Documento de Despliegue

TT2025

https://github.com/dfleper

16/02/2026

El Documento de Despliegue describe el proceso de instalación, configuración y puesta en funcionamiento de la aplicación TT2025 – Turbo Taller en un entorno real. Incluye los requisitos técnicos, la configuración del servidor y base de datos, y los pasos necesarios para garantizar una implantación correcta, reproducible y segura del sistema.

Contenido

Documento de Despliegue

Nombre del proyecto:

Turbo Taller – Sistema web de gestión de citas para taller de mecánica rápida

Código del proyecto *(APCODE):*

TT2025

Integrantes del equipo:

dfleper (Domingo Fleitas) · GitHub

Fecha y versión del documento:

15/02/2026 – Versión 1.0

Historial de Versiones

Versión	Fecha	Autor	Cambios Realizados	Motivos / Referencia
1.0	**15/02/2026**	**https://github.com/dfleper**	**Documento de Despliegue**	**Versión inicial del documento de despliegue (Render + Docker + MariaDB Cloud 11.8 LTS)**

1. INTRODUCCIÓN

Este documento describe el despliegue e implantación del proyecto **TT2025 (Turbo Taller)** en un entorno de producción gestionado por **Render** utilizando **Docker**, con base de datos en **MariaDB Cloud 11.8 LTS.**

El objetivo es permitir que un tercero pueda desplegar la aplicación siguiendo pasos reproducibles, con configuración mediante variables de entorno y sin necesidad de modificar el código en producción.

Características técnicas del despliegue (según ficheros del proyecto):

- **Spring Boot 3.5.8**
- **Java 17**
- Empaquetado **WAR** (`packaging=war`)
- Artefacto compilado: `target/TT2025.war` (`finalName=TT2025`)
- Ejecución del WAR mediante java -jar (WAR ejecutable)
- Contenerización con **Dockerfile** multi-stage (build con Maven + runtime JRE)
- Base de datos externa MariaDB (driver runtime)
- Migraciones automáticas con **Flyway** (flyway-core + flyway-mysql)
- Hibernate en producción configurado como validate (no crea/modifica esquema)

2. DETALLE DE LAS MEJORAS

2.1.Mejoras técnicas

- Uso de **Spring Boot 3.5.8** y **Java 17** (compatibilidad garantizada).
- Empaquetado como **WAR** y ejecución en contenedor.
- Despliegue en Render mediante **Dockerfile** multi-stage.
- Build: maven:3.9-eclipse-temurin-17.
- Runtime: eclipse-temurin:17-jre.
- Configuración de producción en application-prod.properties con:
  - `server.port=${PORT:8080}` (compatible con Render)
  - `spring.jpa.hibernate.ddl-auto=validate`
- Flyway habilitado (`spring.flyway.enabled=true`)
- Hikari configurado para entornos cloud con valores ajustados y parametrizables
- Seguridad incluida (Spring Security + Thymeleaf extras)
- Actuator incluido para monitorización.

- Swagger/OpenAPI incluido como dependencia pero deshabilitado en producción.
- Sistema de correo incluido como dependencia, pero deshabilitado en producción por configuración.

2.2.Mejoras funcionales

La release desplegada corresponde a una versión operativa del sistema **TT2025** que incluye:

- Aplicación web con frontend server-side mediante Thymeleaf.
- Persistencia en MariaDB con entidades JPA.
- Control de acceso basado en autenticación y autorización (Spring Security).
- Gestión de esquema de base de datos y datos controlados por migraciones Flyway.

3. SERVIDOR DE APLICACIONES

3.1.Identificación del servidor

Plataforma: Render

Modalidad: Web Service con **Docker**

Ejecución: Contenedor Linux gestionado por Render

Artefacto ejecutado en runtime: `/app/app.war` copiado desde `(target/TT2025.war)`

Directorio de trabajo del contenedor: `/app`

Repositorio origen del despliegue: https://github.com/dfleper/PRW-TT2025

3.2.Requisitos mínimos

- **Java (JRE):** 17 (eclipse-temurin:17-jre)
- **Spring Boot:** 3.5.8
- **Maven:** 3.9 (maven:3.9-eclipse-temurin-17, fase de build)
- **Formato del binario:** WAR `(target/TT2025.war)`
- **Arranque:** `java -jar /app/app.war`
- Dependencias relevantes incluidas en el proyecto:
  - Web MVC, Thymeleaf, Validation, Data JPA, MariaDB JDBC, Flyway, Security, Actuator, Mail.

5.1.Configuración recomendada

- **Puerto:** Render inyecta la variable PORT ; la aplicación la consume mediante:
  - `server.port=${PORT:8080}` en application-prod.properties
- **Perfil recomendado en Render:**
  - `SPRING_PROFILES_ACTIVE=prod`
- **Variables de entorno obligatorias (BBDD):**
  - `SPRING_DATASOURCE_URL`
  - `SPRING_DATASOURCE_USERNAME`
  - `SPRING_DATASOURCE_PASSWORD`
- **Variables opcionales (pool Hikari):**
  - `TT2025_DB_POOL_MAX` (por defecto 5)
  - `TT2025_DB_POOL_MIN` (por defecto 0)

4. SERVIDOR DE BASE DE DATOS

4.1.Identificación del servidor

- **Motor:** MariaDB
- **Versión:** 11.8 LTS
- **Entorno:** MariaDB Cloud
- **Esquema:** tt2025

4.2.Requisitos mínimos

- Acceso remoto habilitado.
- Usuario con permisos suficientes para:
  - Crear tablas
  - Modificar tablas
  - Crear índices
- Conexión JDBC válida configurada por variable de entorno.

6.1.Scripts

La gestión oficial del esquema se realiza mediante Flyway:

```
spring.flyway.enabled=true
spring.flyway.locations=classpath:db/migration
```

En el primer arranque, Flyway aplica automáticamente las migraciones.

Se incluye adicionalmente el fichero:

```
DAW_PRW_TT2025_5.sql
```

Este fichero contiene una exportación del esquema vacío generado localmente y se entrega como:

- Copia de seguridad
- Soporte documental
- Método alternativo de restauración manual

No es necesario ejecutarlo manualmente si se utiliza el despliegue estándar con Docker y Flyway.

5. FICHEROS O BINARIOS A DESPLEGAR

5.1.Ubicación Se entrega el archivo:

```
DAW_PRW_TT2025_5_cod.zip
```

Contenido:

- TT2025.war
- Dockerfile
- .dockerignore
- application-prod.properties
- README_DEPLOY.md

Generación del WAR:

```
mvn -q -DskipTests clean package
```

Salida:

```
target/TT2025.war
```

6. OPERATIVA

6.1.Operaciones de base de datos

1. Disponer de instancia en MariaDB Cloud 11.8 LTS.
2. Configurar las variables:
   - `SPRING_DATASOURCE_URL`
   - `SPRING_DATASOURCE_USERNAME`
   - `SPRING_DATASOURCE_PASSWORD`
3. En caso de requerir SSL con certificado, ajustar la URL JDBC en Render.
4. Arrancar la aplicación.
5. Flyway aplicará automáticamente las migraciones.

6.2.Operaciones de despliegue

1. Confirmar que el repositorio contiene:
   - pom.xml
   - Dockerfile
   - .dockerignore
   - `src/ application-prod.properties`

2. Crear Web Service en Render.
3. Conectar repositorio GitHub:
   - https://github.com/dfleper/PRW-TT2025
4. Configurar variables de entorno.
5. Lanzar despliegue.
6. Verificar logs de arranque.

El Dockerfile realiza:

- Build del WAR con Maven.
- Copia de `target/TT2025.war`
- Ejecución con `java -jar` .

6.3.Condiciones y verificación de éxito

El despliegue se considera correcto si:

- El servicio aparece como Running en Render.
- No existen errores de conexión a MariaDB.
- Flyway aplica migraciones sin fallos.
- La aplicación escucha en el puerto asignado por Render.
- No existen errores críticos en logs.

7. CONTROL DE VERSIONES

Repositorio remoto:
https://github.com/dfleper/PRW-TT2025

El despliegue en Render se realiza directamente desde este repositorio mediante integración automática

8. OBSERVACIONES

- Hibernate en producción está configurado como:
  - `spring.jpa.hibernate.ddl-auto=validate`
- Flyway gestiona el versionado del esquema.
- Swagger está deshabilitado en producción.
- SMTP está deshabilitado en producción.
- .dockerignore excluye artefactos y documentación para optimizar el build.

GitHub Project Manager TT2025 PROYECTO-PRW-TT2025