**Patrón de fondo

El contenido generado por IA puede ser incorrecto.**



**DESPLIEGUE E IMPLEMENTACIÓN**

Sistema de gestión y visualización de rutas de transporte urbano en tiempo real

**Texto

El contenido generado por IA puede ser incorrecto.**

*Patrón de fondo

El contenido generado por IA puede ser incorrecto.*

# Documento de despliegue e implementación

Brayan Estiven Carvajal Padilla

Diego Fernando Cuellar Hernandez

Andres Felipe Suaza Bustos

Carlos Javier Rodriguez Manchola

*Aprendices*

Carlos Julio Cadena

*Instructor*

ANÁLISIS Y DESARROLLO DE SOFTWARE

FICHA 2899747

SERVICIO NACIONAL DE APRENDIZAJE SENA

CENTRO DE LA INDUSTRIA DE LA EMPRESA Y LOS SERVICIOS

REGIONAL HUILA

2025

Contenido

[Documento de despliegue e implementación 2](#_Toc210843228)

[1. Propósito y alcance 4](#_Toc210843229)

[2. Resumen de arquitectura 4](#_Toc210843230)

[3. Requisitos previos 5](#_Toc210843231)

[4. Obtención del código y ramas 6](#_Toc210843232)

[5. Variables de entorno 6](#_Toc210843233)

[6. Despliegue local con Docker Compose (desarrollo) 6](#_Toc210843234)

[7. Despliegue en producción (servidor Linux) 10](#_Toc210843235)

[8. Seguridad y operación 13](#_Toc210843236)

[9. Verificación posterior al despliegue 13](#_Toc210843237)

[10. Solución de problemas 14](#_Toc210843238)

[11. Control de versiones 14](#_Toc210843239)

# 1. Propósito y alcance

Este documento describe los procedimientos oficiales para la implementación y el despliegue del sistema UrbanTracker en entornos Linux utilizando Docker y Docker Compose. Cubre tanto el entorno de desarrollo local como un entorno de producción en servidor, incluyendo requisitos, variables de entorno, pasos de despliegue, verificación y operación.

# 2. Resumen de arquitectura

UrbanTracker está compuesto por los siguientes servicios principales:

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Tecnología** | **Puerto (cont.)** | **Descripción** |
| Base de datos | PostgreSQL 15 (alpine) | 5432 | Persistencia de datos del sistema. |
| Broker de mensajería | Eclipse Mosquitto 2.x | 1883 | Mensajería MQTT para actualizaciones en tiempo real. |
| Backend (API) | Java 17 + Spring Boot | 8080 | Servicios REST, seguridad, negocio y acceso a datos. |
| Web Admin | Next.js (Node 18) | 3000 | Interfaz de administración y monitoreo. |
| Web Client (opcional) | Next.js (Node 18) | 3001 (host) | Interfaz para usuarios finales si aplica. |

# 3. Requisitos previos

|  |  |
| --- | --- |
| **Requisito** | **Versión/Detalle** |
| Linux | Ubuntu 20.04+ (usuario con sudo) |
| Git | 2.x |
| JDK | Java 17 |
| Maven | 3.9.x |
| Node.js | 18 LTS |
| Docker | 20.x+ |
| Docker Compose | Plugin o binario 2.x+ |

Instalación rápida en Ubuntu/Debian:

|  |
| --- |
| sudo apt update && sudo apt install -y docker.io docker-compose git openjdk-17-jdk maven nodejs |

# 4. Obtención del código y ramas

Clonar el repositorio y cambiar a la rama de desarrollo:

|  |
| --- |
| git clone https://github.com/AFSB114/UrbanTracker.git cd UrbanTracker |

# 5. Variables de entorno

Configurar las siguientes variables según el entorno. Se recomienda usar un archivo `.env` en la raíz del proyecto o variables en el orquestador.

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Componente** | **Ejemplo** | **Descripción** |
| POSTGRES\_USER | DB | urbantracker | Usuario de PostgreSQL. |
| POSTGRES\_PASSWORD | DB | urbantracker123 | Contraseña de PostgreSQL. |
| POSTGRES\_DB | DB | urbantracker | Base de datos por defecto. |
| SPRING\_DATASOURCE\_URL | Backend | jdbc:postgresql://db:5432/urbantracker | URL JDBC a la BD (nombre de host según Compose). |
| SPRING\_DATASOURCE\_USERNAME | Backend | urbantracker | Usuario BD. |
| SPRING\_DATASOURCE\_PASSWORD | Backend | urbantracker123 | Contraseña BD. |
| NEXT\_PUBLIC\_API\_URL | Web | http://localhost:8080 | URL base de la API a consumir por el frontend. |
| JWT\_SECRET | Backend | cambiar-por-secreto | Secreto para firma de tokens (producción). |

# 6. Despliegue local con Docker Compose (desarrollo)

Crear el archivo `docker-compose.yml` en la raíz del proyecto con el contenido siguiente:

|  |
| --- |
| version: "3.9" services:  db:  image: postgres:15-alpine  container\_name: urbantracker-db  environment:  POSTGRES\_USER: ${POSTGRES\_USER:-urbantracker}  POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD:-urbantracker123}  POSTGRES\_DB: ${POSTGRES\_DB:-urbantracker}  volumes:  - pgdata:/var/lib/postgresql/data  ports:  - "5432:5432"  networks: [urbantracker-net]   broker:  image: eclipse-mosquitto:2.0  container\_name: urbantracker-broker  ports:  - "1883:1883"  networks: [urbantracker-net]   backend:  build:  context: ./Backend  dockerfile: Dockerfile  image: urbantracker-backend:dev  container\_name: urbantracker-backend  depends\_on:  - db  environment:  SPRING\_DATASOURCE\_URL: jdbc:postgresql://db:5432/${POSTGRES\_DB:-urbantracker}  SPRING\_DATASOURCE\_USERNAME: ${POSTGRES\_USER:-urbantracker}  SPRING\_DATASOURCE\_PASSWORD: ${POSTGRES\_PASSWORD:-urbantracker123}  JWT\_SECRET: ${JWT\_SECRET:-cambiar-por-secreto}  ports:  - "8080:8080"  networks: [urbantracker-net]   webadmin:  build:  context: ./Web-Admin  dockerfile: Dockerfile  image: urbantracker-webadmin:dev  container\_name: urbantracker-webadmin  depends\_on:  - backend  environment:  NEXT\_PUBLIC\_API\_URL: http://localhost:8080  ports:  - "3000:3000"  networks: [urbantracker-net]  networks:  urbantracker-net:  volumes:  pgdata: |

Crear el Dockerfile del backend (multi-stage) en `Backend/Dockerfile`:

|  |
| --- |
| # Etapa de build FROM maven:3.9-eclipse-temurin-17 AS build WORKDIR /workspace COPY pom.xml . COPY src ./src RUN mvn -q -DskipTests clean package  # Etapa de runtime FROM eclipse-temurin:17-jre-alpine WORKDIR /app COPY --from=build /workspace/target/\*.jar app.jar EXPOSE 8080 ENTRYPOINT ["java","-jar","/app/app.jar"] |

Crear el Dockerfile de la aplicación Web-Admin en `Web-Admin/Dockerfile`:

|  |
| --- |
| FROM node:18-alpine WORKDIR /app COPY package\*.json ./ RUN npm install COPY . . RUN npm run build ENV NODE\_ENV=production EXPOSE 3000 CMD ["npm", "run", "start"] |

Construcción y arranque del entorno local:

|  |
| --- |
| docker compose build docker compose up -d # Verificar contenedores docker compose ps |

Accesos por defecto: API en http://localhost:8080, Swagger UI en /swagger-ui/index.html y web admin en http://localhost:3000.

# 7. Despliegue en producción (servidor Linux)

En producción se recomienda usar Docker Compose con políticas de reinicio y un proxy reverso con TLS. Crear un archivo `docker-compose.prod.yml` y un archivo `.env` con las credenciales. Asegurar copias de seguridad del volumen de PostgreSQL y aplicar actualizaciones controladas.

|  |
| --- |
| version: "3.9" services:  db:  image: postgres:15-alpine  container\_name: urbantracker-db  restart: unless-stopped  environment:  POSTGRES\_USER: ${POSTGRES\_USER}  POSTGRES\_PASSWORD: ${POSTGRES\_PASSWORD}  POSTGRES\_DB: ${POSTGRES\_DB}  volumes:  - pgdata:/var/lib/postgresql/data  networks: [urbantracker-net]   broker:  image: eclipse-mosquitto:2.0  container\_name: urbantracker-broker  restart: unless-stopped  ports:  - "1883:1883"  networks: [urbantracker-net]   backend:  image: urbantracker-backend:prod  build:  context: ./Backend  dockerfile: Dockerfile  container\_name: urbantracker-backend  restart: unless-stopped  depends\_on: [db]  environment:  SPRING\_DATASOURCE\_URL: jdbc:postgresql://db:5432/${POSTGRES\_DB}  SPRING\_DATASOURCE\_USERNAME: ${POSTGRES\_USER}  SPRING\_DATASOURCE\_PASSWORD: ${POSTGRES\_PASSWORD}  JWT\_SECRET: ${JWT\_SECRET}  networks: [urbantracker-net]   webadmin:  image: urbantracker-webadmin:prod  build:  context: ./Web-Admin  container\_name: urbantracker-webadmin  restart: unless-stopped  depends\_on: [backend]  environment:  NEXT\_PUBLIC\_API\_URL: ${NEXT\_PUBLIC\_API\_URL}  networks: [urbantracker-net]   nginx:  image: nginx:1.25-alpine  container\_name: urbantracker-proxy  restart: unless-stopped  depends\_on: [webadmin, backend]  volumes:  - ./deploy/nginx.conf:/etc/nginx/nginx.conf:ro  - certs:/etc/ssl/certs:ro  ports:  - "80:80"  - "443:443"  networks: [urbantracker-net]  networks:  urbantracker-net:  volumes:  pgdata:  certs: |

Archivo `deploy/nginx.conf` (proxy reverso con rutas para API y Web-Admin; integrar certificados TLS con certbot o proveedor equivalente):

|  |
| --- |
| events {} http {  server {  listen 80;  server\_name ejemplo.urbantracker.com;   location / {  proxy\_pass http://urbantracker-webadmin:3000;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  }   location /api/ {  proxy\_pass http://urbantracker-backend:8080/;  proxy\_set\_header Host $host;  proxy\_set\_header X-Real-IP $remote\_addr;  }  } } |

Despliegue en producción:

|  |
| --- |
| cp .env.example .env # definir credenciales reales sudo docker compose -f docker-compose.prod.yml build sudo docker compose -f docker-compose.prod.yml up -d sudo docker compose -f docker-compose.prod.yml ps |

# 8. Seguridad y operación

Buenas prácticas: usar JWT\_SECRET fuerte y almacenarlo como secreto, restringir puertos públicos (exponer solo 80/443 y 1883 si es necesario), configurar copias de seguridad periódicas del volumen de PostgreSQL, actualizar imágenes de manera planificada, habilitar logs y monitoreo de contenedores.

# 9. Verificación posterior al despliegue

|  |  |  |
| --- | --- | --- |
| **Elemento** | **Cómo verificar** | **Resultado esperado** |
| API | http(s)://<host>/swagger-ui/index.html | Carga de documentación Swagger UI. |
| Web Admin | http(s)://<host>/ | Interfaz de administración operativa. |
| Base de datos | Conexión desde backend sin errores | Migraciones y tablas disponibles. |
| Broker MQTT | Conexión a puerto 1883 y publicación/suscripción de prueba | Mensajería operativa. |

# 10. Solución de problemas

|  |  |  |
| --- | --- | --- |
| **Síntoma** | **Causa probable** | **Acción recomendada** |
| 'Connection refused' BD | Variables DB incorrectas o contenedor no iniciado | Revisar logs `db` y variables SPRING\_DATASOURCE\_\*. |
| Web sin datos | URL API en frontend incorrecta | Ajustar NEXT\_PUBLIC\_API\_URL y reconstruir imagen. |
| Puerto en uso | Servicios locales ocupando 5432/8080/3000 | Liberar puertos o remapear en Compose. |
| 502 con Nginx | Backends no alcanzables | Revisar `proxy\_pass` y red de Docker. |

# 11. Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Fecha** | **Descripción** | **Autor** |
| 1.0 | 2025-10-07 | Primera edición del documento de despliegue e implementación. | Equipo UrbanTracker |