Ficha Técnica

MyGestor

[MGSCC]. [My Gestor]
[MGSCC]. [Sistema de Control de Costes]

Nombre del fichero:	DAW_PRW_[MGSCC]_4. Ficha técnica.odt
Fecha de esta versión:	18/01/2026

Historial de revisiones

Fecha	Descripción	Autor
[21/12/2025]	[Planteamiento inicial]	[Sidney Cadahia Cardoso]
[25/12/2025]	[Diagrama de casos de uso]	[Sidney Cadahia Cardoso]
[25/12/2025]	[Diagrama E/R o Diagrama Relacional]	[Sidney Cadahia Cardoso]
[18/01/2025]	[Ampliación del Diagrama Relacional]	[Sidney Cadahia Cardoso]
[18/01/2025]	[Ficha Técnica]	[Sidney Cadahia Cardoso]

ÍNDICE

1 INTRODUCCIÓN

El presente documento técnico describe las tecnologías y herramientas utilizadas en el desarrollo del sistema My Gestor (MGSCC), una aplicación web orientada a la gestión económica y administrativa de una liga de fútbol. El sistema permite registrar facturas emitidas entre entidades (proveedores, clubes y jugadores), gestionar el detalle de conceptos, registrar pagos manuales con validación, asociar documentación adjunta y administrar incidencias mediante tickets con chat integrado.
El objetivo de este documento es identificar de forma clara y estructurada todos los componentes tecnológicos implicados en el proyecto, indicando su función dentro del sistema y justificando su elección. Este documento servirá como referencia durante el desarrollo y como soporte en la defensa del proyecto, asegurando que la implementación coincide con las tecnologías declaradas.

2 REQUISITOS TÉCNICOS

2.1 Plataforma y herramientas de desarrollo software

2.1.1. Requisitos front-end

El proyecto implementa la interfaz de usuario mediante Vue 3 y componentes renderizados a través de Inertia.js, manteniendo el control del enrutado y la lógica principal en Laravel.

I) Tecnologías / framework

- *HTML5*

Función: Estructuración del contenido de las vistas y formularios de la aplicación.

-

- *CSS3*

Función: Maquetación y estilos visuales de la interfaz, incluyendo diseño responsive para facilitar el uso desde distintos dispositivos.

- *JavaScript(ES6+)*

Función: Lógica del lado del cliente, interacción dinámica en formularios y actualización de componentes en la interfaz.

- *Vue.js3*

Función: Framework principal para el desarrollo del frontend mediante componentes reutilizables. Uso en el proyecto: Construcción de dashboards según perfil (proveedor, gestor, jugador), formularios de facturación y vistas de incidencias con chat.

- *Inertia.js*

Función: Capa de integración entre Laravel y Vue que permite construir una aplicación con experiencia SPA sin exponer una API pública.
Uso en el proyecto: Renderizado de vistas Vue desde controladores Laravel, navegación sin recargas completas y paso de datos mediante props.

- *Vite*

Función: Herramienta de compilación y empaquetado de recursos frontend (JS/CSS).
Uso en el proyecto: Compilación en desarrollo con recarga rápida (HMR) y generación de assets optimizados para ejecución.

II) Otras Herramientas de desarrollo, depuración, etc

- *DevTools del navegador (Chrome/Firefox)*

Función: Depuración de componentes, revisión de errores JavaScript, inspección de estilos y validación de peticiones.

- *Cursor*

Función: Editor principal de código, con extensiones para Vue, PHP y Laravel.

2.1.2. Requisitos back-end

El backend se desarrolla mediante Laravel 12, implementando la lógica de negocio, persistencia de datos, validaciones, control de acceso por roles y notificaciones del sistema.

I) Plataforma de desarrollo

a) Framework, runtime y versiones

- PHP 8.3
  Runtime necesario para la ejecución del backend del proyecto, compatible con los requisitos mínimos de Laravel 12.
- Laravel 12
  Framework principal del backend. Se utilizará para la gestión de rutas, controladores, validaciones, autenticación, autorización por roles, notificaciones y acceso a base de datos.
- Eloquent ORM (incluido en Laravel 12)
  ORM utilizado para el mapeo objeto-relacional y la gestión de entidades del sistema (usuarios, clubes, proveedores, facturas, pagos, incidencias, documentos y modificaciones).
- MySQL 8.4
  Sistema gestor de base de datos relacional utilizado para la persistencia de datos del sistema.
- Node.js (LTS) + NPM
  Herramientas necesarias para la compilación y ejecución del frontend mediante Vite y Vue 3, integrados en Laravel mediante Inertia.js.

b) Esquema de desarrollo

- Arquitectura MVC (Modelo–Vista–Controlador)
  El proyecto se desarrolla siguiendo el patrón MVC de Laravel, donde la lógica de negocio se implementa en controladores y servicios, el acceso a datos se gestiona mediante modelos Eloquent y las vistas se renderizan mediante Inertia.js.
- Aplicación monolítica con Inertia.js
  La aplicación se implementa como un sistema monolítico, evitando el uso de una API REST pública. La comunicación entre backend y frontend se realiza mediante Inertia, enviando datos desde Laravel a componentes Vue en formato de props y manteniendo una experiencia de navegación tipo SPA.

c) IDE

- Cursor 2.20
  Entorno de desarrollo utilizado para la edición del código fuente, con soporte para PHP, Laravel, JavaScript y Vue.

d) Otras herramientas o gestores de proyecto

- Composer (gestor de dependencias PHP)
  Utilizado para la instalación y actualización de librerías y dependencias del backend.
- Git (control de versiones)
  Utilizado para el control de cambios del proyecto, permitiendo mantener trazabilidad del desarrollo mediante commits.
- Laragon (entorno de desarrollo local en Windows)
  Utilizado para ejecutar el servidor web (Apache), configurar PHP y gestionar MySQL en local, facilitando el desarrollo y pruebas del proyecto.
- Mailpit (correo local de desarrollo)
  Utilizado para capturar y visualizar correos enviados por la aplicación en entorno local, permitiendo validar notificaciones sin necesidad de un servidor SMTP externo.

e) Otras dependencias

- Laravel Breeze (opcional según configuración inicial)
  Paquete utilizado para generar la base de autenticación (login, logout, sesiones) e integración inicial con Inertia.js y Vue.
- Laravel Notifications (incluido en Laravel)
  Sistema utilizado para el envío de notificaciones por correo electrónico ante eventos del sistema (facturas creadas, pagos registrados, pagos validados/rechazados, incidencias).
- Laravel Filesystem / Storage (incluido en Laravel)
  Sistema utilizado para la gestión de documentos adjuntos (facturas, pagos, incidencias), incluyendo validación de tamaño máximo de archivo y almacenamiento en disco.
- Laravel Soft Deletes (incluido en Laravel)
  Funcionalidad utilizada para borrado lógico de registros mediante el campo deleted_at, preservando la integridad histórica de la información.

2.2 Plataforma de Ejecución

2.2.1. Servidor

La aplicación se ejecuta en un entorno local de desarrollo configurado mediante Laragon.

Servidor web y versión: Apache (Laragon) 8.3
Sistema operativo y versión: Windows 11
Runtime/Interprete y versión: PHP 8.3
Observaciones a la configuración: Laragon proporciona un entorno integrado con servidor web y base de datos, permitiendo configurar dominios locales (por ejemplo, mygestor.test) y ejecutar la aplicación Laravel con facilidad.

2.2.2. Alojamiento físico/ Entornos de ejecución

El proyecto no contempla un despliegue en producción durante la fase actual, por lo que únicamente se define el entorno local de desarrollo.

Desarrollo: Windows + Laragon
Pre-Producción/Pruebas: N/A
Producción: N/A

2.3 Puntos de acceso a la aplicación

A continuación se indican los puntos de acceso url a la aplicación [solo si se conocieran]:

Desarrollo: http://mygestor.test
Pre-Producción: N/A
Producción: N/A

3 Bases de Datos

Base de datos: [MySql]
Versión:[8.4.3]
Esquema: my_gestor

4 Interfaces externas

En el alcance actual del proyecto no se integran servicios externos de terceros (pasarelas de pago, APIs bancarias o plataformas externas). El registro de pagos se realiza manualmente y se valida dentro del sistema mediante justificantes adjuntos.
El sistema está preparado para incorporar integraciones futuras si se amplía el alcance del proyecto, pero dichas integraciones no forman parte de esta fase.

5 Seguridad

Las medidas de seguridad y control consideradas en el proyecto son:

- Autenticación mediante sesiones de Laravel (login/logout).
- Control de acceso basado en roles (Administrador, Proveedor, Gestor de club y Jugador).
- Protección CSRF integrada en Laravel para formularios y peticiones.
- Validación de datos en backend para garantizar consistencia e integridad.
- Soft deletes (deleted_at) para evitar eliminación física inmediata y conservar trazabilidad.
- Gestión segura de documentos adjuntos, incluyendo validación de tamaño máximo (6MB) y control de almacenamiento.
- Registro de último inicio de sesión (last_login_at) como medida de trazabilidad y cumplimiento de buenas prácticas de seguridad y protección de datos.

6 Control de versiones

El proyecto utiliza Git como sistema de control de versiones para mantener el historial de cambios y facilitar el seguimiento del desarrollo.
Repositorio remoto: GitHub
Uso: almacenamiento del código, control de versiones y trazabilidad del proyecto.

Enlace del repositorio:
https://github.com/VacuumCoyote43/My-Gestor-SCC

7 Observaciones

El sistema se desarrolla como aplicación monolítica con Laravel 12 + Inertia.js + Vue 3, lo que permite una experiencia de usuario fluida sin necesidad de exponer una API REST pública.

El entorno de desarrollo se ejecuta en Windows mediante Laragon, facilitando la configuración del servidor web y MySQL.

Para el envío de correos en local se utiliza Mailpit, permitiendo validar notificaciones sin dependencias externas.

El modelo de datos incorpora auditoría mediante una tabla de modificaciones polimórfica, mejorando la trazabilidad de cambios y validaciones dentro del sistema.