service**.

Tablas Relacionales

1. appointment_has_service

Descripción: Relaciona las citas con los servicios ofrecidos.

- **FK_Appointment**: Clave foránea que conecta con la tabla **appointment**.
- **FK_Service**: Clave foránea que conecta con la tabla **service**.
- **price**: Precio del servicio en la cita.
- **time_aprox**: Tiempo aproximado del servicio.

2. resource_has_service

Descripción: Relaciona los recursos con los servicios que pueden proporcionar.

- **FK_Resource**: Clave foránea que conecta con la tabla **resource**.
- **FK_Service**: Clave foránea que conecta con la tabla **service**.

3. user_has_role

Descripción: Relaciona los usuarios con los roles que tienen asignados.

- **FK_User**: Clave foránea que conecta con la tabla **user**.
- **FK_Role**: Clave foránea que conecta con la tabla **role**.

4. role_has_page

Descripción: Relaciona los roles con las páginas que pueden acceder.

- **FK_Role**: Clave foránea que conecta con la tabla **role**.
- **FK_Page**: Clave foránea que conecta con la tabla **page**.

- **can_create**: Indica si el rol puede crear contenido en la página.
- **can_edit**: Indica si el rol puede editar contenido en la página.
- **can_delete**: Indica si el rol puede eliminar contenido en la página.

Tablas Auxiliares

1. role

Descripción: Define los roles que pueden tener los usuarios del sistema (por ejemplo, Administrador, Cliente).

- **id**: Identificador único del rol.
- **name**: Nombre del rol.
- **description**: Descripción del rol.

2. resource

Descripción: Almacena los recursos que pueden ser utilizados o reservados en el sistema.

- **id**: Identificador único del recurso.
- **name**: Nombre del recurso.
- **image**: Imagen asociada al recurso.

3. attribute

Descripción: Define los atributos disponibles para los recursos.

- **id**: Identificador único del atributo.
- **name**: Nombre del atributo.
- **description**: Descripción del atributo.

4. comment

Descripción: Almacena los comentarios realizados por los usuarios.

- **id**: Identificador único del comentario.
- **FK_User**: Clave foránea que conecta con la tabla **user**.
- **comment**: Texto del comentario.
- **value**: Valoración del servicio/producto (de 1 a 5).

Relaciones Clave

- **user - role**: Un usuario puede tener uno o más roles en el sistema.
- **appointment - user**: Las citas están asociadas a usuarios.
- **appointment - service**: Cada cita puede incluir múltiples servicios.
- **appointment - resource**: Las citas pueden estar asociadas a un recurso específico.
- **bill - appointment**: Cada factura está relacionada con una cita específica.

Ejecución

Frontend

Para levantar el servidor de desarrollo:

```
ng serve
```

Base de Datos

Para asegurarse de que el servidor de **MariaDB** está corriendo:

```
sudo systemctl start mariadb
```

Este es el diseño de la base de datos actualizado, reflejando la migración a **Spring Boot** y el uso de las tablas y relaciones proporcionadas en el script SQL.

Créditos

Este manual ha sido creado con el propósito de proporcionar una guía técnica completa para la implementación y uso del software de **App-Magazine** desarrollado por **CodenBugs**. El contenido cubre la configuración del sistema, especificaciones técnicas, arquitectura del software, instalación, mantenimiento, solución de problemas, y mejores prácticas para optimizar su desempeño.

Equipo de Desarrollo - CodenBugs

- **Leví Isaac Hernández Sapón** (201930693)

  Estudiante de Ingeniería en Sistemas
- **Erick Daniel Morales Xicará** (201930699)

  Estudiante de Ingeniería en Sistemas
- **Oscar Antonio de Leon Urizar** (201830498)

  Estudiante de Ingeniería en Sistemas

Este manual fue preparado como parte del curso de **Análisis y Diseño 2**, bajo la guía del **Ing. Moises Granados** en el **Centro Universitario de Occidente** durante el segundo semestre de 2024.

Agradecemos su confianza en nosotros para la implementación de esta solución. Para cualquier consulta técnica o soporte adicional, no dude en contactar a nuestro equipo de atención técnica.

Manual Técnico - App-Magazine