AppGestionCitasFront

This project was generated with Angular CLI version 16.2.16.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The application will automatically reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory.

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via a platform of your choice. To use this command, you need to first add a package that implements end-to-end testing capabilities.

Further help

To get more help on the Angular CLI use `ng help` or go check out the Angular CLI Overview and Command Reference page.

Sistema de Gestión de Citas con Angular

Descripción general

Este proyecto es un sistema de gestión de citas para un entorno profesional, desarrollado en Angular. Permite a usuarios con diferentes roles (administrador y profesionales) gestionar y reservar citas con especialistas, filtrando por especialidad y profesional, y visualizando un calendario con los huecos disponibles. El sistema utiliza localStorage para persistencia de datos y simula autenticación básica.

Objetivos

Implementar un sistema de login con roles y gestión de sesión persistente.

Permitir la visualización y filtrado de profesionales disponibles.

Mostrar un calendario dinámico con los huecos disponibles para reservar citas, tomando en cuenta la duración del servicio, horario laboral y citas previas.

Gestionar la reserva de citas con validación para evitar solapamientos.

Utilizar Angular para organizar la aplicación con componentes reutilizables y servicios que gestionan la lógica de negocio.

Persistir datos localmente mediante localStorage para mantener estado entre recargas sin necesidad de backend .

Arquitectura y organización

La aplicación está organizada principalmente en:

Modelos: Definen la estructura de datos para usuarios, especialistas, servicios y citas.

Servicios: Encapsulan la lógica de negocio, incluyendo autenticación (AuthService), gestión de especialistas, servicios, y reservas.

Componentes: Se encargan de la interfaz de usuario, incluyendo vistas para login, listado y filtrado de profesionales, calendario y reservas.

Almacenamiento: Uso de localStorage para persistencia de datos y mantenimiento de sesión.

Modelos principales:

Usuario:
interface Usuario {
id: number;
email: string;
password: string;
rol: 'admin' | 'profesional';
especialista?: Especialista;
}

Especialista:
interface Especialista {
id: number;
nombre: string;
apellidos: string;
email: string;
telefono: string;
foto_url: string;
servicios_asignados: Servicio[];
horario: Horario[];
}
interface Horario {
dia: string[];
hora_inicio: string;
hora_fin: string;
}

Cita:
interface Cita {
id: number;
servicio: Servicio;
especialista: Especialista;
cliente_nombre: string;
cliente_telefono: string;
cliente_email?: string;
notas_cliente?: string;
notas_profesional?: string;
inicio: Date;
fin: Date;
estado: 'pendiente' | 'terminada' | 'cancelada';}

```
Servicio:
interface Servicio {
```

id: number;
nombre: string;
duracion: number; // en minutos
descripcion?: string;
}

Servicios principales

-AuthService

Gestiona la autenticación y sesión de usuarios.

usuarios: Usuario[]: Array con usuarios disponibles, incluyendo admin y profesionales.

usuarioActivo?: Usuario: Usuario que ha iniciado sesión.

Funciones clave:

cargarUsuariosProfesionales(): Carga usuarios profesionales desde EspecialistasService para sincronizar usuarios.

getUsuarios(): Devuelve todos los usuarios.

getUsuarioPorEmail(email: string): Busca un usuario por email.

login(email: string, password: string): Intenta autenticar con email y contraseña, devuelve el usuario si coincide.

setUsuarioActivo(usuario: Usuario): Guarda el usuario activo en memoria y en localStorage para persistencia.

getUsuarioActivo(): Obtiene el usuario activo, ya sea desde memoria o localStorage.

logout(): Cierra la sesión eliminando usuario activo de memoria y localStorage.

-EspecialistasService

Provee la lista de especialistas profesionales con sus datos y horarios. Esta información es base para generar usuarios profesionales y para mostrar disponibilidad.

crear, editar o eliminar especialistas

-Servicios de Citas

Gestionan las reservas y verifican disponibilidad.

Comprobación de solapamientos basados en:

Horario laboral del especialista.

Duración del servicio seleccionado.

Citas ya reservadas.

Lógica para mostrar un calendario con huecos disponibles.

Permiten reservar una cita solo si el hueco está libre.

Funcionalidades detalladas

Gestión de usuarios y autenticación

·Usuario administrador creado por defecto con email admin@sirona.com y contraseña 12345.

·Usuarios profesionales cargados automáticamente desde los especialistas.

·Login con verificación de email y contraseña.

·Persistencia de sesión en localStorage para mantener sesión tras recargas.

·Logout que elimina sesión activa.

Visualización y filtrado de profesionales

·Listado claro con nombre, especialidad y foto.

·Filtros para seleccionar profesionales por nombre o especialidad.

·Datos cargados desde EspecialistasService.

Calendario de disponibilidad

·Visualización dinámica de los huecos disponibles según:

·Horario laboral del profesional.

·Duración del servicio que el usuario quiere reservar.

·Reservas ya existentes para evitar conflictos.

·Permite navegar por días y horas para seleccionar citas.

Reserva de citas

·Selección de servicio y profesional.

·Validación que solo permite reservar si el hueco es suficiente para la duración del servicio.

·Al reservar, la cita queda almacenada y bloquea ese horario para nuevas reservas.

Uso de LocalStorage

·Se almacena el usuario activo para mantener sesión.

·Se pueden almacenar citas para persistir reservas.

·Esta estrategia simplifica el desarrollo y pruebas sin necesidad de backend.

ACCESOS:

- rol administrador:
  email: admin@sirona.com
  contraseña: 12345
- rol profesional:
  email: nombre.apellido@sirona.com
  contraseña:12345