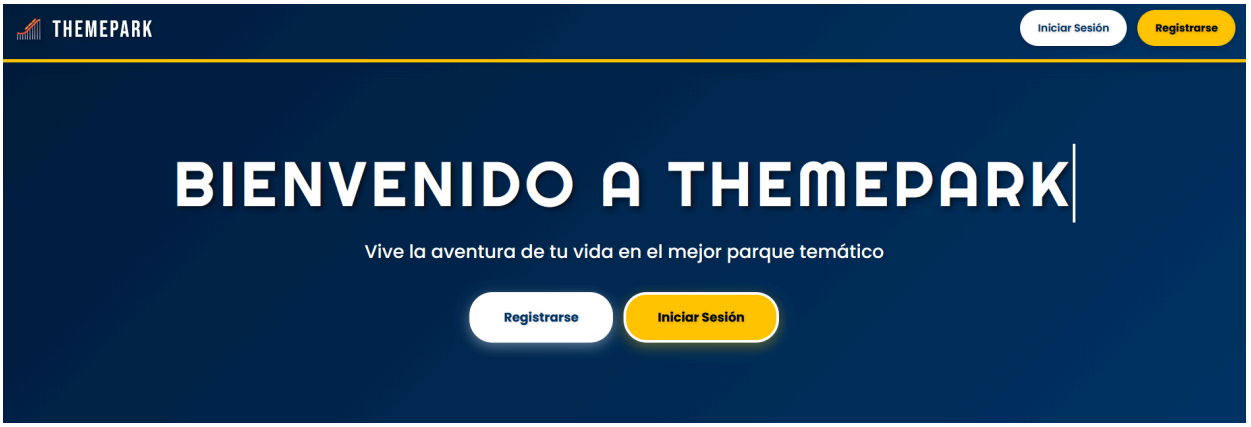
Guía de Deploy
ThemePark Application



Nicolas Bidenti (305108)

Santiago Canadell (282542)

Felipe Delgado (281987)

Tutores: Alexander Wieler

Juan Barrios

Fernando Spillere

2025

https://github.com/IngSoft-DA2/305108-282542-281987

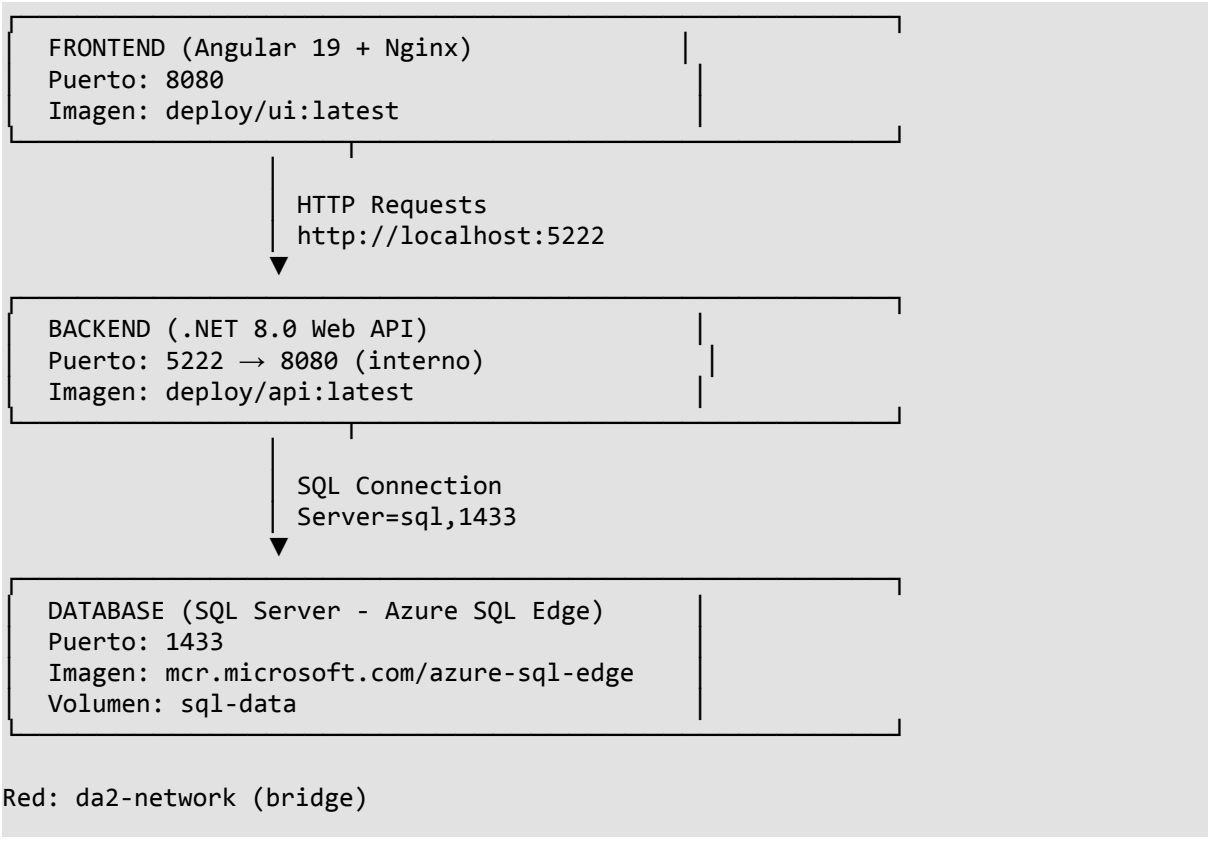
INDICE:

Introducción

ThemePark Application es un sistema completo de gestión de parques temáticos construido con tecnologías modernas:

- Frontend: Angular 19 (interfaz de usuario)
- Backend: ASP.NET Core 8.0 (API REST)
- Base de Datos: SQL Server (Azure SQL Edge)

Arquitectura de Contenedores



Requisitos Previos

Software Necesario

Software	Versión Mínima	Obligatorio	Descarga
Docker Desktop	Última versión	Sí	docker.com
Git	Última versión	Sí	git-scm.com
.NET SDK 8.0	8.0.0	Si	dotnet.microsoft.com

Requisitos de Sistema

- **Sistema Operativo**: Windows 10/11, macOS, o Linux
- **RAM**: Mínimo 8 GB (recomendado 16 GB)
- **Espacio en Disco**: Al menos 10 GB libres
- **Docker Desktop**: Debe estar corriendo antes de comenzar

Verificar Instalación

Abre una terminal y ejecuta:

```
# Verificar Docker
docker --version
# Salida esperada: Docker version 24.x.x o superior

# Verificar Docker Compose
docker-compose --version
# Salida esperada: Docker Compose version v2.x.x o superior

# Verificar Git
git --version
# Salida esperada: git version 2.x.x
```

IMPORTANTE: Asegúrate de que Docker Desktop esté corriendo antes de continuar.

Limpieza de Docker

Antes de comenzar el deploy, es crucial limpiar cualquier imagen previa del proyecto para evitar corrupción o conflictos.

Paso 1: Verificar Imágenes Existentes

```
docker images
```

Busca en la lista las siguientes imágenes:

- `deploy/api`
- `deploy/ui`
- `mcr.microsoft.com/azure-sql-edge`

Paso 2: Eliminar Imágenes del Proyecto (si existen)

Si encuentras alguna de las imágenes anteriores, elimínalas:

```
# Eliminar imagen de la API
docker rmi deploy/api:latest

# Eliminar imagen del Frontend
docker rmi deploy/ui:latest

# Eliminar imagen de SQL Server (si quieres una versión fresca)
docker rmi mcr.microsoft.com/azure-sql-edge
```

Nota: Si recibes un error diciendo que la imagen está en uso, primero detén y elimina los contenedores:

```
# Detener todos los contenedores del proyecto
docker-compose down

# Eliminar contenedores detenidos
docker container prune -f

# Luego intenta eliminar las imágenes nuevamente
docker rmi deploy/api:latest deploy/ui:latest
```

Paso 3: Limpiar Volúmenes Antiguos (Opcional)

ADVERTENCIA: Esto eliminará todos los datos de la base de datos. Solo hazlo si quieres empezar completamente de cero.

```
# Ver volúmenes existentes
docker volume ls

# Eliminar el volumen de la base de datos (si existe)
docker volume rm 305108-282542-281987_sql-data
```

Tip: Si no estás seguro de qué eliminar, simplemente ejecuta las primeras dos secciones. Docker Compose se encargará del resto.

Descarga del Código

Paso 1: Clonar el Repositorio

Abre una terminal y navega a la carpeta donde quieres descargar el proyecto:

```
# Opción A: Si tienes acceso al repositorio remoto
git clone https://github.com/IngSoft-DA2/305108-282542-281987.git
cd 305108-282542-281987

# Opción B: Si ya tienes el código
cd E:\Facultad\DA2\305108-282542-281987
```

Paso 2: Verificar Estructura del Proyecto

Tu proyecto debe tener la siguiente estructura:

```
305108-282542-281987/
├── docker-compose.yml          ← Archivo principal de orquestación
├── ThemePark/                  ← Backend (.NET)
│   ├── Dockerfile              ← Build de la API
│   ├── ThemeParkApi/
│   ├── ThemePark.BusinessLogic/
│   ├── ThemePark.DataAccess/   ← Migraciones de EF Core
│   └── ThemePark.sln
├── ThemeParkUI/                ← Frontend (Angular)
│   ├── Dockerfile              ← Build del UI
│   ├── nginx.conf
│   ├── package.json
│   └── src/
└── Datos/
    └── BaseDeDatos/            ← Scripts SQL
```

```
        ├── DB_vacia.sql
        └── InsertarDatosDePrueba.sql
```

Verifica que el archivo `docker-compose.yml` existe en la raíz:

```
# Windows
dir docker-compose.yml

# Linux/Mac
ls docker-compose.yml
```

Configuración de Base de Datos

Hay **dos opciones** para inicializar la base de datos. Ambas son válidas y el resultado es el mismo.

¿Cuál opción elegir?

Opción	Cuándo usarla	Ventajas
A: Entity Framework	Deploy automatizado, CI/CD, desarrollo	Automático Control de versiones
B: Script SQL	Deploy manual, troubleshooting, datos específicos	Control total No requiere .NET SDK

Opción A: Entity Framework Migrations (Recomendado)

Esta opción usa las migraciones de EF Core para crear automáticamente el esquema de la base de datos.

A.1 - Requisitos

- .NET SDK 8.0 instalado en tu máquina
- Base de datos SQL Server corriendo (puede ser en Docker)

A.2 - Iniciar Solo la Base de Datos

Primero, inicia únicamente el contenedor de SQL Server:

```
docker-compose up -d sql
```

Espera 10-15 segundos para que SQL Server termine de iniciar.

A.3 - Verificar Conexión a la Base de Datos

```
# Verificar que el contenedor está corriendo
docker ps

# Deberías ver el contenedor 'sql' en estado 'Up'
```

A.4 - Aplicar Migraciones

Desde la raíz del proyecto, ejecuta:

```
# Navegar a la carpeta del backend
cd ThemePark

# Aplicar y crear migraciones

dotnet ef migrations add --project
ThemePark.DataAccess\ThemePark.DataAccess.csproj --startup-project
ThemeParkApi\ThemeParkApi.csproj --context ThemePark.DataAccess.ThemeParkDbContext
--configuration Release Initial --output-dir Migrations


dotnet ef database update --project
ThemePark.DataAccess\ThemePark.DataAccess.csproj --startup-project
ThemeParkApi\ThemeParkApi.csproj --context ThemePark.DataAccess.ThemeParkDbContext
--configuration Release Initial



# Advertencia: Para mac con (/)

dotnet ef migrations add --project
ThemePark.DataAccess/ThemePark.DataAccess.csproj --startup-project
ThemeParkApi/ThemeParkApi.csproj --context ThemePark.DataAccess.ThemeParkDbContext
--configuration Release Initial --output-dir Migrations

dotnet ef database update --project
ThemePark.DataAccess/ThemePark.DataAccess.csproj --startup-project
ThemeParkApi/ThemeParkApi.csproj --context ThemePark.DataAccess.ThemeParkDbContext
--configuration Release Initial
```

Salida esperada:

```
Build started...
Build succeeded.
Applying migration '20251115002458_Initial'.
Done.
```

A.5 - Verificar que se Creó la Base de Datos

Puedes verificar conectándote a SQL Server:

- **Host**: `localhost`
- **Puerto**: `1433`
- **Usuario**: `SA`
- **Contraseña**: `MyPass@word`
- **Base de Datos**: `ThemeParkDb`

Tip: Usa herramientas como **Azure Data Studio**, **DBeaver**, o **SQL Server Management Studio** para conectarte.

A.6 - Cargar Datos de Prueba (Opcional)

Las migraciones crean el esquema vacío. Si quieres datos de prueba, ve a la Opción B - Paso B.4.

Opción B: Script SQL (Manual)

Esta opción ejecuta scripts SQL directamente contra la base de datos.

B.1 - Iniciar Solo la Base de Datos

```
docker-compose up -d sql
```

Espera 10-15 segundos para que SQL Server termine de iniciar.

B.2 - Conectarse a SQL Server

Usa tu cliente SQL favorito (Azure Data Studio, DBeaver, SSMS) con las siguientes credenciales:

- **Server**: `localhost,1433`
- **Authentication**: SQL Server Authentication
- **Username**: `SA`
- **Password**: `MyPass@word`
- **Trust Server Certificate**: `Yes` / `True`

B.3 - Ejecutar Script de Esquema

1. Abre el archivo: `Datos/BaseDeDatos/DB_vacia.sql`
2. Ejecuta todo el contenido del script

Este script creará:

- La base de datos `ThemeParkDb`
- Todas las tablas necesarias (Users, Attractions, Tickets, Events, etc.)
- Constraints, índices y relaciones

IMPORTANTE: Asegúrate de ejecutar TODO el script de una sola vez.

B.4 - Ejecutar Script de Datos de Prueba

1. Abre el archivo: `Datos/BaseDeDatos/InsertarDatosDePrueba.sql`
2. Ejecuta todo el contenido del script

Este script insertará:

- Usuarios de ejemplo (admin, operador, visitantes)
- Atracciones de ejemplo (T-Rex Roller Coaster, Jurassic Simulator, etc.)
- Eventos de muestra
- Configuraciones del sistema

B.5 - Verificar Datos

Ejecuta una consulta simple para verificar:

```
USE ThemeParkDb;

SELECT COUNT(*) AS TotalUsuarios FROM Users;
```

```
SELECT COUNT(*) AS TotalAtracciones FROM Attractions;
SELECT COUNT(*) AS TotalEventos FROM Events;
```

Deberías ver varios registros en cada tabla.

Deploy con Docker Compose

Ahora que la base de datos está lista, vamos a levantar toda la aplicación.

Paso 1: Regresar a la Raíz del Proyecto

```
# Si estás en ThemePark/, regresa a la raíz 305108-282542-281987
cd ..

# Verifica que estás en la carpeta correcta
# Debes ver docker-compose.yml
dir  # Windows
ls   # Linux/Mac
```

Paso 2: Construir y Levantar Todos los Servicios

```
docker-compose build

docker-compose up -d
```

¿Qué hace este comando?

1. **Build**: Construye las imágenes de Docker para:
    - Backend API (desde `ThemePark/Dockerfile`)
    - Frontend UI (desde `ThemeParkUI/Dockerfile`)
2. **Pull**: Descarga la imagen de SQL Server si no existe
3. **Start**: Inicia los 3 contenedores:
    - `sql` - Base de datos
    - `api` - Backend API
    - `frontend` - Frontend Angular con Nginx

Paso 3: Observar los Logs

Verás logs de los 3 servicios. Busca estos mensajes clave:

SQL Server:

```
sql         | SQL Server is now ready for client connections
```

Backend API:

```
api         | Now listening on: http://[::]:8080
api         | Application started. Press Ctrl+C to shut down.
```

Frontend:

```
frontend    | Configuration complete; ready for start up
```

Tip: Si quieres ejecutar en segundo plano (detached mode):

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

Paso 4: Acceder a la Aplicación

Una vez que todos los servicios estén corriendo, abre tu navegador:

Servicio	URL	Descripción
Frontend	http://localhost:8080	Aplicación principal
Backend API	http://localhost:5222	API REST
Base de Datos	localhost:1433	SQL Server (usar cliente SQL)

Paso 5: Detener la Aplicación

Para detener todos los servicios:

```
# Si está corriendo en terminal (Ctrl+C primero, luego):
docker-compose down
```

Troubleshooting

Soluciones a problemas comunes durante el deploy.

Problema 1: Puertos Ocupados

Síntoma:

```
Error: Bind for 0.0.0.0:8080 failed: port is already allocated
```

Causa: Otro programa está usando el puerto 8080, 5222, o 1433.

Solución:

Opción A - Identificar y Detener el Proceso:

```
# Windows
netstat -ano | findstr :8080
netstat -ano | findstr :5222
netstat -ano | findstr :1433

# Luego matar el proceso (usa el PID de la columna final)
taskkill /PID <PID> /F

# Linux/Mac
lsof -i :8080
lsof -i :5222
lsof -i :1433

# Luego matar el proceso
kill -9 <PID>
```

Opción B - Cambiar Puertos en docker-compose.yml:

Edita el archivo `docker-compose.yml` y cambia los puertos:

```
services:
  frontend:
    ports:
      - "8081:8080"  # Cambiar 8080 a 8081

  api:
    ports:
      - "5223:8080"  # Cambiar 5222 a 5223

  sql:
    ports:
      - "1434:1433"  # Cambiar 1433 a 1434
```

IMPORTANTE: Si cambias puertos, también debes actualizar:

- `ThemeParkUI/src/environments/environment.ts` (apiUrl)
- `ThemePark/ThemeParkApi/appsettings.json` (ConnectionString)

Problema 2: Contenedores No Inician

Síntoma:

```
docker ps
# Muestra menos de 3 contenedores
```

Solución - Ver Logs:

```
# Ver logs de todos los servicios
docker-compose logs

# Ver logs de un servicio específico
docker-compose logs sql
docker-compose logs api
docker-compose logs frontend

# Ver logs en tiempo real
docker-compose logs -f api
```

Identifica el error en los logs y busca la solución específica abajo.

Problema 3: Error de Build en Backend

Síntoma:

```
ERROR [api build 6/8] RUN dotnet build ...
```

Posibles Causas y Soluciones:

A) Archivos faltantes:

```
# Verifica que todos los archivos estén presentes
ls ThemePark/
ls ThemePark/ThemeParkApi/
```

B) Error de restauración de paquetes:

```
# Limpiar y rebuild
docker-compose down
docker system prune -f
docker-compose up --build
```

C) Version de .NET incorrecta:

Verifica el Dockerfile de backend (`ThemePark/Dockerfile`). Debe usar .NET 8.0:

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
```

Problema 4: Error de Build en Frontend

Síntoma:

```
ERROR [frontend build 7/8] RUN npm run build
```

Posibles Causas y Soluciones:

A) Dependencies no instaladas correctamente:

```
# Rebuild desde cero
docker-compose down
docker rmi deploy/ui:latest
docker-compose up --build
```

B) Error en package.json:

Verifica que `ThemeParkUI/package.json` tenga todas las dependencias correctas.

C) Memoria insuficiente:

Si el build se queda sin memoria, aumenta la memoria de Docker:

- Docker Desktop → Settings → Resources → Memory: 4GB o más

Problema 5: Base de Datos No Conecta

Síntoma:

```
api | Failed to connect to server sql:1433
api | A network-related or instance-specific error occurred
```

Soluciones:

A) SQL Server aún no está listo:

El contenedor SQL puede tardar 30-60 segundos en estar completamente listo.

```
# Esperar y verificar logs
docker-compose logs sql
```

```
# Buscar este mensaje:
# "SQL Server is now ready for client connections"
```

B) Verificar que SQL está corriendo:

```
docker ps | grep sql

# Si no aparece, iniciarlo:
docker-compose up -d sql
```

C) Credenciales incorrectas:

Verifica el archivo `ThemePark/ThemeParkApi/appsettings.json`:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=sql,1433;Database=ThemeParkDb;User
Id=SA;Password=MyPass@word;TrustServerCertificate=true;MultipleActiveResultSets=tr
ue"
  }
}
```

La contraseña debe coincidir con la variable de entorno en `docker-compose.yml`

Problema 6: Frontend No Carga

Síntoma:

- Navegador muestra "This site can't be reached" en http://localhost:8080
- O muestra página en blanco

Soluciones:

A) Verificar que el contenedor está corriendo:

```
docker ps | grep frontend

# Si no aparece:
docker-compose logs frontend

# Buscar errores en el log
```

B) Verificar configuración de Nginx:

El archivo `ThemeParkUI/nginx.conf` debe tener:

```
listen 8080;
```

C) Limpiar caché del navegador:

- Abre las herramientas de desarrollo (F12)
- Haz clic derecho en el botón de recargar
- Selecciona "Empty Cache and Hard Reload"

D) Verificar en el navegador correcto:

Asegúrate de usar: `http://localhost:8080` (HTTP, no HTTPS)

Problema 7: Errores de Comunicación Frontend ↔ Backend

Síntoma:

- Frontend carga pero no muestra datos
- Errores CORS en la consola del navegador
- Errores 404 al llamar a la API

Soluciones:

A) Verificar URL de la API:

Archivo: `ThemeParkUI/src/environments/environment.ts`

```
export const environment = {
  production: true,
  apiUrl: 'http://localhost:5222'  // ← Debe coincidir con puerto de API
};
```

B) Verificar CORS en Backend:

El backend debe permitir peticiones desde `http://localhost:8080`.

C) Verificar que Backend responde:

```
# Probar endpoint de salud
curl http://localhost:5222

# O en navegador:
# http://localhost:5222/swagger
```

Problema 8: Permisos de Docker (Linux/Mac)

Síntoma:

```
permission denied while trying to connect to the Docker daemon socket
```

Solución:

```
# Opción 1: Agregar usuario al grupo docker
sudo usermod -aG docker $USER

# Luego logout/login

# Opción 2: Ejecutar con sudo
sudo docker-compose up --build
```

Problema 9: Volumen de Base de Datos Corrupto

Síntoma:

- Base de datos tiene datos viejos
- Errores de schema inconsistente

- Migraciones fallan

Solución - Eliminar y Recrear Volumen:

ADVERTENCIA: Esto eliminará TODOS los datos de la base de datos.

```
# 1. Detener todos los contenedores
docker-compose down

# 2. Eliminar el volumen
docker volume rm 305108-282542-281987_sql-data

# 3. Volver a iniciar (creará nuevo volumen)
docker-compose up -d sql

# 4. Esperar a que SQL inicie
# (30-60 segundos)

# 5. Reaplicar migraciones o scripts
# Ver sección "Configuración de Base de Datos"

# 6. Levantar el resto de servicios
docker-compose up -d
```
