Inmobiliaria API

API REST desarrollada en ASP.NET Core para gestionar propietarios, inmuebles, contratos y pagos de la inmobiliaria ULP. Expone endpoints protegidos con JWT, persiste datos en MySQL y publica recursos estáticos (imágenes) para el cliente móvil Android.

Tecnologías utilizadas

- .NET 8 (ASP.NET Core Web API)
- Entity Framework Core + Pomelo MySql Provider
- Autenticación JWT
- Serilog para logging
- Docker & Docker Compose
- Swagger / OpenAPI

Requisitos previos

Herramienta	Versión recomendada
.NET SDK	8.0.x
MySQL	8.0.x
Docker	24+ *(opcional)*
Docker Compose	2.x *(opcional)*

Configuración de variables y secretos

La API lee la configuración desde `appsettings.json`, variables de entorno o *user-secrets*. Los valores mínimos a definir son:

Variable / clave	Descripción	Valor por defecto
`ConnectionStrings:DefaultConnection`	Cadena de conexión MySQL	`Server=localhost;Port=3306;Database=INMOBILIARIAULP;User=root;Password=root123;`
`Jwt:Secret`	Llave secreta para firmar tokens JWT	`h7H9gK2pQz8LwS6rXjD4fN1tVbY0eUA23SAa23`
`Jwt:Issuer`	Emisor del token	`InmobiliariaAPI`
`Jwt:Audience`	Audiencia del token	`InmobiliariaClients`
`Salt`	Salt utilizado para el hash de contraseñas seed	`h7H9gK2pQz8LwS6rXjD4fN1tVbY0eU`

> Recomendado: sobrescribir estos valores en `appsettings.Development.json`, variables de entorno (`ConnectionStrings__DefaultConnection`, `Jwt__Secret`, etc.) o `dotnet user-secrets` antes de subir a producción.

Ejecución local (sin Docker)

1. Clonar el repositorio y posicionarse en la carpeta `InmobiliariaAPI`.
2. Crear/configurar la base de datos MySQL apuntada por `ConnectionStrings:DefaultConnection`.
3. Restaurar dependencias y compilar:

```
dotnet restore
dotnet build
```

4. Aplicar migraciones y seed inicial (crea tablas y datos base):

```
dotnet ef database update
```

5. Ejecutar la API:

```
dotnet run
```

6. La API quedará disponible en `https://localhost:5001` y `http://localhost:5000` por defecto. Swagger se expone en `/swagger`.

Ejecución con Docker Compose

1. Asegurarse de que `docker` y `docker compose` estén instalados.
2. Desde `InmobiliariaAPI`, levantar los servicios:

```
docker compose up -d --build
```

   - Levanta MySQL (`inmobiliaria_mysql`) y la API (`inmobiliaria_api`).
   - Los datos persisten en el volumen `inmobiliariaapi_mysql_data`.
   - Para logs e imágenes se montan las carpetas `./logs` y `./wwwroot/uploads`.

3. Detener los servicios:

```
docker compose down
```

   (agregar `-v` si querés borrar el volumen de datos).

Migraciones y datos semilla

- Crear una nueva migración:

```
dotnet ef migrations add NombreMigracion --output-dir Infrastructure/Data/Migrations
```

- Aplicar migraciones (local o dentro del contenedor):

```
dotnet ef database update
```

La clase `DataSeeder` se ejecuta en `OnModelCreating`, por lo que cada `database update` aplica automáticamente el seed de propietarios, inmuebles, contratos, imágenes y pagos.

Carpeta de uploads

Las imágenes se sirven desde `wwwroot/uploads`. En entorno Docker se monta como volumen para persistir archivos.

Proyecto académico – Laboratorio de Programación III (ULP).