



FacturaApp

PROYECTO DE DAW 2024/2025

ALEXEY SUZDALENKO

Contenido

Indice:

1. Resumen ... 4
2. Palabras clave ... 4
3. Introducción ... 5
4. Objetivos
 - a. Objetivo general ... 5
 - b. Objetivos específicos ... 6
5. Análisis del contexto
 - a. Análisis del contexto ... 7
 - b. Estado del arte ... 8
 - c. Competencia ... 9
 - d. Innovación ... 13
 - e. Análisis DAFO ... 14
6. Análisis de requisitos
 - a. Requisitos funcionales ... 15
 - b. Requisitos no funcionales ... 16
 - c. Requisitos de seguridad y protección de datos ... 18
 - d. Requisitos de accesibilidad y usabilidad ... 19
7. Diseño
 - a. Diagrama de interfaces ... 20
 - b. Diagrama físico y/o lógico de red ... 25
 - d. Diagrama entidad/relación (E/R) ... 26
 - e. Diagrama de Bases de Datos ... 29
 - f. Casos de uso ... 36
 - g. Mapa de navegación y estructura de menús ... 38
8. Planificación
 - a. Diagrama de Gantt (actividades y tiempos) ... 39
 - b. Recursos y logística ... 40
 - c. Cronología del desarrollo (timeline) ... 43
9. Implementación
 - a. Lenguajes, tecnologías y herramientas utilizadas ... 44
 - b. Organización del proyecto (frontend / backend) ... 45
 - c. Código fuente representativo ... 49
 - d. Integración con servicios externos ... 62
 - e. Control de versiones y GitHub ... 63
 - f. Mejoras y funcionalidades destacadas ... 63
10. Puesta en marcha, explotación
 - a. Configuración del entorno de producción ... 63
 - b. Despliegue en servidores / hosting ... 65
 - c. Accesos públicos y pruebas en producción ... 66
11. Prueba y control de calidad
 - a. Tipos de pruebas realizadas (unitarias, funcionales...) ... 66
 - b. Resultados y validación de funcionalidades ... 68
12. Plan de empresa

- a. Objetivos generales ... 69
- b. Recursos humanos y estructura de la empresa ... 70
- c. Inversiones, compras y gastos ... 72
- d. Ingresos ... 73
- 13. Gestión económica de un proyecto
 - a. Recursos materiales ... 75
 - b. Costes y gastos desglosados ... 76
 - c. Horas de trabajo y valoración económica ... 76
 - d. Umbral de rentabilidad ... 76
 - e. Estimación del coste total de desarrollo ... 77
- 14. Conclusiones y valoración personal
 - a. Valoración personal del proceso ... 77
 - b. Dificultades encontradas y cómo se resolvieron ... 78
 - c. Conclusión general del proyecto ... 78
- 15. Bibliografía ... 79
- 16. Anexos ... 79
- 17. Futuras mejoras
 - a. Funciones previstas(JWT token,IA, OCR, pagos, app móvil)
 - b. Escalabilidad o nuevas versiones

1. Resumen

FacturaApp es un proyecto innovador que nace con la vocación de convertirse en una solución digital accesible y eficaz para autónomos y pequeñas empresas que necesitan gestionar su facturación de forma profesional y sencilla. La aplicación está diseñada para permitir la creación, gestión y envío de facturas legalmente válidas, así como el control de clientes, artículos y datos empresariales, todo ello desde una interfaz clara y amigable.

El valor diferencial de FacturaApp reside en su enfoque centrado en la facilidad de uso, la seguridad de los datos y la adaptabilidad a las necesidades reales de los usuarios que no cuentan con conocimientos técnicos avanzados. A través de una plataforma web funcional y responsive, los usuarios podrán generar facturas en formato PDF con solo unos clics, almacenarlas, enviarlas y mantener el control de sus operaciones comerciales sin complicaciones.

El sistema estará compuesto por un frontend construido con tecnologías web estándar (HTML, CSS, JS) y un backend robusto basado en Django (Python), que garantiza estabilidad, escalabilidad y seguridad. La información se gestiona mediante bases de datos estructuradas y se proyecta una futura integración con sistemas inteligentes de escaneado de documentos y lectura automática mediante tecnologías OCR e IA.

La propuesta de valor se completa con un diseño intuitivo, una navegación fluida y funcionalidades clave como la autenticación segura, el historial de facturas, la personalización del perfil de empresa y la exportación de datos. Además, se contempla su evolución hacia una herramienta multiplataforma y multiempresa, escalable y adaptada a las nuevas normativas fiscales.

Desde el punto de vista de marketing, FacturaApp apostará por una estrategia digital basada en la presencia online, colaboraciones con comunidades de emprendedores y autónomos, y posicionamiento SEO, además de ofrecer una versión gratuita como puerta de entrada para nuevos usuarios. En fases futuras, se incorporarán planes de suscripción y módulos adicionales para adaptar el producto a diferentes segmentos profesionales.

FacturaApp aspira a consolidarse como la herramienta de facturación de referencia para los profesionales que valoran la sencillez, la eficiencia y la calidad, permitiéndoles centrarse en su negocio mientras la tecnología se ocupa del resto.

2. Palabras clave

- # Facturas
- # Aplicación
- # Empresas
- # Autónomos
- # Contabilidad
- # Gestión
- # Artículos
- # Clientes
- # PDF

- # Email
- # Digitalización
- # Documentos
- # Base de datos
- # Seguridad
- # Django
- # Interfaz web

3. Introducción

En un contexto donde la digitalización de procesos se ha vuelto imprescindible, muchos autónomos y pequeñas empresas siguen enfrentándose a dificultades para gestionar su facturación de forma eficiente, profesional y sin complicaciones técnicas. La mayoría de las soluciones disponibles en el mercado son de pago, complejas o requieren conocimientos avanzados de software, lo que genera una barrera importante para quienes buscan una herramienta sencilla y accesible para el día a día de su actividad económica.

Emitir facturas legales, organizarlas por cliente, controlar artículos o servicios prestados y cumplir con las exigencias normativas son tareas esenciales, pero también repetitivas y propensas a errores cuando se hacen manualmente o con herramientas genéricas como hojas de cálculo. En este entorno de alta exigencia administrativa, **FacturaApp** nace como una respuesta directa a las necesidades reales de los profesionales que no cuentan con un departamento de gestión o conocimientos técnicos avanzados.

FacturaApp es más que una aplicación web para emitir facturas: es una solución gratuita, ligera y funcional que permite registrar clientes, artículos, empresas y generar facturas en PDF de forma rápida y segura. Su diseño intuitivo, su despliegue online y su estructura modular hacen posible que cualquier usuario pueda gestionar su facturación sin depender de terceros ni de software costoso.

La motivación detrás del desarrollo de **FacturaApp** radica en la democratización del acceso a herramientas digitales útiles, en particular para emprendedores, freelancers y pequeños negocios que buscan profesionalizar su actividad. En un entorno económico que exige eficiencia y cumplimiento legal, FacturaApp aporta control, claridad y autonomía, sin renunciar a la simplicidad.

A través de esta plataforma se busca no solo resolver un problema técnico, sino también empoderar al usuario con una herramienta fiable, segura y cercana, que acompañe su crecimiento profesional y se adapte a sus necesidades reales. En definitiva, **FacturaApp** es una solución digital nacida para hacer que la gestión de la facturación deje de ser un obstáculo, y se convierta en una ventaja.

4. Objetivos

a. Objetivo general

El objetivo general de **FacturaApp** es proporcionar a autónomos y pequeñas empresas una herramienta digital gratuita, sencilla y funcional que les permita gestionar de forma eficaz su facturación,

ofreciendo funcionalidades clave como la creación de facturas legales en PDF, el registro de clientes y artículos, y la organización de datos empresariales, todo ello a través de una interfaz web accesible y segura.

La aplicación aspira a consolidarse como una solución de referencia para quienes buscan autonomía administrativa sin necesidad de conocimientos técnicos ni costes añadidos, destacándose por su usabilidad, fiabilidad y capacidad de adaptación a las necesidades reales del usuario profesional. Además, **FacturaApp** se compromete a mantener una experiencia intuitiva y libre de complicaciones, siendo una herramienta cercana y eficiente para el día a día de quienes emprenden.

b. Objetivos específicos

OBJ_1: Desarrollar una Plataforma de Facturación Completa

Diseñar y construir una aplicación web que permita gestionar clientes, artículos y empresas, así como generar y almacenar facturas legales en PDF de manera estructurada y eficiente.

OBJ_2: Ofrecer una Herramienta Gratuita y Accesible

Garantizar que FacturaApp sea de uso gratuito, sin necesidad de instalación, y disponible para cualquier profesional, especialmente autónomos y pequeñas empresas con recursos limitados.

OBJ_3: Garantizar una Experiencia de Usuario Intuitiva

Diseñar una interfaz limpia, responsive y fácil de usar, que permita a los usuarios gestionar sus facturas sin conocimientos técnicos ni necesidad de formación especializada.

OBJ_4: Asegurar la Seguridad y Privacidad de los Datos

Implementar buenas prácticas de desarrollo seguro para proteger los datos empresariales y personales, mediante autenticación, cifrado y control de acceso.

OBJ_5: Establecer una Arquitectura Modular Escalable

Separar frontend y backend en una arquitectura mantenible que permita futuras mejoras, integración con APIs, sistemas externos u otros módulos (como IA o OCR).

OBJ_6: Facilitar el Acceso y la Disponibilidad Online

Publicar la aplicación en servidores accesibles (Firebase, PythonAnywhere), asegurando su funcionamiento desde cualquier dispositivo con conexión a internet.

OBJ_7: Integrar Funcionalidades Avanzadas a Futuro

Prever la inclusión de herramientas como la digitalización de documentos, lectura automática de datos con inteligencia artificial y funciones multiempresa.

OBJ_8: Potenciar la Profesionalización del Usuario

Ayudar al usuario a presentar facturas formales, ordenadas y legales, fortaleciendo su imagen profesional ante clientes, organismos públicos y entidades financieras.

5. Análisis del contexto

a. Análisis del contexto

Describe el entorno en el que operará FacturaApp, considerando factores como el uso de herramientas digitales por autónomos y pymes, tendencias tecnológicas, digitalización administrativa y el marco normativo.

- Analiza cómo la situación económica actual y la transformación digital pueden influir en el uso de soluciones gratuitas de facturación.
- Examina la normativa aplicable en materia fiscal, de facturación electrónica, protección de datos y obligaciones del profesional autónomo.

Mercado:

El mercado de herramientas de facturación digital está en crecimiento, impulsado por el auge del emprendimiento, la simplificación de trámites fiscales y el aumento del trabajo autónomo. En España, existen más de 3 millones de trabajadores por cuenta propia y pymes, muchos de los cuales carecen de herramientas específicas para emitir facturas legales de forma profesional, organizada y conforme a la normativa.

Tendencias:

- Digitalización de procesos administrativos: La automatización de tareas repetitivas como la emisión de facturas es una necesidad creciente en el entorno actual, donde el ahorro de tiempo y la reducción de errores son fundamentales.
- Demanda de soluciones accesibles y gratuitas: Muchos profesionales no están dispuestos a pagar licencias por software de facturación, y prefieren alternativas gratuitas y fáciles de usar.
- Crecimiento del uso de tecnologías web: La adopción de aplicaciones en la nube, accesibles desde cualquier dispositivo, está desplazando el uso de herramientas locales como Excel o plantillas manuales.
- Enfoque en la experiencia de usuario: Los usuarios valoran la simplicidad, la navegación intuitiva y la rapidez de uso por encima de la complejidad o la personalización avanzada.
- Conciencia sobre la ciberseguridad: Cada vez hay más atención a la protección de los datos fiscales y personales, lo que obliga a los desarrolladores a integrar medidas de seguridad incluso en soluciones gratuitas.

Oportunidades:

- Ofrecer una solución gratuita y profesional: Muchos usuarios buscan una aplicación que les permita emitir facturas con un aspecto formal y legal sin necesidad de pagar.
- Simplificar la vida del autónomo: Automatizar tareas como la numeración, el cálculo de totales, el almacenamiento de PDF o la gestión de clientes ahorra tiempo y reduce errores.
- Accesibilidad desde cualquier lugar: Una app basada en web permite emitir una factura desde el móvil o el portátil sin depender de instalaciones.

- Integrar futuras mejoras: OCR, firma digital, envío automático por correo, o facturación recurrente son características valoradas para crecer junto con el usuario.

Desafíos:

- Alta competencia: Existen herramientas consolidadas en el mercado, algunas gratuitas y otras de pago, lo que obliga a ofrecer algo diferencial (simplicidad, diseño, privacidad, etc.).
- Usuarios con bajo perfil técnico: No todos los usuarios conocen conceptos como “hosting”, “backend” o “API”, por lo que la interfaz debe ser autoexplicativa y clara.
- Cambio normativo constante: Las obligaciones fiscales evolucionan (por ejemplo, el impulso de la factura electrónica obligatoria), lo que implica actualizar la app de forma periódica.
- Resistencia al cambio: Algunos usuarios siguen prefiriendo métodos tradicionales como Excel o plantillas Word, por lo que la app debe convencer con su facilidad y valor añadido

b. Estado del arte

Crecimiento:

- Mercado global del software de facturación: más de 10.000 millones de dólares en 2023.
- Proyección para 2030: superará los 20.000 millones, impulsado por la digitalización de pymes y autónomos.
- En España, la implantación obligatoria de la factura electrónica para autónomos en los próximos años acelerará la adopción de herramientas digitales.

Tendencias:

- Digitalización administrativa: Automatización de procesos como la creación y envío de facturas, cálculo de impuestos o gestión de clientes.
- Facturación electrónica obligatoria: Normativa nacional y europea avanza hacia la factura electrónica como estándar para empresas y autónomos.
- SaaS y soluciones cloud: Uso creciente de aplicaciones basadas en la nube por su facilidad de acceso, bajo coste y mantenimiento cero.
- Democratización del software: Aumento de herramientas gratuitas o freemium para profesionales independientes y micropymes.
- Valoración de la privacidad y seguridad: Los usuarios esperan que incluso las apps gratuitas cumplan estándares de protección de datos.

Innovaciones:

- Generación de facturas PDF automatizadas con numeración, totales, impuestos y datos fiscales.
- Integración con IA (en versiones avanzadas) para escanear tickets o extraer datos de otros documentos (OCR).

- Sistemas multiempresa y multiusuario para escalabilidad.
- Paneles de control con estadísticas de facturación y rendimiento.
- Despliegue en plataformas web sin instalación y accesibles desde cualquier dispositivo.

Competencia:

- Herramientas gratuitas como FacturaScripts, Debitoor (versión gratuita limitada), Billin o Invoice Ninja.
- Software de pago como Holded, Quipu, Contasimple o Sage.
- Diferencias clave: curva de aprendizaje, coste, complejidad técnica, soporte y personalización.

Diferenciación:

- FacturaApp apuesta por una solución:
 - Totalmente gratuita
 - De uso inmediato y sencillo
 - Sin necesidad de instalación
 - Con un diseño limpio y centrado en el usuario no técnico.
- Adaptada al autónomo real, no a contables ni técnicos.
- Potencial de expansión con funciones como lectura automática de documentos o gestión multiempresa.

En resumen:

- El mercado del software de facturación está en expansión y presenta una gran oportunidad, especialmente ante los nuevos requisitos legales.
- FacturaApp destaca por su accesibilidad, gratuidad, sencillez, despliegue en la nube y enfoque en el usuario autónomo que busca una herramienta funcional, sin barreras técnicas ni económicas.

c. Competencia

El mercado de software de facturación para autónomos y pequeñas empresas en España es altamente competitivo, con una amplia gama de soluciones que varían en funcionalidad, precio y facilidad de uso. A continuación, se presenta un análisis de algunos de los principales competidores en este sector:

1. FacturaScripts

- Empresa: Proyecto de código abierto desarrollado por la comunidad.

- Usuarios: Autónomos y pequeñas empresas que buscan una solución personalizable.
- Características principales: Gestión de facturas, clientes, proveedores, productos y servicios; extensible mediante plugins.
- Lo mejor: Gratuito, personalizable, comunidad activa.
- Lo peor: Requiere conocimientos técnicos para su instalación y mantenimiento.
- Precio: Gratuito.

2. Billin

- Empresa: Startup española especializada en facturación electrónica.
- Usuarios: Autónomos y pymes que necesitan emitir facturas electrónicas de forma sencilla.
- Características principales: Emisión y recepción de facturas electrónicas, integración con la Agencia Tributaria, gestión de presupuestos y gastos.
- Lo mejor: Interfaz intuitiva, cumplimiento con normativas fiscales.
- Lo peor: Algunas funcionalidades avanzadas requieren suscripción de pago.
- Precio: Plan gratuito con funciones básicas; planes de pago desde 10€/mes.

3. Facturas Cloud

- Empresa: Software de facturación en la nube.
- Usuarios: Autónomos y pymes que buscan una solución online accesible desde cualquier dispositivo.
- Características principales: Emisión de facturas, gestión de clientes y productos, informes financieros.
- Lo mejor: Accesibilidad desde cualquier lugar, interfaz sencilla.
- Lo peor: Limitaciones en el plan gratuito, soporte técnico limitado.
- Precio: Plan gratuito con funciones básicas; planes de pago desde 7,95€/mes.

4. Contasimple

- Empresa: Empresa española especializada en software de gestión para autónomos.
- Usuarios: Autónomos y profesionales que necesitan llevar la contabilidad y facturación de su negocio.

- Características principales: Facturación, contabilidad, gestión de impuestos, libros contables.
- Lo mejor: Cumplimiento con normativa fiscal española, generación de modelos oficiales.
- Lo peor: Interfaz menos moderna, curva de aprendizaje.
- Precio: Plan gratuito con funciones limitadas; planes de pago desde 6€/mes.

5. Invoice Ninja

- Empresa: Proyecto de código abierto con opción de alojamiento propio o en la nube.
- Usuarios: Freelancers y pequeñas empresas que buscan una solución flexible y personalizable.
- Características principales: Facturación, seguimiento de pagos, gestión de clientes y proyectos.
- Lo mejor: Altamente personalizable, opción de autoalojamiento.
- Lo peor: Requiere conocimientos técnicos para su implementación.
- Precio: Gratuito para autoalojamiento; planes en la nube desde 10€/mes.

6. STEL Order

- Empresa: Empresa española con sede en Murcia.
- Usuarios: Pymes y autónomos que necesitan una solución integral de gestión.
- Características principales: Facturación, gestión de clientes, control de stock, agenda de trabajo.
- Lo mejor: Versión gratuita adaptada a la normativa española, interfaz intuitiva.
- Lo peor: Algunas funciones avanzadas requieren suscripción de pago.
- Precio: Versión gratuita disponible; planes de pago según funcionalidades.

7. Factura Directa

- Empresa: Empresa española especializada en software de facturación online.
- Usuarios: Autónomos y pymes que buscan una solución sencilla y efectiva.
- Características principales: Emisión de facturas, presupuestos, albaranes, gestión de clientes y productos.
- Lo mejor: Interfaz amigable, cumplimiento con normativa fiscal.

- Lo peor: Limitaciones en el plan gratuito, soporte técnico limitado.
- Precio: Plan gratuito con funciones básicas; planes de pago desde 10€/mes.

8. Factusol

- Empresa: Software DELSOL, empresa española con amplia trayectoria.
- Usuarios: Pymes que requieren una solución de escritorio para la gestión de su negocio.
- Características principales: Facturación, contabilidad, gestión de almacén, compras y ventas.
- Lo mejor: Solución robusta y completa para la gestión empresarial.
- Lo peor: Requiere instalación en el equipo, interfaz menos moderna.
- Precio: Versión gratuita disponible; planes de pago según funcionalidades.

9. Cuéntica

- Empresa: Startup española enfocada en la gestión financiera para autónomos.
- Usuarios: Autónomos y freelancers que buscan una solución sencilla para llevar su contabilidad.
- Características principales: Facturación, contabilidad, gestión de impuestos, informes financieros.
- Lo mejor: Interfaz moderna, enfoque en la simplicidad.
- Lo peor: Limitaciones en el plan gratuito, soporte técnico limitado.
- Precio: Planes de pago desde 10€/mes.

10. Quipu

- Empresa: Empresa española especializada en software de gestión para autónomos y pymes.
- Usuarios: Autónomos y pequeñas empresas que buscan una solución integral de facturación y contabilidad.
- Características principales: Facturación, contabilidad, gestión de impuestos, integración bancaria.
- Lo mejor: Automatización de procesos, cumplimiento con normativa fiscal.
- Lo peor: Precio más elevado en comparación con otras soluciones.
- Precio: Planes de pago desde 13€/mes.

Resumen comparativo

Producto	Precio desde	Tipo de solución	Lo mejor	Lo peor
FacturaScripts	Gratuito	Código abierto	Personalizable, gratuito	Requiere conocimientos técnicos
Billin	10€/mes	SaaS	Interfaz intuitiva, cumplimiento fiscal	Funciones avanzadas en planes de pago
Facturas Cloud	7,95€/mes	SaaS	Accesibilidad, interfaz sencilla	Limitaciones en plan gratuito
Contasimple	6€/mes	SaaS	Cumplimiento fiscal, generación de modelos	Interfaz menos moderna
Invoice Ninja	Gratuito	Código abierto	Personalizable, opción de autoalojamiento	Requiere conocimientos técnicos
STEL Order	Gratuito	SaaS	Adaptado a normativa española, intuitivo	Funciones avanzadas en planes de pago
Factura Directa	10€/mes	SaaS	Interfaz amigable, cumplimiento fiscal	Limitaciones en plan gratuito
Factusol	Gratuito	Escritorio	Solución robusta y completa	Requiere instalación, interfaz menos moderna
Cuéntica	10€/mes	SaaS	Interfaz moderna, enfoque en simplicidad	Limitaciones en plan gratuito
Quipu	13€/mes	SaaS	Automatización, cumplimiento fiscal	Precio más elevado

d. Innovación

FacturaApp se diferenciará de la competencia por su enfoque innovador dentro del mercado de herramientas de facturación para autónomos y pequeñas empresas, destacando en los siguientes aspectos:

- Solución gratuita, sin limitaciones esenciales: A diferencia de muchas plataformas con funciones básicas restringidas, FacturaApp ofrece todas sus funciones principales sin coste, permitiendo al usuario emitir facturas legales completas desde el primer momento.
- Interfaz intuitiva para usuarios no técnicos: El diseño de la aplicación prioriza la sencillez y usabilidad, evitando la complejidad típica de los programas contables. Todo está organizado por módulos independientes (clientes, artículos, facturas, empresa) y accesible desde una navegación clara.
- Generación automática de facturas en PDF: La aplicación genera facturas legalmente válidas con numeración automática, sumas calculadas, y estructura profesional, preparadas para su envío o impresión, sin necesidad de plantillas externas.
- Accesibilidad desde cualquier dispositivo: Al estar desplegada en la nube (Firebase para frontend, PythonAnywhere para backend), los usuarios pueden acceder a la plataforma desde cualquier navegador web, sin instalaciones, desde PC, tablet o móvil.

- Preparada para integrar tecnologías avanzadas: FacturaApp está diseñada para escalar, incluyendo planes futuros de integración con tecnologías de OCR, IA y lectura automatizada de documentos (por ejemplo, para convertir tickets o facturas escaneadas en datos digitales).
- Desarrollo propio y transparente: El código fuente está disponible en GitHub, lo que promueve la transparencia y permite a otros desarrolladores colaborar o adaptar el proyecto a sus propias necesidades.
- Enfoque educativo y de empoderamiento digital: Además de su valor funcional, FacturaApp sirve como modelo educativo de buenas prácticas en desarrollo web, arquitectura cliente-servidor, seguridad, y despliegue real de aplicaciones útiles.
- Diseñada para cumplir con normativa vigente: Desde el almacenamiento de datos hasta el formato de las facturas generadas, FacturaApp está pensada para ayudar al usuario a cumplir con los requisitos legales básicos sin complicaciones.

e. Análisis DAFO

Debilidades

1. Falta de reconocimiento de marca: Como herramienta nueva, FacturaApp aún no cuenta con una base sólida de usuarios ni visibilidad en el mercado.
2. Recursos limitados: La disponibilidad de tiempo, personal y financiación para desarrollar nuevas funcionalidades o implementar soporte técnico puede ser limitada.
3. Ausencia de soporte profesional: Al ser un proyecto gratuito y abierto, carece de asistencia técnica dedicada, lo que puede afectar a usuarios que necesiten ayuda inmediata.
4. Curva de aprendizaje en ciertos casos: Algunos usuarios con bajo nivel digital podrían requerir orientación inicial, aunque la interfaz esté diseñada para ser intuitiva.

Amenazas

1. Alta competencia: Existen muchas soluciones de facturación bien posicionadas, tanto gratuitas como de pago, lo que dificulta captar usuarios sin diferenciación clara.
2. Cambios en la normativa fiscal: La legislación tributaria cambia con frecuencia; no adaptar la aplicación a tiempo podría hacerla obsoleta o legalmente insuficiente.
3. Percepción de lo gratuito: Algunos usuarios pueden asociar lo gratuito con baja calidad o inseguridad, lo que puede frenar su adopción.
4. Riesgos tecnológicos: Fallos en el servidor, pérdida de datos o ataques de seguridad pueden generar desconfianza si no se implementan medidas preventivas claras.

Fortalezas

1. **Gratuidad total:** FacturaApp es completamente gratuita, sin limitaciones funcionales esenciales, lo que la hace muy atractiva para autónomos y pequeños negocios.
2. **Accesibilidad desde cualquier lugar:** Al estar basada en la nube, se puede acceder a ella desde cualquier dispositivo con navegador.
3. **Generación automática de facturas en PDF:** Permite emitir documentos legales con numeración, totales e información fiscal sin depender de plantillas externas.
4. **Enfoque en la simplicidad:** Interfaz modular, limpia y pensada para usuarios no técnicos.
5. **Código abierto y transparente:** El proyecto está disponible en GitHub, lo que favorece la colaboración, revisión y adaptación.
6. **Desarrollo escalable:** Arquitectura separada (frontend y backend) que facilita la incorporación de mejoras futuras.

Oportunidades

1. **Creciente digitalización de autónomos y pymes:** Cada vez más profesionales buscan soluciones digitales que les faciliten tareas administrativas sin coste.
2. **Entrada en vigor de la factura electrónica obligatoria:** El marco legal impulsa a los usuarios a buscar herramientas compatibles con los nuevos requisitos.
3. **Integración futura con IA y OCR:** Tecnologías como el reconocimiento automático de documentos pueden convertir a FacturaApp en una solución avanzada con bajo umbral de entrada.
4. **Promoción en comunidades de emprendedores:** Grupos, foros y redes de autónomos pueden ser canales efectivos para ganar usuarios y visibilidad.

6. Análisis de requisitos

a. Requisitos funcionales

Los requisitos funcionales definen las capacidades esenciales que debe tener FacturaApp para cumplir con su propósito como herramienta de facturación digital gratuita para autónomos y pequeñas empresas. A continuación, se detallan los requisitos clave:

1. **Inicio de sesión y autenticación de usuarios**
La aplicación debe permitir el registro y acceso de usuarios mediante un sistema seguro de autenticación con credenciales personales.
2. **Gestión de empresas**
El sistema debe permitir al usuario registrar y editar la información de su empresa (nombre fiscal, NIF, dirección, datos de contacto), asociada a sus facturas.

3. Gestión de clientes

Los usuarios deben poder crear, visualizar, modificar y eliminar fichas de clientes, incluyendo sus datos fiscales, dirección y contacto.

4. Gestión de artículos o servicios

Se debe ofrecer un módulo para registrar productos o servicios facturables, con nombre, descripción, precio unitario y tipo de IVA.

5. Creación y gestión de facturas

Los usuarios deben poder generar facturas que incluyan una empresa, un cliente, fecha, número automático, líneas de productos/servicios y totales calculados automáticamente.

6. Generación automática de facturas en formato PDF

La aplicación debe permitir generar y descargar cada factura como archivo PDF, con formato profesional y válido legalmente.

7. Visualización y no edición de facturas existentes

Los usuarios deben poder acceder al historial de facturas emitidas, ordenarlas por fecha o cliente y pero nunca modificar su contenido aun que fuese necesario; para este caso existen facturas rectificativas en la que se indicara la factura que rectifica.

8. Eliminar registros

La aplicación debe ofrecer la opción de eliminar clientes, artículos pero nunca facturas (ya que una factura nunca se puede eliminar según la ley Española), con confirmación previa para evitar errores.

9. Validación de datos y control de errores

Todos los formularios de entrada deben incluir validación para evitar campos vacíos, formatos incorrectos o duplicación de datos.

10. Gestión multiusuario (previsto)

Aunque no en la versión inicial, se prevé una futura funcionalidad que permita que una misma cuenta tenga múltiples usuarios con distintos roles (administrador, técnico...).

11. Panel de usuario

Debe existir un área personal desde la cual el usuario pueda consultar y gestionar todos sus datos, incluyendo empresa, facturas, clientes y artículos.

12. Acceso desde dispositivos móviles y escritorio

La interfaz debe ser responsive para que el usuario pueda operar cómodamente desde móviles, tablets o PCs.

13. Integración con backend mediante llamadas asíncronas (AJAX)

Las acciones de guardado, edición o borrado deben funcionar sin recargar toda la página, mediante peticiones HTTP al backend.

b. Requisitos no funcionales

Los requisitos no funcionales definen las características de calidad que debe cumplir FacturaApp para ofrecer una experiencia fiable, segura y profesional a sus usuarios. Estos requisitos complementan a los funcionales y son fundamentales para el correcto funcionamiento y mantenimiento de la plataforma.

1. Rendimiento

La plataforma debe ser rápida y eficiente, con tiempos de respuesta inferiores a 2 segundos en operaciones básicas como el guardado de datos o la generación de PDFs, garantizando una experiencia de usuario fluida.

2. Seguridad

Se implementarán medidas de seguridad sólidas, incluyendo:

- Cifrado de contraseñas mediante algoritmos hash.
- Validación de entradas para prevenir inyecciones de código (XSS, SQLi).
- Control de sesiones y autenticación segura.
- Cumplimiento con el RGPD (Reglamento General de Protección de Datos).

3. Escalabilidad

La arquitectura del sistema (frontend-backend desacoplado) debe permitir su ampliación para soportar un mayor volumen de usuarios y funcionalidades sin comprometer el rendimiento.

4. Usabilidad

La interfaz debe ser intuitiva y amigable para el usuario, minimizando la curva de aprendizaje y evitando la necesidad de conocimientos técnicos. El diseño debe facilitar la navegación, el acceso rápido a funciones y la edición de datos.

5. Disponibilidad

La plataforma debe estar operativa 24/7, con un tiempo de inactividad mínimo. El backend está desplegado en PythonAnywhere y el frontend en Firebase para garantizar accesibilidad desde cualquier lugar.

6. Compatibilidad

El sistema debe funcionar correctamente en los principales navegadores web (Chrome, Firefox, Edge, Safari) y en distintos sistemas operativos (Windows, macOS, Linux, Android, iOS).

7. Accesibilidad

Se debe garantizar que la aplicación sea accesible para usuarios con discapacidades, siguiendo recomendaciones básicas de accesibilidad web (contraste, navegación con teclado, etiquetas claras).

8. Mantenibilidad

El código debe estar estructurado y comentado adecuadamente para facilitar su mantenimiento, corrección de errores y futuras ampliaciones por otros desarrolladores. La separación de lógica por módulos (clientes, facturas, artículos...) contribuye a ello.

9. Cumplimiento normativo

La aplicación debe ajustarse a la normativa vigente en materia de facturación, protección de datos (RGPD) y comercio electrónico, asegurando que los documentos generados sean válidos a nivel legal.

c. Requisitos de seguridad y protección de datos

La seguridad y la protección de los datos personales y fiscales de los usuarios son aspectos críticos en el desarrollo de FacturaApp, especialmente por el tratamiento de información sensible como datos de clientes, facturas, empresas y correos electrónicos. A continuación se detallan los principales requisitos en esta área:

1. Cifrado de contraseñas

- Las contraseñas de los usuarios deben almacenarse de forma segura mediante algoritmos de hash (por ejemplo, PBKDF2, bcrypt o sha256 con salt).
- No se deben guardar contraseñas en texto plano en ningún momento.

2. Autenticación segura

- Se debe implementar un sistema de login robusto que controle el acceso no autorizado.
- Las sesiones deben gestionarse mediante tokens o cookies seguras con expiración automática.

3. Validación y sanitización de datos

- Todos los datos introducidos por el usuario deben ser validados tanto en frontend como en backend para evitar ataques como:
 - Inyecciones SQL
 - XSS (Cross-Site Scripting)
 - CSRF (Cross-Site Request Forgery)

4. Acceso controlado a los datos

- Cada usuario solo podrá ver, editar o eliminar los datos pertenecientes a su propia empresa.
- No se debe permitir acceso cruzado entre cuentas o usuarios.

5. Copia de seguridad (Backup)

- Se deben realizar copias de seguridad periódicas de la base de datos para garantizar la recuperación de datos en caso de fallo técnico.

6. Política de privacidad

- La aplicación debe contar con un aviso legal y política de privacidad, especificando el uso y almacenamiento de los datos personales conforme al RGPD.

7. Derechos de los usuarios

- Los usuarios deben poder solicitar la modificación o eliminación de sus datos personales, y eliminar su cuenta completamente si lo desean.

8. Protección de facturas y documentos

- Los documentos PDF generados deben estar protegidos frente a accesos no autorizados (no públicos por defecto).
- Se debe evitar exponer URLs previsibles para archivos sensibles.

9. Auditoría básica de actividad (*previsto*)

- En versiones futuras, se contempla la implementación de un sistema de trazabilidad que registre actividades críticas (edición, eliminación, accesos) para mayor control.

10. Cumplimiento legal (RGPD)

- FacturaApp debe cumplir con el Reglamento General de Protección de Datos en cuanto al consentimiento, tratamiento, almacenamiento y eliminación de información personal.

d. Requisitos de accesibilidad y usabilidad

La accesibilidad y la usabilidad son aspectos fundamentales en el diseño de FacturaApp, especialmente porque está dirigida a un público diverso, compuesto por autónomos y pequeñas empresas que pueden tener distintos niveles de experiencia digital. El objetivo es garantizar que cualquier usuario pueda utilizar la aplicación de forma sencilla, eficiente y sin barreras.

Usabilidad

1. Interfaz intuitiva

- La plataforma debe presentar una estructura clara, organizada por módulos (clientes, facturas, artículos, empresa), con menús y botones comprensibles y bien posicionados.

2. Navegación fluida

- Los usuarios deben poder acceder a cualquier sección con un máximo de 2 o 3 clics desde la pantalla principal.

3. Lenguaje claro y directo

- El contenido textual (etiquetas, mensajes de error, botones) debe usar un lenguaje accesible, evitando tecnicismos innecesarios.

4. Feedback inmediato

- Cada acción realizada por el usuario (guardar, borrar, enviar) debe generar una respuesta visual clara (mensajes de éxito, error o advertencia).

5. Diseño responsive

- La aplicación debe adaptarse correctamente a diferentes dispositivos y tamaños de pantalla (ordenador, tablet, móvil) mediante CSS y media queries.

Accesibilidad

1. Compatibilidad con teclado

- Todas las funciones clave deben poder ejecutarse usando únicamente el teclado, para facilitar el uso a personas con dificultades motrices.

2. Contraste de colores adecuado

- Los elementos visuales deben respetar un contraste suficiente entre texto y fondo (mínimo 4.5:1), siguiendo las pautas WCAG.

3. Tamaños de fuente legibles

- Los textos deben ser ampliables sin pérdida de estructura o legibilidad.

4. Uso de etiquetas y atributos ARIA

- Los formularios deben contar con etiquetas descriptivas (label) y atributos ARIA para ser compatibles con lectores de pantalla.

5. Evitar dependencias exclusivas de color

- La información no debe depender únicamente del color para transmitirse (por ejemplo, usar iconos y texto además de color para mostrar errores).

6. Mensajes de error accesibles

- Los mensajes deben ser claros, visibles y relacionados directamente con el campo que ha generado el error.

7. Diseño

a. Diagrama de interfaces

En **FacturaApp**, la experiencia del usuario es uno de los pilares del diseño. Se ha trabajado con especial cuidado para ofrecer una interfaz visual limpia, accesible y adaptada a profesionales que necesitan gestionar su facturación de forma rápida y sin complicaciones.

Paleta de colores: Profesionalismo y simplicidad

La paleta cromática está inspirada en la neutralidad visual y la profesionalidad. Se han elegido tonos sobrios como el azul, el gris oscuro y el blanco para transmitir confianza, claridad y orden, en sintonía con el propósito administrativo de la aplicación.

- Azul oscuro (#0d6efd): Representa la estabilidad, la confianza y la seguridad. Es el color principal en encabezados y botones de acción.
- Gris claro (#f8f9fa): Utilizado como fondo, aporta ligereza visual y facilita la lectura.
- Blanco (#ffffff): Para mantener la claridad y limpieza en formularios, tablas y secciones amplias.
- Negro / Gris oscuro (#212529): Para títulos y texto, asegurando contraste y legibilidad.

Tipografía:

Se emplea la fuente "Console" por su legibilidad, estilo moderno y soporte multi-dispositivo.

Logotipo: Un sello de funcionalidad

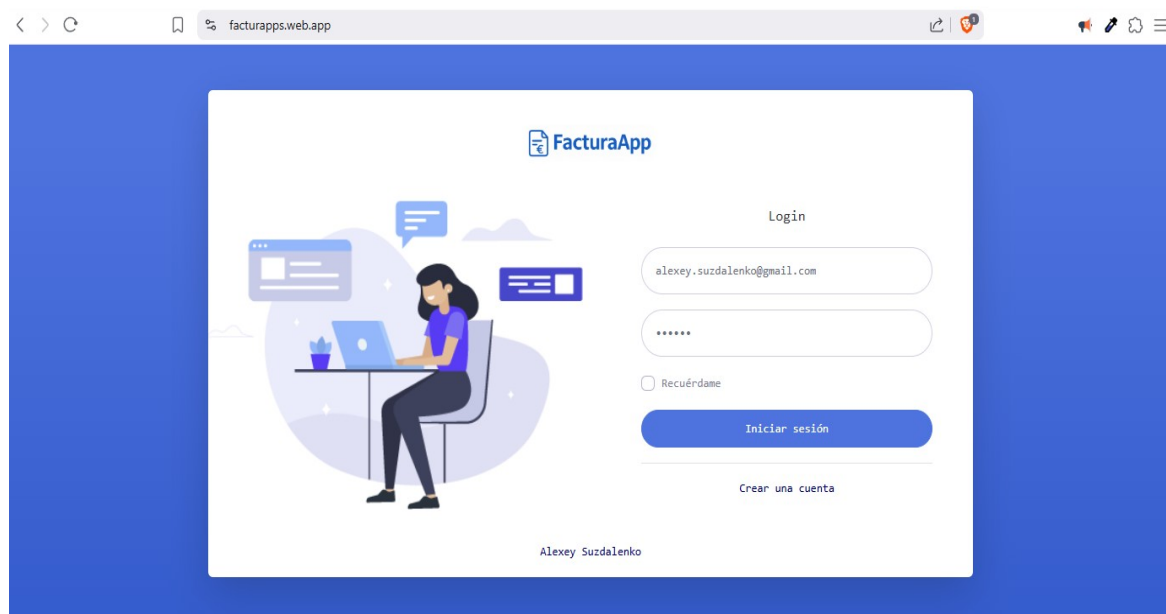
El logotipo de FacturaApp está compuesto por el nombre de la marca acompañado de un icono simple que representa una factura, reforzando los valores de legalidad, eficacia y sencillez.

Estructura común de las páginas

Todas las páginas y secciones de la aplicación comparten una estructura base, que garantiza coherencia y facilita el uso:

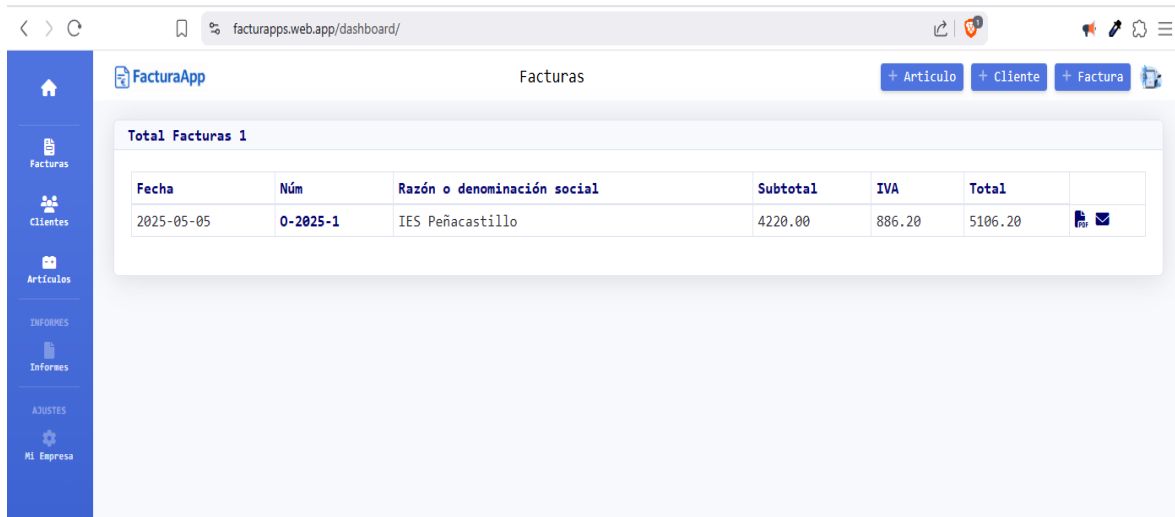
1. Página login



- Logo de la aplicación
- Interfaz de inicio de sesión simple con campos para email y contraseña
- Registro con campos obligatorios para crear una cuenta vinculada a una empresa
- Botón de "Iniciar sesión"
- Enlaces a "Crear una cuenta"
- Diseño responsive para móviles y tablets



2. Dashboard / Página de inicio tras iniciar sesión

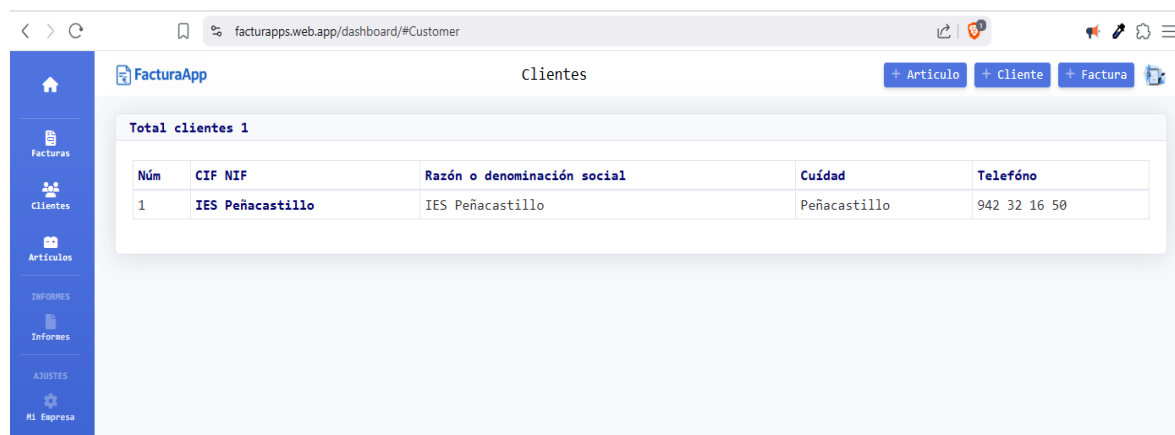
- Acceso rápido a las secciones clave: “Facturas”, “Clientes”, “Artículos”, “Mi Empresa”
- Una vez que hemos creado una factura la vamos a visualizar nada mas entrar en dashboard.
- Botones de acciones “+ Artículo”, “+ Cliente”, “+ Factura” para crear nuevas entidades.



Fecha	Núm	Razón o denominación social	Subtotal	IVA	Total	
2025-05-05	0-2025-1	IES Peñacastillo	4220.00	886.20	5106.20	 

3. Gestión de clientes

- Tabla con listado de clientes: Numero, CIF/NIF, Nombre, Ciudad, teléfono
- Botón para añadir, editar clientes



Núm	CIF NIF	Razón o denominación social	Ciudad	Teléfono
1	IES Peñacastillo	IES Peñacastillo	Peñacastillo	942 32 16 50

- Formulario de creación de nuevo cliente

FacturaApp

Crear Cliente

+ Artículo + Cliente + Factura

Denominación

Código cliente

CIF NIF

Razón o denominación social

Nombre y apellidos persona

Email

Teléfono

Dirección

País

Provincia

Código postal

Ciudad

Dirección

Guardar

4. Gestión de artículos

- Listado de productos o servicios con número, descripción, precio, IVA, tipo de IVA.

FacturaApp

Artículos

+ Artículo + Cliente + Factura

Total artículos 2

Número	Descripción	Precio unidad	IVA	Tipo
2	Informe Power BI	3000.00 €	21.00 %	norm
1	Desarrollo App Personalizada	1000.00 €	21.00 %	norm

- Funcionalidad CRUD (crear, leer, actualizar)

FacturaApp

Crear Artículo

+ Artículo + Cliente + Factura

Descripción artículo

Nombre

Datos de facturación

Precio €/unidad

Porcentaje de IVA

21 %

Guardar

5. Crear nueva factura

- Formulario con selección de tipo de factura (ordinaria/rectificativa/abono), cliente, datos de vehículo (matricula/marca).
- Lineas de factura que incluyen el nombre de articulo, cantidades, descuentos, importe, el IVA aplicado y el importe por linea.
- Cálculo automático de totales e impuestos
- Linea adicional con el mano de obra el precio y cantidad.
- Campo opcional para observaciones de la factura.
- Botón crear la factura, una vez que la factura se crea ya es imposible modificarla o borrarla. Al crear la factura esta automáticamente llega por correo tanto al cliente como al la misma empresa(autónomo) que esta facturando.

The screenshot shows the 'Crear Factura' (Create Invoice) form in the FacturaApp web application. The form is divided into several sections:

- Tipo Factura:** A dropdown menu showing 'Factura Ordinaria'.
- Datos cliente y tipo de factura:** Fields for 'Número cliente' (1), 'NIF CIF cliente' (IES Peñacastillo), and 'Razón o denominación social' (IES Peñacastillo).
- Datos vehículo:** Fields for 'Matrícula' and 'Marca/Modelo/Kilómetros'.
- Líneas de factura:** A table with columns: Código Art., Descripción Art., Cantidad, Precio, % Dto., Importe, and IVA. It contains two lines:

Código Art.	Descripción Art.	Cantidad	Precio	% Dto.	Importe	IVA
1	Desarrollo App Personalizada	2	1000,00	0	2000,00	21 %
2	Informe Power BI	2	3000,00	0	6000,00	21 %

 Below the table, there is a '+ Linea' button and a new line with 'Mano de obra', '10' quantity, '22,00' price, and '21 %' IVA.
- Summary:** Subtotal: 8220.00, IVA: 1726.20, Total: 9946.20.
- Observaciones:** A text area labeled 'texto de observacion'.
- Buttons:** '+ Articulo', '+ Cliente', '+ Factura', and a large 'Crear Factura' button at the bottom.

6. Gestión de empresa

- Panel para introducir o actualizar los datos fiscales y de contacto de la empresa emisora de las facturas

FacturaApp Mi Empresa

+ Artículo + Cliente + Factura

Denominación

Razón o denominación social
Factura App SLNE

Nombre y apellidos persona
Suzdalenko Alexey

Email cara cliente
alexey.suzdalenko@gmail.com

CIF-NIF
A123456

Email login
alexey.suzdalenko@gmail.com

Dirección

País
España

Provincia
Cantabria

Código postal
39694

Ciudad
Santa Maria de Cayon

Dirección
El Traguezo 66

Teléfono principal
+34657666135

Teléfono
+34657666135

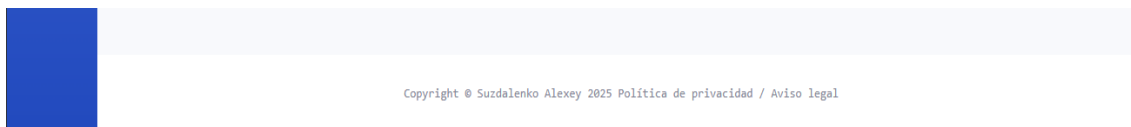
Precio

Precio €/hora mano de obra
22,00

Guardar

7. Footer (pie de página)

- Enlace a Política de privacidad y Aviso legal
- Información básica de contacto y créditos del proyecto



b. Diagrama físico y/o lógico de red

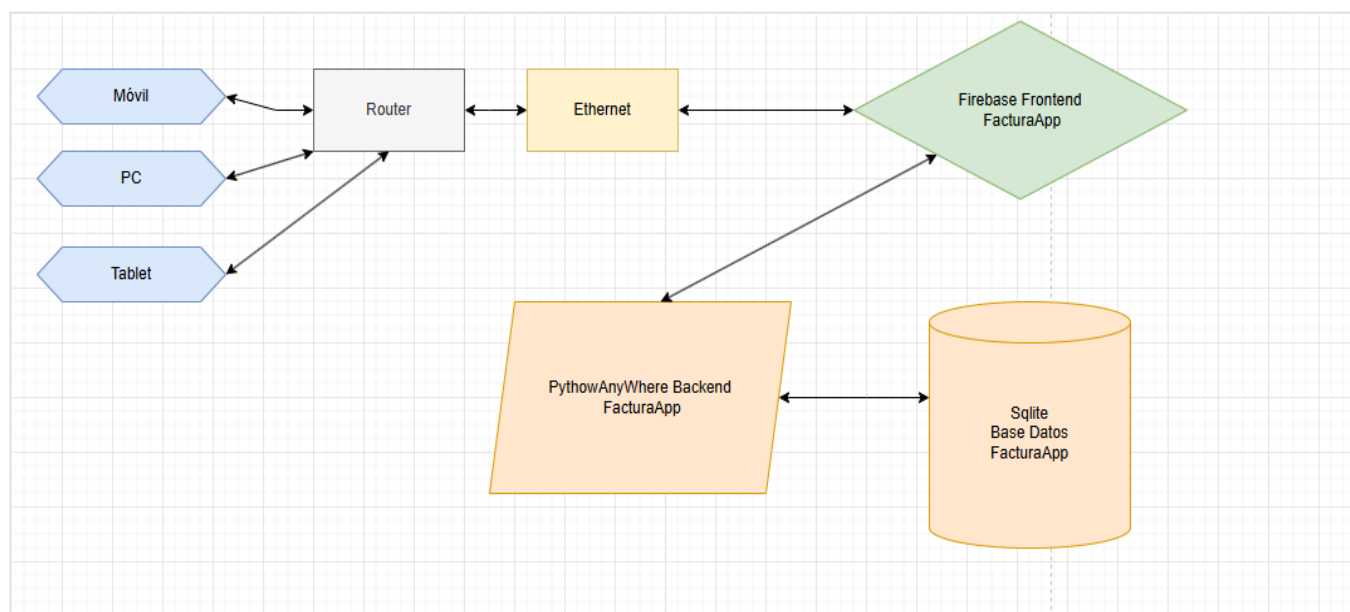
El funcionamiento de FacturaApp se basa en una arquitectura web modular, donde el frontend y el backend están desplegados de forma independiente y se comunican mediante peticiones HTTP. Esta separación favorece la escalabilidad, el mantenimiento y la seguridad de la aplicación.

Elementos del diagrama:

- FacturaApp Frontend (Firebase Hosting)
Representa la interfaz web accesible desde el navegador, desarrollada en HTML, CSS y JavaScript. Se encarga de mostrar los formularios, tablas, menús y permite la interacción del usuario con la aplicación.
Desplegado en Firebase.
- FacturaApp Backend (PythonAnywhere)
Este servidor, construido con Django (Python), recibe las peticiones del frontend y responde con los datos necesarios (clientes, facturas, artículos, empresa). Además, genera los PDFs y gestiona la autenticación.
Protegido por firewall de PythonAnywhere y control de sesiones.
- SQLite Database
La base de datos utilizada es SQLite, alojada en el backend. Contiene la información de los

usuarios, empresas, artículos, clientes y facturas. Es ligera y adecuada para el tipo de uso previsto.

- Router / Ethernet / WiFi
Es el medio por el que los usuarios acceden a la aplicación desde cualquier ubicación. Puede ser mediante conexión de red cableada o inalámbrica.
- Dispositivos del usuario final (PC, móvil, tablet)
Desde cualquiera de estos dispositivos, el usuario puede acceder a FacturaApp a través del navegador, sin necesidad de instalación local.



Resumen de funcionamiento del sistema (FacturaApp)

El usuario accede a FacturaApp desde cualquier dispositivo (PC, móvil o tablet) mediante un navegador web, a través de una conexión de red (router/Ethernet). El frontend de la aplicación, alojado en Firebase Hosting, se encarga de mostrar la interfaz al usuario y enviar peticiones HTTP al backend desarrollado en Django y desplegado en PythonAnywhere. El backend procesa estas peticiones, consulta o modifica los datos almacenados en la base de datos SQLite, y responde con la información solicitada.

Toda la comunicación entre frontend, backend y base de datos está protegida por un sistema de seguridad en el servidor remoto, garantizando la confidencialidad, integridad y disponibilidad de los datos.

d. Diagrama entidad/relación (E/R)

El diagrama de base de datos de FacturaApp refleja cómo se estructuran las entidades clave del sistema, cómo se relacionan entre sí mediante claves foráneas y cómo se almacena la información crítica relacionada con la facturación. Este modelo físico está diseñado sobre SQLite, optimizado para rendimiento, consistencia de datos y escalabilidad.

Entidades:

- COMPANY (Empresa)
Identificada por id, contiene los datos de la empresa emisora de facturas.
Atributos: razón social, CIF, email, teléfono, dirección, país, código postal, usuario, etc.
- CUSTOMER (Cliente)
Identificada por id, representa a los clientes receptores de las facturas.
Atributos: nombre o razón social, CIF/NIF, email, dirección, localidad, etc.
- ARTICLE (Artículo)
Representa los productos o servicios facturables.
Atributos: descripción, código, precio, IVA, tipo de IVA.
- FACTURA
Es la factura principal, identificada por id.
Atributos: número, serie, fecha, subtotal, total, desglose de IVA, observaciones, etc.
- FACTURALINEAS
Representa las líneas de cada factura.
Atributos: cantidad, precio, artículo, descuento, importe, IVA aplicado, etc.
- DOCUMENT (Documento)
Documento contable adicional (numeradores).
Atributos: descripción, valor, ejercicio.
- VEHICLEDATA
Datos de vehículos asociados opcionalmente a una factura (por ejemplo, en talleres).
Atributos: matrícula, otro dato (color).
- COUNTRY
Lista de países. Atributos: id, descripción.
- TIPOIVA
Representa los distintos tipos de IVA que pueden aplicarse (por ejemplo, general, reducido, exento con el valor de porcentaje).
Atributos: tipo (nombre) y porcentaje.

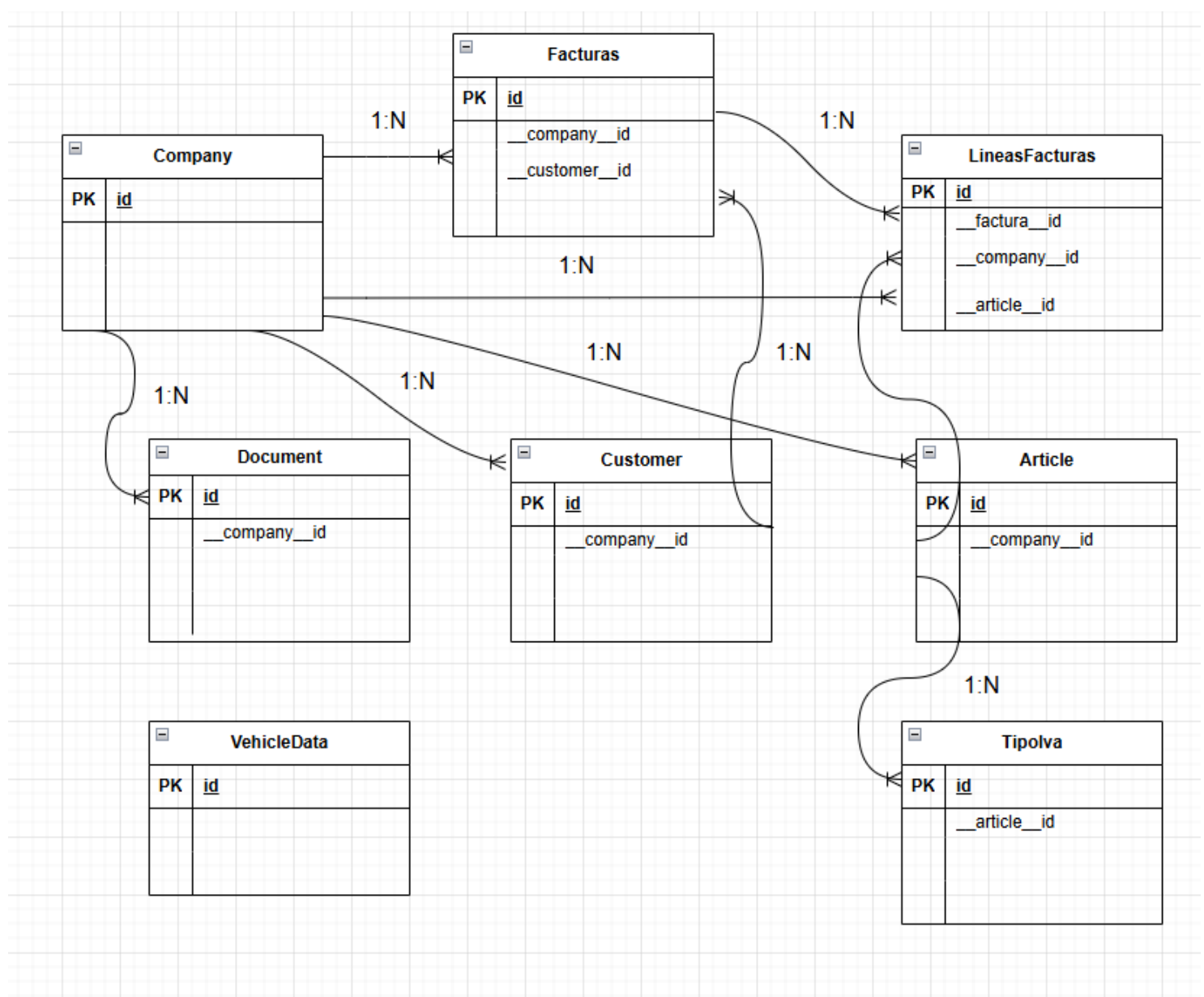
Relaciones:

- Una empresa (Company) puede tener muchas facturas, clientes, artículos, documentos y vehículos.
- Una factura pertenece a una única empresa y a un único cliente.
- Una factura tiene muchas líneas de factura (FacturaLineas).
- Cada línea de factura hace referencia a un artículo.
- Cada artículo está asociado a un tipo de IVA.
- Opcionalmente, una factura puede tener asociada información de un vehículo.

Relaciones entre claves:

Tabla	PK	FK
Facturas	id	__company_id, __customer_id
LineasFacturas	id	__factura_id, __company_id, __article_id
Customer	id	__company_id
Article	id	__company_id
Document	id	__company_id
Tipolva	id	__article_id

Captura de la Diagrama Entidad-Relación de la Base de Datos de FacturaApp:



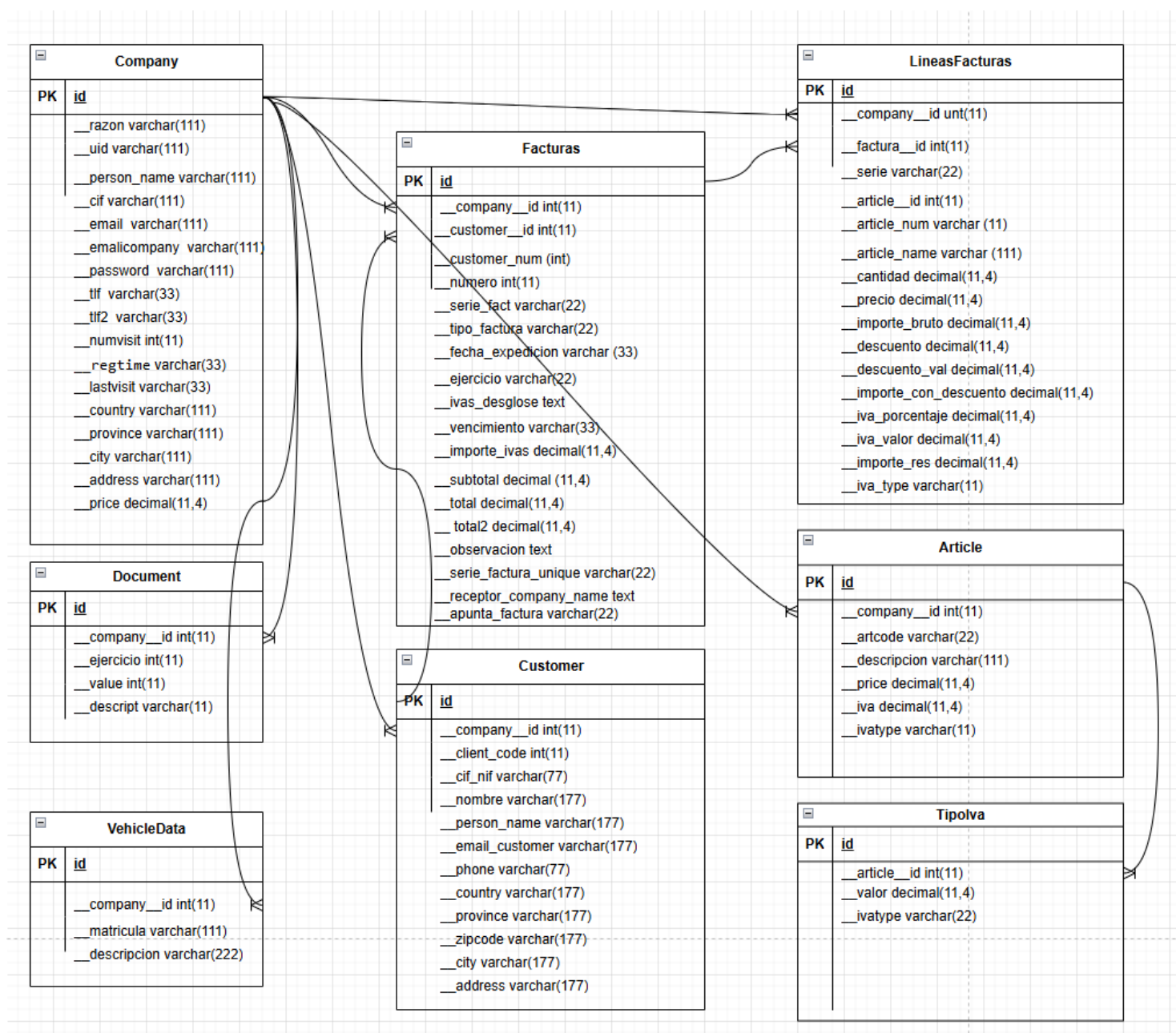
7.d. Diagrama de Base de Datos

El diseño de la base de datos de **FacturaApp** está basado en una arquitectura relacional que permite almacenar y organizar eficientemente la información generada por las empresas que utilizan el sistema. Esta base de datos, implementada en **SQLite**, está optimizada para gestionar facturación, clientes, artículos y documentos contables con integridad y rendimiento.

A continuación, se presenta el diagrama de base de datos en el que se muestran las tablas principales, las claves primarias (PK) y foráneas (FK), así como las relaciones lógicas entre entidades.

Descripción de las tablas principales

Tabla	Descripción
Company	Contiene los datos de las empresas que utilizan el sistema para emitir facturas.
Customer	Registra los datos fiscales y de contacto de los clientes asociados a una empresa.
Article	Incluye los productos o servicios facturables: descripción, precio e IVA.
Factura	Representa las facturas emitidas: cliente, empresa, fecha, totales, serie, etc.
FacturaLineas	Detalle de cada factura: productos, cantidades, precios, descuentos e impuestos.
Document	Registra datos contables como series y ejercicios.
VehicleData	Contiene información de vehículos asociados a una factura (opcional).
Tipolva	Tabla con los tipos de IVA disponibles (general, reducido, exento...).
Country	Lista de países utilizada para empresas y clientes.



Tablas:

1. Company

Contiene los datos fiscales y de contacto de la empresa emisora de facturas.

Campos clave:

- id (PK)
- razon, cif, email, password, address, etc.

Relaciones:

- 1:N con Customer, Article, Factura, Document, VehicleData.

2. Customer

Cientes de la empresa.

Campos clave:

- id (PK), company_id (FK)
- CIF/NIF, nombre, dirección, email, etc.

Relaciones:

- 1:N con Factura.
- N:1 con Company.

Article

Productos o servicios facturables.

Campos clave:

- id (PK), company_id (FK)
- Código del artículo, descripción, precio, IVA.

Relaciones:

- 1:N con LineasFacturas.
- N:1 con Company.
- 1:1 con Tipolva.

Factura

Factura emitida a un cliente.

Campos clave:

- id (PK), company_id (FK), customer_id (FK)
- Fecha, subtotal, total, observaciones, etc.

Relaciones:

- 1:N con LineasFacturas.
- N:1 con Company.
- N:1 con Customer.

LineasFacturas

Detalle línea por línea de una factura.

Campos clave:

- id (PK)
- factura_id, company_id, article_id (FKs)

- Cantidad, precio, IVA, descuentos, total línea.

Relaciones:

- N:1 con Factura.
- N:1 con Article.
- N:1 con Company.

Document

Control de numeración y ejercicio fiscal.

Campos clave:

- id (PK), company_id (FK)
- ejercicio, value, description

Relaciones:

- N:1 con Company.

VehicleData

Datos de vehículo vinculados a una factura (opcional, por ejemplo, talleres).

Campos clave:

- id (PK)
- invoice_id, company_id (FKs)
- Matrícula, otros datos (color, modelo...)

Relaciones:

- N:1 con Company.
- N:1 con Factura.

Tipolva

Tabla para los tipos de IVA aplicables a artículos.

Campos clave:

- id (PK), article_id (FK)
- Porcentaje de IVA, tipo

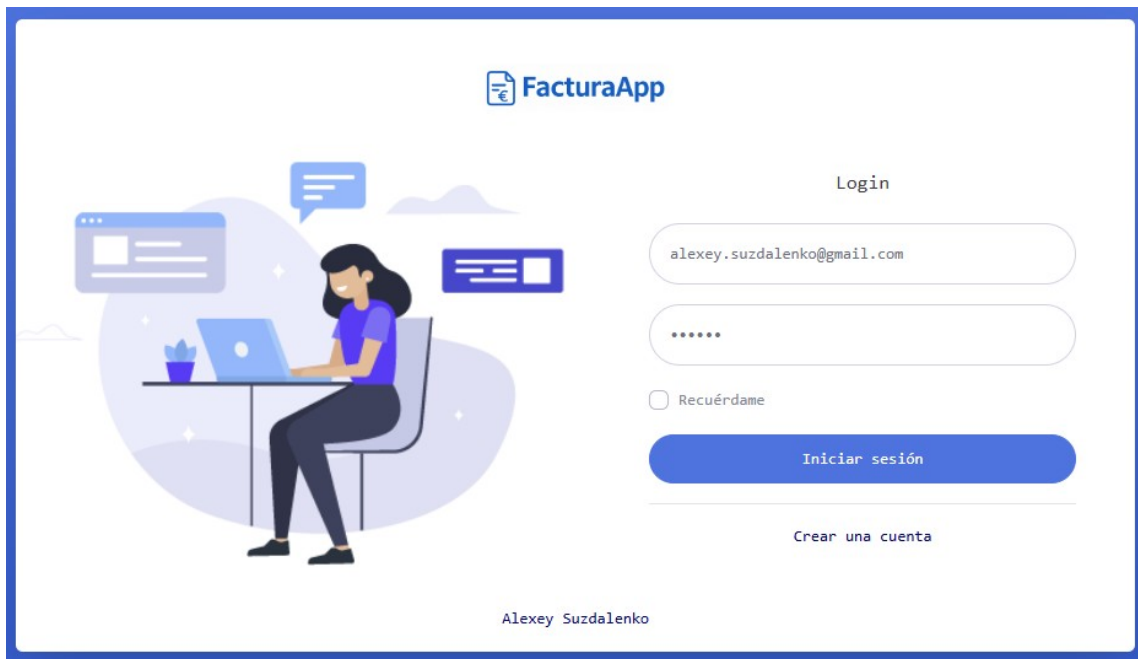
Relaciones:

- N:1 con Article.

7.f. Casos de uso

Caso de uso 1 – Emitir una factura

- Actor: Usuario (empresa registrada)
- Objetivo: Generar una nueva factura a un cliente.
- Flujo básico:
 1. El usuario accede al sistema con sus credenciales.



2. En el menú, selecciona “Crear nueva factura”.
3. Elige un cliente ya registrado.
1. Selecciona los artículos (productos o servicios) o añade uno nuevo.
4. El sistema calcula automáticamente subtotal, IVA y total.

FacturaApp

Crear Factura

+ Artículo

+ Cliente

+ Factura

Tipo Factura

Tipo
Ordinaria/Rectificativa/
Abono
Factura Ordinaria

Datos cliente y tipo de factura

Número cliente
1

NIF CIF cliente
IES Peñacastillo

Razón o denominación social
IES Peñacastillo

Datos vehículo

Matrícula

Marca/Modelo/Kilómetros

Líneas de factura

Código Art.	Descripción Art.	Cantidad	Precio	% Dto.	Importe	IVA
1	Desarrollo App Personalizada	2	1000,00	0	2000,	21 %
	Configuracion Servidor	1		0	0,00	21 %
+ Línea						
	Mano de obra	10	22,00		220,0	21 %

Sin

Subtotal: 2220.00
IVA: 466.20
Total: 2686.20

Crear Factura

5. El usuario confirma y descarga el PDF.

O-2025-2_IES Peñacastillo.pdf

1 / 1 | 75% +

FACTURA

Numero: O-2025-2
Fecha: 2025-05-09
Vencimiento: 2025-05-20
F.Pago: CONTADO

Emisor
Factura App SLNE
Suzdalenko Alexey
Cantabria, Santa Maria de Cayon
39694 El Traguezo 66
NIF: A123456 TEL: +34657666135

Cliente
#1 CIF: IES Peñacastillo IES Peñacastillo
IES Peñacastillo
39011 C. Eduardo García España
Cantabria, Peñacastillo
TEL: 942 32 16 50

Código Art.	Descripción	Cantidad	Precio	Dto.	Importe
1	Desarrollo App Personalizada	2.00	1000.00	0.00	2000.00
3	Configuracion Servidor	1.00	30.00	0.00	30.00
0	Mano de obra	10.00	22.00	0.00	220.00

BASE IMPONIBLE		I.V.A		I.R.E	
Base	%	Importe	%	Importe	
2250.00	21	472.50	5.20	0.00	
0.00	10	0.00	1.40	0.00	
0.00	4	0.00	0.00	0.00	
0.00	0	0.00	0.00	0.00	
0.00	DEXENTO	0.00	0.00	0.00	

RESUMEN	
Suma Importes	2250.00
I V A	472.50
R.Equivalencia	0.00
Total factura	2722.50
Firma Cliente	

Observaciones: Sin

- Excepciones: No hay artículos en la base de datos / Falta el CIF del cliente.

Caso de uso 2 – Registrar un cliente

- Actor: Usuario (empresa)
- Objetivo: Añadir un nuevo cliente a su base de datos.
- Flujo básico:

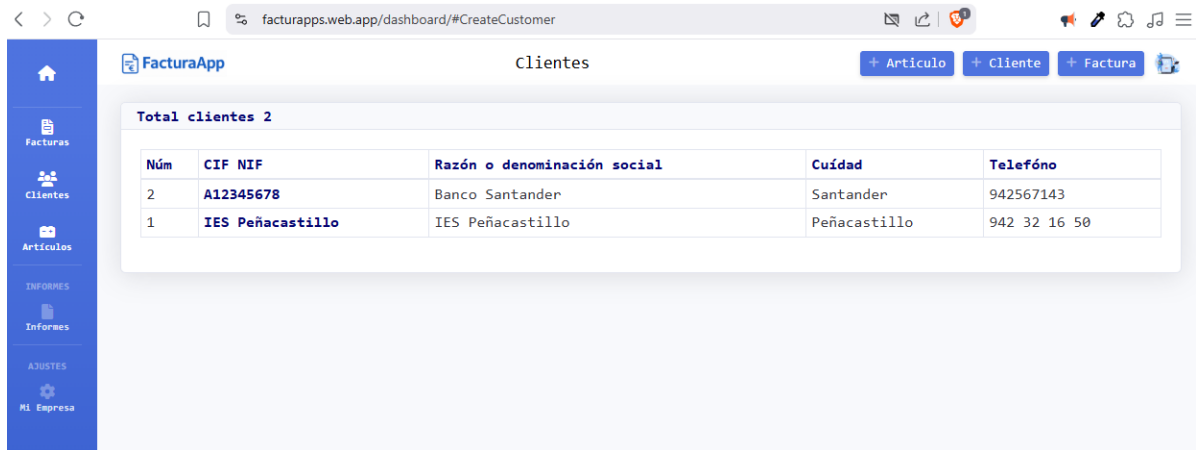
1. El usuario accede a “+ Cliente”.

The screenshot shows the 'Crear Cliente' (Create Client) form in the FacturaApp web application. The browser address bar shows 'facturapps.web.app/dashboard/#CreateCustomer'. The form is divided into two main sections: 'Denominación' (Name) and 'Dirección' (Address). The 'Denominación' section includes fields for 'Código cliente' (Client code), 'CIF NIF', 'Razón o denominación social' (Reason or social denomination), 'Nombre y apellidos persona' (Person's name and surnames), 'Email', and 'Teléfono' (Phone). The 'Dirección' section includes fields for 'País' (Country), 'Provincia' (Province), 'Código postal' (Postal code), 'Cuidad' (City), and 'Dirección' (Address). A 'Guardar' (Save) button is located at the bottom left of the form. The sidebar on the left contains navigation links for 'Facturas', 'Clientes', 'Artículos', 'Informes', 'Ajustes', and 'Mi Empresa'.

2. Rellena campos: nombre, NIF/CIF, dirección, etc.

The screenshot shows the 'Crear Cliente' form with sample data filled in. The 'Denominación' section contains: 'Código cliente' (empty), 'CIF NIF' (A12345678), 'Razón o denominación social' (Banco Santander), 'Nombre y apellidos persona' (Pedro Romeral), 'Email' (banco@santander.com), and 'Teléfono' (942567143). The 'Dirección' section contains: 'País' (España), 'Provincia' (Cantabria), 'Código postal' (39001), 'Cuidad' (Santander), and 'Dirección' (Calle Alcalá 11). The 'Guardar' button is still visible at the bottom left.

3. Guardo el registro y ahora visualizo le listado de clientes.



Núm	CIF NIF	Razón o denominación social	Ciudad	Teléfono
2	A12345678	Banco Santander	Santander	942567143
1	IES Peñacastillo	IES Peñacastillo	Peñacastillo	942 32 16 50

Caso de uso 3 – Añadir un artículo

- Actor: Usuario (empresa)
- Objetivo: Registrar un nuevo producto o servicio.
- Pasos:

1. Entra en “+ Artículos”.



Descripción artículo

Nombre

Datos de facturación

Precio €/unidad

Porcentaje de IVA

21 %

Guardar

2. Crea uno nuevo: descripción, precio, tipo de IVA.

FacturaApp Crear Artículo

+ Artículo + Cliente + Factura

Descripción artículo

Nombre
Desarrollo API personalizada

Datos de facturación

Precio €/unidad
1000

Porcentaje de IVA
21 %

Guardar

3. Guarda.

4. El artículo queda disponible para facturar.

FacturaApp Artículos

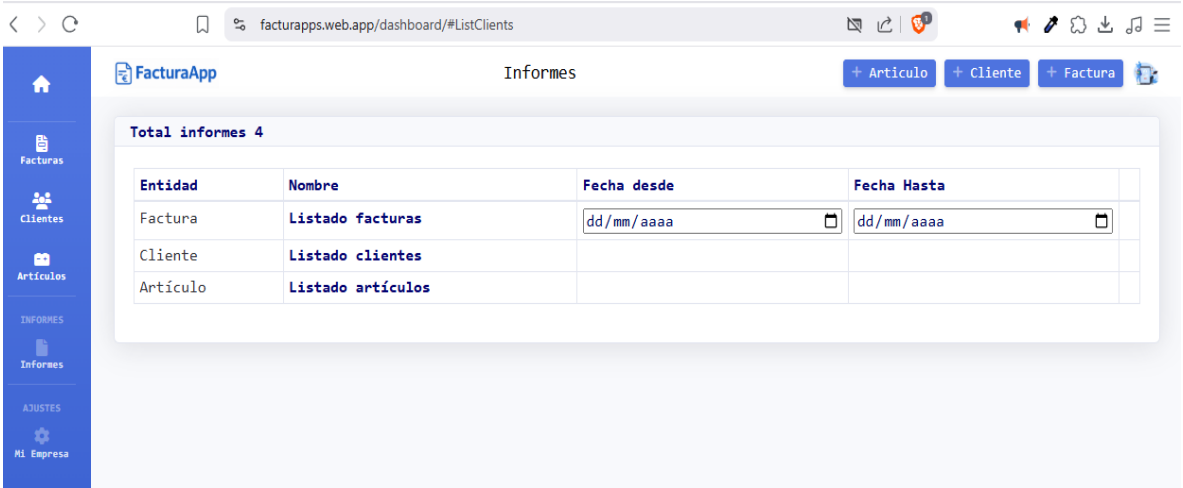
+ Artículo + Cliente + Factura

Total artículos 4

Número	Descripción	Precio unidad	IVA	Tipo
4	Desarrollo API personalizada	1000.00 €	21.00 %	norm
3	Configuracion Servidor	30.00 €	21.00 %	norm
2	Informe Power BI	3000.00 €	21.00 %	norm
1	Desarrollo App Personalizada	1000.00 €	21.00 %	norm

Caso de uso 5 – Ver historial de facturas

- Actor: Usuario
- Objetivo: Consultar facturas anteriores.
- Pasos:
 1. Accede a “Informes”.
 2. Seleccionamos el informe “Listado de facturas”.

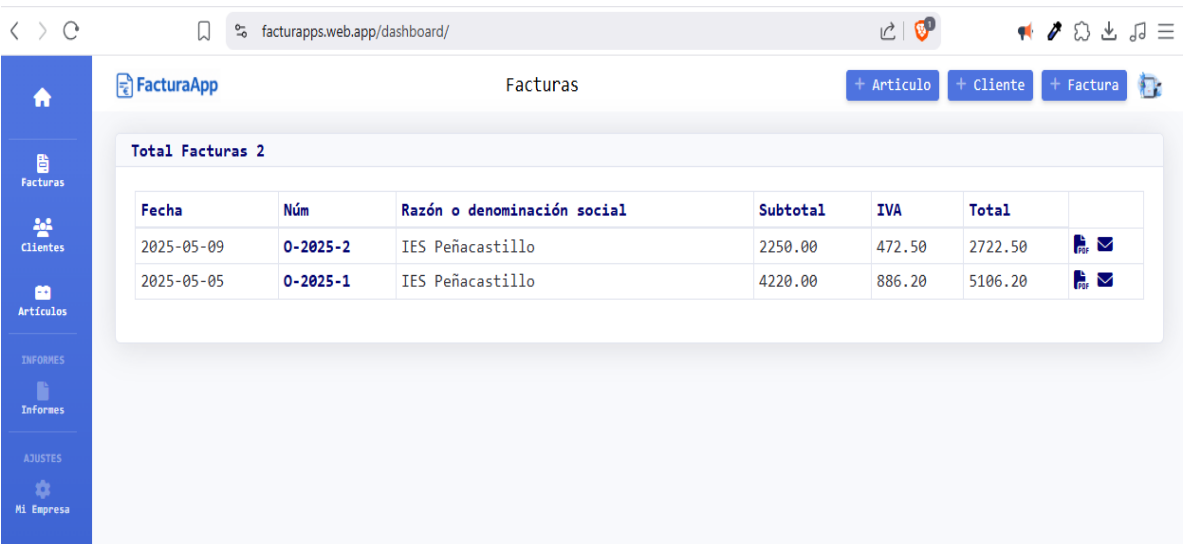


3. Descargarmos el excel con facturas existentes.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Fecha expedición	Número	Ejercicio	Tipo factura	Importe líneas	Importe IVA	Importe total	Observación	Apunta a factura	Cliente número	CIF NIF	Razon social - nombre
2	2025-05-09	0-2025-2	2025	FACTURA	2250	472,5	2722,5	Sin			1 IES Peñacastillo	IES Peñacastillo
3	2025-05-05	0-2025-1	2025	FACTURA	4220	886,2	5106,2	El trabajo ejecutado antes de tiempo			1 IES Peñacastillo	IES Peñacastillo
4												
5												
6												
7												
8												
9												

7.g. Mapa de navegación y estructura de menús

La interfaz de FacturaApp se estructura mediante un sistema de navegación claro y consistente, compuesto por un **menú lateral izquierdo** permanente y una **barra superior de acciones rápidas**. Esto permite al usuario acceder a todas las funcionalidades de forma directa desde cualquier parte del sistema.



Estructura del menú lateral (izquierda)

Este menú actúa como barra de navegación principal. Incluye accesos a las secciones clave del sistema:

- Inicio

- Facturas
- Clientes
- Artículos
- Informes
- Mi Empresa (Ajustes)

Este menú permanece visible y accesible en todo momento, incluso al cambiar de sección, lo que mejora la usabilidad y reduce los pasos necesarios para navegar entre áreas.

Acciones rápidas (barra superior)

En la parte superior derecha de la interfaz se encuentran accesos rápidos para la creación de registros:

- + Artículo
- + Cliente
- + Factura

Estos botones están siempre disponibles, permitiendo al usuario generar nueva información desde cualquier sección del sistema sin necesidad de regresar al panel principal.

8. Planificación

a. Diagrama de Gantt (actividades y tiempos)

En el desarrollo del proyecto FacturaApp, se han definido y planificado las siguientes actividades principales:

- **Análisis del contexto:** Investigación sobre la necesidad de una herramienta gratuita y sencilla de facturación para autónomos y pymes. Evaluación de herramientas existentes, tecnologías viables y necesidades del usuario final.
- **Diseño del sistema:** Definición de las entidades y relaciones de base de datos, diseño de la arquitectura frontend-backend, y selección de tecnologías (Django, SQLite, Firebase).
- **Modelado de datos:** Creación del modelo entidad-relación y posterior estructura física de base de datos (tablas, claves foráneas, tipos de datos).
- **Implementación del backend:** Desarrollo de la API en Django, configuración de la base de datos y lógica para generar facturas, clientes, artículos, etc.
- **Implementación del frontend:** Desarrollo de la interfaz HTML/CSS/JS para interacción con el usuario. Conexión con el backend y despliegue en Firebase.

- Pruebas funcionales: Revisión de las funcionalidades básicas: creación de facturas, registro de clientes, gestión de artículos, generación de PDFs.
- Gestión de errores y mejoras: Corrección de fallos detectados durante las pruebas, y mejoras en la experiencia de usuario (UX).
- Tutorías: Reuniones periódicas con el tutor para orientación técnica y revisión del progreso.
- Entregas parciales: Fases intermedias de entrega de documentación, diagramas y primeras versiones funcionales.
- Documentación y memoria: Redacción y revisión de la memoria del proyecto, incluyendo anexos técnicos, capturas e informes.
- Entrega final y defensa del proyecto.

Duración estimada por actividad

Actividad	Duración estimada
Análisis del contexto	1 semana
Diseño del sistema	1 semana
Modelado de datos	1 semana
Implementación backend	3 semanas
Implementación frontend	2 semanas
Pruebas funcionales	1 semana
Gestión de errores y mejoras	2 semanas
Entregas parciales	3 entregas
Documentación y memoria	2 semanas
Entrega final y defensa	1 semana

8.b. Recursos y logística

1. Recursos materiales

Para el desarrollo del proyecto FacturaApp se han utilizado los siguientes recursos físicos, tecnológicos y digitales:

Mobiliario

- Mesa de escritorio amplia, para poder trabajar con portátil y documentos a la vez.
- Silla de oficina ergonómica con soporte lumbar.
- Lámpara LED con brazo articulado para sesiones nocturnas.
- Cuaderno y archivador físico para organización de esquemas y bocetos iniciales.

Equipos informáticos

- Ordenador portátil: ASUS VivoBook / HP / Lenovo con procesador i5, 16GB de RAM y SSD.
- Monitor externo: 24" Full HD para facilitar la visualización de código y diagramas.
- Teléfono móvil Android: utilizado para pruebas de accesibilidad en móvil.
- Impresora doméstica (uso opcional para pruebas de facturas PDF).

Software utilizado

- Visual Studio Code: para el desarrollo de código HTML, CSS, JavaScript y Python (backend).
- Django (Python): framework principal del backend.
- SQLite: sistema gestor de base de datos ligero e integrado.
- Firebase Hosting: para desplegar el frontend de la aplicación.
- PythonAnywhere: para alojar el backend (API REST).
- Google Chrome / Firefox: navegadores para pruebas de interfaz y rendimiento.
- draw.io para la creación de diagramas (ER, Gantt, navegación, etc.).
- LibreOffice Writer: para la redacción de la memoria.
- Paint.NET: para edición básica de capturas de pantalla.

Antivirus y seguridad

- Windows Defender (integrado) y revisión periódica con Malwarebytes Free.
- Buenas prácticas de contraseñas y copia de seguridad de archivos locales y en nube.

Conexión a Internet

- Proveedor: Digi.
- Velocidad contratada: 300Mb simétricos (fibra óptica).
- Router Wi-Fi doble banda, conexión estable para pruebas y despliegue.

Alojamiento y servicios web

- Firebase Hosting del frontend HTML/CSS 11€/Mes.
- PythonAnywhere (plan gratuito para desarrolladores): Hosting del backend Django 5€/Mes.

- GitHub: Repositorio para control de versiones del código.
- Google Drive: copia de seguridad de archivos del proyecto y memoria.

Todo el desarrollo se ha realizado con herramientas mayoritariamente gratuitas o de código abierto, lo que demuestra que es posible llevar a cabo un proyecto funcional y profesional sin necesidad de realizar inversiones elevadas. El uso de entornos de despliegue reales (Firebase y PythonAnywhere) ha permitido simular un entorno de producción accesible desde cualquier dispositivo.

Recursos humanos, Equipo del proyecto:

Alexey Suzdalenko

- Técnico Superior en Desarrollo de Aplicaciones Web.
- Desarrollador principal y responsable integral del proyecto FacturaApp.
- Áreas de responsabilidad:
 - Análisis y diseño del sistema.
 - Desarrollo backend en Django.
 - Desarrollo frontend en HTML/CSS/JS.
 - Modelado y diseño de base de datos.
 - Gestión del despliegue (Firebase / PythonAnywhere).
 - Documentación técnica y memoria del proyecto.
 - Pruebas funcionales y control de calidad.

Colaboración externa puntual (*opcional*)

- Apoyo informal de compañeros o tutor académico durante sesiones de revisión técnica.
- Revisión externa de redacción y estructura del documento por parte de profesores o evaluadores.

Logística:

Registro de la Propiedad Intelectual (*opcional para fases futuras reales*)

- Coste estimado: 150,45 €
- Objetivo: Proteger el código fuente y diseño visual de la herramienta.
- Renovación: Indefinida (una vez registrado).

Registro de marca (nombre FacturaApp)

- Entidad: Oficina Española de Patentes y Marcas
- Coste estimado: 150,45 €
- Objetivo: Protección jurídica del nombre comercial para uso público.
- Duración: 10 años renovables
- Pago: Plataforma electrónica oficial

Recursos adicionales utilizados

- Trello / Google Keep: Organización de tareas y planificación personal.
- Google Sheets: Control de tiempos, gastos y estructura de tablas.
- GitHub: Control de versiones y colaboración técnica en línea.
- draw.io: Diseño de diagramas E/R, navegación, Gantt y clases.

8.c. Cronología del desarrollo (timeline)

La siguiente cronología recoge las principales fases del desarrollo de **FacturaApp**, desde su concepción inicial hasta su entrega final. El proyecto ha sido ejecutado de forma progresiva, combinando la planificación, la codificación, las pruebas y la documentación a lo largo de varios meses.

Mes / Semana	Actividad realizada
Marzo - Semana 1	Inicio formal del proyecto. Estudio de mercado: herramientas de facturación actuales, ventajas y carencias. Identificación de necesidades del usuario autónomo.
Marzo - Semana 2	Análisis de la competencia: comparación de herramientas como Billin, Contasimple, FacturaScripts, etc. Evaluación de tecnologías posibles (PHP vs Python) para el backend.
Marzo - Semana 3	Elección del stack tecnológico: Django (Python) + SQLite + HTML/CSS/JS. Diseño de la arquitectura base (frontend-backend separado). Primeros bocetos de la base de datos.
Marzo - Semana 4	Diseño definitivo de la base de datos: entidades, relaciones, claves. Creación de los modelos Django. Inicio del desarrollo backend (views, lógica de facturación).
Abril - Semana 1	Desarrollo de la interfaz frontend en HTML y JS. Inicio de la integración con el backend mediante peticiones Fetch.
Abril - Semana 2	Generación de facturas en PDF. Pruebas iniciales. Validación de la lógica de negocio y del flujo de datos cliente-servidor.
Abril - Semana 3	Desarrollo del sistema de informes y pruebas funcionales. Corrección de errores y mejoras visuales (UX/UI).

Mes / Semana	Actividad realizada
Abril - Semana 4	Entrega parcial del proyecto. Revisión con el tutor. Incorporación de sugerencias. Comienzo del despliegue en Firebase y PythonAnywhere.
Mayo - Semana 1-3	Despliegue completo y funcional del sistema. Redacción de la memoria técnica. Inserción de diagramas y capturas. Preparación de la defensa oral.

9. Implementación

a. Lenguajes, tecnologías y herramientas utilizadas

Los lenguajes y tecnologías utilizadas en el desarrollo del proyecto **FacturaApp** se han seleccionado por su versatilidad, facilidad de despliegue y compatibilidad con plataformas gratuitas de hosting. A continuación se detallan:

Lenguajes de programación

- Python
Utilizado en el backend con el framework Django. Permite manejar la lógica del servidor, la conexión con la base de datos y la generación de facturas en PDF, envío de Email.
- JavaScript
Utilizado en el frontend para capturar eventos, realizar validaciones de formularios y ejecutar peticiones AJAX al backend.
- HTML5
Estructura base de todas las páginas del proyecto.
- CSS3
Usado para el diseño visual del sitio, incluyendo colores, espaciados, y maquetación.

Base de datos

- SQLite
Elegida por su facilidad de integración con Django y por no requerir instalación de un servidor. Almacena clientes, artículos, facturas, usuarios y documentos.

Frameworks y bibliotecas

- Bootstrap 4
Framework CSS utilizado para crear un diseño responsive, adaptable a móviles, tablets y escritorio. Se ha usado para:
 - Menús y navegación lateral.
 - Formularios y botones.
 - Layout con sistema de columnas.
 - Componentes visuales como alertas, tarjetas y modales.

- **FontAwesome**
Biblioteca de iconos empleada para mejorar la apariencia de botones y menús (iconos de factura, cliente, PDF, etc.).

Otras herramientas y recursos

- **Django PDF / xhtml2pdf**
Para generar facturas en formato PDF directamente desde el backend, incluyendo datos del cliente, artículos y totales con IVA.
- **Firebase Hosting**
Usado para desplegar el frontend de la aplicación (páginas HTML/CSS/JS).
- **PythonAnywhere**
Hosting para Python/Django. Permite alojar el backend, la base de datos SQLite y las rutas API necesarias.
- **GitHub**
Repositorio para control de versiones, seguimiento de cambios y copia de seguridad del proyecto.
- **draw.io**
Herramienta usada para diseñar diagramas de base de datos, navegación, Gantt y ER.

Durante el desarrollo, se ha personalizado Bootstrap para adaptarlo a la identidad visual del proyecto. Se han utilizado clases como `rounded`, `mt-4`, `card`, `btn-primary`, `container`, entre otras, para lograr un diseño limpio, profesional y funcional.

Además, los formularios HTML fueron validados tanto en el cliente (JavaScript) como en el servidor (Django), garantizando la consistencia de los datos introducidos.

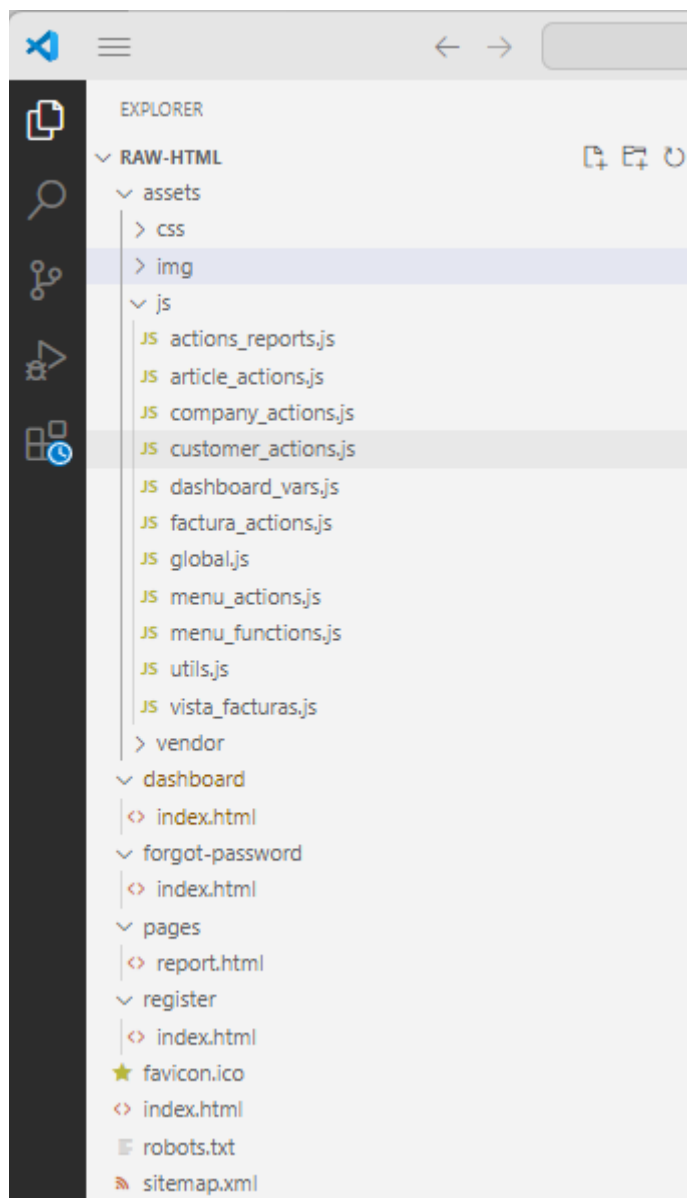
8.b. Organización del proyecto (frontend / backend)

El desarrollo de FacturaApp se ha organizado bajo una arquitectura frontend-backend desacoplada, en la que cada parte del sistema se encuentra claramente diferenciada y alojada en plataformas distintas. Esta separación facilita el mantenimiento, el escalado futuro y el despliegue independiente.

Frontend (Interfaz de usuario)

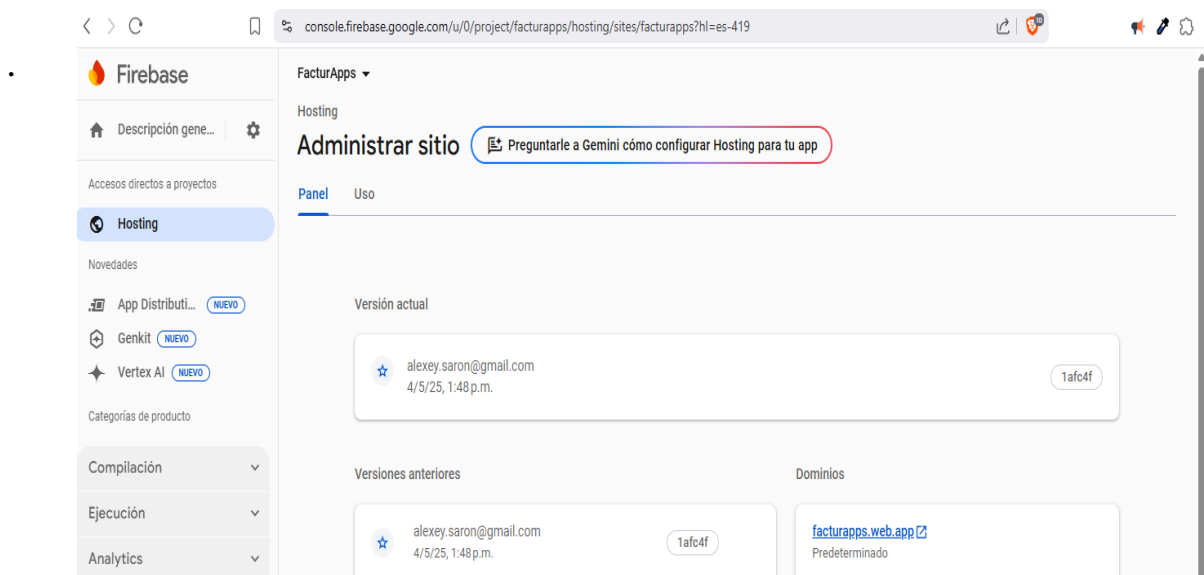
- **Tecnologías utilizadas:**
 - HTML5, CSS3, JavaScript (puro), Bootstrap 5.
 - FontAwesome para iconos.
- **Estructura:**
 - Páginas organizadas por funcionalidad:
 - `index.html`, `dashboard/index.html`, `forgot-password/index.html`, `register/index.html`

- Directorios organizados por tipo de recurso:
 - /assets/css/ para estilos.
 - /assets/js/ para scripts organizados por módulo (cliente, factura, artículo...).
 - /assets/img/ para logotipos e iconos.
- Menú lateral fijo y barra superior con botones de acción rápida (+ Cliente, + Factura, etc.).



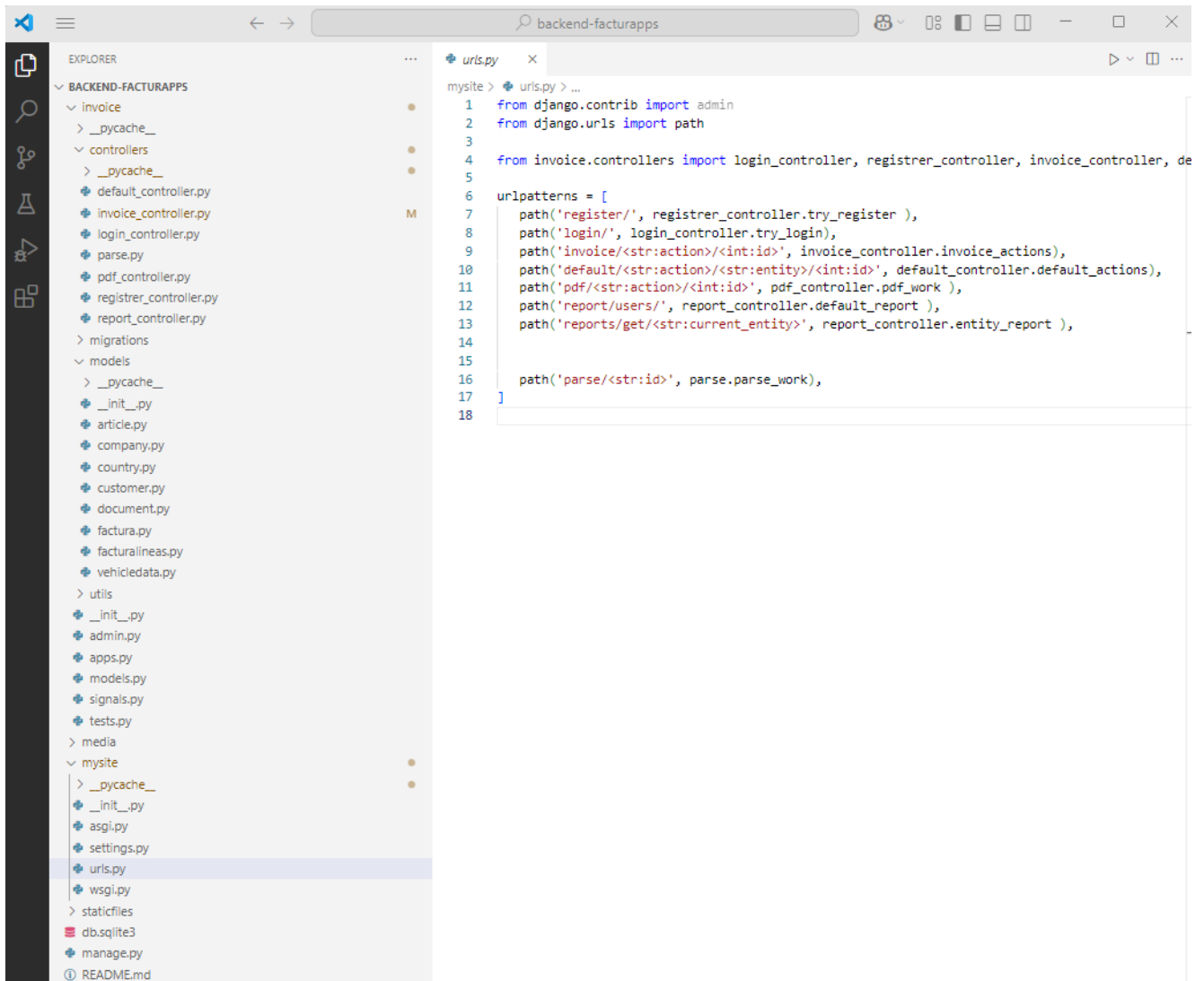
- Responsabilidades del frontend:
 - Presentar los datos recibidos del backend de forma clara y organizada.
 - Permitir la interacción del usuario con formularios y botones.
 - Enviar peticiones HTTP al backend mediante fetch().

- Validar campos antes del envío de datos.
- Despliegue:
 - Alojado en Firebase Hosting, accesible públicamente desde cualquier navegador.



Backend (Lógica de negocio y datos)

- Tecnología principal:
 - Django (Python), con SQLite como base de datos.
- Estructura del backend:
 - models.py define todas las tablas: Empresa (Company), Cliente (Customer), Factura (Factura), Artículos (Article), etc.
 - controllers/ contiene la lógica de procesamiento de peticiones y generación de respuestas JSON.
 - views.py y URLs mapean rutas a funciones de respuesta.
 - Generación de facturas PDF mediante xhtml2pdf.



- Responsabilidades del backend:

- Recibir datos del frontend, validarlos y almacenarlos en base de datos.
- Procesar operaciones CRUD sobre clientes, artículos, facturas, etc.
- Calcular totales, IVAs y descuentos en cada factura.
- Servir los archivos PDF generados.
- Proteger el acceso a datos mediante autenticación básica.

- Despliegue:

- Alojado en PythonAnywhere, accesible como servicio web por el frontend.

The screenshot shows the PythonAnywhere web interface. At the top, there's a navigation bar with links: Dashboard, Consoles, Files, Web, Tasks, Databases. Below this, the user's profile 'simplefactura' is shown with a '4% full' quota indicator. The main area is divided into 'Directories' and 'Files' sections. The 'Directories' section lists folders like .cache/, .ipython/, .local/, .virtualenvs/, media/, mysite/, and static/. The 'Files' section lists files like .bashrc, .gitconfig, .profile, .pythonstartup.py, .viminfo, .vimrc, and README.txt, each with a download icon, a trash icon, a timestamp, and a size. At the bottom, there's an 'Upload a file' button with a note '100MiB maximum size'.

Comunicación Frontend-Backend

- Tipo de comunicación: `fetch()` con método POST o GET (según el caso).
- Formato de intercambio: JSON.
- Ejemplo de flujo:
 1. El usuario añade un cliente desde el frontend.
 2. Se envía una petición POST al backend con los datos.
 3. El backend guarda el cliente en SQLite y responde con éxito o error.
 4. El frontend actualiza dinámicamente la tabla de clientes.

9.c. Código fuente representativo

A continuación se presentan fragmentos de código clave que reflejan el funcionamiento interno del proyecto FacturaApp, tanto en el lado del cliente (frontend) como en el servidor (backend). Estos fragmentos muestran cómo se estructura la lógica de facturación, la interacción entre usuario y base de datos, y la generación de facturas en PDF.

Al lado de frontend se trata de creacion de lineas de factura y calculos de valores ivas, totales, descuentos este codigo se usa en la vista de creacion de la factura:

FacturaApp

Crear Factura

+ Artículo + Cliente + Factura

Tipo Factura

Tipo Ordinaria/Rectificativa/Abono

Factura Ordinaria

Datos cliente y tipo de factura

Número cliente 2 NIF CIF cliente A12345678

Razón o denominación social Banco Santander

Datos vehículo

Matrícula

Marca/Modelo/Kilómetros

Líneas de factura

Código Art.	Descripción Art.	Cantidad	Precio	% Dto.	Importe	IVA
3	Configuracion Servidor	1	3000,00	5	2850,00	21 %
	Apaptacion de Software	1	50000	10	45000,00	21 %
+ Línea	Mano de obra	10	22,00		220,00	21 %

Subtotal: 48070.00
IVA: 10094.70
Total: 58164.70

Trabajo en los servidores de Banco Santander

Crear Factura

en el código de frontend de esta vista tenemos:

```
async function traerDatosClientArtFacturas(){
  postRequest('default/get/cliente/0', new FormData()).then(listadoClientes => {
    if(listadoClientes.res.length == 0){
      showM('Para crear factura al menos tiene que tener 1 cliente creado previamente..', 'warning');
    }
    LISTADO_CLIENTES_FACTURAS = [];
    LISTADO_CLIENTES_FACTURAS = listadoClientes.res;
  });
  postRequest('default/get/articulo/0', new FormData()).then(listadoArticles => {
    if(listadoArticles.res.length == 0){
      showM('Para crear factura al menos tiene que tener 1 articulo creado previamente..', 'warning');
    }
    LISTADO_ARTICULOS_FACTURAS = [];
    LISTADO_ARTICULOS_FACTURAS = listadoArticles.res;
  });
  postRequest('default/get/empresa/0', new FormData()).then(empresaData => {
    if(empresaData.res.length == 0){
      showM('Para crear factura al menos rellene los datos de la empresa..', 'warning');
    }
    EMPRESA_FACTURAS = {};
    EMPRESA_FACTURAS = empresaData.res[0];
    document.getElementById('precioManoObra').value = EMPRESA_FACTURAS.price;
  });
}
```

```
function showFormInvoiceCreation(){
  LINE_COUNTER = 0;
  /* traer listado de clientes y articulos */
```

```

traerDatosClientArtFacturas());

pageTitle.innerText = 'Crear Factura'
let htmlTipoInvoice = '<select id="selectTypeInvoice" onchange="cambioTipoFacturaMano(event)">';
INVOICES_LIST.forEach(invoiceLine => { htmlTipoInvoice += `<option value="${invoiceLine.letter}">${invoiceLine.title}</option>`; });
htmlTipoInvoice += '</select>';
let htmlSkeleton = `<div class="container-fluid">
    <div class="row">
        <div class="col-lg-3">
            <div class="card shadow mb-3">
                <div class="card-header py-1">
                    <h6 class="m-0 font-weight-bold text-primary">Tipo Factura</h6>
                </div>
                <div class="card-body">
                    <div class="mt-0 mb-1 "><code>Tipo Ordinaria/Rectificativa/Abono</code></div>
                    ${htmlTipoInvoice}
                    <div id="espacioParaSelectTypeFactura"></div>
                </div>
            </div>
        </div>
        <div class="col-lg-5">
            <div class="card shadow mb-3">
                <div class="card-header py-1">
                    <h6 class="m-0 font-weight-bold text-primary">Datos cliente y tipo de factura</h6>
                </div>
                <div class="card-body">
                    <div class="row">
                        <div class="col-lg-3" style="display:none;">
                            <input type="text" id="clientIdDeveloper" disabled>
                        </div>
                        <div class="col-lg-6">
                            <div class="mt-0 mb-0"><code>Número cliente</code></div>
                            <input type="text" id="clientNumber" disabled>
                        </div>
                        <div class="col-lg-6">
                            <div class="mt-0 mb-0"><code>NIF CIF cliente</code></div>
                            <input type="text" id="inputNifCif" disabled>
                        </div>
                    </div>
                    <div class="mt-2 mb-1 "><code>Razón o denominación social</code></div>
                    <input type="text" id="autocomplete_input" placeholder="Nombre cliente"
oninput="handleInputClient(event)">
                    <div id="suggestions" class="suggestions-list"></div>
                </div>
            </div>
        </div>
        <div class="col-lg-4">
            <div class="card shadow mb-3">
                <div class="card-header py-1">
                    <h6 class="m-0 font-weight-bold text-primary">Datos vehículo</h6>
                </div>
                <div class="card-body">

```

```

        <div class="mt-0 mb-1 "><code>Matrícula</code></div>
        <input type="text" id="inputVehicleMatricula" value="">
        <div class="mt-2 mb-1 "><code>Marca/Modelo/Kilómetros</code></div>
        <input type="text" id="inputVehicleMarca" value="">
    </div>
</div>
</div>
</div>
<div class="row">
    <div class="col-lg-12">
        <div class="card shadow mb-3">
            <div class="card-header py-1">
                <h6 class="m-0 font-weight-bold text-primary">Líneas de factura</h6>
            </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-lg-1"><code>Código Art.</code></div>
                    <div class="col-lg-4"><code>Descripción Art.</code></div>
                    <div class="col-lg-1"><code>Cantidad</code></div>
                    <div class="col-lg-2"><code>Precio</code></div>
                    <div class="col-lg-1"><code>% Dto. </code></div>
                    <div class="col-lg-1"><code>Importe</code></div>
                    <div class="col-lg-1"><code>IVA</code></div>
                    <div class="col-lg-1"></div>
                </div>
                <div id="divContrainerNewLines">
                    <div class="row mt-1" id="deleteDivNumber0" data-line_counter="0">
                        <div class="col-lg-1"><input type="number" class="suzdal_none" id="idArt0"><input
type="number" disabled="" id="numberArt0"></div>
                        <div class="col-lg-4">
                            <input type="text" id="descriptionArt0" oninput="handleInputDescription(0, event)">
                            <div id="suggestionsArticles0" class="suggestions-list"></div>
                        </div>
                        <div class="col-lg-1"><input type="number" id="cantidadArt0" value="1"
oninput="invoiceCalculate()"></div>
                        <div class="col-lg-2"><input type="number" id="precioArt0"
oninput="invoiceCalculate()"></div>
                        <div class="col-lg-1"><input type="number" value="0" id="descuentoArt0"
oninput="invoiceCalculate()"></div>
                        <div class="col-lg-1"><input type="number" disabled="" id="totalArt0"></div>
                        <div class="col-lg-1" id="ivaArt0"><select class="factura_select"
oninput="invoiceCalculate()" id="selectIva0"><option value="21"> 21 % </option><option value="10"> 10 % </option><option value="4"> 4 %
</option><option value="0"> 0 % </option><option value="0EXENTO"> 0 EXENTO </option></select></div>
                        <div class="col-lg-1"> <i class="fa fa-trash" aria-hidden="true"
onclick="deleteThisDiv(0)"></i></div>
                    </div>
                </div>
            </div>
        </div>
        <div class="row mt-4">
            <div class="col-lg-1"><span class="simulate_link" onclick="idAddNewLine()"><i class="fa fa-
plus" aria-hidden="true"></i> Línea</span></div>
        </div>
    </div>
</div>

```

```

        <div class="col-lg-1"></div>
        <div class="col-lg-4"><input type="text" value="Mano de obra" disabled></div>
        <div class="col-lg-1"><input type="number" id="cantidadManoObra" value=""
oninput="invoiceCalculate()"></div>
        <div class="col-lg-2"><input type="number" id="precioManoObra"
oninput="invoiceCalculate()"></div>
        <div class="col-lg-1"><input type="number" value="" id="descuentoManoObra"
oninput="invoiceCalculate()"></div>
        <div class="col-lg-1"><input type="number" disabled id="totalManoObra"></div>
        <div class="col-lg-1"><select class="factura_select" id="ivaManoObra"
oninput="invoiceCalculate()"><option value="21"> 21 % </option><option value="10"> 10 % </option><option value="4"> 4 % </option><option
value="0"> 0 % </option><option value="0EXENTO"> 0 EXENTO </option></select></div>
        <div class="col-lg-1"></div>
    </div>
    <br>
    <div class="row mb-1"><div class="col-lg-10"></div><div class="col-lg-2">
        <table cellpadding="1"><tbody>
            <tr><td><code>Subtotal:</code></td><td class="align_right"><code
id="idSubTotal">0</code></td></tr>
            <tr><td><code>IVA:</code></td><td class="align_right"><code
id="idIvaTotal">0</code></td></tr>
            <tr><td><code>Total:</code></td><td class="align_right"><code
id="idTotalFactura">0</code></td></tr>
        </tbody></table></div>
    </div>
    <div class="row mb-1">
        <div class="col-lg-5"><textarea placeholder="Observaciones" class="obstextarea"
id="obstextareaid"></textarea></div>
    </div>
</div>
</div>
</div>
</div>
    <a href="#" class="btn btn-facebook" onclick="clickCreateInvoice()" id="idCreateInvoice"><i class="fas fa-save"
aria-hidden="true"></i> Crear Factura</a>
</div>`;

```

```

pageMainContent.innerHTML = htmlSkeleton;
}

```

```

function invoiceCalculate(){
    FACTURA_LINEAS = {lineas:[], manoObra:{}, factura:{}, cliente:{}, vehicle:{}, desglose:{}, observaciones:{}};
    let importeSubtotal = 0;
    let importeIvas    = 0;

    let divContrainerNewLines = document.getElementById('divContrainerNewLines');
    let childDivs = divContrainerNewLines.children;
    Array.from(childDivs).forEach((childDiv, index) => {
        let numLine    = childDiv.dataset.line_counter;
        let idArticle1 = document.getElementById('idArt'+numLine).value.trim();
        let description= document.getElementById('descriptionArt'+numLine).value.trim();
        let cantidad1  = document.getElementById('cantidadArt'+numLine).value.trim();
    });
}

```

```

    let precio1    = document.getElementById('precioArt'+numLine).value.trim();
    let descPorc   = document.getElementById('descuentoArt'+numLine).value.trim();
    if(descPorc == '') descPorc = 0;
    let imp1Bruto  = cantidad1 * precio1;
    let desc1Valor = descPorc / 100 * imp1Bruto;
    let importe1   = imp1Bruto - desc1Valor;
    importeSubtotal += importe1;
    document.getElementById('totalArt'+numLine).value = parseFloat(importe1).toFixed(2);
    let ivaType    = document.getElementById('selectIva'+numLine).value.trim();
    let ivaPorcent = document.getElementById('selectIva'+numLine).value.trim() == '0EXENTO' ? 0 :
document.getElementById('selectIva'+numLine).value.trim();
    let ivaVallin1 = ivaPorcent / 100 * importe1;
    importeIvas += ivaVallin1;
    FACTURA_LINEAS.lineas.push({numLine, idArticle1, description, cantidad1, precio1, descPorc, imp1Bruto, desc1Valor, importe1,
ivaPorcent, ivaVallin1, ivaType});
});
let canridadManoObra = document.getElementById('cantidadManoObra').value.trim();
if(canridadManoObra == '') canridadManoObra = 0;
let precioManoObra   = document.getElementById('precioManoObra').value.trim();
let ivaPorcentManoOb = document.getElementById('ivaManoObra').value.trim() == '0EXENTO' ? 0 :
document.getElementById('ivaManoObra').value.trim();
let tipoIvaManoObra  = document.getElementById('ivaManoObra').value.trim();
let porcentajeDesMOB = document.getElementById('descuentoManoObra').value.trim();

/* linea mano de obra */
let importeBrutoManoObra = canridadManoObra * precioManoObra;
let descManoObr          = porcentajeDesMOB / 100 * importeBrutoManoObra;
let importeManoObra      = importeBrutoManoObra - descManoObr;
document.getElementById('totalManoObra').value = parseFloat(importeManoObra).toFixed(2);

document.getElementById('idSubTotal').innerText = parseFloat(importeManoObra + importeSubtotal).toFixed(2);
let valorIvaManoObra = ivaPorcentManoOb / 100 * importeManoObra
document.getElementById('idIvaTotal').innerText = parseFloat(valorIvaManoObra + importeIvas).toFixed(2);
let valorTotalFactura = valorIvaManoObra + importeManoObra + importeSubtotal + importeIvas;
document.getElementById('idTotalFactura').innerText = parseFloat(valorTotalFactura).toFixed(2);

FACTURA_LINEAS.manoObra = {canridadManoObra, precioManoObra, importeBrutoManoObra, descManoObr, importeManoObra, ivaPorcentManoOb,
valorIvaManoObra, tipoIvaManoObra};
FACTURA_LINEAS.factura = { idSubTotal: importeManoObra + importeSubtotal, idIvaTotal: valorIvaManoObra + importeIvas, idTotalFactura:
valorTotalFactura };
FACTURA_LINEAS.desglose = DESGLOSE;
}

function deleteThisDiv(x){
    let divDelete = document.getElementById('deleteDivNumber'+x);
    if(divDelete) divDelete.remove();
    invoiceCalculate();
}

function idAddNewLine(){
    LINE_COUNTER++;
    let divContrainerNewLines = document.getElementById('divContrainerNewLines');

```

```

let tempDiv = document.createElement('div');
let contenidoHTML = `<div class="row mt-1" id="deleteDivNumber${LINE_COUNTER}" data-line_counter="${LINE_COUNTER}">
    <div class="col-lg-1"><input type="number" class="suzdal_none" id="idArt${LINE_COUNTER}"><input type="number"
disabled id="numberArt${LINE_COUNTER}"></div>
    <div class="col-lg-4">
        <input type="text" id="descriptionArt${LINE_COUNTER}" oninput="handleInputDescription(${LINE_COUNTER}, event)">
        <div id="suggestionsArticles${LINE_COUNTER}" class="suggestions-list"></div>
    </div>
    <div class="col-lg-1"><input type="number" id="cantidadArt${LINE_COUNTER}" value="1"
oninput="invoiceCalculate()"></div>
    <div class="col-lg-2"><input type="number" id="precioArt${LINE_COUNTER}" oninput="invoiceCalculate()"></div>
    <div class="col-lg-1"><input type="number" value="0" id="descuentoArt${LINE_COUNTER}"
oninput="invoiceCalculate()"></div>
    <div class="col-lg-1"><input type="number" disabled id="totalArt${LINE_COUNTER}"></div>
    <div class="col-lg-1" id="ivaArt${LINE_COUNTER}"><select class="factura_select" oninput="invoiceCalculate()"
id="selectIva${LINE_COUNTER}"><option value="21"> 21 % </option><option value="10"> 10 % </option><option value="4"> 4 % </option><option
value="0"> 0 % </option><option value="0EXENTO"> 0 EXENTO </option></select></div>
    <div class="col-lg-1"> <i class="fa fa-trash" aria-hidden="true" onclick="deleteThisDiv($
${LINE_COUNTER})"></i></div>
    </div>`;
tempDiv.innerHTML = contenidoHTML;
let newElement = tempDiv.firstElementChild;
divContrainerNewLines.appendChild(newElement);
};

function returnIvaType(num, ivatype, iva){
    let selectedType = '';
    let htmlIva = `<select id="selectIva${num}" oninput="invoiceCalculate()" class="factura_select">`;
    IVAS_LIST.forEach(item => {
        if(ivatype == 'norm' && item.title == iva){ selectedType = 'selected'; }
        if(ivatype == 'exento' && item.title == '0EXENTO'){ selectedType = 'selected'; }
        htmlIva += `<option value="${item.title}" ${selectedType}>${item.percentage}</option>`;
        selectedType = '';
    });
    htmlIva += `</select>`;
    return htmlIva;
}

function handleInputDescription(num, event){
    let query = event.target.value.toLowerCase();
    let suggestionsArticles = document.getElementById('suggestionsArticles'+num);
    suggestionsArticles.innerHTML = '';
    if(query){
        let i = 0;
        let queryWords = query.split(' ');
        let filteredOptions = LISTADO_ARTICULOS_FACTURAS.filter(option => queryWords.every(word =>
String(option.description).toLowerCase().includes(word)));
        if (filteredOptions.length > 0) {
            suggestionsArticles.style.display = 'block';
            for(y = 0; y < filteredOptions.length; y++){
                let option = filteredOptions[y];
                if(i < 11){

```

```

        let suggestionItem = document.createElement('div');
        suggestionItem.classList.add('suggestion_item');
        suggestionItem.textContent = option.description;
        suggestionItem.addEventListener('click', function() {
            document.getElementById('idArt'+num).value = option.id;
            document.getElementById('numberArt'+num).value = option.artcode;
            document.getElementById('descriptionArt'+num).value = option.description;
            document.getElementById('precioArt'+num).value = option.price;
            document.getElementById('ivaArt'+num).innerHTML = returnIvaType(num, option.ivatype, option.iva);
            suggestionsArticles.innerHTML = '';
            suggestionsArticles.style.display = 'none';
            invoiceCalculate();
        });
        suggestionsArticles.appendChild(suggestionItem);
    } else {
        break;
    }
    i++;
};
}

invoiceCalculate();
document.getElementById('idArt'+num).value = ''; // he seleccionado el articulo, pero le cambio la descripcion
document.getElementById('numberArt'+num).value = '';
}

```

```

function handleInputClient(event){
    let query = event.target.value.toLowerCase();
    let suggestionsContainer = document.getElementById('suggestions');
    suggestionsContainer.innerHTML = '';
    if (query) {
        let i = 0;
        let filteredOptions = LISTADO_CLIENTES_FACTURAS.filter(option => option.razon.toLowerCase().includes(query));
        if (filteredOptions.length > 0) {
            suggestionsContainer.style.display = 'block';
            for(y = 0; y < filteredOptions.length; y++){
                let option = filteredOptions[y];
                if(i < 11){
                    let suggestionItem = document.createElement('div');
                    suggestionItem.classList.add('suggestion_item');
                    suggestionItem.textContent = option.razon;
                    suggestionItem.addEventListener('click', function() {
                        document.getElementById('clientIdDeveloper').value = option.id;
                        document.getElementById('clientNumber').value = option.clientcode;
                        document.getElementById('inputNifCif').value = option.cif_nif;
                        document.getElementById('autocomplete_input').value = option.razon;
                        suggestionsContainer.innerHTML = '';
                        suggestionsContainer.style.display = 'none';
                    });
                    suggestionsContainer.appendChild(suggestionItem);
                } else {
                    break;
                }
            }
        }
    }
}

```



```

        }
        i++;
    };
}

document.getElementById('clientIdDeveloper').value = '';
document.getElementById('clientNumber').value = '';
document.getElementById('inputNifCif').value = '';
}

function clickCreateInvoice(){
    let tipo_factura = document.getElementById('selectTypeInvoice').value.trim();
    let name_factura = '';
    INVOICES_LIST.forEach(item => { if(item.letter == tipo_factura){ name_factura = item.name_factura; }});
    let apunta_facturaValor = document.getElementById('selectFacturaRectificar');
    if(apunta_facturaValor && apunta_facturaValor.value) { apunta_facturaValor = apunta_facturaValor.value.trim(); } else { apunta_facturaValor = ''; }

    let clientIdDeveloper = document.getElementById('clientIdDeveloper').value.trim();
    let clientNumber = document.getElementById('clientNumber').value.trim();
    let clienteRazon = document.getElementById('autocomplete_input').value.trim();
    if(!clientIdDeveloper || !clientNumber || !clienteRazon) { showM('Cliente no definido', 'warning'); return; }
    let inputVehicleMatricula = document.getElementById('inputVehicleMatricula').value.trim();
    let inputVehicleMarca = document.getElementById('inputVehicleMarca').value.trim();
    let obstextareaid = document.getElementById('obstextareaid').value.trim()

    if(!FACTURA_LINEAS || !FACTURA_LINEAS.lineas || FACTURA_LINEAS.lineas.length == 0) { showM('Añade líneas en la factura', 'warning');
return; }
    let error = false;
    FACTURA_LINEAS.lineas.forEach(linea => {
        if(!linea.description || linea.description.length < 3){ showM('Descripcion del artículo', 'warning'); error = true; return; }
        if(!linea.cantidad1 || linea.cantidad1 == 0 || linea.cantidad1.trim() == ''){ showM('Cantidad del artículo', 'warning'); error = true;
return; }
        if(!linea.precio1 || linea.precio1 == 0 || linea.precio1.trim() == ''){ showM('Precio del artículo', 'warning'); error = true;
return; }
    });
    if(error == true) { return; }
    FACTURA_LINEAS.cliente = {clientIdDeveloper, clientNumber, clienteRazon };
    FACTURA_LINEAS.vehicle = {inputVehicleMatricula, inputVehicleMarca};
    FACTURA_LINEAS.factura.tipo_factura = tipo_factura; FACTURA_LINEAS.factura.name_factura = name_factura;
    FACTURA_LINEAS.factura.apunta_factura = apunta_facturaValor;
    FACTURA_LINEAS.observaciones = {obstextareaid}

    if(FACTURA_CREATION_CLICKED) { return; }
    FACTURA_CREATION_CLICKED = true;

    invoicePutRequest('invoice/put/0', FACTURA_LINEAS).then(response => {
        if(response && response.status == 'ok' && response.factura_id && response.factura_id > 0){
            DEFAULT_ENTITY = {name:'factura', title:'Facturas'};
            defaultController(DEFAULT_ACTION, DEFAULT_ENTITY, 0);
            showCustomPDF(response.factura_id, 'sent_email_action')
        } else {
            showM('Error al crear la factura..', 'error');

```

```

    }
    document.getElementById('idCreateInvoice').innerHTML = '<i class="fas fa-save" aria-hidden="true"></i> Grabando..';
  });
}

async function invoicePutRequest(url, data){
  try {
    data.credentials = {};
    data.credentials.company_id = window.localStorage.getItem('company_id');
    data.credentials.email      = window.localStorage.getItem('email');
    data.credentials.cif        = window.localStorage.getItem('cif');
    data.credentials.uid        = window.localStorage.getItem('uid');
    data.credentials.password   = window.localStorage.getItem('password');

    let uri = HTTP_URL + url;
    let response = await fetch(uri, {method: 'POST', headers: {'Content-Type': 'application/json'}, body: JSON.stringify(data) });

    let responseData = await response.json();
    if(responseData && responseData.status == 'error'){
      setTimeout(() => { showM(responseData.message, 'warning'); }, 1000);
    }
    return responseData;
  } catch (error) {
    showM('Error 11 ' + error, 'error');
    console.error('There was a problem with the fetch operation:', error);
  }
}

function cambioTipoFacturaMano(event){
  let espacioParaSelectTypeFactura = document.getElementById('espacioParaSelectTypeFactura');
  let tipoFactura = document.getElementById(event.target.id).value.trim();

  if(tipoFactura == 'R'){
    let htmlOptions = '';
    LISTADO_FACTURAS.forEach(factura => {
      htmlOptions += `<option value="${factura.serie_fact}" ${factura.fecha_expedicion}"> ${factura.serie_fact} $
{factura.fecha_expedicion} </option>`;
    });
    let htmlData = `<div class="mt-1 mb-1 ">
      <code>Numero factura a rectificar</code>
      <select id="selectFacturaRectificar">
        ${htmlOptions}
      </select>
    </div>`;
    espacioParaSelectTypeFactura.innerHTML = htmlData;
  } else {
    espacioParaSelectTypeFactura.innerHTML = '';
  }
}

```

Al pulsar el boton se ejecuta la accion peticion fetch hacia backend donde la logica de negocio en python crea nuevas lineas en la tabla facturas y lineasFacturas, el código en si de Python que crea nueva factura desde esta vista, empieza en el controlador:

```
path('invoice/<str:action>/<int:id>', invoice_controller.invoice_actions),
```

que llama siguiente funcion, que se ejecuta y devuelve el JSON indicando que se ha creado nueva factura:

```
def invoice_actions(request, action, id):
    if request.body:
        try:
            data = json.loads(request.body.decode('utf-8'))
        except json.JSONDecodeError:
            return json_suzdal({'status': 'error', 'message': 'Cuerpo de la solicitud no es JSON válido'})
    else:
        return json_suzdal({'status': 'error', 'message': 'Cuerpo de la solicitud vacío'})

    auth_status, company = user_auth(request, data)
    if auth_status is None or company is None:
        return json_suzdal({'login': False, 'status': 'error', 'message': 'Usuario no esta logeado'})

    desglose = data['desglose']
    ejercicio = str(datetime.now().strftime('%Y')).strip()
    tipo_factura = str(data['factura']['tipo_factura']).strip()
    lineas = data['lineas']
    if len(lineas) == 0:
        return json_suzdal({'status': 'error', 'message': 'Factura sin lineas'})

    try:
        customer = Customer.objects.get(id=data['cliente']['clientIdDeveloper'], clientcode=data['cliente']['clientNumber'],
company_id=company['id'])

        factura = Factura.objects.create(company_id=company['id'], tipo_factura=tipo_factura, ejercicio=ejercicio)
        document = Document.objects.filter(company_id=company['id'], description=tipo_factura, ejercicio=ejercicio).values('value').first()
        factura.name_factura = str(data['factura']['name_factura']).strip()
        factura.apunta_factura = str(data['factura']['apunta_factura']).strip()
        factura.numero = document['value']
        factura.serie_fact = f"{tipo_factura}-{ejercicio}-{factura.numero}"
        factura.serie_fact_unique = f"{tipo_factura}-{ejercicio}-{factura.numero}-{company['id']}"
        factura.fecha_expedicion = fecha_expedicion()
        factura.vencimiento = get_time_11days()

        factura.customer_id = customer.id
        factura.customer_num = customer.clientcode
        factura.receptor_company_name = customer.razon

        SUBTOTAL_FACTURA = 0
        IMP_IVAS_FACTURA = 0
        TOTAL_FACTURA = 0
        LINEAS_FACTURA = []

        # Base Imponible = Precio del artículo × Cantidad de artículos
```

```

for linea in lineas:
    description = str(linea.get('description', 'none')).strip()
    idArticle1 = str(linea.get('idArticle1', '')).strip()
    precio1 = float(linea.get('precio1', 0))
    cantidad1 = float(linea.get('cantidad1', 0))
    descPorc = float(linea.get('descPorc', 0))
    ivaPorcent = float(linea.get('ivaPorcent', 0))
    ivaTypeStr = str(linea.get('ivaType', '0'))

    if idArticle1.isdigit(): # Comprobar si es un número válido
        articulo_current = Article.objects.filter(id=idArticle1, company_id=company['id']).first()
    else:
        art_created, articulo_current = factura_new_article(description, company['id'], precio1, ivaTypeStr, ivaPorcent)
        if art_created is None or articulo_current is None:
            return json_suzdal({'status':'error', 'message':'Error al crear artículo nuevo'})

    importe_inicio = cantidad1 * precio1
    valor_descuento = descPorc / 100 * importe_inicio
    importe_con_descuento = importe_inicio - valor_descuento
    SUBTOTAL_FACTURA += importe_con_descuento
    valor_iva = ivaPorcent / 100 * importe_con_descuento
    IMP_IVAS_FACTURA += valor_iva
    importe_final = importe_con_descuento + valor_iva
    TOTAL_FACTURA += importe_final

    for d in desglose:
        if str(d['iva']) == ivaTypeStr:
            d['base_imponible'] += importe_con_descuento
            d['valor_iva'] += valor_iva
            d['total_con_iva'] += d['base_imponible'] + d['valor_iva']

    linea_factura = {'invoice_id':0, 'serie': '', 'company_id':company['id'], 'article_id':articulo_current.id,
'artcle_num':articulo_current.artcode, 'artcle_name':articulo_current.description, 'cantidad':cantidad1, 'precio':precio1,
'descuento':descPorc, 'iva_porcent':ivaPorcent, 'iva_type':ivaTypeStr}
    LINEAS_FACTURA.append(linea_factura)

# ahora el calculo de mano de obra
canridadManoObra = float(data['manoObra']['canridadManoObra'])
precioManoObra = float(data['manoObra']['precioManoObra'])
descManoObr = float(data['manoObra']['descManoObr'])
ivaPorcentManoOb = float(data['manoObra']['ivaPorcentManoOb'])
tipoIvaManoObra = str(data['manoObra']['tipoIvaManoObra'])

# cuando existe mano de obra
if canridadManoObra > 0 and precioManoObra > 0:
    importe_inicio_mo = canridadManoObra * precioManoObra
    valor_descuento_mo = descManoObr / 100 * importe_inicio_mo
    importe_con_descuento_mo = importe_inicio_mo - valor_descuento_mo
    SUBTOTAL_FACTURA += importe_con_descuento_mo
    valor_iva_mo = ivaPorcentManoOb / 100 * importe_con_descuento_mo
    IMP_IVAS_FACTURA += valor_iva_mo
    importe_final_mo = importe_con_descuento_mo + valor_iva_mo

```

```

TOTAL_FACTURA          += importe_final_mo

linea_factura = {'invoice_id':0, 'serie': '', 'company_id':company['id'], 'article_id':0, 'article_num':0, 'article_name':'Mano
de obra', 'cantidad':canridadManoObra, 'precio':precioManoObra, 'descuento':descManoObr, 'iva_porcent':ivaPorcentManoOb,
'iva_type':tipoIvaManoObra}

LINEAS_FACTURA.append(linea_factura)

for d in desglose:
    if str(d['iva']) == tipoIvaManoObra:
        d['base_imponible'] += importe_con_descuento_mo
        d['valor_iva'] += valor_iva_mo
        d['total_con_iva'] += d['base_imponible'] + d['valor_iva']

factura.ivas_desglose = json.dumps(desglose)
factura.subtotal      = SUBTOTAL_FACTURA
factura.importe_ivas  = IMP_IVAS_FACTURA
factura.total         = TOTAL_FACTURA
factura.total2        = SUBTOTAL_FACTURA + IMP_IVAS_FACTURA
factura.observacion   = str(data['observaciones']['obstextareaid']).strip()[:251]
factura.save()

# pongo a las lineas de iva ID de la factura
for linea_fac in LINEAS_FACTURA:
    linea_fac['invoice_id'] = factura.id
    linea_fac['serie']      = factura.serie_fact_unique

factura_new_lines(LINEAS_FACTURA)

inputVehicleMatricula = str(data['vehicle']['inputVehicleMatricula']).strip()
inputVehicleMarca     = str(data['vehicle']['inputVehicleMarca']).strip()
if inputVehicleMatricula != '' and len(inputVehicleMatricula) > 3:
    get_or_save_vehicle(factura.id, company['id'], customer.id, inputVehicleMatricula, inputVehicleMarca)

wr_invoice_in_thread(data, factura.serie_fact, customer.cif_nif)
if factura.id > 0:
    pass
else:
    return json_suzdal({'status':'error', 'message':'Fallo al crear factura'})
except Exception as e:
    wr_invoice_in_thread(data, factura.serie_fact, customer.cif_nif)
    return json_suzdal({'status':'error', 'message':str(e)})

rdata = {
    'factura_id': factura.id,
    'status': 'ok',
    'message': 'Factura creada '
}

return json_suzdal(rdata)

```

9.d. Integración con servicios externos

Aunque FacturaApp es una aplicación sencilla y autónoma, durante su desarrollo y despliegue se ha hecho uso de varios servicios externos clave que complementan su funcionamiento:

Firebase Hosting (Google)

- Función: Alojamiento gratuito del frontend (archivos HTML, CSS, JS).
- Ventaja: Permite servir la aplicación desde cualquier navegador sin necesidad de servidor propio.
- Tipo de integración: Mediante despliegue de archivos estáticos desde la terminal con el comando `firebase deploy`.

PythonAnywhere

- Función: Alojamiento del backend Django y la base de datos SQLite.
- Ventaja: Ejecuta el servidor Python y expone rutas HTTP/JSON que son consumidas por el frontend.
- Tipo de integración: Configuración del entorno virtual, despliegue manual de código Python, rutas y base de datos.

3. GitHub

- Función: Repositorio de código y control de versiones.
- Ventaja: Facilita la gestión del desarrollo, colaboración futura y documentación pública del proyecto.
- Tipo de integración: Sincronización mediante Git (`push`, `pull`, `commit`).

4. xhtml2pdf

- Función: Generación de facturas en PDF desde plantillas HTML.
- Ventaja: Permite crear documentos imprimibles desde el backend sin servicios de terceros pagos.
- Tipo de integración: Instalado como paquete Python y usado dentro del código Django.

5. FontAwesome y Google Fonts

- Función: Iconografía y tipografía personalizada para mejorar la experiencia visual del usuario.
- Tipo de integración: Vía CDN en el archivo HTML principal.

9.e. Control de versiones y GitHub

Durante el desarrollo de FacturaApp, se ha utilizado el sistema de control de versiones Git junto con la plataforma GitHub como repositorio remoto, lo cual ha sido fundamental para la organización, seguimiento del progreso y copia de seguridad del código fuente del proyecto.

Ventajas del uso de control de versiones

- Permite mantener un historial completo de cambios realizados en el código.
- Facilita la detección y corrección de errores al poder volver a versiones anteriores.
- Posibilita el trabajo modular: frontend y backend se desarrollaron en repositorios separados.
- Asegura la disponibilidad del proyecto en la nube desde cualquier equipo.

Estructura de los repositorios

- Frontend
<https://github.com/suzdalenko-dev/FacturApp>
Contiene los archivos HTML, CSS, JS, estructura de navegación, y scripts de interacción.
 - Organización por carpetas (/assets/css, /assets/js, /img, etc.).
 - Versionado de funcionalidades como: creación de facturas, listado de clientes, conexión AJAX.
- Backend
<https://github.com/suzdalenko-dev/backend-facturapps>
Contiene el proyecto Django con modelos, controladores y vistas.
 - Estructura Django estándar: models.py, views.py, controllers/, urls.py.
 - Control de versiones en momentos clave como:
 - Creación de modelos,
 - Lógica de facturación,
 - Generación de PDFs,
 - Despliegue en PythonAnywhere.

9.f. Mejoras y funcionalidades destacadas

Durante el desarrollo de FacturaApp, se han implementado una serie de mejoras funcionales que van más allá de una simple herramienta de facturación básica. Estas funcionalidades aportan valor añadido al usuario final, optimizan la experiencia de uso y profesionalizan el resultado.

1. Generación automática de facturas en PDF

- Cada factura se puede descargar como un documento PDF profesional, listo para enviar o imprimir.

- Se incluyen todos los elementos legales: número, cliente, empresa, fecha, desglose de IVA, totales, etc.
- El diseño PDF es generado desde plantillas HTML mediante xhtml2pdf.

2. Sistema modular y escalable

- Separación clara de frontend y backend, lo que permite escalar fácilmente el proyecto en el futuro (por ejemplo, migrar a REST API, usar frameworks JS, añadir usuarios).

3. Gestión completa de clientes, artículos y facturas

- CRUD completo (crear, leer, actualizar, eliminar) para:
 - Clientes con datos fiscales y contacto.
 - Artículos con IVA y precios.
 - Facturas y sus líneas con cálculo automático de totales.

4. Cálculo automático de totales e IVA

- FacturaApp calcula automáticamente:
 - Subtotal por línea.
 - IVA correspondiente.
 - Total final.
- Permite trabajar con diferentes tipos de IVA (tipología y valor), listos para adaptar a normativa española.

5. Interfaz intuitiva y accesible

- Navegación por menú lateral persistente.
- Botones de acción rápida para crear cliente, factura o artículo desde cualquier pantalla.
- Diseño responsive gracias a Bootstrap, compatible con móviles y tablets.

6. Exportación y visualización de historial

- Posibilidad de ver el historial de facturas, filtrado por cliente o fecha.
- Cada factura se puede consultar, descargar en PDF.

10.b. Puesta en marcha, explotación

a. Configuración del entorno de producción

Para poder utilizar FacturaApp como una aplicación web real y accesible desde cualquier navegador, fue necesario preparar un entorno de producción con los siguientes pasos:

1. Preparación del backend (Django):

- Configuración de settings.py en modo producción (deshabilitar debug, permitir host externo).
- Creación de superusuario para acceso administrativo (opcional).
- Configuración del archivo requirements.txt para despliegue.

2. Adaptación del frontend (HTML/CSS/JS):

- Comprobación de rutas relativas.
- Inclusión de enlaces a los scripts correctos.
- Optimización de formularios y validaciones para evitar errores al pasar a entorno real.

3. Gestión de la base de datos:

- Uso de SQLite por su ligereza y compatibilidad inmediata con PythonAnywhere.
- Carga inicial de datos de prueba (clientes, artículos, facturas) para validar funcionalidad.

b. Despliegue en servidores / hosting

El proyecto FacturaApp se despliega en dos entornos separados:

♦ Frontend: **Firebase Hosting**

- Subida de archivos HTML/CSS/JS mediante línea de comandos (firebase deploy).
- Proyecto público accesible desde navegador sin necesidad de login.
- Ventaja: alta disponibilidad y carga rápida, sin coste.

♦ Backend: **PythonAnywhere**

- Alojamiento gratuito para proyectos Django.
- Subida de código por GitHub.
- Configuración del wsgi.py para conectar el proyecto con el servidor web de producción.
- Base de datos SQLite subida y conectada con el entorno virtual de Python.

♦ Repositorio del código: **GitHub**

- Repositorio frontend: github.com/suzdalenko-dev/FacturaApp
- Repositorio backend: github.com/suzdalenko-dev/backend-facturapps

c. Accesos públicos y pruebas en producción

Una vez desplegado, se realizaron pruebas funcionales directamente sobre el entorno real. Se validaron los siguientes aspectos:

- Acceso al panel principal desde dispositivos móviles y ordenadores.
- Creación de nuevas facturas y generación en PDF.
- Registro y gestión de clientes y artículos.
- Flujo de datos entre frontend y backend mediante peticiones AJAX.
- Estabilidad de los datos almacenados en SQLite (verificación manual de registros).

Medidas de seguridad aplicadas

- Uso de HTTPS mediante el certificado gratuito proporcionado por Firebase.
- Validación de formularios en el cliente (JS) y en el servidor (Django) para evitar entradas vacías o incorrectas.
- Control de acceso básico mediante filtros en backend para evitar inserciones duplicadas o inválidas.
- Copias de seguridad periódicas del código y la base de datos mediante Git y exportación de SQLite.

FacturaApp ha sido desplegada en entornos reales utilizando servicios de momento gratuitos pero potentes, que permiten demostrar la funcionalidad completa del sistema sin necesidad de infraestructura de pago. Esto facilita futuras evoluciones del proyecto e incluso su publicación para usuarios reales.

11. Prueba y control de calidad

a. Tipos de pruebas realizadas (unitarias, funcionales...)

Durante el desarrollo de **FacturaApp**, se llevaron a cabo distintas pruebas con el objetivo de verificar la estabilidad del sistema, la precisión de los cálculos, la experiencia del usuario y el correcto funcionamiento de las funcionalidades principales.

Se han aplicado los siguientes tipos de pruebas:

1. Pruebas funcionales (end-to-end)

- Objetivo: Verificar que cada funcionalidad implementada cumpla su propósito esperado.

- Acciones realizadas:
 - Crear cliente → Guardar → Mostrar en listado.
 - Crear artículo → Verificar precios y tipo de IVA.
 - Crear factura → Seleccionar cliente y artículos → Calcular totales → Generar PDF.
 - Descargar factura PDF y comprobar que incluye los datos correctos.

Resultado esperado: Flujo completo correcto, sin errores en la navegación ni en el envío de datos.

2. Pruebas unitarias básicas (manuales)

- Objetivo: Verificar que funciones individuales de backend (como los controladores de factura, cálculo de totales o inserciones en base de datos) funcionen de forma aislada.
- Ejemplos:
 - Cálculo automático de subtotal, IVA y total en `factura_controller.py`.
 - Comprobación de creación de objetos `FacturaLineas` tras recibir una petición POST.
 - Prueba de conversión de datos a PDF (sin frontend).

Resultado esperado: Los valores devueltos por cada función coinciden con los datos de entrada procesados.

3. Validaciones del frontend (JavaScript)

- Objetivo: Evitar envío de formularios incompletos o datos erróneos.
- Comprobaciones incluidas:
 - Verificar que los campos obligatorios estén rellenos.
 - Comprobar que los valores numéricos (precio, cantidad, IVA) sean válidos.
 - Prevenir el envío de formularios si hay errores.

Resultado esperado: Mensajes de error claros, sin envío si el formulario no es válido.

4. Pruebas de compatibilidad

- Dispositivos probados:
 - PC (Windows)
 - Smartphone Android (Chrome)
 - Tablet (modo horizontal)

- Navegadores utilizados:
 - Google Chrome
 - Mozilla Firefox

Resultado esperado: Diseño adaptativo (responsive), navegación funcional en todos los dispositivos.

5. Pruebas de integración (frontend ↔ backend)

- Objetivo: Comprobar la correcta comunicación AJAX.
- Pruebas realizadas:
 - Envío de datos de factura desde JS → recepción y procesamiento en Django.
 - Respuestas JSON correctas: success: true, ID devuelto, errores en formato claro si falla.

Resultado esperado: Comunicación estable y eficaz en ambos sentidos.

Estas pruebas han permitido garantizar la calidad básica del sistema, detectar errores a tiempo y validar que FacturaApp es funcional, usable y confiable tanto en entornos de desarrollo como de producción.

11.b. Resultados y validación de funcionalidades

Una vez finalizado el desarrollo de las principales funcionalidades de FacturaApp, se procedió a su validación en condiciones reales de uso, mediante pruebas funcionales, de integración y de uso general por parte del propio desarrollador. Los resultados obtenidos fueron positivos y reflejan un correcto funcionamiento del sistema.

Resumen de funcionalidades validadas:

Funcionalidad	Resultado	Observaciones
Registro y edición de clientes	Correcto	Se almacenan todos los datos fiscales y de contacto.
Registro y edición de artículos	Correcto	Soporta distintos tipos de IVA y precio con decimales.
Creación de facturas	Correcto	Flujo completo validado, con cliente y artículos seleccionables.
Cálculo de subtotales, IVA y total	Correcto	Resultado preciso con redondeo decimal correcto.
Generación de PDF de factura	Correcto	Documento descargable con formato profesional.
Visualización del historial de facturas	Correcto	Listado con orden cronológico y cliente asociado.
Comunicación frontend-backend (AJAX)	Estable	Peticiones JSON bien recibidas y procesadas.
Validaciones de formularios	Correcto	Previene el envío de formularios vacíos o con datos inválidos.

Funcionalidad	Resultado	Observaciones
Despliegue en servidores reales	Funcional	Frontend en Firebase y backend en PythonAnywhere.
Acceso desde distintos dispositivos	Responsive	Navegación fluida en PC, móvil y tablet.

Pruebas realizadas en entorno real

Se han realizado comprobaciones completas en el entorno de producción:

- Generación de varias facturas reales con diferentes clientes y artículos.
- Descarga de los PDF generados y comprobación visual del contenido.
- Verificación de que la base de datos almacena correctamente los registros creados.
- Prueba de funcionamiento tras recargar la página o cerrar sesión.

Conclusión

Las funcionalidades clave de FacturaApp han sido correctamente implementadas y validadas. El sistema funciona de forma fluida, sin errores críticos, y es capaz de cumplir su propósito como herramienta gratuita de facturación para autónomos y pequeñas empresas.

12. Plan de empresa

a. Objetivos generales

El objetivo principal de **SNLE**, empresa desarrolladora de la aplicación **FacturaApp**, es convertirse en una solución accesible, profesional y gratuita de facturación digital dirigida a autónomos, microempresas y emprendedores que necesiten emitir facturas legales sin depender de plataformas complejas o costosas.

La visión a medio plazo es ofrecer servicios complementarios (como copias de seguridad, facturación electrónica avanzada o generación automática de modelos fiscales) mediante planes escalables, manteniendo siempre una versión gratuita funcional.

Los objetivos específicos para los primeros tres años de operación son los siguientes:

1. Primer Año – Puesta en marcha y validación del modelo

- Registrar la empresa SNLE como persona jurídica y formalizar el servicio de FacturaApp.
- Establecer una versión estable y funcional de la aplicación accesible públicamente.
- Alojamiento inicial (Firebase, PythonAnywhere) con miras a migrar a servidores propios o VPS según la demanda.
- Contratar a un primer desarrollador de soporte para mantenimiento, resolución de incidencias y mejoras técnicas.
- Promocionar la herramienta mediante redes sociales, foros para autónomos y pequeñas campañas de marketing digital.

- Implementar encuestas de satisfacción y formularios de contacto para recoger sugerencias y errores.

Segundo Año – Consolidación y fidelización

- Recoger feedback del primer año para lanzar una versión mejorada de FacturaApp (v2.0).
- Añadir nuevas funciones como:
 - Facturación recurrente.
 - Informes personalizados por trimestre.
 - Soporte multiempresa para asesorías o freelancers con varios clientes.
- Implementar un sistema freemium: mantener la versión gratuita y ofrecer mejoras premium opcionales.
- Crear una comunidad de usuarios: canal de Telegram, sistema de soporte colaborativo, newsletter.

Tercer Año – Escalado y profesionalización

- Establecer a SNLE como una referencia nacional en software de facturación accesible.
- Comenzar la traducción multilingüe de FacturaApp para su posible expansión a otros países.
- Solicitar la homologación oficial de la herramienta para facturación electrónica con validación legal.
- Ampliar el equipo: incorporar perfiles de atención al cliente, legal o marketing si se confirma el crecimiento.
- Invertir en infraestructura propia (servidores dedicados, copias de seguridad automáticas, certificados SSL avanzados).

Visión a largo plazo

FacturaApp aspira a consolidarse como una **herramienta sencilla, segura y legalmente válida** para cualquier profesional que necesite emitir facturas sin conocimientos técnicos, y convertirse en una alternativa ética y eficiente frente a plataformas comerciales cerradas.

12.b. Recursos humanos y estructura de la empresa

La empresa desarrolladora del proyecto FacturaApp será constituida bajo el nombre de SNLE, como una empresa tecnológica enfocada en soluciones de facturación para autónomos y pequeñas empresas.

La estructura organizativa inicial es sencilla y funcional, adaptada a un entorno de emprendimiento individual con proyección de crecimiento.

1. Área de Desarrollo

- Responsable: Alexey Suzdalenko.

- Funciones:
 - Desarrollo del backend (Django, SQLite).
 - Desarrollo del frontend (HTML, CSS, JavaScript).
 - Mantenimiento de servidores (Firebase, PythonAnywhere).
 - Mejora y actualización continua del sistema.

2. Área de Diseño y UX

- Responsable: Alexey Suzdalenko.
- Funciones:
 - Diseño visual del sistema y estructura de navegación.
 - Adaptación responsive (móvil/tablet/escritorio).
 - Optimización de la experiencia de usuario.

3. Área de Dirección y Administración

- Responsable: Alexey Suzdalenko.
- Funciones:
 - Toma de decisiones estratégicas.
 - Coordinación del proyecto.
 - Comunicación con gestoría externa para trámites legales y fiscales.

4. Área de Soporte Técnico y Atención al Cliente (*futuro*)

- En una fase posterior se incorporará un perfil de soporte que atenderá dudas de usuarios, incidencias técnicas y peticiones de ayuda.

Plan de contratación y crecimiento

Año 1 – Puesta en marcha como SLNE

- La empresa SNLE inicia su actividad ya constituida como Sociedad Limitada Nueva Empresa.
- Se contrata desde el primer momento a un trabajador asalariado que cumplirá una función polivalente:
 - Atención al cliente y soporte técnico básico.

- Apoyo en marketing digital y redes sociales.
- Asistencia en tareas de desarrollo no críticas (testeo, documentación, feedback funcional).
- El contrato podrá ser a jornada parcial o completa, adaptado al flujo de trabajo y presupuesto inicial.
- El fundador, Alexey Suzdalenko, asumirá la dirección técnica y estratégica de todo el proyecto.

Año 2 – Estabilización y crecimiento orgánico

- Objetivo: consolidar la base de usuarios y ampliar la funcionalidad de FacturaApp según la demanda real.
- Se valorará la contratación de un desarrollador junior para acelerar la mejora continua de la plataforma.
- Se podrían subcontratar servicios especializados (legal, fiscal, SEO) según necesidades puntuales.

Año 3 – Ampliación del equipo y profesionalización

- Si se alcanza un volumen de usuarios y facturación suficientes:
 - Se contratará un segundo técnico especializado o perfil de soporte y atención comercial.
 - Se formalizará la externalización de la gestión fiscal y contable.
 - Se explorará el lanzamiento de una versión premium de FacturaApp con funcionalidades avanzadas.

Conclusión

Desde el inicio, SNLE está estructurada como una empresa real con capacidad de operar profesionalmente, contratar personal y ofrecer un servicio estable y escalable. La decisión de comenzar como sociedad limitada en lugar de autónomo refuerza el compromiso con la viabilidad y crecimiento sostenible del proyecto.

12.c. Inversiones, compras y gastos

Sueldo del fundador (Alexey Suzdalenko) – Año 1

Aunque Alexey Suzdalenko es socio y administrador único de SNLE, se reserva una retribución mínima mensual equivalente a 1.100 € netos, lo cual supone:

Concepto	Importe estimado (€)
Sueldo neto mensual	1.100,00
Sueldo neto anual	$1.100 \times 12 = 13.200,00$ €
Sueldo bruto anual	~17.500 € (estimado con IRPF ~24% + SS empresa)
Coste empresa total	~19.500 € (bruto + Seguridad Social empresa)

Coste de 1 trabajador asalariado (salario mínimo)

El trabajador contratado desde el inicio desempeñará tareas de soporte, atención al cliente y marketing. Se calcula su coste basado en el SMI 2024: 1.134 € brutos \times 14 pagas = 15.876 €/año.

Concepto	Importe estimado (€)
Sueldo bruto anual	15.876,00
Seguridad Social empresa	960,00
IRPF estimado (empleado)	2.000,00
Coste total empresa	~16.836,00

Resumen de inversión inicial - Año 1

Concepto	Importe (€)
Equipos informáticos	1.256,00
IVA de equipos (21%)	263,76
Registro marca y propiedad intelectual	300,00
Constitución SLNE	250,00
Herramientas y alojamiento	132,00
Sueldo anual Alexey Suzdalenko (fundador)	19.500,00
Sueldo y coste laboral trabajador contratado	16.836,00
Total estimado inversión Año 1	38.537,76 €

Los cálculos salariales incluyen estimaciones de IRPF y Seguridad Social y pueden variar ligeramente según régimen fiscal exacto y comunidad autónoma. Se ha asumido un entorno fiscal estándar para 2024.

12.d. Ingresos

Año 1 - Etapa gratuita de captación

- Modelo: 100% gratuito para todos los usuarios.
- Objetivo: Obtener una base de usuarios activos, consolidar la marca y validar la utilidad de la plataforma.
- Ingresos estimados: 0 €

- Acciones paralelas:
 - Solicitud de ayudas para digitalización o emprendimiento joven.
 - Posible colaboración con academias, escuelas o coworkings para dar visibilidad.

Año 2 – Introducción del modelo freemium

- Se mantendrá una versión gratuita con funcionalidades básicas.
- Se ofrecerán planes premium para usuarios avanzados que deseen más prestaciones:

Plan	Precio estimado	Incluye
Gratis	0 €/mes	Hasta 30 facturas/mes, sin soporte prioritario.
Pro Autónomo	4,99 €/mes	Facturas ilimitadas, exportación Excel, soporte por email.
Pro Empresa	9,99 €/mes	Soporte rápido, plantillas personalizadas, más espacio en la nube.
Asesoría o Multiusuario	19,99 €/mes	Acceso multiempresa y usuarios, informes trimestrales, marca blanca.

Escenario estimado (moderado):

- 500 usuarios activos
- 10% contratan el plan Pro Autónomo
- 2% contratan el plan Pro Empresa

Ingresos mensuales estimados (Año 2):

- $(50 \times 4,99 \text{ €}) + (10 \times 9,99 \text{ €}) = 249,50 \text{ €} + 99,90 \text{ €} = \sim 349,40 \text{ €/mes}$
- Ingresos anuales estimados (Año 2): $\sim 4.192,80 \text{ €}$

Año 3 – Expansión y diversificación

- Aumento de usuarios y mejora de los servicios.
- Posibilidad de incluir:
 - Integración con Hacienda (factura electrónica real).
 - Publicidad contextual (sin comprometer privacidad).
 - Servicio adicional de formación o acompañamiento fiscal.

Ingresos estimados:

- 1.000 usuarios activos
- Mismos porcentajes de conversión
- Ingresos mensuales: ~699 €
- Ingresos anuales estimados (Año 3): ~8.388 €

FacturaApp nace como una herramienta gratuita para aportar valor y posicionarse. Sin embargo, a partir del segundo año, el modelo **freemium escalable** permite comenzar a generar ingresos de forma progresiva sin dejar de ofrecer una opción útil y gratuita. Esto facilita un crecimiento sostenible y adaptable a la demanda real del mercado.

Ya que proyección de ganancias económicas no cubren los cerca de 40000 euros que se necesitan cada año para estar al corriente de todos los pagos SLNE, por estos motivos se contempla:

- La solicitud de subvenciones públicas para emprendimiento digital y primera contratación.
- El escalado progresivo del modelo freemium con nuevas funciones premium que aporten valor real.
- La búsqueda de fuentes de ingresos complementarias, como formación, servicios a medida o colaboración con asesorías.
- Una planificación financiera prudente que permita adaptar los recursos humanos al ritmo de crecimiento real del proyecto.

Estas medidas están orientadas a lograr la sostenibilidad del proyecto en un plazo razonable de 2 a 3 años, sin renunciar a ofrecer una herramienta gratuita y útil desde el primer día.

13. Gestión económica de un proyecto

a. Recursos materiales

Durante el desarrollo del proyecto FacturaApp, se han utilizado los siguientes recursos materiales:

- Portátil ASUS VivoBook – herramienta principal de trabajo para desarrollo backend, frontend, documentación y gestión del proyecto.
- Monitor externo LG 24” – mejora de la productividad visual, especialmente útil para tareas de diseño, comparación de interfaces, edición simultánea de código y documentación.
- Smartphone Android – empleado para realizar pruebas reales de visualización en dispositivos móviles y asegurar la compatibilidad responsive del sistema.
- Impresora multifunción Epson – utilizada para impresión de pruebas de facturas, generación de documentación para anexar y material de consulta.
- Software gratuito y de código abierto, incluyendo:

- VS Code para programación.
- Django como framework backend.
- Bootstrap para diseño responsivo.
- LibreOffice para la redacción de la memoria y documentación.
- Draw.io para la elaboración de diagramas técnicos.

Estos recursos han permitido desarrollar FacturaApp en un entorno económico, accesible y suficientemente profesional para cumplir con los objetivos del proyecto.

13.b. Costes y gastos desglosados

Concepto	Importe estimado (€)
Equipos informáticos	1.256,00
IVA equipos (21%)	263,76
Registro de marca + propiedad intelectual	300,00
Constitución de la SLNE	250,00
Hosting backend (PythonAnywhere, plan pago)	60,00
G Suite / correo profesional	72,00
Salario bruto Alexey Suzdalenko (fundador)	17.500,00
Coste laboral trabajador (salario + SS)	16.836,00
Total estimado gastos Año 1	36.537,76 €

13.c. Horas de trabajo y valoración económica

Periodo de trabajo efectivo:

- Intensivo entre marzo y abril (2 meses).
- Estimación media: 10 días por mes × 6 h/día = 120 horas totales.

Cálculo	Valor
Días totales trabajados	20 días
Horas totales	120 h
Valor estimado por hora	23 €/h
Coste estimado personal	120 h × 23 €/h = 2.760 €

13.d. Umbral de rentabilidad

El umbral de rentabilidad representa el mínimo ingreso mensual que FacturaApp debería generar para cubrir gastos operativos básicos + salario personal net

Cálculo realista del umbral mensual (empresa SNLE con 2 personas)

Concepto	Importe mensual estimado (€)
Sueldo bruto Alexey Suzdalenko	1.460,00
Cotización Seguridad Social fundador	380,00
Sueldo bruto trabajador contratado (SMI)	1.134,00
Cotización Seguridad Social trabajador	375,00
Herramientas digitales (hosting, G Suite)	11,00
Luz, internet, suministros oficina estimado	70,00
Total gastos mensuales	~3.430,00 €

Cálculo del umbral por hora realista con 160 h/mes, si mantenemos el mismo umbral de rentabilidad mensual = 3.430 €, pero con 160 horas trabajadas:

Cálculo	Resultado
Umbral mensual total	3.430,00 €
Horas trabajadas al mes	160 h
Valor mínimo por hora rentable	~21,44 €/h

13.e. Estimación del coste total de desarrollo

El desarrollo de FacturaApp ha sido realizado por el propio fundador en un periodo intensivo de 2 meses a jornada parcial. No obstante, para estimar su valor económico real, se considera tanto el tiempo efectivo invertido como los recursos humanos y materiales que serían necesarios si el proyecto fuera encargado a un equipo externo o desarrollado en condiciones de mercado.

Valoración por horas de trabajo personal

Concepto	Valor estimado
Horas de trabajo reales	120 horas
Valor/hora (estándar freelance junior-medio)	23 €/hora
Coste estimado personal	2.760 €

14. Conclusiones y valoración personal – FacturaApp

a. Valoración personal del proceso

Desarrollar **FacturaApp** ha sido una experiencia exigente, pero profundamente enriquecedora. Gracias a este proyecto, no solo he reforzado mis conocimientos técnicos en desarrollo web, sino que también me he adentrado en el **funcionamiento real del mundo de la facturación y la normativa fiscal en España**.

Al diseñar una herramienta que emite facturas legales y completas, me vi obligado a estudiar **cómo se crean, validan y emiten las facturas reales**, y a entender en profundidad la estructura de impuestos como el **IVA**, los **recargos de equivalencia** y otras normativas contables. Fue entonces cuando tomé conciencia de la **enorme complejidad innecesaria** que muchas veces rodea procesos administrativos que deberían ser simples.

Tuve que rehacer parte del sistema de cálculo de impuestos cuando cambiaron las **normas sobre tipos de IVA en España**, lo que me hizo pensar en algo importante: **si yo, como desarrollador individual, necesité horas para adaptar un simple sistema**, ¿cuánto costará para todas las empresas que dependen de grandes ERP o CRM? Cientos de compañías deben ajustar sus sistemas constantemente solo por cambios fiscales. El coste acumulado de este tipo de decisiones legislativas mal planificadas es brutal –y no solo económico, sino también humano: **más carga administrativa, más horas de trabajo, más burocracia y, al final, precios más altos para todos**.

Este tipo de impacto no siempre se ve a simple vista, pero **afecta directamente a la calidad de vida de las personas**. Me resultó muy llamativo descubrir que **el IVA no existía en España hasta hace unas décadas**, y que antes la economía funcionaba sin este tipo de presión fiscal sobre cada venta. Esto me hizo reflexionar mucho sobre la carga que suponen los impuestos indirectos y sobre la desconexión entre quienes toman decisiones normativas y quienes las implementamos en la práctica.

14.b. Dificultades encontradas y cómo se resolvieron

El principal obstáculo técnico fue la generación de facturas en formato PDF. Necesitaba que el diseño del documento fuera limpio, profesional y adaptable a los datos variables introducidos por el usuario. La solución fue utilizar una librería Python especializada (xhtml2pdf), pero su instalación y configuración en el entorno de producción de PythonAnywhere fue complicada debido a restricciones del sistema y dependencias adicionales.

Tuve que investigar en foros, probar alternativas, ajustar plantillas HTML y hacer múltiples pruebas hasta conseguir una generación estable y correcta de PDFs con diseño coherente y datos precisos.

Otro reto fue la integración AJAX entre frontend y backend, que requería validar los datos de forma segura, mantener coherencia entre cliente y servidor, y manejar errores correctamente en tiempo real.

A pesar de estas dificultades, cada reto técnico me sirvió para aprender de forma práctica, y hoy me siento más preparado para enfrentar problemas reales en cualquier entorno de desarrollo profesional.

14.c. Conclusión general del proyecto

El desarrollo de **FacturaApp** ha sido, sin duda, uno de los aprendizajes más completos y útiles de todo el ciclo de **Desarrollo de Aplicaciones Web (DAW)**. Me permitió aplicar conocimientos de **JavaScript, Django, HTML/CSS, bases de datos relacionales**, gestión de proyectos y despliegue en servidores reales.

He aprendido no solo a programar mejor, sino también a planificar, investigar por mi cuenta, superar frustraciones y, sobre todo, a **desarrollar una herramienta que resuelve un problema real**.

Más allá de lo técnico, este proyecto me ayudó a entender cómo las decisiones que parecen pequeñas – como un cambio en los tipos de IVA – tienen un efecto dominó que **repercute en miles de profesionales, empresas y usuarios finales**. Ojalá los legisladores entendieran mejor las implicaciones prácticas de sus decisiones.

FacturaApp ha sido una herramienta para aprender, crear y también reflexionar. Y me ha confirmado algo importante: la programación no solo sirve para escribir código, sino también para comprender mejor el mundo que nos rodea.

15. Bibliografía

Normativa y fiscalidad

- Agencia Tributaria Española (AEAT) – Información oficial sobre facturación e IVA
<https://www.agenciatributaria.es>
- BOE – Real Decreto 1619/2012, por el que se aprueba el Reglamento de facturación
<https://www.boe.es>
- Historia del IVA en España – Blog y artículos de asesores fiscales y prensa económica
Ej.: <https://www.eleconomista.es>

Tecnología y desarrollo

- Documentación oficial de Django
<https://docs.djangoproject.com>
- Bootstrap 5 – Guía de componentes y diseño responsive
<https://getbootstrap.com>
- Mozilla Developer Network (MDN) – JavaScript, HTML y CSS
<https://developer.mozilla.org>
- Documentación xhtml2pdf – Generación de PDFs desde Django
<https://xhtml2pdf.readthedocs.io>
- Firebase Hosting – Guía de despliegue
<https://firebase.google.com/docs/hosting>
- PythonAnywhere – Configuración y despliegue de proyectos Django
<https://help.pythonanywhere.com>

Fuentes complementarias

Stack Overflow – Resolución de errores y dudas técnicas

<https://stackoverflow.com>

- YouTube – Tutoriales sobre Django, AJAX y generación de PDFs

<https://www.youtube.com/>

16. Anexos – FacturaApp

A continuación, se incluyen los documentos y materiales complementarios que acompañan el desarrollo del proyecto FacturaApp, organizados por categoría:

A. Capturas de pantalla

- A.1. Vista del panel principal (dashboard) tras el login.

facturapps.web.app/dashboard/

Facturas

+ Artículo + Cliente + Factura

Total Facturas 2

Fecha	Núm	Razón o denominación social	Subtotal	IVA	Total	
2025-05-09	0-2025-2	IES Peñacastillo	2250.00	472.50	2722.50	 
2025-05-05	0-2025-1	IES Peñacastillo	4220.00	886.20	5106.20	 

- A.2. Formulario de creación de factura.

facturapps.web.app/dashboard/#CreateInvoice

Crear Factura

+ Artículo + Cliente + Factura

Tipo Factura

Tipo
Ordinaria/Rectificativa/
Abono

Factura Ordinaria

Datos cliente y tipo de factura

Número cliente NIF CIF cliente

Razón o denominación social


Nombre cliente

Datos vehículo

Matrícula

Marca/Modelo/Kilómetros

Líneas de factura

Código Art.	Descripción Art.	Cantidad	Precio	% Dto.	Importe	IVA	
		1		0		21 %	
+ Línea							
	Mano de obra		22,00			21 %	

Subtotal: 0
IVA: 0
Total: 0

Observaciones

Crear Factura

- A.3. Gestión de artículos y clientes.

facturapps.web.app/dashboard/#Article

FacturaApp Artículos

+ Artículo + Cliente + Factura

Total artículos 4

Número	Descripción	Precio unidad	IVA	Tipo
4	Desarrollo API personalizada	1000.00 €	21.00 %	norm
3	Configuracion Servidor	30.00 €	21.00 %	norm
2	Informe Power BI	3000.00 €	21.00 %	norm
1	Desarrollo App Personalizada	1000.00 €	21.00 %	norm

facturapps.web.app/dashboard/#Customer

FacturaApp Clientes

+ Artículo + Cliente + Factura

Total clientes 2

Núm	CIF NIF	Razón o denominación social	Ciudad	Teléfono
2	A12345678	Banco Santander	Santander	942567143
1	IES Peñacastillo	IES Peñacastillo	Peñacastillo	942 32 16 50

A.4. Vista de historial de facturas.

facturapps.web.app/dashboard/#Reports

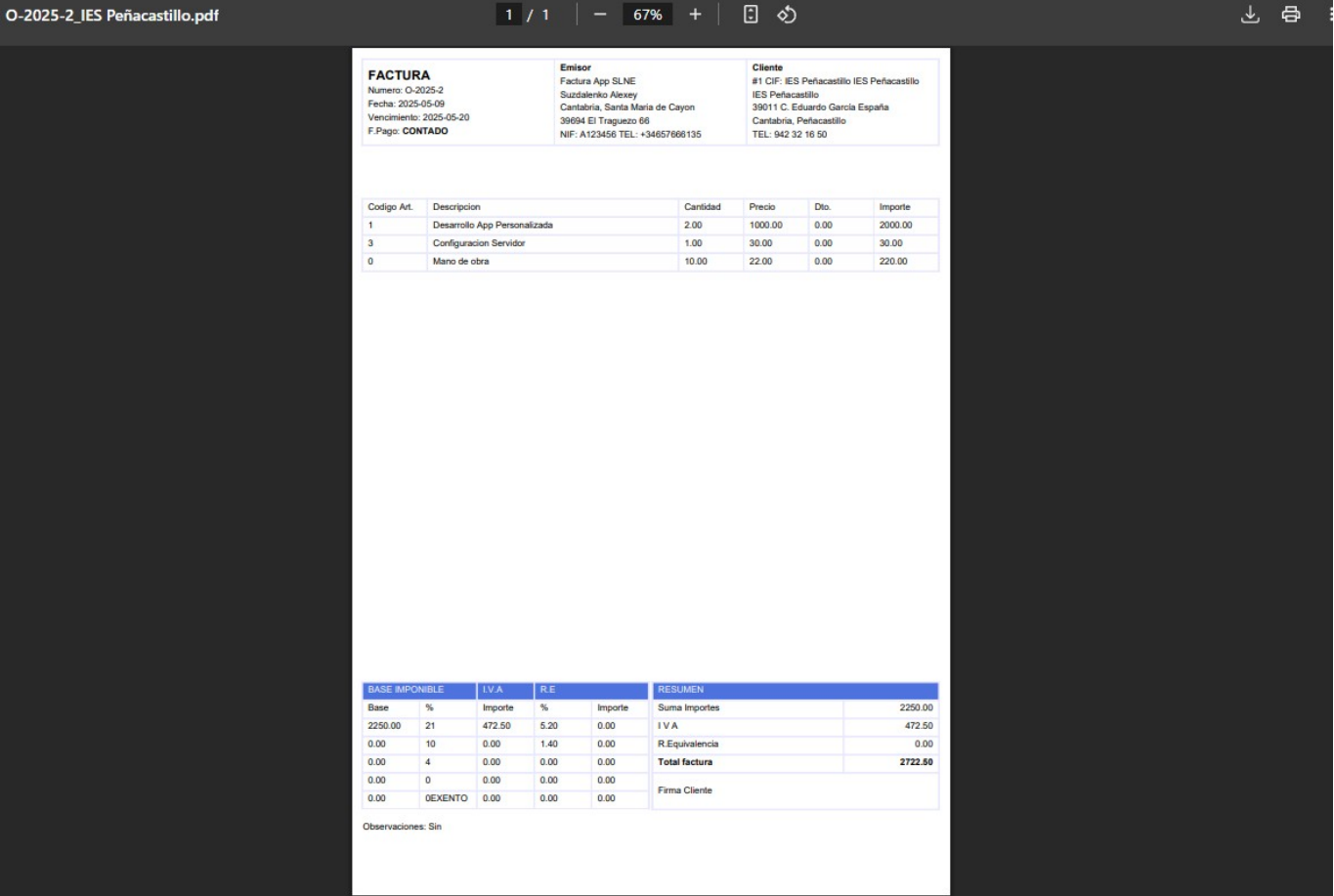
FacturaApp Informes

+ Artículo + Cliente + Factura

Total informes 4

Entidad	Nombre	Fecha desde	Fecha Hasta
Factura	Listado facturas	dd/mm/aaaa	dd/mm/aaaa
Cliente	Listado clientes		
Artículo	Listado artículos		

A.5. Ejemplo de factura generada en PDF.



- A.6. Versión móvil del sistema (responsive).

facturapps.web.app/dashboard/#MyCompany

FacturaApp Mi Empresa + Artículo + Cliente + Factura

Denominación

Razón o denominación social
Factura App SLNE

Nombre y apellidos persona
Suzdalenko Alexey

Email cara cliente
alexey.suzdalenko@gmail.com

CIF-NIF
A123456

Email login
alexey.suzdalenko@gmail.com

Dirección

País
España

Provincia
Cantabria

Código postal
39694

Cuidad
Santa Maria de Cayon

Dirección
El Traguezo 66

Teléfono principal
+34657666135

Teléfono
+34657666135

B. Fragmentos de código

- B.1. Código JavaScript para envío de datos vía fetch.

```
async function postRequest(url, formData){  
  
try {  
  formData.append('company_id', window.localStorage.getItem('company_id'));
```

```

formData.append('email', window.localStorage.getItem('email'));
formData.append('cif', window.localStorage.getItem('cif'));
formData.append('uid', window.localStorage.getItem('uid'));
formData.append('password', window.localStorage.getItem('password'));

let uri = HTTP_URL+url;
let response = await fetch(uri, {method:'POST', body: formData});
let data = await response.json();
if(data && data.status == 'error'){
    setTimeout(() => { showM('4. ' + data.message, 'error'); }, 1000);
}
return data;
} catch (error) {
    showM('Error 1 '+error, 'error')
    console.error('There was a problem with the fetch operation:', error);
}
}

```

- B.2. Código Django para generación de facturas (factura_controller.py).

```

try:

    customer = Customer.objects.get(id=data['cliente']['clientIdDeveloper'], clientcode=data['cliente']['clientNumber'],
company_id=company['id'])

    factura = Factura.objects.create(company_id=company['id'], tipo_factura=tipo_factura, ejercicio=ejercicio)
    document = Document.objects.filter(company_id=company['id'], description=tipo_factura, ejercicio=ejercicio).values('value').first()
    factura.name_factura = str(data['factura']['name_factura']).strip()
    factura.apunta_factura = str(data['factura']['apunta_factura']).strip()
    factura.numero = document['value']
    factura.serie_fact = f"{tipo_factura}-{ejercicio}-{factura.numero}"
    factura.serie_fact_unique = f"{tipo_factura}-{ejercicio}-{factura.numero}-{company['id']}"
    factura.fecha_expedicion = fecha_expedicion()
    factura.vencimiento = get_time_11days()

    factura.customer_id = customer.id
    factura.customer_num = customer.clientcode
    factura.receptor_company_name = customer.razon

    SUBTOTAL_FACTURA = 0
    IMP_IVAS_FACTURA = 0
    TOTAL_FACTURA = 0
    LINEAS_FACTURA = []

    # Base Imponible = Precio del artículo × Cantidad de artículos
    for linea in lineas:
        description = str(linea.get('description', 'none')).strip()
        idArticle1 = str(linea.get('idArticle1', '')).strip()
        precio1 = float(linea.get('precio1', 0))
        cantidad1 = float(linea.get('cantidad1', 0))
        descPorc = float(linea.get('descPorc', 0))
        ivaPorcent = float(linea.get('ivaPorcent', 0))
        ivaTypeStr = str(linea.get('ivaType', '0'))

```

```

if idArticle1.isdigit(): # Comprobar si es un número válido
    articulo_current = Article.objects.filter(id=idArticle1, company_id=company['id']).first()
else:
    art_created, articulo_current = factura_new_article(description, company['id'], precio1, ivaTypeStr, ivaPorcent)
    if art_created is None or articulo_current is None:
        return json_suzdal({'status':'error', 'message':'Error al crear artículo nuevo'})

importe_inicio      = cantidad1 * precio1
valor_descuento     = descPorc / 100 * importe_inicio
importe_con_descuento = importe_inicio - valor_descuento
SUBTOTAL_FACTURA   += importe_con_descuento
valor_iva           = ivaPorcent / 100 * importe_con_descuento
IMP_IVAS_FACTURA    += valor_iva
importe_final       = importe_con_descuento + valor_iva
TOTAL_FACTURA      += importe_final

for d in desglose:
    if str(d['iva']) == ivaTypeStr:
        d['base_imponible'] += importe_con_descuento
        d['valor_iva'] += valor_iva
        d['total_con_iva'] += d['base_imponible'] + d['valor_iva']

linea_factura = {'invoice_id':0, 'serie': '', 'company_id':company['id'], 'article_id':articulo_current.id,
'articulo_num':articulo_current.artcode, 'articulo_name':articulo_current.description, 'cantidad':cantidad1, 'precio':precio1,
'descuento':descPorc, 'iva_porcent':ivaPorcent, 'iva_type':ivaTypeStr}
LINEAS_FACTURA.append(linea_factura)

# ahora el calculo de mano de obra
canridadManoObra = float(data['manoObra']['canridadManoObra'])
precioManoObra   = float(data['manoObra']['precioManoObra'])
descManoObr     = float(data['manoObra']['descManoObr'])
ivaPorcentManoOb = float(data['manoObra']['ivaPorcentManoOb'])
tipoIvaManoObra  = str(data['manoObra']['tipoIvaManoObra'])

# cuando existe mano de obra
if canridadManoObra > 0 and precioManoObra > 0:
    importe_inicio_mo      = canridadManoObra * precioManoObra
    valor_descuento_mo     = descManoObr / 100 * importe_inicio_mo
    importe_con_descuento_mo = importe_inicio_mo - valor_descuento_mo
    SUBTOTAL_FACTURA      += importe_con_descuento_mo
    valor_iva_mo           = ivaPorcentManoOb / 100 * importe_con_descuento_mo
    IMP_IVAS_FACTURA       += valor_iva_mo
    importe_final_mo       = importe_con_descuento_mo + valor_iva_mo
    TOTAL_FACTURA         += importe_final_mo
    linea_factura = {'invoice_id':0, 'serie': '', 'company_id':company['id'], 'article_id':0, 'articulo_num':0, 'articulo_name':'Mano
de obra', 'cantidad':canridadManoObra, 'precio':precioManoObra, 'descuento':descManoObr, 'iva_porcent':ivaPorcentManoOb,
'iva_type':tipoIvaManoObra}
    LINEAS_FACTURA.append(linea_factura)

for d in desglose:
    if str(d['iva']) == tipoIvaManoObra:
        d['base_imponible'] += importe_con_descuento_mo

```

```

        d['valor_iva'] += valor_iva_mo
        d['total_con_iva'] += d['base_imponible'] + d['valor_iva']

    factura.ivas_desglose = json.dumps(desglose)
    factura.subtotal      = SUBTOTAL_FACTURA
    factura.importe_ivas  = IMP_IVAS_FACTURA
    factura.total         = TOTAL_FACTURA
    factura.total2        = SUBTOTAL_FACTURA + IMP_IVAS_FACTURA
    factura.observacion   = str(data['observaciones']['obstextareaid']).strip()[251]
    factura.save()

    # pongo a las lineas de iva ID de la factura
    for linea_fac in LINEAS_FACTURA:
        linea_fac['invoice_id'] = factura.id
        linea_fac['serie']      = factura.serie_fact_unique

    factura_new_lines(LINEAS_FACTURA)

    inputVehicleMatricula = str(data['vehicle']['inputVehicleMatricula']).strip()
    inputVehicleMarca      = str(data['vehicle']['inputVehicleMarca']).strip()
    if inputVehicleMatricula != '' and len(inputVehicleMatricula) > 3:
        get_or_save_vehicle(factura.id, company['id'], customer.id, inputVehicleMatricula, inputVehicleMarca)

    wr_invoice_in_thread(data, factura.serie_fact, customer.cif_nif)
    if factura.id > 0:
        pass
    else:
        return json_suzdal({'status': 'error', 'message': 'Fallo al crear factura'})
except Exception as e:
    wr_invoice_in_thread(data, factura.serie_fact, customer.cif_nif)
    return json_suzdal({'status': 'error', 'message': str(e)})

rdata = {
    'factura_id': factura.id,
    'status': 'ok',
    'message': 'Factura creada '
}

```

- B.3. Modelo de base de datos (models.py – tabla Artículo, Company, Cliente).

```

class Article(models.Model):

    id = models.AutoField(primary_key=True)
    company_id = models.BigIntegerField(null=True)

    artcode = models.PositiveBigIntegerField(default=0)
    description = models.CharField(max_length=254, null=True)
    price = models.DecimalField(max_digits=11, null=True, decimal_places=2)

```

```

iva            = models.DecimalField(max_digits=11, null=True, decimal_places=2)
ivatype        = models.CharField(max_length=11, null=True)

class Meta:
    indexes = [
        models.Index(fields=['id']),
        models.Index(fields=['company_id']),
        models.Index(fields=['description']),
    ]

class Company(models.Model):

    id            = models.AutoField(primary_key=True)
    razon         = models.CharField(max_length=111, null=True)
    person_name   = models.CharField(max_length=111, null=True)
    cif           = models.CharField(max_length=33, unique=True, null=True,)
    email         = models.CharField(max_length=111, unique=True, null=True)
    emailcompany  = models.CharField(max_length=111, null=True)
    password      = models.CharField(max_length=222, null=True)
    tlf           = models.CharField(max_length=33, null=True)
    tlf2          = models.CharField(max_length=33, null=True)

    numvisit      = models.BigIntegerField(default=0)
    regtime       = models.CharField(max_length=33, null=True)
    lastvisit     = models.CharField(max_length=33, null=True)
    uid           = models.CharField(max_length=111, null=True)

    country       = models.CharField(max_length=33, null=True)
    province      = models.CharField(max_length=33, null=True)
    zipcode       = models.CharField(max_length=22, null=True)
    city          = models.CharField(max_length=33, null=True)
    address       = models.CharField(max_length=111, null=True)
    price         = models.DecimalField(max_digits=11, decimal_places=2, default=0)

class Meta:
    indexes = [
        models.Index(fields=['id']),
    ]

class Customer(models.Model):

    id            = models.AutoField(primary_key=True)
    company_id    = models.BigIntegerField(null=True)
    clientcode    = models.PositiveBigIntegerField(default=0)

    cif_nif       = models.CharField(max_length=33, null=True)
    razon         = models.CharField(max_length=255, null=True)
    person_name   = models.CharField(max_length=255, null=True)
    emailcustomer = models.CharField(max_length=111, null=True)
    phone         = models.CharField(max_length=33, null=True)

```

```

country      = models.CharField(max_length=111, default='España')
province     = models.CharField(max_length=111, null=True)
zipcode      = models.CharField(max_length=11, null=True)
city         = models.CharField(max_length=111, null=True)
address      = models.CharField(max_length=255, null=True)

```

```

class Meta:
    indexes = [
        models.Index(fields=['id']),
        models.Index(fields=['company_id']),
        models.Index(fields=['cif_nif']),
    ]

```

- B.4. Fragmento código creacion del PDF:

```

def pdf_work(request, action, id):
    file_path      = ""
    url_email      = ""
    invoice_name    = ""
    company_mail    = ""
    custome_mail    = ""
    try:
        auth_status, company = user_auth(request, None)
        if auth_status is None or company is None:
            return json_suzdal({'login': False, 'status': 'error', 'message': 'Usuario no esta logeado'})

        facturaObj = Factura.objects.get(id=id, company_id=company['id'])
        customerObj = Customer.objects.get(id=facturaObj.customer_id, company_id=company['id'])
        vehicle     = Vehicledata.objects.filter(invoice_id=id, company_id=company['id']).first()
        lineasFact  = Facturalineas.objects.filter(invoice_id=id, company_id=company['id'])
        company_mail= str(company['emailcompany']).strip()
        custome_mail= str(customerObj.emailcustomer).strip()

        current_time = datetime.now()
        year = str(current_time.strftime('%Y'))
        folder_path = os.path.join(settings.BASE_DIR, 'media', year, str(company['id']), 'pdf')
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
        file_name = f"{facturaObj.serie_fact}_{customerObj.cif_nif}.pdf"
        file_path = os.path.join(folder_path, file_name)

        html_template_path = os.path.join(settings.BASE_DIR, 'media', 'fac.html')
        with open(html_template_path, 'r') as file:
            html = file.read()

        html = html.replace('@name_factura@', str(facturaObj.name_factura))
        html = html.replace('@numero_factura@', str(facturaObj.serie_fact))
        invoice_name = f"{str(facturaObj.name_factura)} {str(facturaObj.serie_fact)}"
        html = html.replace('@fecha_factura@', str(facturaObj.fecha_expedicion))

```



```

html = html.replace('@fecha_vencimiento@', str(facturaObj.vencimiento))
if len(str(facturaObj.apunta_factura)) > 11: html = html.replace('@apunta_a_factura@', 'APUNTA A: '+str(facturaObj.apunta_factura))
else: html = html.replace('@apunta_a_factura@', '')
html = html.replace('@razon@', company['razon'])
html = html.replace('@person_name@', company['person_name'])
html = html.replace('@province@', company['province'])
html = html.replace('@city@', company['city'])
html = html.replace('@zipcode@', company['zipcode'])
html = html.replace('@address@', company['address'])
html = html.replace('@cif@', company['cif'])
html = html.replace('@tlf@', company['tlf'])

html = html.replace('@customer_num@', str(facturaObj.customer_num))
html = html.replace('@razon_cl@', customerObj.razon)
html = html.replace('@person_name_cl@', customerObj.person_name)
html = html.replace('@province_cl@', customerObj.province)
html = html.replace('@city_cl@', customerObj.city)
html = html.replace('@zipcode_cl@', customerObj.zipcode)
html = html.replace('@address_cl@', customerObj.address)
html = html.replace('@cif_nif@', customerObj.cif_nif)
html = html.replace('@phone@', customerObj.phone)
html = html.replace('@country@', customerObj.country)
if vehicle:
    vehicle_data = f"""<div class="div_vehicle">
        <span class="datos_vehicle">DATOS DEL VEHICULO</span>
        <table style="border: 1px solid rgb(233, 233, 255); color: black;">
            <tbody><tr><td>Matricula<br>{str(vehicle.matricula)}</td><td>Marca / Modelo /
Kilometros<br>{str(vehicle.other_data)}</td></tr></tbody>
        </table>
    </div><br>"""
else:
    vehicle_data = ''
html = html.replace('@vehicle_data@', vehicle_data)

lines_content = ''
for linea in lineasFact:
    lines_content += """<tr><td>"""+str(linea.article_num)+"""</td><td style="width: 333px;">"""+str(linea.article_name)
+"""</td><td>"""+str(linea.cantidad)+"""</td><td>"""+str(linea.precio)+"""</td><td>"""+str(linea.descuento)+"""</
td><td>"""+str(linea.importe_con_descuento)+"""</td></tr>"""
    html = html.replace('@lines_content@', lines_content)

html_ivas = ''
json_string = json.loads(facturaObj.ivas_desglose)
for jsonObj in json_string:
    html_ivas += f"""<tr><td>{jsonObj['base_imponible']:.2f}</td><td>{jsonObj['iva']}</td><td>{jsonObj['valor_iva']:.2f}</
td><td>{jsonObj['recec']:.2f}</td><td>0.00</td></tr>"""

html = html.replace('@html_ivas@', html_ivas)
html = html.replace('@suma_importes@', f"""{facturaObj.subtotal:.2f}""")
html = html.replace('@importe_ivas@', f"""{facturaObj.importe_ivas:.2f}""")
html = html.replace('@factura_total@', f"""{facturaObj.total:.2f}""")
html = html.replace('@observaciones@', str(facturaObj.observacion))

```

```

with open(file_path, "wb") as pdf_file:
    # Convertir el HTML a PDF y guardarlo en el archivo
    pisa_status = pisa.CreatePDF(html, dest=pdf_file)

url_email = file_path
file_path = file_path.split('media')

except Exception as e:
    return json_suzdal({'message': str(e), 'status': 'error'})

if "create_and_sent" == action:
    # envio de correo al empresario
    if ';' in company_mail:
        partes = company_mail.split(';')
        for parte in partes:
            mail_company = str(parte).strip()
            enviar_correo(url_email, invoice_name, mail_company)
    else:
        enviar_correo(url_email, invoice_name, company_mail)

    # envio de correo al cliente
    if ';' in custome_mail:
        partes = custome_mail.split(';')
        for parte in partes:
            mail_company = str(parte).strip()
            enviar_correo(url_email, invoice_name, mail_company)
    else:
        enviar_correo(url_email, invoice_name, custome_mail)

rdata = {
    'status': 'ok',
    'message': 'PDF creado',
    'url': 'media'+file_path[1],
    'id': id
}

return json_suzdal(rdata)

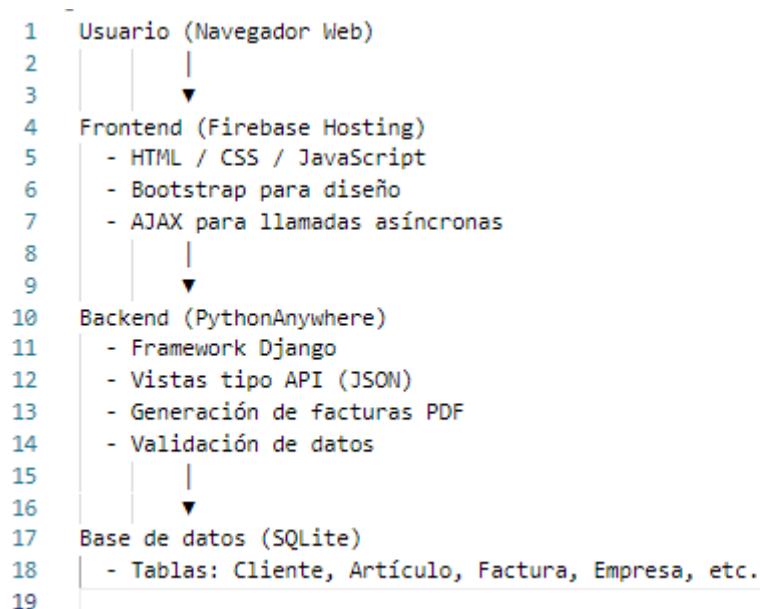
```

C. Diagramas técnicos

- C.1. Mapa de navegación de la aplicación.

1	/dashboard	→ Página principal tras el login
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

- C.2. Mapa de arquitectura frontend/backend.



D. Enlaces relevantes

- F.1. Repositorio Frontend: <https://github.com/suzdalenko-dev/FacturApp>
- F.2. Repositorio Backend: <https://github.com/suzdalenko-dev/backend-facturapps>
- F.3. Aplicación desplegada <https://facturapps.web.app/>

17. Futuras mejoras – FacturaApp

a. Funciones previstas

El desarrollo de FacturaApp ha cubierto las funcionalidades esenciales de creación, gestión y emisión de facturas. Sin embargo, se han identificado nuevas funciones clave que podrían implementarse en futuras versiones para mejorar la seguridad, eficiencia y valor añadido del sistema:

1. Autenticación segura con JWT (JSON Web Tokens)

Actualmente, el sistema utiliza autenticación básica. En futuras versiones, se implementará un sistema de autenticación basado en JWT para:

- Garantizar sesiones seguras y escalables en APIs REST.
- Integrar fácilmente con aplicaciones móviles.
- Permitir autorización granular y caducidad de tokens.

2. Aplicación móvil multiplataforma

El desarrollo de una versión móvil de FacturaApp será prioritario para mejorar la accesibilidad desde cualquier lugar. Esta app permitirá:

- Emitir facturas desde el teléfono.
- Visualizar historial y estadísticas.
- Escanear tickets o documentos para importar automáticamente datos (ver punto 3).

3. Facturas recibidas y lectura automática mediante IA + OCR

Actualmente, la app solo gestiona facturas emitidas. Una gran mejora será incorporar:

- Lectura automática de facturas recibidas (por proveedores).
- Uso de tecnologías OCR para extraer datos desde PDF o imágenes de facturas.
- Aplicación de inteligencia artificial para clasificar gastos, detectar duplicados y asignar categorías automáticamente.
- Posibilidad de importar facturas por correo electrónico o escaneando el documento físico desde la app móvil.

4. Integración con pasarelas de pago

Se contempla la integración con plataformas como Stripe, PayPal o Bizum Empresas para:

- Enviar facturas con enlace de pago.
- Marcar automáticamente una factura como “pagada” cuando se reciba el ingreso.
- Automatizar el seguimiento de cobros.

5. Exportación avanzada y compatibilidad contable

Se mejorará la exportación de datos:

- A formatos XLSX y CSV personalizables.
- Compatibilidad con software contable y asesorías.
- Envío automático de informes mensuales por correo.

b. Escalabilidad o nuevas versiones

FacturaApp ha sido diseñado desde el inicio con una arquitectura modular y separada (frontend + backend), lo cual facilita su escalabilidad futura, tanto técnica como comercial.

Escalabilidad técnica prevista:

- Migración de SQLite a bases de datos más robustas como PostgreSQL o MySQL en caso de aumento de usuarios.
- Despliegue en servidores cloud (Google Cloud, AWS) con balanceo de carga si se superan los límites de PythonAnywhere y Firebase.
- Separación en microservicios en caso de evolución a un SaaS con muchas empresas simultáneamente.

Posibles nuevas versiones:

- FacturaApp Asesorías: versión adaptada a gestores o despachos contables con multicitiente.
- FacturaApp Autónomos PRO: versión con contabilidad simplificada, libro de ingresos y gastos.
- FacturaApp Internacional: soporte para múltiples monedas, idiomas y tipos de IVA/IGIC/VAT según país.