**Backend (.NET Core + EF Core + SQLite)**

**Tecnologías:**

* .NET 7 / ASP.NET Core Web API
* Entity Framework Core
* SQLite
* JWT (JSON Web Token) para autenticación
* Swagger para documentación de API

**Funcionalidades principales:**

1. **Autenticación:**
   * Endpoint POST /api/auth/login para login.
   * Se valida el usuario y contraseña desde la base de datos Usuarios.
   * Se genera un token JWT válido por 2 horas y se devuelve al frontend.
2. **Gestión de tickets:**
   * Endpoint GET /api/tickets para obtener tickets.
   * Endpoint POST /api/tickets para crear un ticket.
   * Endpoint PUT /api/tickets/{id} para actualizar un ticket.
   * Endpoint DELETE /api/tickets/{id} para eliminar un ticket.
   * Filtros opcionales por estado y prioridad.
3. **Base de datos:**
   * SQLite (ticketing.db) con tablas:
     + Usuarios: contiene usuario y contraseña (ejemplo: admin / 123456).
     + Tickets: campos Titulo, Descripcion, Prioridad, Estado, UsuarioAsignado, FechaCreacion.
4. **Seguridad y CORS:**
   * CORS habilitado para http://localhost:4200.
   * JWT obligatorio para acceder a los endpoints de tickets ([Authorize]).

**Frontend (Angular 20)**

**Tecnologías:**

* Angular 20 (standalone components)
* Reactive Forms
* Angular Router con lazy loading
* HttpClient para consumir API REST
* Interceptor para enviar JWT en los headers (Authorization: Bearer <token>)

**Funcionalidades principales:**

1. **Login:**
   * Componente LoginComponent con formulario reactivo.
   * Llama al AuthService.login() para autenticarse.
   * Guarda el token JWT en localStorage.
   * Redirige a /tickets si el login es exitoso.
2. **Tickets:**
   * Lazy loaded module TicketsModule con rutas:
     + /tickets → TicketListComponent (lista de tickets).
     + /tickets/nuevo → TicketFormComponent (crear ticket).
   * Servicio TicketService para CRUD de tickets.
   * JWT enviado automáticamente en cada petición gracias al AuthInterceptor.
3. **Routing:**
   * AppRoutingModule maneja login y carga perezosa (lazy load) de tickets.
   * TicketsRoutingModule usa RouterModule.forChild() para rutas internas.

**Pruebas realizadas**

**Backend**

* **Postman**:
  + Verificación de login (POST /api/auth/login).
  + Acceso a tickets (GET /api/tickets, POST, PUT, DELETE).
  + Validación de autorización con JWT (401 si no hay token).
* **SQLite Studio / CLI**:
  + Inspección de base de datos ticketing.db.
  + Inserción de tickets de prueba.

**Frontend**

* **Navegador**:
  + Comprobación de login y redirección a /tickets.
  + Visualización de tickets, creación, actualización y eliminación mediante UI.
  + Verificación de envío del JWT con **Interceptor**.
* **Angular Unit Tests** (Jasmine + TestBed + HttpClientTestingModule + RouterTestingModule):
  + **TicketListComponent**:
    - Se crea correctamente.
    - Carga tickets al iniciar (mockeados).
    - Navegación al formulario de creación de tickets.
    - Eliminación de tickets y recarga de lista.
  + Uso de **spies** para simular TicketService y evitar llamadas reales al backend.
  + Simulación de confirmaciones (window.confirm).