Diseño Técnico

TT2025

https://github.com/dfleper

18/01/2026

El Documento Técnico de TT2025 describe la arquitectura del sistema, las tecnologías utilizadas y la estructura interna de la aplicación, detallando cómo está construido el proyecto a nivel funcional y técnico.

Contenido

Documento Técnico

Fase de diseño

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:

18/01/2026 – Versión 1.1

Versión	Fecha	Autor	Cambios Realizados	Motivos / Referencia
1.0	**21/12/2025**	**https//github.com/dfleper**	**Documento Técnico.**	**Entrega inicial UT01.3**
1.1	**18/01/2026**	**https//github.com/dfleper**	**Corrección herramientas de despliegue + control de calidad**	**Corrección por feedback (UT01.4)**

1. Introducción

Este documento técnico describe las tecnologías, herramientas y configuración mínima necesarias para desarrollar, ejecutar y desplegar el proyecto **Turbo Taller (TT2025)**, una aplicación web de gestión de citas para un taller de mecánica rápida. El objetivo es dejar trazabilidad de las decisiones técnicas (frontend, backend, base de datos, seguridad, despliegue y control de versiones) y servir como referencia para el desarrollo y la defensa del proyecto

2. Requisitos técnicos

2.1 Plataforma y herramientas de desarrollo software

2.1.1 Versionado y función (front-end)

- **HTML5**: estándar actual (sin versión fija). Se usa para estructurar páginas, formularios y componentes de navegación
- **CSS3**: estándar actual (sin versión fija). Se usa para estilos, maquetación responsive y consistencia visual.
- **JavaScript ES6+**: estándar moderno (sin versión fija). Se usa para validaciones en cliente, mejoras UX y peticiones asíncronas (Fetch/AJAX) cuando aplique.
- **Bootstrap 5.x (si se usa en el proyecto)**: framework CSS para acelerar maquetación responsive y reutilizar componentes UI con consistencia.
- **Thymeleaf 3.x (gestionado por Spring Boot 3.5.8)**: motor de plantillas del lado servidor para renderizar vistas, reutilizar fragmentos y enlazar datos con modelos de Spring.

Motivo de elección (front-end)

Se elige **HTML/CSS/JS** por ser tecnologías estándar, compatibles con cualquier navegador moderno y suficientes para el MVP. **Thymeleaf** encaja con Spring MVC y permite entregar una aplicación completa sin SPA, simplificando el desarrollo y la defensa académica. **Bootstrap** reduce tiempo de diseño y asegura un responsive correcto con componentes reutilizables.

Herramientas de apoyo

- **Navegadores de prueba:** Chrome/Firefox/Edge.
- **DevTools del navegador:** inspección, consola JS, red (Network) y performance.

2.1.2 Requisitos back-end

La aplicación se implementa en Java mediante Spring Boot, siguiendo un estilo MVC (controladores + servicios + acceso a datos) y separación clara de capas. Se emplea empaquetado **WAR** para despliegue en un contenedor externo (Tomcat) y también es posible ejecutar en modo embebido durante el desarrollo.

Elemento	Tecnología / versión	Función en el proyecto
Lenguaje	**Java17**	**Lenguaje base del backend. Compatible con Spring Boot 3.x.**
Framework	**Spring Boot 3.5.8**	**Arranque, configuración, MVC, inyección de dependencias y gestión de beans**
MVC Web	**spring-boot-starter-web**	**Controladores, routing, validación y renderizado de vistas.**
Plantillas	**spring-boot-starter-thymeleaf**	**Integración Thymeleaf con Spring.**
Seguridad	**spring-boot-starter-security**	**Autenticación, autorización y protección de endpoints.**
JDBC	**spring-boot-starter-jdbc**	**Acceso a datos vía JDBC (DataSource, JdbcTemplate).**
Thymeleaf extras	**thymeleaf-extras-springsecurity6**	**Uso de sec:authorize en vistas.**
Driver BD	**mariadb-java-client (runtime)**	**Conector JDBC para MariaDB.**
Servidor	**Tomcat (provided)**	**Despliegue en Tomcat externo (WAR).**
Build	**Maven**	**Gestión de dependencias y empaquetado (pom.xml).**
IDE	**Spring Tool Suite 4.32.0**	**IDE principal para desarrollo y depuración**

Motivo de elección (resumen)

- Spring Boot + Thymeleaf permite una aplicación web completa con MVC y vistas servidor, ideal para el alcance académico y un backoffice.
- Spring Security aporta control de acceso basado en roles y buenas prácticas por defecto (sesión, CSRF, filtros).
- JDBC ofrece control explícito y simplicidad para el acceso a datos en una primera versión, manteniendo compatibilidad con MariaDB.

2.1.3 Herramientas de despliegue y control de calidad

Herramientas de despliegue (DevOps)

- **Maven 3.9.x**: construcción del proyecto, gestión de dependencias y empaquetado **WAR** para despliegue en Tomcat.
- **Apache Tomcat 10.1.x**: contenedor de servlets compatible con Jakarta EE (requerido por Spring Boot 3.x) para despliegue externo.
- **Docker / Docker Compose (opcional recomendado)**: estandariza el entorno (por ejemplo MariaDB) y permite levantar el sistema de forma reproducible en local y pruebas.
- **Variables de entorno / perfiles Spring (dev/prod)**: separación de configuración por entorno (credenciales, URLs, flags de producción).

Control de calidad (QA) y verificación

- **JUnit 5 + Spring Boot Test**: pruebas unitarias e integración para validar servicios, controladores y reglas (por ejemplo disponibilidad de citas).
- **MockMvc (si aplica MVC)**: pruebas de controladores sin levantar servidor real.
- **Postman / Thunder Client**: pruebas manuales de endpoints durante desarrollo.
- **Checkstyle o Spotless (recomendado)**: formato consistente de código y prevención de estilos incorrectos.
- **JaCoCo (recomendado)**: informe de cobertura de tests para medir calidad mínima.
- **GitHub Actions (CI) (recomendado)**: pipeline automático (build + tests) en cada push/PR para evitar "romper" la rama principal.

Motivo de elección (despliegue y calidad)

Estas herramientas aseguran **reproducibilidad** (mismos resultados en distintos equipos), **trazabilidad** (builds y tests verificables), y **calidad mínima** (tests + estilo + cobertura). Además, **Maven + WAR + Tomcat** encaja con un despliegue clásico de servidor, fácil de justificar y ejecutar en un entorno académico.

2.2 Plataforma de ejecución

A continuación se indican los requisitos mínimos recomendados para ejecutar la aplicación en un entorno local o servidor.

Servidor web / contenedor	**Apache Tomcat 10.1.x (compatible con Jakarta EE / Spring Boot 3.x)**
Sistema operativo	**Windows 10/11 o Linux (Ubuntu 22.04+ recomendado)**
Runtime	**Java 17 (JRE/JDK)**
Base de datos	**MariaDB 12.1.2**
Observaciones	**En producción se recomienda activar HTTPS, variables de entorno para credenciales y rotación de logs.**

Directorios orientativos de despliegue

Desarrollo	**Proyecto local en STS. WAR generado en target/ (Maven).**
Pre-producción / pruebas	**Servidor de pruebas con Tomcat: $CATALINA_BASE/webapps/tt2025.war**
Producción	**Servidor de producción con Tomcat y configuración externa**

2.3 Puntos de acceso a la aplicación

Desarrollo	**http://localhost:8080/ (contexto raíz en ejecución local)**
Pre-producción	**https://pre.tt2025.local/ (orientativo)**
Producción	**https://tt2025.midominio.com/ (orientativo)**

3. Bases de datos

El sistema utiliza una base de datos relacional para almacenar usuarios, roles, clientes, vehículos, servicios, citas, órdenes de trabajo, piezas y facturación. La capa de acceso a datos se realiza mediante JDBC.

Base de datos	**MariaDB**
Versión	**12.1.2**
Esquema	**tt2025 (propuesto)**
Conector	**mariadb-java-client (JDBC, scope runtime)**
Políticas	**Integridad referencial mediante claves foráneas, índices en campos de búsqueda.**

4. Interfaces externas

En su alcance mínimo, Turbo Taller puede funcionar de forma autónoma. No obstante, contempla integraciones externas habituales para un entorno real.

- **Servicio de correo (SMTP):** envío de confirmaciones y recordatorios de cita (notificaciones).
- **Pasarela de pago** (opcional / futura): integración con proveedor externo para pagos online.
- **Servicios de mapas** (opcional): mostrar ubicación del taller y rutas (Google Maps u OpenStreetMap).

5. Seguridad

La aplicación implementa medidas de seguridad orientadas a proteger datos personales, controlar accesos y reducir riesgos comunes en aplicaciones web.

- **Autenticación:** login con credenciales; contraseñas almacenadas con hash seguro (BCrypt recomendado).
- **Autorización (RBAC):** acceso por roles. Roles previstos: CLIENTE, RECEPCIONISTA, MECANICO, JEFE_TALLER, ADMIN.
- **Protección CSRF:** habilitada por defecto en Spring Security para peticiones con sesión.
- **Gestión de sesión:** expiración por inactividad, invalidación al cerrar sesión y protección contra fijación de sesión.
- **Validación de entrada:** validación en servidor de formularios y parámetros, evitando inyección SQL (uso de consultas parametrizadas en JDBC).
- **Transporte seguro:** HTTPS obligatorio en pre-producción/producción; HSTS recomendado.
- **Registro y auditoría básica:** logs de accesos y acciones relevantes (crear/modificar/cancelar cita, cierre de orden).

- **Protección de datos:** mínimo privilegio; evitar exposición de datos en URLs; no registrar datos sensibles en logs.

6. Control de versiones

Se utiliza control de versiones distribuido para garantizar trazabilidad, revertir cambios y facilitar la entrega académica.

Herramienta	**Git**
Repositorio remoto	**GitHub (o equivalente)**
Estrategia recomendada	**Ramas feature/* para desarrollo, main para versiones estables. Commits pequeños y frecuentes.**
Etiquetado	**Tags por hitos**

7. Observaciones

- El proyecto se orienta a una primera versión (MVP) centrada en la reserva y gestión de citas, con backoffice para el taller.
- La elección de WAR y Tomcat externo facilita el despliegue en servidores clásicos. Para entornos cloud, puede considerarse también ejecución con Tomcat embebido (JAR) en una evolución.
- En caso de incorporar notificaciones reales, se recomienda configurar un servicio SMTP de pruebas (Mailtrap) en desarrollo y un proveedor SMTP en producción.
- Las versiones indicadas (STS 4.32.0, Java 17, Spring Boot 3.5.8 y MariaDB 12.1.2) deben mantenerse consistentes con el código entregado