**App Design Document – CleanFix Web Platform.**

## Índice – App Design Document: CleanFix

1. Visión General del Proyecto.
2. Funcionalidades Principales.
3. Arquitectura Técnica.
4. Integración de Inteligencia Artificial.
5. Métricas y Visualización.
6. Seguridad y Configuración.
7. Pruebas y Calidad.
8. Roadmap de Desarrollo.

### 1. Visión General del Proyecto.

* **Nombre de la aplicación:** CleanFix.
* **Tipo de solución:** Plataforma web empresarial para la gestión integral de servicios de mantenimiento, limpieza e incidencias.
* **Propósito del sistema:** CleanFix nace como respuesta a la necesidad de digitalizar y automatizar los procesos operativos de empresas que prestan servicios técnicos y de mantenimiento. Su objetivo principal es centralizar la gestión de solicitudes, incidencias, tareas, facturación y comunicación, en una única plataforma accesible, segura y escalable.
* **Motivación del proyecto:** En entornos donde la coordinación entre clientes, técnicos y administradores es crítica, los métodos tradicionales (correo electrónico, llamadas, hojas de cálculo) resultan ineficientes y propensos a errores. CleanFix busca eliminar estas fricciones mediante una solución moderna que permita:
* Registrar y rastrear solicitudes en tiempo real.
* Resolver incidencias con trazabilidad completa.
* Automatizar la generación de facturas.
* Facilitar la comunicación mediante un asistente inteligente.
* Visualizar métricas clave para la toma de decisiones.

**1.1 Objetivos estratégicos:**

* **Eficiencia operativa:** Reducir los tiempos de gestión y respuesta mediante automatización y flujos optimizados.
* **Transparencia:** Ofrecer a los clientes visibilidad sobre el estado de sus solicitudes y tareas.
* **Escalabilidad:** Diseñar una arquitectura modular que permita incorporar nuevas funcionalidades sin comprometer el rendimiento.
* **Innovación:** Integrar inteligencia artificial para mejorar la experiencia del usuario y agilizar procesos administrativos.
* **Seguridad y confiabilidad:** Garantizar la protección de datos sensibles y la estabilidad del sistema en entornos productivos.
* **Alcance funcional inicial:** La primera versión de CleanFix incluye los siguientes módulos:
  + Gestión de solicitudes de servicio.
  + Registro y seguimiento de incidencias.
  + Visualización de tareas completadas.
  + Generación de facturas en PDF.
  + Panel de administración con estadísticas.
  + Chat asistido por IA para soporte y automatización.

**1.2 Tecnologías empleadas:**

* **Frontend:** Angular + Tailwind CSS.
* **Backend:** .NET con Clean Architecture.
* **Base de datos:** Azure SQL.
* **IA:** Azure OpenAI.
* **Infraestructura:** Azure Cloud Services.

**Valor diferencial:** CleanFix no es solo una herramienta de gestión, sino una plataforma inteligente que combina diseño moderno, automatización avanzada y capacidades conversacionales para transformar la forma en que las empresas de mantenimiento operan y se relacionan con sus clientes.

### 2. Funcionalidades Principales.

La plataforma CleanFix está diseñada para cubrir de forma integral las necesidades operativas de empresas dedicadas al mantenimiento y limpieza. A continuación, se detallan las funcionalidades clave, organizadas por módulos, con sus respectivas capacidades técnicas y operativas:

#### 2.1 Gestión de Solicitudes.

Este módulo permite a los usuarios (clientes y administradores) crear, visualizar y gestionar solicitudes de servicio. Las funcionalidades incluyen:

* **Formulario de creación de solicitud:** Interfaz intuitiva que permite ingresar datos como tipo de servicio, ubicación, prioridad, fecha deseada y observaciones.
* **Visualización en tabla:** Las solicitudes se presentan en una tabla paginada con filtros por estado (pendiente, en proceso, completada), cliente, fecha y tipo de servicio.
* **Detalle de solicitud:** Acceso a información completa de cada solicitud, incluyendo historial de actualizaciones, comentarios, archivos adjuntos y técnico asignado.
* **Edición y seguimiento:** Los administradores pueden modificar solicitudes, asignar técnicos y actualizar el estado conforme avanza el proceso.

#### 2.2 Gestión de Incidencias.

Este módulo permite registrar, clasificar y resolver problemas detectados durante la ejecución de servicios. Está orientado principalmente a técnicos y administradores.

* **Registro de incidencias:** Formulario para documentar el tipo de problema, ubicación, prioridad, descripción y evidencia visual.
* **Clasificación automática:** Las incidencias se agrupan por tipo (eléctrica, estructural, limpieza, etc.) y prioridad (alta, media, baja).
* **Seguimiento y resolución:** Se asignan responsables, se actualiza el estado (abierta, en revisión, resuelta) y se documentan las acciones correctivas.
* **Asociación a solicitudes:** Cada incidencia puede vincularse a una solicitud específica para mantener la trazabilidad.

#### 2.3 Tareas Completadas.

Este módulo ofrece una vista consolidada de los trabajos finalizados, útil para administradores, técnicos y clientes.

* **Listado paginado:** Visualización eficiente de tareas completadas con filtros por técnico, cliente, fecha, tipo de servicio y ubicación.
* **Detalle de tarea:** Acceso a información como duración, materiales utilizados, comentarios del técnico, imágenes antes/después y firma digital.
* **Historial de ejecución:** Registro cronológico de todas las acciones realizadas en cada tarea, útil para auditorías internas y control de calidad.

#### 2.4 Facturación.

La facturación está automatizada para reducir la carga administrativa y asegurar precisión en los cálculos.

* **Generación de facturas en PDF:** A partir de las tareas completadas, el sistema genera facturas con desglose por servicio, IVA aplicado y totales.
* **Configuración fiscal:** Parametrización de tasas impositivas, formatos de factura y datos fiscales de la empresa.
* **Envío automático por correo:** Las facturas pueden enviarse directamente al cliente desde la plataforma, con confirmación de recepción.
* **Historial de facturación:** Registro de todas las facturas emitidas, con estado de pago, fecha de emisión y cliente asociado.

#### 2.5 Chat Asistido por IA.

Una de las funcionalidades más innovadoras de CleanFix es el asistente conversacional integrado, diseñado para mejorar la interacción y automatizar tareas.

* **Interfaz de chat integrada:** Componente visual que permite a los usuarios interactuar con el asistente desde cualquier sección de la plataforma.
* **Procesamiento de lenguaje natural:** El asistente comprende preguntas en lenguaje cotidiano y responde con precisión contextual.
* **Acciones automatizadas:** Puede generar facturas, consultar el estado de solicitudes, explicar procesos o sugerir soluciones.
* **Aprendizaje continuo:** El sistema se adapta a las consultas más frecuentes y mejora sus respuestas con el tiempo.

#### 2.6 Panel de Administración.

Este módulo centraliza el control operativo y estratégico de la plataforma, exclusivo para usuarios con rol de administrador.

* **Dashboard de métricas:** Visualización de KPIs como número de solicitudes activas, incidencias abiertas, tareas completadas y facturas emitidas.
* **Gestión de usuarios:** Creación, edición y eliminación de cuentas, asignación de roles y control de acceso.
* **Configuración del sistema:** Ajustes generales como horarios de atención, tipos de servicio, parámetros fiscales y estilos visuales.
* **Auditoría y trazabilidad:** Registro de acciones realizadas por cada usuario, útil para control interno y cumplimiento normativo.

### 3. Arquitectura Técnica.

La arquitectura de CleanFix ha sido diseñada bajo principios de escalabilidad, modularidad y mantenibilidad, utilizando tecnologías modernas tanto en el frontend como en el backend. A continuación, se detalla cada capa y componente clave del sistema:

#### 3.1 Frontend.

* **Tecnología principal:** Angular.
* **Estilos y diseño:** Tailwind CSS + Prettier.
* **Objetivos del frontend:**
* Ofrecer una experiencia de usuario fluida, moderna y responsiva.
* Facilitar la navegación entre módulos mediante componentes reutilizables.
* Integrar visualmente el asistente de IA y los paneles administrativos.

**3.1.1 Componentes destacados:**

* ServiceCard: Tarjetas informativas para mostrar servicios disponibles
* StatsSection: Visualización de métricas clave en formato gráfico
* ChatBox: Componente de interacción conversacional con el asistente IA
* LoginForm: Formulario de autenticación con validación de campos
* Snackbar: Sistema de notificaciones emergentes para feedback del usuario
* WhyUsSection: Sección de presentación institucional con animaciones

**3.1.2 Características adicionales:**

* Diseño responsivo para dispositivos móviles y escritorio
* Navegación basada en rutas protegidas según roles de usuario
* Animaciones suaves y transiciones entre vistas
* Integración con servicios RESTful del backend

#### 3.2 Backend.

* **Tecnología principal:** .NET (C#)
* **Arquitectura utilizada:** Clean Architecture
* **Objetivos del backend:**
* Separar responsabilidades por capas para facilitar el mantenimiento
* Garantizar la integridad de los datos y la lógica de negocio
* Permitir la escalabilidad horizontal y vertical del sistema

**3.2.1 Estructura por capas:**

* **Domain:** Contiene las entidades del negocio, reglas y validaciones
* **Application:** Define los casos de uso, comandos, consultas y servicios
* **Infrastructure:** Implementa la persistencia de datos, servicios externos y configuración
* **Presentation (API):** Expone los endpoints RESTful para consumo del frontend

**3.2.2 Patrones implementados:**

* **CQRS (Command Query Responsibility Segregation):** Separación entre operaciones de lectura y escritura para mejorar la claridad y el rendimiento
* **Mediator Pattern:** Uso de mediadores para desacoplar la lógica de negocio de los controladores

**3.2.3 Base de datos:**

* **Motor:** Azure SQL Database
* **ORM:** Entity Framework Core
* **Migraciones:** Automatizadas para mantener la estructura sincronizada
* **Claves primarias:** Uso de INT para entidades simples y GUID para entidades distribuidas o sensibles.

**3.2.4 Servicios adicionales:**

* Control de errores centralizado.
* Validación de modelos con FluentValidation.
* Configuración de CORS para permitir acceso desde dominios autorizados.

#### 3.3 Seguridad.

La seguridad es un pilar fundamental en CleanFix, abordada desde múltiples capas:

* **Autenticación:**
  + Formulario de login con validación de credenciales.
  + Control de visibilidad de contraseña.
  + Tokens de sesión seguros (JWT o similar).
* **Autorización:**
  + Roles definidos por perfil de usuario (Administrador, Técnico, Cliente).
  + Protección de rutas y endpoints según permisos.
* **Configuración segura:**
  + Uso de variables de entorno para credenciales y claves.
  + Exclusión de archivos sensibles mediante .gitignore.
  + Validación de entrada para prevenir inyecciones y ataques XSS.
* **Envío de correos:**
  + Integración con SMTP (Gmail) para notificaciones y envío de facturas.
  + Configuración de Webhooks para automatizar respuestas y alertas.

#### 3.4 Control de versiones y calidad de código.

* **Sistema de control:** Git
* **Validaciones automáticas:**
  + Hooks de pre-commit con Husky.
  + Linting y formateo con lint-staged y Prettier.
  + Ejecución de pruebas unitarias antes de cada commit.
* **Estructura del repositorio:**
  + Separación clara por módulos.
  + Documentación interna con comentarios y archivos README.
  + Exclusión de archivos temporales y de entorno.

### 4. Integración de Inteligencia Artificial.

La incorporación de inteligencia artificial en CleanFix representa uno de los pilares más innovadores del proyecto. Su propósito es mejorar la eficiencia operativa, reducir la carga administrativa y ofrecer una experiencia de usuario más intuitiva y proactiva. La IA se integra principalmente a través de un asistente conversacional que interactúa con los usuarios y automatiza tareas clave.

#### 4.1 Asistente Conversacional Inteligente.

* **Tecnología base:** Azure OpenAI.
* **Tipo de modelo:** Procesamiento de lenguaje natural (NLP).
* **Propósito funcional:** El asistente está diseñado para comprender preguntas en lenguaje natural, ofrecer respuestas contextualizadas y ejecutar acciones dentro de la plataforma. Su objetivo es actuar como un copiloto operativo para administradores, técnicos y clientes.

**4.1.2 Capacidades del asistente:**

* **Consulta de datos:** El usuario puede preguntar cosas como “¿Cuántas incidencias están abiertas esta semana?” o “¿Cuál es el estado de la solicitud #1023?” y recibir respuestas precisas.
* **Generación de facturas:** A partir de tareas completadas, el asistente puede generar una factura en PDF con desglose de servicios e IVA.
* **Soporte técnico:** Explica cómo usar funcionalidades de la plataforma, como crear una solicitud o registrar una incidencia.
* **Automatización de tareas:** Puede ejecutar comandos como “Crear una solicitud para limpieza en Madrid el lunes” o “Enviar la factura de Juan Pérez por correo”.

**4.1.3 Interfaz de interacción:**

* Componente visual ChatBox integrado en el frontend
* Diseño responsivo y accesible desde cualquier módulo
* Historial de conversación persistente por sesión

#### 4.2 Plugins y Servicios de IA.

Para enriquecer las capacidades del asistente, se han desarrollado componentes específicos que permiten acceder a datos internos, ejecutar acciones y mantener la coherencia conversacional.

* **DBPlugin:** Este plugin permite al asistente consultar directamente la base de datos del sistema, accediendo a entidades como solicitudes, incidencias, tareas y facturas. Utiliza filtros y estructuras DTO para devolver respuestas precisas y seguras.
* **AssistantService:** Servicio que orquesta las peticiones del asistente, interpreta la intención del usuario y decide qué acción ejecutar. También gestiona la generación de contenido dinámico, como mensajes personalizados o documentos.
* **ChatBoxIAController:** Controlador backend que recibe las entradas del usuario, las envía al modelo de IA, interpreta la respuesta y la traduce en acciones concretas dentro de la plataforma. También gestiona el envío de correos automáticos y la validación de comandos.

#### 4.3 Seguridad y Control.

La integración de IA se ha diseñado con especial atención a la seguridad y la privacidad:

* **Filtrado de contenido:** El asistente no responde a preguntas fuera del contexto del sistema ni permite acciones no autorizadas.
* **Control de acceso:** Las acciones disponibles varían según el rol del usuario (administrador, técnico, cliente).
* **Auditoría de interacciones:** Todas las conversaciones se registran para análisis posterior y mejora continua.
* **Protección de datos:** Las respuestas del asistente nunca exponen información sensible sin autorización.

#### 4.4 Beneficios Estratégicos.

La IA en CleanFix no es un complemento, sino un componente central que transforma la experiencia de uso:

* **Reducción de carga operativa:** Automatiza tareas repetitivas como generación de facturas, búsqueda de datos y envío de correos.
* **Mejora de la experiencia del usuario:** Ofrece asistencia inmediata, contextual y personalizada.
* **Escalabilidad funcional:** Permite incorporar nuevas capacidades sin rediseñar la interfaz.
* **Innovación competitiva:** Diferencia a CleanFix de otras soluciones tradicionales del mercado.

### 5. Métricas y Visualización.

La plataforma CleanFix incorpora un sistema robusto de visualización de métricas operativas y estructuración de datos, diseñado para ofrecer a los administradores una visión clara y en tiempo real del rendimiento del sistema. Esta funcionalidad es clave para la toma de decisiones estratégicas, el seguimiento de procesos y la mejora continua.

#### 5.1 Panel de Métricas.

El módulo de estadísticas está integrado en el panel de administración y presenta indicadores clave de rendimiento (KPIs) mediante componentes visuales interactivos.

**Indicadores principales:**

* Número total de solicitudes activas, completadas y canceladas.
* Incidencias abiertas, en revisión y resueltas.
* Tareas completadas por técnico, cliente o tipo de servicio.
* Facturas emitidas, pendientes de pago y abonadas.
* Tiempo promedio de resolución por tipo de incidencia.

**Visualización:**

* Gráficos de barras, líneas y pastel para representar evolución temporal.
* Tarjetas resumen con cifras destacadas.
* Sección comparativa entre periodos (semanal, mensual, anual).

**Componente destacado:** AppStatsSection – Diseñado en Angular, este componente permite mostrar métricas en tiempo real con animaciones suaves y diseño responsivo. Se adapta automáticamente al tipo de dato y al dispositivo del usuario.

#### 5.2 Estructura de Datos y DTOs.

Para garantizar una visualización eficiente y segura, CleanFix utiliza estructuras de datos específicas conocidas como DTOs (Data Transfer Objects). Estos objetos permiten encapsular la información relevante para cada vista, evitando la exposición innecesaria de datos sensibles.

**DTOs principales:**

* PaginatedDataDto: Estructura base para paginación de listas (solicitudes, tareas, incidencias).
* SolicitationBriefDto: Contiene información resumida de cada solicitud (ID, cliente, estado, fecha).
* IncidenceDto: Detalles de cada incidencia (tipo, prioridad, estado, técnico asignado).
* InvoiceDto: Información fiscal de cada factura (importe, IVA, fecha, estado de pago).

**Ventajas del uso de DTOs:**

* Reducción del tamaño de las respuestas en el frontend.
* Mejora del rendimiento en consultas masivas.
* Mayor seguridad al limitar la exposición de campos internos.
* Flexibilidad para adaptar la visualización según el rol del usuario.

#### 5.3 Filtros y Paginación.

La plataforma permite aplicar filtros avanzados en todas las vistas de datos, lo que facilita la búsqueda y el análisis.

**Filtros disponibles:**

* Por fecha, cliente, técnico, estado, tipo de servicio o incidencia.
* Combinación de múltiples criterios.
* Ordenamiento ascendente/descendente por cualquier campo.

**Paginación:**

* Implementada tanto en frontend como en backend.
* Control de número de elementos por página.
* Navegación entre páginas con indicadores visuales.

#### 5.4 Exportación y Análisis.

Aunque la plataforma no exporta archivos directamente, los datos visualizados pueden ser utilizados para análisis externo mediante integración con herramientas de BI o informes personalizados.

**Posibilidades futuras:**

* Integración con Power BI o Tableau.
* Generación de informes PDF desde el panel de métricas.
* Alertas automáticas basadas en umbrales definidos por el administrador.

### 6. Seguridad y Configuración.

La seguridad y la correcta configuración del entorno son pilares fundamentales en el diseño de CleanFix. Desde el control de acceso hasta la protección de datos sensibles, cada aspecto ha sido cuidadosamente estructurado para garantizar la integridad del sistema, la privacidad de los usuarios y la estabilidad operativa en entornos reales.

#### 6.1 Autenticación y Autorización.

**Autenticación:**

* Implementación de un formulario de login con validación de campos (email, contraseña).
* Control de visibilidad de la contraseña mediante íconos interactivos.
* Gestión de sesiones seguras mediante tokens (JWT o similar), con expiración configurable.
* Prevención de ataques de fuerza bruta mediante limitación de intentos fallidos.

**Autorización:**

* Sistema de roles claramente definidos: Administrador, Técnico, Cliente.
* Protección de rutas en el frontend según permisos asignados.
* Validación de acceso a endpoints en el backend mediante filtros y atributos.
* Restricción de acciones sensibles (como generación de facturas o edición de usuarios) a perfiles autorizados.

#### 6.2 Configuración del Entorno.

**Variables de entorno:**

* Separación entre entornos de desarrollo, pruebas y producción.
* Uso de archivos .env para almacenar claves, credenciales y configuraciones sensibles.
* Exclusión de archivos de configuración mediante .gitignore para evitar exposición en repositorios públicos.

**Configuración SMTP:**

* Integración con servidores de correo (como Gmail) para el envío automático de notificaciones, facturas y alertas.
* Validación de credenciales y configuración de puertos seguros (TLS/SSL).
* Gestión de errores en el envío y reintentos automáticos en caso de fallo.

**Webhooks:**

* Implementación de webhooks para automatizar acciones como confirmaciones de pago, actualizaciones de estado o alertas de sistema.
* Validación de origen y autenticación de peticiones entrantes.
* Registro de eventos para auditoría y trazabilidad.

#### 6.3 Protección de Datos.

**Validación de entrada:**

* Sanitización de datos en formularios para prevenir inyecciones SQL y ataques XSS.
* Uso de librerías como FluentValidation para asegurar la integridad de los modelos.

**Cifrado y almacenamiento:**

* Cifrado de contraseñas mediante algoritmos seguros (bcrypt, SHA-256).
* Almacenamiento de datos sensibles en campos protegidos y acceso restringido.
* Registro de logs de acceso y acciones críticas para auditoría interna.

**Política de backups:**

* Programación de copias de seguridad automáticas de la base de datos.
* Almacenamiento en servidores seguros con redundancia geográfica.
* Procedimientos de restauración documentados y probados.

#### 6.4 Control de Versiones y Calidad.

**Sistema de control de versiones:**

* Uso de Git como sistema principal, con ramas organizadas por entorno y funcionalidad.
* Convenciones de nombres para commits y ramas (feature/, fix/, hotfix/, release/).

**Validaciones automáticas:**

* Hooks de pre-commit con Husky para ejecutar pruebas, linting y validación de estructura.
* Uso de lint-staged para aplicar reglas de estilo y formato antes de cada commit.
* Integración con herramientas de CI/CD para despliegue seguro y automatizado.

#### 6.5 Auditoría y Trazabilidad.

* Registro de acciones realizadas por cada usuario (creación, edición, eliminación).
* Historial de cambios en solicitudes, incidencias y facturas.
* Panel de administración con acceso a logs y métricas de seguridad.
* Alertas automáticas ante comportamientos anómalos o intentos de acceso no autorizado.

### 7. Pruebas y Calidad.

La calidad del software es un componente esencial en CleanFix, abordada desde el inicio del desarrollo mediante prácticas automatizadas, pruebas unitarias y validaciones estructurales. El objetivo es garantizar que cada funcionalidad se comporte de forma predecible, segura y eficiente, tanto en entornos de desarrollo como en producción.

#### 7.1 Estrategia de Pruebas.

**Pruebas unitarias:**

* Se aplican a componentes visuales del frontend (Angular) y a servicios del backend (.NET).
* Validan el comportamiento esperado de funciones individuales, como controladores, servicios, validadores y DTOs.
* Utilizan frameworks como Jasmine/Karma (frontend) y xUnit/NUnit (backend).

**Pruebas de integración (en evolución):**

* Verifican la interacción entre módulos, como la conexión entre el asistente IA y el sistema de facturación.
* Simulan escenarios reales para asegurar que los flujos de datos funcionen correctamente.

**Pruebas manuales:**

* Se realizan en etapas clave del desarrollo para validar la experiencia de usuario, la navegación y el diseño visual.
* Incluyen pruebas de usabilidad, accesibilidad y responsividad en distintos dispositivos.

#### 7.2 Validaciones Automáticas.

**Hooks de pre-commit:**

* Implementados con **Husky**, permiten ejecutar tareas antes de confirmar cambios en el repositorio.
* Validan que el código cumpla con las reglas de estilo, pase las pruebas unitarias y no introduzca errores estructurales.

**Linting y formateo:**

* Uso de **lint-staged** y **Prettier** para asegurar consistencia en la escritura del código.
* Aplicación automática de reglas de estilo en archivos modificados.

**Control de estructura:**

* Validación de convenciones de nombres, organización de carpetas y uso correcto de componentes.
* Revisión automática de dependencias y exclusión de archivos no deseados.

#### 7.3 Datos Ficticios para Desarrollo.

**Generación de datos simulados:**

* Se crean registros ficticios de solicitudes, incidencias, tareas y facturas para probar funcionalidades sin comprometer datos reales.
* Permiten validar paginación, filtros, visualización de métricas y rendimiento del sistema.

**Ventajas:**

* Acelera el desarrollo de interfaces y pruebas visuales.
* Facilita la detección de errores en etapas tempranas.
* Permite simular escenarios complejos sin necesidad de intervención manual.

#### 7.4 Control de Calidad en el Repositorio.

**Estructura del repositorio:**

* Organización modular por capas (frontend/backend), con carpetas específicas para componentes, servicios, modelos y pruebas.
* Inclusión de archivos README explicativos en cada módulo.

**Exclusión de archivos sensibles:**

* Uso de .gitignore para evitar la inclusión de archivos de configuración, credenciales, logs y datos temporales.

**Convenciones de commits:**

* Uso de mensajes estructurados para facilitar el seguimiento de cambios.
* Integración con herramientas de CI/CD para despliegue automatizado y validación continua.

#### 7.5 Mejora Continua.

* Revisión periódica del código por parte del equipo técnico.
* Refactorización de componentes para mejorar rendimiento y legibilidad.
* Documentación de buenas prácticas y patrones recomendados.
* Retroalimentación constante entre desarrolladores, testers y usuarios finales.

### 9. Roadmap de Desarrollo.

El desarrollo de CleanFix se ha estructurado en fases progresivas que permiten construir la plataforma de forma ordenada, escalable y con entregables claros en cada etapa. Este enfoque facilita la colaboración entre equipos, la validación temprana de funcionalidades y la incorporación de mejoras continuas.

#### Fase 1 – Configuración Inicial del Proyecto.

**Objetivo:** Establecer las bases técnicas y organizativas del proyecto.

**Actividades realizadas:**

* Creación del repositorio con estructura modular.
* Configuración de herramientas de control de versiones (Git).
* Definición de convenciones de código y estilo.
* Implementación de .gitignore para exclusión de archivos sensibles.
* Configuración de Husky y lint-staged para validaciones automáticas.

**Resultado:** Un entorno de desarrollo limpio, controlado y preparado para escalar.

#### Fase 2 – Desarrollo del Backend.

**Objetivo:** Construir la lógica de negocio y los servicios fundamentales de la plataforma.

**Actividades realizadas:**

* Implementación de Clean Architecture con capas separadas (Domain, Application, Infrastructure).
* Desarrollo de entidades, servicios, comandos y consultas usando CQRS y Mediator.
* Configuración de la base de datos en Azure SQL con migraciones automatizadas.
* Creación de endpoints RESTful para solicitudes, incidencias, tareas y facturas.

**Resultado:** Un backend robusto, modular y preparado para integrarse con el frontend.

#### Fase 3 – Implementación del Frontend.

**Objetivo:** Diseñar e implementar la interfaz de usuario con componentes reutilizables y diseño responsivo.

**Actividades realizadas:**

* Desarrollo en Angular con Tailwind CSS para estilos personalizados.
* Creación de componentes visuales (ServiceCard, StatsSection, ChatBox, etc.).
* Integración con los servicios del backend mediante HTTPClient.
* Validación de formularios, navegación protegida y diseño adaptativo.

**Resultado:** Una interfaz moderna, intuitiva y funcional para todos los perfiles de usuario.

#### Fase 4 – Integración de Inteligencia Artificial.

**Objetivo:** Incorporar un asistente conversacional que automatice tareas y mejore la experiencia del usuario.

**Actividades realizadas:**

* Conexión con Azure OpenAI para procesamiento de lenguaje natural.
* Desarrollo de plugins personalizados (DBPlugin, AssistantService).
* Implementación del ChatBoxIAController para gestionar interacciones.
* Pruebas de generación de facturas, respuestas automatizadas y comandos conversacionales.

**Resultado:** Un asistente inteligente capaz de interactuar con los usuarios y ejecutar acciones dentro del sistema.

#### Fase 5 – Optimización Visual y Experiencia de Usuario.

**Objetivo:** Refinar la interfaz, mejorar el rendimiento y asegurar una experiencia fluida.

**Actividades realizadas:**

* Implementación de paginación y filtros avanzados.
* Ajuste de animaciones, transiciones y estilos visuales.
* Revisión de responsividad en distintos dispositivos.
* Mejora de la navegación y accesibilidad.

**Resultado:** Una experiencia de usuario pulida, eficiente y visualmente atractiva.

#### Fase 6 – Pruebas Finales, Documentación y Despliegue.

**Objetivo:** Validar el sistema, documentar su funcionamiento y preparar el entorno de producción.

**Actividades en curso:**

* Ejecución de pruebas unitarias y de integración.
* Generación de datos ficticios para simulaciones.
* Redacción de documentación técnica y funcional.
* Configuración del entorno de producción en Azure.
* Preparación del despliegue y monitoreo post-lanzamiento.

**Estado actual:** Fase activa, con tareas en progreso y revisión final antes del lanzamiento oficial.