

PROYECTO



Ismael Rodriguez Cuenca

Digitech FP

GSFP en Desarrollo Aplicaciones Multiplataforma

Javier Valverde Nolting

Málaga, España

16 Junio 2025

Contenido

- 1 Identificación de necesidades
 - 1.1 Contextualización
 - 1.2 Justificación
 - 1.3 Otros aspectos
- 2 Diseño del proyecto
 - 2.1 Contenido
 - 2.2 Objetivos y recursos
 - 2.3 Viabilidad económica
 - 2.4 Modelo de solución
- 3 Ejecución del proyecto
 - 3.1 Planificación temporal
 - 3.2 Revisión de recursos

1 Identificación de necesidades

1.1 Contextualización

Existen diversos ERP para la gestión de empresas de reparaciones, lamentablemente orientados a empresas medianas o grandes debido a que estos suelen conllevar un coste que para los pequeños comercios no se sería asumible, generalmente usan una aplicación de facturación sencilla para controlar las finanzas pero no suelen ser tan precisos con el control de las reparaciones por falta de un software de gestión de bajo o cero coste, que sea sencillo de usar, o que no requiere una configuración compleja para el uso del mismo en este ámbito.

Precisamente en el contexto de las limitaciones económicas... Durante el desarrollo de este proyecto se tendrá en cuenta aquellas tiendas que generan poco beneficio y que no pueden invertir en un software personalizado o profesional debido al bajo presupuesto de las mismas. Estará enfocada a la reparación de dispositivos con IMEI/SN, permitiendo ser usada por comercios que se dediquen a la reparación de dispositivos electrónicos tales como móviles, tablets, PC, portátiles, consolas, dispositivos de movilidad tales como patinetes, hoverboards, skates, monociclos eléctricos, pudiendo incluir aquellos no electrónicos, entre otra gran variedad de dispositivos...

El proyecto estará enfocado en cubrir las necesidades de aquellos pequeños comercios que puedan estar haciendo uso de métodos más rudimentarios para la gestión y facturación de las reparaciones (como hojas de cálculo o registros en papel), buscando aumentar significativamente la eficiencia y la visualización clara de sus finanzas. A la misma vez, mejorará su control detallado sobre cada reparación, permitiéndoles desarrollar un servicio más eficaz, disminuir e incluso eliminar cualquier posible error o intento de engaño por parte de los clientes, lo que a su vez puede mejorar la satisfacción del cliente y la rentabilidad del negocio.

1.2 Justificación

ERPair permitirá enmendar todos los problemas a los que se pueden enfrentar los comercios de baja facturación que hacen uso de programas de facturación sin seguimiento de reparaciones, ayudándolos a tener una facturación más dinámica y adaptada a su flujo de trabajo. Al estar orientado específicamente a reparaciones, eliminará las limitaciones derivadas del uso de otros sistemas no enfocados en reparación, mejorando significativamente la visión de sus finanzas, aclarando posibles malversaciones por parte de los clientes gracias al registro de cada reparación, y haciéndolo accesible para todas ellas al ser una solución de bajo o nulo coste.

- El proyecto sería principalmente gratuito, con suscripción premium a distintas características, para empezar echemos un vistazo rápido a la competencia:

Nombre	Coste	Divisa	Orientado al mercado angloparlante	Otros
RepairShopr	720	USD	Sí	-
RepairCMS	300	EUR	Sí	Limitado a 30 tickets/mes
Orca	240	EUR	No	Coste módulo SAT a parte
myGestion	535	EUR	No	-
RepairDesk	1200	USD	Sí	-

El hipotético ingreso de la aplicación vendría en la opción gratuita a través de banner publicitarios, pudiendo ser eliminados en la opción premium que agregaría la implementación de VeriFact, eliminación de marcas de agua en la generación de la factura, aviso automático por correo o Whatsapp y copia en la nube, entre otras características...

El proyecto se desarrollará con Laravel 12, Filament y con una base de datos mysql. Contendría las siguiente funcionalidades:

Facturación: Gestión ágil de ingresos, gastos, caja registradora y facturación, incluyendo la generación de facturas, la administración de pagos y el control de anticipos.

Clientes: Registro de Número de Identificación Fiscal, datos personales, información de contacto y los dispositivos asociados a cada cliente.

Dispositivos: Identificación única por IMEI o Número de Serie, además de marca, modelo, color y código de desbloqueo.

Parte de Reparación: Seguimiento detallado con número identificativo, avería detectada o motivo de visita, estado del terminal (aspecto físico y funcionalidades), piezas utilizadas, coste total y observaciones post-reparación.

Piezas: Información completa de cada pieza, incluyendo marcas y modelos compatibles, coste de adquisición y Precio de Venta al Público (PVP), proveedor y niveles de stock.

Técnico: Asignación del usuario responsable de la reparación.

Usuario: Gestión de credenciales, historial de dispositivos reparados y ventas realizadas por cada usuario del sistema.

Auditoría: Registro de fechas de creación y modificación de la información, motivos de los cambios y el usuario que los realizó.

Informes: Generación de informes actualizados y totales sobre gastos e ingresos para el análisis financiero.

1.3 Otros aspectos

La puesta en marcha de un ERP debería tener en cuenta una alta cantidad de obligaciones fiscales y legales para poder llevar a cabo el proyecto y tener un uso en la vida real. Veamos brevemente cuáles serían:

Solicitud de la Denominación Social: Obtener un certificado negativo de denominación social del Registro Mercantil Central para asegurar la disponibilidad del nombre "ERPair S.L.".

Apertura de una Cuenta Bancaria: Abrir una cuenta bancaria a nombre de la sociedad en constitución y depositar el capital social mínimo.

Elaboración de los Estatutos Sociales: Redactar los estatutos de la sociedad, que regirán su funcionamiento interno (objeto social, domicilio, órganos de administración, etc.).

Firma de la Escritura Pública de Constitución: Formalizar la constitución de la sociedad ante notario mediante la firma de la escritura pública.

Obtención del NIF Provisional: Solicitar el Número de Identificación Fiscal (NIF) provisional a la Agencia Estatal de Administración Tributaria (AEAT).

Liquidación del Impuesto sobre Transmisiones Patrimoniales y Actos Jurídicos Documentados (ITP-AJD): Aunque generalmente exenta para la constitución de sociedades, se debe presentar el modelo correspondiente.

Inscripción en el Registro Mercantil: Inscribir la escritura pública de constitución en el Registro Mercantil Provincial correspondiente.

Obtención del NIF Definitivo: Una vez inscrita la sociedad en el Registro Mercantil, solicitar el NIF definitivo a la AEAT.

Alta en el Censo de Empresarios, Profesionales y Retenedores (Modelo 036/037): Declarar el inicio de la actividad económica y las obligaciones fiscales de la empresa.

PRL (Prevención de riesgos laborales): Evaluación de los riesgos laborales y plan de prevención de riesgos y enfermedad laboral.

Posibles ayudas o subvenciones: Es crucial llevar a cabo una investigación de toda subvención disponible ya que esta aceleraría el crecimiento y puesta en marcha del proyecto. Exploramos distintos puntos de información como la Cámara de Comercio, páginas de organismos gubernamentales local, autonómico o estatal; también podríamos considerar la opción de recurrir a consultorías especializadas.

A Nivel Nacional:

- **Programa Kit Digital:** Ofrece bonos digitales para que las PYMEs y autónomos inviertan en la adopción de soluciones digitales. ERPair, como desarrollador de una solución de gestión, podría beneficiarse del gran conocido Kit Digital para su propia digitalización:
- **Deducciones Fiscales por I+D+i:** Podría beneficiarse de deducciones en el Impuesto sobre Sociedades por los gastos e inversiones realizados en actividades de investigación y desarrollo para la mejora o creación de nuevas funcionalidades de su software.
- **Líneas ICO (Instituto de Crédito Oficial):** El ICO ofrece líneas de financiación con condiciones ventajosas para empresas y emprendedores, incluyendo aquellas del sector tecnológico y para proyectos de digitalización.

A Nivel Autonómico:

- **Digitalización de Empresas:** Programas similares al Kit Digital a nivel regional o complementarios.
- **Innovación y Desarrollo Tecnológico:** Subvenciones para proyectos de desarrollo de software, inteligencia artificial, etc.
- **Emprendimiento:** Ayudas para la creación y consolidación de nuevas empresas innovadoras.
- **Formación y Capacitación Digital:** Subvenciones para la formación de los empleados en habilidades digitales.

A Nivel Europeo:

- **Programa Horizonte Europa:** El principal programa de financiación de i+D de la Unión Europea.
- **Programa Digital Europe:** Financia proyectos para impulsar la transformación digital de Europa, incluyendo áreas como IAI, la ciberseguridad y la computación de alto rendimiento.

Elaboración de plan para la ejecución del proyecto:

Fase 1: Análisis y Definición

- Analizar el flujo de trabajo de tiendas de reparación (Recepción, Diagnóstico, Presupuesto, Reparación, Entrega, Facturación).
- Establecer el stack tecnológico: Laravel 12, Filament, MySQL, Git.

Fase 2: Diseño

- Documentar funcionalidades detalladas.
- Diseñar la estructura de la base de datos (MER y esquema relacional).
- Planificar la estructura básica de la interfaz de usuario (UI).

Fase 3: Desarrollo

- Implementar modelos, controladores y rutas en Laravel.
- Desarrollar la lógica de negocio principal (CRUDs, facturación, reparaciones, etc.).
- Implementar la interfaz de administración básica con Filament.

Fase 4: Testeo

- Realizar pruebas unitarias e de integración.
- Llevar a cabo pruebas para identificar errores y problemas de usabilidad.

Fase 5: Mejora y Personalización

- Corregir errores basados en el testeo.
- Personalizar la interfaz de usuario (UI) y mejorar la experiencia del usuario (UX).

Fase 6: Implementación de Monetización

- Integrar anuncios (versión gratuita).
- Implementar la lógica de suscripción premium y funcionalidades asociadas.

2 Diseño del proyecto

2.1 Contenido

Veamos los distintos aspectos y su propósito dentro del software ERPair:

- **Facturación:** Módulo encargado de la gestión de documentos económicos, permitiendo la creación y administración de:
 - **Facturas:** Documentos de cobro definitivos por los servicios o productos ofrecidos.
 - **Proformas:** Presupuestos no vinculantes para informar al cliente del coste estimado de la reparación.
 - **Anticipos:** Registro y gestión de pagos parciales realizados por los clientes antes de la finalización del servicio.
- **Cientes:** Sección dedicada a la administración de la información de los clientes:
 - **CRUD clientes:** Funcionalidades para Crear, Leer, Actualizar y Borrar registros de clientes, incluyendo datos fiscales, personales y de contacto.
 - **Listar dispositivos:** Visualización de los dispositivos asociados a cada cliente, facilitando el acceso rápido al historial de reparaciones.
- **Marcas:** Gestión de un catálogo de fabricantes de dispositivos:
 - **Guardado de listado de Marcas Registradas:** Almacenamiento de una lista de marcas comunes para facilitar la selección al registrar dispositivos o piezas.
 - **CRUD de marcas en determinadas vistas:** Posibilidad de crear, leer, actualizar y borrar marcas desde las interfaces donde sea necesario (ej., al registrar un nuevo dispositivo o pieza).
- **Roles:** Administración de los diferentes roles de usuario dentro del sistema:
 - **CRUD de roles asociados a distintos usuarios:** Definición y gestión de permisos y responsabilidades para cada tipo de usuario (comercial, técnico, etc.).
- **Tipos:** Clasificación de los diferentes tipos de ítems que se gestionan en el sistema:
 - **CRUD de tipos de ítems:** Creación, lectura, actualización y borrado de categorías de ítems (ej., "servicio", "repuesto", "accesorio").
- **Cientes:** (Se repite, se entiende que es la gestión de la entidad Cliente)
 - **CRU cliente:** (Crear, Leer, Actualizar cliente)

- **Acceso a CRUD de dispositivos asociados:** Desde la vista del cliente, acceso directo a la gestión de sus dispositivos.
- **Impuesto:** Gestión de los diferentes tipos de impuestos aplicables:
 - **CRUD de tipos de impuesto asociados al tipo de ítem:** Definición de los tipos de IVA u otros impuestos y su vinculación a las categorías de ítems.
 - **Métodos:** Administración de las formas de pago aceptadas.
 - **CRUD de métodos de pagos asociados a las facturas:** Registro de las diferentes maneras en que los clientes pueden realizar los pagos (efectivo, tarjeta, etc.).
- **Modelos:** Gestión de los modelos específicos de los dispositivos:
 - **CRUD de modelos:** Creación, lectura, actualización y borrado de modelos de dispositivos (ej., "iPhone 16", "Samsung Galaxy S25 Ultra").
 - **Listado ítems de tipo pieza asociados:** Visualización de las piezas compatibles con cada modelo de dispositivo.
- **Dispositivos:** Gestión de la información de cada dispositivo reparado:
 - **CRU dispositivos:** Creación, lectura y actualización de la información de los dispositivos (IMEI/SN, marca, modelo, color, código de desbloqueo).
 - **Acceso a partes asociados:** Desde la vista del dispositivo, acceso directo a los partes de reparación relacionados.
- **Partes:** Seguimiento detallado de cada intervención de reparación:
 - **CRU parte:** Creación, lectura y actualización de la información del parte de reparación (número identificativo, avería, estado del terminal, etc.).
 - **Agregar ítems asociados a un parte:** Vinculación de las piezas utilizadas y la mano de obra al parte de reparación.
 - **Generar factura, devoluciones, anticipos o proforma:** Creación de documentos económicos directamente desde el parte de reparación.
 - **Tiene relación con una tienda y con un cierre:** Vinculación del parte a la tienda donde se realizó y al cierre de caja correspondiente.
- **Items:** Gestión del inventario de productos y servicios:
 - **CRUD items:** Creación, lectura, actualización y borrado de ítems (tanto piezas como servicios).
 - **Tienen relación tipo de ítem:** Vinculación de cada ítem a su categoría (ej., "repuesto", "mano de obra").

- **Cierres:** Registro de la información de los cierres de caja:
 - **Registra la información de los cierres de partes:** (Se entiende que registra el cierre económico asociado a los partes finalizados).
 - **Tiene un único parte asociado:** (Esto parece incorrecto, un cierre debería poder agrupar múltiples partes finalizados y facturados. Se debería revisar esta relación).
- **Facturas:** Gestión de los documentos de cobro finales:
 - **Registro de ítems asociados a una factura:** Vinculación de los productos o servicios incluidos en la factura.
 - **Puede o no tener relación con un parte:** Permite generar facturas directamente (ej., por la venta de un accesorio) o a partir de un parte de reparación.
- **Empresas:** Gestión de la información del propio negocio:
 - **CRUD empresas:** Creación, lectura, actualización y borrado de los datos de la tienda o empresa (nombre, dirección, datos fiscales).
 - **Puede estar relacionado con facturas:** Vinculación de la información de la empresa emisora a las facturas.
- **Cajas:** Control del flujo de efectivo diario:
 - **Genera un registro del dinero calculado gracias a la facturas con fecha actual, comparándolo con los datos introducidos por el usuario:** Permite comparar el efectivo teórico (según las facturas) con el efectivo real al cierre de caja.
- **Tiendas:** Gestión de múltiples ubicaciones del negocio (si aplica):
 - **CRUD tiendas:** Creación, lectura, actualización y borrado de la información de las diferentes tiendas.
 - **Listar partes asociados:** Visualización de los partes de reparación gestionados en cada tienda.
- **Usuarios:** Administración de las cuentas de acceso al sistema:
 - **CRUD usuarios:** Creación, lectura, actualización y borrado de las cuentas de los empleados.
 - **Acceso a tiendas asociados, partes, etc.:** Vinculación de los usuarios a las tiendas donde trabajan y visualización de su actividad (partes asignados, etc.).
- **usuarios_tiendas:** Tabla de relación muchos a muchos entre usuarios y tiendas, permitiendo que un usuario trabaje en varias tiendas y una tienda tenga varios usuarios.

- **items_modelos:** Tabla de relación muchos a muchos entre ítems (piezas) y modelos de dispositivos, indicando la compatibilidad de las piezas.
- **lineas_facturas:** Tabla de relación muchos a muchos entre ítems y facturas, detallando los ítems incluidos en cada factura y su cantidad.
- **lineas_partes:** Tabla de relación muchos a muchos entre ítems y partes de reparación, detallando las piezas y la mano de obra utilizada en cada reparación.

Estudio de viabilidad técnica:

Lenguaje de programación: Laravel 12 es un framework PHP de alto nivel. Su arquitectura MVC (Modelo-Vista-Controlador) facilita la organización del código, la escalabilidad y el mantenimiento a largo plazo. La amplia comunidad de desarrolladores de Laravel asegura una gran cantidad de recursos de aprendizaje y ayuda.

Framework de administración: Filament es un framework de panel de administración para Laravel que simplifica enormemente la creación de interfaces, permitirá generar rápidamente los CRUD (Create, Read, Update, Delete) y las vistas necesarias para nuestro ERP.

Base de datos: MySQL es un sistema de gestión de bases de datos de código abierto, ampliamente probado, escalable y utilizado en una gran variedad de aplicaciones web. Ofrece un buen rendimiento y escalabilidad.

Control de versiones: Git es un sistema de control de versiones usado dentro del proyecto de ERPair, esencial para la gestión del código fuente y el seguimiento de los cambios realizados durante el desarrollo. Permite “*recoger cable*” cuando es necesario.

Las fases clave para la ejecución del proyecto ERPair, las hemos podido ver anteriormente en el punto 1.3.3, se detallan a continuación:

Fase 1: Análisis y Definición: Investigación del mercado de talleres, identificación de problemas y definición de requisitos funcionales y no funcionales para ERPair. Se establece el alcance y los criterios de éxito.

Fase 2: Diseño: Traducción de necesidades en diseño técnico detallado: estructura de base de datos, interfaz de usuario (UI), experiencia de usuario (UX) y arquitectura del software con sus módulos.

Fase 3: Desarrollo: Construcción del software. Desarrollo del backend con Laravel 12 y frontend de administración con Filament. Implementación de funcionalidades (clientes, dispositivos, reparaciones, facturación, inventario, etc.).

Fase 4: Testeo: Pruebas exhaustivas: unitarias, de integración, de usuario (UAT), rendimiento y seguridad para identificar errores y problemas de usabilidad.

Fase 5: Mejora y Personalización: Corrección de errores, refactorización del código y posible implementación de mejoras o personalizaciones basadas en el feedback del testeo. Fase iterativa.

Fase 6: Implementación de monetización: Integración de banners publicitarios (versión gratuita) y sistema de suscripción/funcionalidades premium (versión de pago). Aseguramiento de su correcto funcionamiento sin afectar la experiencia del usuario.

2.2 Objetivos y recursos

1. Objetivos del proyecto:

- ERP de bajo coste para pequeños talleres de reparación.
- Funcionalidades clave: partes de reparación (IMEI/SN), inventario, facturación, clientes, informes básicos.
- Versión gratuita y premium asequible.
- Interfaz sencilla y fácil de usar.
- Mejorar la eficiencia, control y transparencia de sus finanzas
- Modelo freemium sostenible.
- Cumplimiento legal básico en facturación.
- Arquitectura escalable.

2. Recursos hardware y software.

- **Hardware:** Estación de trabajo, servidor de pruebas (opcional), hosting.
- **Software:** SO moderno, IDE, PHP, Composer, MySQL, cliente DB, navegador, Git, (opcional: diseño UI/UX).

3. Recursos materiales y personales.

- **Materiales:** Internet, documentación, recursos de aprendizaje.
- **Personales:** Desarrolladores/testers, marketing/ventas, asesoramiento legal/fiscal

2.3 Viabilidad económica

Coste de desarrollo: Estimando un coste por hora de desarrollo de 20€ (este valor es orientativo y representa el coste de contratación de un desarrollador), el coste de desarrollo inicial sería de: $120 \text{ horas} \times 16\text{€/hora} = \mathbf{2400 \text{ € brutos}}$.

Coste de los equipos: Contaremos para el cálculo la inversión en un equipo y entorno para el desarrollo:

- Portatil: **800€**
- Monitor: **120€**
- Teclado y ratón ergonómicos: **50€**
- Reposamuñecas: **20€**
- Mesa de trabajo: **100€**
- Silla de escritorio: **60€**
- Reposapiés: **20€**
- Conexión a Internet: 20€/mes. **240€/año**

Costes de software : Se asume un coste de 0€, ya que se utilizará software libre como Laravel, Filament, MySQL, Visual Code, etc...

Costes de infraestructura:

- **Dominio:** El registro de un dominio ".es" podría costar alrededor de 10€, aunque con el plan de hosting el primer año es gratuito.
- **Hostinger:** Plan de hosting "Premium" Tiene un coste anual de **144€/anuales** (12€/mes).

Costes legales y fiscales (iniciales): Los gastos asociados a la constitución de la sociedad "ERPair S.L." junto con los trámites asociados a la creación de la misma y los servicios notariales, podrían ser hasta **600€**.

Costes de marketing básico (iniciales): Un presupuesto muy básico para dar a conocer la aplicación en su lanzamiento comenzaría con la creación de perfiles en redes sociales y alguna campaña publicitaria online de bajo coste (ej., anuncios en redes sociales). Estimaremos un gasto inicial de **120 €/año**.

Esto nos dejaría con un total de **4674€**.

Veamos la financiación que podemos tener a nuestra disposición:

- Financiación propia (ahorros): Utilizar ahorros propios para cubrir los costes iniciales. Implicaría asumir el riesgo financiero directamente.
- Financiación por *Love Money*: Este tipo de financiación es aquella que pudiera venir por parte de familia y amigos, que estén dispuestos a aportar capital a la empresa.
- Préstamo personal: Existe la posibilidad de pedir un préstamo para inyectar el capital en el proyecto. Este tipo contiene más riesgo debido a los intereses, plazos o incluso posibles recargos por pago tardío/impago

En caso de que ninguno de estos tipos de financiación “propios” estuvieran disponibles, existen otros tipos de financiación tales como:

- Ayudas y Subvenciones Públicas: Existen variedad de las mismas como hemos podido ver en el punto 1.3.2
- Crowdfunding: Podemos presentar nuestro proyecto en plataformas de crowdfunding donde pequeños inversores pueden apoyar tu idea.
- Business Angels: Son individuos con experiencia empresarial y capital que invierten en proyectos en fases tempranas a cambio de una participación de las mismas.
- Aceleradoras o Incubadoras: Son programas que en determinados casos también pueden ofrecer financiación junto con apoyo, orientación y recursos.

2.4 Modelo de solución:

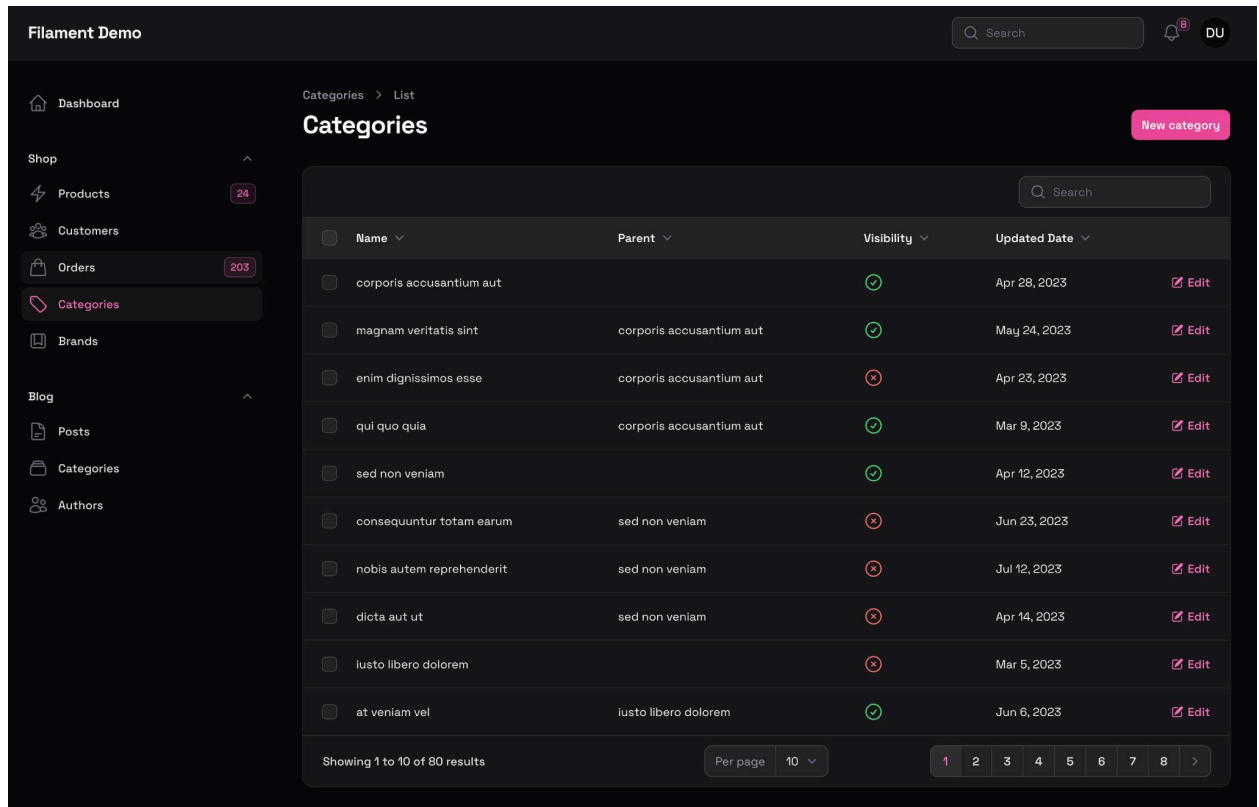
Mi proyecto de ERP de bajo o nulo coste tiene como finalidad disminuir el coste de todos aquellos pequeños talleres de reparación de dispositivos electrónicos y analógicos. Su existencia es debida a esta nula presencia de soluciones de *low cost* para dicho nicho de comercios, que actualmente usan métodos rudimentarios o poco cohesionados entre sí.

Este se presenta como gratuito con acceso a suscripción premium agregando funciones tales como: eliminación de publicidad, implementación del sistema VeriFact, sistema CMS para mejorar la comunicación con el cliente y copia en la nube).

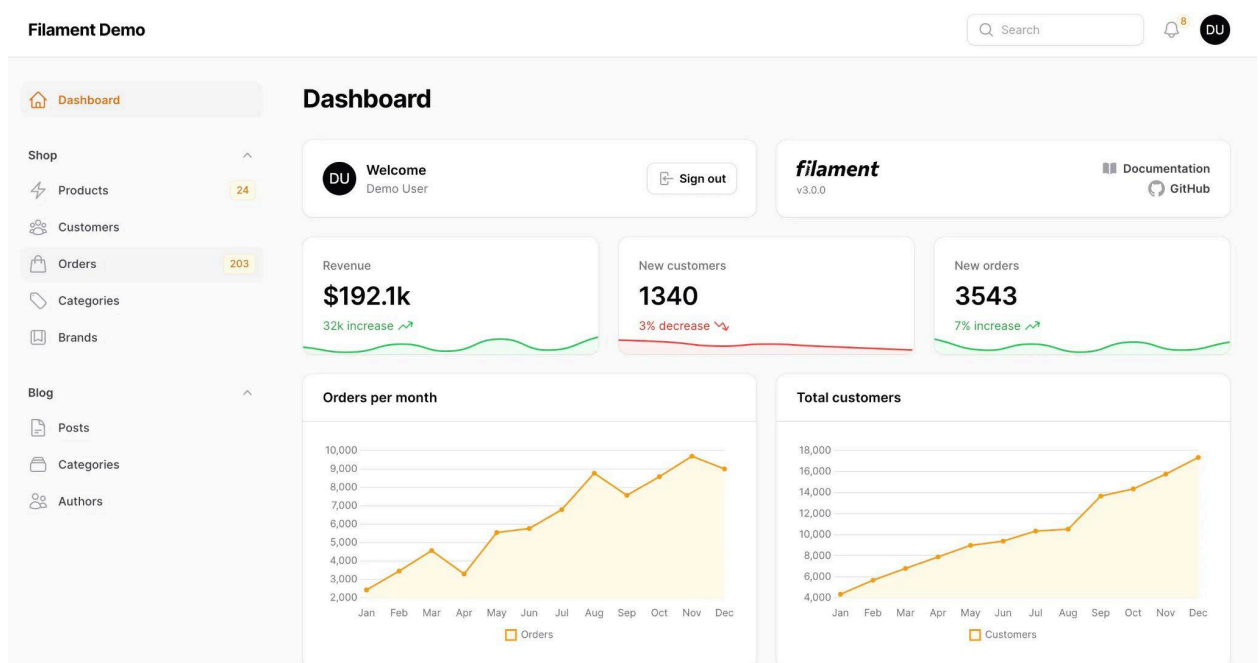
Desarrollado con un Stack Tecnológico compuesto principalmente por PHP gracias a la ayuda de Laravel/Filament y MySQL, incluyendo funcionalidades de facturación, gestión de clientes, dispositivos, partes de trabajo, piezas, usuarios, administración, auditoría e informes entre otros... Para la puesta en marcha del mismo requerimos de un mecanismo legal y económico del cual hemos podido hablar anteriormente en este documento, detallando y perfilando todo lo necesario para que este viera la luz.

Gráficos: Toda la UI de la aplicación se va a basar en resources de Filament, por lo que tendrás un aspecto similar al siguiente:

Modo Oscuro:











Modo claro:



Branding:

Colores corporativos:

Color	Hex	Posibles usos
	#083b5d	Fondo, headers, botones
	#0d5d8c	Elementos activos, íconos
	#2f85b6	Hover, detalles visuales
	#f8f5f1	Fondo general o secciones suaves

Isotipo	Logotipo	Imagotipo Vertical	Imagotipo Horizontal
			

Tipografías: Roboto.

- Identificación de los controles de calidad

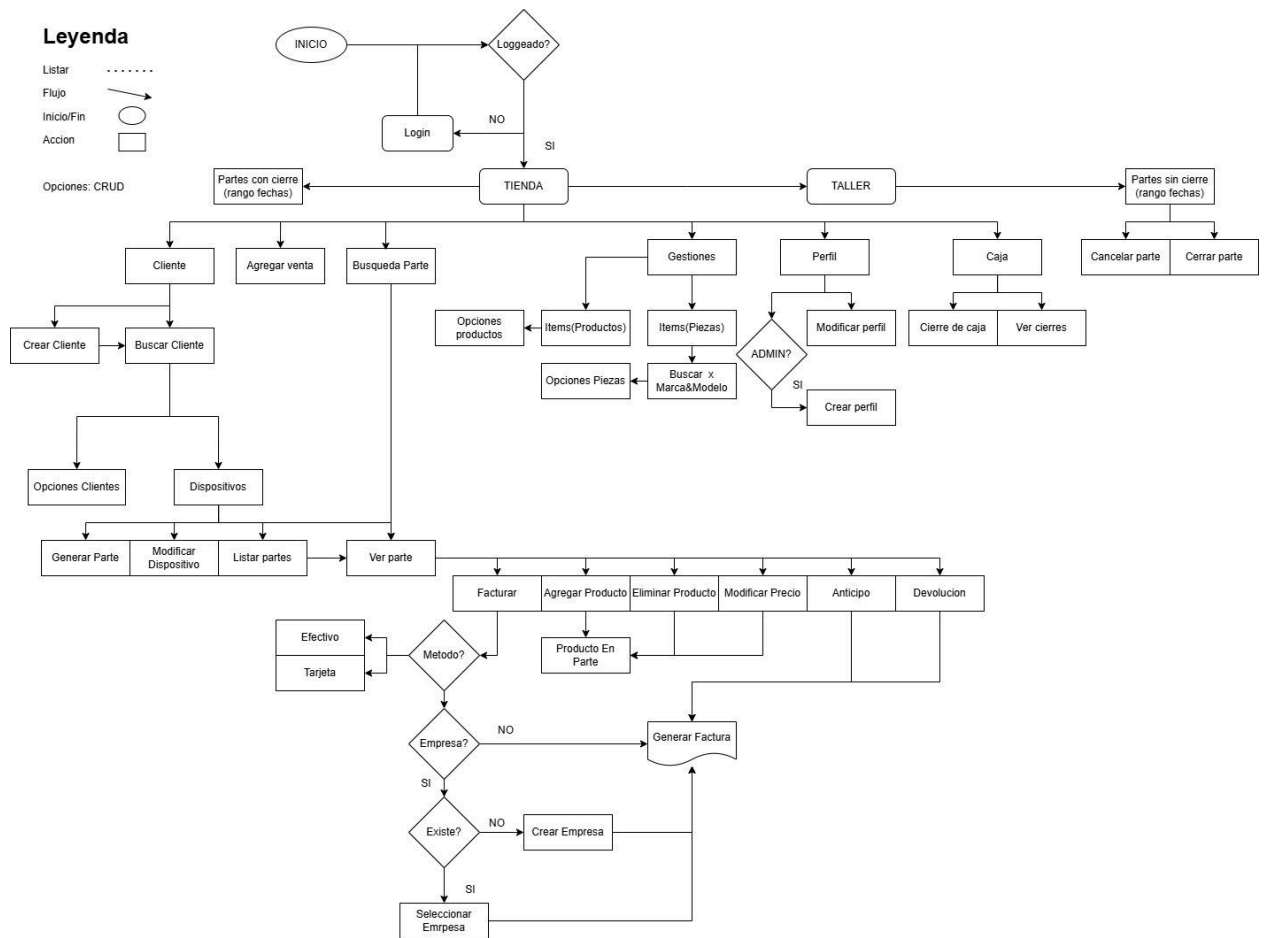
Revisiones de Código:

- **Propósito:** Identificar errores, malas prácticas de programación, problemas de rendimiento y posibles vulnerabilidades de seguridad en el código fuente.
- **Actividades:** Inspección del código por parte de otros desarrolladores (peer review) utilizando guías de estilo de codificación y herramientas de análisis estático de código.

Pruebas Unitarias:

- **Propósito:** Verificar la correcta funcionalidad de cada componente individual del software (funciones, clases, métodos) de forma aislada.
- **Actividades:** Desarrollo y ejecución de pruebas automatizadas que verifican diferentes escenarios y casos de borde para cada unidad de código.

✓ Diagramas de flujo de datos:



✓



3 Ejecución del proyecto

3.1 Planificación

Hagamos un repaso para ver los puntos claves que veremos durante el desarrollo del proyecto:

Análisis de requisitos

En primer lugar, se recopilaron las necesidades y funcionalidades que debe cubrir la aplicación. Durante esta fase se consultó a los usuarios y al tutor académico, registrando todos los requisitos relevantes para garantizar una visión integral del proyecto. Este paso resulta fundamental, ya que de él se deriva la planificación y el conjunto de tareas para el desarrollo posterior.

Diseño de la base de datos

Con la lista de requisitos definida, se procedió al diseño de la base de datos utilizando herramientas como MySQL Workbench, dbdiagram.io, draw.io y DBeaver, especialmente su función de generación automática de diagramas UML. Además, se aprovecharon las migraciones de Laravel para la construcción progresiva de la estructura de datos. En esta etapa se determinaron las entidades principales (como clientes, dispositivos y facturas) y sus relaciones. Es habitual que, a lo largo del desarrollo, la base de datos requiera ajustes para adaptarse a nuevas necesidades.

Desarrollo backend

Una vez definida la estructura de datos, se inició el desarrollo backend empleando Laravel 12 y Filament. Esta fase comprende la implementación de la lógica de negocio, el almacenamiento de datos, la gestión de usuarios y permisos, así como la generación de facturas y otras operaciones principales del sistema. Para el desarrollo se utilizó el editor de código Visual Studio Code y el servidor local XAMPP, además de Composer, Node.js y Git para la gestión de dependencias y el control de versiones.

Desarrollo frontend

De forma paralela o posterior, se llevó a cabo la implementación del frontend, utilizando principalmente Filament para el desarrollo de la interfaz de usuario. El objetivo fue ofrecer un panel de administración intuitivo y accesible, facilitando el acceso y la gestión de la información para los usuarios finales. Para esta tarea se empleó el navegador web y el editor de código correspondiente.

Pruebas y validación

Finalizada la implementación, se realizaron pruebas para verificar el correcto funcionamiento de la aplicación, tanto desde la perspectiva del usuario como desde la administración. Se utilizó un entorno de pruebas o staging, herramientas específicas de testing y datos de prueba generados mediante seeders de Laravel, así como conjuntos de datos extraídos de Kaggle o generados con inteligencia artificial. Los errores detectados se corrigieron antes del despliegue final.

Redacción de manuales y documentación

Por último, se elaboraron los manuales de usuario y la documentación técnica del proyecto. Se utilizaron procesadores de texto como Google Docs y herramientas de documentación para código, como PHPDocumentor o los propios decoradores en los comentarios del código fuente. El objetivo fue asegurar que cualquier persona encargada del uso o la administración futura del sistema dispusiera de toda la información necesaria.

Recursos empleados para cada fase:

- **Diseño de la base de datos:**
 - Herramientas de diagramas (MySQL Workbench, dbdiagram.io, draw.io, DBeaver).
 - Software de gestión de base de datos (MySQL Workbench).
- **Desarrollo backend:**
 - Editor de código (Visual Studio Code).
 - Frameworks y dependencias (Laravel 12, Filament, Composer, Node.js, Git).
 - Servidor local (XAMPP).
- **Desarrollo frontend:**
 - Editor de código.
 - Navegador web.
 - Filament.
- **Pruebas y validación:**

- Entorno de pruebas/staging.
- Herramientas de testing.
- Seeders de Laravel, datos de Kaggle o generados por IA.
- **Documentación y manuales:**
 - Procesador de textos (Google Docs).
 - Editor de código para documentación técnica (PHPDocumentor).

3.2 Revisión de recursos:

Aplicando lo anteriormente explicado, el desarrollo de la aplicación se vería de la siguiente manera desde comienzo a fin:

Dibujar la base de datos (UML)

Lo primero es pensar qué datos necesitas guardar y cómo se relacionan entre sí. Para eso, se suele hacer un esquema (UML) donde se ven las tablas principales, a modo de mapa de la app.

Comenzar proyecto Laravel e instalar Filament

Una vez finalizado el diseño, se crea el proyecto con Laravel. Luego instalas Filament, que es un panel que te ahorra mucho trabajo para el backend.

Crear *Migrations*

Con el comando *make:migration nombreTabla* genera los archivos de migración, se rellenan con los datos que requiera para montar cada tabla en la base de datos. Es mucho más rápido que hacerlo de forma manual. Una vez finalizadas y revisadas todas las migraciones con sus respectivas relaciones se realiza la *migration* con *php artisan make:migration - --seed*.

Agregar Seeders:

Durante la creación he ido agregando seeder para poder ir agregando datos y no tener que ir agregando manualmente cada vez que se modificara algo en la base de datos.

Desarrollar Models

Ahora toca crear los modelos de Laravel, que son las clases que representan cada tabla de la base de datos y te permiten manejar los datos de forma sencilla.

Generar los recursos con Filament

Para cada modelo, puedes generar un recurso en Filament (*php artisan make:filament-resource nombreResource*), crea un recurso de filament que permite generar funciones como editar, ver y borrar registros desde el panel de administración sin necesidad de generar tanto código. Se le pueden agregar funciones de búsqueda, filtro, etc junto con los RelationManagers (relaciones dentro de filament).

Preparar la documentación

Por último, se realiza el manual de instalación del proyecto, donde especificamos cómo buildar el proyecto, junto con la documentación del proyecto y cada uno de las funciones del mismo.

En cuanto a la planificación del proyecto ha sido la siguiente:

Planificación del TFG (15/04 – 10/06)

- **Semana 1 (15/04 – 21/04): Análisis de requisitos y diseño de base de datos**
Definir qué necesita la aplicación y hacer el esquema inicial de las tablas y relaciones en un UML.
- **Semana 2 (22/04 – 28/04): Puesta en marcha del proyecto**
Iniciar el proyecto en Laravel y Filament, instalar dependencias y configurar el entorno de desarrollo.
- **Semanas 3 y 4 (29/04 – 12/05): Estructura básica y primeras funcionalidades**
Crear migraciones y modelos, generar las tablas y comenzar a montar los recursos principales de la aplicación.
- **Semanas 5 y 6 (13/05 – 02/06): Desarrollo de funcionalidades avanzadas**
Añadir módulos secundarios, mejorar pantallas y relaciones, y dejar lista toda la parte funcional clave del sistema.
- **Semanas 8 y 9 (03/06 – 10/06): Documentación, mejoras, ajustes y cierre**
Realizar pruebas del proyecto, verificando el correcto funcionamiento del mismo. Redactar los manuales y la documentación técnica, hacer capturas y preparar todo para la entrega final.

//Te adjunto el proyecto, lo que llevo hasta el momento, puesto que no existen manuales, capturas, ni está desplegado todavía