UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Desarrollo de una aplicación web para la emisión de facturas electrónicas y gestión administrativa

Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

AUTOR/A: Renau Esteve, Sarah

Tutor/a: López Patiño, José Enrique

CURSO ACADÉMICO: 2023/2024

Resumen

El proyecto consiste en el desarrollo de una aplicación web que permitirá a autónomos y pequeñas empresas crear y gestionar los documentos mercantiles más comunes como facturas, presupuestos y albaranes. Se desarrolla específicamente para un grupo amplio de empresarios que en muchos casos no utilizan programas específicos de facturación, y que, por tanto, no tienen la capacidad de generar facturas electrónicas. En ese punto es importante destacar, que en la Ley de 2022 conocida como Crea y Crece, se establece que antes del mes de julio del año 2025 será obligatorio que todos los empresarios expidan y remitan facturas electrónicamente en las relaciones comerciales con otras empresas y autónomos.

Resum

El projecte consisteix en el desenvolupament d'una aplicación web que permetrà a autònoms i empreses menudes crear i gestionar els documents mercantils més comuns com factures, presupostos y albarans. Es desenvolupa específicament per a un ampli grup d'empresaris que en molts casos no utilitzen programes específics de facturació i que, per tant, no tenen la capacitat de generar factures electròniques. En aquest punt, és important destacar que, en la Llei de 2022 coneguda com "Crea y Crece", s'estableix que abans del mes de juliol de l'any 2025 serà obligatori que tots els empresaris expedisquen i remeten factures electrònicament en les relacions comercials amb altres empreses i autònoms.

Abstract

The project consists of developing a web application that will allow freelancers and small businesses to create and manage common commercial documents such as invoices, quotes, and delivery notes. It is specifically developed for a broad group of entrepreneurs who, in many cases, do not use specific invoicing software and, therefore, do not have the ability to generate electronic invoices. At this point, it is important to highlight that the 2022 lax known as "Crea y Crece" (Create and Glow) establish that by July 2025, all entrepreneurs will be required to issue and send invoices electronically in commercial relations with other companies and freelancers.

RESUMEN EJECUTIVO

La memoria del TFG del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación debe desarrollar en el texto los siguientes conceptos, debidamente justificados y discutidos, centrados en el ámbito de la ingeniería de telecomunicación

CONCEPT (ABET)	**CONCEPTO (traducción)**	**¿Cumple? (S/N)**	**¿Dónde? (páginas)**
1. IDENTIFY:	1. IDENTIFICAR:		
1.1. Problem statement and opportunity	1.1. Planteamiento del problema y oportunidad	S	1
1.2. Constraints (standards, codes, needs, requirements & specifications)	1.2. Toma en consideración de los condicionantes (normas técnicas y regulación, necesidades, requisitos y especificaciones)	S	4-5
1.3. Setting of goals	1.3. Establecimiento de objetivos	S	2
2. FORMULATE:	2. FORMULAR:		
2.1. Creative solution generation (analysis)	2.1. Generación de soluciones creativas (análisis)	S	9-23
2.2. Evaluation of multiple solutions and decision-making (synthesis)	2.2. Evaluación de múltiples soluciones y toma de decisiones (síntesis)	S	26-36
3. SOLVE:	3. RESOLVER:		
3.1. Fulfilment of goals	3.1. Evaluación del cumplimiento de objetivos	S	40-56
3.2. Overall impact and significance (contributions and practical recommendations)	3.2. Evaluación del impacto global y alcance (contribuciones y recomendaciones prácticas)	S	57

Índice

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1. Contexto y justificación del proyecto

En los últimos años, la digitalización ha transformado considerablemente la forma en que las empresas gestionan sus procesos administrativos y comerciales. La adopción de tecnologías digitales no solo ha mejorado la eficiencia y transparencia, sino que también ha facilitado la comunicación entre empresas. En este contexto, la facturación electrónica ha emergido como una herramienta crucial para la modernización de procesos comerciales, permitiendo una gestión más eficaz de las transacciones y reduciendo el uso del papel.

Esta solución ya ha sido acogida por las autoridades fiscales de muchos países, los cuales han empezado a imponer regulaciones a las empresas, obligándolas a emitir y recibir facturas electrónicas. En particular, a partir de julio de 2025 en España, todos los empresarios y autónomos deberán emitir facturas electrónicas en sus relaciones comerciales. Esta normativa busca asegurar que todas las transacciones sean transparentes, fácilmente auditables y libres de errores manuales.

La implementación de la facturación electrónica no solo responde a una obligación legal, sino que también ofrece múltiples beneficios para las empresas, especialmente para autónomos y pequeñas empresas que tradicionalmente han trabajado con procesos manuales y papel. No obstante, una gran parte de estos pequeños empresarios no utiliza aún programas de gestión empresarial, dificultándoles la transición hacia la digitalización de sus procesos.

Ibermedia Servicios TIC SL es una empresa con más de 20 años de experiencia en el mantenimiento, asistencia y desarrollo de servicios informáticos. Al mantenerse al día con las últimas tecnologías e innovaciones, proporciona una amplia gama de servicios que cubren todas las necesidades informáticas y de telecomunicaciones de sus clientes, ofreciéndoles una atención rápida, efectiva y personalizada.

Entre sus servicios, destaca el desarrollo de aplicaciones personalizadas para empresas que facilitan la automatización de la mayoría de las operaciones y procesos productivos, facilitando la gestión de la información en todas las áreas de la empresa.

La necesidad de este proyecto surge de la inquietud de algunos de sus clientes ante la obligación inminente de cumplir con la normativa de facturación electrónica. La creación de una aplicación web que facilite la generación de documentos mercantiles comunes, como facturas, presupuestos y albaranes, se convierte en una solución clave para estos usuarios.

1.2. Objetivos del proyecto

1.2.1 Objetivo general

Aunque ya existen otras aplicaciones que permiten esto mismo, el objetivo de este proyecto es desarrollar una aplicación web de facturación electrónica que permita a autónomos y pequeñas empresas crear, gestionar y emitir documentos mercantiles (facturas, presupuestos y albaranes) de manera eficiente, cumpliendo con la normativa vigente que entrará en vigor en julio de 2025.

1.2.2 Objetivos específicos

1. Facilitar la transición a la Facturación Electrónica:
   - Proporcionar una herramienta sencilla e intuitiva que permita emitir facturas electrónicas, asegurando el cumplimiento de los requisitos legales establecidos por la normativa española.

2. Implementar Firmas Digitales:
   - Integrar un sistema de firmas digitales que garantice la autenticidad e integridad de las facturas electrónicas.

3. Gestionar Documentos Comerciales:
   - Permitir la creación y gestión de presupuestos, albaranes y facturas.

4. Automatizar Procesos Operativos
   - Desarrollar funcionalidades que permitan la automatización de procesos, mejorando la eficiencia en la gestión de las empresas.

5. Incorporar Gestión de Clientes y Productos:
   - Facilitar la organización y categorización de clientes y productos.

6. Proporcionar Almacenamiento Seguro de Facturas:
   - Implementar un sistema de almacenamiento seguro para las facturas electrónicas, asegurando su accesibilidad y conservación durante al menos 6 años.

7. Ofrecer Soporte y Actualizaciones Continuas:
   - Proveer un sistema de soporte técnico y actualizaciones para adaptarse a cualquier cambio en la normativa y mejorar la funcionalidad de la aplicación.

8. Aumentar la Transparencia y Eficiencia de las Transacciones:
   - Facilitar la transparencia y eficiencia en las transacciones comerciales, contribuyendo a la modernización y agilidad de los procesos comerciales de los usuarios.

Se pretende completar los objetivos específicos para dar por alcanzado el objetivo general del proyecto.

Capítulo 2. Metodología

El desarrollo de este proyecto se dividió en las siguientes fases:

- **Investigación**. Se recopila y analiza información relevante para el desarrollo de la aplicación, tanto la normativa vigente sobre facturación electrónica en España como los requisitos del formato Facturae. Además, se investigan otras aplicaciones similares disponibles en el mercado para identificar funcionalidades necesarias. También se realizan entrevistas a autónomos y pequeñas empresas para entender sus necesidades.

- **Análisis**. Se estudian los datos obtenidos en la fase anterior, se definen los requisitos detallados del sistema y se diseña la arquitectura de la aplicación. Es en esta fase en la que se crean diagramas para visualizar la interacción de los usuarios con la aplicación, además de diseñar la estructura de la base de datos.

- **Implementación.** Se trata de la fase más costosa, en la que se desarrolla la aplicación de acuerdo con los requisitos y diseños definidos. Se prepara la base de datos, se realiza la programación del Backend y Frontend y se integra el sistema de firmas digitales.

- **Pruebas y validación.** Se efectúan las pruebas necesarias para asegurar que la aplicación funcione correctamente, cumpla con los requisitos y normativas, y sea usable por los usuarios finales. Se realizan tanto pruebas unitarias como pruebas de integración para verificar que los diferentes componentes de la aplicación funcionan correctamente en conjunto. Finalmente se realizan pruebas de usabilidad por usuarios finales para identificar y corregir errores.

Capítulo 3. Marco Teórico

3.1 Facturación Electrónica

La facturación electrónica no es más que el proceso mediante el cual se emite, recibe y almacena una factura en formato digital, garantizando su autenticidad e integridad. El concepto de factura permanece inalterado con respecto a una factura en papel, es decir, sigue siendo un justificante de determinadas operaciones comerciales o financieras. Sin embargo, la principal diferencia es que, para su emisión, es necesario el consentimiento del destinatario [1].

La implementación de la facturación se ha convertido en un componente crucial en la modernización de los procesos empresariales, siendo Italia el país pionero en implantarlo en el sector privado, extendiéndose por el resto de los países de la unión europea, que han fijado como objetivo su obligatoriedad en un plazo inferior a cinco años. Mientras que, en el ámbito de las administraciones públicas, más del ochenta por cien de estos países, incluido España, ya han implementado este proceso de forma obligatoria [2].

Los beneficios de esta nueva forma de facturación son múltiples:

- **Eficiencia**: La automatización del proceso de facturación puede reducir errores humanos y acelerar el ciclo de facturación.
- **Ahorro de costos**: Disminuye el uso de papel y los costos de impresión, almacenamiento y envío de las facturas, lo que la hace también una solución muy favorable con el medio ambiente.
- **Transparencia**: Facilita el cumplimiento de las normativas fiscales y mejora la transparencia en las transacciones comerciales.
- **Acceso y almacenamiento**: Facilita el acceso y gestión de las facturas, ya que permite un almacenamiento seguro y accesible desde cualquier lugar.

3.2 Normativa legal y estándares

Las normas que debe cumplir tanto una factura de papel tradicional como una factura electrónica están reguladas por el Real Decreto 1619/2012, de 30 de noviembre [3]. Dentro del reglamento se establecen aspectos como los siguientes:

- Obligación y excepciones de expedir factura.
- Los tipos de factura: completa y simplificada
- El contenido de los distintos tipos de factura
- Concepto y requisitos de una factura electrónica: autenticidad e integridad.
- Plazos de expedición
- Otras facturas
- Particularidades
- Conservación de facturas y otros documentos

Por otro lado, es la Ley Crea y Crece, aprobada el 15 de septiembre de 2022 [4], la que establece la obligatoriedad de expedir y remitir facturas electrónicas a todos los autónomos y empresas [5].

Existen unos estándares técnicos para la estructuración de estas facturas [1]:

- **XML (Extensible Markup Language)**: es un metalenguaje desarrollado para almacenar datos de forma organizada [6]. Es, además, el formato más comúnmente utilizado para la estructuración de datos en facturas electrónicas.
- **UBL (Universal Business Language)**: Es un estándar basado en XML diseñado específicamente para el ámbito empresarial con el fin de representar documentos como facturas [7].

- **Facturae**: Es el formato oficial para facturas electrónicas en España, basado en XML, que garantiza el cumplimiento de la normativa nacional [8].

3.3 Firmas digitales

Una firma digital es un mecanismo criptográfico que permite garantizar la autenticidad, integridad y no repudio de un documento electrónico. [9] [10] Es esencial para asegurar que las facturas electrónicas no han sido alteradas desde su emisión y que el emisor es quien afirma ser.

Tecnología:

Los certificados digitales, a partir de los cuales se genera la firma digital, están basados en la criptografía de clave pública o asimétrica. En este tipo de criptografía se utiliza un par de claves (una pública y una privada) para crear y verificar la firma. Este proceso garantiza que cualquier cambio en el documento después de la firma será detectable [9].

Importancia en la facturación electrónica:

Aunque dentro del Real Decreto 1612/2012, de 30 de noviembre, la firma digital no aparece cómo obligatoria para las facturas electrónicas, si es uno de los mecanismos que se puede utilizar para asegurar la integridad y autenticidad, lo cual si es un requisito. Esto último hace que la implementación de una firma digital adecuada sea crucial para la aceptación legal de la factura en transacciones comerciales. [3]

3.4 Gestión de Clientes y Productos en Aplicaciones Web

Gestión de Clientes:

La gestión de clientes en aplicaciones web implica la organización y almacenamiento de información relacionada con los clientes, como datos de contacto o historial de compras [11]. Esto es esencial para mejorar los servicios ofrecidos y la relación con los clientes, además de optimizar estrategias de venta entre otros.

Gestión de Productos:

La gestión de productos incluye la categorización, almacenamiento y actualización de la información sobre los productos o servicios que una empresa ofrece. Esto es esencial para facilitar el proceso de ventas y mejorar la experiencia del cliente.

Importancia en la Facturación Electrónica:

Integrar la gestión de clientes y productos en una aplicación de facturación electrónica permite automatizar parte del proceso de facturación, asegurando que los datos sean precisos y estén actualizados. Además, mejora la eficiencia operativa al centralizar la información necesaria para la emisión de facturas, presupuestos y albaranes.

3.5 Herramientas y tecnologías utilizadas

Laravel

Laravel es un framework de desarrollo web de código abierto basado en PHP, diseñado para hacer el desarrollo de aplicaciones más sencillo y eficiente. Laravel sigue el patrón de diseño MVC (Modelo-Vista-Controlador), lo que permite una organización clara y modular del código. Más adelante en la memoria se detallará como funciona esta arquitectura. Entre las características más destacadas de este framework se encuentran un sistema de enrutamiento flexible y sencillo, un ORM (Eloquent) para el manejo intuitivo de bases de datos, un sistema de plantillas (Blade) y otras herramientas integradas para tareas como la autenticación, gestión de sesiones y manejo de caché.

¿Por qué Laravel?

Laravel es el framework que utilizan los desarrolladores en la empresa con la que he realizado este proyecto, y una de las principales razones por las que optaron por utilizarlo es su vasta comunidad. Laravel cuenta con una comunidad de desarrolladores muy activa y colaborativa, lo que se traduce en una gran cantidad de recursos disponibles, como foros, tutoriales, y otras bibliotecas de terceros. Tener este tipo de comunidad es especialmente valioso cuando te enfrentas a problemas o dudas, ya que es probable que otros desarrolladores hayan enfrentado situaciones similares y compartido soluciones.

Además, Laravel se destaca por su documentación detallada y su facilidad para integrarse con otros [12] sistemas y herramientas, lo que lo convierte en una de las mejores opciones para desarrollar una aplicación web de facturación electrónica, donde la seguridad, la escalabilidad y la mantenibilidad son aspectos cruciales.

PHP

PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto ampliamente utilizado en el desarrollo web. Fue diseñado específicamente para la creación de páginas web dinámicas y aplicaciones web, permitiendo a los desarrolladores generar contenido dinámico que pueda interactuar con bases de datos y otros servicios en el servidor [13].

PHP es un lenguaje interpretado del lado del servidor, lo que significa que el código es procesado por el servidor antes de ser enviado al navegador del usuario, cosa que no ocurre, por ejemplo, con JavaScript. Esto lo convierte en una herramienta poderosa para crear aplicaciones web interactivas y funcionales [14].

¿Por qué PHP?

La respuesta es sencilla, como se ha comentado anteriormente, su uso es fundamental porque Laravel está basado en PHP, por tanto, todas sus funcionalidades y características se construyen sobre este lenguaje de programación. Al utilizar Laravel, estoy aprovechando la solidez y flexibilidad de PHP para desarrollar una aplicación web de facturación electrónica eficiente, segura y escalable. PHP, al ser un lenguaje ampliamente soportado, asegura que mi aplicación pueda ejecutarse en una gran variedad de entornos y servidores.

JavaScript/jQuery/Ajax

JavaScript, junto con HTML y CSS, es una de las tecnologías fundamentales de la World Wide Web (WWW). Es un lenguaje de programación interpretado, es decir, un lenguaje que no necesita ser compilado previamente a su utilización [15], definido como orientado a objetos basado en prototipos. Su función principal es habilitar la creación de páginas web interactivas, utilizado principalmente del lado del cliente, y es una pieza clave en el desarrollo de aplicaciones web modernas [16]. En este proyecto, JavaScript se ha complementado con su principal biblioteca multiplataforma, jQuery.

jQuery es una biblioteca que simplifica la interacción con los documentos HTML, facilitando la manipulación del DOM (Document Object Model), la gestión de los eventos y el desarrollo de animaciones [17]. Además, jQuery permite integrar de manera sencilla la funcionalidad de Ajax, una técnica de desarrollo web que permite realizar actualizaciones parciales en una página sin necesidad de recargarla por completo [18]. Esto mejora significativamente la interactividad, velocidad y usabilidad de la aplicación o página web, proporcionando una experiencia de usuario más fluida y eficiente.

¿Por qué JavaScript, jQuery y Ajax?

En primer lugar, la elección de JavaScript se debe a que es un lenguaje esencial en el desarrollo web moderno, ampliamente soportado por todos los navegadores. Por su parte, jQuery fue seleccionada por su capacidad para simplificar tareas comunes, como la manipulación del DOM, lo que reduce el código JavaScript necesario. Otra razón es su integración con Ajax, ya que esta funcionalidad es crucial para una aplicación donde es necesario realizar actualizaciones en tiempo real sin interrumpir el flujo de trabajo del usuario.

HTML/CSS/Bootstrap

HTML (HyperText Markup Language) es el lenguaje estándar utilizado para estructurar el contenido de las páginas web [12]. (CITA DEVELOPER MOZILLA) Define la organización y la presentación de texto, imágenes, enlaces formularios y otros elementos dentro de una página a través de etiquetas.

CSS (Cascading Style Sheets) es un lenguaje utilizado para controlar la apariencia y el diseño visual de una página web. Mientras que HTML define el contenido y la estructura, CSS se encarga de darle estilo, como colores, fuentes, alineación, etc.

Bootstrap es un framework multiplataforma o conjunto de herramientas de código abierto, utilizado para el diseño de sitios y aplicaciones web. Ofrece una amplia variedad de plantillas de diseño predefinidas que incluyen tipografía, formularios, botones, cuadros y otros muchos elementos visuales basados en HTML y CSS. Además, Bootstrap incorpora extensiones de JavaScript que amplían sus capacidades para crear interfaces más dinámicas e interactivas.

¿Por qué HTML/CSS/Bootstrap?

El uso de HTML y CSS en este proyecto es fundamental, ya que son las tecnologías base para el desarrollo de la interfaz de usuario. Por su parte, Bootstrap es un framework ampliamente adoptado y bien documentado, lo que facilita su implementación. Su enfoque en el desarrollo front-end permite crear interfaces atractivas y funcionales de manera rápida, sin necesidad de desarrollar cada componente desde cero. Además, su sistema de diseño responsivo permite que la aplicación pueda adaptarse automáticamente a diferentes tamaños de pantalla y dispositivos.

MySQL/HeidiSQL

MySQL es uno de los sistemas de gestión de bases de datos relacionales más populares y ampliamente utilizados en el mundo. Es de código abierto y está diseñado para manejar grandes volúmenes de datos de manera eficiente, lo que lo convierte en una opción ideal para aplicaciones que requieren un almacenamiento confiable y de acceso rápido a la información, como es el caso de mi aplicación de facturación electrónica [19].

Por su parte, HeidiSQL es un software libre y de código abierto que permite la administración de bases de datos MySQL. Se caracteriza por tener una interfaz intuitiva y tener una alta capacidad de simplificación de tareas como la creación, modificación y consulta de bases de datos [20].

¿Por qué MySQL y HeidiSQL?

La principal razón por la cual se ha escogido el uso de MySQL y HeidiSQL es porque fueron las utilizadas durante el grado, en la asignatura de Sistemas Telemáticos para la Gestión de la Información, por lo que ya estaba familiarizada y facilitó mis tareas. Además, la compatibilidad de MySQL con Laravel es excelente, lo que permite una integración fluida y directa entre el framework y la base de datos.

XAMPP

XAMPP es un paquete de software libre que proporciona una solución completa para desarrollar aplicaciones web en un entorno local. Incluye Apache (un servidor web), MySQL (el sistema de gestión de bases de datos utilizado en este proyecto) y PHP (el lenguaje de programación mayormente utilizado y en el que se basa el framework Laravel) [21].

¿Por qué XAMPP?

XAMPP está diseñado para ser fácil de instalar y usar, lo que lo convierte en una excelente opción para desarrolladores que buscan configurar rápidamente un entorno de desarrollo local. Simula de manera precisa el entorno de producción, lo que asegura que la aplicación funcione perfectamente cuando sea desplegada en un servidor real. Además, es totalmente compatible con las otras tecnologías y herramientas utilizadas para desarrollar este proyecto.

Facturae-PHP

Facturae-PHP es un paquete escrito puramente en PHP, desarrollado por José Miguel Moreno. Permite generar facturas electrónicas cumpliendo el formato Facturae, que es el formato oficial en España [22].

¿Por qué Facturae-PHP?

Este paquete cumple con todas las normativas establecidas para la emisión de facturas electrónicas en España y se encuentra en continuo desarrollo y mejoras. El uso de este paquete ha simplificado significativamente la generación de facturas en el formato Facturae.

GitHub

GitHub es una forja (plataforma de desarrollo colaborativo), desarrollada para alojar proyectos utilizando sistema de control de versiones Git. Permite a los desarrolladores gestionar y almacenar sus proyectos de software en repositorios, pudiendo colaborar con otros desarrolladores manteniendo un historial detallado de los cambios en el código [23]. En este caso, ha sido crucial la descarga de la aplicación GitHub Desktop, que permite realizar los comandos de Git, como Push y Pull, de forma más sencilla y optimizada.

¿Por qué GitHub?

El uso de GitHub en este proyecto ha permitido mantener el control de versiones y sobre todo, actualizar los cambios entre el entorno de desarrollo local y el entorno en el servidor real, ya que es totalmente compatible con el servidor Plesk, que es el utilizado en Ibermedia, la empresa en la que se ha desarrollado este proyecto. También cabe destacar su interfaz intuitiva, y las ventajas que ofrecen sus plataformas como GitHub Desktop.

Capítulo 4. Análisis y Diseño del Sistema

Tras realizar la tarea de investigación y recolección de datos, se obtiene la información necesaria para establecer los requisitos que debe cumplir el sistema, y el modo en que se organizarán y representarán sus diferentes componentes.

4.1 Requisitos del sistema

4.1.1 Requisitos funcionales

Los requisitos funcionales describen las funcionalidades específicas que el sistema debe implementar para cumplir con los objetivos del proyecto. Estos son los aspectos clave que se considera que la aplicación debe cubrir:

- **Gestión de usuarios:**
  - Registro y autenticación de usuarios.
- **Gestión de Empresas:**
  - Registro de empresa/autónomo.
  - Mantenimiento y almacenamiento de la información.
  - Personalización de series de facturación e información de pago.
- **Gestión de Clientes:**
  - Crear, editar y eliminar registros de clientes.
  - Consultar el historial de transacciones de cada cliente.
- **Gestión de Productos**:
  - Crear, editar y eliminar productos y categorías de productos.
- **Creación de Documentos Mercantiles:**
  - Generar presupuestos, albaranes y facturas.
  - Soporte para generar facturas en formato Facturae.
- **Firmado de Facturas:**
  - Integración con certificados digitales para firmar electrónicamente las facturas.
- **Exportación y Envío de Facturas:**
  - Exportar facturas en formato XML y PDF.
  - Enviar facturas por correo electrónico desde la aplicación.
- **Exportación y Envío de otros Documentos Mercantiles:**
  - Exportación presupuestos y albaranes en PDF.
  - Enviar presupuestos y albaranes por correo electrónico desde la aplicación.

4.1.2 Requisitos No Funcionales

Los requisitos no funcionales describen las características que definen la calidad del sistema:

- **Seguridad**:
  - Autenticación y autorización segura para el acceso a la aplicación
  - Encriptación de datos sensibles, como contraseñas.
- **Usabilidad**:
  - Interfaz de usuarios intuitiva y fácil de usar.
- **Rendimiento**:
  - Respuesta rápida en la generación y firma de facturas.
- **Compatibilidad**:
  - Compatible con los navegadores web más utilizados (Chrome, Firefox, Safari).
  - Soporte para dispositivos móviles y tablets.
- **Mantenimiento**:
  - Código estructurado y documentado para facilitar el mantenimiento y la actualización del sistema.

- Test unitarios y de integración para asegurar la funcionalidad del código.

4.2 Diagramas y Modelado

4.2.1 Diagramas de Flujo

Para estructurar y definir los procesos clave de la aplicación, se han creado varios diagramas de flujo que representan tanto las interacciones de los usuarios con el sistema como los procesos internos del mismo.

El objetivo principal de estas interacciones es que sean lo más sencillas e intuitivas posible. Se busca que la aplicación pueda ser utilizada por cualquier tipo de usuario, independientemente de su experiencia previa. Al simplificar al máximo las interacciones, se facilita un uso correcto de la aplicación y se evita que los usuarios se sientan abrumados por opciones innecesarias.

- **Registro de usuarios**

  Los usuarios tendrán la posibilidad tanto de registrarse en la aplicación como de registrar otros usuarios. En caso de no estar registrado, deberá introducir los gastos de una nueva empresa.

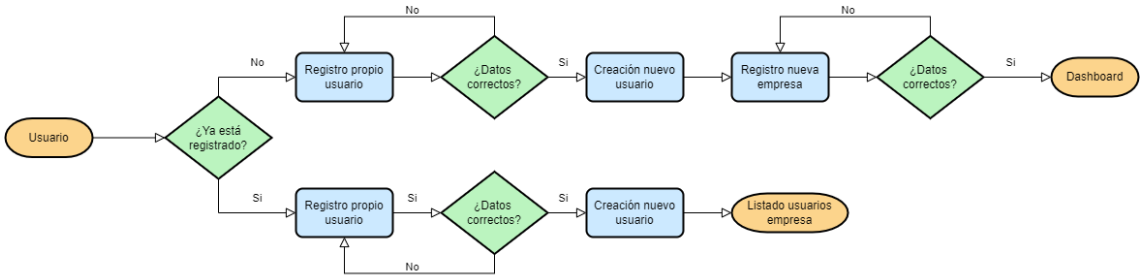


Figura 1. Diagrama de flujo para el registro de usuarios.

- **Inicio de sesión del usuario**

  Pueden darse dos escenarios en los cuales sea necesario el inicio de sesión de un usuario. El primer escenario sería el inicio de sesión desde la página principal de la aplicación.

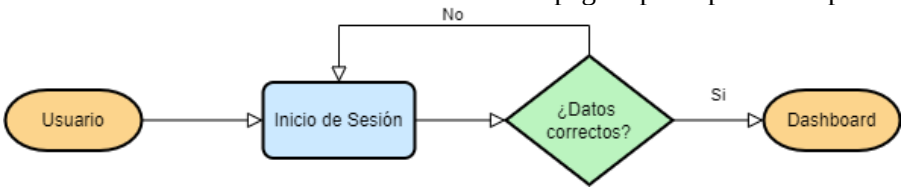


Figura 2. Diagrama de flujo para el inicio de sesión desde la página de inicio.

El segundo escenario ocurre cuando el usuario ha dejado la aplicación abierta durante un período prolongado sin realizar ninguna interacción. En este caso, se cerrará automáticamente la sesión por motivos de seguridad, requiriendo que el usuario inicie sesión nuevamente para poder acceder a la página en la que se encontraba previamente.

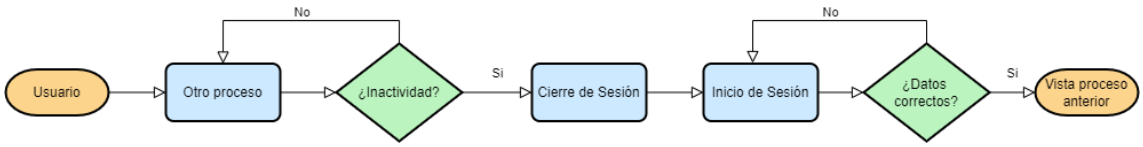


Figura 3. Diagrama de flujo para el inicio de sesión tras inactividad.

- **Sesión de un usuario**

  Una vez efectuado el inicio de sesión y accedido al 'Dashboard', el usuario puede acceder a las otras secciones de la aplicación. Estas son acceder al listado de los documentos

mercantiles (presupuestos, albaranes y facturas) que haya creado dentro de la aplicación, los clientes almacenados, el listado de productos y categorías de productos y acceder a la vista que contiene la información de la propia empresa para poder modificarla.

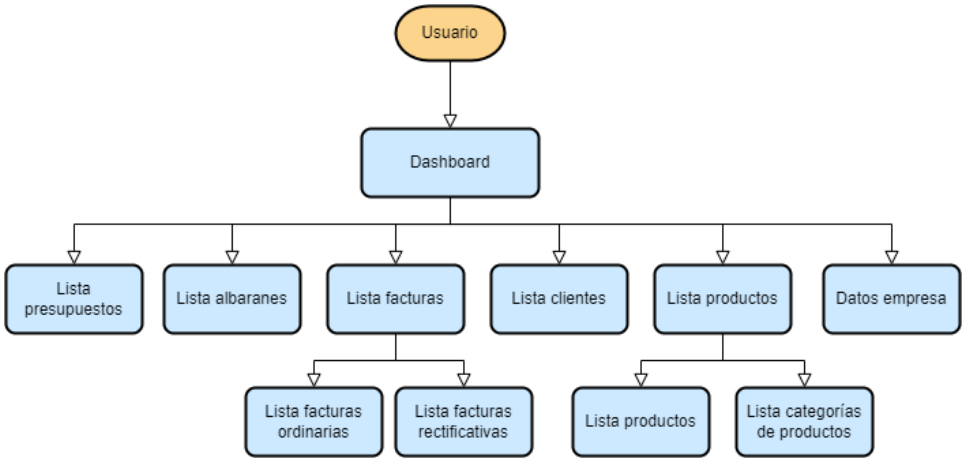


Figura 4. Diagrama de flujo de una sesión de un usuario.

❖ Lista presupuestos

Desde el listado de presupuestos, será posible realizar distintas acciones sobre cada uno, dependiendo del estado en el que se encuentre, además de crear un nuevo presupuesto. Estas acciones incluirán: acceder a editar un presupuesto, siempre que nunca haya sido enviado al cliente; cambiar manualmente su estado; enviarlo por correo electrónico; descargarlo en formato PDF; visualizarlo; y, si ya ha sido emitido y todavía no ha vencido, generar un albarán o una factura a partir de él.

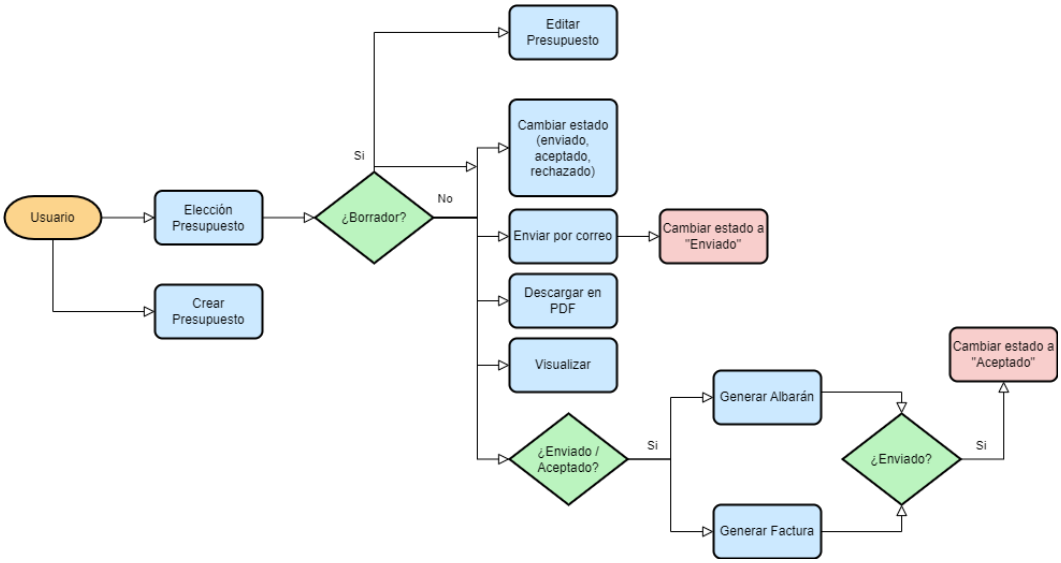


Figura 5. Diagrama de flujo en el que un usuario accede al listado de presupuestos

❖ Edición/creación presupuestos

Algunas de las acciones disponibles tanto para un presupuesto ya creado como para uno nuevo serán comunes, como añadir detalles del presupuesto, aplicar descuentos, retenciones o impuestos, y adjuntar archivos. Sin embargo, durante la creación del presupuesto se introducirá la información sobre el cliente, y las fechas relevantes. Podemos observarlo en el siguiente flujograma.

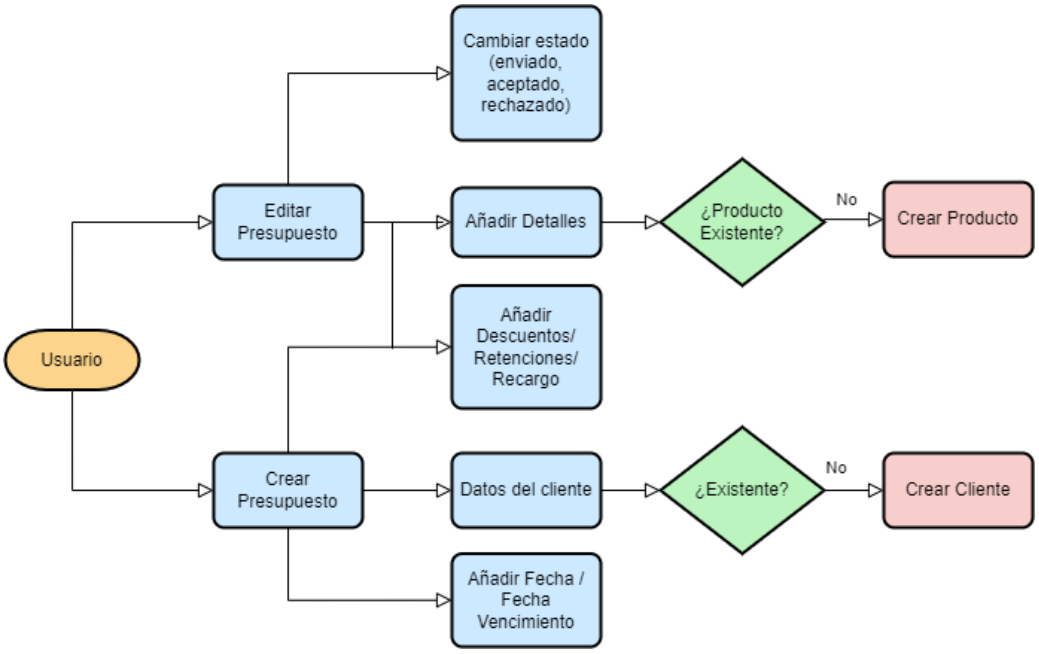


Figura 6. Diagrama de flujo en el que un usuario accede a 'Editar Presupuesto' o 'Nuevo Presupuesto'

❖ Visualización presupuestos

Existirá una ventana en la aplicación que permitirá visualizar toda la información de un presupuesto tal como aparecería en el PDF final. Desde esta vista, se podrán realizar varias acciones, como cambiar el estado del presupuesto, descargarlo en formato PDF, eliminarlo o acceder a la edición si se encuentra en estado 'borrador'. Además, si el presupuesto no está en estado 'borrador' ni 'rechazado', será posible generar un albarán o una factura directamente desde esta ventana.

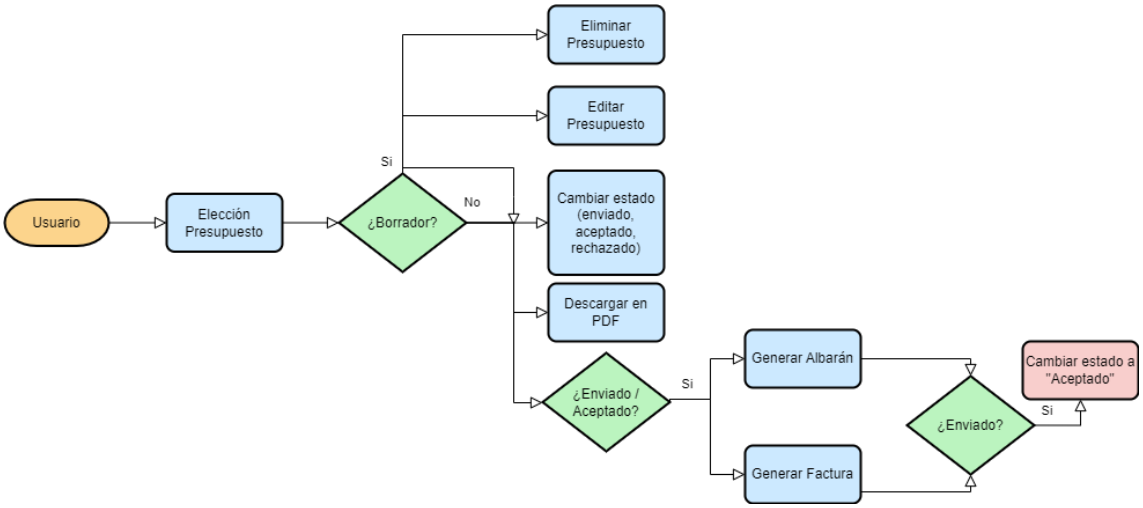


Figura 7. Diagrama de flujo en el que un usuario accede a la vista 'Visualizar Presupuesto'.

❖ Lista albaranes

En la lista de albaranes, se podrán realizar diversas acciones según las necesidades del usuario. Será posible crear un nuevo albarán, o en el caso de seleccionar uno existente, editarlo, generar una factura a partir del mismo, enviarlo por correo electrónico, descargarlo en formato PDF y acceder a visualizar su contenido.

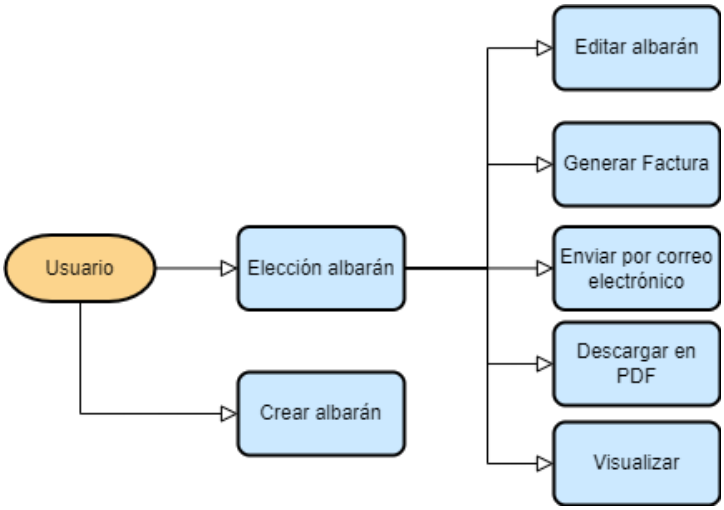


Figura 8. Diagrama de flujo en el que el usuario al listado de albaranes

- **Edición/creación de albarán**

  En la ventana de creación o edición de un albarán, los usuarios podrán realizar diversas acciones para gestionar y completar la información necesaria. Entre las opciones disponibles, el usuario podrá añadir detalles del albarán (el sistema verificará si el producto o servicio ya existe en la base de datos), aplicar descuentos, retenciones o impuestos, y crear una factura basada en el albarán. Además, durante la creación del albarán se introducirá la información sobre el cliente, y las fechas relevantes. Podemos observarlo en el siguiente flujograma.

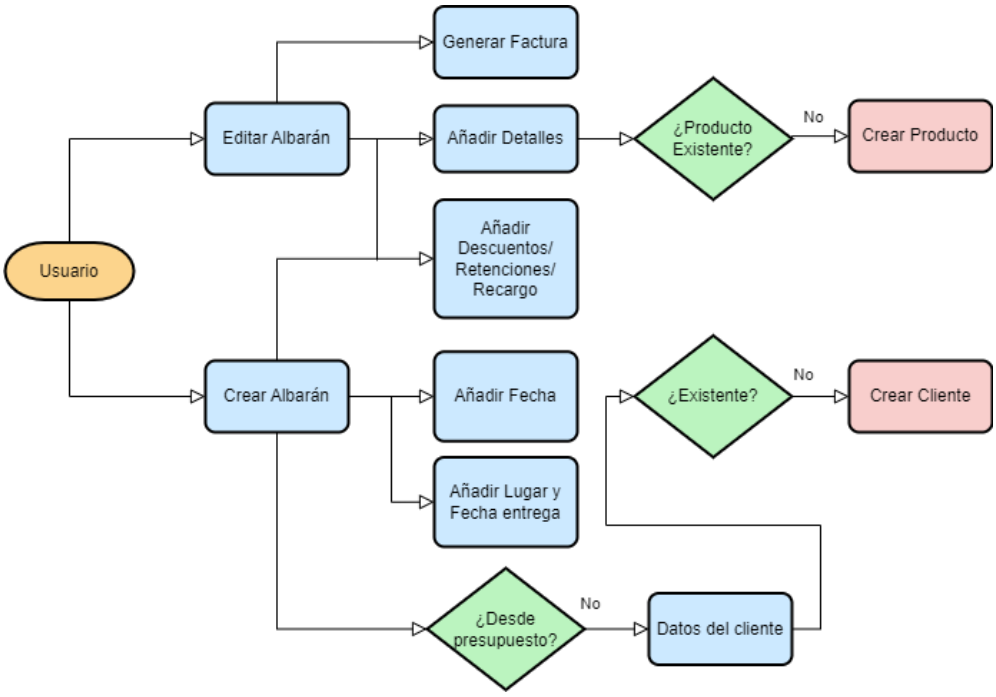


Figura 9. Diagrama de flujo en el que un usuario accede a 'Editar albarán o 'Nuevo albarán'.

- **Visualización albarán**

En la aplicación, existirá una ventana que permitirá visualizar toda la información de un albarán tal como aparecería en el PDF final. Desde esta vista, será posible realizar varias

acciones: editar sus detalles cuando sea necesario, descargarlo en formato PDF para archivarlo o enviarlo, eliminarlo si ya no es requerido o se ha generado por error, y generar una factura a partir del albarán, siempre que los detalles de este no hayan sido incluidos previamente en ninguna factura.

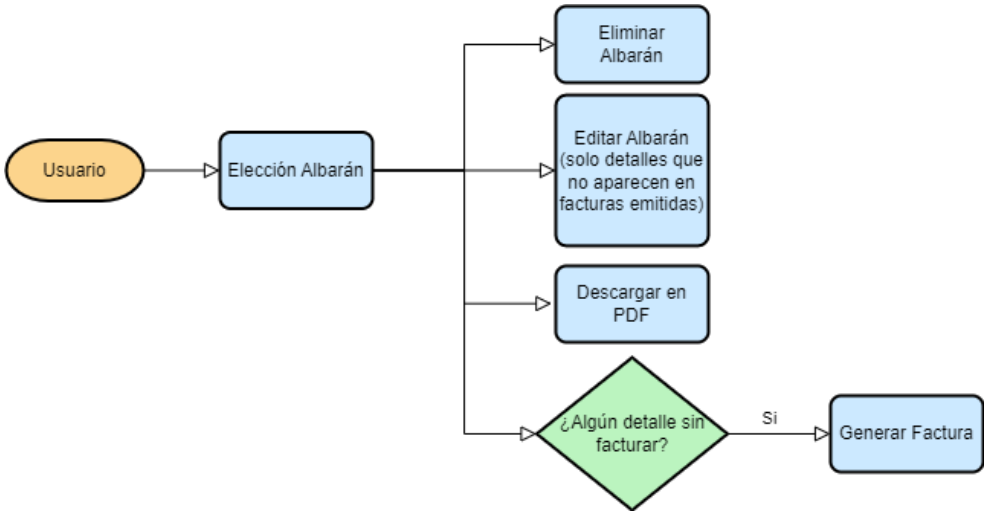


Figura 10. Diagrama de flujo en el que un usuario accede a la vista 'Visualizar Albarán'.

- **Lista facturas / facturas rectificativas**

  Desde el listado de facturas y de facturas rectificativas, será posible realizar diversas acciones sobre cada una, dependiendo del estado en el que se encuentre, además de la opción de crear una nueva factura. Entre estas acciones se incluirán: acceder a editar una factura, siempre que no haya sido enviada al cliente; cambiar su estado manualmente; enviarla por correo electrónico; descargarla en formato PDF; y visualizarla. Algunas de estas acciones podrán desencadenar procesos adicionales en el sistema. Por ejemplo, enviar una factura por correo directamente desde la aplicación automáticamente cambiará su estado a 'Emitida' y generará la factura electrónica correspondiente, lo que bloqueará la posibilidad de realizar más cambios sobre es factura.

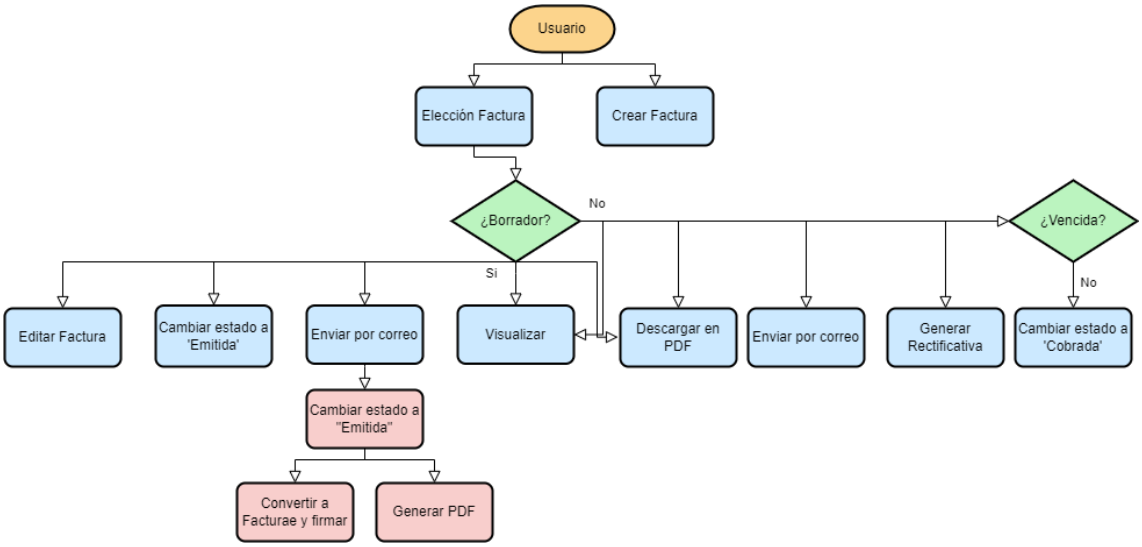


Figura 11. Diagrama de flujo en el que el usuario al listado de facturas.

- **Edición/creación de facturas y facturas rectificativas**

  Este diagrama presenta ciertas similitudes con el de "edición/creación presupuestos". Las acciones disponibles para una factura, tanto si ya ha sido creada como si es nueva,

incluirán: añadir detalles de la factura, aplicar descuentos, retenciones o impuestos, y adjuntar archivos. No obstante, durante la creación de la factura, se deberá introducir información adicional, como los datos del cliente y las fechas relevantes.

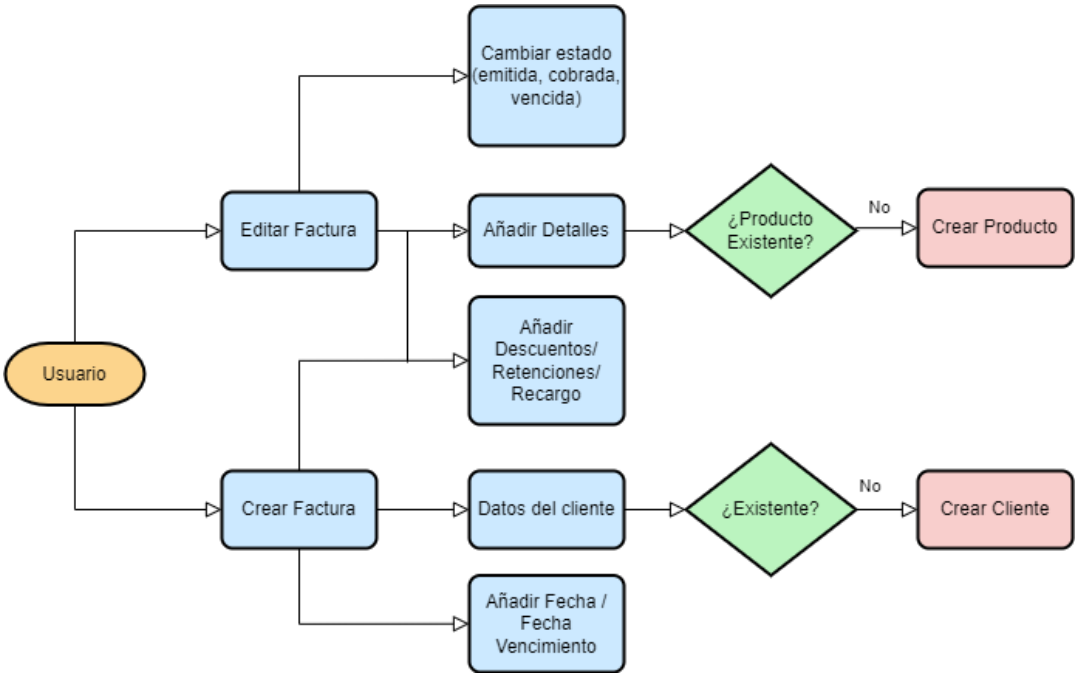


Figura 12. Diagrama de flujo en el que un usuario accede a 'Editar factura o 'Nueva factura'.

❖ **Visualización factura**

Existirá una ventana en la aplicación que permitirá visualizar toda la información de una factura tal como aparecería en el PDF final desde la cual se podrán realizar varias acciones según el estado de la factura. En caso de estar en 'Borrador', estas acciones serán: acceder a la vista 'Editar Factura', cambiar su estado a 'emitida', lo que supondrá generar su factura electrónica y firmarla, eliminarla, descargarla en formato PDF o generar directamente la factura electrónica. En caso de existir la factura electrónica pero no estar firmada, se podrá firmar directamente y descargar.

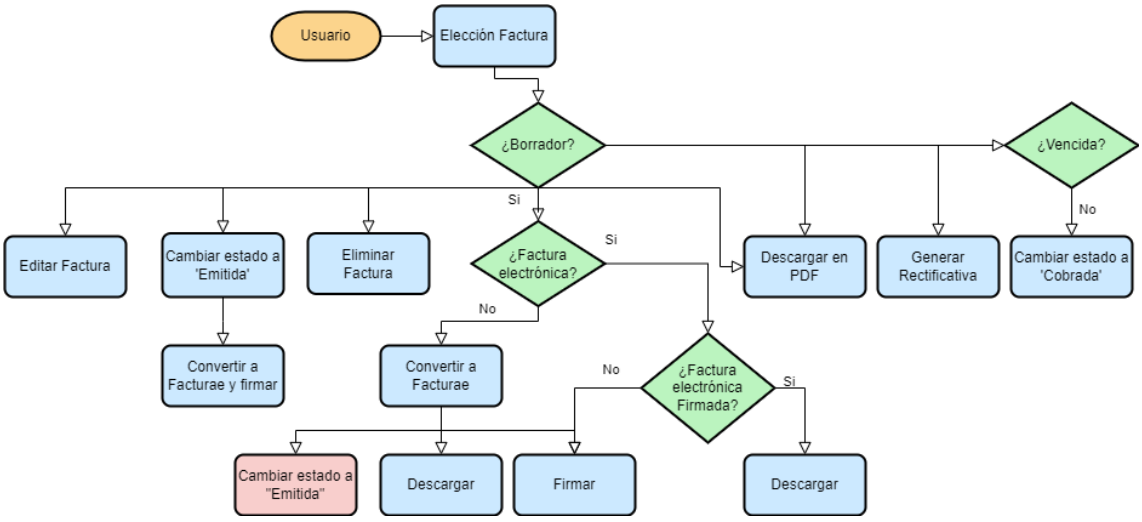


Figura 13. Diagrama de flujo en el que un usuario accede a la vista 'Visualizar Factura.

❖ **Lista clientes**

En la lista de clientes, se podrán realizar diversas acciones según las necesidades del usuario. Será posible crear un nuevo cliente, o en el caso de seleccionar uno existente, editarlo o desactivarlo.

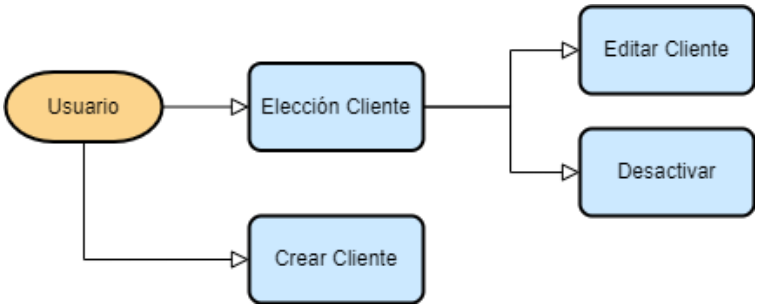


Figura 14. Diagrama de flujo en el que el usuario al listado de clientes.

❖ **Editar / creación clientes**

En la ventana de creación o edición de un cliente, los usuarios podrán realizar diversas acciones para gestionar y completar la información necesaria. Entre las opciones disponibles, el usuario podrá añadir, editar o eliminar contactos del cliente y también gestionar las actividades con el mismo, es decir, añadir, editar y eliminar.

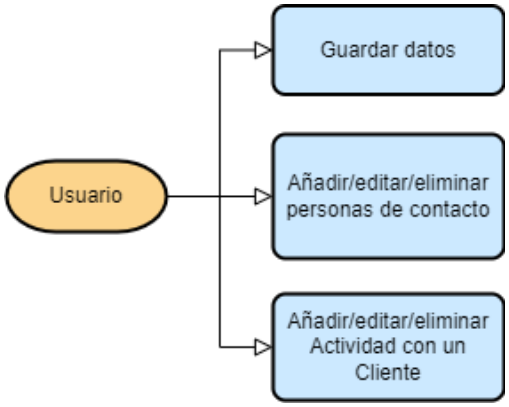


Figura 15. Diagrama de flujo en el que un usuario accede a 'Editar Cliente o 'Nuevo Cliente.

❖ **Lista categorías de productos y lista de productos**

En las listas de categorías de productos y de productos, se podrá llevar a cabo diferentes acciones según las necesidades del usuario. Estas incluirán la creación de una nueva categoría o producto, así como la edición o desactivación de un recurso existente.

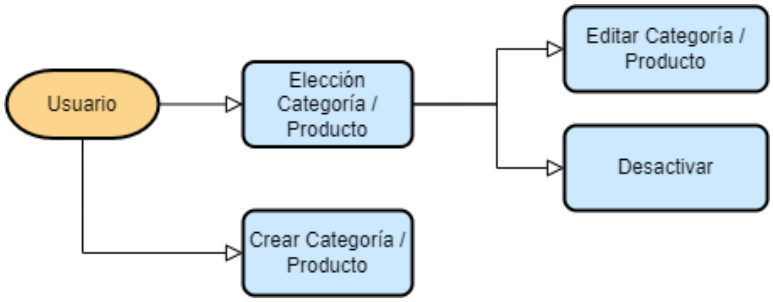


Figura 16. Diagrama de flujo en el que el usuario al listado de categorías de productos y productos.

❖ **Editar / creación categorías de producto y productos**

Las acciones disponibles para las categorías de productos y productos serán limitadas: únicamente será posible guardar la información necesaria y activar o desactivar el recurso.

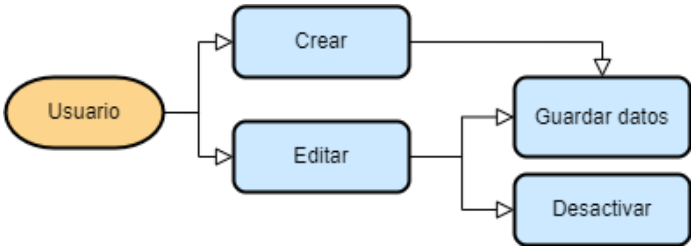


Figura 17. Diagrama de flujo en el que un usuario accede a 'Editar/Nueva Categoría de producto' o ''Editar/Nuevo Producto".

❖ **Datos empresa**

En la vista 'mantenimiento', se podrán realizar diversas acciones para la administración y configuración de la aplicación. Estas incluyen guardar la información de la propia empresa, añadir, editar o eliminar las series de facturación y los métodos de pago, gestionar los archivos disponibles (con la opción de visualizarlos y descargarlos), así como añadir nuevos usuarios o acceder a la edición de los ya existentes.

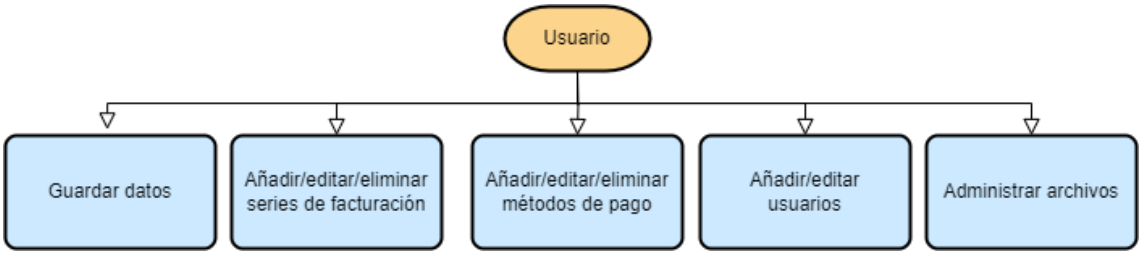


Figura 18. Diagrama de flujo en el que un usuario accede a la vista 'Mantenimiento'.

❖ **Edición / Creación Usuario**

En la ventana de creación o edición de un usuario, se podrán realizar varias acciones para gestionar y completar la información requerida. Entre las opciones disponibles, será posible añadir, editar o eliminar actividades asociadas al usuario en cuestión, así como activarlo o desactivarlo según sea necesario.

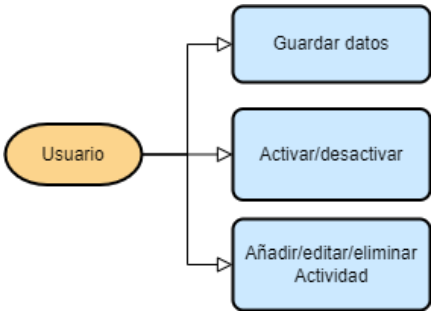


Figura 19. Diagrama de flujo en el que un usuario accede a 'Editar/Nuevo Usuario'.

4.2.2 Diagrama de la Tarea Programada

Para gestionar el control del vencimiento de las facturas, se implementará una tarea programada que se ejecutará directamente en el servidor. Esta tarea revisará todas las facturas emitidas de la aplicación que aún no hayan sido marcadas como ‘Cobrada’ y cuya fecha de vencimiento ya haya expirado, cambiando automáticamente su estado a ‘vencida’.

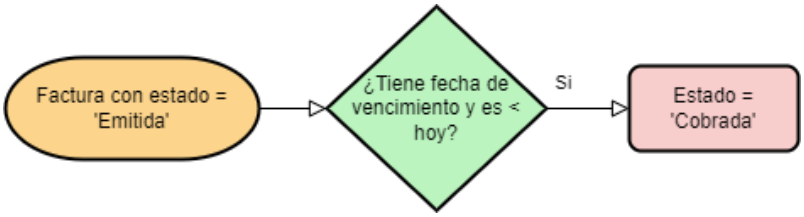


Figura 20. Diagrama de la tarea programada

4.2.3 Clases

Una vez identificadas las principales funciones que deberá cumplir la aplicación, se pasa a diseñar la base de datos. En este apartado se esboza un esquema general con las tablas que contendrá la base de datos, con todos sus campos y las relaciones que existirán entre las distintas tablas. Este esquema es clave para entender cómo se organiza internamente el código y como interactúan las distintas partes del sistema.

Es importante señalar que, igual que sucede durante todo el proceso de desarrollo, este esquema también ha experimentado ajustes y mejoras a medida que surgieron nuevas necesidades y se resolvieron desafíos técnicos.

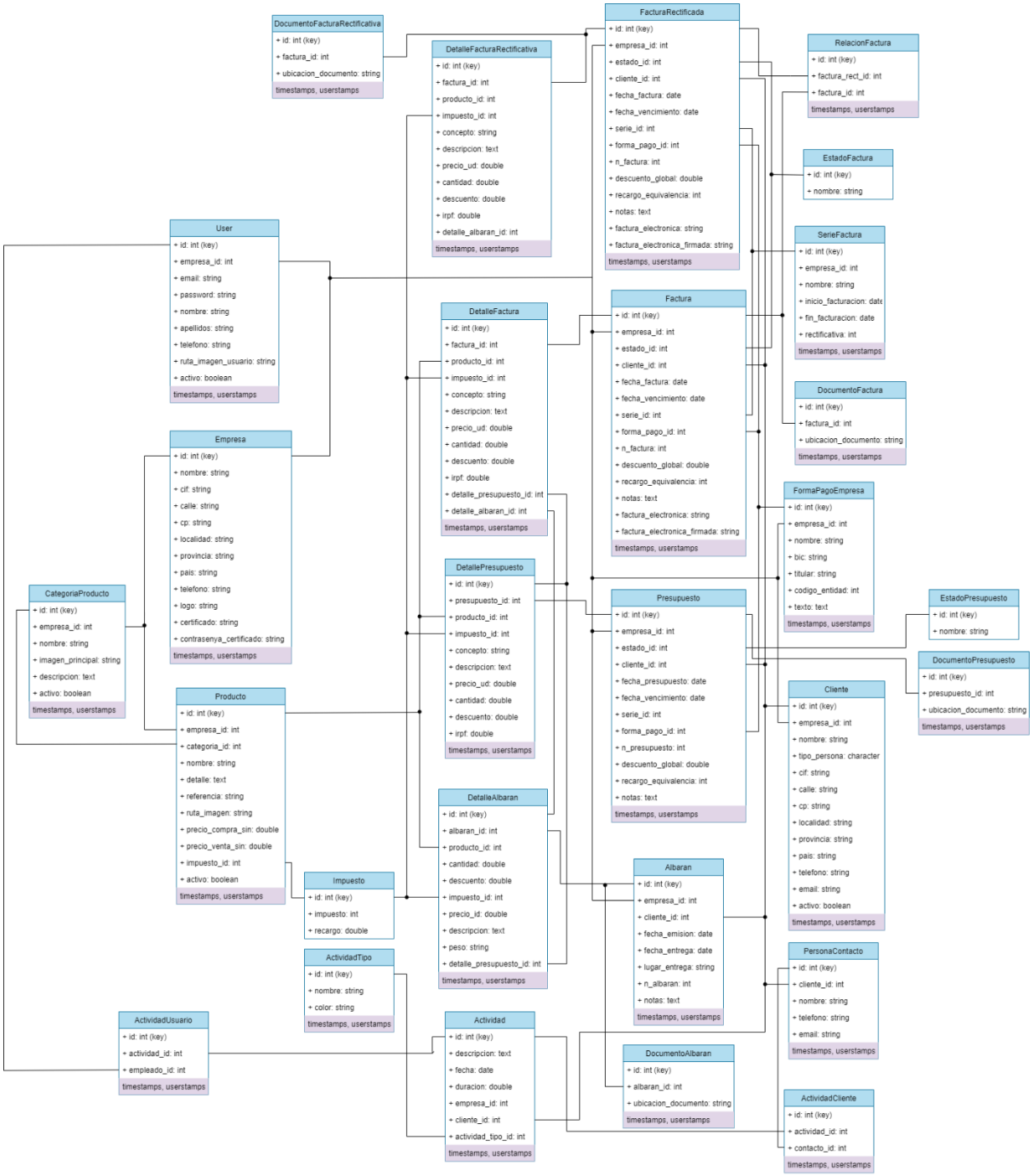


Figura 21. Esquema general con las clases de la aplicación.

Con tal de simplificar el diagrama, ya de por si un tanto lioso, se han omitido el sentido de las relaciones entre las tablas, todas de (1:n). Poniendo de ejemplo las clases ‘User’ y ‘Empresa’, la relación sería 1 desde el campo “id” de ‘Empresa’ y n desde el campo ‘empresa_id’ de ‘User’, de manera que a cada ocurrencia de la clase ‘Empresa’ le pueden corresponder varios ‘User’, mientras que a cada ocurrencia de la segunda clase solo le corresponde una de la primera. Es decir, mientras que una empresa puede tener muchos usuarios, un usuario solo pertenece a una empresa.

Por otro lado, los campos timestamps y userstamps, que se repiten en muchas tablas, son realmente un conjunto de cuatro campos distintos. Timestamps engloba los campos 'created_at' y 'updated_at', que son de tipo DateTime con la fecha de creación y última fecha de modificación respectivamente. Por su parte, userstamps engloba los campos 'created_by', que sería de tipo int con el id del usuario que creó el recurso, y 'updated_by' también de tipo int con el id del último usuario que modificó el recurso. Estos campos sirven para seguir la trazabilidad de cambios efectuados sobre los recursos.

La base de datos se organiza en tres grupos de clases. El primer grupo corresponde a las clases gestionadas exclusivamente por el administrador de la aplicación, el segundo a las clases principales que constituyen el núcleo funcional del sistema y por último a las que se crean a partir de otras clases para complementarlas.

1. Clases gestionadas por el administrador

Estas clases solo pueden ser creadas o modificadas directamente a través de migraciones o manualmente en la base de datos:

- **Impuesto**: Esta clase contiene un listado con los distintos porcentajes de IVA aplicables en España y sus respectivos recargos de equivalencia.
- **ActividadTipo**: Incluye un listado de tipos de actividades relacionadas con la interacción con clientes, como llamadas, reuniones o envío de correos electrónicos.
- **EstadoFactura**: Enumera los distintos tipos de estados en los que se puede encontrar una factura, como 'Emitida', 'Cobrada', 'Vencida' y 'Borrador'. Cabe destacar que una factura solo puede ser editada mientras se encuentre en estado 'Borrador', lo que impide modificaciones posteriores a su emisión y asegura la transparencia en las transacciones comerciales.
- **EstadoPresupuesto**: Contiene un listado con los distintos estados posibles para un presupuesto, incluyendo 'Borrador', 'Enviado', 'Aceptado', 'Rechazado' y 'Vencido'. En este caso, los estados sirven principalmente para mejorar la organización administrativa.

2. Las clases principales del sistema

- **Empresa**: Es la clase de la cual parten las demás. Se encarga de gestionar la información relevante de las empresas o autónomos, incluyendo datos esenciales para la emisión de facturas, como nombre, dirección fiscal, NIF/CIF, y otros datos legales.
- **User**: Laravel tiene esta clase predefinida, que maneja la autenticación de usuarios con campos como el email y la contraseña. En este proyecto, se han añadido campos adicionales como 'activo', 'ruta_imagen_usuario' o 'teléfono', permitiendo a los usuarios registrar información adicional sobre sí mismos. Estos podrán ser creados tanto por ellos mismos como por otros usuarios ya registrados.
- **CategoriaProducto**: Esta clase contiene un listado con las categorías de productos creadas por cada empresa. Las categorías pueden estar activas o inactivas según la necesidad de la empresa, permitiendo una organización eficiente de los productos y servicios.
- **Producto**: Constituye el listado de productos y servicios que cada empresa registrada crea. Incluye datos esenciales como el precio de compra, el precio de venta o el impuesto asociado. Los productos pueden ser categorizados, y pueden activarse y desactivarse según sea necesario.
- **FormaPagoEmpresa**: Esta clase gestiona las diferentes formas de pago aceptadas por cada empresa.
- **Cliente**: Los clientes son creados por los usuarios de la aplicación y pueden ser completados con la información necesaria para la emisión de facturas. Los datos incluyen

el código o número de identificación fiscal, la dirección, tipo de entidad (persona física o jurídica), nombre, entre otros.

- **Actividad**: En esta clase se registran las distintas interacciones con los clientes. Algunas actividades son generadas automáticamente por el propio sistema, como la emisión de una factura o el alta de un cliente. Otras pueden ser creadas manualmente por un usuario, como una llamada con un cliente, incluyendo detalles como la fecha y la duración de la interacción.
- **Presupuesto**: Esta clase gestiona los presupuestos creados por los usuarios de la empresa. Cada presupuesto incluye detalles como la información del cliente, el estado del presupuesto, descuentos generales o algún tipo impositivo específico. También es posible generar albaranes o facturas a partir de un presupuesto aprobado, como se indica en la **¡Error! No se encuentra el origen de la referencia.**.
- **Albarán**: Se encarga de gestionar los albaranes e incluyen detalles como la fecha y dirección de entrega.
- **Factura**: La clase 'Factura' gestiona la creación y almacenamiento de las facturas generadas por la empresa. Incluye información detallada sobre el cliente, impuestos aplicables y el estado de la factura, además de la ubicación en la que se almacena la factura electrónica una vez emitida y la correspondiente firmada. Las facturas pueden generarse a partir de presupuestos o albaranes, o manualmente.
- **FacturaRectificada**: Esta clase maneja las facturas rectificativas, que son emitidas para corregir errores en facturas previamente emitidas.

3. Clases complementarias

Las clases complementarias son fundamentales para la operatividad de la aplicación, ya que amplían y detallan la funcionalidad de las clases principales. Estas clases no podrían existir de forma independiente, ya que dependen directamente de la información generada por las clases principales.

- **PersonaContacto**: Esta clase gestiona las posibles personas de contacto asociadas a un cliente. Incluye información como nombre, email y teléfono, permitiendo una gestión más personalizada y eficiente de la relación con los clientes.
- **ActividadUsuario**: Esta clase se encarga de vincular las actividades con los usuarios de la aplicación. Es esencial, ya que una única actividad puede estar relacionada con varios usuarios. Por ejemplo, en el caso de una reunión con un cliente, es posible que varios empleados participen en la misma, lo que justifica la necesidad de esta clase para registrar todas las interacciones de manera precisa.
- **ActividadCliente**: Similar a la clase anterior, 'ActividadUsuario' gestiona las relaciones entre las actividades y las distintas personas de contacto involucradas en las mismas. Esto permite un seguimiento detallado de las interacciones que no solo involucran a la empresa, sino también a diferentes contactos dentro de una organización cliente.
- **DetallePresupuesto**: Esta clase contiene toda la información detallada de los productos o servicios incluidos en un presupuesto. Los datos registrados incluyen el producto, su precio, el impuesto asociado, posibles retenciones y descuentos. La suma de todos estos elementos constituye el total del presupuesto.
- **DetalleAlbaran**: Similar a 'DetallePresupuesto', esta clase almacena la información detallada de los productos o servicios entregados que figuran en un albarán. Incluye datos como el producto, la cantidad entregada y cualquier observación relevante sobre la entrega. En caso de crear un albarán a partir de un presupuesto, estos quedarían relacionados a partir del campo 'detalle_presupuesto_id' de esta clase.
- **DetalleFactura**: Esta clase gestiona los detalles específicos de los productos o servicios facturados, incluyendo el precio, impuestos aplicables, descuentos y

retenciones. Cada línea de la factura está representada en esta clase, permitiendo una descomposición detallada del total facturado.

- **DetalleFacturaRectificativa**: Similar a 'DetalleFactura', pero en este caso se refiere a los detalles de las facturas rectificativas, que son emitidas para corregir errores en facturas previamente emitidas. Esta clase es esencial para mantener la exactitud y legalidad en la facturación.
- **DocumentoPresupuesto**: Almacena los documentos adjuntos a un presupuesto. Esto permite que toda la información relacionada con un presupuesto esté centralizada y facilita el acceso.
- **DocumentoAlbaran**: Esta clase guarda los documentos adjuntos a un albarán. Puede incluir comprobantes de entrega o cualquier otro documento.
- **DocumentoFactura**: Similar a las anteriores, esta clase almacena los documentos asociados a una factura.
- **DocumentoFacturaRectificativa**: Al igual que 'DocumentoFactura', pero específicamente para las facturas rectificativas.
- **RelacionFactura**: Esta clase es utilizada para guardar las relaciones entre las facturas originales y sus respectivas facturas rectificativas. Es esencial para rastrear la historia de cualquier factura que haya sido modificada, garantizando la transparencia y el cumplimiento normativo.

Con la base de datos presentada en la **Figura *21***, es posible gestionar de manera completa la aplicación de facturación electrónica. Esta base de datos central permite manejar todos los aspectos esenciales del sistema.

No obstante, para garantizar un registro preciso de todas las modificaciones y eliminaciones que se realicen en la aplicación, se ha planteado la creación de dos bases de datos adicionales. Estas bases de datos complementarias están diseñadas específicamente para llevar un control exhaustivo de los cambios realizados y asegurar la trazabilidad de las operaciones en el sistema.

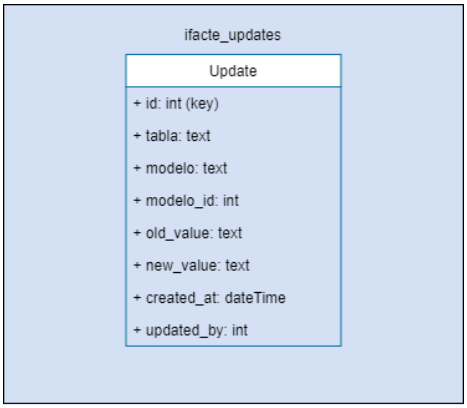


Figura 22. Esquema general de la base de datos updates.

En la **Figura *22***, se puede observar la estructura de la primera base de datos complementaria, que está compuesta por una única clase denominada 'Update'. Esta clase tiene como propósito almacenar toda la información relacionada con las modificaciones efectuadas en la base de datos principal. Los datos registrados incluyen el nombre de la tabla afectada, el nombre del modelo o clase, el ID del recurso modificado, los valores anteriores y los actualizados, también la fecha de la modificación y el ID del usuario que realizó el cambio. De esta manera, se mantiene un historial detallado de cada alteración, permitiendo una auditoría completa y asegurando la transparencia en la gestión de datos.

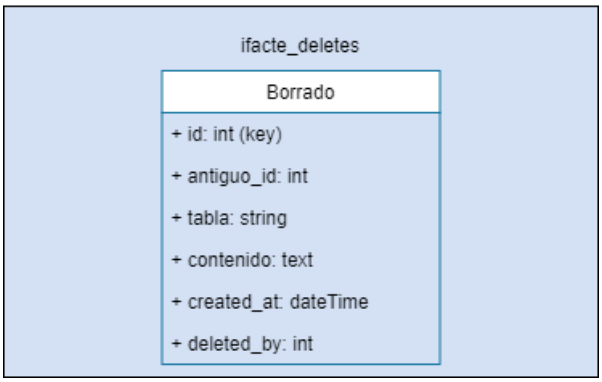


Figura 23. Esquema general de la base de datos deletes.

Por otro lado, la **Figura *23*** muestra la organización de la base de datos denominada ‘deletes’. Esta base también se compone de una única clase, ‘Borrado’, que se encarga de registrar la información relacionada con los recursos eliminados. Los datos almacenados incluyen el ID antiguo del recurso, la tabla a la que pertenecía, su contenido en el momento de la eliminación, la fecha en que se realizó el borrado y el ID del usuario que lo ejecutó.

Capítulo 5. Implementación

En esta sección se detallará el proceso de implementación de la aplicación, describiendo cómo se ha llevado a cabo el desarrollo de cada uno de los módulos y funcionalidades, desde la lógica del backend hasta la interfaz de usuario.

Como se ha mencionado anteriormente, este proyecto se desarrollará utilizando el framework Laravel.

5.1 Configuración del entorno

Esta fase del proceso requiere una configuración cuidadosa del entorno, un paso fundamental para iniciar el desarrollo con Laravel. A continuación, se presenta una descripción detallada de esta configuración.

5.1.1 Instalación XAMPP y PHP

El primer paso consiste en descargar e instalar XAMPP en el ordenador local, donde trabajaremos antes de migrar el proyecto al servidor remoto donde se alojará. En este proyecto, se utiliza XAMPP con la versión de PHP 8.2.0.

5.1.2 Crear alias

Dado que en el entorno local en el que se desarrolla el proyecto se utilizan varias versiones de PHP, se ha tenido que crear alias en el CMD para ejecutar cada versión según lo requiera cada proyecto del entorno local. En este caso, al utilizar la versión 8.2, se han creado los alias 'php82' y 'composer82' para utilizar PHP 8.2 y Composer con la versión de PHP 8.2 respectivamente, facilitando así la administración de las distintas versiones de PHP en el entorno de desarrollo.

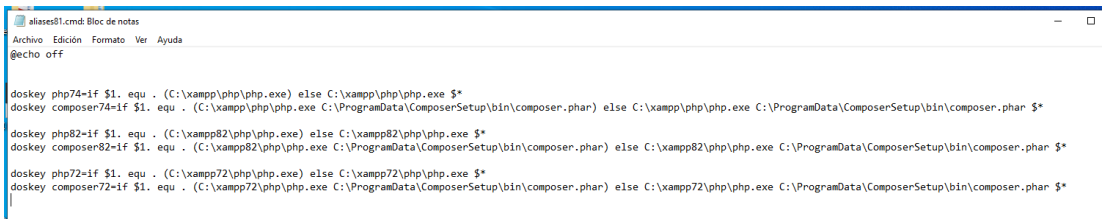


Figura 24. Alias en el CMD

5.1.3 Instalación Laravel

Una vez instalado XAMPP, disponemos de todas las dependencias necesarias para ejecutar nuestro proyecto de forma local y gestionar la base de datos sin depender de servidores externos. Con esta infraestructura lista, procederemos a instalar Laravel y para ello optaremos por usar Composer, un gestor de paquetes para PHP que proporciona un formato estándar para manejar dependencias y librerías.

Se procede a la instalación y creación de un nuevo proyecto de Laravel en la versión 9 con el siguiente comando:

```
composer82 create-project laravel/laravel:^9.0 example-app
```

5.1.4 Estructura de la aplicación

Laravel utiliza una arquitectura MVC (Modelo-Vista-Controlador), un patrón de diseño que organiza una aplicación web de forma clara y separada en tres componentes:

1. **Modelo (Model)**: Se utilizan para interactuar con la base de datos, definiendo la estructura de los datos, las relaciones entre diferentes tablas, y las operaciones que se pueden realizar sobre esos datos.

2. **Vista (View)**: Son archivos de plantilla que se encargan de renderizar el contenido HTML, CSS y JavaScript de la aplicación.
3. **Controlador (Controller)**: recibe las solicitudes del usuario (a través de rutas), interactúa con el modelo para procesar los datos necesarios y luego selecciona la vista para representar los datos al usuario.

Tras la creación del proyecto con el comando anteriormente mencionado, podemos observar esta arquitectura en la **Figura *25***.

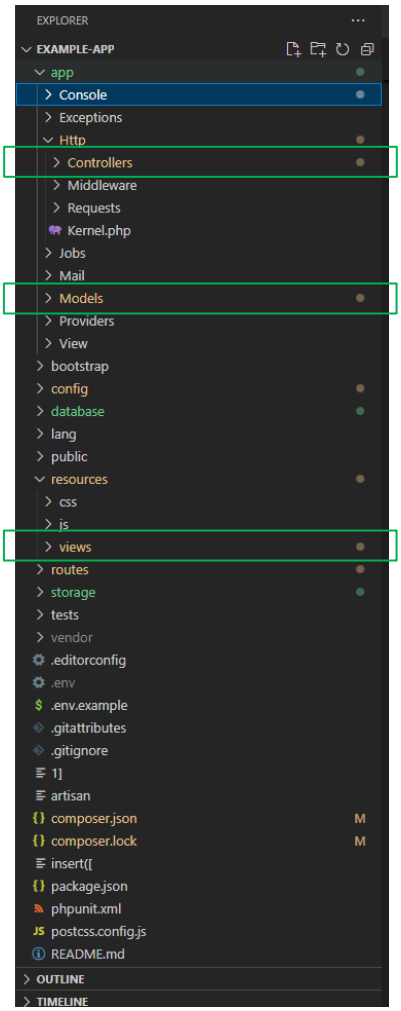


Figura 25. Arquitectura del proyecto

5.2 Desarrollo del backend

Tal y como se ha explicado anteriormente, el backend de la aplicación se basa en la configuración y desarrollo de los modelos, controladores y rutas, que son los componentes clave para manejar la lógica del proyecto, gestionar las interacciones con la base de datos y definir cómo se procesan las solicitudes del usuario.

5.2.1 Implementación de la base de datos

Como se explicó en el marco teórico, la gestión de las bases de datos se realizará mediante HeidiSQL. Primero, crearemos las bases de datos directamente en HeidiSQL, configurándolas en el mismo puerto donde se activa el servicio en XAMPP, que en este caso es el puerto 3307.

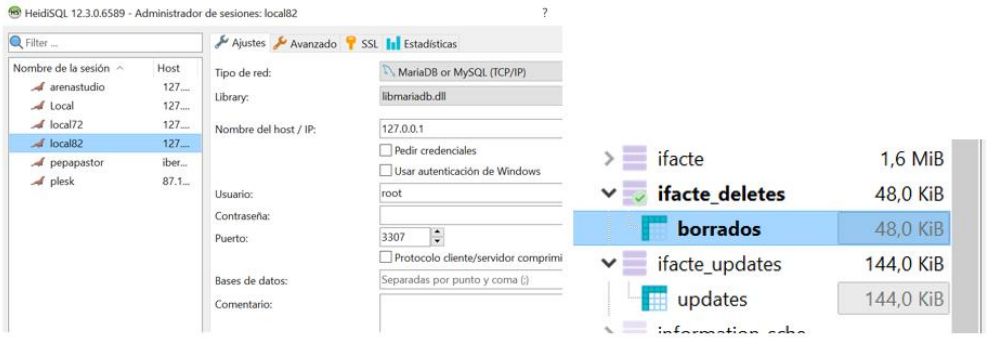


Figura 26. Creación bases de datos.

Ahora que las bases ya están creadas, pero la principal está vacía, el siguiente paso es conectar la base de datos con el proyecto y generar las migraciones necesarias para crear las tablas definidas en los esquemas anteriores.

Para establecer la conexión entre las bases de datos y nuestro proyecto, modificaremos los archivos '.env' y 'database.php'. En el archivo ''.env' configuramos las variables de los parámetros básicos de la base de datos principal.

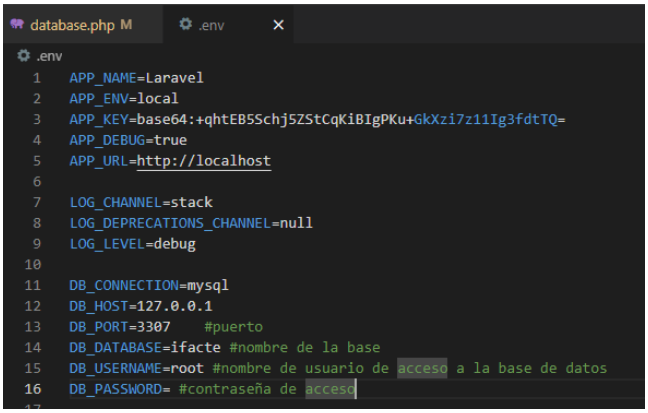


Figura 27. Configuración archivo '.env'.

Por otro lado, en el archivo 'database.php' se configuran los parámetros necesarios para la gestión de las ediciones y eliminaciones en las bases de datos.

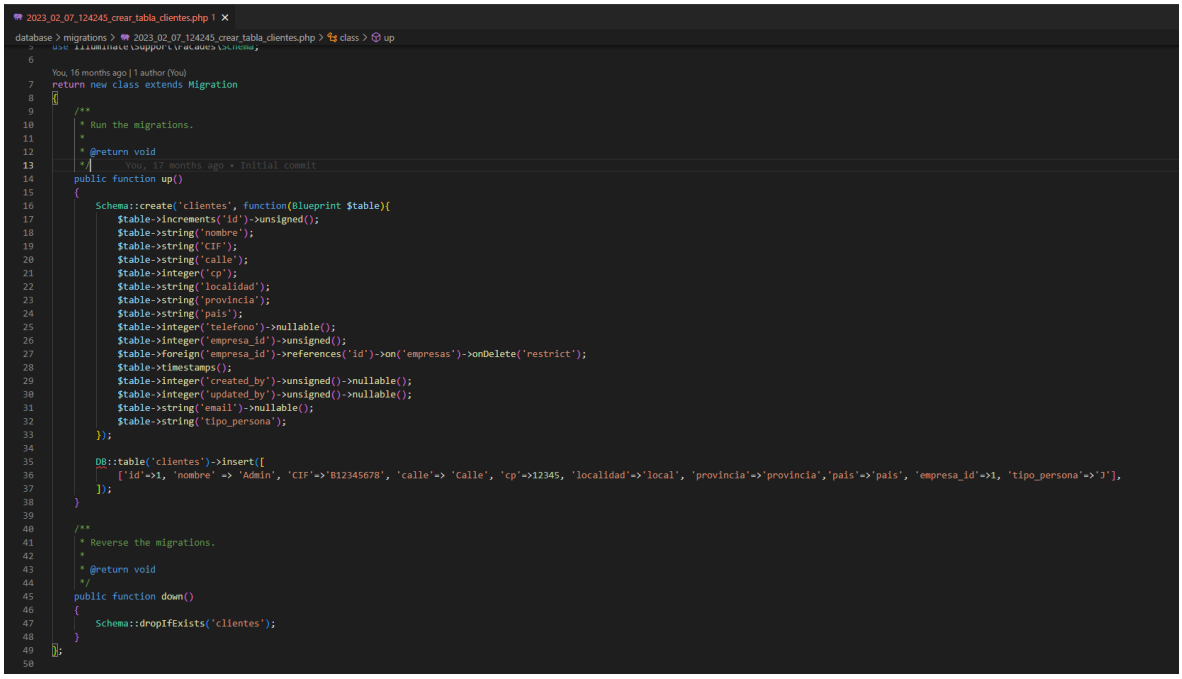
Figura 28. Configuración archivo 'database.php'.

Una vez configuradas todas las bases, se procede a generar las migraciones necesarias utilizando el comando de Laravel:

```
php82 artisan make:migration nombre_migración
```

Una migración contiene dos métodos: up y down. El primero se utiliza para agregar tablas y columnas nuevas, mientras que el método down se utiliza par revertir las operaciones realizadas por el método up.

Pongamos como ejemplo la migración de la clase 'Cliente' para ver cómo se configuran las columnas y las relaciones entre otras tablas:



```php
2023_02_07_124245_crear_tabla_clientes.php 1 ×
database > migrations > 2023_02_07_124245_crear_tabla_clientes.php > class > up
use Illuminate\Support\Facades\Schema;

You, 16 months ago | 1 author (You)
return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     *  You, 17 months ago • Initial commit
     */
    public function up()
    {
        Schema::create('clientes', function(Blueprint $table){
            $table->increments('id')->unsigned();
            $table->string('nombre');
            $table->string('CIF');
            $table->string('calle');
            $table->integer('cp');
            $table->string('localidad');
            $table->string('provincia');
            $table->string('pais');
            $table->integer('telefono')->nullable();
            $table->integer('empresa_id')->unsigned();
            $table->foreign('empresa_id')->references('id')->on('empresas')->onDelete('restrict');
            $table->timestamps();
            $table->integer('created_by')->unsigned()->nullable();
            $table->integer('updated_by')->unsigned()->nullable();
            $table->string('email')->nullable();
            $table->string('tipo_persona');
        });

        DB::table('clientes')->insert([
            ['id'=>1, 'nombre' => 'Admin', 'CIF'=>'B12345678', 'calle'=> 'Calle', 'cp'=>12345, 'localidad'=>'local', 'provincia'=>'provincia','pais'=>'pais', 'empresa_id'=>1, 'tipo_persona'=>'J'],
        ]);
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('clientes');
    }
};
```

Figura 29. Migración 'crear_tablas_clientes'.

Para crear la tabla 'clientes', utilizamos el generador de esquemas con el método 'Schema:.create()'. Si necesitamos modificar una tabla existente en lugar de crearla, utilizamos el método 'Schema::table()'. A continuación, creamos las columnas de la tabla utilizando comandos como '$table->'tipo_de_dato'('nombre_columna')'.

Para definir una relación entre distintas tablas, primero se crea la columna correspondiente y luego se establece la relación utilizando los siguientes comandos:

- 'foreign': especifica la columna en la que se definirá la relación.
- 'references': indica la columna de la otra tabla con la cual se relacionará.
- 'on': define la tabla con la que se establece la relación
- 'onDelete', especifica la acción a realizar sobre la primera columna si el recurso de la otra tabla es eliminado. En este caso, utilizamos 'restrict' para evitar la eliminación del recurso 'empresa' mientras exista una relación con los clientes, garantizando así la integridad referencial.

Una vez todas las migraciones están creadas, procedemos a ejecutar la migración con el comando:

```
php82 artisan migrate
```

Y finalmente podemos observar el esquema final de la base de datos facilitada por el software 'MySQL Workbench':

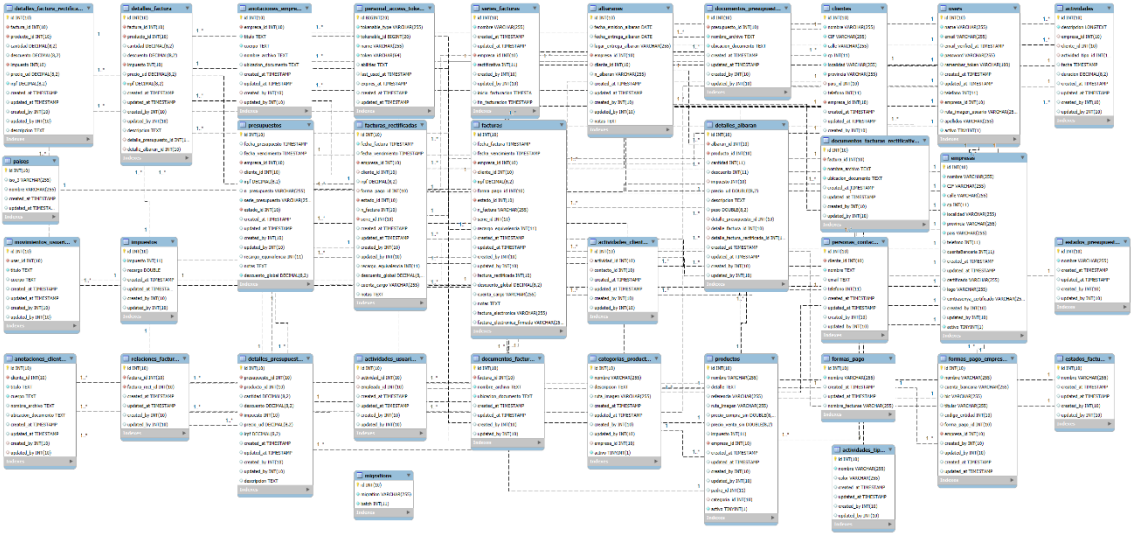


Figura 30. Esquema de la base de datos 'ifacte' facilitada por 'MySQL Workbench'.

5.2.2 Implementación de las funciones

- **MODELOS**

Una vez configurada la base de datos, el siguiente paso es crear los modelos correspondientes a cada tabla. En estos modelos se definirán las funciones específicas que permitirán recuperar y gestionar los datos necesarios.

Se crean inicialmente 3 modelos:

- **Update**: relacionado con la tabla 'updates' de la base 'ifacte_updates'.
- **Borrado**: relacionado con la tabla 'borrados' de la base 'ifacte_deletes'.
- **BaseModel**: aunque este modelo no está directamente asociado a ninguna tabla, todos los demás modelos de la aplicación heredarán de él. Su principal función es registrar automáticamente cada modificación o eliminación de recursos en la base 'updates' y 'deletes'.

Seguidamente se crean el resto de los modelos definidos en el apartado 4.2.2.

Además, durante la fase de implementación surgió la necesidad de crear dos modelos adicionales para la correcta generación de facturas electrónicas utilizando Facturae-PHP:

- **FormaPago**: Este modelo define las distintas formas de pago permitidas en Facturae-PHP junto con su correspondiente denominación en este paquete. De este modo, cuando una empresa añade nuevas formas de pago, debe seleccionar la que corresponde en Facturae-PHP.
- **País**: Incluye dos campos: nombre e iso_3, ya que uno de los requisitos de la factura electrónica es el campo 'CountryCode', que utiliza el código ISO 3166-1 alfa-3. Para rellenar esta tabla con todos los países y sus respectivos códigos, se implementó un comando que hace uso de la API RestCountries. Desde la API se recoge el código ISO de cada país y el nombre común traducido en castellano, aunque en caso de no existir tal traducción, cogerá el nombre original [25].

```php
class ImportarPaises extends Command
{
    /**
     * @var string
     */
    protected $signature = 'command:importarPaises';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Command description';

    /**
     * Execute the console command.
     *
     * @return int
     */
    public function handle()
    {
        $client = new Client();
        $response = $client->get('https://restcountries.com/v3.1/all');
        $paises = json_decode($response->getBody(), true);

        foreach ($paises as $pais) {
            $iso3 = $pais['cca3'] ?? null;
            $nombre = $pais['translations']['spa']['common'] ?? $pais['name']['common'] ?? null;

            if ($iso3 && $nombre) {
                Pais::updateOrCreate(
                    ['iso_3' => $iso3],
                    ['nombre' => $nombre]
                );
            }
        }

        $this->info('Países importados correctamente.');
        return Command::SUCCESS;
    }
}
```

Figura 31. Comando importar listado países.

En un modelo también es posible definir funciones personalizadas que encapsulan lógicas o tareas repetitivas con los datos del modelo y reutilizables en diferentes partes de la aplicación. En los modelos de factura, presupuesto y albaranes se gestiona, por ejemplo, la correcta asignación de numeración para mantener la correlatividad.

Utilizaremos el modelo 'Factura', uno de los más completos, a modo de ejemplo:

```
app > Models > Factura.php > Factura > comprobarNumeracion
You, 2 minutes ago | 1 author (You)
class Factura extends BaseModel
{
    protected $table = 'facturas';
    public $timestamps = true;

    public function Cliente()
    {
        return $this->belongsTo(Cliente::class, 'cliente_id');

    }

    public function Empresa()
    {
        return $this->belongsTo(Empresa::class, 'empresa_id');

    }

    public function Estado()
    {
        return $this->belongsTo(EstadoFactura::class, 'estado_id');
    }

    public function Serie(){
        return $this->belongsTo(SerieFactura::class, 'serie_id');
    }

    public function FormaPago(){
        return $this->belongsTo(FormaPagoEmpresa::class, 'forma_pago_id');
    }

    public function Documentos()
    {
        return $this->hasMany(DocumentoFactura::class, 'factura_id');
    }

    public function detallesFactura()
    {
        return $this->hasMany(DetalleFactura::class, 'factura_id');
    }

```

Figura 32. Primera captura modelo 'Factura'.

En la **Figura *32*** podemos observar las funciones que establecen las relaciones del modelo 'Factura' con los otros modelos. La función 'belongsTo()' indica una relación de uno a muchos (1:∞), donde este modelo pertenece a otro modelo, que puede tener varios registros asociados. Por otro lado, la función 'hasMany()' también indica una relación 1:∞, pero en este caso, es el modelo actual el que puede tener varios registros asociados.

```php
class Factura extends BaseModel
{
    //calculamos el total de una factura (iva aplicado)
    public function total()
    {
        $detalles_factura = $this->detallesFactura;
        $subtotal = 0;
        foreach ($detalles_factura as $detalle) {
            $subtotal += $detalle->Total();
        }

        if($this->recargo_equivalencia!=0){
            $recargo = $this->calcRecargo();
            $subtotal = $subtotal+$recargo;
        }
        return $subtotal;
    }

    public function totalTasas($iva){
        $detalles_factura = DetalleFactura::join('impuestos', 'impuestos.id', 'detalles_factura.impuesto')->where('detalles_factura.factura_id', $this->id)
                            ->where('impuestos.impuesto', $iva)->select('detalles_factura.*')->get();
        $total = 0;
        foreach ($detalles_factura as $detalle) {
            $total += $detalle->calcIVA();
        }
        return $total;
    }

    //calculamos la base de una factura (sin el iva aplicado)
    public function totalBase(){
        $detalles_factura = $this->detallesFactura;
        $total = 0;
        foreach ($detalles_factura as $detalle) {
            if($detalle->producto_id!=1){
                $total += $detalle->TotalSin();
            }
        }

        return $total;
    }
```

Figura 33. Segunda captura modelo 'Factura'.

En la **Figura *33***, se muestran las funciones creadas en este modelo que se reutilizarán a lo largo del proyecto. En estas funciones que se observan se calculan algunos de los totales de la factura y hacen uso de las relaciones definidas anteriormente para obtener los registros asociados a este modelo, además de utilizar las propias de esos registros. Para mostrar el funcionamiento, nos centraremos en la primera función.

```
$detalles_factura = $this->detallesFactura;
```

En esta línea se invoca la función 'detallesFactura', la cual devuelve los detalles asociados a esta factura actual. A continuación, en el bucle 'foreach', se recorre cada detalle:

```
foreach ($detalles_factura as $detalle) {$subtotal += $detalle->Total();}
```

En cada iteración, se llama a la función Total() de cada detalle, que calcula el total del mismo, y se va acumulando en el subtotal de la factura. La función en cuestión es la siguiente:

```php
app > Models > DetalleFactura.php > DetalleFactura > calcRecargo
 8   class DetalleFactura extends BaseModel
49
50      //Funcion que devuelve el total de un detalle con el IVA aplicado
51      public function Total(){
52          $base = $this->totalDescuentoFactura();
53          $tasas = $this->calcIVA();
54          $irpf = $this->calcRetencion();
55          return round($base+$tasas-$irpf,2);
56      }
57
```

Figura 34. Captura función Total() de DetalleFactura.

La función responsable de generar una factura electrónica se encuentra en el modelo 'Factura', y dado que la emisión de facturas electrónicas es uno de los objetivos principales de la aplicación, a continuación, se adjunta dicha función para su análisis.

```php
class Factura extends BaseModel

    public function generarFacturaElectronica(){
        $fac = new Facturae();
        $fac->setNumber($this->Serie->nombre, $this->n_factura);
        $fac->setIssueDate($this->fecha_factura);

        $fac->setSeller(new FacturaeParty([
        "taxNumber" => $this->Empresa->CIF,
        "name"      => $this->Empresa->nombre,
        "address"   => $this->Empresa->calle,
        "postCode"  => $this->Empresa->cp,
        "town"      => $this->Empresa->localidad,
        "province"  => $this->Empresa->provincia
        ]));
        if($this->Cliente->tipo_persona == 'J'){      You, 6 months ago • cambios
        $fac->setBuyer(new FacturaeParty([
            "isLegalEntity" => true, // Se asume true si se omite
            "taxNumber"     => $this->Cliente->CIF,
            "name"          => $this->Cliente->nombre,
            "address"       => $this->Cliente->calle,
            "postCode"      => $this->Cliente->cp,
            "town"          => $this->Cliente->localidad,
            "province"      => $this->Cliente->provincia,
            "countryCode"   => $this->Cliente->Pais->iso_3,  // Se asume España si se omite
            "email"   => $this->Cliente->email,
            "phone"   =>  $this->Cliente->telefono
        ]));
        }else{
        $fac->setBuyer(new FacturaeParty([
            "isLegalEntity" => false,
            "taxNumber"     => $this->Cliente->CIF,
            "name"          => $this->Cliente->nombre,
            "firstSurname"  => $this->Cliente->primer_apellido,
            "lastSurname"   =>  $this->Cliente->segundo_apellido,
            "address"       => $this->Cliente->calle,
            "postCode"      => $this->Cliente->cp,
            "town"          =>  $this->Cliente->localidad,
            "province"      =>  $this->Cliente->provincia,
            "countryCode"   => $this->Cliente->Pais->iso_3,
        ]));
        }
        foreach($this->detallesFactura as $linea){
            $item = [];
            if($this->recargo_equivalencia != 0){
                $taxes[Facturae::TAX_IVA] = ['rate'=> $linea->Iva->impuesto, "surcharge"=>$linea->Iva->recargo];
            }else{
                $taxes[Facturae::TAX_IVA] = $linea->Iva->impuesto;
```

app > Models > Factura.php > Factura > generarFacturaElectronica

```php
class Factura extends BaseModel
    public function generarFacturaElectronica(){
        foreach($this->detallesFactura as $linea){
            $item = [];
            if($this->recargo_equivalencia != 0){
                $taxes[Facturae::TAX_IVA] = ['rate'=> $linea->Iva->impuesto, "surcharge"=>$linea->Iva->recargo];
            }else{
                $taxes[Facturae::TAX_IVA] = $linea->Iva->impuesto;
            }
            if($linea->irpf!=0){
                $taxes[Facturae::TAX_IRPF] = $linea->irpf;
            }
            $item = [
                "name" => $linea->Productos->nombre,
                "description" => $linea->descripcion,
                "quantity" => $linea->cantidad,
                "unitPriceWithoutTax" => $linea->precio_ud,
                "unitOfMeasure" => Facturae::UNIT_DEFAULT,
                "taxes" => $taxes
            ];
            if($linea->descuento != 0){
                $item["discounts"] = [["reason" => "Descuento del ".$linea->descuento."%", "rate" => $linea->descuento]];
            }

            $fac->addItem(new FacturaeItem($item));
        }

        ($this->descuento_global != 0) ? $fac->addDiscount('Descuento del '.$this->descuento_global.'%', $this->descuento_global) : '';
        $payment["method"] = $this->FormaPago->FormaPago->nombre_facturae;
        ($this->fecha_vencimiento) ? $payment['dueDate'] = date('Y-m-d', strtotime($this->fecha_vencimiento)) : '';
        ($this->FormaPago->cuenta_bancaria) ? $payment["iban"] = $this->FormaPago->cuenta_bancaria : '';
        ($this->FormaPago->bic) ? $payment["bic"] = $this->FormaPago->bic : '';

        $fac->addPayment(new FacturaePayment($payment));

        if (!file_exists(public_path('/storage/'.$this->empresa_id.'/facturas/'.$this->Serie->nombre.$this->n_factura.'/'))) {
            mkdir(public_path('/storage/'.$this->empresa_id.'/facturas/'.$this->Serie->nombre.$this->n_factura.'/'),0777,true);
        }


        $fac->export(public_path('/storage/'.$this->empresa_id.'/facturas/'.$this->Serie->nombre.$this->n_factura.'/'.$this->Serie->nombre.$this->n_factura.".xml"));

        $this->factura_electronica = "/storage/".$this->empresa_id."/facturas/".$this->Serie->nombre.$this->n_factura."/".$this->Serie->nombre.$this->n_factura.".xml";

        $this->save();
    }
```

Figura 35. Capturas función encargada de la generación de una factura electrónica.

Podemos observar cómo se utiliza el paquete Facturae-PHP para crear la factura, estableciendo los datos tanto del vendedor como del comprador. Además, se genera un bloque 'item' para cada detalle de la factura, aplicando los descuentos e impuestos correspondientes, y se define el método de pago. Finalmente, la factura se guarda como un archivo, garantizando así su accesibilidad y almacenamiento adecuado.

Adicionalmente, se ha desarrollado una función encargada de firmar la factura electrónica con el certificado digital proporcionado por la empresa:

```php
public function firmarFacturaElectronica(){
    // Creación y configuración de la instancia
    $signer = new FacturaeSigner();
    $signer->loadPkcs12(storage_path('app/'.$this->Empresa->certificado), $this->Empresa->contrasenya_certificado);

    // Firma electrónica
    $xml = file_get_contents(public_path($this->factura_electronica));

    try {
        $signedXml = $signer->sign($xml);
    } catch (RuntimeException $e) {
        info('Error al firmar la factura electrónica: ' . $e->getMessage());
    }

    file_put_contents(public_path('/storage/'.$this->empresa_id.'/facturas/'.$this->Serie->nombre.$this->n_factura.'/'.$this->Serie->nombre.$this->n_factura.".xsig"), $signedXml);
    $this->factura_electronica_firmada = "/storage/".$this->empresa_id."/facturas/".$this->Serie->nombre.$this->n_factura."/".$this->Serie->nombre.$this->n_factura.".xsig";
    $this->save();
}
```

Figura 36. Captura de la función encargada de firmar una factura electrónica.

Es importante señalar que los certificados de las empresas se almacenan en una ubicación segura fuera de la ruta pública de la aplicación, para proteger su integridad y seguridad.

En el resto de los modelos se mantiene la misma estructura, pero con las funciones necesarias para cada uno.

- **CONTROLADORES**

A continuación, se desarrollan los controladores necesarios para gestionar las peticiones del usuario. Para esta aplicación se ha decidido que será suficiente con los siguientes para mantener un buen control sobre la aplicación:

1. **AlbaranController**: Permite la gestión de todas las acciones relacionadas con los albaranes, incluyendo todas las descritas en la **Figura *8***, la **Figura *9*** y la **Figura *10***.
    - **Listar albaranes:** Si la petición no incluye filtros, se obtiene un listado de los albaranes generados por la empresa. En caso contrario, se listan los albaranes según los filtros aplicados.
    - **Editar**: Edita la información del albarán con los datos introducidos por el usuario, incluyendo la creación, eliminación o modificación de sus detalles.
    - **Almacenar**: Crea un nuevo albarán con los datos introducidos, añadiendo los detalles correspondientes.
    - **Ver albarán**: Accede a la vista detallada del albarán.
    - **Eliminar**: Elimina el albarán solicitado.
    - **Previsualizar PDF**: Genera una vista previa del PDF del albarán.
    - **Descargar PDF**: Devuelve la descarga del PDF.
    - **Enviar por correo electrónico:** Envía por correo electrónico el PDF del albarán a las direcciones introducidas, con el asunto y el cuerpo introducidos por el usuario.
    - **Facturar:** Genera una factura a partir de los datos del albarán, replicando el cliente y los detalles.
2. **CategoriaProductoController**: Incluye las funciones disponibles para la gestión de las categorías de productos:
    - **Listar Categorías**: Si la petición no incluye filtros, se obtiene un listado de las categorías añadidas por la empresa. En caso contrario, se listan según los filtros aplicados.

- **Obtener modal creación/edición**: Devuelve codificada en JSON una vista con un modal en el que se introducirán los datos de la nueva categoría a crear o, en caso de haber seleccionado previamente una categoría, se mostrará en el modal la información de esta.
- **Almacenar/Editar**: Si se devuelve una categoría, modifica la información de esta con los datos introducidos por el usuario. En caso contrario, creará una nueva a partir de estos datos.
- **Eliminar**: Elimina la categoría de producto solicitada.

3. **ClienteController**: Incluye todas las funciones relacionadas con los clientes tal y como se ha definido en la **Figura *14*** y la **Figura *15***:
   - **Listar clientes**: Si la petición no incluye filtros, se obtiene un listado de los clientes añadidos por la empresa. En caso contrario, se listan según los filtros aplicados.
   - **Ver cliente**: devuelve la vista con la información del cliente a mostrar.
   - **Almacenar/Editar**: Si se devuelve un cliente, modifica la información de esta con los datos introducidos por el usuario. En caso contrario, creará uno nuevo a partir de estos datos.
   - **Eliminar**: Elimina el cliente en caso de no aparecer en ningún registro.
   - **Almacenar/Editar persona de contacto**: se crea o actualiza la información de una persona de contacto.
   - **Eliminar persona de contacto**: Elimina la persona de contacto.
4. **EmpresaController**: En este controlador se almacenan todas las funciones permitidas con relación a la propia empresa, que son las descritas en la **Figura *18*** y en la **Figura *19***:
   - **Ver empresa**: devuelve la vista con la información de la empresa, las carpetas a mostrar y los usuarios.
   - **Editar**: modifica la información de esta con los datos introducidos por el usuario.
   - **Descargar archivo**: devuelve la descarga del archivo seleccionado.
   - **Almacenar/Editar usuario nuevo**: se crea o actualiza la información de un usuario.
   - **Eliminar usuario**: Elimina el usuario.
   - **Almacenar certificado:** almacena de forma segura el certificado digital de la empresa.
   - **Almacenar/Editar serie de facturación:** se crea o actualiza la información de una serie de facturación.
   - **Eliminar serie de facturación**: elimina la serie si no aparece en ninguna factura.
   - **Almacenar/Editar forma de pago:** se crea o actualiza la información de una forma de pago.
   - **Eliminar forma de pago**: elimina la forma de pago si no aparece en ninguna factura.
5. **FacturaController**: Incluye las funciones que aparecen en la **Figura *11***, en la **Figura *12*** y en la **Figura *13***:
   - **Listar facturas:** Si la petición no incluye filtros, se obtiene un listado de las facturas generadas por la empresa. En caso contrario, se listan las facturas según los filtros aplicados.
   - **Editar**: Edita la información de la factura con los datos introducidos por el usuario, incluyendo la creación, eliminación o modificación de sus detalles. Además de verificar la existencia del cliente y producto, y crearlo en caso necesario.
   - **Almacenar**: Crea una nueva factura con los datos introducidos, añadiendo los detalles correspondientes. Además de verificar la existencia del cliente y producto, y crearlo en caso necesario.

- **Ver factura**: Accede a la vista detallada de la factura.
- **Eliminar**: Elimina la factura solicitada, en caso de no haber sido emitida.
- **Previsualizar PDF**: Genera una vista previa del PDF de la factura.
- **Descargar PDF**: Devuelve la descarga del PDF.
- **Enviar por correo electrónico:** Comprueba si se ha emitido la factura electrónica, en caso negativo la genera. Envía por correo electrónico el PDF de la factura con el QR que contiene la factura electrónica, y también la factura electrónica y los posibles archivos adjuntos a las direcciones introducidas, con el asunto y el cuerpo introducidos por el usuario.
- **Cambiar el estado:** actualiza el estado de la factura según se requiera. En caso de actualizar a 'Emitida', genera la factura electrónica.
- **Crear factura rectificativa:** crea una factura rectificativa clonando la información del cliente y los detalles.

6. **FacturaRectificadaController**: Se incluyen las mismas funciones que en el controlador anterior, excepto la de crear una factura rectificativa, ya que la aplicación no permite crear una factura rectificativa a partir de otra rectificativa.
7. **ProductoController**: Incluye las funciones relacionadas con los productos, que son las definidas en la **Figura *16*** y la **Figura *17***:
   - **Listar productos**: Si la petición no incluye filtros, se obtiene un listado de los productos añadidos por la empresa. En caso contrario, se listan según los filtros aplicados.
   - **Ver producto**: devuelve la vista con la información del producto a mostrar.
   - **Almacenar/Editar**: Si se devuelve un producto, modifica la información de este con los datos introducidos por el usuario. En caso contrario, creará uno nuevo a partir de estos datos.
   - **Eliminar**: Elimina el producto en caso de no aparecer en ningún otro registro.
   - **Activar/desactivar**: activa o desactiva el producto según su estado.

Para la gestión de login, logout y autenticación del usuario, utilizamos los controladores propios de Laravel que se encuentran en la carpeta 'App/Http/Controllers/Auth' y sobre los que no se ha efectuado ningún cambio.

- **RUTAS**

Por otro lado, tenemos el archivo en el que se almacenan todas las rutas de la aplicación. Este archivo se denomina 'web.php' y se encuentra en la carpeta 'routes'. En él se definen todas las rutas, incluyendo el tipo de petición (GET, POST, DELETE), la URL y el controlador con su correspondiente acción. También se le podría dar un nombre a esta ruta, que podría ser utilizada en las vistas en lugar de indicar la URL completa. En esta aplicación existen dos grupos de rutas, en el primer grupo se encuentran las rutas que no necesitan autenticación, como pueden el login, register o errores del tipo '404' o '500', mientras que el resto de las rutas deben pasar primero por el middleware 'auth' para verificar la autenticación del usuario.

Pongamos de ejemplo la siguiente ruta, que si deberá estar autenticado el usuario para que se realice la petición:

```
Route::get('/facturas/nueva_fact/{id}',[FacturaController::class,'nuevaFact']);
```

En este ejemplo. se utiliza una petición de tipo GET con URN '/facturas/nueva_fact/{id}', que tiene asociada la función 'nuevaFact' del controlador 'FacturaController'. El parámetro {id} de la ruta en este caso será el id de la empresa a la que pertenece usuario.

Con el siguiente comando podemos obtener la lista completa de las rutas definidas en la aplicación:

```
php82 artisan route:list
```

Lo ejecutamos y obtenemos un listado con las 167 rutas que contiene el proyecto:

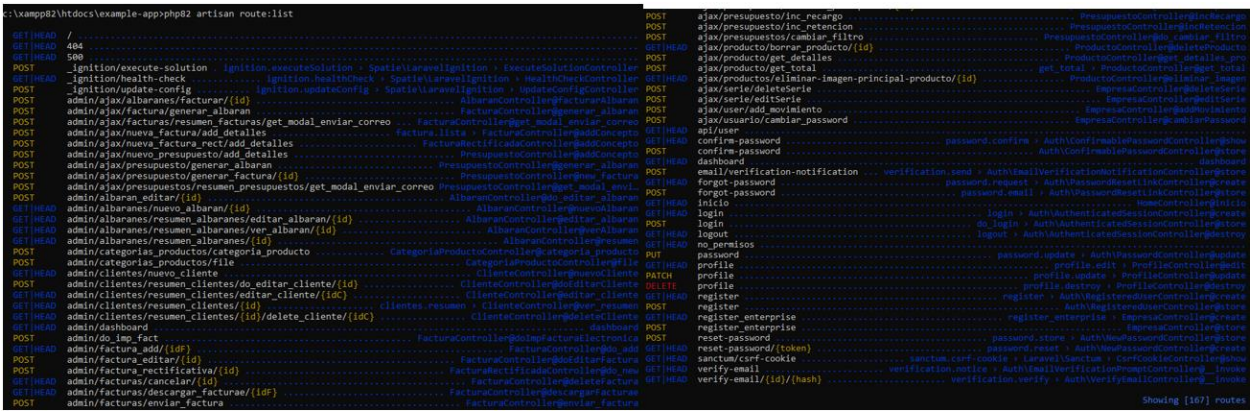
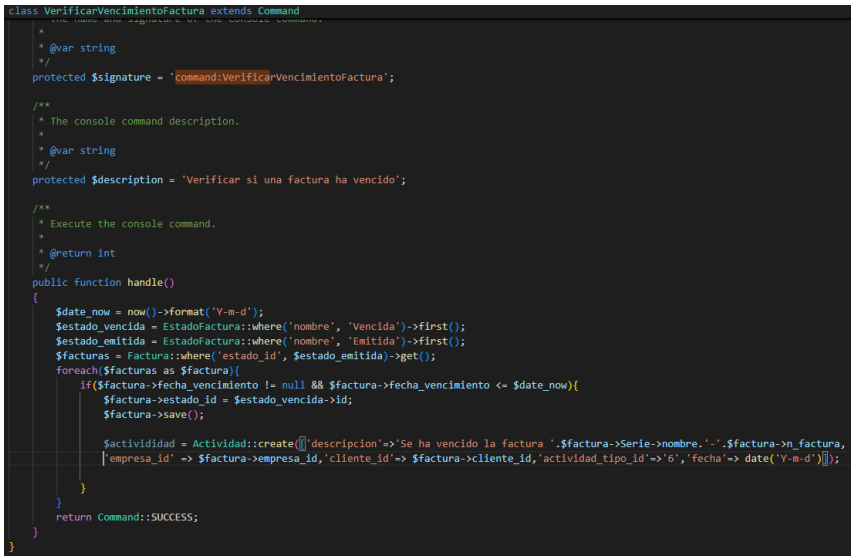


Figura 37. Listado de rutas

- TAREA PROGRAMADA

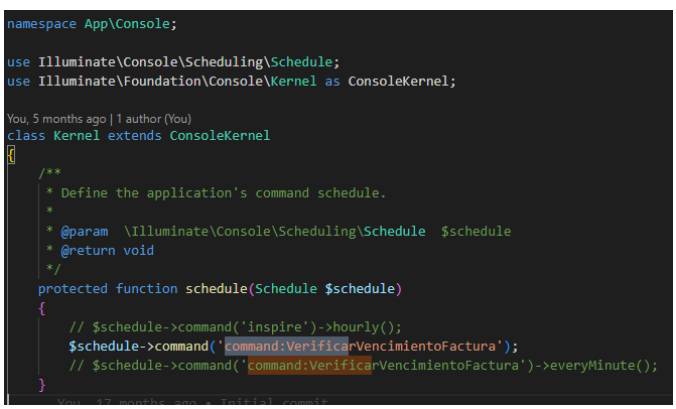
Para la configuración de la tarea programada definida en la **Figura *20***, primero es necesario crear el comando que ejecutará la función deseada:



```php
class VerificarVencimientoFactura extends Command
{
    /**
     * @var string
     */
    protected $signature = 'Command:VerificarVencimientoFactura';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Verificar si una factura ha vencido';

    /**
     * Execute the console command.
     *
     * @return int
     */
    public function handle()
    {
        $date_now = now()->format('Y-m-d');
        $estado_vencida = EstadoFactura::where('nombre', 'Vencida')->first();
        $estado_emitida = EstadoFactura::where('nombre', 'Emitida')->first();
        $facturas = Factura::where('estado_id', $estado_emitida)->get();
        foreach($facturas as $factura){
            if($factura->fecha_vencimiento != null && $factura->fecha_vencimiento <= $date_now){
                $factura->estado_id = $estado_vencida->id;
                $factura->save();

                $actividad = Actividad::create(['descripcion'=>'Se ha vencido la factura '.$factura->Serie->nombre.'-'.$factura->n_factura,
                'empresa_id' => $factura->empresa_id,'cliente_id'=> $factura->cliente_id,'actividad_tipo_id'=>'6','fecha'=> date('Y-m-d')]);

            }
        }
        return Command::SUCCESS;
    }
}
```

Figura 38. Comando VerificarVencimientoFactura.

Luego debe ser añadido al archivo ‘Kernel.php’ de Laravel, dentro de la función ‘schedule’:



```php
namespace App\Console;

use Illuminate\Console\Scheduling\Schedule;
use Illuminate\Foundation\Console\Kernel as ConsoleKernel;

You, 5 months ago | 1 author (You)
class Kernel extends ConsoleKernel
{
    /**
     * Define the application's command schedule.
     *
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        // $schedule->command('inspire')->hourly();
        $schedule->command('Command:VerificarVencimientoFactura');
        // $schedule->command('Command:VerificarVencimientoFactura')->everyMinute();
    }
```

Figura 39. Configuración Kernel.php

Finalmente, la tarea se programará para ejecutarse diariamente utilizando el programador de tareas del servidor, que en el entorno local se activará a través de un comando de Artisan:

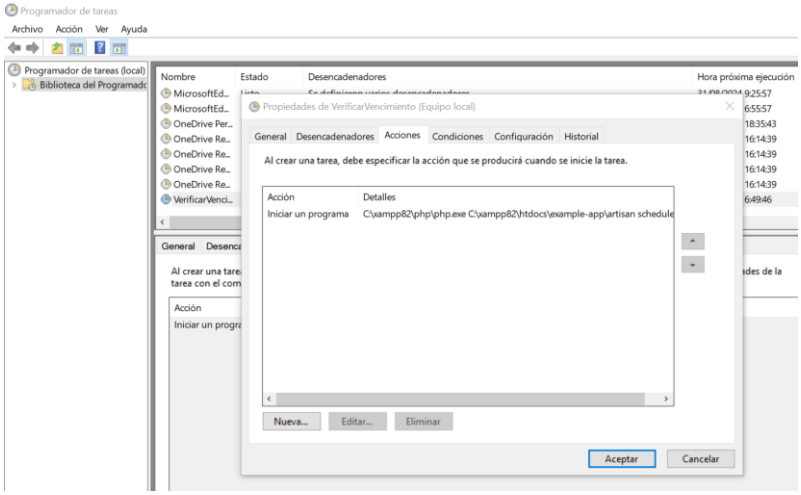


Figura 40. Configuración Programador de Tareas.

5.3 Desarrollo del Frontend

A continuación, abordamos el trabajo realizado en el frontend, es decir, la parte de la aplicación visible para el usuario. En esta fase del proyecto, se desarrollarán las vistas necesarias para implementar de manera visible todas las funciones que se han sido mencionadas a lo largo de la memoria. Para ello, utilizaremos las plantillas Blade que ofrece Laravel, tal como se explicó en el marco teórico. Cabe destacar que las vistas han sido implementadas de manera responsivas, es decir, diseñadas para adaptarse a cualquier resolución y tamaño de pantalla. Además, se ha utilizado Nifty [26], una plantilla de aplicaciones web de Bootstrap, que nos proporcionará un conjunto de estilos CSS ya definidos.

Dada la cantidad de vistas, se han organizado en carpetas para una mejor gestión. A continuación, se comentarán las vistas más relevantes:

2. **Vistas de albaranes**: dentro de esta carpeta tenemos las vistas encargadas de:
    - **Resumen y Lista**: Mostrar el listado de los albaranes junto con el formulario necesario para poder filtrarlos. En caso de filtrar, se recurrirá a AJAX y JQuery para obtener el listado de los albaranes filtrados sin necesidad de recargar la pantalla.
    - **Nuevo albarán y editar albarán**: estas vistas contienen los formularios con la información necesaria a rellenar para crear/editar/eliminar un albarán.
    - **Vista albarán**: en esta vista se puede visualizar toda la información del albarán que aparecerá en el PDF de este.
    - **Modal Correo**: esta vista contiene un modal con el formulario para enviar por correo el PDF del albarán y los posibles archivos adjuntos.
3. **Vistas de autenticación**: en esta carpeta están las vistas encargadas de la autenticación del usuario:
    - **Login**: contiene el formulario para hacer el login
    - **Register**: contiene el formulario para hacer el registro del nuevo usuario y la nueva empresa.
4. **Vistas de clientes**: en esta carpeta se encuentran las siguientes vistas:
    - **Resumen y Lista**: Mostrar el listado de los clientes junto con el formulario necesario para poder filtrarlos.

- **Ver cliente**: vista con el formulario para poder ver y editar la información del cliente, además de un listado de las actividades del mismo, sus facturas, presupuestos y albaranes.
- **Modal Actividad**: esta vista contiene un modal con en el que se muestra toda la información de una actividad.

5. **Vistas de errores**: en esta carpeta solo hay dos vistas:
   - **Error 404**: vista informando de que ha habido un error.
   - **Error 500**: vista informando de un error 500.
6. **Vistas de Facturas**:
   - **Resumen y Lista**: Mostrar el listado de las facturas junto con el formulario necesario para poder filtrarlas. Según el estado de la factura, permite realizar unas acciones u otras.
   - **Nueva factura y editar factura**: estas vistas contienen los formularios con la información necesaria a rellenar para crear/editar/eliminar una factura.
   - **Vista factura**: en esta vista se puede visualizar toda la información de la factura tal como aparecerá en el PDF.
   - **Modal Correo**: esta vista contiene un modal con el formulario para enviar por correo el PDF de la factura, la factura electrónica y los posibles archivos adjuntos.
7. **Vistas de Facturas Rectificativas**: contiene las mismas vistas que la carpeta anterior, pero adaptado a las facturas rectificativas.
8. **Layouts**: en esta carpeta se agrupan las vistas principales de la aplicación:
   - **Main**: esta vista contiene la estructura central de la página web. En ella se incluyen las carpetas con los estilos que deben aplicarse a las vistas, el footer y el menú de navegación. Además, con las etiquetas @yield se definen huecos que deberán rellenarse con las otras vistas.
   - **Navigation**: vista con el menú de navegación de la aplicación.
   - **Footer**: pie de página que aparecerá en todas las demás vistas.
9. **Vistas de mantenimiento**:
   - **Ver empresa**: vista con el formulario para poder ver y editar la información de la propia empresa, además de un listado con lo archivos que se han ido almacenando de esta por carpetas y el listado de usuarios que perteneces a esta.
   - **Nuevo/Editar cliente**: estas vistas contienen los formularios con la información necesaria a rellenar para crear/editar/eliminar un usuario.
10. **Vistas de PDF**:
    - **Factura y su cabecera**: estas vistas están diseñadas para generar en PDF la información de la factura.
    - **Factura Rectificativa y su cabecera**: igual que las anteriores pero adaptadas a las facturas rectificativas.
    - **Presupuesto y su cabecera**: igual que las anteriores, pero para los presupuestos.
    - **Albarán y su cabecera**: igual que las anteriores, pero para los albaranes.
11. **Vistas de Presupuestos**:
    - **Resumen y Lista**: Mostrar el listado de los presupuestos junto con el formulario necesario para poder filtrarlos. Según el estado del presupuesto, permite realizar unas acciones u otras.
    - **Nuevo presupuesto y editar presupuesto**: estas vistas contienen los formularios con la información necesaria a rellenar para crear/editar/eliminar un presupuesto.
    - **Vista presupuesto**: en esta vista se puede visualizar toda la información del presupuesto tal como aparecerá en el PDF.
    - **Modal Correo**: esta vista contiene un modal con el formulario para enviar por correo el PDF del presupuesto y los posibles archivos adjuntos.

12. **Vistas de Productos**:
    - **Resumen y Lista**: Mostrar el listado de los productos junto con el formulario necesario para poder filtrarlos.
    - **Nuevo producto y editar producto**: estas vistas contienen los formularios con la información necesaria a rellenar para crear/editar/eliminar un producto.
    - **Resumen y Lista de Categorías de Productos**: Mostrar el listado de las categorías de productos junto con el formulario necesario para poder filtrarlas.
    - **Modal nueva/editar categoría de producto**: esta vista contiene un modal con el formulario para editar o crear una categoría de producto.

Capítulo 6. Pruebas y Validación

A lo largo del proceso de implementación, se ha llevado a cabo un seguimiento para asegurar que la aplicación cumpliese con los requisitos que se habían planteado. Finalmente, obtenemos una aplicación funcional que cumple con lo requerido y como muestra de resultados vamos a efectuar una clase de pruebas manuales.

Además, para la validación de las facturas electrónicas creadas por el sistema, utilizaremos dos ejemplos oficiales del gobierno. [27]

Una vez accedemos a la web, el primer paso es la autenticación del usuario:

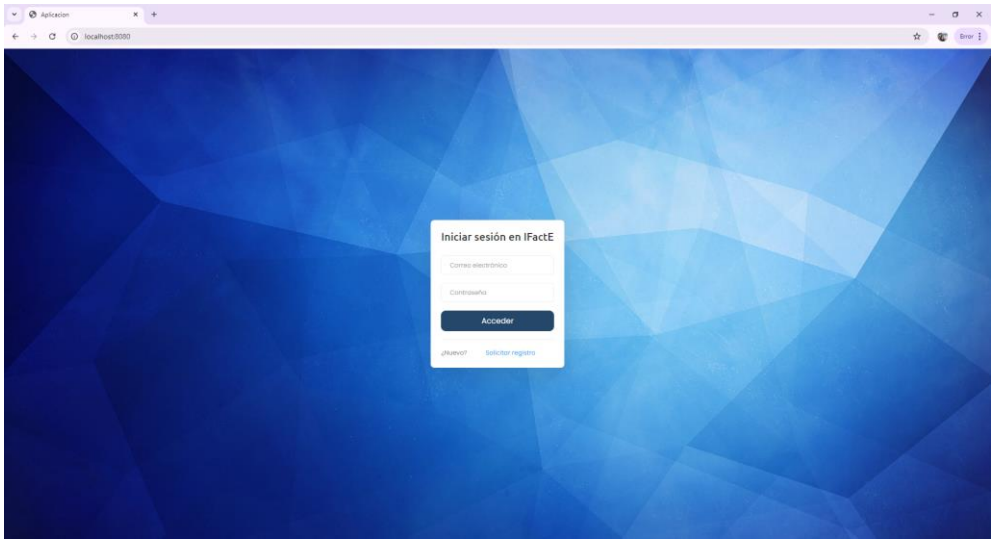


Figura 41. Vista 'Login'.

En este caso, empezaremos creando un usuario y una empresa nueva.

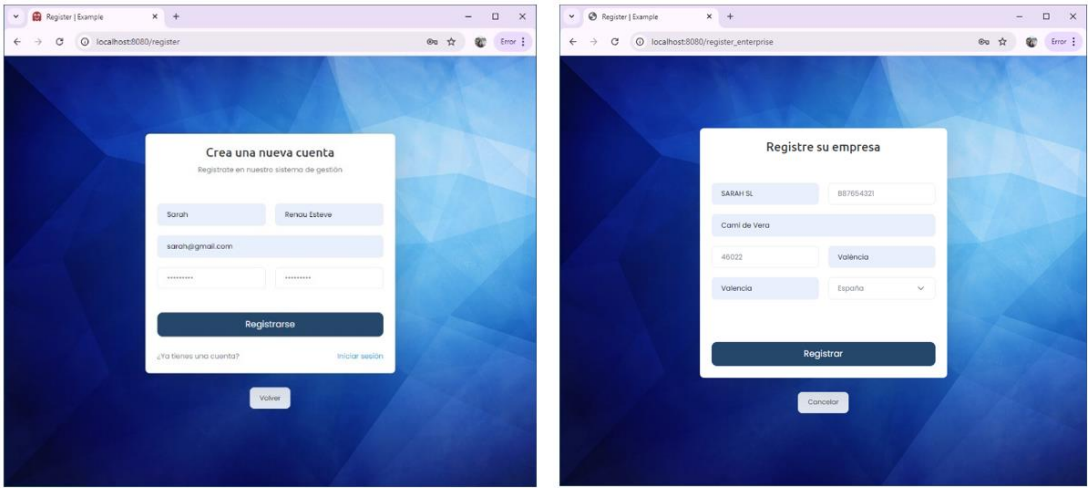


Figura 42. Vistas registro nuevo usuario y nueva empresa.

Una vez registrados, accedemos a la página principal de la aplicación, 'Dashboard':

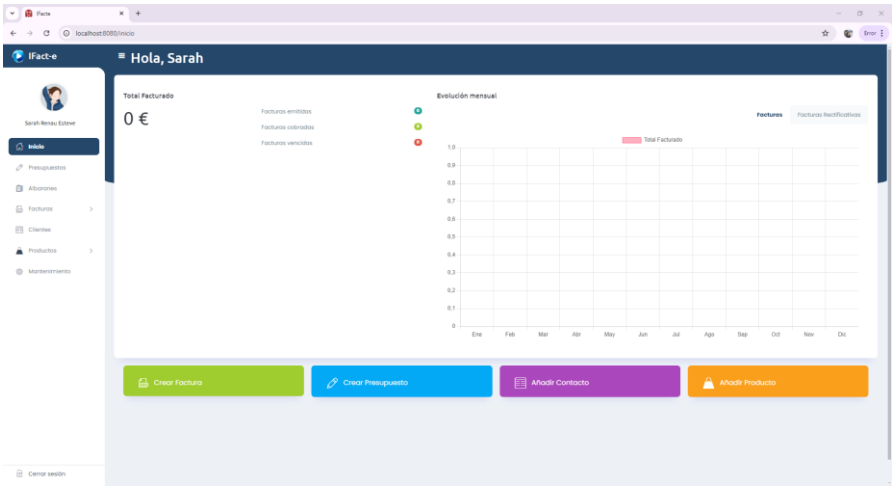


Figura 43. Vista 'Dashboard'.

Ahora vamos a acceder a la pestaña 'Mantenimiento'. Desde esta vista podemos ver y editar los datos de la propia empresa, las carpetas que crea el sistema automáticamente en las que se almacenarán todos los documentos de esta, y también los usuarios relacionados. El propio sistema también crea dos series de facturación por defecto, aunque siempre pueden editarse o eliminarse. Desde esta pestaña puede subirse el certificado digital que se utilizará para firmar las facturas y el logo de la empresa que aparecerá en los PDFs de los distintos documentos mercantiles. Para este ejemplo, se ha añadido el logo de Ibermedia.

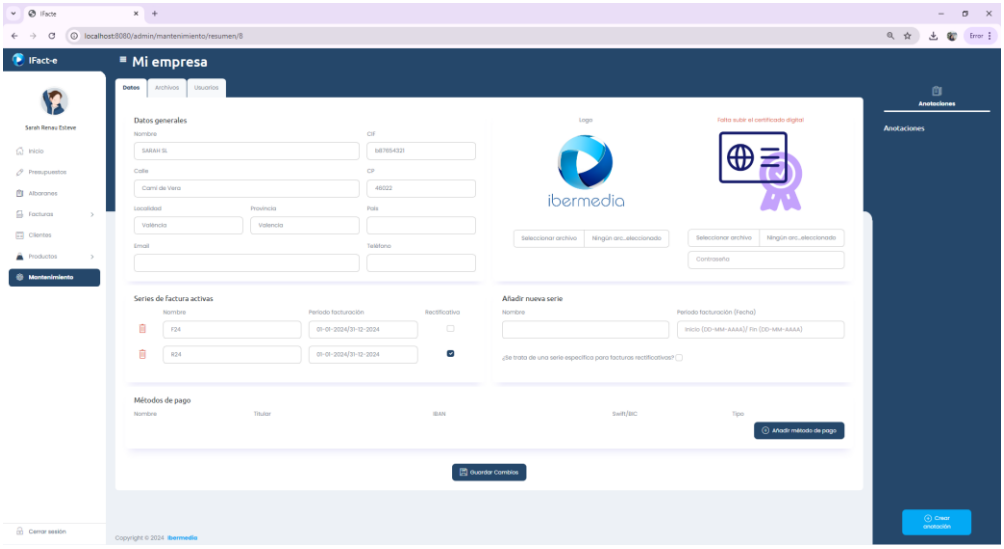


Figura 44. Vista edición datos empresa.

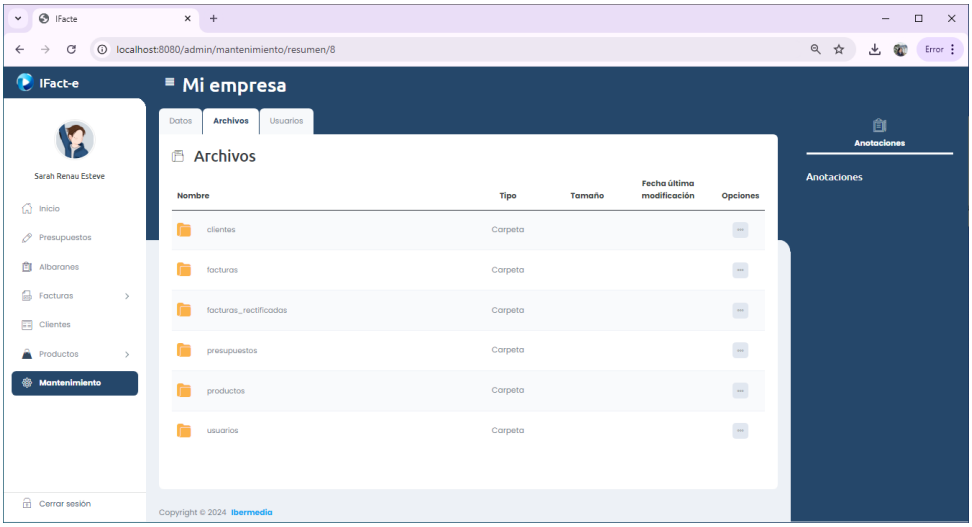


Figura 45. Pestaña carpetas creadas por el sistema.

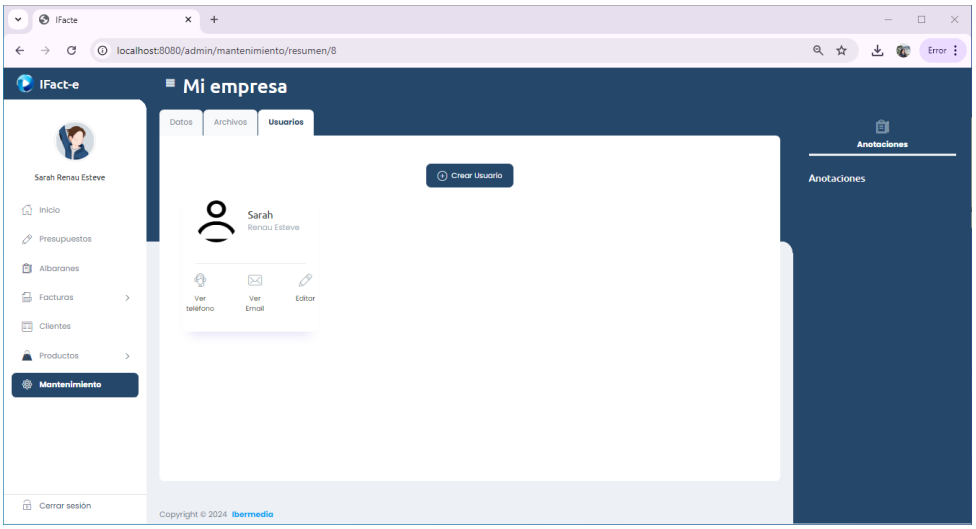


Figura 46. Pestaña usuarios relacionados con la empresa.

Si clicamos ahora sobre el botón ‘Clientes’ o ‘Productos’, observamos que no hay ningún registro.

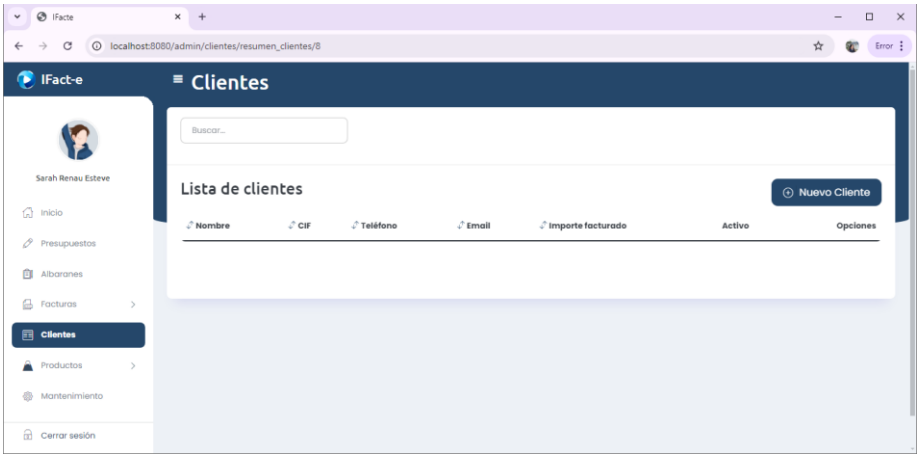


Figura 47. Listado inicial clientes.

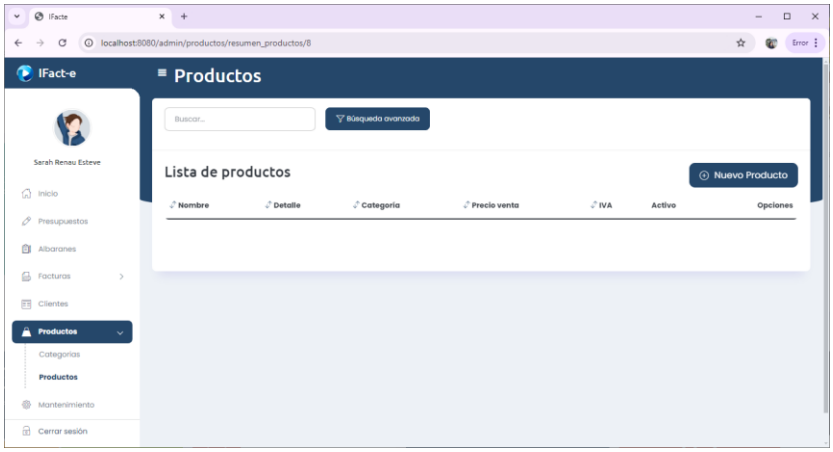


Figura 48. Listado inicial productos

Por tanto, crearemos el cliente y los productos directamente desde nuestro primer presupuesto. Utilizaremos el primer ejemplo de factura electrónica [27] para rellenar todos los datos.

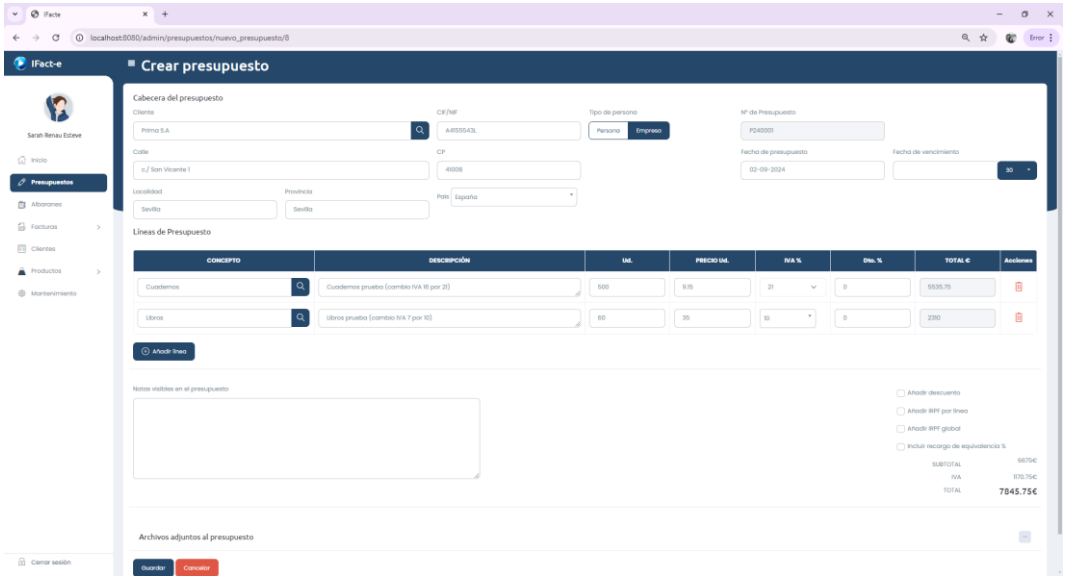


Figura 49. Vista creación presupuesto.

Una vez le damos a guardar, podemos ver en el listado nuestro presupuesto ya creado, y como todavía no ha sido enviado al cliente final, puede seguir editándose.

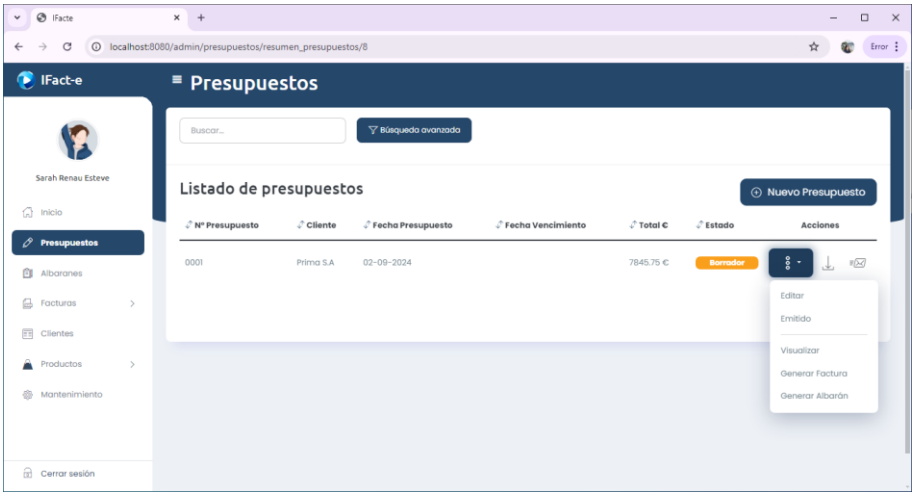


Figura 50. Listado presupuestos.

Vamos a ver ahora, como tanto el cliente como los productos han sido registrados correctamente y pueden observarse en sus correspondientes listados:

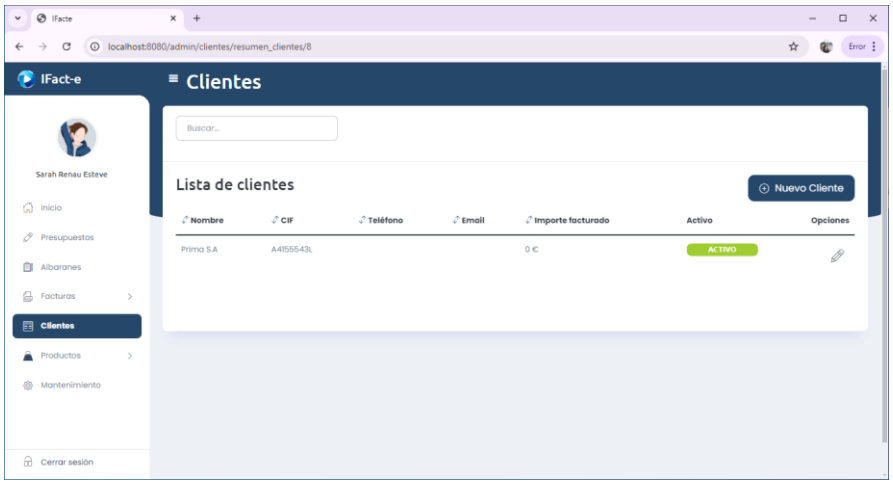


Figura 51. Listado clientes tras añadir presupuesto.

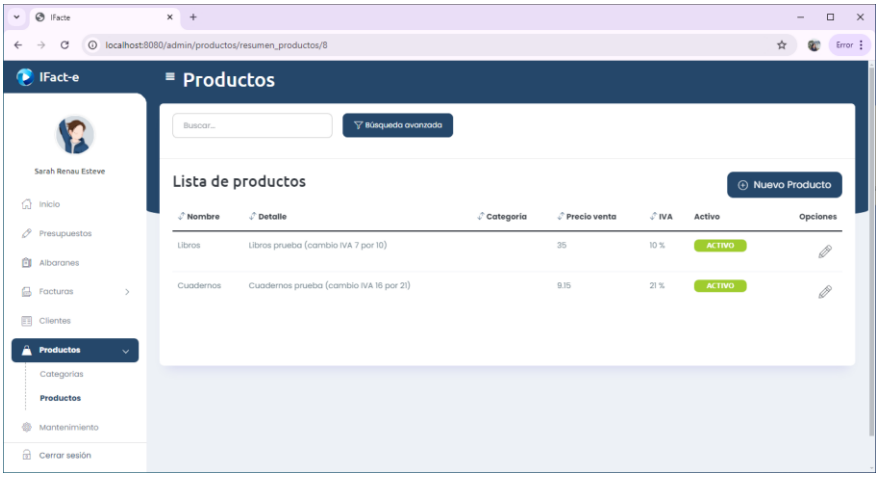


Figura 52. Listado productos tras añadir presupuesto.

Vamos a generar ahora un nuevo albarán directamente desde el presupuesto que acabamos de crear. Para ello, en el listado de opciones que hemos observado antes en la ***Figura 50***, elegimos la opción 'Generar Albarán'. Podemos ver como crea el albarán con los mismos datos del cliente, y los detalles del presupuesto, aunque pueden editarse o eliminarse. Además, el sistema añade una dirección de entrega por defecto, que coincide con la dirección del cliente, y añade por también como fecha de entrega la fecha de 'hoy'.

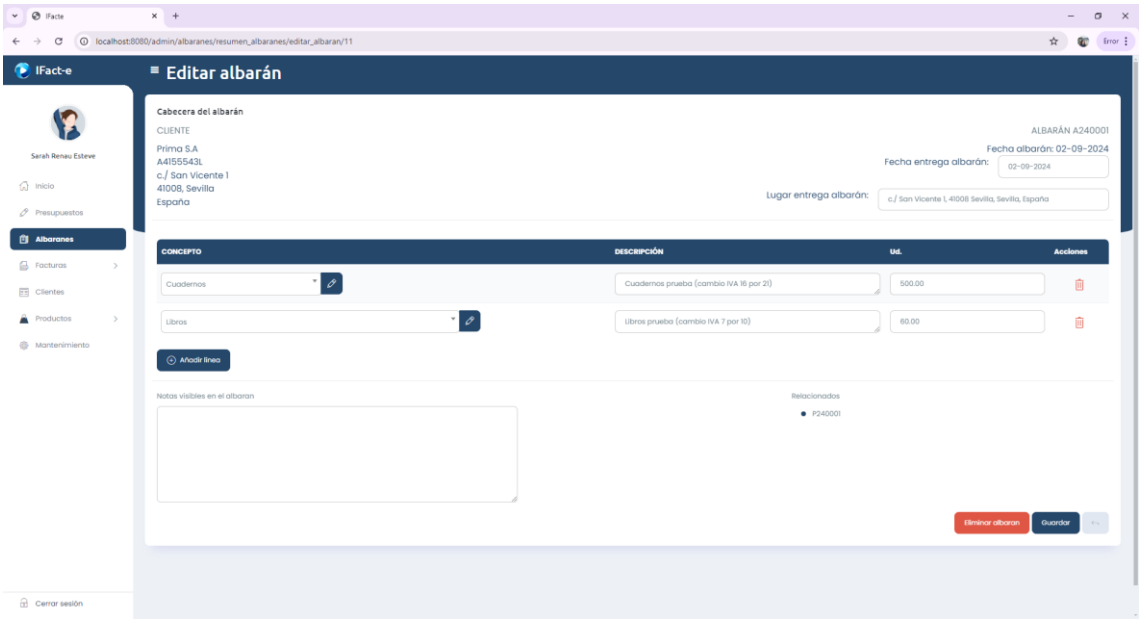


Figura 53. Vista edición albarán tras crearlo desde un presupuesto.

Comprobamos los datos y guardamos, de esta forma ya aparece en el listado de albaranes:

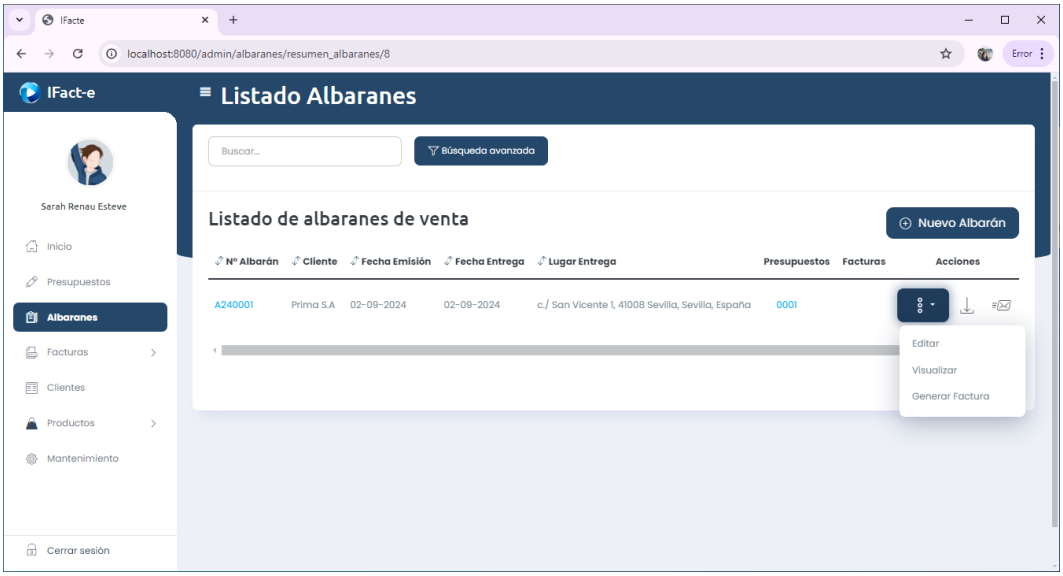


Figura 54. Listado de albaranes.

Para continuar probando la funcionalidad de la aplicación, vamos a generar directamente una factura desde el propio albarán. Dado que pueden existir distintas series de facturación,

debemos elegir en cuál queremos añadir la factura. En este ejemplo, escogemos la primera generada por el propio sistema.

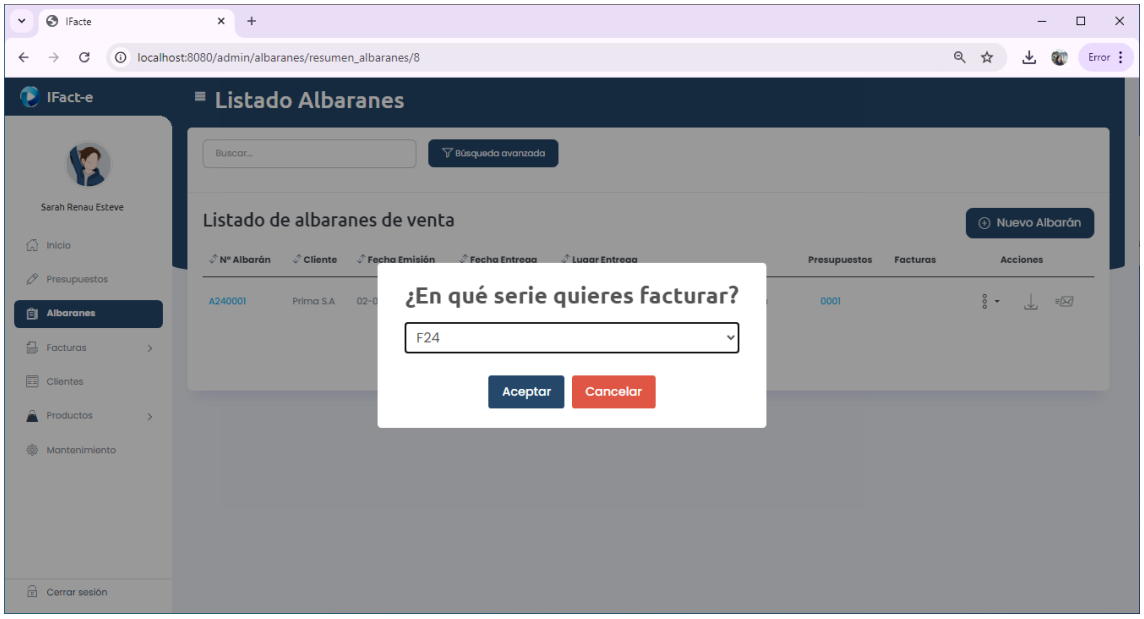


Figura 55. Elección serie de facturación.

Le damos a aceptar y accedemos directamente a la edición de la factura que se ha creado a partir de este albarán. Por defecto, ha añadido los precios e impuesto del propio producto.

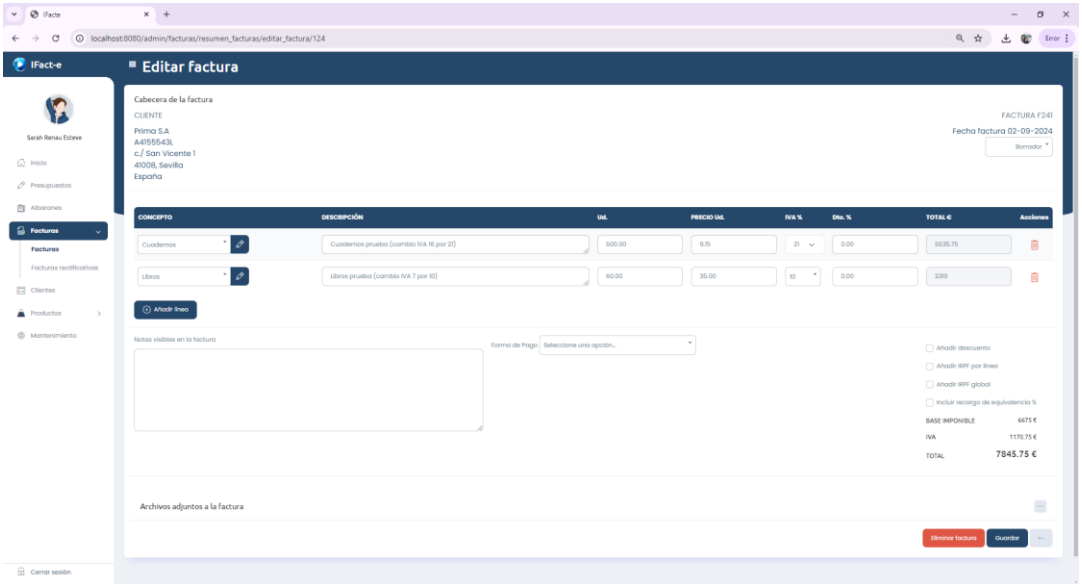


Figura 56. Vista edición factura generada a partir de un albarán.

Si está todo correcto, podemos dar al botón de guardar y accedemos a la visión de la factura.

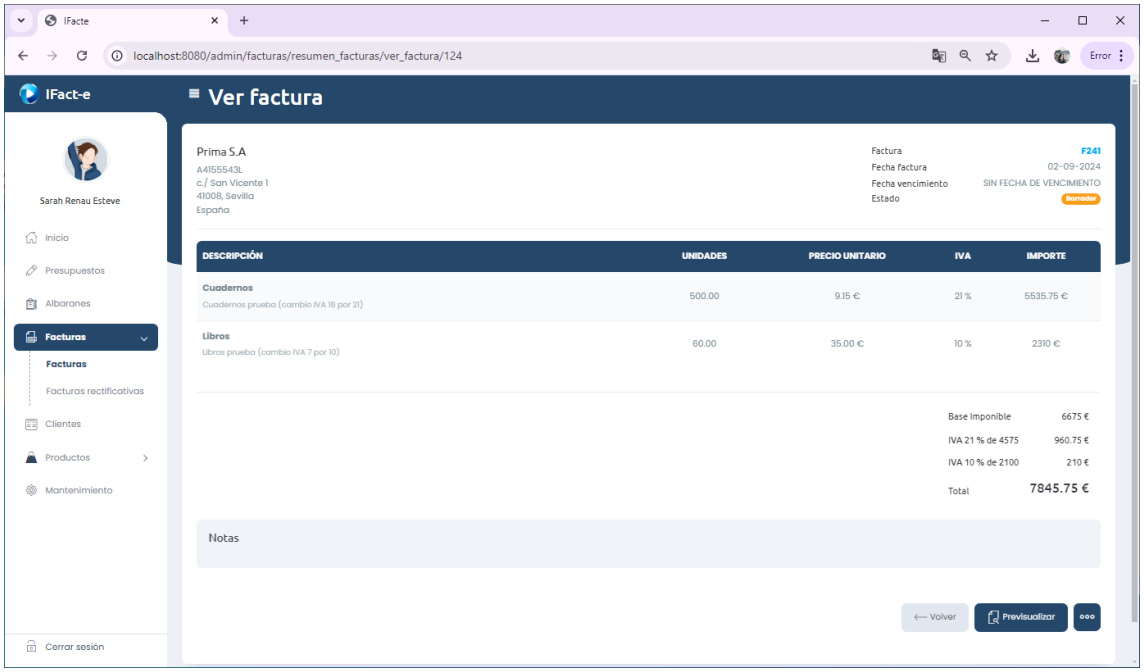


Figura 57. Vista visualización factura en estado 'Borrador'.

Desde esta ventana, podemos realizar una serie de acciones como previsualizar el PDF de la factura. Dado que la factura está como 'Borrador', el PDF no contiene el QR con toda la información de la factura electrónica, y además incluye una marca de agua que indica este estado.

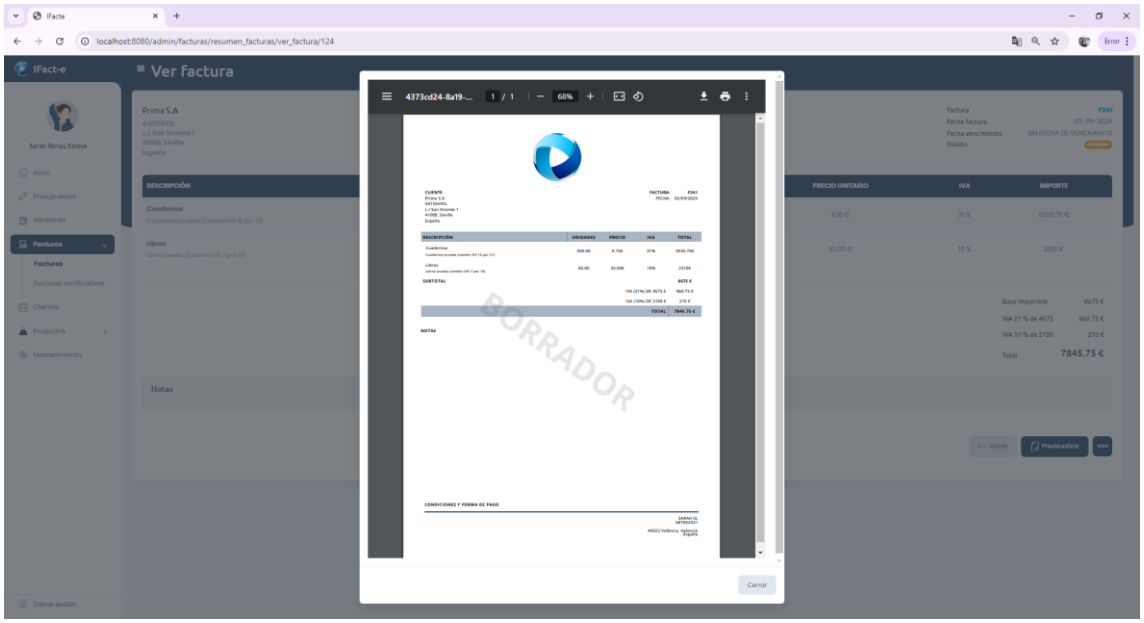


Figura 58. Previsualización factura en PDF estado 'Borrador'.

Además de previsualizar el PDF, podemos realizar otras opciones:

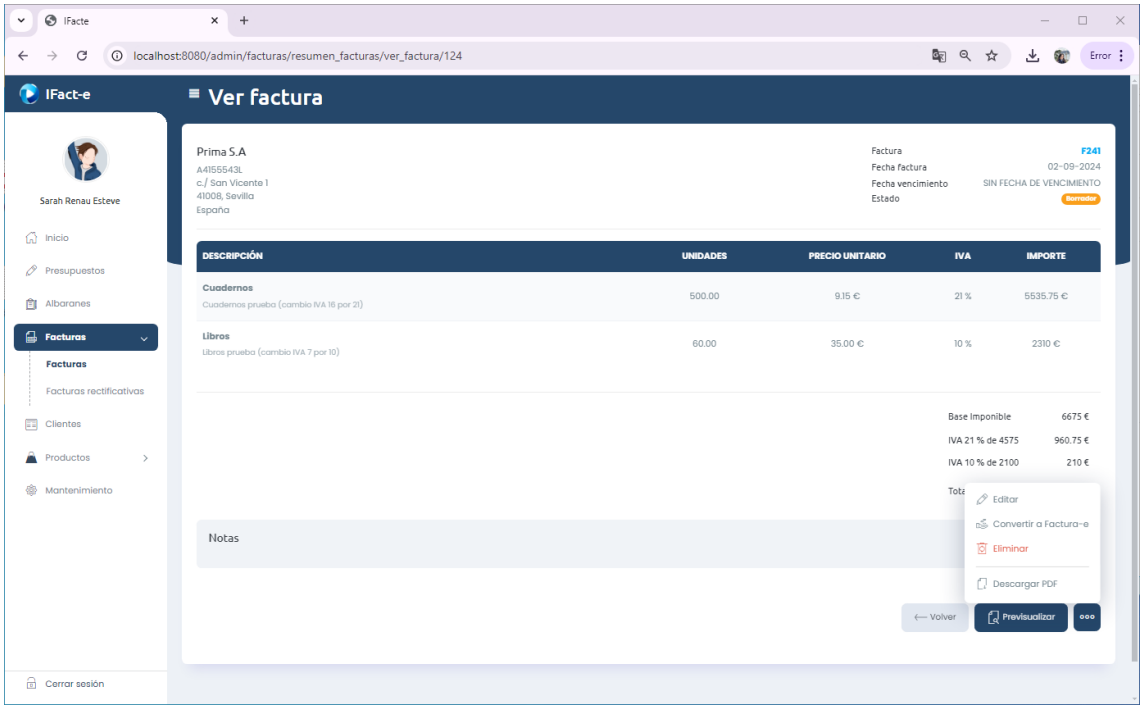


Figura 59. Opciones permitidas para una factura en estado ‘Borrador’.

Nuestro siguiente paso es generar la factura electrónica, aunque como no se ha subido aún el certificado digital, esta no estará firmada. Elegimos la opción ‘Convertir a Factura-e’ y seguidamente podemos ver la factura en formato XML:

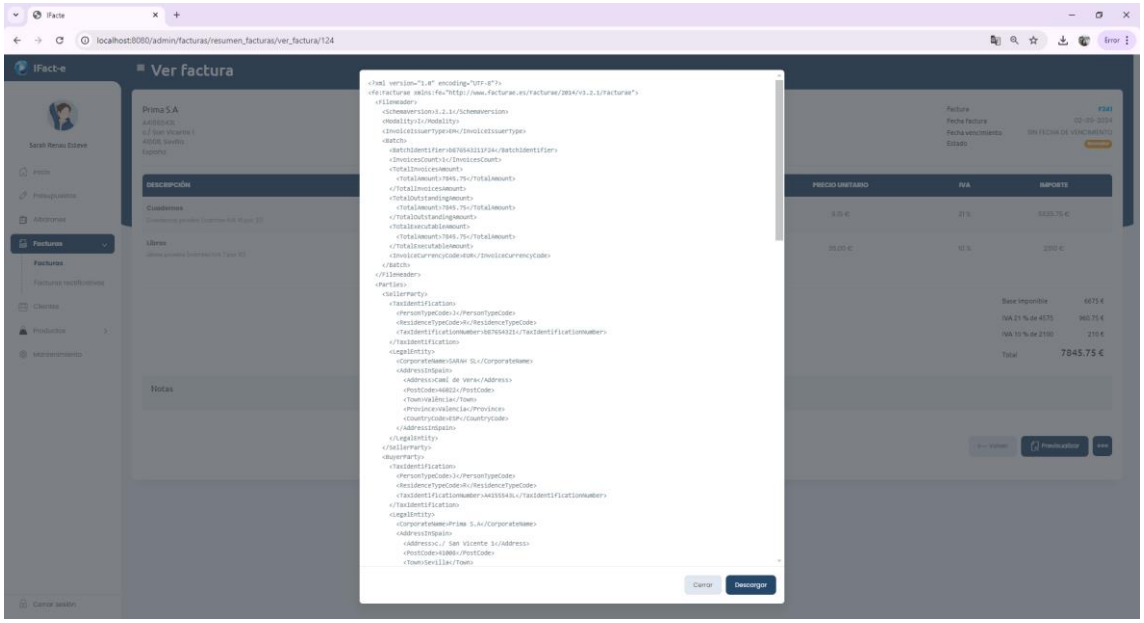


Figura 60. Ejemplo de la primera factura electrónica.

Una vez generada la factura electrónica, el sistema actualizará su estado a ‘Emitida’ y se podrá realizar otras acciones sobre esta como indicarla como ‘Cobrada’ o descargar el PDF final:

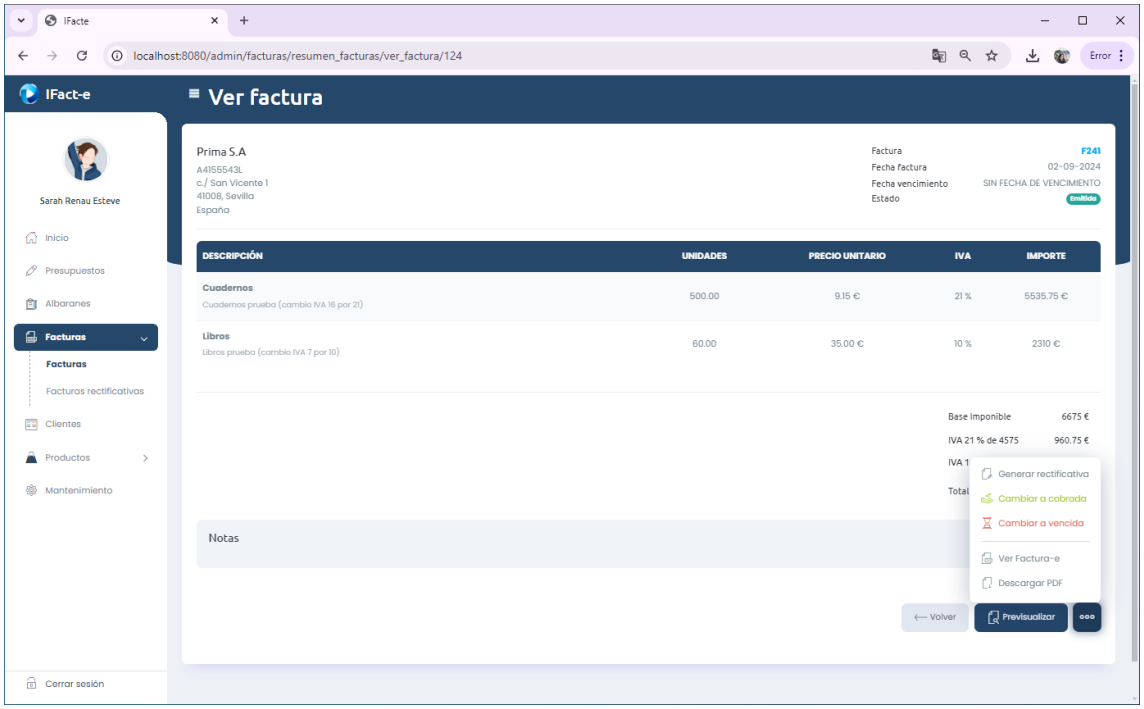


Figura 61. Opciones permitidas para una factura en estado 'Emitida'.

Una vez emitida, vamos a ver cómo sería ahora el PDF:

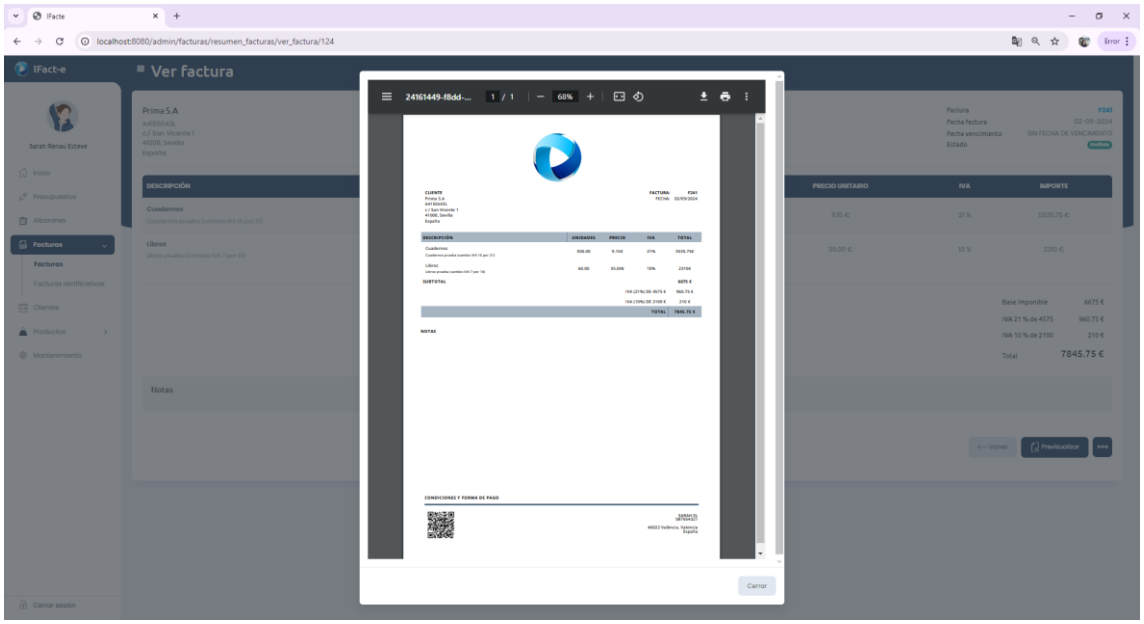


Figura 62. Previsualización factura en PDF estado 'Emitida'.

Una vez se han realizado las pruebas con respecto a la primera factura, vamos a crear y editar una categoría de producto.

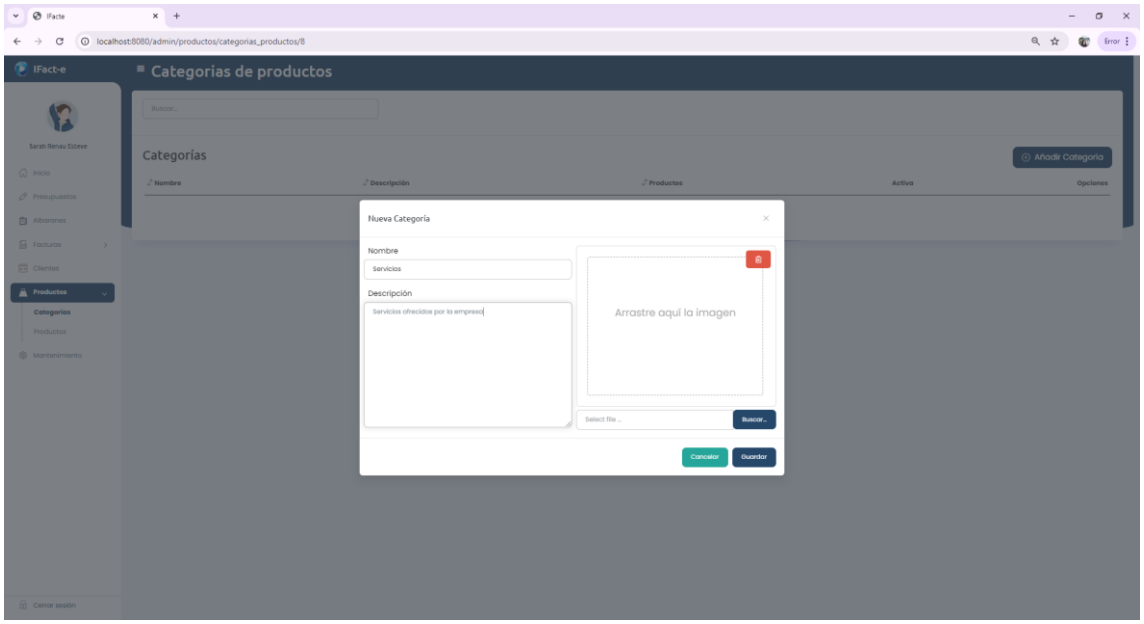


Figura 63. Modal crear nueva categoría de producto.

Una vez creada una categoría, se puede activar o desactivar en caso de no querer eliminarla:

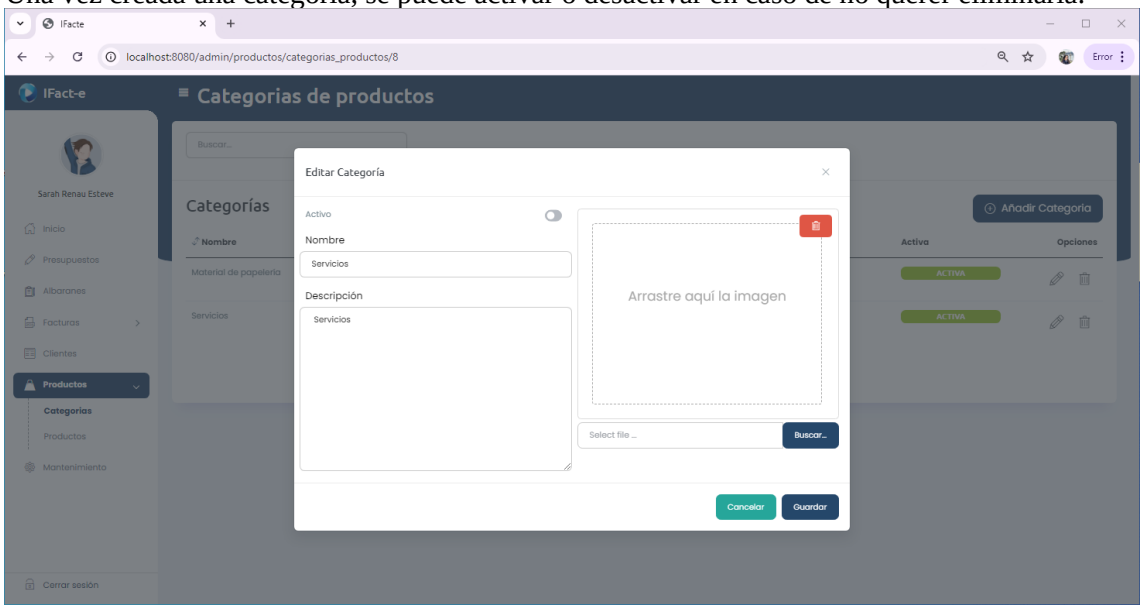


Figura 64. Modal editar categoría de producto.

Ahora que ya existen categorías de productos, vamos a asignarles a los productos la categoría a la que pertenecen. En este caso hemos desactivado la categoría ‘Servicios’ por tanto, está opción se muestra pero no se permite seleccionarla.

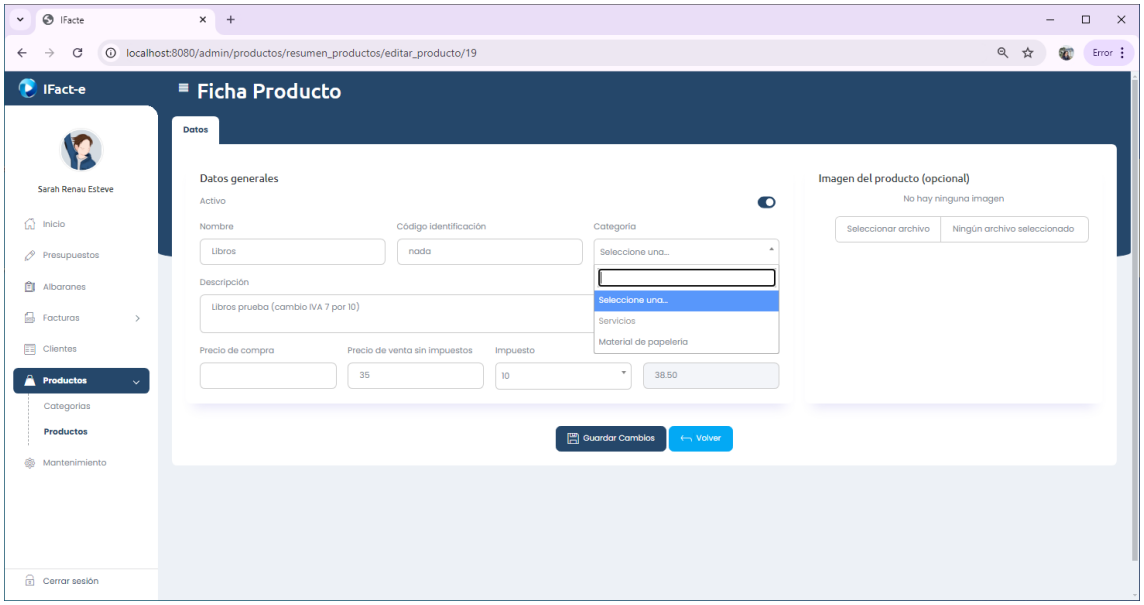


Figura 65. Vista editar producto.

Hemos visto cómo pueden creares presupuestos, albaranes y facturas relacionados entre sí, y además como no es necesario tener registros de clientes ni productos para poder crear estos documentos. Ahora vamos a realizar las pruebas en sentido inverso, es decir, crearemos primero el cliente y un producto, y a partir de estos, recrearemos la segunda factura de ejemplo de [27].

Por tanto, creamos el cliente y rellenamos los datos que tenemos. Para el producto hemos seguido los mismos pasos y además lo hemos añadido a la categoría ‘Servicios’ que ya hemos vuelto a activar.

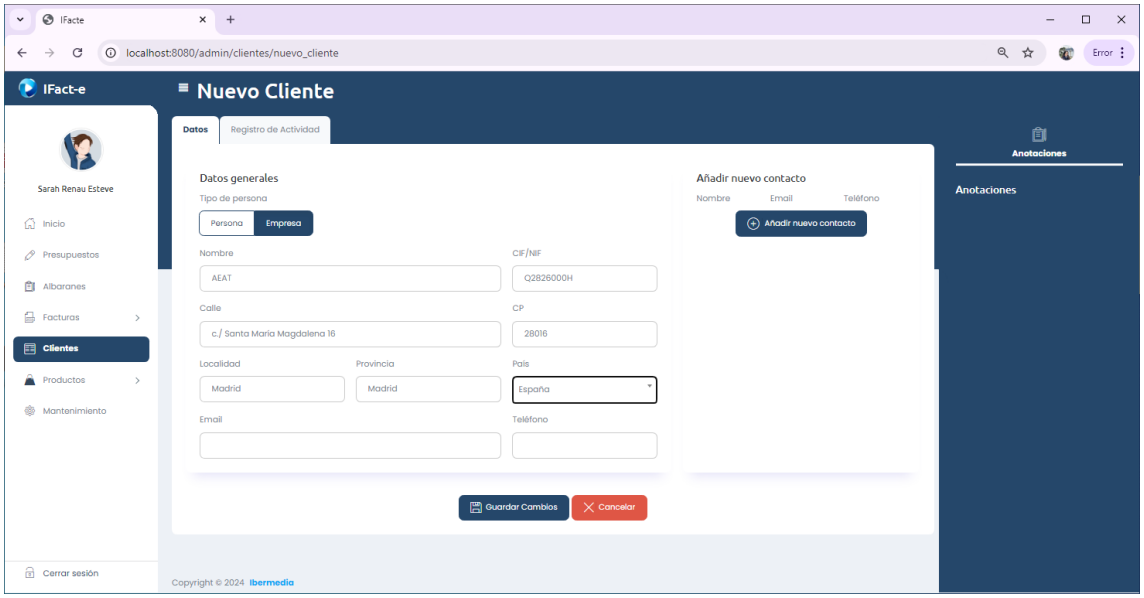


Figura 66. Vista crear nuevo cliente.

Para la siguiente factura vamos a probar a firmarla con un certificado, por tanto, subimos al servidor el certificado, añadimos la contraseña necesaria para acceder a él y también creamos un

nuevo método de pago de la empresa para asignárselo a la factura. En este caso se utiliza para las pruebas el certificado 'facturae.p12' proporcionado en [28] y su correspondiente contraseña.

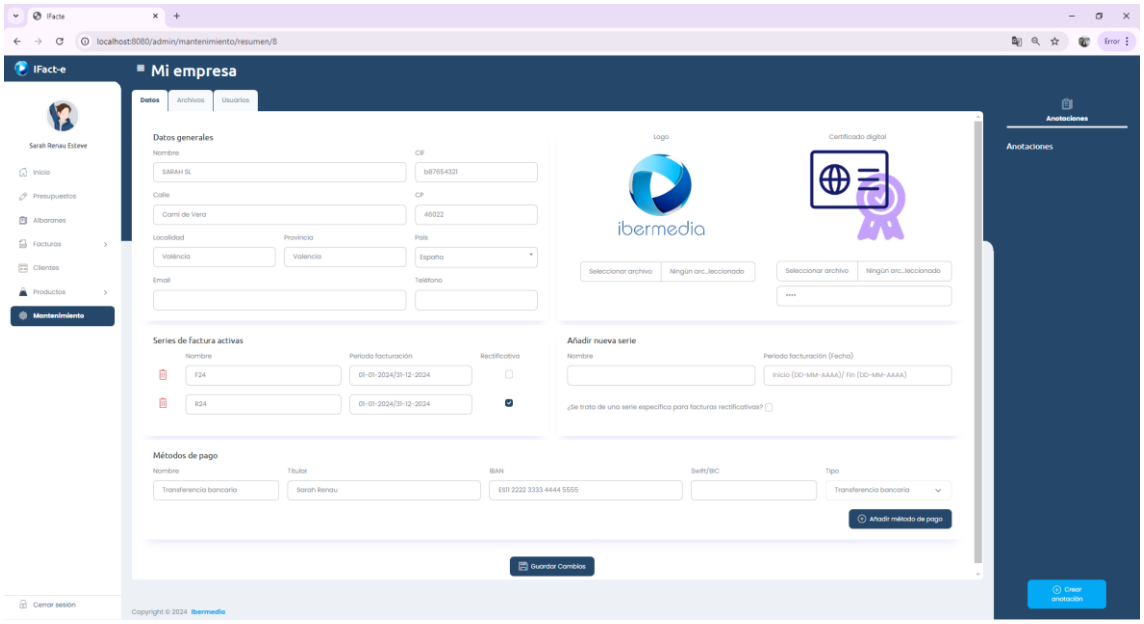


Figura 67. Vista mantenimiento tras añadir certificado y método de pago.

Accedemos ahora a 'Añadir Factura' desde el listado de facturas y seleccionamos el cliente que acabamos de crear para que se autorrellenen los campos de la factura correspondientes al cliente, e igualmente con el producto 'Análisis Orgánico'.

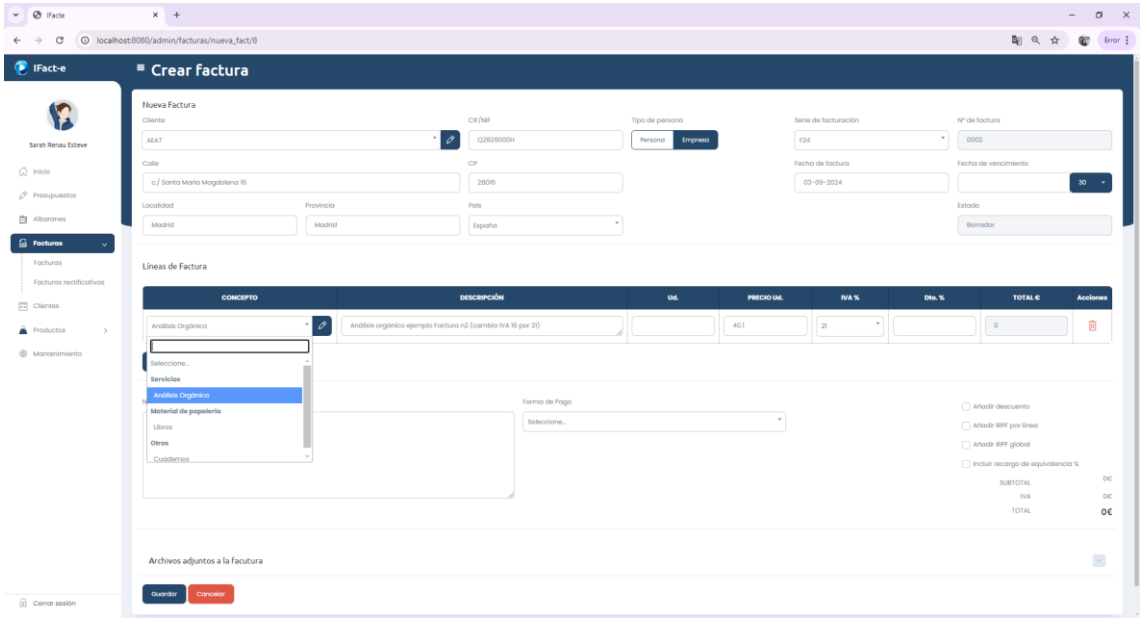


Figura 68. Vista crear nueva factura.

Rellenamos todos los datos, añadimos también un descuento del 10% sobre el total de la factura y accedemos a la vista de la factura ya completa. Una vez estamos en esta vista, repetimos el proceso de antes y la convertimos a Factura-e. Como esta vez ya hemos subido el certificado, vemos como ya aparece la opción de firmarla.

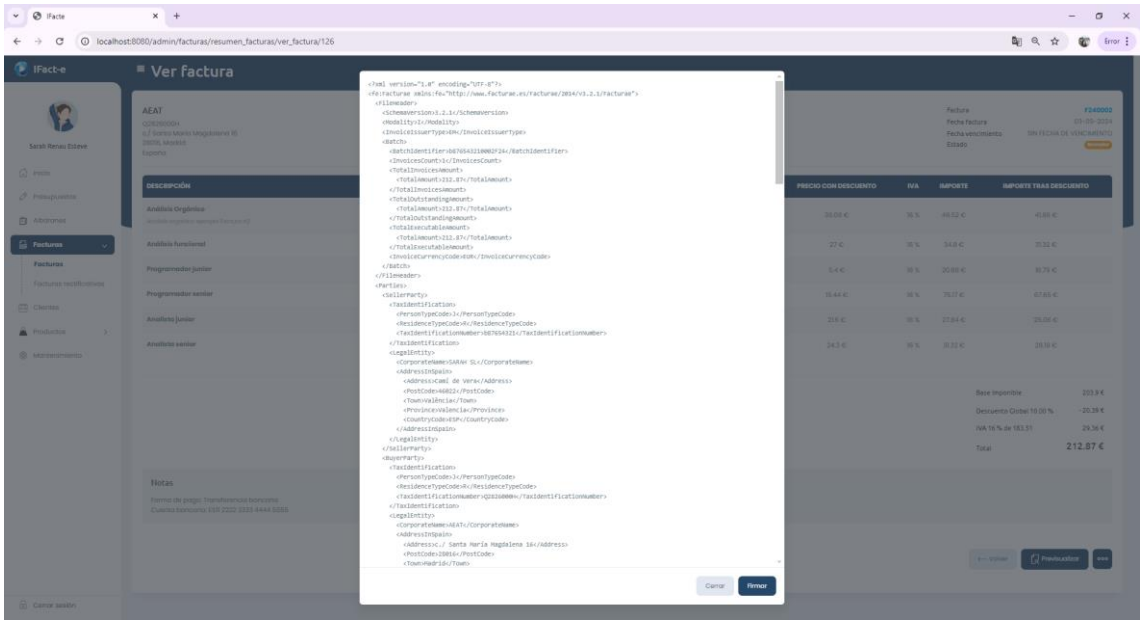


Figura 69. Segunda factura en formato Facturae.

Le damos a firmar y vemos como ya aparece la factura electrónica con toda la información relativa al certificado y el sello de tiempo de la firma.

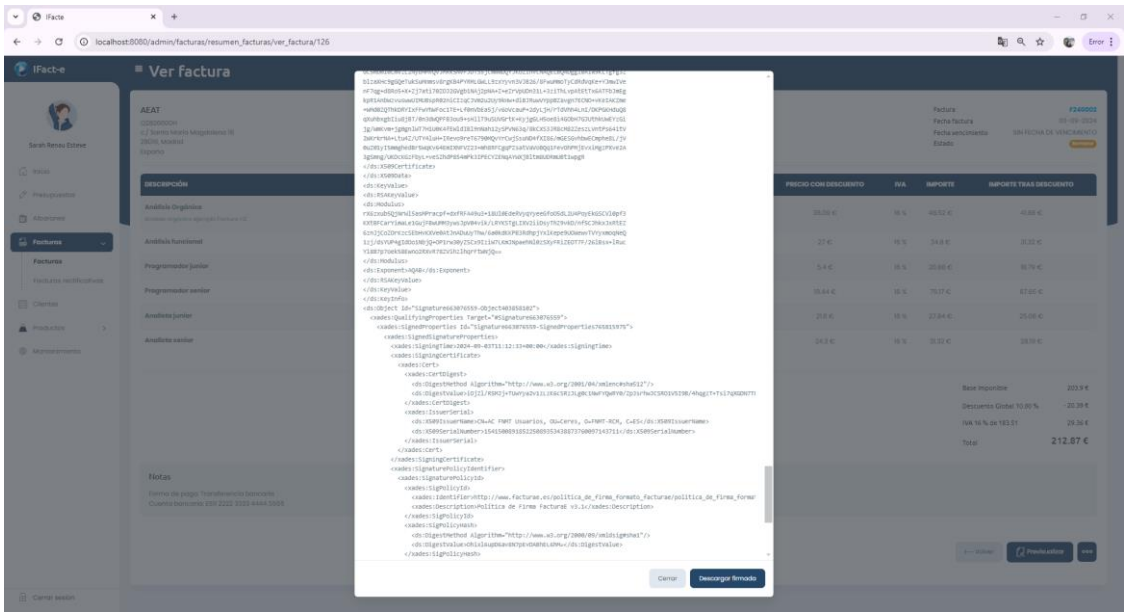


Figura 70. Segunda factura en formato Facturae firmada.

Para verificar que todo esté correcto, utilizamos varios validadores de facturas electrónicas [29] [30] y obtenemos un resultado válido en ambos, por lo que la factura sería válida y todos los datos coinciden.

Figura 71. Validación factura facturaedirecta.

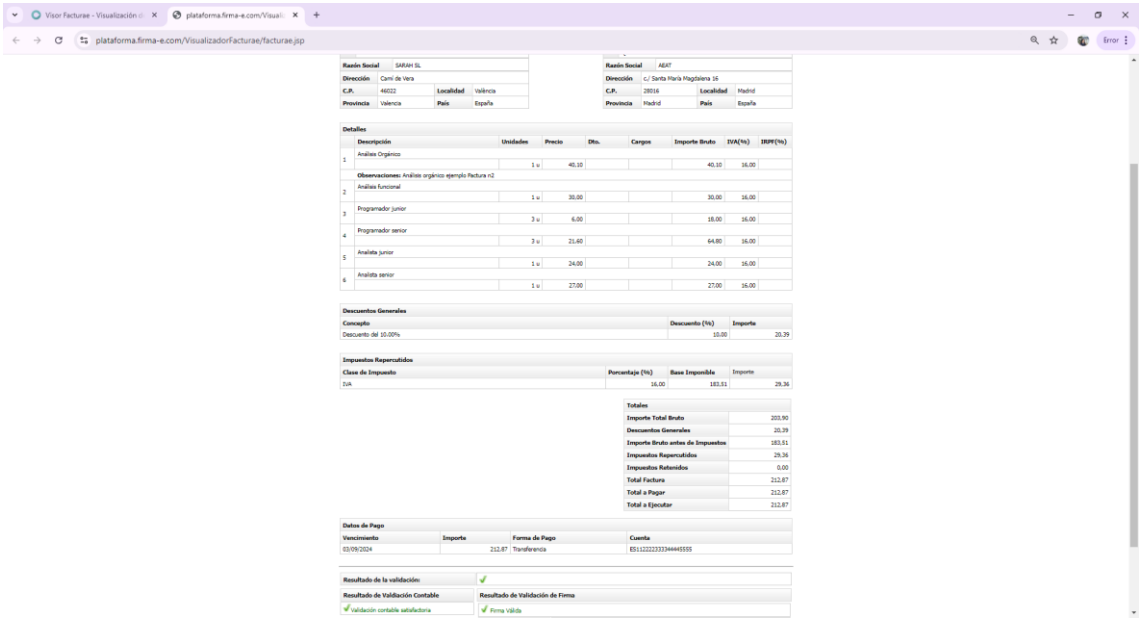


Figura 72. Validación factura plataforma.firma-e.

Por último, vamos a enviar la factura por correo electrónico, ya que forma parte de los requisitos funcionales.

Al estar en un entorno local, se ha configurado una plataforma de entrega de email en línea llamad Mailtrap [31] para comprobar todo el funcionamiento antes de lanzar el proyecto al entorno de producción.

Vamos al listado de factura otra vez, y clicamos sobre el icono del correo. Esto nos abre un modal en el que rellenamos la información como el correo al que queremos enviar la factura, el cuerpo y el asunto y le damos a enviar. Además, se adjuntan por defecto la factura en formato PDF y la factura electrónica firmada en formato XSIG.

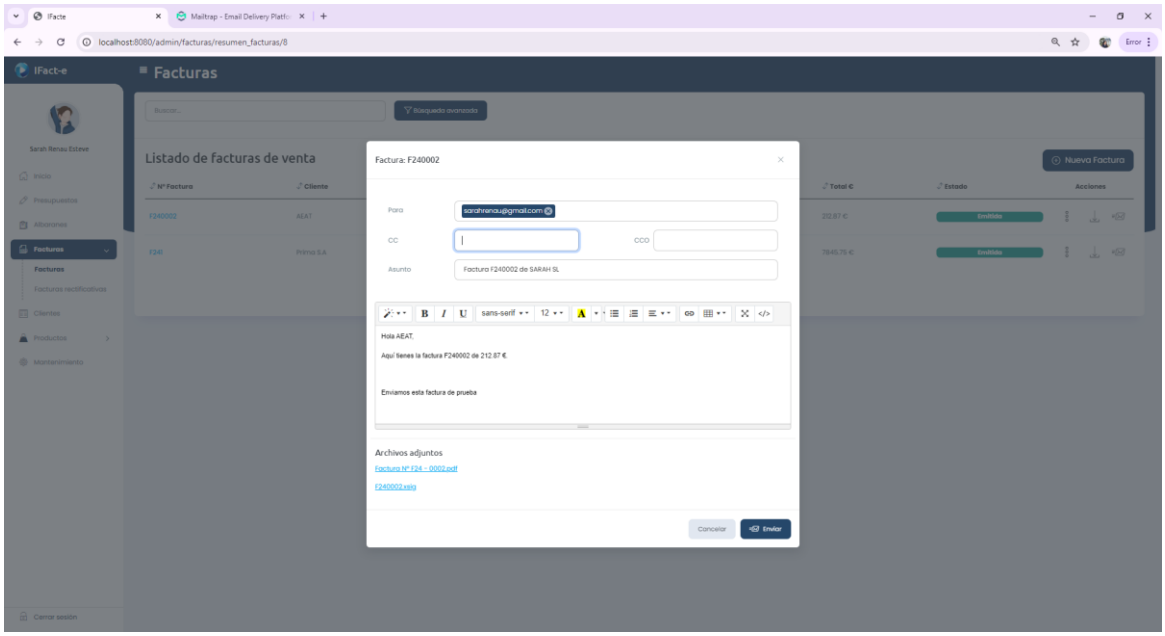


Figura 73. Preparación envío por correo de la factura.

Enviamos el correo y finalmente podemos observar la recepción de este en la plataforma Mailtrap. Vemos que se envía efectivamente al correo que hemos definido y además se reciben los dos documentos adjuntos.

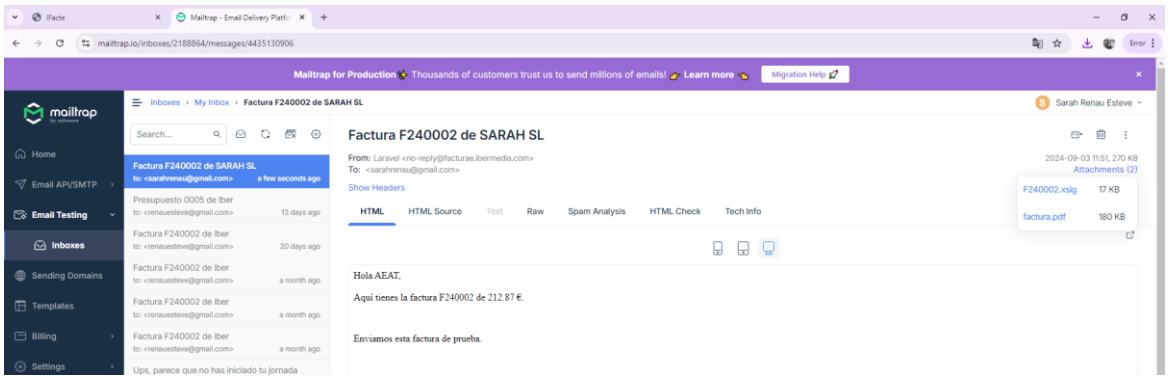


Figura 74. Recepción de la factura por correo electrónico.

Por último, vamos a ver cómo se registran los cambios en la base de datos. Para ello, modificamos el nombre de un cliente, por ejemplo. En la captura se muestra cómo se vería la ficha del cliente en un iPhone 12 PRO, por tal de ver que se cumple la responsividad de las pantallas.

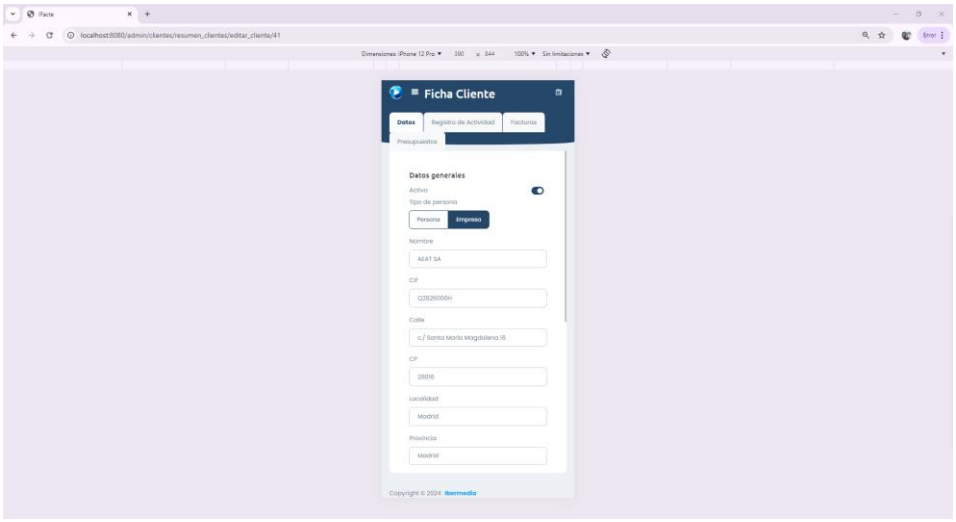


Figura 75. Editar nombre cliente.

Accedemos a la base de datos ifacte_updates y podemos observar cómo se ha quedado registrado el cambio que hemos realizado.

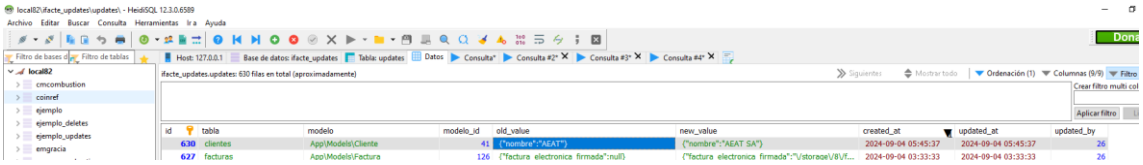


Figura 76. Captura ifacte_updates.

Capítulo 7. Conclusiones

7.1 Conclusiones y propuestas de trabajo futuro

El desarrollo de la aplicación de facturación electrónica ha permitido cumplir con los objetivos iniciales planteados en el proyecto. A lo largo de las diferentes fases, desde el análisis y diseño hasta la implementación y las pruebas, se ha logrado construir una solución que satisface con los requisitos funcionales, además de proporcionar una plataforma eficiente para la gestión de los documentos mercantiles más comunes. Además, esta aplicación fomentará la total transparencia entre las empresas y la Hacienda Pública, y contribuirá significativamente a la reducción del uso de papel.

Cabe destacar que la elección de tecnologías como Laravel para el desarrollo de la aplicación completa, y MySQL como sistema de gestión de bases de datos, ha sido clave. Estas herramientas han permitido construir una aplicación funcional y de fácil mantenimiento de una manera optimizada y más sencilla, en comparación con trabajar libremente con otras tecnologías.

A nivel personal, el desarrollo de esta aplicación ha sido una gran experiencia formativa, ya que me ha permitido entender la implementación de aplicaciones web y mejorar mis habilidades en programación.

Por otro lado, aunque la aplicación cumple con los requisitos planteados, existen una serie de mejoras y funciones que pueden implementarse en el futuro.

En primer lugar, se identifica la necesidad de poder personalizar más el formato PDF de los distintos documentos mercantiles, como, por ejemplo, los colores elegidos o la ubicación de ciertos elementos, como el logo de la empresa o el lugar en el que se muestran las condiciones de pago.

Además, otra característica que mejoraría considerablemente la aplicación sería posibilidad de crear distintos tipos de usuarios, como, por ejemplo, gerente y usuario, otorgando permisos diferenciados sobre ciertos elementos de la aplicación. Otra opción que se considera importante es la creación de un tipo de usuario para asesorías o gestores, permitiéndoles acceder a la información de sus propios clientes para obtener, por ejemplo, el listado de las facturas, sin necesidad de que sea el cliente quien deba entregárselas.

Además de estas nuevas funciones, es importante llevar a cabo el correcto mantenimiento de la aplicación, para prevenir la aparición de nuevos errores y mantenerla siempre actualizada con respecto a nuevas normativas o, simplemente, adaptarse a nuevos estilos.

En resumen, el proyecto ha cumplido con éxito sus objetivos, proporcionando a Ibermedia Servicios TIC SL una herramienta eficaz y flexible para la gestión de facturación electrónica para sus posibles clientes. El proceso de desarrollo ha sido enriquecedor y ha permitido no solo alcanzar los resultados esperados, sino también identificar oportunidades de mejora continua. La aplicación está lista para ser utilizada y ya está publicada en el siguiente dominio: https://ifacte.ibermedia.com/ y servirá como una base sólida para futuras extensiones y mejoras.

7.2 Objetivos de Desarrollo Sostenible (ODS)

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero**				x
ODS 3. **Salud y bienestar.**				x
ODS 4. **Educación de calidad.**				x
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**	x			
ODS 9. **Industria, innovación e infraestructuras.**	x			
ODS 10. **Reducción de las desigualdades.**				x
ODS 11. **Ciudades y comunidades sostenibles.**				x
ODS 12. **Producción y consumo responsables.**				x
ODS 13. **Acción por el clima.**	x			
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas**				x
ODS 17. **Alianzas para lograr objetivos**				x

Tabla 1. Relación del trabajo con los Objetivos Sostenibles de Desarrollo Sostenible.

La aplicación ayuda a autónomos y pequeñas empresas a gestionar sus facturas y documentos de manera más eficiente, lo que puede reducir costos y apoyar el propio crecimiento económico. Además de asegurar la transparencia en sus facturas y contribuir al crecimiento económico del país, cumpliendo así con el ODS 8.

Por otro lado, el proyecto introduce una solución que moderniza y mejora la gestión de documentos comerciales, impulsando la innovación y adopción de tecnología en pequeñas empresas, cumpliendo con el ODS 9.

Por último, al usar facturación electrónica, el proyecto está reduciendo el uso del papel, contribuyendo a la protección del medio ambiente y ayudando a reducir el impacto del cambio climático, aplicando al ODS 13.

Capítulo 8. Bibliografía

[1] «Factura Electrónica - ¿Qué es la factura electrónica ?» Accedido: 27 de julio de 2024. [En línea]. Disponible en: https://www.facturae.gob.es/factura-electronica/Paginas/factura-electronica.aspx

[2] «La factura electrónica en Europa: despliegue», Channel Partner. Accedido: 9 de agosto de 2024. [En línea]. Disponible en: https://www.channelpartner.es/negocios/la-factura-electronica-en-europa/

[3] «BOE-A-2012-14696 Real Decreto 1619/2012, de 30 de noviembre, por el que se aprueba el Reglamento por el que se regulan las obligaciones de facturación.» Accedido: 10 de agosto de 2024. [En línea]. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-2012-14696

[4] R. D.-C. Quipu, «Ley crea y crece: qué es, medidas y entrada en vigor (2024)», Autónomos, empresas y asesorías. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://getquipu.com/blog/ley-crea-y-crece/

[5] «El Gobierno lanza en audiencia la Ley Crea y Crece que facilita la creación de empresas, lucha contra la morosidad comercial y reduce trabas a la expansión del negocio de las pymes». Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://portal.mineco.gob.es/ca-es/comunicacion/Pagines/210727_np_crea.aspx

[6] «XML», *Wikipedia*. 19 de julio de 2024. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=XML&oldid=1235487560

[7] «Universal Business Language», *Wikipedia, la enciclopedia libre*. 6 de mayo de 2024. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Universal_Business_Language&oldid=159954849

[8] «Facturae», Facturae. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://facturae.info/

[9] «¿Qué es la firma digital? Tipos, ventajas y más | Proofpoint ES», Proofpoint. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://www.proofpoint.com/es/threat-reference/digital-signature

[10] «Certificado digital», *Wikipedia, la enciclopedia libre*. 19 de diciembre de 2023. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=Certificado_digital&oldid=156129812

[11] Datisa, «ERP y la gestión de relaciones con los clientes», Datisa ERP. Accedido: 11 de agosto de 2024. [En línea]. Disponible en: https://datisa.es/erp-y-la-gestion-de-relaciones-con-los-clientes/

[12] «Conceptos básicos de HTML - Aprende desarrollo web | MDN». Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics

[13] «PHP: ¿qué es, para qué sirve y cuáles son sus características?», Rock Content - ES. Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://rockcontent.com/es/blog/php/

[14] S. A. IT Customer-Care, «¿Qué es PHP y cómo lo utilizo? | STRATO», STRATO AG. Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://www.strato.es/faq/hosting/que-es-php-y-como-lo-utilizo/

[15] «¿Qué son los lenguajes de programación interpretados? | UNIR». Accedido: 31 de agosto de 2024. [En línea]. Disponible en: https://www.unir.net/ingenieria/revista/lenguajes-programacion-interpretados/

[16] «JavaScript - Wikipedia, la enciclopedia libre». Accedido: 31 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/JavaScript

[17] «jQuery», *Wikipedia, la enciclopedia libre*. 8 de enero de 2024. Accedido: 31 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=JQuery&oldid=156828302

[18] «AJAX», *Wikipedia, la enciclopedia libre*. 23 de junio de 2024. Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=AJAX&oldid=160906811

[19] «MySQL», *Wikipedia, la enciclopedia libre*. 10 de julio de 2024. Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=MySQL&oldid=161233363

[20] «HeidiSQL», *Wikipedia, la enciclopedia libre*. 26 de abril de 2024. Accedido: 15 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=HeidiSQL&oldid=159717805

[21] «XAMPP», *Wikipedia, la enciclopedia libre*. 23 de abril de 2024. Accedido: 16 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=XAMPP&oldid=159620305

[22] J. M. Moreno, *josemmo/Facturae-PHP*. (11 de julio de 2024). PHP. Accedido: 22 de julio de 2024. [En línea]. Disponible en: https://github.com/josemmo/Facturae-PHP

[23] «GitHub», *Wikipedia, la enciclopedia libre*. 19 de junio de 2024. Accedido: 16 de agosto de 2024. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=GitHub&oldid=160846747

[24] «8 ejemplos de diagramas de casos de uso para inspirarse | Boardmix». Accedido: 16 de agosto de 2024. [En línea]. Disponible en: https://boardmix.com/es/articles/use-case-examples/

[25] F. Florez, «REST Countries». Accedido: 31 de agosto de 2024. [En línea]. Disponible en: https://restcountries.com/

[26] «Nifty - Bootstrap 5 Admin Template by ThemeOn». Accedido: 31 de agosto de 2024. [En línea]. Disponible en: https://wrapbootstrap.com/theme/nifty-bootstrap-admin-template-WB0048JF7

[27] «Formato Facturae».

[28] «Facturae-PHP/tests/certs/facturae-private.pem at master · josemmo/Facturae-PHP», GitHub. Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://github.com/josemmo/Facturae-PHP/blob/master/tests/certs/facturae-private.pem

[29] «Visor Facturae - Visualización de facturas electrónicas y validación de firma digital». Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://viewer.facturadirecta.com/

[30] «Firma, Proyectos y Formación S.L. - Visualizador de Factura Electrónica». Accedido: 4 de septiembre de 2024. [En línea]. Disponible en: https://plataforma.firma-e.com/VisualizadorFacturae/

[31] «Mailtrap: Email Delivery Platform». Accedido: 3 de septiembre de 2024. [En línea]. Disponible en: https://mailtrap.io/?gad_source=1&gclid=CjwKCAjw59q2BhBOEiwAKc0ijV8PO1r4xjDRMFhMZapQDx53C5TGxVB0HXFp1bJe7d1A8ZpBmwjqMhoCL6sQAvD_BwE