MANUAL DE INSTALACIÓN (BACK-END)

Citas Médicas Teruel

Versión
2.0

Grupo 6 gestión de citas médicas

Tabla de contenido

Tabla de contenido

1. Introducción

El proyecto Appointments-Project-Back es un sistema backend desarrollado en .NET 8 (LTS) para la gestión integral de citas médicas. Su arquitectura está diseñada para ser agnóstica a la infraestructura, permitiendo despliegues tanto en servidores tradicionales (On-Premise) como en entornos contenerizados mediante Docker.

Este manual detalla los procedimientos técnicos para la configuración del entorno de desarrollo, la instalación de dependencias, la ejecución de migraciones de base de datos y el despliegue en entornos productivos utilizando prácticas de CI/CD con Jenkins.

1.1. Características Clave de Infraestructura

- Arquitectura: Basada en Capas y Microservicios (lógica desacoplada).
- Persistencia: Soporte multi-motor (SQL Server, PostgreSQL, MySQL) gestionado por EF Core 9.
- Tiempo Real: Comunicación WebSocket vía SignalR para bloqueos de agenda concurrentes.
- Containerización: Orquestación de servicios (API, BD, Redis, Proxy) mediante Docker Compose.

2. Requisitos del Sistema

Para garantizar el correcto funcionamiento y compilación de la solución, el entorno host debe cumplir con las siguientes especificaciones.

2.1. Requisitos de Hardware (Servidor / Estación de Trabajo)

Componente	Mínimo (Desarrollo)	Recomendado (Producción)
Sistema Operativo	Windows 10/11, macOS 10.15+, Ubuntu 20.04+	Linux (Ubuntu 22.04 LTS / Debian 11)

Procesador (CPU)	2 vCPU	4 vCPU o superior
Memoria RAM	4 GB	8 GB - 16 GB (según carga)
Almacenamiento	10 GB disponibles	50 GB SSD (para persistencia de BD y Logs)

2.2. Requisitos de Software (Stack Tecnológico)

- .NET SDK: Versión **8.0.x o superior**.
- Motor de Base de Datos:
  - **SQL Server 2022** (Recomendado para este despliegue).
  - Alternativas: PostgreSQL 13+, MySQL 8.0+.
- Docker Engine: Versión **20.10+** (para entornos contenerizados).
- Docker Compose: Versión **2.0+**.
- Git: Para el control de versiones.
- IDE Sugerido: Visual Studio 2022 o VS Code (con extensión C# Dev Kit).

3.Instalación de Prerrequisitos

3.1. Instalación del SDK de .NET 8

El SDK es necesario para compilar y ejecutar comandos de CLI de .NET.

Windows (PowerShell Admin):

winget install Microsoft.DotNet.SDK.8

Linux (Ubuntu/Debian):

sudo apt-get update && \

sudo apt-get install -y dotnet-sdk-8.0

3.2. Instalación de Docker y Docker Compose

Docker es esencial para levantar los servicios auxiliares (Redis, SQL Server) y para el despliegue final.

Linux (Ubuntu):

sudo apt-get update

sudo apt-get install -y docker.io docker-compose

sudo systemctl enable --now docker

sudo usermod -aG docker $USER # Reiniciar sesión para aplicar cambios de grupo

4. Configuración del Entorno de Desarrollo

4.1. Obtención del Código Fuente

Clone el repositorio oficial utilizando Git:

git clone <url-del-repositorio-git>

cd Appointments-Project-Back

4.2. Configuración de Variables de Entorno (.env)

Cree un archivo `.env` en la raíz de la solución para definir el contexto de ejecución. *Nota: Este archivo no debe ser commiteado al repositorio si contiene credenciales de producción.*

Archivo: .env

ENVIRONMENT=develop

Variables opcionales para docker-compose

MSSQL_SA_PASSWORD=TuPasswordSeguro123!

4.3. Configuración de la Aplicación (appsettings.json)

Navegue al directorio `Web_back/` y verifique el archivo `appsettings.json`. Asegúrese de configurar correctamente las cadenas de conexión y proveedores.

Segmento Crítico de Configuración:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost,1433;Database=CitasDB;User Id=sa;Password=TuPasswordSeguro123!;TrustServerCertificate=true;",
    "Redis": "localhost:6379"
  },
  "DatabaseProvider": "sqlserver", // Opciones: "sqlserver", "postgresql", "mysql"
  "JwtSettings": {
    "Key": "SUPER_SECRET_KEY_MIN_32_CHARS",
    "Issuer": "Appointments.API",
    "Audience": "Appointments.Client",
    "ExpiresInMinutes": 60
  }
}
```

5. Configuración de Base de Datos (Migraciones)

El sistema utiliza **Entity Framework Core Code-First**. No debe crear las tablas manualmente; el sistema las generará basándose en los modelos.

5.1. Instalación de Herramientas EF

dotnet tool install --global dotnet-ef

5.2. Ejecución de Migraciones

Desde la raíz del proyecto, ejecute el siguiente comando para aplicar el esquema a la base de datos configurada en `DefaultConnection`:

Sintaxis: dotnet ef database update --project <ProyectoEntidades> --startup-project <ProyectoWeb>

dotnet ef database update --project Entity-Back/Entity-Back.csproj --startup-project Web_back/Web_back.csproj

Nota: Esto creará automáticamente la base de datos `CitasDB` y poblará las tablas con los datos semilla (Roles, Permisos, Tipos de Documento) definidos en el método `OnModelCreating`.

6. Construcción y Ejecución Local

6.1. Restauración y Compilación

Restaurar paquetes NuGet

dotnet restore Appointments-Project-Back.sln

Compilar la solución

dotnet build --configuration Release

6.2. Ejecución de la API

cd Web_back

dotnet run

La API estará disponible por defecto en `http://localhost:5103` (o el puerto configurado en `launchSettings.json`). Puede verificar el estado accediendo a Swagger: `http://localhost:5103/swagger`.

7. Despliegue Contenerizado (Docker)

Para entornos de Staging o Producción, se recomienda utilizar la orquestación con Docker Compose, que levanta la API, la Base de Datos, Redis y el Proxy Inverso (Nginx).

7.1. Estructura de Servicios

El archivo docker-compose.yml orquesta los siguientes contenedores:

- appointments-api: La aplicación .NET 8.

- appointments-sqlserver: Motor de base de datos persistente.
- redis: Cache distribuido para gestión de sesiones y locks.
- nginx: Proxy reverso para manejo de SSL y enrutamiento.

7.2. Despliegue

Ejecute el siguiente comando en la raíz de la solución:

docker-compose up -d --build

7.3. Verificación de Contenedores

docker ps

Debe observar los servicios con estado Up. Si algún servicio falla, inspeccione los logs:

docker logs appointments-api

8. Integración Continua (CI/CD) con Jenkins

Este proyecto incluye un `Jenkinsfile` para automatizar el ciclo de vida de entrega de software.

8.1. Configuración del Pipeline

El pipeline está configurado para detectar la rama del repositorio (`main`, `develop`, `staging`) y desplegar en el entorno correspondiente.

Etapas del Pipeline:

1. **Checkout:** Descarga del código fuente desde Git.
2. **Build:** Compilación del proyecto y ejecución de pruebas unitarias (dotnet test).
3. **Docker Build:** Creación de la imagen Docker (appointments-api:latest).
4. **Deploy:** Despliegue mediante docker-compose en el servidor destino (AWS/On-Premise).

8.2. Requisitos del Agente Jenkins

El servidor donde corre Jenkins debe tener instalados:

- Docker y Docker Compose.

- Plugin "Docker Pipeline" en Jenkins.
- Credenciales de acceso configuradas (SSH Keys para despliegues remotos).

9. Solución de Problemas

9.1. Error de Conexión a Base de Datos

- **Síntoma:** Excepción SqlException: A network-related or instance-specific error occurred.
- **Solución:** Verifique que el contenedor de SQL Server esté corriendo (docker ps). Asegúrese de que la cadena de conexión en appsettings.json apunte al nombre del servicio Docker (Server=sqlserver) si corre en contenedor, o a localhost si corre nativo.

9.2. Error de CORS en el Frontend

- **Síntoma:** El navegador bloquea las peticiones de Angular/React.
- **Solución:** Agregue la URL del frontend en la sección OrigenesPermitidos del appsettings.json.

9.3. Problemas con Migraciones

- **Síntoma:** La base de datos no tiene las tablas creadas.
- **Solución:** Si usa Docker, asegúrese de que el contenedor de la API tenga un script de entrada (entrypoint.sh) que ejecute dotnet ef database update al iniciar, o ejecútelo manualmente contra el contenedor.

10. Glosario Técnico

- CI/CD: Integración Continua / Despliegue Continuo.
- JWT: JSON Web Token (Estándar de autenticación).
- Reverse Proxy (Nginx): Servidor que se sitúa delante de la API para manejar seguridad, SSL y balanceo de carga.
- SignalR: Biblioteca para añadir funcionalidad web en tiempo real.
- Seed Data: Datos iniciales precargados en la base de datos para que el sistema sea funcional desde el primer inicio.

11. Backup SQL Server

```
docker exec appointments-sqlserver-develop /opt/mssql-tools/bin/sqlcmd \
 -S localhost -U sa -P password \
 -Q "BACKUP DATABASE proyectoCtT2 TO DISK =
'/var/opt/mssql/backup/proyectoCtT2.bak'"
```