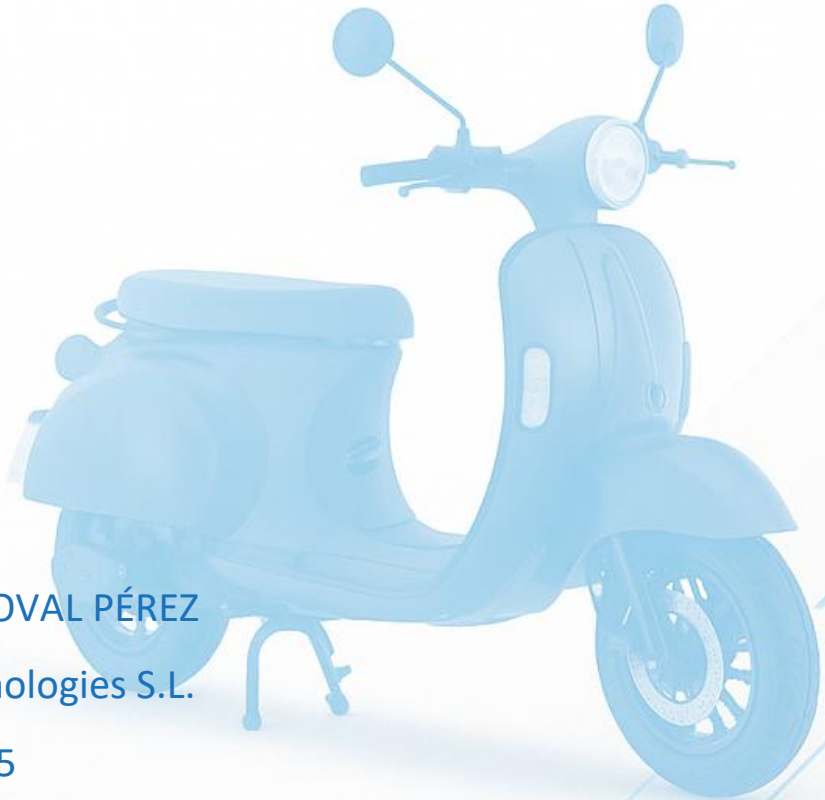
Quazzar Tasks

Documentación Técnica del Sistema



MATÍAS SANDOVAL PÉREZ

QuaZZar Technologies S.L.

Diciembre 2025

ÍNDICE

1. Introducción Técnica

Quazzar Tasks es una plataforma web integral desarrollada para **QuaZZar Technologies S.L.**, cuyo objetivo es **digitalizar la gestión de tareas industriales** en los distintos módulos productivos: *Pintura, Chasis, Premontaje y Montaje*.
El sistema unifica la trazabilidad de bastidores (VIN), los colores, las fechas de ejecución y el control de progreso de cada área.

La aplicación se construyó bajo una arquitectura **Angular + Laravel + MySQL**, ofreciendo una solución moderna, segura y escalable.
A través de un entorno intuitivo, los operarios pueden **iniciar, guardar, reanudar y finalizar tareas**, mientras los administradores gestionan usuarios y catálogos desde un panel dedicado.

Este proyecto representa un paso clave en la **transformación digital de los procesos industriales** de la empresa, optimizando tiempo, recursos y precisión.

2. Objetivos del Proyecto

2.1. Objetivos Funcionales

- Centralizar toda la información de tareas en una única aplicación web.
- Permitir guardar el progreso en cualquier punto (“tareas pendientes”).
- Asegurar continuidad entre sesiones y usuarios.
- Facilitar la gestión de usuarios y contraseñas desde el panel de administración.
- Proporcionar estadísticas básicas de avance y control de VIN.

2.2. Objetivos Técnicos

- Implementar una arquitectura **cliente-servidor desacoplada**.
- Desarrollar una **API REST segura** para el intercambio de datos.
- Estandarizar el código y seguir buenas prácticas de Angular y Laravel.
- Optimizar las consultas SQL y evitar redundancias.
- Garantizar persistencia y validación robusta de datos.

3. Arquitectura General del Sistema

La aplicación se compone de tres capas principales:

Capa	Tecnología	Función
Frontend (SPA)	Angular 17+, CSS.	Presentación, interacción con el usuario, formularios y vistas dinámicas.
Backend (API REST)	Laravel 10	Lógica de negocio, validaciones, controladores y persistencia.
Base de Datos	MySQL 8	Almacenamiento estructurado de tareas, usuarios y registros.

Diagrama lógico simplificado:

Usuario → Angular (Frontend) → Laravel API (Backend) → MySQL (BD)

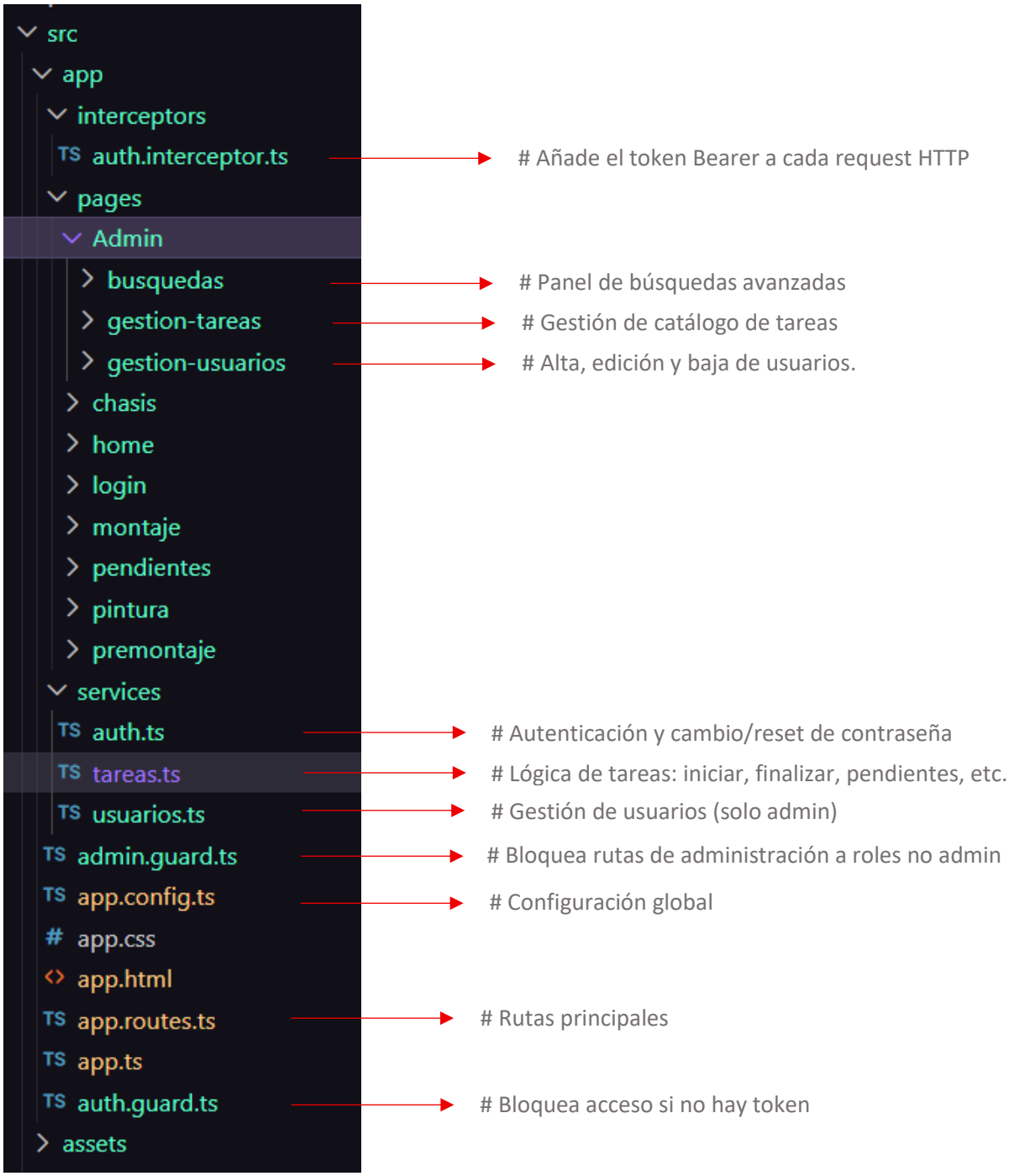
4. Tecnologías Utilizadas

- **Angular 17** — Framework SPA basado en componentes standalone.
- **TypeScript** — Tipado estricto y mantenimiento a largo plazo.
- **Laravel 10 (PHP 8.2)** — Backend con validación, routing y ORM Eloquent.
- **MySQL 8** — Base de datos relacional optimizada para trazabilidad.
- **TailwindCSS + estilos propios "qp-"** — Identidad visual corporativa.
- **RxJS / Signals API** — Gestión reactiva de estados y formularios.
- **Postman y Artisan CLI** — Pruebas y migraciones.

5. Estructura Global del Proyecto

5.1. Componentes Frontend

El frontend de **Quazzar Tasks** sigue una organización modular clara y escalable.
Cada grupo de páginas o funcionalidades se encapsula en una carpeta independiente dentro de src/app/pages/, mientras que los servicios compartidos residen en src/app/services/.



El frontend consume los servicios REST mediante observables y señales reactivas, aplicando una gestión centralizada del estado para sincronizar tareas entre vistas. El uso de localStorage y sessionStorage asegura la continuidad del trabajo incluso ante interrupciones.

5.2. Interacción con el backend

- Todas las vistas del módulo admin consumen los endpoints del backend (UsuarioController, TareaCatalogoController, TareasController) mediante los servicios usuarios.ts y tareas.ts.
- Cada formulario usa **Reactive Forms** y validaciones client-side antes de enviar cambios.

5.3. Componentes Backend

Estructura principal

- Controladores:
    - **AuthController**: login, validación y registro.
    - **UsuarioController**: gestión de usuarios, restablecimiento de contraseñas (POST /api/usuarios/{id}/reset-password).
    - **TareasController**: control de tareas por área (iniciar, finalizar, update, delete).
- Rutas definidas en **routes/api.php**.
- Validación mediante **Request** y reglas Rule::in(['pintura','chasis','premontaje','montaje']).

Flujo típico

1. Angular solicita /api/tareas?proceso=chasis.
2. Laravel responde con el catálogo filtrado (label, activa, proceso).
3. El frontend mapea el resultado a tareas visuales.
4. Al finalizar, el sistema envía fecha_inicio y fecha_fin al backend para registrar la ejecución.

5.4. Tareas Pendientes y Modales

Este módulo permite guardar y restaurar el estado de tareas en progreso, así como gestionar altas y ediciones de registros mediante modales interactivos. Implementa localStorage para persistencia local y componentes reutilizables con @Input() y @Output() para modales dinámicos.

6. Paginación y Búsquedas Avanzadas

El panel administrativo implementa un sistema de **paginación moderna y responsiva**, adaptado a la línea gráfica corporativa.
Incluye:

- Botones numerados con elipsis dinámicas.
- Indicadores activos/hover/disabled accesibles.
- Navegación por teclado y soporte responsive.
- Filtros en tiempo real por **fecha**, **trabajador** y **área de producción**.
- Ordenación ascendente/descendente con persistencia de estado.

Este diseño mejora la usabilidad y permite al personal administrativo realizar búsquedas y auditorías de forma rápida y precisa.

7. Seguridad y Roles

El sistema integra un modelo de **autenticación segura** y **control de acceso por roles**, garantizando la protección de los datos y la integridad de la información gestionada.

7.1. Autenticación

El acceso a la API se realiza mediante **token Bearer** generado al iniciar sesión.
Cada solicitud HTTP incluye el token en la cabecera, validado por el middleware de Laravel antes de procesarse.
Esto asegura sesiones independientes, evita accesos no autorizados y mantiene la confidencialidad del usuario.

7.2. Gestión de Roles

Existen dos niveles principales de acceso:

- **Administrador:** Puede gestionar usuarios, catálogos y tareas desde el panel de control.
- **Operario:** Accede únicamente a las tareas de su área (Premontaje, Chasis, Pintura o Montaje) y puede iniciar, guardar o finalizar tareas propias.

Los roles se almacenan en la base de datos y se verifican en cada petición mediante middleware, limitando las operaciones según los permisos asignados.

7.3. Buenas Prácticas

- Contraseñas cifradas con `Hash::make()` y validaciones `Request::validate()`.
- Tokens con expiración automática y cierre de sesión manual.
- Conexión obligatoria bajo protocolo **HTTPS**.

Este sistema refuerza la seguridad general de Quazzar Tasks, evitando accesos indebidos y garantizando la confidencialidad de los datos operativos.

8. Modelo de Base de Datos

La base de datos de **Quazzar Tasks** está diseñada bajo el modelo relacional de **MySQL 8**, priorizando la integridad referencial y la escalabilidad del sistema.

Tablas principales

Tabla	Descripción	Clave primaria	Relaciones
usuarios	Contiene la información de los operarios y administradores.	id	Relacionada con las tablas de tareas por el campo usuario_id.
tareas_premontaje, tareas_chasis, tareas_pintura, tareas_montaje	Almacenan las tareas específicas de cada módulo productivo.	id	FK usuario_id → usuarios.id.
catalogos_tareas	Define el conjunto de tareas base por proceso.	id	FK proceso enlaza con cada tabla de tareas.

Relaciones

- Un **usuario** puede tener **muchas tareas** en distintos módulos.
- Cada tarea pertenece a **un proceso (área)** y a **un usuario**.
- Las tareas pueden estar en estado *pendiente*, *en curso* o *finalizada*.

Este diseño permite realizar consultas transversales por área, fecha o trabajador, optimizando la trazabilidad.

9. Requisitos Técnicos y Entorno de Ejecución

Requisitos del servidor

Requisito	Versión mínima	Descripción
PHP	8.2	Necesario para ejecutar Laravel 10.
Composer	2.6	Gestión de dependencias del backend.
MySQL	8.0	Motor de base de datos relacional.
Node.js	20+	Requerido para Angular CLI.
NPM	10+	Gestión de paquetes frontend.
Servidor Web	Apache 2.4 o Nginx	Compatible con HTTPS y CORS.

Entorno de desarrollo

- **IDE recomendado:** Visual Studio Code.
- **Gestor de versiones:** Git (GitHub / GitLab).
- **Herramientas de prueba:** Postman, Artisan, PhpMyAdmin.
- **Despliegue:** Docker o servidor local con XAMPP/WAMP.

10. Mantenimiento y Futuras Ampliaciones

El sistema está diseñado bajo una arquitectura modular que permite su evolución sin afectar la estabilidad.
Entre las próximas mejoras previstas se incluyen:

- **Dashboard de métricas en tiempo real.**
- **Exportación de reportes en formato PDF/CSV.**
- **Integración con sistemas ERP internos de QuaZZar Technologies.**
- **Módulo de notificaciones automáticas por correo o intranet.**

El mantenimiento preventivo incluye actualizaciones de seguridad de Laravel, Angular y dependencias NPM/Composer, garantizando un entorno actualizado y estable.

11. Experiencia de Usuario y Diseño

El diseño visual de Quazzar Tasks sigue los lineamientos de **QuaZZar Technologies S.L.**:

- **Color corporativo:** Azul QP #1D71B8.
- **Tipografía:** *Montserrat*, adaptada a entorno industrial.
- **Estructura:** Cabecera fija con logotipo, menú lateral y área de trabajo limpia.
- **Componentes UI:**
    - Botones redondeados con sombras suaves.
    - Tablas minimalistas y legibles.
    - Modales personalizados con confirmación de acciones.

La interfaz equilibra **claridad, funcionalidad y estética profesional**, optimizando la experiencia de uso tanto en planta como en oficina.

12. Resultados y Conclusiones

El desarrollo de **Quazzar Tasks** ha logrado los siguientes resultados:

- Digitalización completa del flujo de tareas industriales.
- Centralización de datos y mejora en la trazabilidad de procesos.
- Reducción de tiempos muertos y errores humanos.
- Integración total entre operarios y supervisores bajo un mismo entorno.

Conclusión general

Quazzar Tasks constituye una solución sólida, escalable y alineada con la visión tecnológica de QuaZZar Technologies S.L.
Su arquitectura modular y su diseño adaptable garantizan su evolución futura hacia funcionalidades más avanzadas.

La aplicación se consolida como **una herramienta estratégica dentro del ecosistema digital de la empresa**, impulsando la eficiencia y la innovación en el entorno industrial.