Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Grado de Ingeniería Informática

▪ Trabajo Fin de Grado ▪

Ingeniería del Software

Autónomo: aplicación de generación y gestión de facturas y presupuestos.

Yara Diaz de Cerio Arzamendi

Noviembre 2022

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Facultad de Informática

Grado de Ingeniería Informática

▪ Trabajo Fin de Grado ▪

Ingeniería del Software

Autónomo: aplicación de generación y gestión de facturas y presupuestos.

Yara Diaz de Cerio Arzamendi

Noviembre 2022

Dirección

Alfredo Goñi Sarriguren

Resumen

Este documento corresponde a la memoria del trabajo de fin de grado desarrollado por la alumna Yara Diaz de Cerio Arzamendi: Autónomo, aplicación de generación y gestión de facturas y presupuestos. Se detallará todo el proceso y todos los pasos que se han seguido para realizar el proyecto.

Se trata de una aplicación multiplataforma desarrollada en Flutter para dispositivos Android, IOS, Windows, macOS, Linux y navegadores web. La aplicación se adaptará a cada dispositivo como si fuese una aplicación nativa, además de ser responsiva, es decir, su diseño se ajustará al tamaño de pantalla o ventana.

Su objetivo principal es facilitar y agilizar el trabajo tanto de gestión como de creación de presupuestos y facturas a los trabajadores autónomos. Al ser multiplataforma, el trabajador podrá acudir a presupuestar al lugar donde trabaja únicamente con su dispositivo móvil, crear el presupuesto en el mismo lugar y enviárselo al cliente directamente o continuarlo después en casa y de manera más cómoda en su ordenador.

Índice de contenidos

Índice de figuras

1

Introducción

Autónomo es una aplicación multiplataforma realizada especialmente para trabajadores autónomos y pretende sustituir el uso de otras aplicaciones como Word o Excel a la hora de generar documentos, tanto presupuestos como facturas. Además, servirá para gestionar los clientes, facturas y presupuestos.

Se ha pretendido crear una aplicación muy sencilla y rápida de utilizar en la que el usuario solo seleccione las partidas introducidas por el mismo, introduzca los datos y el documento se genere automáticamente.

La idea se obtiene tras ver que mi padre, que es electricista, dedica un tiempo excesivo a hacer todos estos documentos en Word con un pequeño cuaderno y calculadora en mano. Al preguntarle por qué no utiliza otros programas siempre he obtenido la misma respuesta, no están adaptados para personas que no tienen conocimientos contables, ya que tienen demasiadas funcionalidades innecesarias que solo hacen que el programa resulte más complicado de utilizar. Todos sus compañeros tienen los mismos problemas, algunos contratan a personas que se encargan de realizar este trabajo, pero la gran mayoría utilizan Word o similares.

La aplicación ha sido desarrollada haciendo uso de Flutter, un SDK novedoso creado en 2017 por Google. También se hace uso de las herramientas que ofrece Firebase, que igualmente es de Google.

2

Planificación inicial

En este capítulo se presenta el estudio del mercado en el que se analizan varias aplicaciones con funciones similares, motivación y objetivos. Después se muestra el estudio hecho para organizar el desarrollo del proyecto, es decir, el diagrama EDT, diagrama de Gantt y una tabla con la estimación de dedicaciones. Por último, se mencionan diferentes riesgos que pueden surgir a lo largo del proyecto junto con su plan de mitigación.

2.1 Antecedentes

Existen muchas aplicaciones de creación de facturas que comparten el mismo problema, todas son muy complejas y se requiere de muchos conocimientos tanto contables como informáticos para poder utilizarlas, por lo que la mayoría de trabajadores acaban optando por Word o programas similares. Este ha sido uno de los puntos más importantes por el que se ha creado esta aplicación, se ha querido obtener un producto muy sencillo e intuitivo.

El usuario tendrá una lista con todas sus partidas que irá creciendo a medida que utiliza el programa. Solo tendrá que seleccionarlas para meterlas a la factura o presupuesto, seleccionar el cliente y añadirle los datos necesarios para crear el documento. En tan solo unos minutos podrá generar un PDF de manera muy sencilla.

Las aplicaciones que existen pretenden gestionar la totalidad del trabajo de oficina de un autónomo: facturación, presupuestos, contabilidad, gestión de almacén, gestión de mano de obra, relación con entidades bancarias, gestiones fiscales, etc. Esto se traduce en programas de gran calidad, pero excesivamente complicados que finalmente el autónomo deja de utilizar.

Las siguientes aplicaciones son un ejemplo de lo anteriormente indicado:

2.1.1 Billage

Herramienta de facturación online, tiene muchas opciones como conexión con el banco, remesas bancarias, control de tesorería. Tiene muchas funcionalidades complicadas. En la Figura 1 se muestra la interfaz de Billage, en el menú de la izquierda se pueden observar las funcionalidades que ofrece.

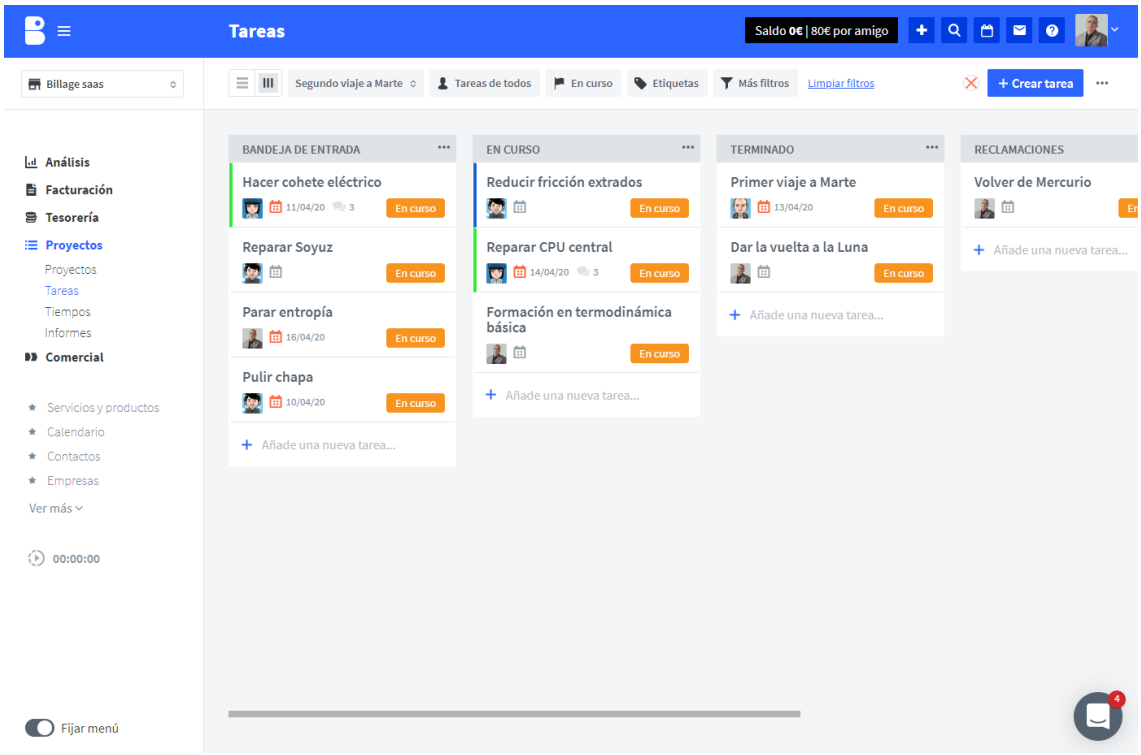


Figura 1. Billage

2.1.2 Debitoor

Es uno de las aplicaciones de facturación más famosas, es un software en la nube que promete tener un diseño muy intuitivo y ser fácil de usar. Es una de las alternativas más llamativas, pero como la gran mayoría, ofrece demasiadas características que pueden complicar su uso. En la Figura 2 se muestra su interfaz.

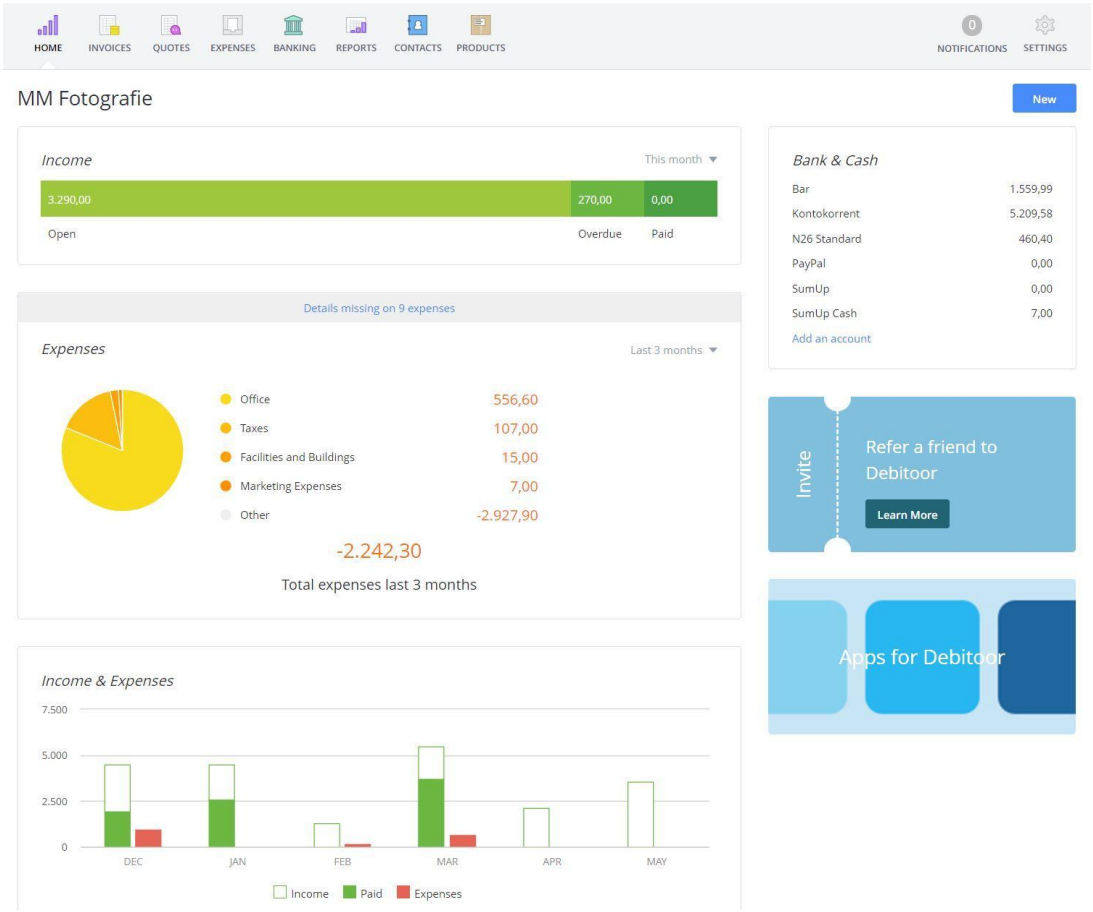


Figura 2. Debitoor

2.1.3 Quipu

Software en la nube de facturación para autónomos y pymes. Como se puede observar tiene demasiadas características que requieren conocimientos contables que no todos los autónomos tendrán. En la siguiente imagen se muestra la aplicación (Figura 3).

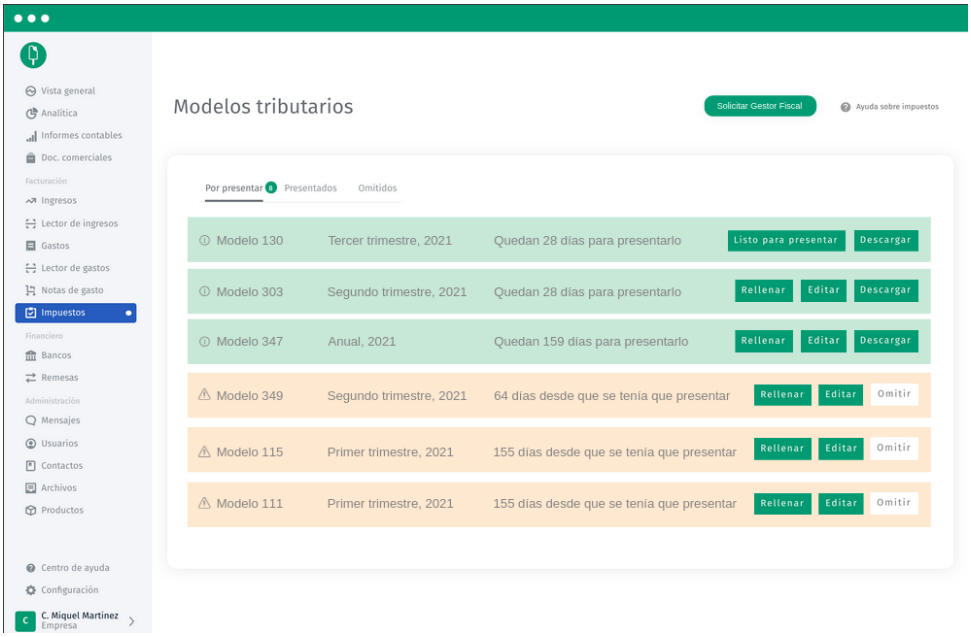


Figura 3. Quipu

2.1.4 Holded

Otro software en línea para autónomos con demasiadas funcionalidades. En la siguiente imagen se muestra la aplicación tanto web como móvil (Figura 4).

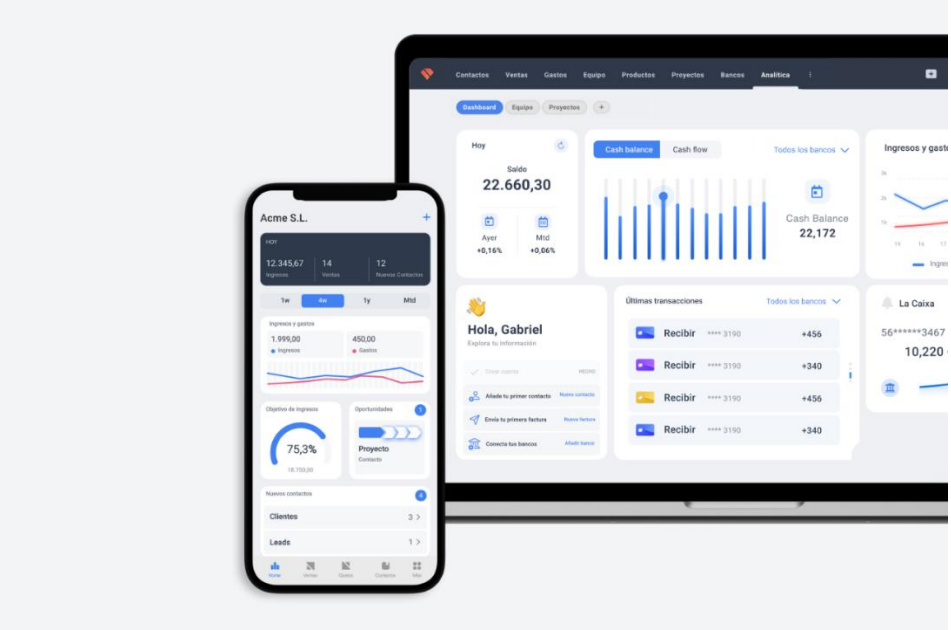


Figura 4. Holded

2.2 Motivación

Actualmente, existe una gran cantidad de trabajadores autónomos, trabajadores manuales especializados que están obligados a realizar tareas para las que no están preparados. Tienen que presupuestar, facturar y llevar el control de la economía de su negocio. Muchos de ellos improvisan y pierden gran parte de su tiempo para poder compensar estas carencias, otros recurren a profesionales con un coste económico y que tampoco les solucionan los problemas reales del día a día.

El procedimiento habitual a la hora de realizar estas tareas suele ser, en primer lugar, acudir libreta en mano a tomar nota de los trabajos a realizar. El siguiente paso, es realizar un presupuesto en base a estos apuntes en un ordenador, en su casa, fuera de horas de trabajo, sin tener una base de datos ni ninguna herramienta que le ahorre tiempo ni le facilite la tarea. Finalmente, llega la hora de realizar la factura, vuelta a empezar de cero, añadir cambios que no contemplaba el presupuesto, subidas o bajadas imprevistas en los precios y pelearse con Excel o con cualquier otro programa de facturación no adaptado para ellos.

Este programa pretende que el trabajador pueda acudir a realizar el presupuesto con una Tablet o con su móvil y que lo pueda hacer sin esfuerzo, disponiendo de unos datos creados por él, a su medida, en base a su experiencia y que después, todo esto, pueda volcarlo directamente a un ordenador y así solo tenga que enviar el presupuesto. Lo mismo a la hora de facturar, que pueda convertir el presupuesto en factura, variando cantidades, porcentajes o nuevas partidas. Todo esto de manera muy intuitiva y sencilla.

Esto permitirá presentar de manera profesional trabajos de oficina para los que no está preparado el autónomo. A la vez que va creando bases de datos con partidas de trabajos, con clientes y todo eso siendo fácil de añadir modificaciones sin tener ningún conocimiento informático ni contable.

2.3 Objetivos y alcance del proyecto

Se pretende conseguir un programa de creación y gestión de facturas y presupuestos responsivo y con posibilidad de uso en diferentes sistemas operativos. El programa será muy intuitivo y sencillo de utilizar para que cualquier persona con unos mínimos conocimientos de informática pueda hacer uso de él.

A continuación, se muestra una lista con los objetivos que se pretenden cumplir en el proyecto:

1. El objetivo principal del proyecto es la creación de una herramienta para que los trabajadores autónomos puedan crear y gestionar de manera sencilla los presupuestos y facturas.
2. Será una aplicación multiplataforma responsiva vía web y además tendrá su propia aplicación para los sistemas operativos más utilizados: Windows, Linux, Mac OS, iOS y Android.
3. El programa será muy sencillo e intuitivo de utilizar para que cualquier persona con conocimientos mínimos de informática pueda hacer uso de él.
4. La aplicación permitirá registrarse a cualquier nuevo usuario que disponga de un correo electrónico. Desde un inicio este deberá de indicar todos sus datos necesarios para la creación del documento PDF.
5. Cuando un usuario inicia sesión, para mayor comodidad, su sesión permanecerá abierta, aunque cierre el programa. Solo se cerrará en el caso de que él mismo lo indique pulsando el botón ‘Cerrar sesión’.

6. Todos los usuarios podrán acceder a todas las funcionalidades.
7. El usuario tendrá la posibilidad de crear un documento, factura o presupuesto, desde cero. Para ello tendrá una lista con todas sus partidas, podrá añadir nuevas, buscar, eliminarlas o modificarlas. Para añadirla al documento sólo tendrá que seleccionarla, indicar la cantidad y se añadirá al documento, el cual podrá visualizar y modificar según desee. Seleccionará entre crear una factura o un presupuesto e introducirá los datos correspondientes, pulsará un botón y se generará su PDF.
8. También podrá crearlas desde un presupuesto existente. Se le mostrará una lista con todos los documentos y podrá buscarlas tanto por nombre de cliente, como por descripción o dirección. Lo seleccionará y se le abrirá la página de modificación de documentos con todos los datos del presupuesto.
9. Podrá visualizar, modificar o eliminar sus facturas, presupuestos o clientes de manera muy sencilla. Además, podrá volver a generar el PDF de cualquier factura o presupuesto en segundos.
10. El usuario podrá modificar sus datos en cualquier momento.
11. Tendrá un menú de navegación superior y uno lateral para poder acceder a cualquier lugar en cualquier momento.

2.4 Estructura de descomposición del trabajo (EDT)

La EDT (Estructura de descomposición de trabajo) es una estructura jerárquica que divide en paquetes de trabajo actividades complejas para lograr completar el proyecto de una manera más sencilla y ordenada.

En el diagrama EDT se diferencian cuatro apartados principales: Gestión, aplicación, memoria final y defensa del proyecto.

2.4.1 Gestión

El apartado de gestión se divide en los siguientes paquetes de trabajo:

- *Planificación:* este paquete incluye todas las tareas necesarias para realizar la planificación inicial.
- *Seguimiento y control:* agrupará las tareas necesarias para garantizar un adecuado desarrollo del proyecto, principalmente, el seguimiento de las dedicaciones, plazos y especificaciones.

2.4.2 Aplicación

En este apartado se incluye todo lo relacionado para realizar la aplicación y asegurar un correcto funcionamiento. Agrupará los siguientes paquetes:

Estudio inicial: Todo lo necesario antes de comenzar el desarrollo de la aplicación

- *Estudio del mercado:* Búsqueda y análisis de aplicaciones de facturación ya existentes. Nuevas funcionalidades y mejoras.
- *Estudio de tecnologías:* obtención de información sobre las tecnologías seleccionadas y estudio de las mismas.
- *Elección de tecnologías:* obtención de información acerca de diferentes tecnologías y selección de la que mejor se adapta al proyecto.

Desarrollo: Todo lo necesario para realizar la aplicación: desde el diseño hasta las pruebas que se realizarán con usuarios.

- *Análisis de requisitos:* *Diseño de los prototipos iniciales,* definición de los requisitos funcionales y no funcionales que deben satisfacerse.
- *Diseño:* mejoras en los diseños de los prototipos iniciales y estudio de requisitos.
- *Implementación:* implementación de los requisitos y despliegue de la aplicación.
- *Pruebas:* pruebas de funcionalidades, de adaptación y con usuarios reales.

2.4.3 Memoria

Realización de la memoria del proyecto.

2.4.4 Defensa del proyecto

Esta sección contiene todo lo necesario para realizar la defensa de manera adecuada, la elección del contenido, guion, preparación de PowerPoint y la preparación de la demostración de la aplicación.

En la siguiente imagen se muestra el esquema EDT (Figura 5).

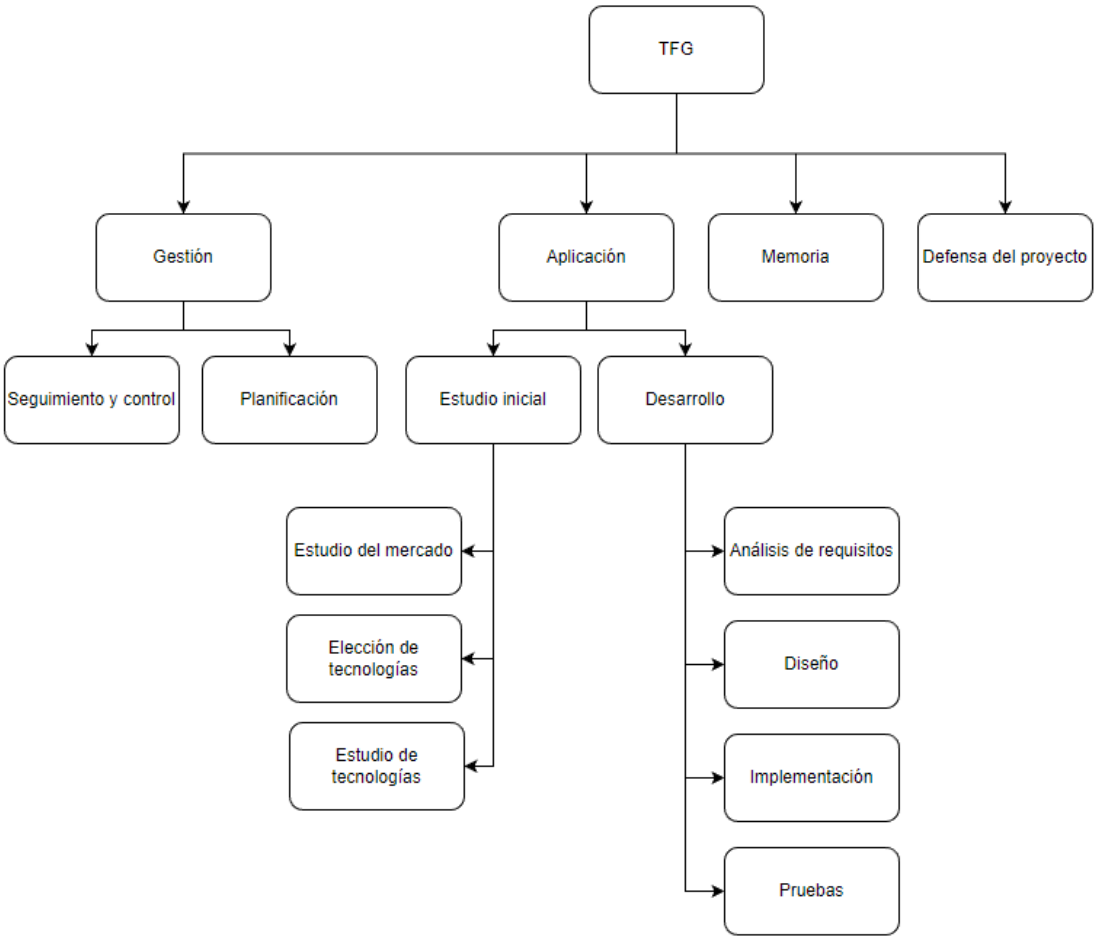


Figura 5. EDT (Estructura de descomposición del trabajo)

2.5 Diagrama de Gantt

En la siguiente imagen (Figura 6) se muestra el diagrama de Gantt planificado inicialmente.

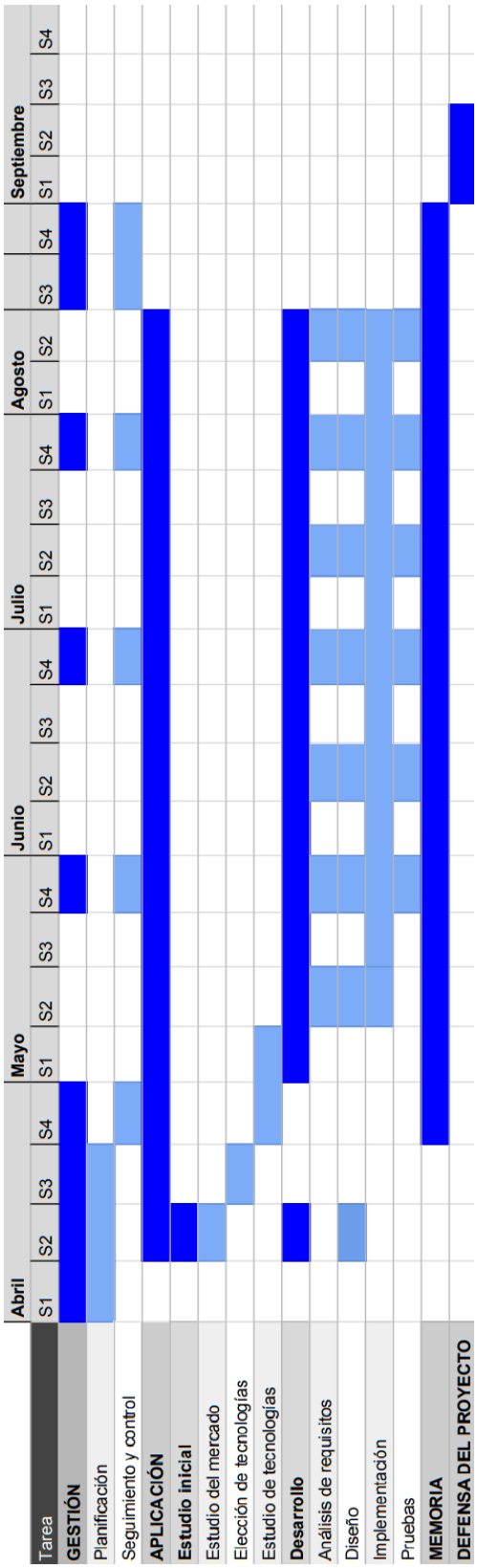


Figura 6. Diagrama de Gantt inicial

2.6 Estimación de dedicaciones

En la Figura 7 se muestra estimación de dedicaciones inicial por cada tarea del EDT.

Tarea	
GESTIÓN	Horas planificadas
Planificación	5
Seguimiento y control	5
APLICACIÓN	-
Estudio inicial	-
Estudio del mercado	2
Elección de tecnologías	2
Estudio de tecnologías	20
Desarrollo	-
Análisis de requisitos	12
Diseño	10
Implementación	150
Pruebas	10
MEMORIA	70
DEFENSA DEL PROYECTO	14
TOTAL	300

Figura 7. Estimación de dedicaciones

2.7 Riesgos

A la hora de realizar el proyecto es importante identificar, evaluar y gestionar los diferentes riesgos que pueden hacer que este se retrase. Se han identificado los siguientes:

2.7.1 Problemas de planificación/tiempo.

La aparición de nuevas tareas o exámenes en alguna asignatura podrían causar retrasos en la planificación.

Probabilidad: Dado la gran dificultad de crear una planificación perfecta existe una probabilidad alta de que esto ocurra, siempre habrá que cambiar alguna cosa.

Medidas a tomar: Se llevarán al día las tareas para que el trabajo acumulado sea menor y no sea necesario parar el proyecto por completo, además de dedicarle más tiempo las próximas semanas.

2.7.2 Cambios en las funcionalidades.

Durante el desarrollo podría surgir la necesidad de modificar o añadir alguna funcionalidad.

Probabilidad: Existe una probabilidad media de que esto ocurra.

Medidas a tomar: Se le dedicará el tiempo necesario al diseño para que las modificaciones sean las mínimas y si aun así existen se dedicará un tiempo extra para solventarlas.

2.7.3 Problemas personales.

En caso de que exista algún problema de salud es posible que no se pueda continuar el trabajo hasta recuperarse.

Probabilidad: Probabilidad media. Dado que el desarrollo del proyecto durará varios meses, existe una probabilidad alta de que haya algún problema de salud.

Medidas a tomar: Si esto ocurre se le dedicarán más horas las semanas siguientes.

2.7.4 Problemas con las tecnologías.

Como se hará uso de tecnologías desconocidas es posible que sea necesario cambiar de herramienta o que se necesite más tiempo del planificado para su estudio. Debido a la falta de experiencia con estas tecnologías posiblemente llevará más tiempo solucionar los errores que puedan ocurrir.

Probabilidad: Existe una probabilidad media de que esto alargue el desarrollo de la aplicación.

Medidas a tomar: Para evitar tener que cambiar de tecnología una vez el proyecto está comenzado, se le dedicará el tiempo necesario a la obtención de información y estudio de las diferentes herramientas.

2.7.5 Pérdida de información.

A lo largo del proyecto habrá mucha información que se puede perder, como documentos, código o páginas con documentación útil.

Probabilidad: Existe un riesgo muy bajo de que esto ocurra y pueda afectar al proyecto.

Medidas a tomar: Respecto al código, este se subirá a GitHub todos los días, así siempre existirá un respaldo a la copia local. De igual manera todos los documentos se guardarán en Drive y además se realizará una copia en local cada semana.

3

Tecnologías

Se muestran las tecnologías analizadas en el estudio inicial del proyecto. Cómo se tomó la decisión del tipo de aplicación que sería, comparación entre las diferentes herramientas estudiadas, justificación de las seleccionadas y por último se explican las tecnologías que se han utilizado.

3.1 Elección de tecnologías

3.1.1 Nativo, híbrido o web

Actualmente, la mayoría de trabajadores autónomos crean sus facturas con programas similares a Word, estos no son los más adecuados para este tipo de tareas. La idea principal fue sustituir su uso por un programa de escritorio para Windows que tuviese como única función la creación de facturas o presupuestos de manera cómoda. Por esta razón en un principio solo se planteó hacer un programa nativo para Windows.

Se ha querido llevar más allá su funcionalidad y hacerla también ejecutable en móviles para así poder crear los presupuestos desde la propia casa del cliente y continuarlos después en su casa. Aquí es donde hubo que tomar una importante decisión: Nativo, híbrido o web.

Una gran ventaja de las aplicaciones web es que pueden ser ejecutadas desde cualquier dispositivo o plataforma, son las más sencillas de mantener y actualizar. Sin embargo, tienen una gran desventaja, van a depender completamente de las funcionalidades que tiene un navegador y no funcionarán sin conexión a internet. Son las más rápidas y fáciles de desarrollar, pero su rendimiento será menor comparado al resto de opciones, además se perderá la posibilidad de acceder a las características hardware del dispositivo. Tampoco es la opción preferida de la gran mayoría de usuarios de dispositivos móviles, aunque el programa se adapte perfectamente a la pantalla siempre es más cómodo utilizar una aplicación para realizar este tipo de tareas por lo que esta opción fue descartada.

A la hora de desarrollar una aplicación multiplataforma las aplicaciones nativas requieren muchas horas de trabajo puesto que cada una tiene un sistema operativo diferente y cada sistema requiere un lenguaje determinado. Las aplicaciones nativas pueden acceder a la mayoría de características hardware y son capaces de adaptarse a todas sus funcionalidades. El mayor inconveniente es que el tiempo de trabajo se multiplicará si se quiere adaptar a varios sistemas operativos.

Por último, están las aplicaciones híbridas, su característica principal es la adaptabilidad de un mismo código a los requerimientos de cada sistema. Permite abordar un mercado más amplio que las anteriores y acceder en cualquier lugar, aunque no tenga conexión. Permite acceder a la gran mayoría de las características hardware y funcionalidades del dispositivo por lo que se obtendrá un producto muy similar a uno nativo. Como desventaja, agregan una capa entre el código y la plataforma, por lo no será tan rápida y potente como puede ser una aplicación nativa, esto será casi imperceptible para el usuario debido a las capacidades de los dispositivos de hoy en día.

Las dos opciones, tanto nativa como híbrida, son opciones válidas pero esta última me ha parecido la más interesante. Al ser un programa que no requerirá de excesiva potencia las ventajas que puede aportar uno nativo no se podrán apreciar.

3.1.2 Tecnologías estudiadas

3.1.2.1 Tecnologías multiplataforma

Una vez decidido que la aplicación será híbrida se han analizado varias alternativas. Como nunca había utilizado nada parecido he buscado y comparado información de varias fuentes. En la siguiente tabla (Figura 8) se muestra un pequeño resumen de las herramientas analizadas:

Figura 8. Tabla de comparación tecnologías multiplataforma

Comparación entre tecnologías principales seleccionadas: Apache Cordova y Flutter

Apache Cordova [1] es un framework para desarrollo de aplicaciones multiplataforma en el que se utilizan herramientas como JavaScript, HTML y CSS. Uno de sus puntos fuertes es que es compatible con la gran mayoría de funciones de los dispositivos móviles, como puede ser la cámara, almacenamiento o GPS.

La idea principal fue desarrollar la aplicación haciendo uso de Apache Cordova y ha sido una de las alternativas más estudiadas. A pesar de que esta opción hubiera sido la más sencilla de realizar, puesto que ya tenía conocimientos y experiencia con HTML5 o JavaScript, fue descartada tras conocer el SDK Flutter, una alternativa más novedosa y con mejores características como las que se mencionan a continuación.

Su interfaz de usuario está basada en web, lo que puede hacer que en muchas ocasiones al ejecutarse en dispositivos móviles no se sientan como nativas. Sin embargo, Flutter ofrece una interfaz muy similar a una nativa, lo que suele ser muy importante para muchos usuarios.

En cuanto al rendimiento, las aplicaciones desarrolladas en Cordova primero son ejecutadas en WebView, esto no se notará en dispositivos de gama alta, pero podría causar problemas en otros más económicos. En cambio, Flutter utiliza su propio motor de renderizado y así elimina la necesidad de que exista un puente de comunicación, lo que ofrecerá al usuario una experiencia más fluida y agradable, aunque su dispositivo sea antiguo.

A continuación, se muestra un pequeño resumen de las características de ambos (Figura 9) [2]:

	Flutter	*Apache Cordova*
Best used for	Any cross-platform apps, from simple to complex ones, especially with animation and graphics	Quick prototypes, simple apps
Performance	Higher performance	Lower performance
Testing and development speed	Supports Hot Reload and automated testing.	Supports LiveReload; testing automation is limited.
UI	Adapts to the used platform. Looks native	UI elements look the same on different platforms, not native
Easy to learn	Requires more time to learn Dart language	Easier to learn — based on web-technologies
Popularity among devs.	More popular	Less popular

Figura 9. Tabla de comparación Flutter y Apache Cordova

Muchos artículos aseguran que Flutter será el SDK con el que se construirán la gran mayoría de las aplicaciones para dispositivos Android, que es probablemente el Sistema Operativo más utilizado a nivel mundial. Por ello, me ha parecido una opción muy interesante para aprender.

Dado que nunca había utilizado Flutter fue necesario hacer varios cursos para comenzar a desarrollar la aplicación. Primero realicé el curso completo de YouTube 'Curso Introducción a Flutter' [3] en el que desarrollé una pequeña aplicación. Después completé otro curso de Udemy [4] para consultar dudas más avanzadas.

3.1.2.2 Base de datos

Uno de los puntos más importantes de la aplicación es la sincronización entre los datos entre móvil y ordenador para que el trabajador autónomo pueda realizar el presupuesto en cualquier lugar, como puede ser la casa del cliente, y pueda después continuarla en el ordenador de su propia casa o pueda sacar la factura final de dicho presupuesto más tarde.

Flutter puede hacer uso de diferentes bases de datos, tanto relacionales como NoSQL u online/offline. Estas son algunas de las más recomendadas:

- SQflite. [5] Base de datos relacional, offline y sin soporte para web.
- Firebase. [6] NoSQL, sincronización de datos entre cliente y servidor, soporte para la mayoría de dispositivos.
- Hive. [7] Base de datos NoSQL y offline

Firebase ofrece dos bases de datos basadas en la nube y que admiten sincronización en tiempo real:

- Realtime Database. [8]Es la base de datos original de Firebase, almacena los datos en un árbol JSON, tiene soporte sin conexión para Apple y Android. Solo se cobra por el ancho de banda y almacenamiento, en general, las tarifas suelen ser mayores que las de Cloud Firestore.
- Cloud Firestore. [9]Nueva base de datos con un nuevo modelo de datos, almacena los datos como colecciones de documentos, tiene soporte sin conexión para Apple, web y Android, y ofrece más funcionalidades. Se cobra por operaciones ejecutadas en la base de datos, la tarifa del ancho de banda y almacenamiento es menor a la de Realtime Database.

Se ha seleccionado Cloud Firestore porque es la que mejor se adapta mejor al proyecto y ofrece mejores prestaciones. Además de ser parte de la plataforma Firebase, que ofrece muchas funcionalidades como pueden ser la autenticación o el hosting. Para aprender a utilizar Cloud Firestore unido a Flutter realicé el curso mencionado anteriormente [4].

3.2 Tecnologías utilizadas

Una vez elegidas las tecnologías a utilizar las describiré más detalladamente en este apartado.

3.2.1 Flutter

Figura 10. Logo Flutter

Flutter [10] es un SDK muy novedoso que lanzó su versión Alpha en mayo de 2017 y es la mejor alternativa multiplataforma que existe a día de hoy. Es un sistema de código abierto creado por Google que permite crear una única aplicación ejecutable en la gran mayoría de las plataformas actuales.

Para interactuar con él se utiliza el lenguaje de programación de código abierto Dart [11], creado también por Google en 2011 para solucionar algunos problemas de JavaScript. Este funciona bajo una máquina virtual, lo que significa que la forma de ejecutarlo es muy rápida y permite recargar la vista sin tener que volver a inicializar todo el contexto gráfico. Se integra

perfectamente con todas las plataformas y se compila en nativo en cada una de ellas, tanto en web como en móvil o escritorio.

Utiliza SKIA [12], que es un motor de renderizado 2D, también de Google. Nos permite crear un UI que se ejecuta en diferentes sistemas y funciona y se visualiza muy parecido a los nativos. El motor gráfico actúa de forma nativa con el botón de pintado de cada SO.

3.2.2 Firebase

Figura 11. Logo Firebase

Es una plataforma creada por Google que nos brinda diferentes herramientas para facilitar el desarrollo de aplicaciones tanto móviles como web. Nos proporciona diferentes servicios como autenticación, almacenamiento en la nube, hosting, bases de datos, configuración remota, notificación de errores, etc.

Firebase [13] es gratuito hasta que haya una gran cantidad de datos, en los siguientes apartados se mostrarán las prestaciones que ofrece en cada servicio el plan Spark, plan gratuito de Firebase, que tiene un límite muy amplio que no se espera superar en un principio. En caso de que la aplicación continúe creciendo mucho se planteará hacer uso del plan Blaze, un plan prepago en el que se pagará según la cantidad de datos que se requiera manejar. [14]

3.2.2.1 Base de datos: Cloud Firestore

Cloud Firestore [9] es una base de datos NoSQL, escalable y flexible que se puede utilizar en la mayoría de entornos y aplicaciones. La información se guarda en documentos, estos tienen diferentes campos en los que se guardarán los valores. Estos documentos se guardan en colecciones, que se usan para mantener todos los datos organizados y hacer consultas. Los documentos también pueden tener subcolecciones, estas son muy útiles para, por ejemplo, guardar cada elemento de una factura. Funciona en tiempo real lo que resulta muy llamativo para 'Autónomo'.

En la segunda columna de la siguiente imagen (Figura 12) se puede observar la cantidad de prestaciones que ofrece el Plan Spark. En la tercera columna se muestra el plan Blaze.

Cloud Firestore

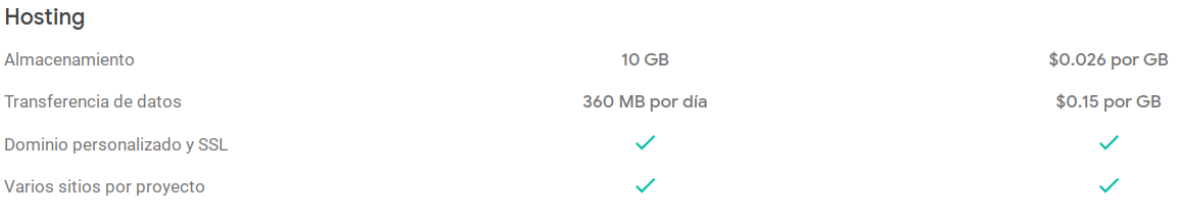
Datos almacenados	1 GiB en total	Sin costo hasta 1 GiB en Luego, pagas $0.108 por GiB
Salida de red	10 GiB por mes	Sin costo hasta 10 GiB po Luego, se aplican los precios de
Operaciones de escritura de documentos	20,000 operaciones de escritura por día	Sin costo hasta 20,000 operaciones d Luego, se aplican los precios de
Operaciones de lectura de documentos	50,000 operaciones de lectura por día	Sin costo hasta 50,000 operaciones d Luego, se aplican los precios de
Eliminaciones de documentos	20,000 eliminaciones por día	Sin costo hasta 20,000 operaciones de Luego, se aplican los precios de

Figura 12. Tabla de presupuesto según planes de Cloud Firestore

3.2.2.2 Hosting: Firebase Hosting

Es un servicio gratuito de Google que permite el alojamiento de contenido web. Hoy en día la seguridad en internet es muy importante, Firebase Hosting [13] consigue aplicaciones muy seguras que proporcionan un certificado SSL. Además, ofrece un servicio de alojamiento muy rápido.

En la siguiente imagen (Figura 13) se muestran las prestaciones que ofrecen ambos planes: en la segunda columna el plan Spark y en la tercera el plan Blaze.



Hosting		
Almacenamiento	10 GB	$0.026 por GB
Transferencia de datos	360 MB por día	$0.15 por GB
Dominio personalizado y SSL	✓	✓
Varios sitios por proyecto	✓	✓

Figura 13. Tabla de presupuesto según planes de Firebase

3.2.2.3 Autenticación de usuarios: Firebase Authentication

Firebase Authentication [13] proporciona servicios de backend, SDK y bibliotecas de UI para autenticar a los usuarios en la app de forma segura. En este caso la autenticación se hace mediante correo electrónico y contraseña.

Como se puede observar en la siguiente imagen (Figura 14) el plan gratuito incluye 'Otros servicios de autenticación', en este se incluye el de verificación por correo y contraseña, que es el único que se usa en Autónomo. El plan Spark en la segunda columna y plan Blaze en la tercera.

Autenticación		
Autenticación telefónica: EE.UU., India y Canadá	10,000 por mes	$0.01 por verificación
Autenticación telefónica: Todos los demás países	10,000 por mes	$0.06 por verificación
Otros servicios de autenticación	✓	✓

Figura 14. Tabla de presupuesto según planes de Firebase Authentication

3.2.2.4 Almacenamiento: Cloud Storage

Cloud Storage [13] es un servicio de almacenamiento seguro y potente ofrecido por Google. En Autónomo se utiliza para almacenar los logos de los usuarios.

En la siguiente imagen (Figura 15) se muestra lo que incluyen el plan gratuito y el de pago (gratuito a la izquierda, de pago a la derecha):

Cloud Storage		
GB almacenados	5 GB	$0.026 por GB
GB descargados	1 GB por día	$0.12 por GB
Operaciones de carga	20,000 por día	$0.05 por cada 10,000
Operaciones de descarga	50,000 por día	$0.004 por cada 10,000
Varios buckets por proyecto	✗	✓

Figura 15. Tabla de presupuesto según planes de Cloud Storage

3.2.3 Otras herramientas

Las siguientes son algunas herramientas que han resultado útiles durante el desarrollo.

3.2.3.1 GitHub

GitHub [15] es una plataforma de alojamiento de código para el control de versiones y colaboración. Se han realizado copias todos los días.

3.2.3.1 Android Studio

Es un entorno de desarrollo de Android que ofrece la posibilidad de probarlo y desplegarlo para diferentes plataformas, ofrece muchos emuladores y utilidades muy útiles para desarrollar en Flutter, es por ello uno de los tres recomendados por la página oficial. [16]

4

Análisis de requisitos

Para desarrollar un buen proyecto software es necesario estudiar en profundidad los problemas que necesita resolver el cliente para conocer las verdaderas necesidades que tendrá el programa y hacer así un buen análisis de requisitos. En este capítulo se presenta el primer prototipo del que se extrajeron los requisitos iniciales y los casos de uso divididos en seis etapas.

4.1 Obtención de requisitos iniciales y primer diseño de la interfaz gráfica

Al inicio de la aplicación se hizo un prototipo en el que participó mi padre (electricista autónomo y la primera persona que utilizará la aplicación), para enseñarle como podría funcionar la aplicación y extraer así los requisitos iniciales. Se comentaron diferentes opciones y funcionalidades que le gustaría que tuviera el programa.

Al comienzo solo se pretendía hacer una herramienta que crease los documentos, por lo que no se muestran todas las funcionalidades finales. En las siguientes imágenes (Figura 16, Figura 17, Figura 18) se muestra el resultado de este primer prototipo:

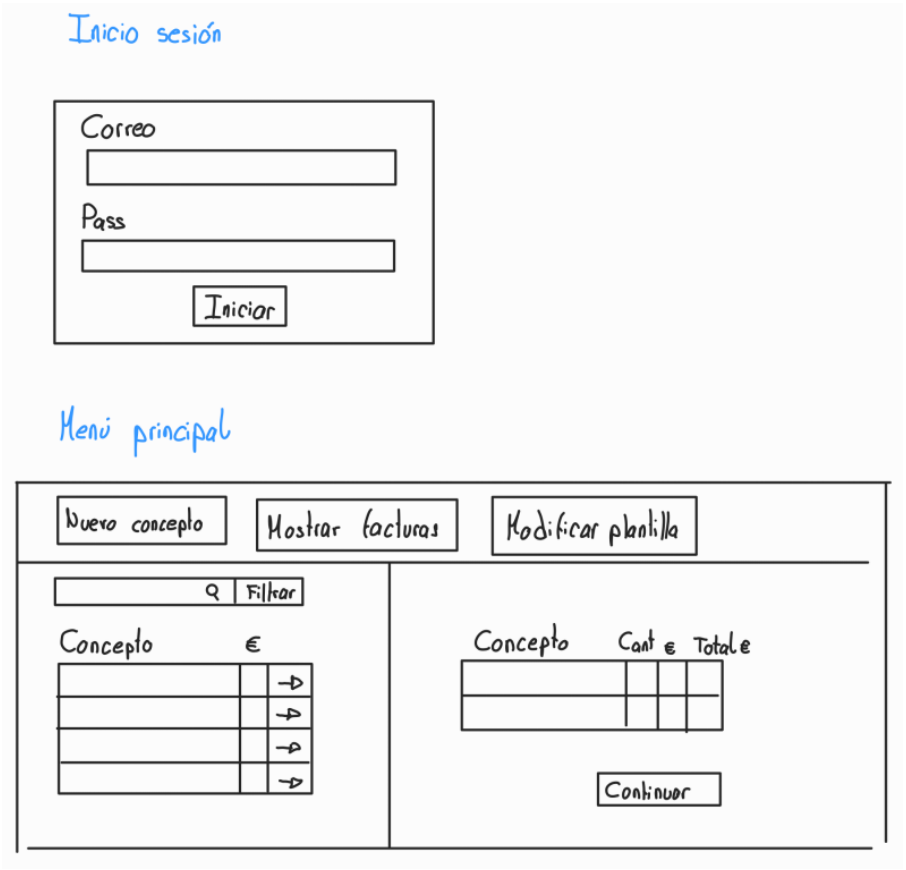


Figura 16. Prototipo inicial. Inicio se sesión y menú principal

Figura 17. Prototipo inicial. Nueva partida y selección de datos de cliente

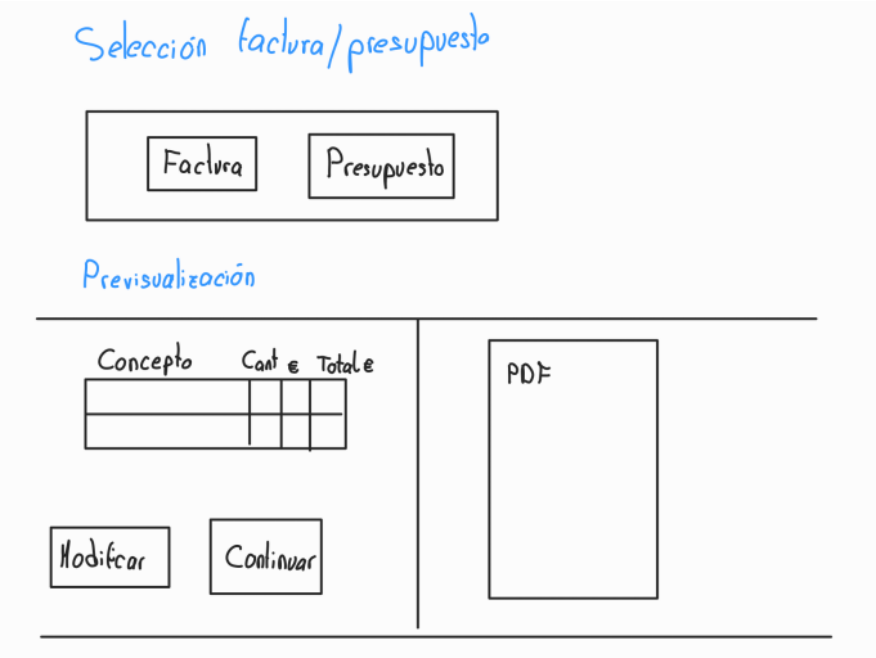


Figura 18. Prototipo inicial. Selección factura/presupuesto y previsualización

4.2 Casos de uso

Como se puede observar en el EDT (Figura 5) las funcionalidades se han ido añadiendo en diferentes partes. Para poder ver mejor los modelos de caso de uso, se mostrarán divididos en sprints junto con una pequeña descripción, el modelo final se encuentra en **Anexo B:** Diagrama de casos de uso final.

Los modelos tienen dos actores:

- Usuario sin loguear: Usuario que o no se ha registrado o no ha iniciado sesión en la aplicación. Solo participará en los casos de uso iniciar sesión y registrar.
- Usuario logueado: Usuario que dispone de cuenta y se ha identificado, es decir, ha iniciado sesión. Este tendrá acceso a todas las funcionalidades.

El flujo de eventos de cada uno de los casos de uso se encuentra en Anexo A: Flujo de eventos.

4.2.1 Sprint 1

En esta primera etapa se realizan todos los casos de uso relacionados a la identificación del usuario en la aplicación: registrar, iniciar sesión y modificar datos personales. (Flujo de eventos: Ver A.1 Sprint 1).

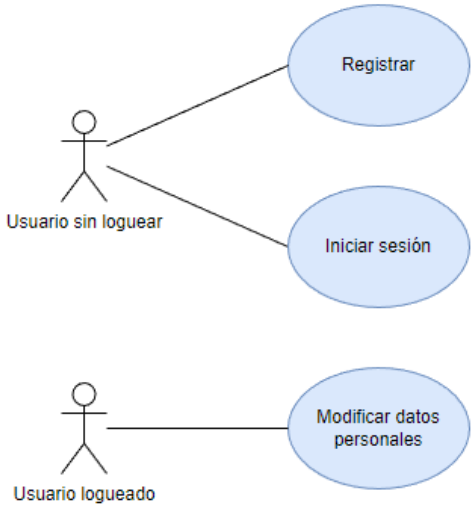


Figura 19. Diagrama de casos de uso Sprint 1

4.2.2 Sprint 2

Casos de uso de la vista que permite añadir las partidas al documento. Cuando el usuario va a crear al documento se le mostrará una lista de partidas (Ver partidas) y sobre ella podrá realizar diferentes acciones, como se puede ver en todos los casos de uso que extienden de Ver partidas.

Desde Ver partidas documento extienden los casos de uso relacionados a modificar esta lista, es decir, modificar el documento actual. (Flujo de eventos: Ver A.2 Sprint 2).

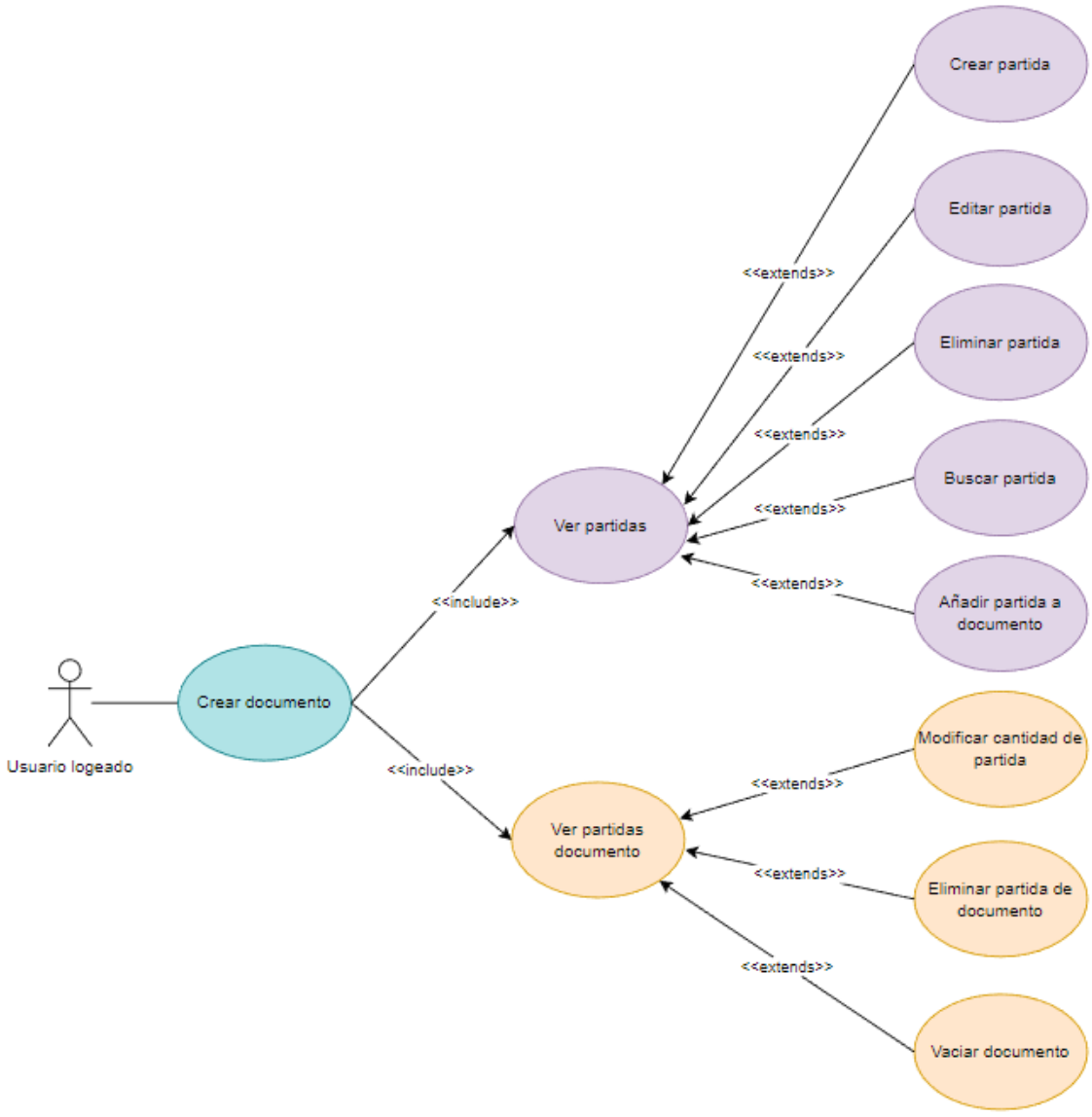


Figura 20. Diagrama de casos de uso Sprint 2

4.2.3 Sprint 3

En este sprint se realizan los casos de uso con los que se continuará la creación del documento, es decir, una página que permitirá al usuario añadir los datos tanto de facturas como presupuestos. Además, se añade el caso de uso que permitirá al usuario cerrar sesión. (Flujo de eventos: Ver A.3 Sprint 3).

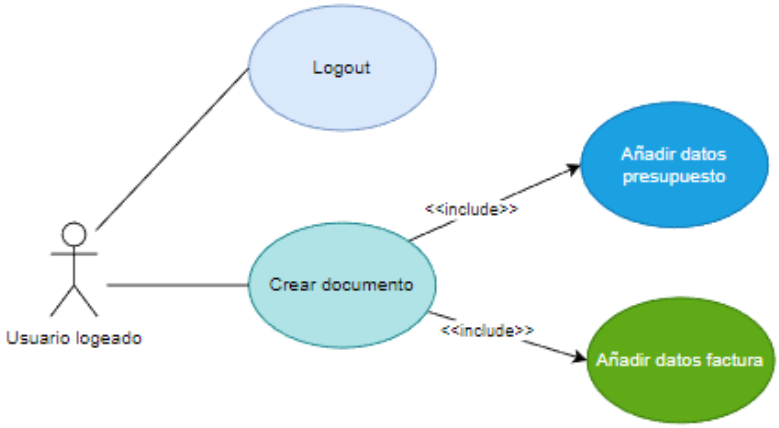


Figura 21. Diagrama de casos de uso Sprint 3

4.2.4 Sprint 4

Se añade la opción de crear clientes. También la generación de documentos en formato PDF y se finaliza el guardado de datos en la base de datos. (Flujo de eventos: Ver A.4 Sprint 4).

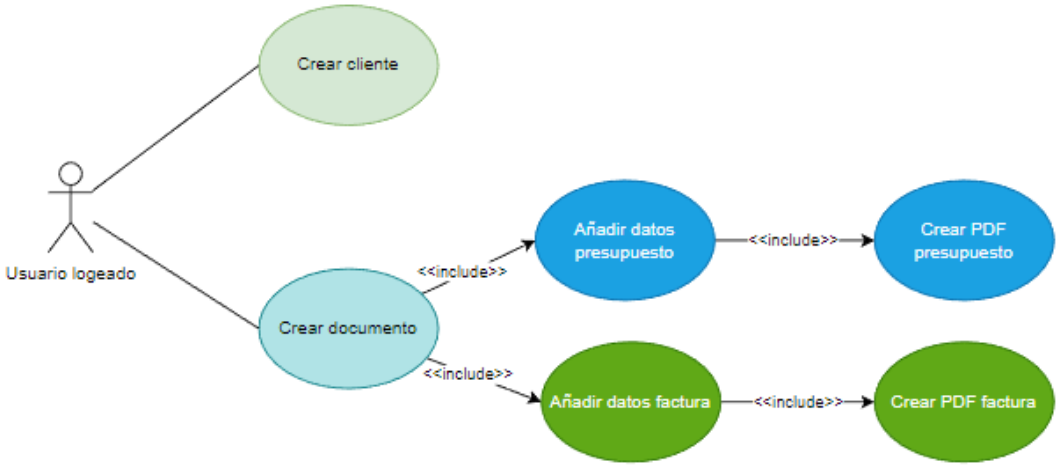


Figura 22. Diagrama de casos de uso Sprint 4

4.2.5 Sprint 5

Se añaden todos los casos de uso relacionados a la gestión de facturas. El usuario verá una lista con todas sus facturas y podrá modificar, buscar o eliminarlo. (Flujo de eventos: Ver A.5 Sprint 5).

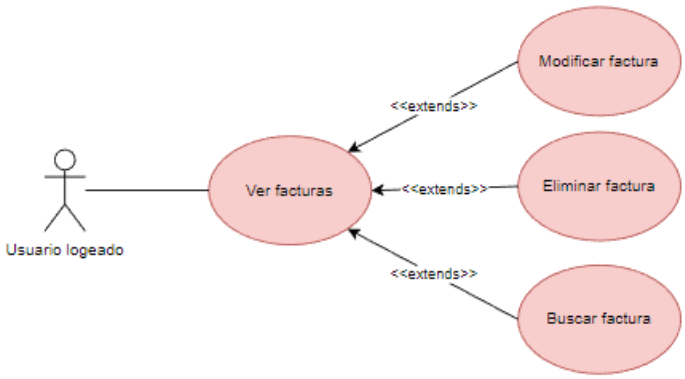


Figura 23. Diagrama de casos de uso Sprint 5

5.2.6 Sprint 6

Se añaden los casos de uso relacionados a la gestión de presupuestos y de clientes. Ver presupuestos se comportará de manera similar a ver facturas. Ver clientes permitirá buscar, editar y eliminar un cliente de la lista. (Flujo de eventos: Ver A.6 Sprint 6).

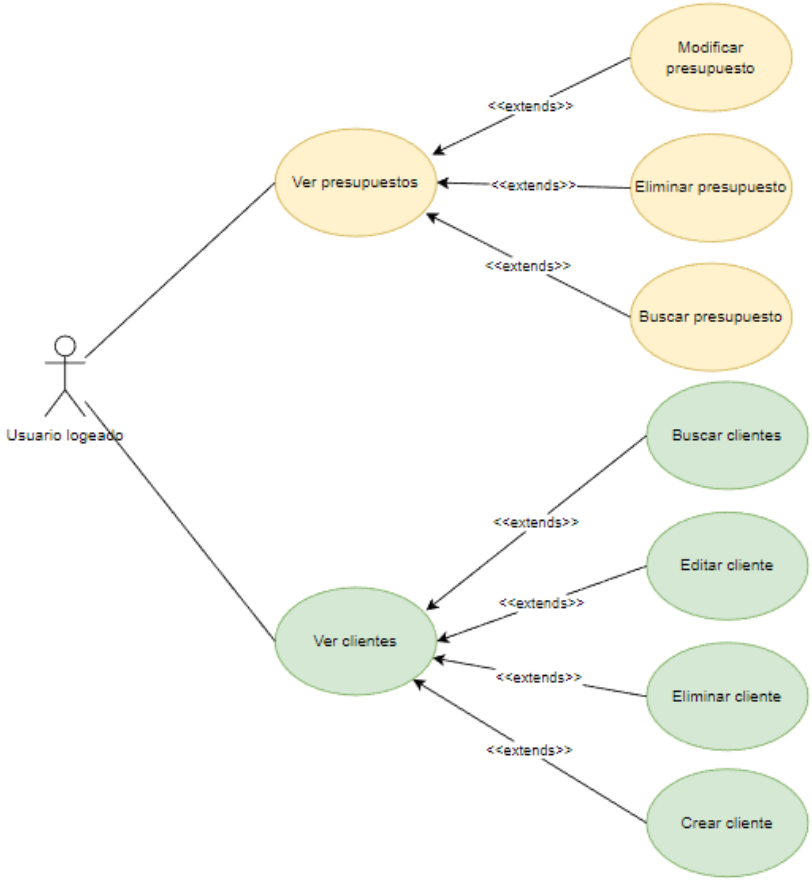


Figura 24. Diagrama de casos de uso Sprint 6

5

Diseño

Se muestra cómo se ha realizado el diseño de la aplicación. Se analizará el patrón de diseño utilizado, la estructura de datos y también se mostrará el diagrama de secuencia de un caso de uso para poder ver mejor esta estructura. Por otra parte, se mostrarán los prototipos iniciales y las decisiones que se han tomado para hacer el diseño visual.

5.1 Arquitectura del sistema: Modelo vista controlador

La aplicación desarrollada sigue uno de los patrones más comunes en el diseño software, la arquitectura MVC (Modelo-vista-controlador, Figura 25). Consiste en repartir las responsabilidades en varias capas para así tener el proyecto mejor organizado y hacer que el mantenimiento sea más sencillo, en caso de hacer cambios en alguna capa estos no afectarán a las demás, respetando así el principio de responsabilidad única. A continuación, se explica cada capa en la aplicación:

5.1.1 Modelo

Se encarga de consultar y manipular los datos obtenidos de la base de datos. En este caso, será la capa que está en contacto directo con Firebase Database, Firebase Authentication y Firebase Storage, se explicará de manera más detallada en el siguiente apartado. El fichero dataBase.dart contiene todas las funciones necesarias para controlar esta.

5.1.2 Vista

Es la representación visual, se encarga de definir como se mostrarán los datos al usuario final. Toda la interfaz de usuario se ha hecho creando y combinando los diferentes Widgets de Flutter. El código de esta capa se encuentra en los directorios screens (vistas principales) y components (Widgets personalizados).

5.1.3 Controlador

El controlador es el intermediario entre las capas modelo y vista. Contiene la lógica que se encargará de actualizar la vista o el modelo en respuesta a las entradas del usuario. Todas estas funciones intermediarias están escritas en el lenguaje de Google, Dart, y se encuentran en controller.dart.

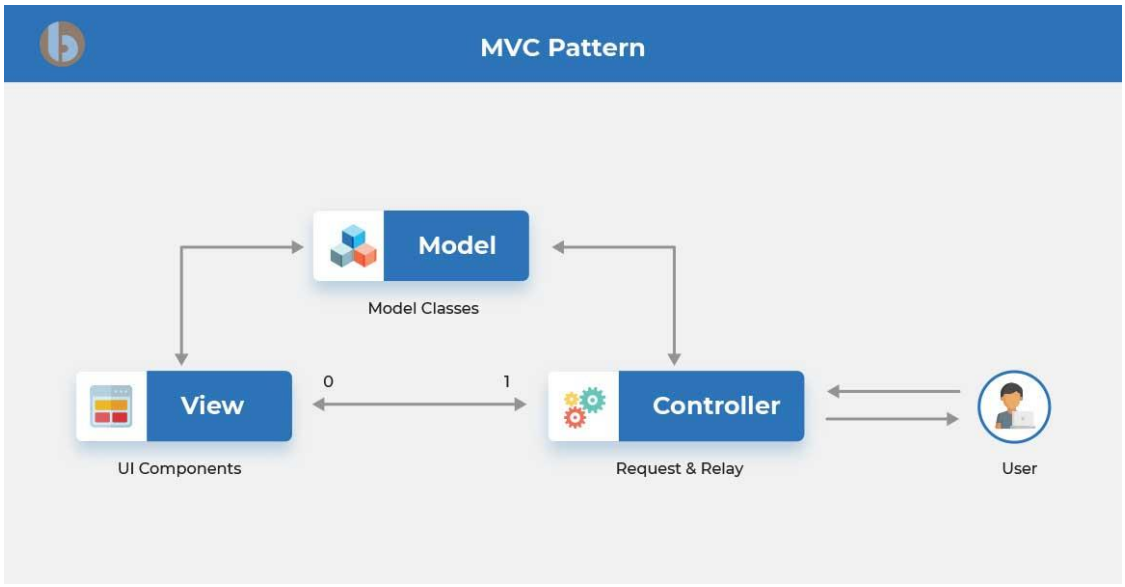


Figura 25. Modelo vista controlador

5.2 Estructura de los servicios Firebase en la aplicación

En este proyecto se hace uso de varios servicios que ofrece Firebase. Con ellos, además de añadir seguridad y agilidad se eliminará la parte del servidor como podría tener una aplicación tradicional. FlutterFire es una herramienta oficial que permite hacer esta conexión de manera fiable y efectiva, ofrece unas librerías muy útiles que permiten acceder a diferentes funciones.

En este apartado se explicará la estructura de la base de datos, el servicio de autenticación y el de almacenaje. En la Figura 26 se puede de manera más clara como actúa la aplicación Flutter con Firebase.

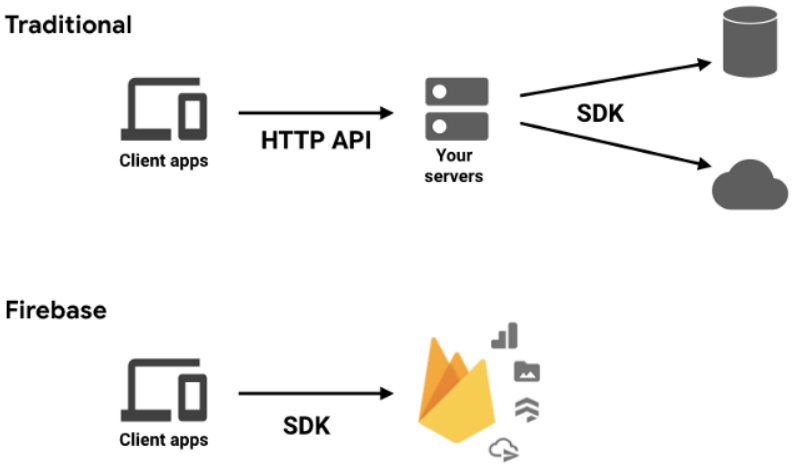


Figura 26. Conexión con Firebase

5.2.1 Base de datos: Firestore Database

Firestore Database es una base de datos NoSQL orientada a documentos. No existen las tablas, solo colecciones de documentos donde cada uno tiene una estructura similar a JSON. Estos documentos tendrán unos campos y a su vez podrán tener otras colecciones.

En teoría, cada documento de una colección puede ser completamente diferente al resto. En el caso de Autónomo todos tendrán el mismo formato, que coincide con la estructura de los modelos.

La aplicación tiene una colección principal llamada 'users', que contiene una lista con todos los usuarios, estos tienen los mismos campos que la clase User (Ver anexo C: C.5.5 User). A medida que el usuario introduce datos se le añadirán nuevas colecciones, un usuario que ha utilizado todas las funcionalidades de la aplicación tendrá las siguientes:

- items: cada documento de esta colección será una partida, tiene los mismos campos que la clase Item (Ver Anexo C: C.5.1 Item).
- currentDocumentItems: partidas añadidas al documento que se está editando, equivalente a la clase DocumentItem (Ver anexo C: C.5.2 DocumentItem).

invoices: colección de facturas, cada factura incluirá sus datos (número, fecha…) y una colección con sus partidas, al igual que la clase Invoice (Ver

6.1.6.1 Invoice).

- budgets: colección de presupuestos, cada presupuesto incluirá sus datos (número, fecha…) y una colección con sus partidas, equivalente a Budget (Ver anexo C: C.5.3 Budget).
- clients: colección que contiene los documentos de cada cliente del usuario, cada cliente tendrá los mismos campos que la clase Client (Ver anexo C: C.5.4 Client).

5.2.2 Autenticación: Firebase Authentication

Firebase Authentication se encarga de guardar de manera segura el correo y contraseña de los usuarios. Cuando el usuario inicia sesión el sistema contacta con el servicio de autenticación para comprobar si los datos son correctos, en caso de serlo, cogerá de Firestore Database todos los datos del usuario.

Los usuarios guardados se verán de la siguiente forma:

Figura 27. Firebase Authentication

5.2.3 Almacenaje: Firebase Storage

Cada usuario, como se puede ver en la clase User (C.5.5 User), tendrá el campo imageUrl. Estas urls redirigen a la imagen del logo que está almacenado Firebase Storage, en el directorio userImages.

5.3 Diagrama de secuencia

En la siguiente imagen (Figura 28) se muestra el diagrama de secuencia del caso de uso iniciar sesión. A pesar de ser un caso de uso muy habitual, se ha seleccionado este por ser uno de los que tienen contacto con varios servicios de Firebase: Firestore Database y Firebase Authentication.

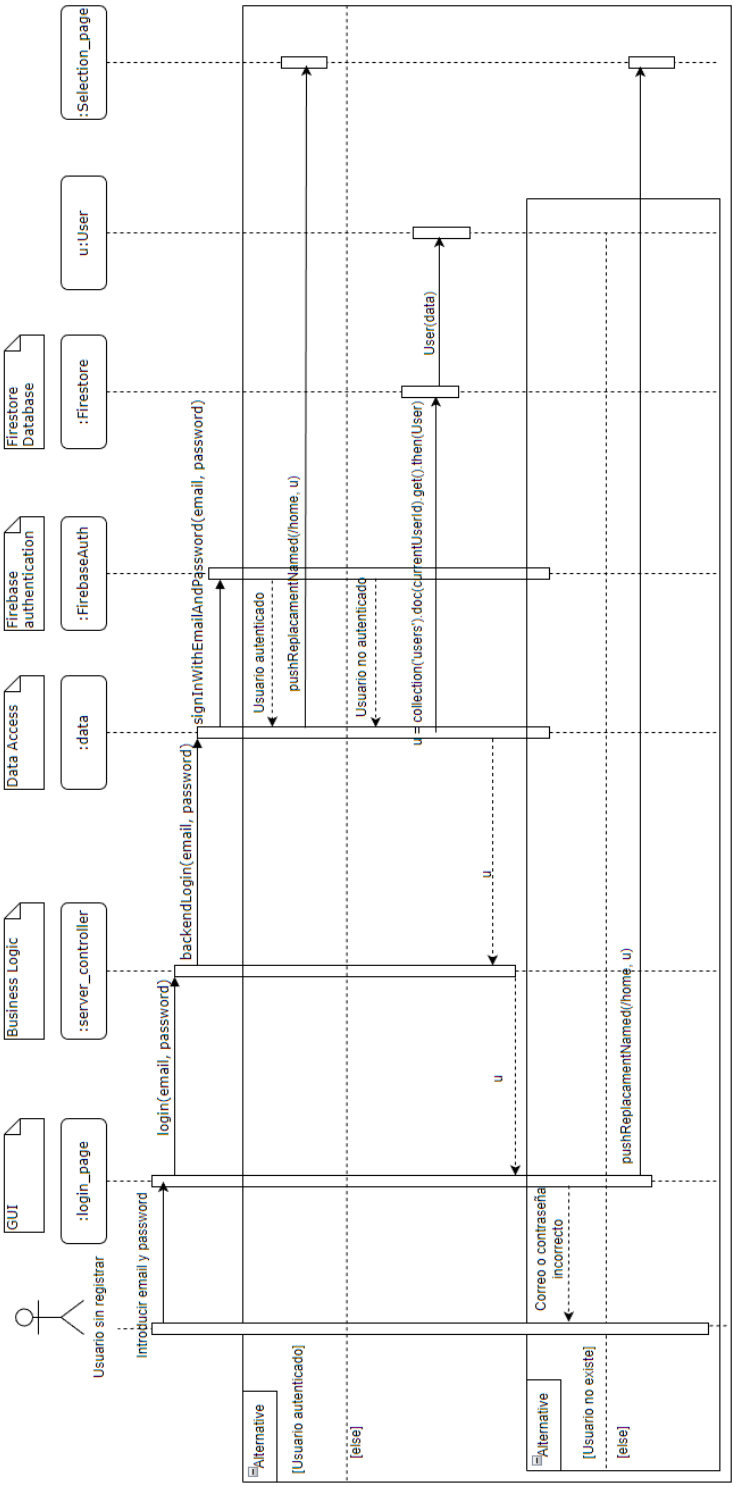


Figura 28. Diagrama de secuencia. Iniciar sesión

5.4 Decisiones acerca del diseño de la interfaz de usuario

5.4.1 Elección de colores e imagen de fondo

El color ha sido un factor importante a tener en cuenta a la hora de hacer el diseño visual. La idea principal fue hacerlo todo en tonos blancos, grises y negros, pretendiendo hacerla más elegante, pero esta fue descartada puesto que quedaba demasiado apagada y sin personalidad, no encajaba con lo que quería que transmitiese la aplicación.

Tras probar varios colores y leer acerca de la psicología de los colores se decidió utilizar el azul. "El azul suele asociarse con empresas grandes y bancos porque no es invasivo y se asocia con la seriedad. Es bueno para temas de salud, tecnología, medicina, ciencia, política y servicios públicos." [17]

Inicialmente el fondo era blanco pero la aplicación quedaba "vacía". Para evitar esto se metió una imagen de fondo en todas las páginas del programa. La imagen (Figura 29) está tomada por mí, se muestra lo que podría ser la creación de un documento antes de la aplicación, es decir, boli, cuaderno y calculadora en mano. Se ha difuminado para no llamar demasiado la atención.

Figura 29. Imagen de fondo

5.4.2 Diseño del logotipo

El logotipo es la primera presentación del producto y puede ser algo importante que el cliente tiene en cuenta a la hora de elegir su aplicación. Se ha intentado hacer un logo muy simple, fácil de recordar y con un aspecto similar al del programa. Este ha sido el diseño final del favicon y también el del icono de la aplicación móvil. (Figura 30)

Figura 30. Logotipo Autónomo

6

Implementación

En este capítulo se muestra cómo se ha realizado toda la implementación del proyecto desde la creación de la aplicación Flutter hasta la subida a la página de hosting o creación del fichero .APK. Se han añadido capturas del código más significativo o diferente, en caso de querer consultar el resto, se encuentra en el repositorio de GitHub del proyecto [18]. Se divide en cinco apartados: Flutter, Firestore, Firebase Storage, despliegue y manual de usuario.

6.1 Flutter

6.1.1 Configuración inicial de la aplicación

El SDK de Flutter está disponible para los principales sistemas operativos tanto para Windows como para Linux, macOS o Chrome OS, todas las indicaciones para hacer la instalación se encuentran en la página oficial [19], en este caso se ha desarrollado todo en Linux.

El primer paso a la hora de realizar un proyecto en Flutter es obtener el SDK, descomprimirlo y agregarlo al PATH. Este nos ofrece una herramienta con la que se podrán ver las dependencias instaladas y las necesarias para completar la instalación, para ello se ejecuta el comando Flutter doctor, que verificará tu entorno y mostrará toda la información por terminal.

Las aplicaciones de Flutter se pueden construir utilizando cualquier editor de texto, pero los más recomendados por la página oficial son Android Studio, IntelliJ y VS Code. En este proyecto se ha utilizado Android Studio, dado que posee características para desarrollar de una forma más óptima y productiva en los dispositivos móviles.

Una vez están todas las dependencias instaladas se instala Android Studio con los complementos Flutter y Dart.

6.1.2 Estructura de los directorios

Tras crear un proyecto configurado para que pueda ser ejecutado en cualquier sistema operativo, Flutter creará automáticamente varios directorios principales, uno por cada sistema operativo y lib, que será el lugar donde estará el código común, código que leerán todos los sistemas.

El directorio lib contiene en su raíz el fichero 'main.dart' (más adelante), que se encarga de inicializar los widgets, la base de datos y la parte principal del programa, 'app.dart' (más adelante). También contiene el fichero 'firebase_options.dart', fichero de configuración de la base de datos.

El directorio lib a su vez contiene dos carpetas principales:

- backend. Contiene el siguiente directorio y fichero:
    - models (más adelante): aquí se encuentran todos los modelos. Usuario, Cliente, Partida…
    - dataBase.dart (más adelante): fichero con acceso a la base de datos Firestore, a Firebase Authentication y a Firebase Storage.
- src. Contiene los siguientes directorios:
    - components (más adelante): pequeños widgets que se han personalizado para añadir en las diferentes pantallas.
    - connection (más adelante): contiene toda la lógica de negocio, el fichero server_controller.dart.
    - Screens (más adelante): todos los ficheros de la interfaz gráfica.
    - utils (más adelante): otras clases útiles, como puede ser ResponsiveLayout, encargada principal de comprobar el tamaño de pantalla y hacer que todo esté correctamente ajustado. O el api encargado de generar PDFs.

Esta estructura puede verse en el Anexo C: C.1 Estructura de los directorios.

6.1.3 Ficheros principales

6.1.3.1 pubspec.yaml

Cuando se crea un proyecto Flutter se genera automáticamente un fichero pubspec básico. Contiene los metadatos sobre el proyecto que tanto Flutter como Dart necesitan conocer. Está escrito en YAML y en él se especifican las dependencias que requiere el proyecto, imágenes,

fuentes y paquetes. Cada vez que se actualiza será necesario ejecutar en la terminal los comandos 'Pub get' y 'Pub upgrade' para que se apliquen los cambios.

Las dependencias a destacar del proyecto se muestran en el Anexo C, C.2 pubspec.yaml :

6.1.3.2 main.dart

Se encarga de inicializar los widgets, la base de datos y la parte principal del programa ('app.dart'). En la siguiente imagen (Figura 31) se muestra el código de la misma.

```dart
import ...

void main(){
  WidgetsFlutterBinding.ensureInitialized();
  Firebase.initializeApp();
  runApp(const MyApp());
}
```

Figura 31. Implementación app.dart

6.1.3.3 app.dart

Este será el fichero en el que está toda la configuración del programa, en él se establecerán el nombre, tema y todas las rutas del programa.

El fichero devolverá un Widget de tipo MaterialApp con todos estos datos, el parámetro onGenerateRoute recibirá un parámetro con la configuración y a su vez devolverá un objeto de tipo MaterialPageRoute, que contendrá el contexto del programa. Dentro del builder se ha creado un switch para que devuelva la página correspondiente de la ruta que se le pasa. Por defecto, el programa redirigirá al usuario al login.

En la siguiente imagen se muestra un fragmento del código de app.dart, donde se puede ver la redirección a dos de las páginas.

```dart
ServerController _serverController = ServerController();

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Autónomo',
      theme: ThemeData(
        brightness: Brightness.light,
        primaryColor: Colors.blueAccent

      ),  // ThemeData
      initialRoute: "/",
      onGenerateRoute: (RouteSettings settings){
        return MaterialPageRoute(builder: (BuildContext context){
          switch (settings.name) {
            case "/":
              return LoginPage(_serverController, context);

            case "/home":
              User loggedUser = settings.arguments as User;
              _serverController.loggedUser = loggedUser;
              return HomePageSelection(serverController: _serverController, context: context);
```

Figura 32. Fragmento de código de main.dart

6.1.4 Implementación de la interfaz gráfica (directorio src/screens)

Cada clase de esta capa tiene un método indispensable llamado build que devuelve un objeto de tipo Widget. Este es el principal constructor de la interfaz gráfica, que está formada de varios widgets que se combinan entre sí y forman un árbol.

Además de los objetos que tiene definidos Flutter, se hace uso de otros Widgets personalizados de tipo StatelessWidget (Widget que no cambiará de estado), que también tienen forma de árbol y están almacenados en el directorio 'components', se mencionarán más adelante (6.1.5 Implementación de la interfaz gráfica. Pequeños Widgets personalizados (directorio src/components)).

En este apartado solo se explica la implementación de la página de inicio de sesión y la de añadir partidas al documento, el resto se encuentran en Anexos: C.3 Implementación de la interfaz gráfica (directorio src/screens).

6.1.4.1 LoginPage

LoginPage es una clase de tipo StatefulWidget, está dentro del fichero login_page.dart. Como parámetros tiene serverController y context. Cuando se inicializa, como en cualquier fichero.dart, se ejecuta la función initState, que ejecutará a su vez isLoggedIn para comprobar si el usuario ya tenía la sesión iniciada. isLoggedIn hará uso de la dependencia 'shared_preferences' [20], que se encarga de mantener el usuario guardado cuando el usuario cierra la aplicación sin cerrar sesión. Si este tenía la sesión iniciada lo redirigirá a la página inicial de selección.

_registerUser se ejecutará cuando el usuario pulsa sobre el texto 'Aún no estás registrado?, su única función será la de redirigirlo a la página de registro.

Por último _login se ejecutará cuando el usuario pulse el botón de inicio de sesión, si los datos son válidos, iniciará sesión con Firebase Authentication, recogerá los datos correspondientes a Firebase Database, guardará el usuario con 'shared_preferences' y lo redirigirá al menú de selección inicial.

Las variables email y password son las utilizadas para recoger la información del formulario. msgError se utiliza para mostrar mensajes que indicarán, por ejemplo, cuando el usuario indicado no existe. loading es un booleano que será true mientras se reciben los datos de la base de datos, mostrará un icono de carga. showPassword también es un booleano, se utiliza para controlar la visibilidad de la contraseña si el usuario pulsa el botón de hacerla visible.

En la siguiente imagen (Figura 33) se muestra el código de esta clase.

```dart
class LoginPage extends StatefulWidget {
  final ServerController serverController;
  final BuildContext context;
  const LoginPage(this.serverController, this.context, {Key? key})
      : super(key: key);

  @override
  State<LoginPage> createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  final GlobalKey<FormState> _formKey = GlobalKey<FormState>();
  String email = "";
  String password = "";

  String _msgError = "";
  bool _loading = false;
  bool showPassword = false;

  @override
  Widget build(BuildContext context) {...}

  @override
  void initState() {...}

  Future<void> _login() async {...}

  void _registerUser(BuildContext context) {...}

  Future<void> isLoggedIn() async {...}
}
```

Figura 33. Fragmento login_page.dart

6.1.4.2 createDocumentResponsivePage, createDocumentDesktopPage, createDocumentMobilePage

Estas clases trabajan en conjunto y se encargan de crear la vista principal de creación de documentos. La primera en ejecutarse será createDocumentResponsivePage, clase de tipo StatefulWidget. Esta se encargará de crear la barra superior junto con sus opciones y el formato de las páginas, como puede ser el tema o la imagen de fondo. Su 'body' tiene un 'Container' que tiene como hijo 'ResponsiveLayout' (6.1.9.3 ResponsiveLayout), al que se le pasarán como parámetros createDocumentDesktopPage y createDocumentMobilePage.

La clase ResponsiveLayout será la encargada de seleccionar la interfaz más adecuada según el tamaño y formato de pantalla. En la siguiente imagen (Figura 34) se puede ver la estructura de createDocumentResponsivePage.

```dart
Widget build(BuildContext context) {
  bool isPC = Responsive.of(context).isPC();
  return Scaffold(
      appBar: AppBar(...),  // AppBar
      drawer: UserDrawer(
        serverController: widget.serverController,
      ),  // UserDrawer
      body: Container(
          decoration: BoxDecoration(...),  // BoxDecoration
          child: BackdropFilter(
              filter: ImageFilter.blur(sigmaX: 10.0, sigmaY: 10.0),
              child: Container(
                decoration:
                    BoxDecoration(color: Colors.white.withOpacity(0.5)),
                child: ResponsiveLayout(
                    mobileBody: CreateBudgetMobile(
                        serverController: widget.serverController,
                        context: widget.context),  // CreateBudgetMobile
                    desktopBody: CreateBudgetDesktop(
                        serverController: widget.serverController,
                        context: widget.context)),  // CreateBudgetDesktop, ResponsiveLayout
              )))  // Container, BackdropFilter, Container

  );  // Scaffold
```

Figura 34. Implementación de método build en createDocumentResponsivePage

createDocumentDesktopPage y createDocumentMobilePage tienen la misma funcionalidad, solo que cambiará la forma en la que se muestran los elementos para que se adapten mejor.

En pantallas grandes se construirá createDocumentDesktopPage. Se mostrarán dos columnas, una con todas las partidas que tiene el usuario y otra con las partidas que introduce al documento.

En pequeñas pantallas se construirá createDocumentMobilePage. Se mostrará una barra superior en la que se podrá seleccionar entre si ver la lista de partidas o ver la lista de partidas añadidas al documento.

6.1.5 Implementación de la interfaz gráfica. Pequeños Widgets personalizados (directorio src/components)

Como se ha mencionado anteriormente, estos son pequeños widgets que se utilizan en las pantallas principales, todos son de tipo StatelessWidget. La mayoría de estos solo contienen las variables necesarias con las que se construirá el Widget y el método build. En este apartado solo se muestra la clase budgetWidget, el resto se explican en Anexos: C.4 Implementación de la interfaz gráfica. Pequeños Widgets personalizados (directorio src/components).

6.1.5.1 budgetWidget

Widget de cada elemento de la lista de presupuestos, se encuentra en el fichero budget_widget.dart. Tiene como variables index, Budget, serverController y budgetDeleteCallBack.

En el método build se decidirá el color principal del elemento y se calculará su precio total, después se creará el Widget haciendo uso de otros más pequeños.

Su único método servirá para abrir en una nueva pestaña el presupuesto en formato PDF.

En la Figura 35 se muestran la constructora y parámetros de esta clase. En la Figura 36 se muestra un fragmento del método build.

```dart
typedef BudgetCallBack = void Function(Budget i);

class BudgetWidget extends StatelessWidget {
  final Budget budget;
  final int index;
  final ServerController serverController;
  final BudgetCallBack budgetDeleteCallBack;

  const BudgetWidget(
      {Key? key,
      required this.serverController,
      required this.budget,
      required this.index,
        required this.budgetDeleteCallBack})
      : super(key: key);
```

Figura 35. Fragmento del código inicial de budget_widget.dart

```dart
@override
Widget build(BuildContext context) {
  Color? backColor = Colors.white;
  if ((index % 2) == 1) {
    backColor = Colors.blue.shade100;
  }
  var suma = 0.0;

  if (budget.billItems.isNotEmpty) {
    suma = budget.billItems
        .map((item) => item.price * item.quantity)
        .reduce((item1, item2) => item1 + item2);
  }

  return GestureDetector(
    onTap: () => _crearPresupuesto(context),
    child: DecoratedBox(
      decoration: BoxDecoration(color: backColor),
      child: Column(
        children: [
          Row(
            children: [
              Expanded(
                child: ListTile(
                  leading: Text(
                    budget.date,
                    style: const TextStyle(
                      fontSize: 16,
                    ),  // TextStyle
                  ),  // Text
                  title: Text(
                    budget.client.name,
                    style: const TextStyle(fontSize: 16),
                  ),  // Text
                  subtitle: Text(budget.description),
```

Figura 36. Fragmento de método build de budget_widget.dart

6.1.6 Modelos (directorio backend/models)

Todos los modelos de la aplicación. Se hará mención de todos los métodos de cada clase, pero dado que todas tienen un formato muy similar en este apartado solo se mostrará Invoice, el resto se encuentran en: C.5 Modelos (directorio backend/models).

6.1.6.1 Invoice

Fichero invoice.dart. Contiene todos los parámetros de una factura: id, userId, invoiceNum, billNum, description, invoiceItems, client, user y date.

Además de la constructora, tiene los setters setId, setUserId y setInvoiceItems. Los primeros dos se utilizarán a la hora de generar cualquier factura. Se le asignará un id generado de forma aleatoria haciendo uso del paquete de Flutter uuid (Figura 51). El userId coincidirá con el id de Firebase Authentication.

setInvoiceItems se utilizará al recoger los datos de Firestore.

Como el resto de las clases de los modelos, tendrá la función toString y toFirestore, esta última se utiliza para convertir el objeto al formato utilizado en Firestore, Map<String, dynamic>.

6.1.7 Lógica de negocio (directorio src/connection)

Fichero server_controller.dart. Contiene todos los métodos necesarios para hacer de intermediario entre el controlador de la base de datos y las vistas. En la Figura 37 se muestran varios métodos.

```
Future<void> init(BuildContext context) async {
  await servus.init();
}

///LOGIN////////////////////////////////////////////////////////////
Future<User?> login(String userName, String password) async {
  return await servus.backendLogin(userName, password);
}

logout() async {
  return await servus.logout();
}

bool isLoggedIn() {
  return servus.isLoggedIn();
}

///USER/////////////////////////////////////////////////////////////
Future<User> getCurrentUser() async {
  return await servus.getCurrentUser();
}

Future<bool> addUser(User nUser) async {
  return await servus.addUser(nUser);
}
```

Figura 37. Fragmento de server_controller.dart

6.1.8 Acceso a base de datos (directorio backend)

Contiene todos los métodos que se comunican con los servicios de Firebase: Firebase Storage, Firebase Authentication y Firestore Database. En la Figura 38 se puede ver la función que se utiliza al registrar un nuevo usuario.

```dart
Future<bool> addUser(User nUser) async {
  try {
    await _auth.createUserWithEmailAndPassword(
        email: nUser.userName.toLowerCase().trim(),
        password: nUser.password.trim());
    final au.User user = _auth.currentUser as au.User;
    final uid = user.uid;
    nUser.setId(uid);
    FirebaseFirestore.instance
        .collection('users')
        .doc(uid)
        .set(nUser.toFirestore());
  } catch (error) {
    return false;

  }
  return true;
}
```

Figura 38. Función addUser para acceso a base de datos

6.1.9 Otros útiles (directorio src/utils)

Clases que serán de utilidad para generar pdfs y realizar la adaptación de pantalla de manera correcta. Dado que la clase PdfInvoiceGenerator es muy similar a PdfBudgetGenerator, que se explicará a continuación, se explicará en Anexos: C.6 Otros útiles (directorio src/utils)

6.1.9.1 PdfApi

Esta clase está en el fichero pdf_api.dart. Contiene todas las funciones esenciales para crear y abrir documentos PDF tanto en móvil como en web. Tiene las funciones saveDocument, saveDocumentWeb, openFile y openFileWeb. En la Figura 39 se pueden observar dos de estas funciones.

```dart
class PdfApi {
  static Future<File> saveDocument({required String name, required Document pdf,}) async {
    final bytes = await pdf.save();
    final dir = await getApplicationDocumentsDirectory();
    final file = File('${dir.path}/$name');

    await file.writeAsBytes(bytes);
    return file;
  }

  static saveDocumentWeb({required String name, required Document pdf})  async {
    var  data = await pdf.save();
    Uint8List bytes = Uint8List.fromList(data);
    final blob = html.Blob([bytes], 'application/pdf');
    final url = html.Url.createObjectUrlFromBlob(blob);
    html.window.open(url, "_blank");
    html.Url.revokeObjectUrl(url);

  }
```

Figura 39. Fragmento pdf_api.dart

6.1.9.2 PdfBudgetGenerator

Se encuentra en el fichero pdf_budget_generator.dart. Tiene como método principal generate (fragmento de su código en la Figura 40), al que se le pasará un presupuesto como parámetro y generará su PDF. Para ello llamará a varios métodos de la misma clase y cada uno se encargará de crear pequeñas partes del mismo. También contiene otros métodos auxiliares para crear los mismos.

Los métodos principales son los siguientes:

- buildHeader1: Creará la parte en la que se muestran los datos tanto del cliente como los del usuario. (Figura 41)
- buildHeader2: Creará el texto en el que se muestran los datos del autónomo
- buildBudgetBody: Creará todo el cuerpo del presupuesto, es decir, la tabla donde se muestran todas las partidas.
- buildResume: Generará la parte final de la tabla, donde se hace la suma de todas las partidas
- buildFooter: Creará el comentario final, donde se mostrará cómo se realizará el pago

```
static Future<File> generate(Budget budget) async {
  var myTheme = ThemeData.withFont(
    base: Font.ttf(await rootBundle.load("assets/OpenSans-Regular.ttf")),
    bold: Font.ttf(await rootBundle.load("assets/OpenSans-Bold.ttf")),
    italic: Font.ttf(await rootBundle.load("assets/OpenSans-Italic.ttf")),
    boldItalic:
    Font.ttf(await rootBundle.load("assets/OpenSans-BoldItalic.ttf")),
  );  // ThemeData.withFont

  final pdf = Document(
    theme: myTheme,
  );

  var url = budget.user.imageUrl;
  var a = Uri.parse(url);
  var client = http.Client();
  http.Response response = await client.get(a,);
  Uint8List bytes = response.bodyBytes;
  final image = pw.MemoryImage(bytes);

  pdf.addPage(MultiPage(
    build: (context) => [
      buildHeader1(budget, image),
```

Figura 40. Fragmento método generate de budget_pdf_generator

```dart
static Widget buildHeader1(Budget budget, final image) {
  return Container(
    color: PdfColors.grey400,
    child: Row(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.spaceAround,
        mainAxisSize: MainAxisSize.max,
        children: [
          Column(
              crossAxisAlignment: CrossAxisAlignment.end,
              children: [
                SizedBox(height: 0.2 * PdfPageFormat.cm),
                Image(image, width: 190, height: 90, fit: pw.BoxFit.fitWidth,),
                SizedBox(height: 0.3 * PdfPageFormat.cm),
                buildSimpleText(value: budget.user.phone, textStyle: TextStyle(fontSize: 9)),
                buildSimpleText(value: budget.user.address, textStyle: TextStyle(fontSize: 9)),
                SizedBox(height: 0.3 * PdfPageFormat.cm),
              ]
          ),  // Column
          Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                SizedBox(height: 0.3 * PdfPageFormat.cm),
                buildSimpleText(value: 'CLIENTE', textStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 11)),
                buildSimpleText(value: budget.client.name, textStyle: TextStyle(fontSize: 11)),
                SizedBox(height: 0.3 * PdfPageFormat.cm),
                buildSimpleText(value: 'DIRECCIÓN', textStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 11)),
                buildSimpleText(value: budget.client.address, textStyle: TextStyle(fontSize: 11)),
                SizedBox(height: 0.3 * PdfPageFormat.cm),
                buildSimpleText(value: 'POBLACIÓN', textStyle: TextStyle(fontWeight: FontWeight.bold, fontSize: 11)),
                buildSimpleText(value: budget.client.town, textStyle: TextStyle(fontSize: 11)),
                SizedBox(height: 0.3 * PdfPageFormat.cm),

              ]
```

Figura 41. Fragmento método buildHeader1 de budget:_pdf_generator

6.1.9.3 ResponsiveLayout

Se encuentra en el fichero responsive_layout.dart. Tiene como variables finales mobileBody y desktopBody, ambas de tipo Widget. Cuando se le llama pasándole ambas variables como parámetro, si el ancho de pantalla es inferior a 1000 devolverá el cuerpo de móvil, si es superior devolverá el de ordenador. En la Figura 42 se muestra todo el código de la misma.

```dart
import 'package:flutter/material.dart';

class ResponsiveLayout extends StatelessWidget {
  final Widget mobileBody;
  final Widget desktopBody;
  const ResponsiveLayout({super.key, required this.mobileBody, required this.desktopBody});

  @override
  Widget build(BuildContext context) {
    return LayoutBuilder(builder: (context, constraints){
      if(constraints.maxWidth < 1000){
        return mobileBody;
      } else {
        return desktopBody;
      }
    });
  }
}
```

Figura 42. Implementación de responsive_layout.dart

6.2 Firestore

El primer paso para asociar el proyecto es crear una cuenta de Google e ingresar en https://console.firebase.google.com/. Una vez dentro de la consola, habrá que seleccionar 'Añadir proyecto' y asignarle un nombre. Para agregar Firebase a la app se ha seleccionado la opción Flutter y se han seguido las instrucciones indicadas:

1. Instalar Firebase cli con el siguiente comando
2. curl -sL https://firebase.tools | bash
3. Iniciar sesión
4. firebase login
5. Ejecutar lo siguiente
6. dart pub global activate flutterfire_cli
7. En la raíz del proyecto ejecutar: flutterfire configure --project=facturas-y

Esto creará el archivo firebase_options.dart, donde se recogerá toda la configuración.

6.3 Firebase storage

Se crea un bucket desde el panel de navegación de la consola de Firebase y se selecciona su ubicación. Para comenzar se definen las reglas para permitirle a la aplicación el acceso a las imágenes que se guardarán aquí. Después será necesario agregar la URL del bucket en el fichero de configuración de Firebase.

6.4 Despliegue

6.4.1 Firebase hosting

Para configurar el hosting de la aplicación se han seguido las siguientes instrucciones:

1. Iniciar Firebase desde el directorio raíz del proyecto con el comando firebase init
2. Seleccionar la opción de configuración del hosting (Hosting: Configure files for Firebase Hosting and (optionally) set up GitHub Action deploys)
3. Completar la configuración respondiendo a las preguntas que se muestran en pantalla

Para finalizar, se genera la aplicación web y se sube de la siguiente manera (directorio raíz):

```
flutter build web --no-sound-null-safety

firebase deploy --only hosting
```

6.4.2 Generación de la apk

El archivo .apk se genera mediante el siguiente comando:

```
flutter --no-color build apk --no-sound-null-safety
```

6.4.3 Modificación del icono de la app y favicon

La imagen del logo está almacenada en el directorio 'assets' con el nombre 'Icon.png'. Se ha añadido la dependencia flutter_launcher_icons con la version "0.10.0" dentro del fichero pubspec.yaml. Después se ha añadido el icono al fichero pubspec.yaml (Figura 54) y se ha ejecutado el siguiente comando.

```
flutter pub run flutter_launcher_icons:main
```

6.5 Manual de usuario

Se ha hecho un manual en el que se muestran las funcionalidades de la aplicación para que los usuarios tengan una experiencia más sencilla. Está dividido en cuatro apartados principales: descarga, funciones básicas, creación de documento, gestión de datos. Este puede encontrarse en el Anexo F: Manual de usuario.

7

Pruebas

Las pruebas se han dividido en tres apartados, por un lado, las pruebas de funcionalidades, donde se analizarán todos los casos de uso. Por otro lado, las pruebas de adaptación, donde se comprobará que la aplicación sea adaptable a cualquier pantalla. Por último, en las pruebas con usuarios se recogerán opiniones y mejoras que han mencionado los usuarios y las soluciones que se les han dado.

7.1 Pruebas de funcionalidades

Las pruebas se han realizado usando la técnica de la caja negra, es decir, verificando la funcionalidad del software sin tener en cuenta la implementación o estructura interna del código. Se tienen en cuenta las entradas y salidas del sistema en diferentes casos. En este apartado solo se muestra las pruebas de 'Crear partida', el resto se encuentran en el Anexo D: Pruebas de funcionalidades.

Pruebas Crear partida

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Concepto' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'El concepto está vacío'.

Salida real: Igual a salida esperada.

Caso 2. El usuario introduce un precio con formato inadecuado.

Entrada: Precio con formato inadecuado. '2,2,2'.

Salida esperada: El sistema no permite escribir la segunda coma, escribe '2,22'.

Salida real: Igual a salida esperada.

Caso 3. El usuario introduce un precio con caracteres inadecuados.

Entrada: Precio con formato inadecuado. 'a2bc'.

Salida esperada: El sistema no permite ningún carácter no válido, escribe solo '2'.

Salida real: Igual a salida esperada.

Caso 4. El usuario introduce todos los campos correctamente.

Entrada: Campo de concepto completo y precio con formato adecuado. Concepto: 'Prueba', precio: '2,5'.

Salida esperada: Se muestra el mensaje 'Concepto guardado' en el centro de la pantalla para indicarle al usuario que se ha completado la tarea con éxito.

Salida real: Igual a salida esperada.

7.2 Pruebas de adaptación

Durante el desarrollo se han hecho pruebas en todo tipo de dispositivos con diferentes tamaños de pantalla. En las siguientes imágenes se muestran cómo se adapta la página de inicio de sesión y la de añadir partidas al documento en una pantalla de móvil y a una de ordenador con monitor grande. El resto se pueden ver en Anexo E: Pruebas de adaptación

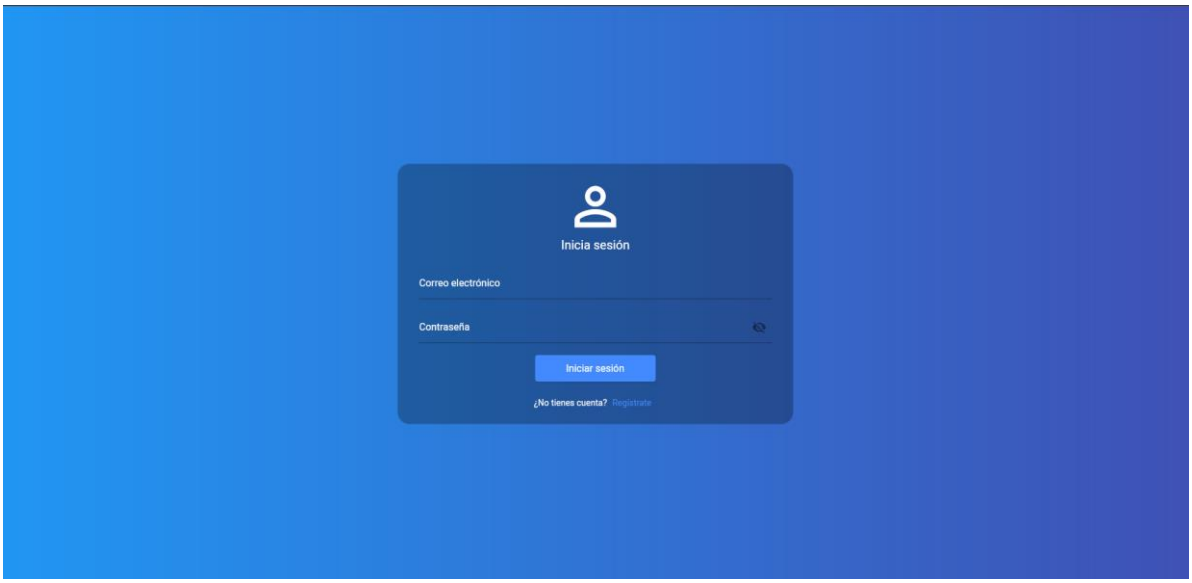


Figura 43. Interfaz ordenador: inicio de sesión

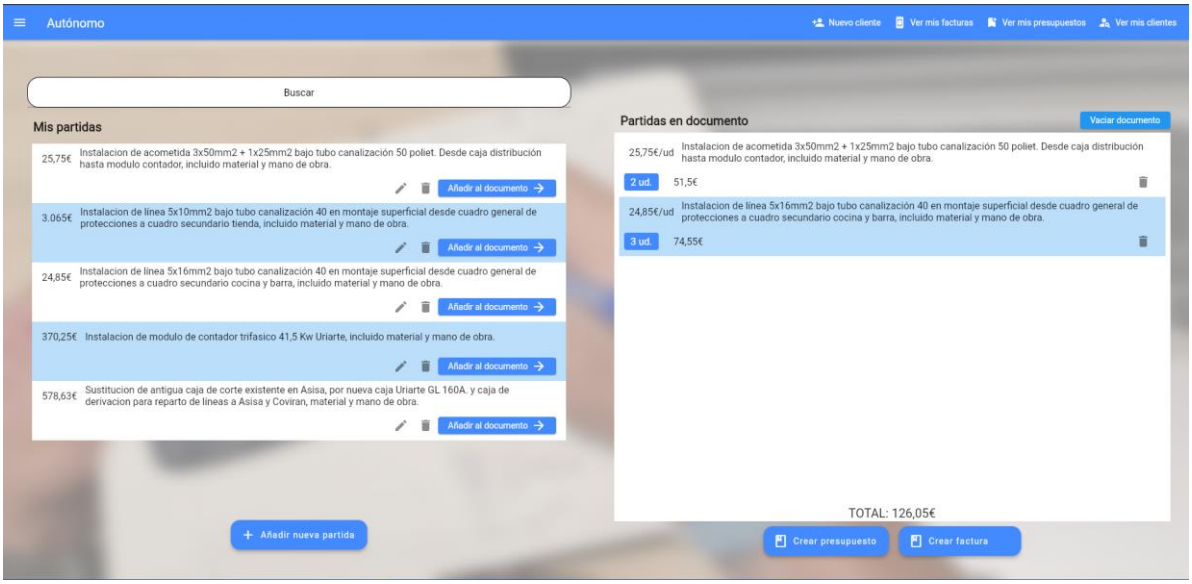


Figura 44. Interfaz ordenador: agregar partidas a documento

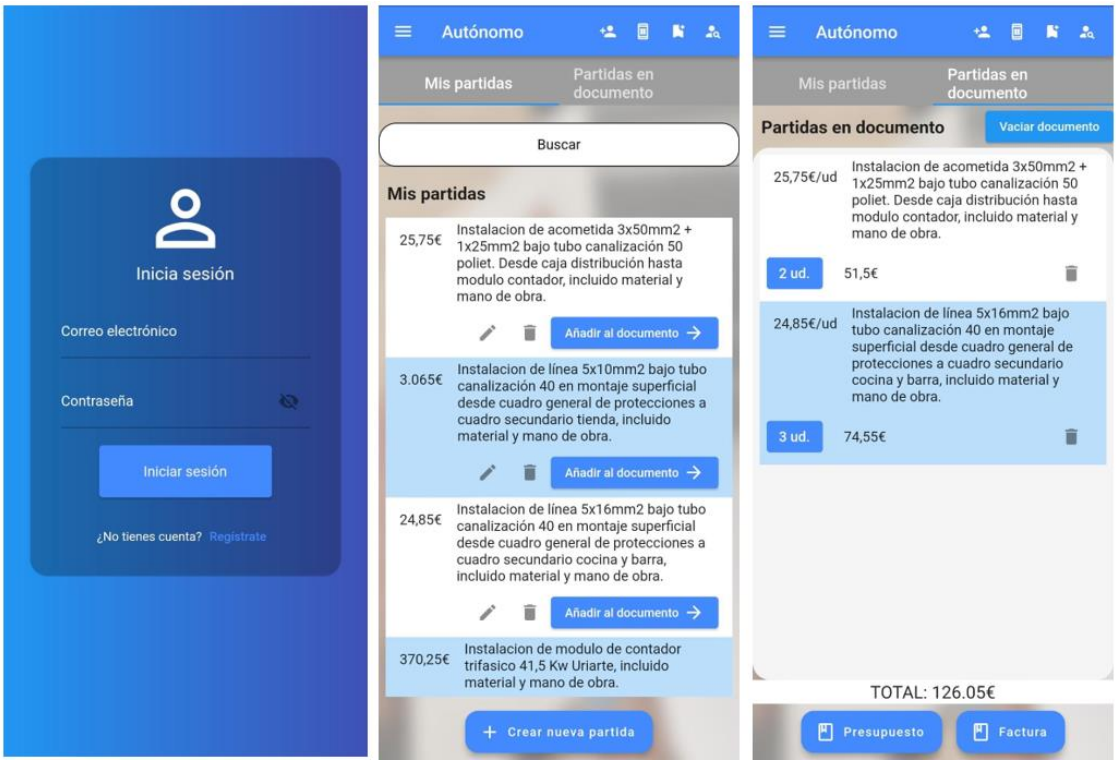


Figura 45. Interfaz inicio de sesión y añadir partidas a documento

7.3 Pruebas con usuarios

Se han hecho pruebas con usuarios reales cada vez que se introducía una nueva funcionalidad, tanto como para ver la manera en la que utiliza la aplicación un usuario que no la conoce, como para ver pequeños cambios que se podían hacer para que la aplicación fuera más sencilla de utilizar o ver fallos que no se habían visto antes.

La mayoría de las mejoras que han planteado los usuarios ya se encuentran introducidas en la aplicación. Estas son algunas de ellas:

- Contraseña visible al iniciar sesión.
- Todas las funcionalidades accesibles desde cualquier lugar.
- Posibilidad de cambiar entre campos de texto pulsando el tabulador y que también responda al ENTER para, por ejemplo, pulsar el botón al completar un formulario.
- Introducir número de factura de manera manual.
- Formato del precio sea el utilizado en España (1.230,50€) y no deje introducir ningún otro para no dejar que el usuario se equivoque.
- Más intuitivo al añadir partida al documento, no quedaba claro cómo se añadía una partida al documento, solo se mostraba una flecha. Se ha cambiado el formato del botón y se ha añadido el mensaje 'Añadir al documento'.
- Confusiones con los botones 'Crear presupuesto' y 'Crear factura'. El usuario los intentaba pulsar antes de añadir partidas al documento, pasaba a la interfaz de introducción de información sin partidas en el documento. Ahora estos botones se muestran en gris y estarán bloqueados si el documento está vacío.
- Posibilidad de vaciar el documento sin pulsar cada partida de una en una. Se ha añadido un botón para ello.

- Posibilidad de buscar partidas, clientes, etc. teniendo en cuenta más campos.
- Cambio de orden en el que se recogen los datos al crear el documento.
- Algunos elementos no se veían muy claros en el móvil.
- La sesión se debería de mantener iniciada cuando cierra la aplicación.
- Últimas partidas se quedan guardadas en la aplicación para poder continuarlo más tarde.

Sin embargo, estas otras propuestas se han dejado como mejoras que se introducirán en un futuro de la aplicación:

- Le gustaría poder registrarse sin tener que añadir una imagen de manera obligatoria.
- Quiere que las notificaciones emergentes que se muestran abajo puedan cerrarse pulsando un botón en lugar de esperar a que se cierren solas.
- Quiere que haya más plantillas para poder crear PDFs con diferentes formatos.

Tras finalizar la aplicación se han medido los tiempos que necesitaba cada usuario para hacer determinadas tareas. En la siguiente tabla (Figura 46) se muestra el resultado de las pruebas, los usuarios 1 y 2 son personas que utilizan de manera habitual el ordenador, los usuarios 3 y 4 son personas que utilizarán la aplicación. Se les ha pedido que utilicen la aplicación como la utilizarían realmente, pero algunos de los datos que se han introducido al hacer las pruebas han sido inventados, como por ejemplo el número de banco, lo que podría hacer que el tiempo de registro real sea algo mayor.

TAREA	USUARIO	TIEMPO USUARIO (s)	TIEMPO MEDIO (s)	TIEMPO ESTIMADO(s)	DIFERENCIA (s)
Registrarse	Usuario 1	58,7	126,65	120	6,65
	Usuario 2	82,2			
	Usuario 3	246,3			
	Usuario 4	216,5			
Iniciar sesión	Usuario 1	7,2	11,71	15	-3,29
	Usuario 2	9,6			
	Usuario 3	14,4			
	Usuario 4	18,9			
Crear partida	Usuario 1	9,26	23,64	30	-6,36
	Usuario 2	16,3			
	Usuario 3	31,3			
	Usuario 4	66,1			
Crear cliente	Usuario 1	6,8	13,40	15	-1,60
	Usuario 2	12			
	Usuario 3	15,3			
	Usuario 4	25,8			
Modificar partida	Usuario 1	4,9	7,15	10	-2,85
	Usuario 2	5,7			
	Usuario 3	10,3			
	Usuario 4	9,1			
Crear lista de partidas para documento (partidas ya añadidas)	Usuario 1	34,1	54,53	60	-5,47
	Usuario 2	56,9			
	Usuario 3	64,8			
	Usuario 4	70,3			
Crear PDF con factura (con la lista de partidas ya hecha)	Usuario 1	59,7	64,76	60	4,76
	Usuario 2	45,2			
	Usuario 3	62,9			
	Usuario 4	103,6			
Ver facturas	Usuario 1	1,3	2,37	3	-0,63
	Usuario 2	1,1			
	Usuario 3	4,9			
	Usuario 4	4,5			

Figura 46. Tiempos obtenidos tras realizar las pruebas con usuarios

8

Seguimiento y control

En este capítulo se describe todo lo relacionado con el seguimiento y control del proyecto. Se mostrarán las incidencias ocurridas durante el desarrollo del mismo y cómo estas han afectado. Se analizarán los cambios principales que ha sufrido la planificación inicial tanto en relación a las fechas como a las horas que han sido necesarias invertir para completar el proyecto.

8.1 Incidencias durante el desarrollo del proyecto

En esta sección se explica qué riesgos de los definidos en la planificación inicial han ocurrido y cómo se han abordado, además de otros no planificados.

A lo largo del desarrollo del proyecto han ocurrido importantes incidencias que han afectado tanto a las fechas como a las horas dedicadas a cada tarea.

8.1.1 Problemas con las tecnologías

En un principio la idea fue realizar el trabajo haciendo uso de Apache Cordova, una tecnología que no conocía, pero en la que se utilizan lenguajes que no requerirían excesivo tiempo de desarrollo, ya que ya tenía conocimientos y cualquier error no supondría demasiado gasto de tiempo. El tiempo de estudio de Flutter no ha requerido más tiempo del planificado, pero al utilizar un lenguaje nuevo (2017) con relativamente poca información en internet y al no tener experiencia con la herramienta, cualquier complicación ha aumentado el tiempo de desarrollo.

Por esto, los mayores cambios en cuanto a tiempo se encuentran en el desarrollo, en el resto de tareas el cálculo ha sido acertado.

8.1.2 Trabajo a tiempo completo

La planificación del proyecto se realizó a principios de abril, teniendo en cuenta que me quedaban tres asignaturas por aprobar, pero lo que no se tuvo en cuenta fue que a comienzos del siguiente mes me llamaría la empresa en la que hice las prácticas para que me incorporase a mediados de mayo a un trabajo a tiempo completo. A pesar de dedicarle todo el tiempo libre que me quedaba por las tardes, noches y fines de semana, llevar el ritmo planificado fue imposible por lo que se explica en el siguiente apartado.

8.1.3 Problemas de salud

Probablemente debido al alto estrés que supuso estudiar, hacer el TFG y trabajar, el día 26 de junio sufrí una grave taquicardia, lo que conllevó una visita de urgencias al hospital. Gracias a esto se me detectó un problema congénito en el corazón que requería intervención, finalmente me operaron el 19 de septiembre.

Debido a esto, por recomendación médica, fue necesario bajar el ritmo de trabajo y llegar a pausarlo por completo las semanas que ocurrieron estos acontecimientos.

Tras la primera taquicardia se aplazó la idea de entregar el TFG de septiembre a octubre.

8.2 Diagrama de Gantt

Debido a las incidencias comentadas en el apartado anterior, ha sido necesario modificar la gran mayoría de las fechas planificadas para cada tarea, por lo que el diagrama de Gantt final es muy diferente al planificado. Por los problemas de salud, ha sido necesario parar por completo el proyecto en varias ocasiones, además de reducir las horas de trabajo diarias que se le hubieran dedicado al TFG en otra situación. Estos parones se muestran en rojo.

En la siguiente tabla se muestra el diagrama de Gantt final (Figura 47).

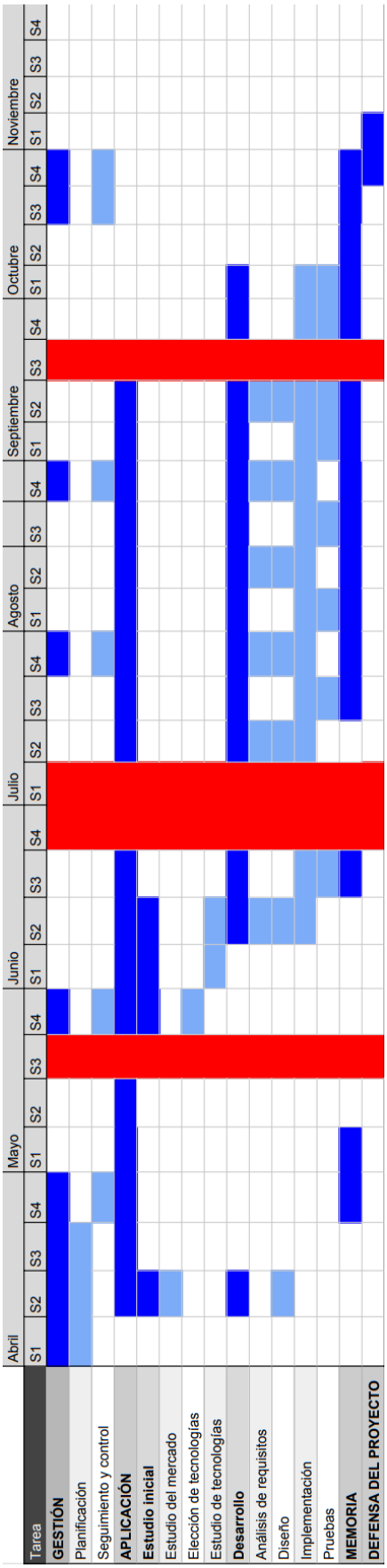


Figura 47. Diagrama de Gantt final

8.3 Comparativa de dedicaciones

Al planificar el proyecto se hizo una estimación del tiempo que requeriría cada tarea.

Esta no es una tarea fácil, ya que dependerá de muchos factores, por lo que el tiempo planificado y el real han sufrido variaciones.

El tiempo total invertido ha sido aproximadamente de 355 horas, 55 horas más que las planificadas. A continuación, se muestra una tabla con el mismo formato que la que se muestra en la planificación inicial, con las horas estimadas iniciales y las reales. Las horas necesarias para la defensa del proyecto se mantienen igual que las planificadas porque la memoria se entregará antes de realizarlo.

Tarea		
GESTIÓN	Horas planificadas	Horas reales
Planificación	5	8
Seguimiento y control	5	2
APLICACIÓN	-	-
Estudio inicial	-	-
Estudio del mercado	2	1
Elección de tecnologías	2	2
Estudio de tecnologías	20	30
Desarrollo	-	-
Análisis de requisitos	12	13
Diseño	10	10
Implementación	150	190
Pruebas	10	10
MEMORIA	70	75
DEFENSA DEL PROYECTO	14	14
TOTAL	300	355

Figura 48. Tabla con estimación de dedicaciones final

9

Conclusiones

En este capítulo se presentan las conclusiones tras finalizar el proyecto. Se mencionan los conocimientos adquiridos durante el desarrollo del mismo, objetivos logrados y el futuro de 'Autónomo'.

9.1 Conocimientos adquiridos

9.1.1 Experiencia personal

Durante la carrera me hubiera gustado haber cursado alguna asignatura relacionada con el desarrollo de aplicaciones móviles y este proyecto me ha dado la oportunidad de hacerlo, utilizando nuevas tecnologías que desconocía y me han resultado muy interesantes.

No es un proyecto tan ambicioso como podría ser el de un software con años de trayectoria, pero requiere mucha dedicación y tiempo, mucha más de la que podría requerir cualquier trabajo hecho durante la carrera. Además de aprender muchos nuevos conceptos y tecnologías he podido comprobar lo importante que es tener un código limpio y hacer una buena gestión del proyecto, teniendo todo muy bien planificado desde el comienzo del trabajo.

Otra cosa que destacaría de lo que he aprendido es el papel tan importante que tiene el cliente desde el comienzo de un proyecto de estas características. He tenido la oportunidad de hacer pruebas que me han aportado un nuevo punto de vista tras finalizar cada iteración. Gracias a esta forma de verlo, he podido mejorar el producto final añadiendo mejoras que tal vez podían haber pasado por alto en otra situación.

He adquirido nuevos conocimientos y en general, creo que ha sido una muy buena experiencia y una buena preparación para incorporarme al mundo laboral. Me he quedado contenta con el resultado obtenido.

9.1.2 Desarrollo y tecnologías

Nunca antes había hecho un trabajo tan completo y considero que ha sido un acierto el haber dividido el trabajo en diferentes iteraciones, pudiendo así enseñárselas al cliente.

A pesar de haber necesitado más tiempo del que hubiera necesitado desarrollando la aplicación con una tecnología conocida, hacer toda la aplicación usando Flutter ha sido una experiencia muy enriquecedora. Es una herramienta muy útil que permite crear aplicaciones para la mayoría de plataformas de una manera muy rápida.

El único inconveniente que le he visto a hacer el desarrollo utilizando un lenguaje tan nuevo ha sido que a la hora de buscar soluciones a errores o hacer cosas nuevas había mucha menos información que en otros lenguajes más famosos. Además, está en constante evolución por lo que muchas soluciones que se pueden encontrar en internet ya no resultan útiles.

9.2 Objetivos logrados

El objetivo principal de la aplicación era hacer una versión más reducida desde el punto de vista de funcionalidades que ofrecen otros softwares existentes, tal y como se ha explicado en el capítulo 2.3, consiguiendo así una aplicación muy sencilla e intuitiva de utilizar.

Considero que se han superado con éxito todos los objetivos, haciendo una aplicación multiplataforma más completa de lo planificado inicialmente. Además, las principales personas que han probado la aplicación se han quedado muy satisfechas.

9.3 Futuro de la aplicación

9.3.1 Futuras mejoras

Se han añadido todas las funcionalidades pensadas en un inicio y aunque el proyecto del TFG haya finalizado, mi intención es continuar mejorándolo y añadiendo alguna nueva característica que tengo en mente que no se ha hecho por falta de tiempo, siempre manteniendo la sencillez que lo caracteriza.

Algunas de las mejoras que se pretenden añadir son las siguientes: posibilidad de seleccionar entre diferentes plantillas la hora de generar el PDF, la autenticación con Google, posibilidad de enviar directamente el correo al cliente o la verificación del correo.

Últimamente se está anunciando mucho un nuevo sistema de facturación llamado TicketBAI, que aún no se aplica a las personas a las que está destinada esta aplicación, pero probablemente será obligatoria en un periodo de tiempo no muy largo. Los próximos meses intentaré registrar la aplicación para que sea posible utilizarla en cuanto esta normativa entre en vigor.

9.3.2 Qué se espera

Espero que pueda ser una aplicación que sirva de ayuda y llene el vacío existente en los programas para los trabajadores autónomos sin conocimientos informáticos ni contables.

Además de mi padre, que es el primer interesado en la aplicación, me consta que hay mucha gente esperando el programa y en colaboración con ellos seguirá creciendo hasta que llegue a ser una herramienta realmente útil.

Aún no se está utilizando, pero la aplicación ya se puede lanzar y dentro de poco empezarán a usarla las personas anteriores. De momento, los usuarios de Apple tendrán que utilizar la versión web puesto que solo es posible generar su app propia en un ordenador Mac del que no dispongo.

En cuanto a la base de datos gratuita, en principio no se espera que tenga tantos usuarios como para necesitar la versión de pago, pero en el caso de que esto ocurra se plantearía la idea de obtenerla.

Bibliografía

[1] Apache Cordova, [En línea]. Available: https://cordova.apache.org/.

[2] V. Mazin, «Flutter VS Apache Cordova: Choosing the Best Technology for Cross-Platform Development,» [En línea]. Available: https://surf.dev/flutter-vs-apache-cordova/.

[3] ChiaTK, «YouTube,» [En línea]. Available: https://www.youtube.com/playlist?list=PLl_hIu4u7P677H9f6zPOHiOz2izkvQq2E.

[4] H. Kachmar, «Udemy,» [En línea]. Available: https://www.udemy.com/course/flutter-with-firebase-build-an-e-commerce-app-from-scratch/.

[5] sqflite, «sqflite,» [En línea]. Available: https://pub.dev/packages/sqflite.

[6] Firebase, [En línea]. Available: https://firebase.google.com/?hl=es.

[7] Hive, [En línea]. Available: https://hive.com/.

[8] Firebase, «Firebase Realtime Database,» [En línea]. Available: https://firebase.google.com/docs/database?hl=es-419.

[9] Firebase, «Cloud FIrestore,» [En línea]. Available: https://firebase.google.com/docs/firestore.

[10] Flutter, «Flutter,» [En línea]. Available: https://flutter.dev/. [Último acceso: 2022].

[11] Dart, «Dart,» [En línea]. Available: https://dart.dev/.

[12] SKIA, «SKIA,» [En línea]. Available: https://skia.org/.

[13] Firebase, «Firebase,» [En línea]. Available: firebase.google.com.

[14] Firebase, «Firebase pricing,» [En línea]. Available: https://firebase.google.com/pricing.

[15] GitHub, «GitHub,» [En línea]. Available: https://github.com/.

[16] Android studio, [En línea]. Available: https://developer.android.com/.

[17] Jimdo, «Cómo elegir los colores para tu página web,» [En línea]. Available: https://www.jimdo.com/es/blog/colores-paginas-web/.

[18] GitHub, «Repositorio GitHub personal (Yakodelaragua),» [En línea]. Available: https://github.com/yakodelaragua/Factura.

[19] Flutter, «Flutter Get Started,» [En línea]. Available: https://docs.flutter.dev/get-started/install.

[20] pub.dev, [En línea]. Available: https://pub.dev/packages.

[21] Y. Diaz de Cerio, «Página web Autónomo,» [En línea]. Available: facturas-y.web.app.

[22] Y. Diaz de Cerio, «Descarga APK Autónomo,» [En línea]. Available: https://drive.google.com/file/d/1M-NUbTsTOl20ToNVKtm9OJ-3YrWqkBQw/view.

Anexo A: Flujo de eventos

Se muestra el flujo de eventos de todos los casos de uso, divididos en diferentes sprints al igual que los casos de uso. Por cada uno se proporcionará la siguiente información:

- Descripción: Describe el objetivo del caso de uso.
- Actor: Rol del usuario.
- Precondiciones: Estado en el que debe de estar necesariamente para que se pueda ejecutar el caso de uso.
- Flujo normal: Comportamiento principal.
- Flujo alternativo: Describe las desviaciones o excepciones del comportamiento principal.

A.1 Sprint 1

A.1.1 Iniciar sesión

Descripción: Permite al usuario iniciar sesión para acceder a todas las funcionalidades de la aplicación.

Actor: Usuario sin loguear.

Precondiciones: El usuario no puede tener la sesión iniciada.

Flujo normal:

1. El sistema comprueba si la sesión del usuario está activa, si no lo está muestra un formulario donde se deberán introducir todos los datos.
2. El usuario indica su correo y contraseña y pulsa el botón 'Iniciar sesión'.
3. El sistema comprueba que los datos sean válidos
4. El sistema comprueba que el usuario existe en 'Firebase Authentication', inicia sesión y obtiene sus datos.
5. El sistema guarda el usuario para que la próxima vez que abra la aplicación su sesión esté activa y redirige al usuario a la página de selección.

Flujo alternativo

1. El sistema comprueba si la sesión del usuario está activa, si ya lo estaba inicia sesión con 'Firebase Authentication' y redirige el usuario a la página de selección

Flujo alternativo

3. El sistema comprueba que los datos sean válidos, si no lo son muestra un mensaje de error.

A.1.2 Registrar

Descripción: Permite al usuario crear una cuenta con sus datos para acceder a todas las funcionalidades de la aplicación.

Actor: Usuario sin loguear.

Precondiciones: El usuario no puede tener la sesión iniciada.

Flujo normal:

1. El usuario pulsa el botón 'Regístrate' y se le redirige a la página de registro.

2. El sistema muestra un formulario con todos los datos necesarios
3. El usuario completa todos los campos del formulario con sus datos: logo, correo electrónico, contraseña, móvil, dirección DNI, cuenta de banco a ingresar, número de trabajador y número de empresa. Pulsa en botón 'registrar'.
4. El sistema comprueba que todos los datos estén completos y sean válidos.
5. El sistema comprueba que el usuario no existe en 'Firebase Authentication', crea una instancia con el correo y la contraseña cifrada y añade los datos asociados al correo en 'Firestore Database'.
6. El usuario recibe un aviso de para avisarle de que la cuenta ha sido creada correctamente y se le redirige a la página de inicio de sesión.

Flujo alternativo

4. Si los datos no están completos o no son válidos el sistema señalará en rojo el campo y mostrará el error.

Flujo alternativo

5. Si el usuario no existe en 'Firebase Authentication' el sistema muestra un mensaje de error.

A.1.3 Modificar datos personales

Descripción: Permite al usuario modificar sus datos personales.

Actor: Usuario logueado

Precondiciones: El usuario debe de tener la sesión iniciada.

Flujo normal:

1. El usuario abre el menú lateral superior izquierdo y pulsa el botón 'Modificar mi usuario'.
2. El sistema lo redirige a la página de modificación de datos, donde se mostrarán todos sus datos actuales.
3. El usuario modifica el dato que desea en el formulario y pulsa el botón 'Actualizar'
4. El sistema comprueba que los datos introducidos sean correctos y los actualiza en la base de datos.
5. El sistema muestra un mensaje para notificar al usuario de que se ha realizado la operación y lo reenvía a la página en la que se encontraba.

Flujo alternativo

4. El sistema comprueba que los datos introducidos sean correctos, si no lo son muestra el mensaje de error.

A.2 Sprint 2

A.2.1 Ver partidas

Descripción: Permite al usuario ver una lista con todas las partidas.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y se encuentra en la vista de crear documentos.

Flujo normal:

1. El usuario pulsa el botón ‘Crear documento’.
2. El sistema obtiene todas las partidas de la base de datos y las muestra en una lista.

Flujo alternativo

2. Si el usuario aún no ha añadido ninguna partida se le indicará mediante un mensaje.

A.2.2 Crear partida

Descripción: Permite al usuario crear una partida.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y debe de estar en la vista de crear un documento.

Flujo normal:

1. El usuario pulsa el botón ‘Añadir nueva partida’.
2. El sistema muestra una ventana con los campos de texto ‘Concepto’ y ‘Precio’.
3. El usuario escribe los datos de la partida que quiere añadir y pulsa ‘Guardar’.
4. El sistema añade la partida a la base de datos.

Flujo alternativo

3. Si el usuario deja algún campo vacío el sistema le mostrará un error con el mensaje ‘El concepto está vacío’ y no guardará la partida.

A.2.3 Editar partida

Descripción: Permite al usuario modificar la partida.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y debe de estar en la vista de crear un documento.

Flujo normal:

1. El usuario pulsa el botón con un icono de un lápiz de la partida que quiere editar.
2. El sistema muestra una ventana con los datos de la partida actuales.
3. El usuario modifica la partida y pulsa ‘Actualizar’.
4. El sistema actualiza la partida en la base de datos.

Flujo alternativo

3. Si el usuario deja algún campo vacío el sistema le mostrará un error con el mensaje ‘El concepto está vacío’ y no actualizará la partida.

A.2.4 Eliminar partida

Descripción: Permite al usuario eliminar la partida.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y debe de estar en la vista de crear un documento.

Flujo normal:

1. El usuario pulsa el botón con un icono de una papelera de la partida que quiere eliminar.

2. El sistema muestra una ventana con el mensaje '¿Seguro que quieres eliminar la partida?'.
3. El usuario pulsa 'Si'.
4. El sistema elimina la partida de la base de datos.

Flujo alternativo

3. El usuario pulsa 'No'.
4. El sistema cierra la ventana y no elimina la partida.

A.2.5 Buscar partidas

Descripción: Permite al usuario buscar una partida por su concepto.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de crear documentos.

Flujo normal:

1. El usuario escribe en la barra de búsqueda la parte del concepto que quiera buscar.
2. El sistema filtra la lista y muestra solo las que cumplen alguna condición.

A.2.6 Añadir partida al documento

Descripción: Permite al usuario agregar una partida de la lista con todas sus partidas a su documento, indicando las unidades que quiere añadir.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y está en la vista de crear documento

Flujo normal:

1. El usuario pulsa el botón 'Añadir al documento'.
2. Si la partida no está en la lista de partidas del documento el sistema muestra una ventana con un campo de texto para indicar la cantidad.
3. El usuario indica la cantidad y pulsa 'Guardar'.
4. Si se ha guardado correctamente el sistema muestra el mensaje de confirmación 'Añadido correctamente'.

Flujo alternativo

2. Si la partida ya estaba en el documento el sistema muestra el mensaje 'La partida ya está en el documento'.

A.2.7 Modificar cantidad de partida

Descripción: Permite al usuario modificar la cantidad de una partida ya añadida al documento.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y la partida cuya cantidad quiere modificar añadida al documento.

Flujo normal:

1. El usuario pulsa sobre el botón el que se indica la cantidad de la partida.
2. El sistema muestra una ventana con un campo de texto para introducir la cantidad.

3. El usuario indica la nueva cantidad y pulsa ‘Guardar’.
4. El sistema modifica la cantidad de la partida.

Flujo alternativo

3. Si el usuario deja el campo de la cantidad vacía no podrá guardarla.

A.2.8 Eliminar partida de documento

Descripción: Permite al usuario eliminar una partida del documento.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada, se encuentra en la vista de crear documento y tiene la partida que quiere eliminar añadida al documento.

Flujo normal:

1. El usuario pulsa el botón con un icono de una papelera de la partida que quiere eliminar.
2. El sistema muestra una ventana con el mensaje ‘¿Seguro que quieres eliminar la partida del documento?’.
3. El usuario pulsa ‘Si’.
4. El sistema elimina la partida.

Flujo alternativo

3. El usuario pulsa ‘No’.
4. La partida no se elimina del documento.

A.2.9 Ver partidas del documento

Descripción: Permite al usuario ver una lista con todas las partidas del documento.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y se encuentra en la vista de crear documentos.

Flujo normal:

1. El usuario pulsa el botón ‘Crear documento’.
2. El sistema obtiene todas las partidas del documento de la base de datos y las muestra en una lista.

Flujo alternativo

2. Si el usuario aún no ha añadido ninguna partida se le indicará mediante un mensaje.

A.3 Sprint 3

A.3.1 Logout

Descripción: Permite al usuario cerrar la sesión.
Actor: Usuario logueado.
Precondiciones: El usuario debe de tener la sesión iniciada.
Flujo normal:

1. El usuario abre el menú lateral superior izquierdo y pulsa el botón ‘Cerrar sesión’.
2. El sistema cierra la sesión de ‘Firebase database’ y elimina la instancia almacenada en el dispositivo del usuario.
3. El sistema redirige al usuario a la página de inicio de sesión.

A.3.2 Añadir datos presupuesto

Descripción: Una vez el usuario tiene las partidas añadidas al documento deberá añadir el número de presupuesto, fecha, comentario inicial, comentario final y seleccionar un cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y tiene partidas añadidas al documento.

Flujo normal:

1. El usuario se encuentra en la vista de crear documentos, con partidas añadidas y pulsa 'Crear presupuesto'.
2. El sistema muestra una nueva vista en la que el usuario podrá modificar sus datos o las partidas añadidas, crear clientes y añadir los datos al presupuesto.
3. El usuario introduce el número de presupuesto, fecha, comentario inicial y comentario final al documento. Selecciona un cliente de la lista desplegable y pulsa 'Crear presupuesto'.
4. Si los datos están completos el sistema guardará el presupuesto y sus datos.

Flujo alternativo:

4. Si algún dato está vacío el sistema lo resaltará e indicará en rojo el mensaje 'Campo obligatorio' bajo el campo de texto.

A.3.3 Añadir datos factura

Descripción: Una vez el usuario tiene las partidas añadidas al documento deberá añadir el número de factura, fecha, comentario inicial y seleccionar un cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada y tiene partidas añadidas al documento.

Flujo normal:

1. El usuario se encuentra en la vista de crear documentos, con partidas añadidas y pulsa 'Crear factura'.
2. El sistema muestra una nueva vista en la que el usuario podrá modificar sus datos o las partidas añadidas, crear clientes y añadir los datos a la factura.
3. El usuario introduce el número de presupuesto, fecha y comentario inicial al documento. Selecciona un cliente de la lista desplegable y pulsa 'Crear factura'.
4. Si los datos están completos el sistema guardará la factura y sus datos.

A.3.4 Crear presupuesto

Descripción: Permite al usuario crear un nuevo presupuesto.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada.

Flujo normal:

1. El usuario selecciona la opción 'Crear nuevo documento'.
2. Selecciona la partida y la cantidad de unidades que se añadirán al documento.
3. El sistema añade estas al documento actual que se guardará en la base de datos por si se cierra por error o por si el usuario quiere continuarla en otro momento.
4. El usuario selecciona entre las 'Crear presupuesto'.

5. El sistema lo redirige a la página en la que deberá introducir toda la información necesaria para crear un presupuesto como la fecha o los comentarios. También seleccionará el cliente y podrá modificar las partidas o sus datos.
6. El usuario pulsa el botón 'Crear presupuesto'.
7. El sistema añadirá el presupuesto a la base de datos y le abrirá el archivo PDF en una nueva ventana.

A.3.5 Crear factura

Descripción: Permite al usuario crear una nueva factura.

Actor: Usuario logueado.

Precondiciones: El usuario deberá de tener la sesión iniciada.

Flujo normal:

1. El usuario selecciona la opción 'Crear nuevo documento'.
2. Selecciona la partida y la cantidad de unidades que se añadirán al documento.
3. El sistema añade estas al documento actual que se guardará en la base de datos por si se cierra por error o por si el usuario quiere continuarla en otro momento.
4. El usuario selecciona entre las 'Crear factura'.
5. El sistema lo redirige a la página en la que deberá introducir toda la información necesaria para crear una factura como la fecha o el comentario. También seleccionará el cliente y podrá modificar las partidas o sus datos.
6. El usuario pulsa el botón 'Crear factura'.
7. El sistema añadirá la factura a la base de datos y abrirá el archivo PDF en una nueva ventana.

A.4 Sprint 4

A.4.1 Crear cliente

Descripción: Permite al usuario crear un cliente y añadirlo a la base de datos.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada.

Flujo normal:

1. El usuario pulsa el botón 'Nuevo cliente'.
2. El sistema muestra un menú con un formulario con los datos necesarios: Nombre, dirección, población, DNI y teléfono.
3. El usuario rellena todos los datos y pulsa el botón 'Guardar'.
4. El sistema comprueba que todos los datos sean correctos y añade el nuevo cliente a la base de datos.

Flujo alternativo:

4. El sistema comprueba que todos los datos estén completos, si no lo están lo indica mediante un mensaje.

A.4.2 Crear PDF presupuesto

Descripción: Crea el PDF del presupuesto.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada, partidas en el documento y todos los datos del presupuesto correctos.

Flujo normal:

1. El usuario pulsa el botón 'Crear presupuesto'.
2. El sistema guarda el presupuesto en la base de datos, genera un PDF y lo abre en una nueva pestaña.

A.4.3 Crear PDF factura

Descripción: Crea el PDF de la factura.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada, partidas en el documento y todos los datos de la factura correctos.

Flujo normal:

1. El usuario pulsa el botón 'Crear presupuesto'.
2. El sistema guarda la factura en la base de datos, genera un PDF y lo abre en una nueva pestaña.

A.5 Sprint 5

A.5.1 Ver facturas

Descripción: Permite al usuario ver una lista con todas sus facturas

Actor: Usuario logueado

Precondiciones: El usuario tiene la sesión iniciada

Flujo normal:

1. El usuario pulsa el botón 'Gestionar facturas'.
2. El sistema obtiene de la base de datos todas las facturas que corresponden al usuario y las muestra en una lista.

Flujo alternativo

2. Si el usuario aún no ha creado ninguna factura se le indicará mediante un mensaje.

A.5.2 Eliminar factura

Descripción: Permite al usuario eliminar una factura de la base de datos.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de gestionar facturas.

Flujo normal:

1. El usuario pulsa el botón de la papelera en la factura que quiere eliminar.
2. El sistema mostrará una nueva ventana con el mensaje '¿Seguro que quieres eliminar la factura?' en el que el usuario podrá decidir si continuar con la eliminación.
3. El usuario pulsa 'Si'.
4. El sistema eliminará la factura de la base de datos y dejará de mostrarse en la lista.

Flujo alternativo

3. El usuario pulsa ‘No’.

4. Se cerrará la ventana y no se eliminará la factura de la lista.

A.5.3 Modificar factura

Descripción: Permite al usuario modificar una factura.
Actor: Usuario logueado.
Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de gestionar facturas.
Flujo normal:

1. El usuario selecciona una factura de la lista.
2. El sistema obtiene todos los datos de la factura seleccionada y envía al usuario a la vista en la que se crean los documentos, para que así pueda modificar los datos necesarios y pueda volver a crear su documento.

A.5.4 Buscar factura

Descripción: Permite al usuario buscar una factura según varios parámetros como descripción, fecha o cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de gestionar facturas.

Flujo normal:

1. El usuario escribe en el buscador el parámetro con el que desea realizar la búsqueda.
2. El sistema filtra la lista general y muestra solo los que cumplen las condiciones.

Flujo alternativo

2. Si ninguna factura cumple las condiciones se mostrará la lista vacía.

A.6 Sprint 6

A.6.1 Ver presupuestos

Descripción: Permite al usuario ver una lista con todos sus presupuestos.
Actor: Usuario logueado.
Precondiciones: El usuario tiene la sesión iniciada.
Flujo normal:

1. El usuario pulsa el botón ‘Gestionar presupuestos’.
2. El sistema obtiene todos los presupuestos del usuario de la base de datos y los muestra en una lista.

Flujo alternativo

2. Si el usuario aún no ha creado ningún presupuesto se le indicará mediante un mensaje.

A.6.2 Eliminar presupuestos

Descripción: Permite al usuario eliminar un presupuesto de la base de datos.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de ver presupuestos.

Flujo normal:

1. El usuario pulsa el botón de la papelera en el presupuesto que quiere eliminar.

2. El sistema mostrará una nueva ventana con el mensaje '¿Seguro que quieres eliminar el presupuesto?' en el que el usuario podrá decidir si continuar con la eliminación.
3. El usuario pulsa 'Si'.
4. El sistema eliminará el presupuesto de la base de datos y dejará de mostrarse en la lista.

Flujo alternativo

3. El usuario pulsa 'No'.
4. Se cerrará la ventana y no se eliminará el presupuesto de la lista.

A.6.3 Modificar presupuesto

Descripción: Permite al usuario modificar un presupuesto.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de ver presupuestos.

Flujo normal:

1. El usuario selecciona un presupuesto de la lista.
2. El sistema obtiene todos los datos del presupuesto seleccionado y envía al usuario a la vista en la que se crean los documentos, para que así pueda modificar los datos necesarios y pueda volver a crear su documento.

A.6.4 Buscar presupuesto

Descripción: Permite al usuario buscar un presupuesto según varios parámetros como descripción, fecha o cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de ver presupuestos.

Flujo normal:

1. El usuario escribe en el buscador el parámetro con el que desea realizar la búsqueda.
2. El sistema filtra la lista general y muestra solo los que cumplen las condiciones.

Flujo alternativo

2. Si ningún presupuesto cumple las condiciones se mostrará la lista vacía.

A.6.5 Ver clientes

Descripción: Permite al usuario ver una lista con los datos de cada cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene la sesión iniciada.

Flujo normal:

1. El usuario pulsa el botón 'Gestionar clientes'.
2. El sistema obtiene todos los clientes del usuario que tiene la sesión iniciada y los muestra en una lista con todos sus datos.

Flujo alternativo:

2. Si el usuario aún no ha creado ningún cliente se mostrará un mensaje indicándolo.

A.6.6 Buscar clientes

Descripción: Permite al usuario buscar un cliente por dirección, nombre, dni o número de teléfono.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y deberá de estar en la vista de ver clientes.

Flujo normal:

1. El usuario escribe en el buscador el parámetro con el que desea realizar la búsqueda.
2. El sistema filtra la lista general y muestra solo los que cumplen las condiciones.

Flujo alternativo

2. Si ningún cliente cumple las condiciones se mostrará la lista vacía.

A.6.7 Editar cliente

Descripción: Permite al usuario modificar el cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y debe de estar en la vista de gestionar clientes.

Flujo normal:

1. El usuario pulsa el botón con el icono del lápiz del cliente que quiere editar.
2. El sistema muestra una ventana con los datos del cliente actuales.
3. El usuario modifica el cliente y pulsa 'Actualizar'.
4. El sistema actualiza el cliente en la base de datos.

Flujo alternativo

3. Si el usuario deja algún campo vacío el sistema le mostrará un error indicando qué campo está vacío y no lo actualizará.

A.6.8 Eliminar cliente

Descripción: Permite al usuario eliminar un cliente de la base de datos.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada y estará en la página de gestionar los clientes.

Flujo normal:

1. El usuario pulsa el botón de la papelera del cliente que quiere eliminar.
2. Si el cliente no tiene ningún documento asociado el sistema mostrará una nueva ventana con el mensaje '¿Seguro que quieres eliminar el cliente?' en el que el usuario podrá decidir si continuar con la eliminación.
3. El usuario pulsa 'Si'.
4. El sistema eliminará el presupuesto de la base de datos y dejará de mostrarse en la lista.

Flujo alternativo 1:

2. Si el cliente tiene documentos asociados no podrá eliminarse, esto se indicará con el mensaje 'No es posible eliminar el cliente, eliminar primero sus facturas o presupuestos'.

Flujo alternativo 2:

3. El usuario pulsa ‘No’.

4. Se cerrará la ventana y no se eliminará el presupuesto de la lista.

A.6.9 Crear cliente

Descripción: Permite al usuario crear un cliente.

Actor: Usuario logueado.

Precondiciones: El usuario tiene que tener la sesión iniciada.

Flujo normal:

1. El usuario pulsa el botón ‘Nuevo cliente’.
2. El sistema muestra una ventana con los campos de texto ‘Nombre’, ‘Dirección’, ‘Población’, ‘DNI’ y ‘Teléfono’.
3. El usuario escribe los datos del cliente y pulsa ‘Guardar’.
4. El sistema añade el cliente a la base de datos.

Flujo alternativo

3. Si el usuario deja algún campo vacío el sistema le mostrará un error indicando qué campo está vacío y no lo actualizará.

Anexo B: Diagrama de casos de uso final

En la siguiente imagen (Figura 49) se muestra el diagrama de casos de uso resultante tras finalizar todas las iteraciones.

Figura 49. Diagrama de casos de uso final

Anexo C: Implementación Flutter

C.1 Estructura de los directorios

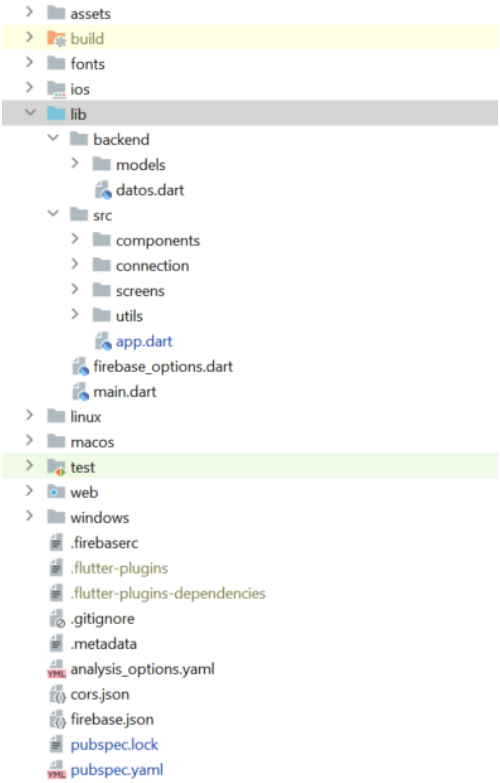


Figura 50. Directorios

C.2 pubspec.yaml dependencias

- Gestión de base de datos, autenticación, almacenaje: [20]

```
cloud_firestore: 3.4.2
firebase_auth: any
uuid: any
firebase_storage: ^10.3.4
cloud_firestore_web: ^2.8.2
```

Figura 51. Dependencias de gestión de base de datos, autenticación, almacenaje

- Mantener la sesión iniciada: [20]

```
shared_preferences: ^2.0.15
```

Figura 52. Dependencia para mantener la sesión iniciada

- Generar PDF: [20]

```
#PDF generator
pdf: ^3.8.1
path_provider: ^2.0.11
open_file: any
syncfusion_flutter_pdf: any
universal_html: ^1.2.1
sqflite: any
image_picker: 0.8.5+3
printing: ^5.7.2
http: ^0.13.5
```

Figura 53. Dependencias para generar PDF

- Logo:

```
flutter_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/Icon.png"
  min_sdk_android: 21
  web:
    generate: true
    image_path: "assets/Icon.png"
    background_color: "#hexcode"
    theme_color: "#hexcode"
  windows:
    generate: true
    image_path: "assets/Icon.png"
    icon_size: 48
```

Figura 54. Dependencias logo

- Fuentes PDF:

```
fonts:
  - family: Montserrat
    fonts:
      - asset: assets/OpenSans-Regular.ttf
      - asset: assets/OpenSans-Bold.ttf
      - asset: assets/OpenSans-Italic.ttf
      - asset: assets/OpenSans-BoldItalic.ttf
```

Figura 55. Fuentes de PDF

C.3 Implementación de la interfaz gráfica (directorio src/screens)

C.3.1 RegisterPage

RegisterPage es una clase de tipo StatefulWidget, está dentro del fichero register_page.dart. Esta misma página se encarga tanto de registrar como de modificar los datos del usuario. Cuando el usuario está modificando sus datos, initState recogerá los datos recibidos y los mostrará en los campos correspondientes. _register() valida los datos y los modifica o registra. showSnackBar() muestra los mensajes con el color y texto que se le pasa como parámetro. _pickImage() gestiona todo lo relacionado con la selección de imagen en cada sistema operativo.

C.3.2 HomeSelectionPage

HomeSelectionPage es una clase de tipo StatefulWidget, está dentro del fichero home_selection_page.dart.

En el método build se construye la cabecera, que tendrá un acceso directo para poder crear un cliente y también el menú desplegable lateral (UserDrawer). En el centro de la pantalla se mostrarán todos los botones que están construidos haciendo uso del Widget SingleScrollView, esto permitirá que en el caso de que el usuario disminuya en exceso el tamaño de pantalla se cree una barra para que se pueda desplazar por los diferentes botones. Según el tamaño de la pantalla también cambiará el tamaño de la fuente.

Los cinco botones centrales son de tipo 'ElevatedButton', se encargan únicamente de redirigir al usuario a la página que desea.

C.3.3 invoiceDataResponsivePage, invoiceDataDesktopPage y invoiceDataMobilePage

invoiceDataResponsivePage, al igual que budgetDataResponsivePage, es un StatefulWidget que solo se encarga de crear la barra superior, añadir la imagen de fondo y seleccionar entre si hacer uso de la interfaz de móvil (invoiceDataMobilePage) o la de ordenador (invoiceDataDesktopPage), usando para ello la clase que se encuentra en el directorio utils, ResponsiveLayout (más atrás).

Al igual que en el anterior punto, invoiceDataDesktopPage será la clase que se utilice cuando el dispositivo sobre el que se está ejecutando es un ordenador o cualquier dispositivo con pantalla grande e invoiceDataMobilePage será la utilizada en móviles o pantallas pequeñas. La única diferencia será que no existe el campo 'Descripción final' y que todas las operaciones como las de guardado de factura o creación de PDFs se harán con facturas.

C.3.4 budgetDataResponsivePage, budgetDataDesktopPage y budgetDataMobilePage

budgetDataResponsivePage es un StatefulWidget que solo se encarga de crear la barra superior, añadir la imagen de fondo y seleccionar entre si hacer uso de la interfaz de móvil (budgetDataMobilePage) o la de ordenador (budgetDataDesktopPage), usando para ello la clase ResponsiveLayout (más atrás).

budgetDataDesktopPage y budgetDataMobilePage tienen las mismas funcionalidades, la única diferencia es que el formato cambiará según el tamaño y formato de pantalla.

Como en todas las clases de Flutter, el método build se encarga de construir en forma de árbol la estructura principal de la interfaz.

Ambas clases comparten los siguientes métodos. initState inicializa la lista de partidas que se han seleccionado anteriormente. createDropdown, createMyData, makeTotal y getCurrentBillItem se encargan de crear los principales Widgets que se visualizan. Son llamados desde el método build, obtienen los datos de la base de datos y les dan el formato adecuado.

deleteBillItem se encarga de eliminar partidas del documento mostrando antes un mensaje para que el usuario confirme si quiere eliminarlo.

Otros métodos que se utilizan son addEditClientDialog (abre un diálogo para introducir un nuevo cliente), showSnackBar (muestra los mensajes de error o confirmación en la parte inferior de la pantalla) y questionDialog (abre un diálogo para confirmar o cancelar una acción).

C.3.5 budgetsManagementPage, invoicesManagementPage

StatefulWidget que serán los encargados de mostrar la lista de presupuestos/facturas junto con las opciones de cada uno (Abrir PDF, modificar y eliminar).

Ambas clases contienen los mismos métodos y funcionalidades, solo que budgetsManagementPage tendrá todos adaptado a presupuestos e invoicesManagementPage a facturas.

Se declaran las variables en las que se guardará la lista de presupuestos completa y la filtrada, además de todas las necesarias para buscar elementos y filtrarlos (searchview, firstSearch, query). El listener al comienzo de la clase se encargará de atender a cualquier cambio que sufra la barra de búsqueda.

El método initState inicializará la lista de presupuestos/facturas y las tres siguientes funciones que crearán unos Widget más pequeños (_createBudgetList/_createInvoiceList, createFilteredBudgetList/createFilteredInvoiceList, createSearchView) que serán llamados desde el método build. También se hará uso del Widget personalizado Budget_widget/Invoice_widget, que se explicarán en el apartado 6.1.5 Implementación de la interfaz gráfica. Pequeños Widgets personalizados (directorio src/components). Este llamará a la función deleteBudget/deleteInvoice en caso de que el cliente pulse en el botón de eliminar del StatelessWidget.

Los otros métodos se encargarán de filtrar (filter), construir el diálogo de creación y edición de clientes (_addEditClientDialog), mostrar mensajes tanto de avisos como de error en la parte inferior de la pantalla (_showSnackBar) y de crear el diálogo en el que el usuario deberá seleccionar entre si continuar o no con una acción.

C.3.6 clientManagementPage

StatefulWidget que será el encargado de mostrar la lista de clientes junto con las opciones de cada una (Modificar y eliminar).

Todos los métodos y widgets son similares a los anteriores (C.3.5 budgetsManagementPage, invoicesManagementPage) pero adaptándose a la lista de clientes.

C.4 Implementación de la interfaz gráfica. Pequeños Widgets personalizados (directorio src/components)

C.4.1 invoiceWidget

Widget de cada elemento de la lista de facturas, está en el fichero invoice_widget.dart. Tiene como variables index, invoice, serverController e invoiceDeleteCallBack.

Al comienzo del método build se asignará el color del elemento según su índice, calculará tanto el precio total de la factura como el precio con IVA y creará el Widget que visualizará el usuario.

createPDF abrirá en una nueva pestaña la factura en formato PDF.

C.4.2 clientWidget

Widget de cada elemento de la lista de clientes, se encuentra en el fichero client_widget.dart. El Widget de tipo ListTile, al igual que se ve en la Figura 36, mostrará toda la información del cliente.

C.4.3 itemWidget

Mostrará cada elemento de la lista de partidas de la página principal de la aplicación, tiene varias variables de tipo ItemCallBack para editar, eliminar y añadir a la lista del documento, que hará que se ejecuten las funciones indicadas que se encuentran en el lugar donde se crea este Widget.

C.4.4 ItemAddedWidget

Mostrará cada partida añadida al documento. Calculará el precio total de la partida y mostrará todos sus datos. También crea un botón que tiene como texto la cantidad de unidades que si se pulsa abrirá el dialogo de edición de cantidad.

C.4.5 UserDrawer

Se hará cargo de mostrar el menú de la barra lateral izquierda. El Widget principal que utiliza es de tipo UserAccountsDrawerHeader, en él se mostrarán los datos del usuario, si se pulsa encima se podrán modificar los datos del mismo.

Desde este se podrá acceder a cualquier funcionalidad del programa.

C.4.6 UserWidget

Widget que muestra los datos del usuario. Este se muestra cuando el usuario va a introducir los datos del documento, en caso de pulsar su botón se abrirá la misma página que se utiliza para el registro, pero esta vez contendrá todos los datos del usuario.

C.5 Modelos (directorio backend/models)

C.5.1 Item

Fichero item.dart. Clase que representa cada partida guardada por el usuario.

Parámetros: id, userId, concept, price.

Métodos: setId, setUserId, toString, toFirestore.

C.5.2 DocumentItem

Fichero document_item.dart. Representa cada partida añadida a un documento.

Parámetros: id, userId, concept, price, quantity.

Métodos: setId, serUserId, setQuantity, toString, toFirestore.

C.5.3 Budget

Fichero budget.dart. Clase que representa un presupuesto.

Parámetros: id, userId, budgetNum, description, finalComment, budgetItems, client, user, date.

Métodos: setId, setUserId, setBillItems, toFirestore, toString.

C.5.4 Client

Fichero client.dart. Clase que representa un cliente.

Parámetros: id, userId, name, address, town, dni, tlf.

Métodos: setId, setUserId, toFirestore, toString.

C.5.5 User

Fichero user.dart. Clase que representa al usuario registrado.

Parámetros: id, email, address, dni, bankAccount, phone, instNum, empNum, imageUrl.

Métodos: login, setId, setUrl, toFirestore, toString.

C.6 Otros útiles (directorio src/utils)

C.6.1 PdfInvoiceGenerator

Contiene las mismas funciones que PdfBudgetGenerator pero adaptadas para generar una factura en lugar de un presupuesto.

La única diferencia estará en los siguientes métodos:

- buildResume: Generará la parte final de la tabla, en la que se mostrará la suma final de las partidas, el IVA y la suma total con el IVA incluido.
- buildFooter: Mostrará automáticamente la cuenta de banco en la que se deberá de ingresar el dinero.

Anexo D: Pruebas de funcionalidades

D.1 Pruebas Login

Caso 1. El usuario deja el campo del correo electrónico y/o contraseña vacíos.

Entrada: Campo de correo electrónico y/o contraseña vacía.

Salida esperada: Los campos vacíos se resaltan en rojo y muestran el mensaje 'Campo obligatorio'.

Salida real: Igual a salida esperada.

Caso 2. El usuario introduce un correo electrónico con formato no adecuado.

Entrada: Campo de correo electrónico tiene un valor con formato no válido. 'yara@'.

Salida esperada: El campo del correo electrónico se resalta en rojo y muestra el mensaje 'El correo no es válido, comprueba su formato'.

Salida real: Igual a salida esperada.

Caso 3. El usuario introduce una contraseña no válida, es decir, menos de 6 caracteres.

Entrada: Valor del campo de la contraseña no válido. 'a'.

Salida esperada: El campo de la contraseña se resalta en rojo y muestra el mensaje 'La contraseña debe de tener al menos 6 caracteres'.

Salida real: Igual a salida esperada.

Caso 4. El usuario quiere visualizar la contraseña.

Entrada: Campo de contraseña con el valor 'passPrueba0' y el usuario pulsa el botón para hacerla visible.

Salida esperada: Se muestra la contraseña 'passPrueba0'.

Salida real: Igual a salida esperada.

Caso 5. El correo o la contraseña no son correctos o la cuenta no existe.

Entrada: Campo de correo y contraseña con formato correcto, pero datos sus valores no corresponden a ninguna cuenta de la base de datos.

Salida esperada: Mensaje de error en rojo 'Correo o contraseña incorrectos'.

Salida real: Igual a salida esperada.

Caso 6. El correo y la contraseña tienen un formato correcto y el usuario registrado introduce sus datos correctamente.

Entrada: Todos los datos son correctos.

Salida esperada: El sistema redirige al usuario a la página de selección.

Salida real: Igual a salida esperada.

D.2 Pruebas register

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Cuenta de banco a ingresar' vacío.

Salida esperada: Los campos vacíos se resaltan en rojo y muestran el mensaje 'Campo obligatorio'.

Salida real: Igual a salida esperada.

Caso 2. El usuario no ha seleccionado logo.

Entrada: Logo no seleccionado.

Salida esperada: En la parte inferior de la pantalla se muestra el mensaje 'Por favor, selecciona una imagen para continuar'.

Salida real: Igual a salida esperada.

Caso 3. El usuario introduce un correo electrónico con formato no adecuado.

Entrada: Campo de correo electrónico tiene un valor con formato no válido. 'yara@'.

Salida esperada: El campo del correo electrónico se resalta en rojo y muestra el mensaje 'El correo no es válido, comprueba su formato'.

Salida real: Igual a salida esperada.

Caso 4. El usuario introduce una contraseña no válida, es decir, menos de 6 caracteres.

Entrada: Valor del campo de la contraseña no válido. 'a'.

Salida esperada: El campo de la contraseña se resalta en rojo y muestra el mensaje 'La contraseña debe de tener al menos 6 caracteres'.

Salida real: Igual a salida esperada.

Caso 5. El usuario quiere visualizar la contraseña.

Entrada: Campo de contraseña con el valor 'passPrueba0' y el usuario pulsa el botón para hacerla visible.

Salida esperada: Se muestra la contraseña 'passPrueba0'.

Salida real: Igual a salida esperada.

Caso 6. Todos los campos tienen un formato correcto y el logo está seleccionado.

Entrada: Todos los datos son correctos.

Salida esperada: El sistema muestra un mensaje para indicarle que se ha registrado correctamente y redirige a la página de selección.

Salida real: Igual a salida esperada.

D.3 Pruebas logout

Caso 1. El usuario pulsa 'Cerrar sesión'.

Entrada: Pulsar 'Cerrar sesión'.

Salida esperada: El sistema cierra la sesión y redirige al usuario a la página de LOGIN.

Salida real: Igual a salida esperada.

D.4 Pruebas Modificar datos personales

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Cuenta de banco a ingresar' vacío.

Salida esperada: Los campos vacíos se resaltan en rojo y muestran el mensaje 'Campo obligatorio'.

Salida real: Igual a salida esperada.

Caso 2. El usuario no ha seleccionado logo.

Entrada: Logo no seleccionado.

Salida esperada: En la parte inferior de la pantalla se muestra el mensaje 'Por favor, selecciona una imagen para continuar'.

Salida real: Igual a salida esperada.

Caso 3. El usuario introduce un correo electrónico con formato no adecuado.

Entrada: Campo de correo electrónico tiene un valor con formato no válido. 'yara@'.

Salida esperada: El campo del correo electrónico se resalta en rojo y muestra el mensaje 'El correo no es válido, comprueba su formato'.

Salida real: Igual a salida esperada.

Caso 4. El usuario introduce una contraseña no válida, es decir, menos de 6 caracteres.

Entrada: Valor del campo de la contraseña no válido. 'a'.

Salida esperada: El campo de la contraseña se resalta en rojo y muestra el mensaje 'La contraseña debe de tener al menos 6 caracteres'.

Salida real: Igual a salida esperada.

Caso 5. El usuario quiere visualizar la contraseña.

Entrada: Campo de contraseña con el valor 'passPrueba0' y el usuario pulsa el botón para hacerla visible.

Salida esperada: Se muestra la contraseña 'passPrueba0'.

Salida real: Igual a salida esperada.

Caso 6. Todos los campos tienen un formato correcto y el logo está seleccionado.

Entrada: Todos los datos son correctos.

Salida esperada: El sistema muestra un mensaje para indicarle que se ha registrado correctamente y redirige a la página de selección.

Salida real: Igual a salida esperada.

D.5 Pruebas Crear cliente

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Nombre' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'El nombre está vacío'.

Salida real: Igual a salida esperada.

Caso 2. El usuario no deja ningún campo vacío.

Entrada: Todos los campos están completos.

Salida esperada: El sistema muestra un mensaje para indicarle al usuario que el cliente se ha registrado correctamente.

Salida real: Igual a salida esperada.

D.6 Pruebas Editar partida

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Concepto' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'El concepto está vacío'.

Salida real: Igual a salida esperada.

Caso 2. El usuario introduce un precio con formato inadecuado.

Entrada: Precio con formato inadecuado. '2,2,2'.

Salida esperada: El sistema no permite escribir la segunda coma, escribe '2,22'.

Salida real: Igual a salida esperada.

Caso 3. El usuario introduce un precio con caracteres inadecuados.

Entrada: Precio con formato inadecuado. 'a2bc'.

Salida esperada: El sistema no permite ningún carácter no válido, escribe solo '2'.

Salida real: Igual a salida esperada.

Caso 4. El usuario introduce todos los campos correctamente.

Entrada: Campo de concepto completo y precio con formato adecuado. Concepto: 'Prueba', precio: '2,5'.

Salida esperada: Se muestra el mensaje 'Concepto guardado' en el centro de la pantalla para indicarle al usuario que se ha completado la tarea con éxito.

Salida real: Igual a salida esperada.

D.7 Pruebas Eliminar partida

Caso 1. El usuario elimina una partida de la lista.

Entrada: El usuario pulsa 'Si' al eliminar para eliminar la partida de la lista.

Salida esperada: Se elimina la partida seleccionada y se actualiza la lista.

Salida real: Igual a salida esperada.

Caso 2. El usuario comienza a eliminar una partida, pero luego cancela la acción.

Entrada: El usuario pulsa 'No' al eliminar para eliminar la partida de la lista, cancela la acción.

Salida esperada: No se elimina nada.

Salida real: Igual a salida esperada.

D.8 Pruebas Añadir partida al documento

Caso 1. La partida ya está en el documento.

Entrada: Se intenta añadir una partida que ya está en el documento.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'La partida ya está en el documento'.

Salida real: Igual a salida esperada.

Caso 2. El usuario deja el campo cantidad vacío.

Entrada: Campo 'Cantidad' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'La cantidad está vacía'.

Salida real: Igual a salida esperada.

Caso 3. El usuario mete una cantidad con un formato no adecuado.

Entrada: El usuario mete como cantidad 'aa2'.

Salida esperada: El sistema no deja meter caracteres no permitidos, solo escribirá '2'.

Salida real: Igual a salida esperada.

Caso 4. El usuario mete como cantidad 0.

Entrada: Campo 'Cantidad' con el valor 0.

Salida esperada: La partida no se añade al documento.

Salida real: Igual a salida esperada.

Caso 5. El usuario mete como cantidad un valor negativo.

Entrada: Campo 'Cantidad' con un valor negativo, '-2'.

Salida esperada: El sistema no deja escribir números negativos, escribe '2'.

Salida real: Igual a salida esperada.

Caso 6. El campo cantidad es correcto.

Entrada: Campo 'Cantidad' con un valor correcto, '2'.

Salida esperada: El sistema añade dos unidades de la partida seleccionada al documento.

Salida real: Igual a salida esperada.

D.9 Pruebas Eliminar partida de documento

Caso 1. El usuario elimina una partida del documento.

Entrada: El usuario pulsa 'Si' al eliminar para eliminar la partida de la lista.

Salida esperada: Se elimina la partida seleccionada y se actualiza la lista de partidas en el documento.

Salida real: Igual a salida esperada.

Caso 2. El usuario comienza a eliminar una partida del documento, pero luego cancela la acción.

Entrada: El usuario pulsa 'No' al eliminar para eliminar la partida, cancela la acción.

Salida esperada: No se elimina nada.

Salida real: Igual a salida esperada.

D.10 Pruebas Modificar cantidad de partida de documento

Caso 1. El usuario deja el campo cantidad vacío.

Entrada: Campo 'Cantidad' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'La cantidad está vacía'.

Salida real: Igual a salida esperada.

Caso 2. El usuario mete una cantidad con un formato no adecuado.

Entrada: El usuario mete como cantidad 'aa2'.

Salida esperada: El sistema no deja meter caracteres no permitidos, solo escribirá '2'.

Salida real: Igual a salida esperada.

Caso 3. El usuario mete como cantidad 0.

Entrada: Campo 'Cantidad' con el valor 0.

Salida esperada: La partida no se añade al documento.

Salida real: Igual a salida esperada.

Caso 4. El usuario mete como cantidad un valor negativo.

Entrada: Campo 'Cantidad' con un valor negativo, '-2'.

Salida esperada: El sistema no deja escribir números negativos, escribe '2'.

Salida real: Igual a salida esperada.

Caso 5. El campo cantidad es correcto.

Entrada: Campo 'Cantidad' con un valor correcto, '2'.

Salida esperada: El sistema añade dos unidades de la partida seleccionada al documento.

Salida real: Igual a salida esperada.

D.11 Prueba Vaciar documento

Caso 1. La lista está vacía.

Entrada: Lista vacía.

Salida esperada: No pasa nada.

Salida real: Igual a salida esperada.

Caso 2. La lista tiene elementos y el usuario confirma que quiere eliminarlos.

Entrada: La lista tiene elementos y el usuario pulsa 'Si' para confirmar que quiere eliminar las partidas.

Salida esperada: Se vacía el documento y se muestra el mensaje 'El documento está vacío, añade partidas para continuar'.

Salida real: Igual a salida esperada.

Caso 3. La lista tiene elementos y el usuario cancela la acción de eliminado.

Entrada: La lista tiene elementos y el usuario pulsa 'No' para cancelar la acción.

Salida esperada: No pasa nada.

Salida real: Igual a salida esperada.

D.12 Pruebas Búsqueda de partidas

Caso 1. Ningún concepto coincide con la búsqueda del usuario.

Entrada: Fragmento que no coincide con ningún concepto.

Salida esperada: Se muestra la lista vacía.

Salida real: Igual a salida esperada.

Caso 2. La búsqueda coincide con algún concepto.

Entrada: Fragmento que coincide con algún concepto.

Salida esperada: Se muestran solo los elementos que coinciden con la búsqueda.

Salida real: Igual a salida esperada.

D.13 Pruebas Ver mis facturas. Búsqueda

Caso 1. Ninguna factura coincide con el nombre del cliente, fecha o comentario inicial.

Entrada: Fragmento que no coincide con ninguna factura.

Salida esperada: Se muestra la lista vacía.

Salida real: Igual a salida esperada.

Caso 2. La búsqueda coincide con algún nombre del cliente, fecha o comentario inicial.

Entrada: Fragmento que coincide con alguna factura.

Salida esperada: Se muestran solo las facturas que coinciden con la búsqueda.

Salida real: Igual a salida esperada.

D.14 Pruebas Ver mis facturas. Descarga

Caso 1. El usuario pulsa el icono de descargar factura.

Entrada: Pulsar el icono descargar de una factura.

Salida esperada: Se muestra el mensaje 'Abriendo factura' y se abre una ventana con el PDF.

Salida real: Igual a salida esperada.

D.15 Pruebas Ver mis facturas. Eliminar

Caso 1. El usuario quiere eliminar una factura y lo confirma.

Entrada: El usuario pulsa 'Si' para eliminar la factura.

Salida esperada: Se elimina la factura seleccionada.

Salida real: Igual a salida esperada.

Caso 2. El usuario comienza a eliminar una factura, pero luego cancela la acción.

Entrada: El usuario pulsa 'No' y cancela la acción.

Salida esperada: No se elimina nada.

Salida real: Igual a salida esperada.

D.16 Pruebas Ver mis facturas. Modificar

Caso 1. El usuario abre la factura para modificarla.

Entrada: El usuario pulsa sobre la factura que quiere abrir.

Salida esperada: Se abre la vista de creación de documentos con todas las partidas que contenía la factura seleccionada.

Salida real: Igual a salida esperada.

D.17 Pruebas Ver mis presupuestos. Búsqueda

Caso 1. Ningún presupuesto coincide con el nombre del cliente, fecha, comentario inicial o comentario final.

Entrada: Fragmento que no coincide con ningún presupuesto.

Salida esperada: Se muestra la lista vacía.

Salida real: Igual a salida esperada.

Caso 2. La búsqueda coincide con algún nombre del cliente, fecha, comentario inicial o comentario final.

Entrada: Fragmento que coincide con algún presupuesto.

Salida esperada: Se muestran solo los presupuestos que coinciden con la búsqueda.

Salida real: Igual a salida esperada.

D.18 Pruebas Ver mis presupuestos. Descarga

Caso 1. El usuario pulsa el icono de descargar presupuesto.

Entrada: Pulsar el icono descargar de un presupuesto.

Salida esperada: Se muestra el mensaje 'Abriendo presupuesto y se abre una ventana con el PDF.

Salida real: Igual a salida esperada.

D.19 Pruebas Ver mis presupuestos. Eliminar

Caso 1. El usuario quiere eliminar un presupuesto y lo confirma.

Entrada: El usuario pulsa 'Si' para eliminar el presupuesto.

Salida esperada: Se elimina el presupuesto seleccionado.

Salida real: Igual a salida esperada.

Caso 2. El usuario comienza a eliminar un presupuesto, pero luego cancela la acción.

Entrada: El usuario pulsa 'No' y cancela la acción.

Salida esperada: No se elimina nada.

Salida real: Igual a salida esperada.

D.20 Pruebas Ver mis presupuestos. Modificar

Caso 1. El usuario abre el presupuesto para modificarlo.

Entrada: El usuario pulsa sobre el presupuesto que quiere abrir.

Salida esperada: Se abre la vista de creación de documentos con todas las partidas que contenía el presupuesto seleccionado.

Salida real: Igual a salida esperada.

D.21 Pruebas Ver mis clientes. Búsqueda

Caso 1. Ninguna factura coincide con el nombre del cliente, DNI, dirección o número de teléfono.

Entrada: Fragmento que no coincide con ningún cliente.

Salida esperada: Se muestra la lista vacía.

Salida real: Igual a salida esperada.

Caso 2. La búsqueda coincide con algún nombre del cliente, DNI, dirección o número de teléfono.

Entrada: Fragmento que coincide con algún cliente.

Salida esperada: Se muestran solo los clientes que coinciden con la búsqueda.

Salida real: Igual a salida esperada.

D.22 Pruebas Ver mis clientes. Eliminar

Caso 1. El usuario quiere eliminar un cliente, pero existen facturas o presupuestos creados para el cliente.

Entrada: Se selecciona un cliente que corresponde a alguna factura.

Salida esperada: El sistema muestra el mensaje 'No es posible eliminar el cliente, elimina primero sus facturas o presupuestos'.

Salida real: Igual a salida esperada.

Caso 2. El usuario intenta eliminar un cliente que no corresponde a ningún documento, pero cancela la acción.

Entrada: El usuario pulsa 'No' y cancela la acción.

Salida esperada: No se elimina nada.

Salida real: Igual a salida esperada.

Caso 3. El usuario pulsa sobre eliminar un cliente que no corresponde a ningún documento y lo confirma.

Entrada: El usuario pulsa 'Si' y confirma la acción.

Salida esperada: Se elimina el usuario.

Salida real: Igual a salida esperada.

D.23 Pruebas Ver mis clientes. Modificar

Caso 1. El usuario deja algún campo vacío.

Entrada: Campo 'Nombre' vacío.

Salida esperada: Se muestra un mensaje en la parte inferior de la pantalla: 'El nombre está vacío'.

Salida real: Igual a salida esperada.

Caso 2. El usuario no deja ningún campo vacío.

Entrada: Todos los campos están completos.

Salida esperada: El sistema muestra un mensaje para indicarle al usuario que el cliente se ha registrado correctamente.

Salida real: Igual a salida esperada.

D.24 Pruebas Crear factura/Crear presupuesto

Caso 1. La lista de partidas del documento está vacía.

Entrada: Lista de partidas vacía y el usuario pulsa 'Crear presupuesto' o 'Crear partida'.

Salida esperada: Los botones se muestran en gris y están inactivos, no ocurre nada al pulsarlos.

Salida real: Igual a salida esperada.

Caso 2. El usuario no deja ningún campo vacío.

Entrada: Todos los campos están completos y el usuario pulsa el botón para generar el PDF.

Salida esperada: Se abre una ventana con el documento PDF generado.

Salida real: Igual a salida esperada.

Caso 3. Algún campo está vacío.

Entrada: Algún campo vacío.

Salida esperada: Se resalta el campo vacío y muestra un mensaje indicando que el campo está vacío.

Salida real: Igual a salida esperada.

Caso 4. El usuario intenta crear un documento y la lista de clientes está vacía.

Entrada: No hay nada para seleccionar en la lista de clientes.

Salida esperada: El sistema muestra un mensaje de error indicando al usuario que debe de introducir un cliente.

Salida real: Igual a salida esperada.

Anexo E: Pruebas de adaptación

E.1 Inicio de sesión, registro, menú lateral

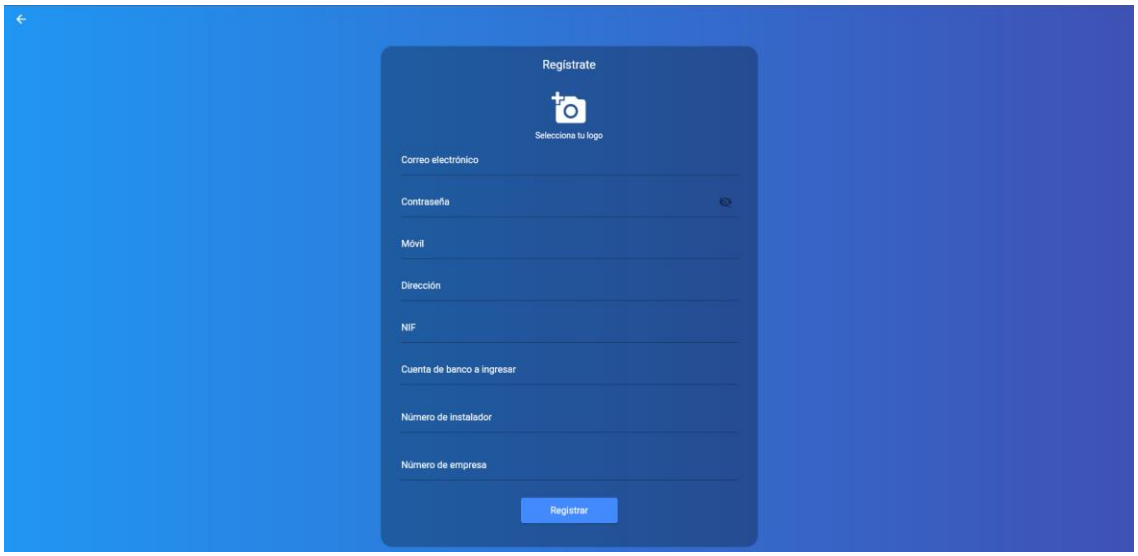


Figura 56. Interfaz ordenador: registro

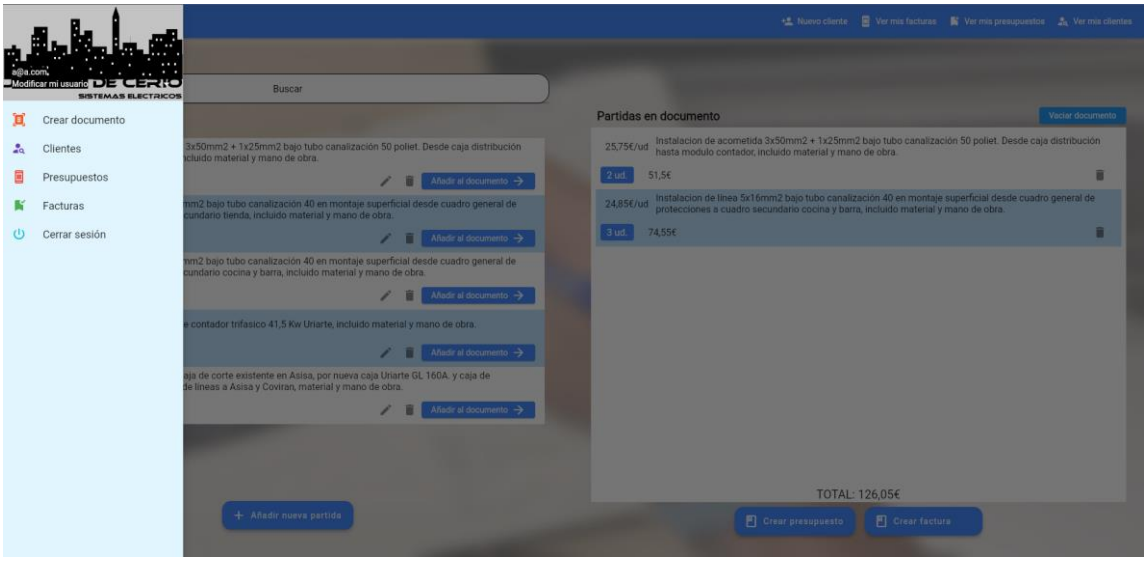


Figura 57. Interfaz ordenador: menú lateral

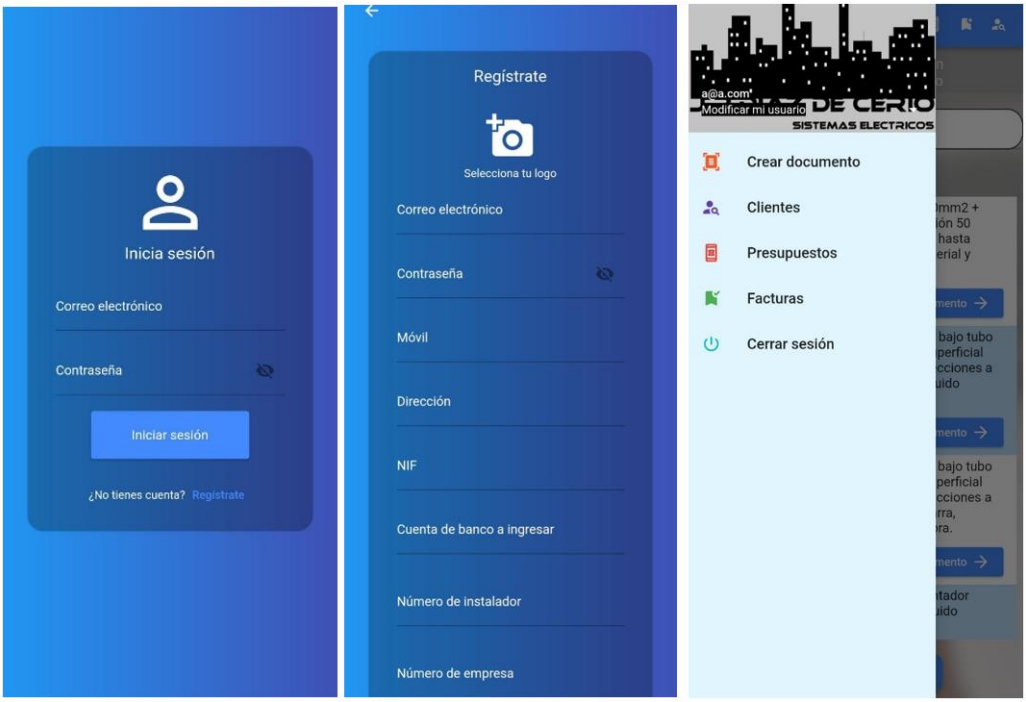
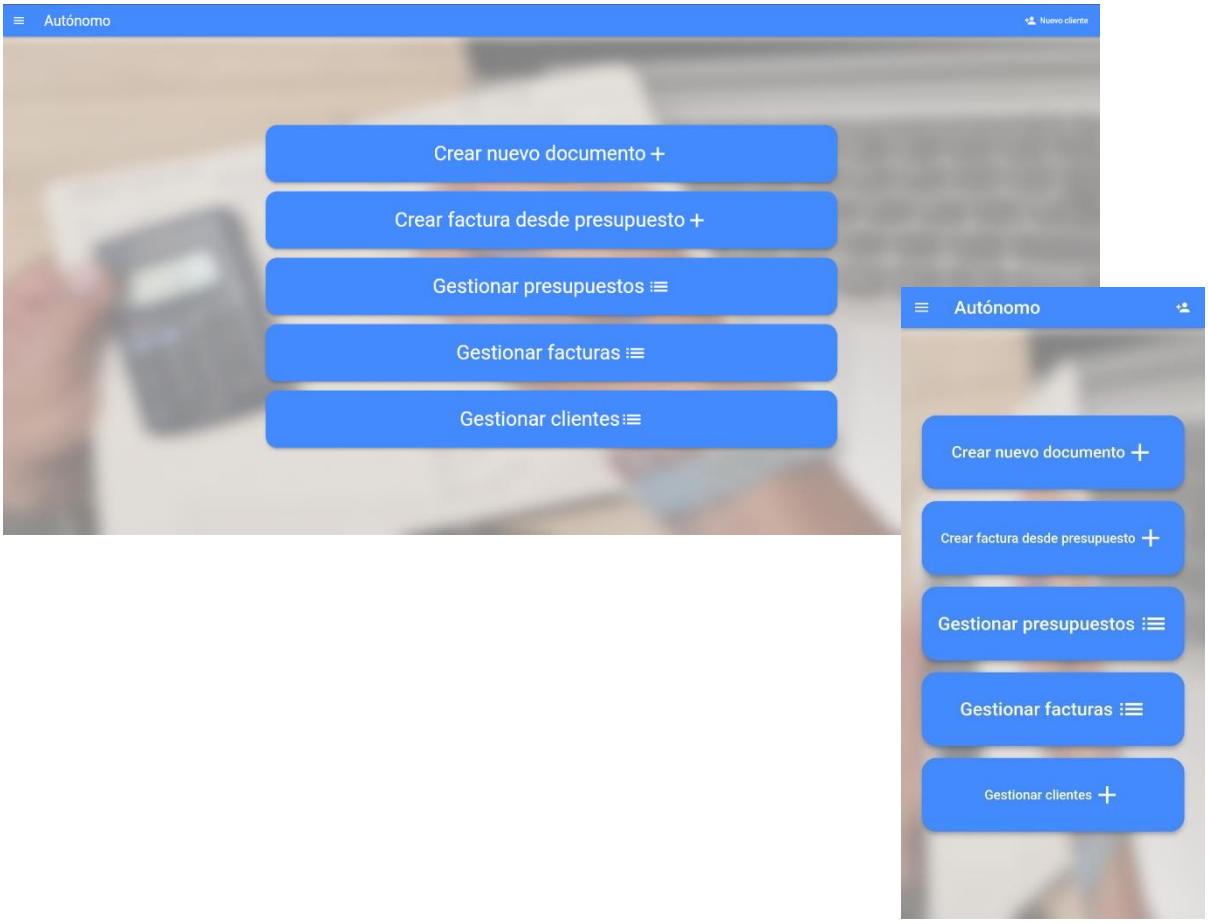


Figura 58. Interfaz móvil: inicio de sesión, registro, menú lateral

E.2 Menú de selección



E.3 Agregar partida, introducir cantidad de partida, introducir datos de presupuesto

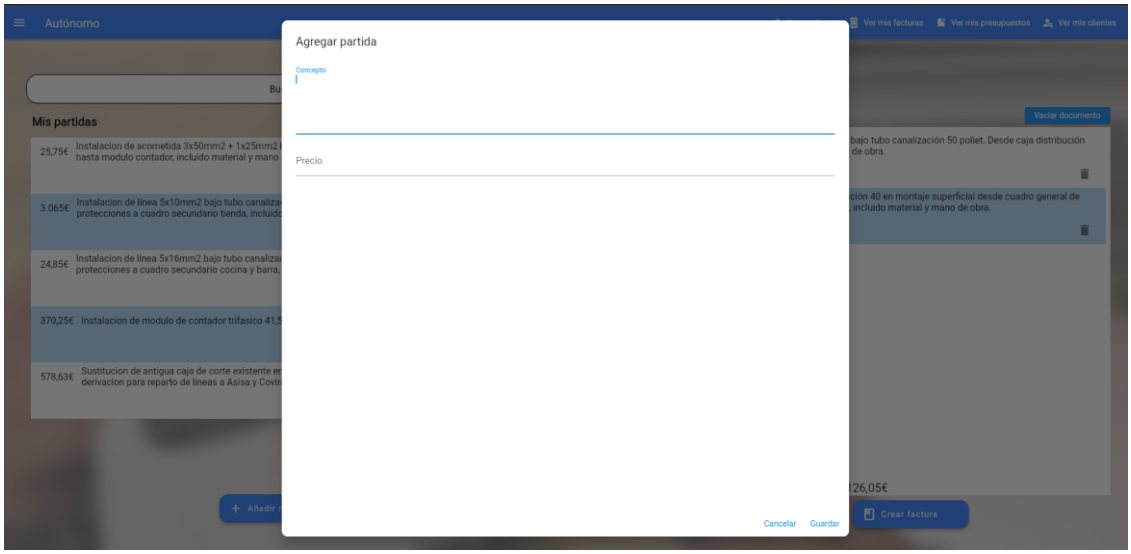


Figura 59. Interfaz ordenador: agregar partida

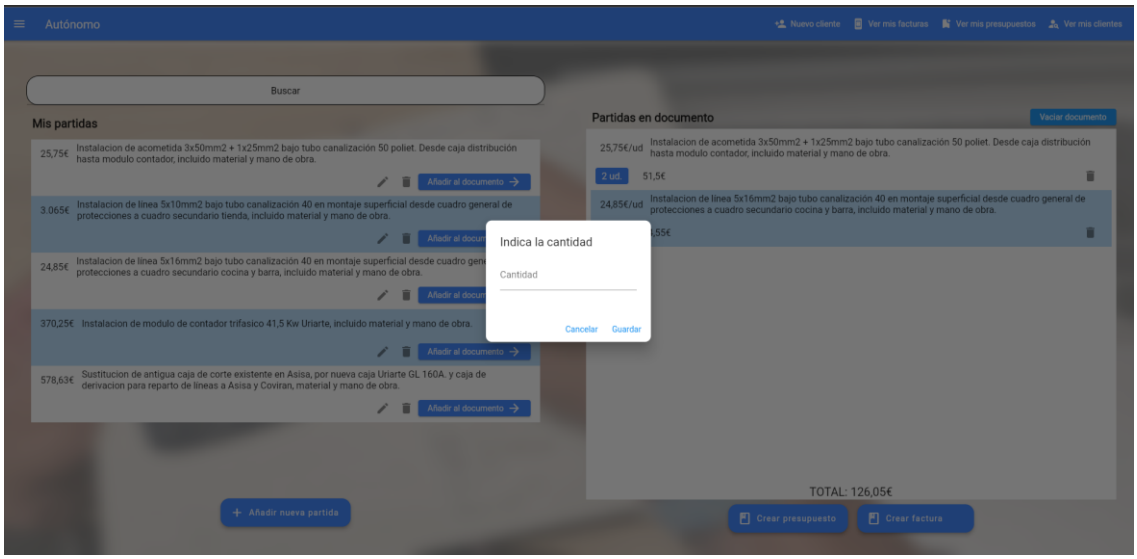


Figura 60. Interfaz ordenador: Introducir cantidad partida

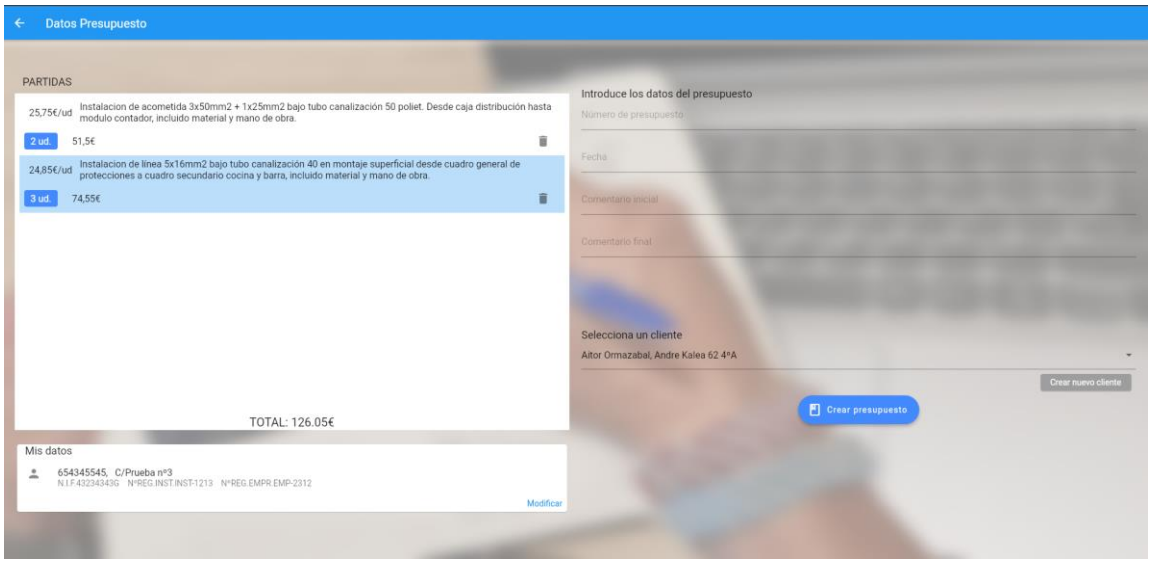


Figura 61. Interfaz ordenador: Introducir datos presupuesto

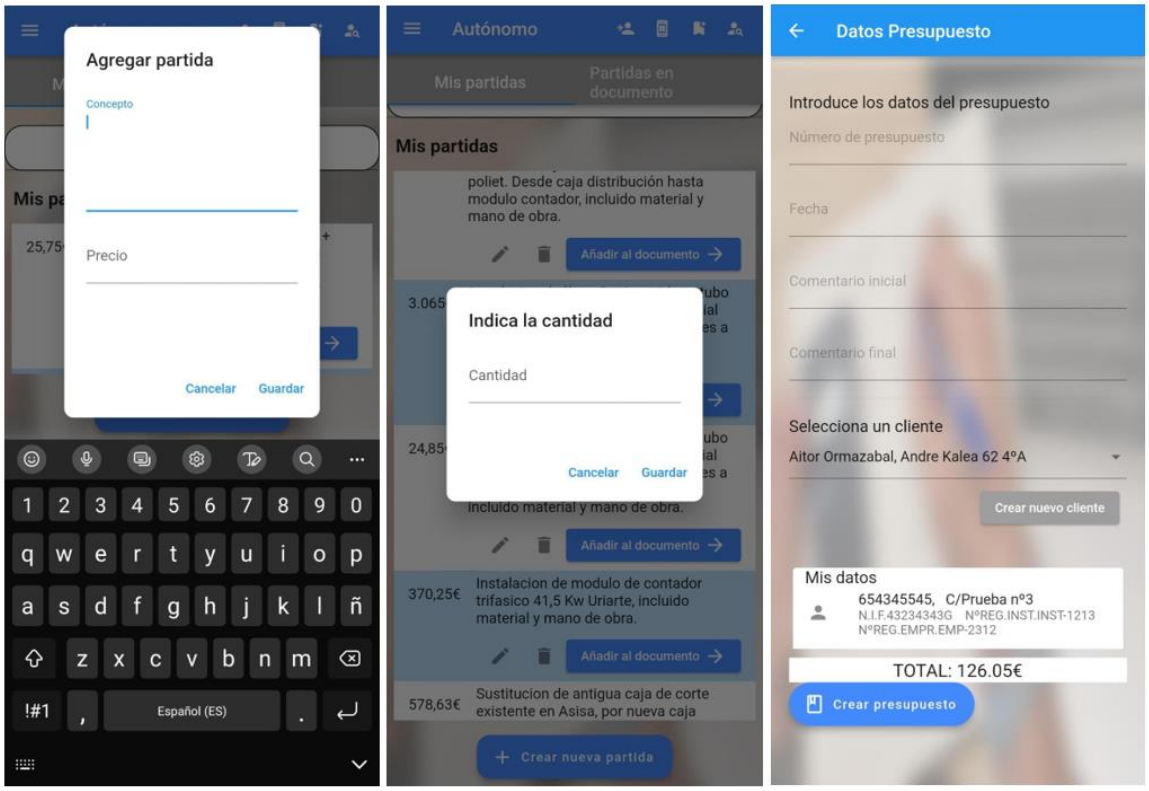


Figura 62. Interfaz móvil: Agregar partida, introducir cantidad, introducir datos presupuesto

7.2.4 Gestión clientes, gestión facturas, gestión presupuestos

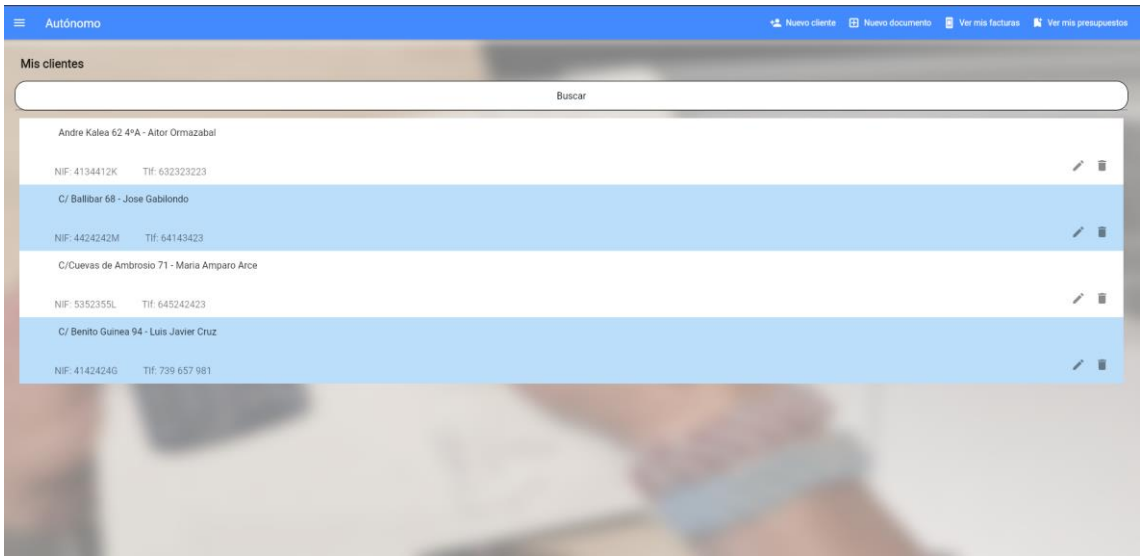


Figura 63. Interfaz ordenador: gestión clientes

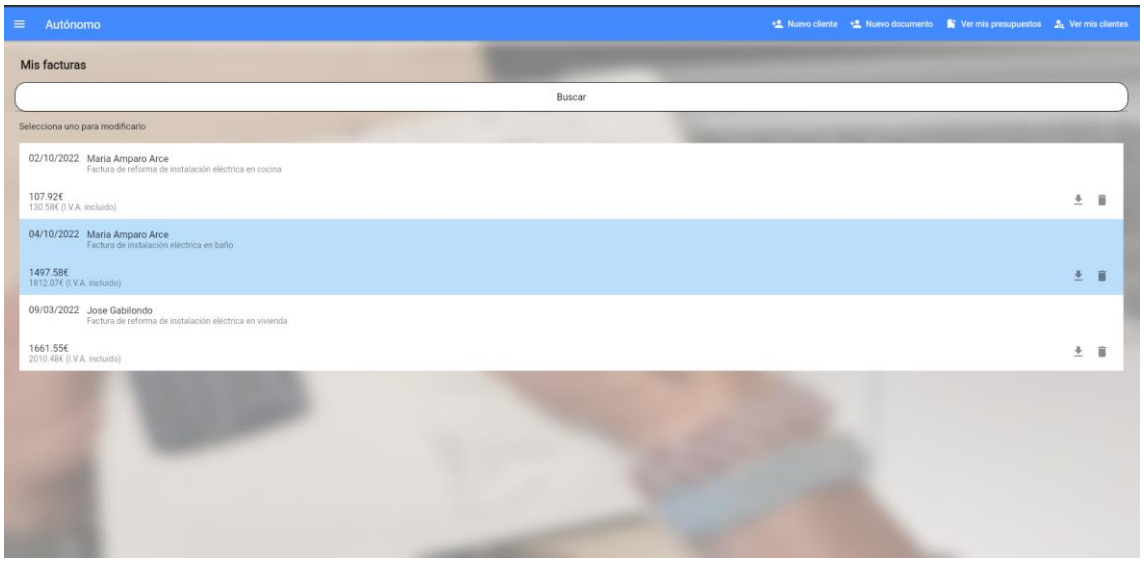


Figura 64. Interfaz ordenador: gestión facturas

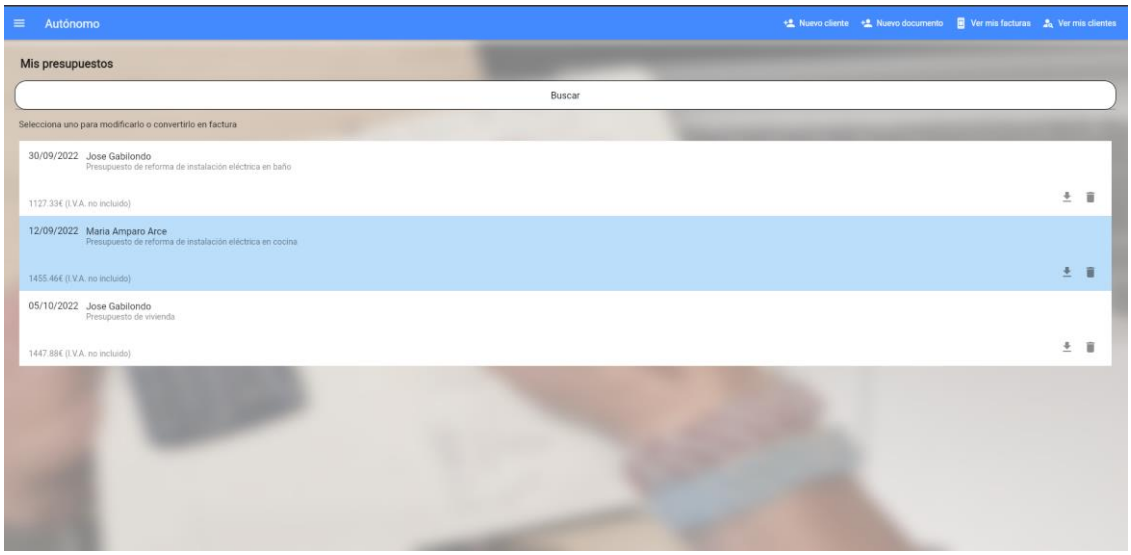


Figura 65. Interfaz ordenador: gestión presupuestos

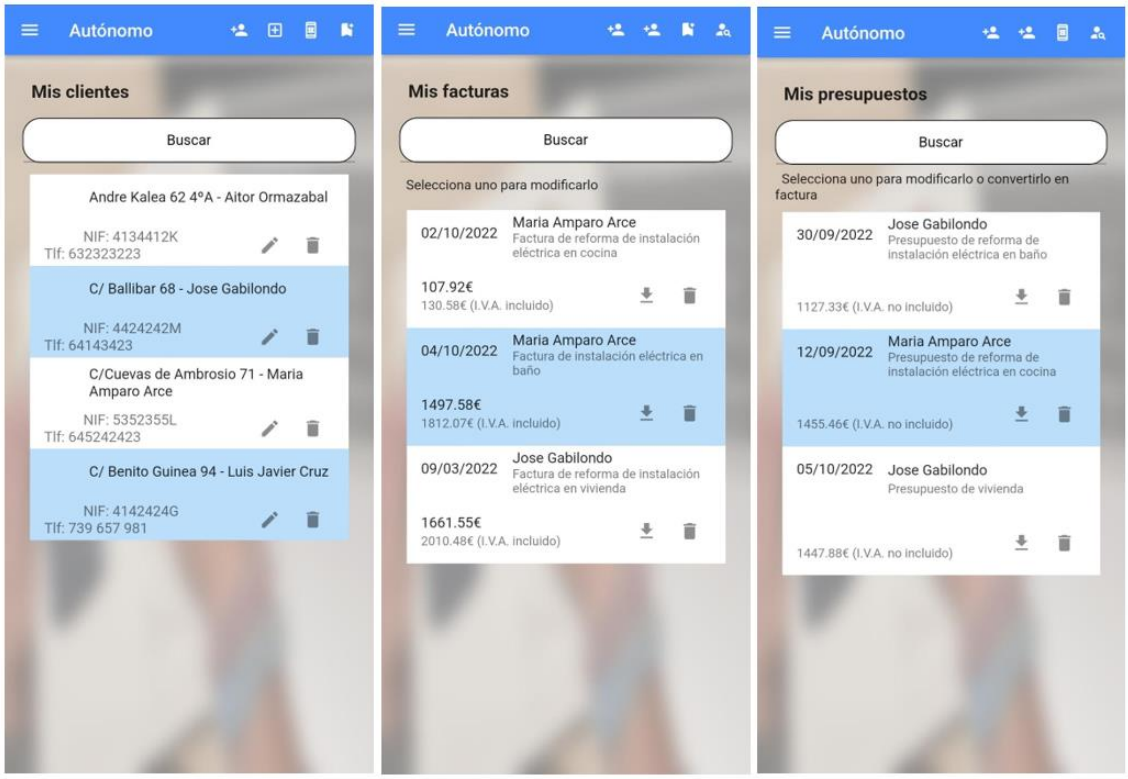


Figura 66. Interfaz móvil: Gestionar clientes, facturas y presupuestos

Anexo F: Manual de usuario

F.1 Descarga

La página web será accesible desde el siguiente enlace:

https://facturas-y.web.app/ [21]

La aplicación de Android podrá descargarse en la siguiente dirección:

Autónomo APK [22]

F.2 Funciones básicas

F.2.1 Registro (Figura 67)

Pulsa el botón “Regístrate” situado en la parte inferior de la página de inicio de sesión. Se abrirá la página de registro para que introduzcas tus datos, será necesario que selecciones la imagen de tu logo y completes todos los campos, la contraseña deberá tener mínimo 6 caracteres.

Cuando tengas todos los datos introducidos pulsa ‘Registrar’, si no has cumplido alguna de las condiciones anteriores verás un mensaje indicándolo, sino se mostrará un mensaje indicando que ya estás registrado (Figura 68). El siguiente paso será iniciar sesión con tu nueva cuenta.

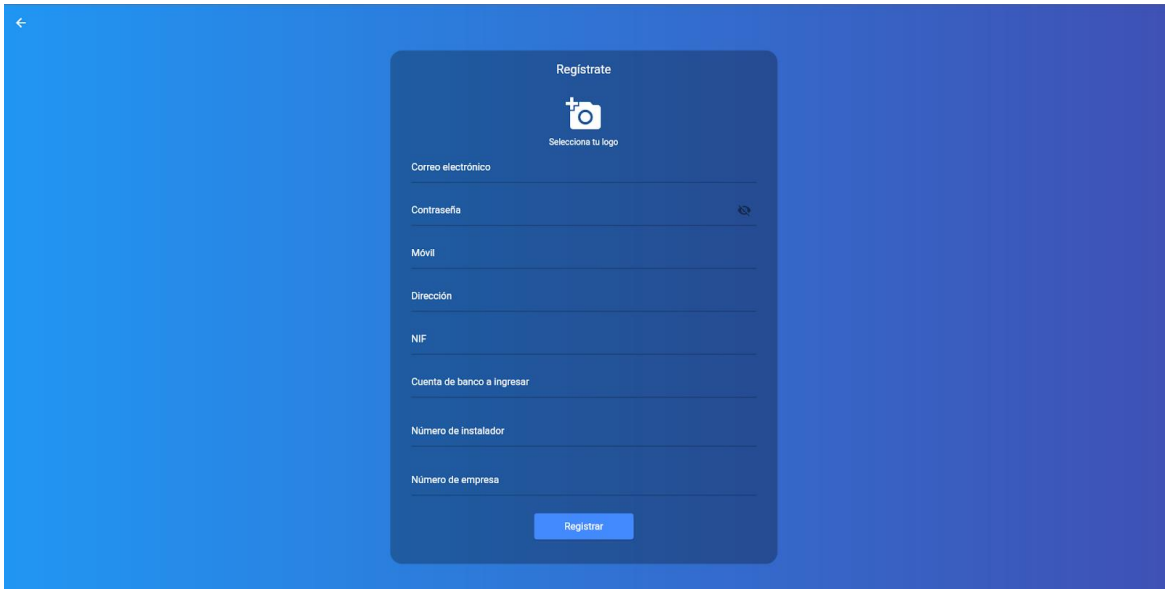


Figura 67. Interfaz registro

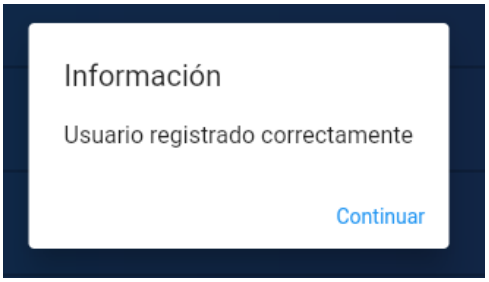


Figura 68. Mensaje registro correcto

F.2.2 Iniciar sesión (Figura 69)

Si ya estás registrado, introduce tu correo y contraseña y pulsa 'Iniciar sesión'. Si los datos son correctos se abrirá el menú principal.

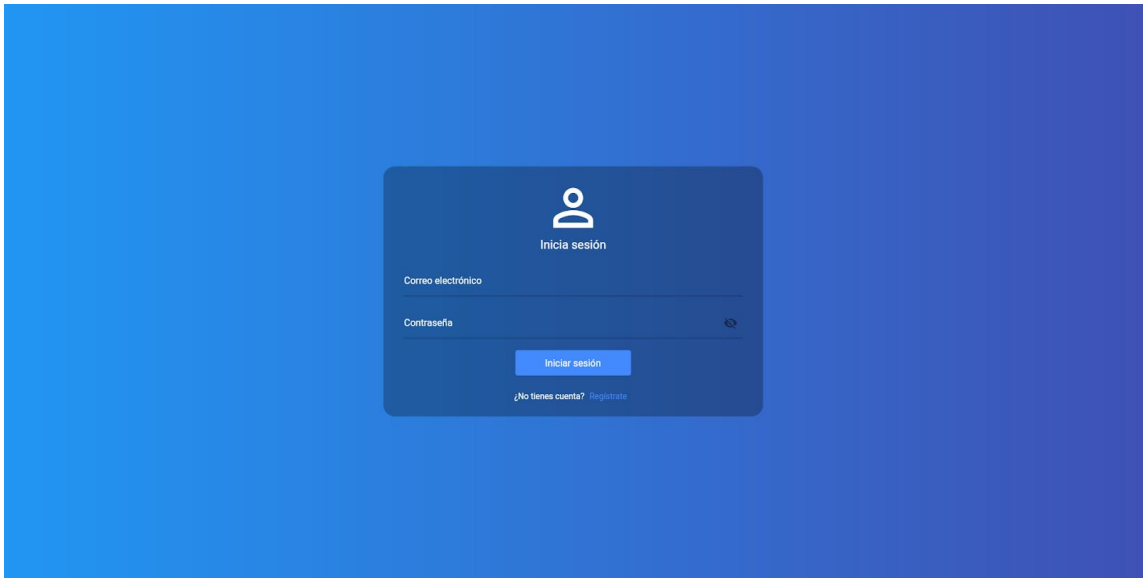


Figura 69. Interfaz inicio de sesión

F.2.3 Menú de selección (Figura 70)

Desde este menú, que se mostrará cuando inicies sesión, podrás navegar a cualquier lugar de la aplicación. En la siguiente imagen se pueden ver las opciones:

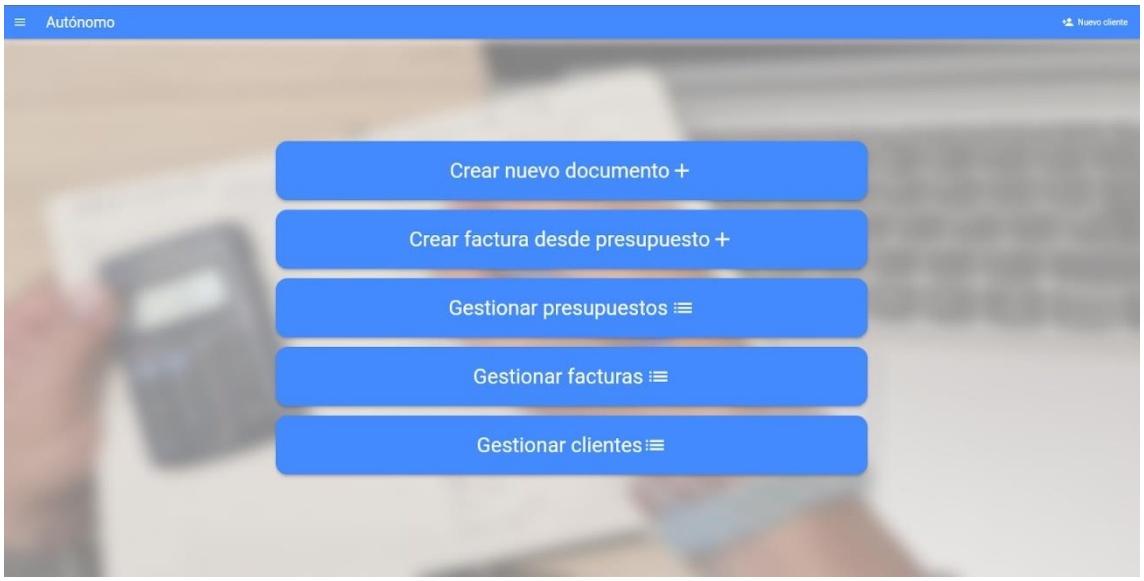


Figura 70. Interfaz menú de selección

F.2.4 Menú lateral

Desde este menú podrás cerrar sesión o navegar a cualquier parte del programa.

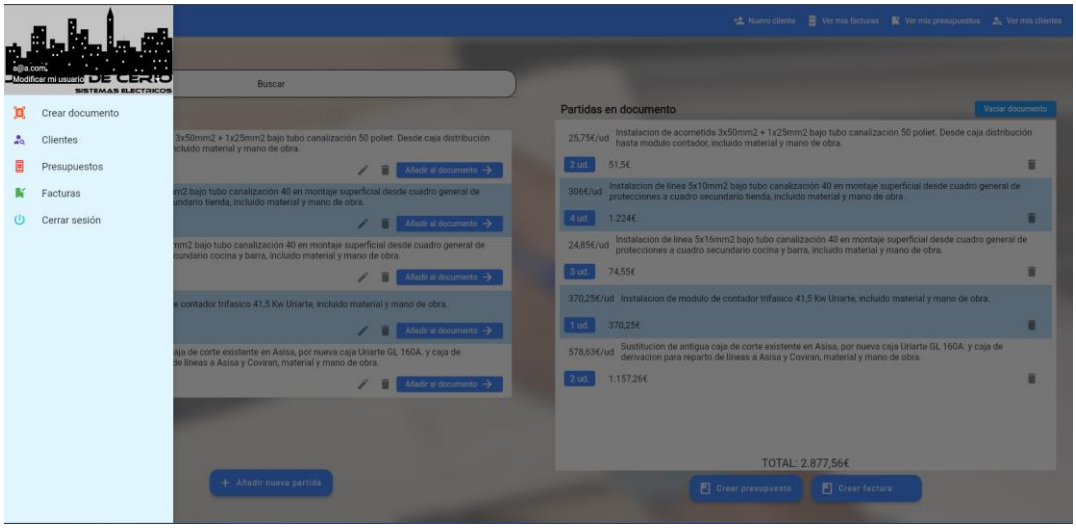


Figura 71. Interfaz menú lateral

F.2.5 Cerrar sesión

Abre el menú lateral y pulsa cerrar sesión.

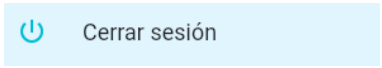


Figura 72. Interfaz botón cerrar sesión

F.3 Creación del documento

F.3.1 Página para añadir las partidas al documento (Figura 73)

En la Figura 73 se muestra la página para añadir las partidas al documento, en los siguientes subapartados se explicarán sus funcionalidades.

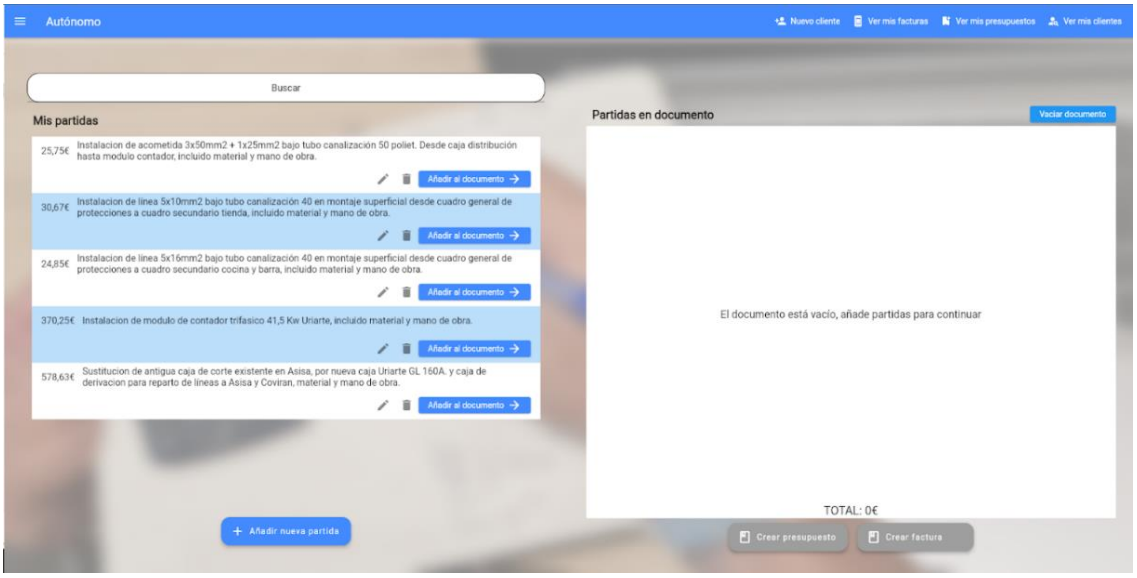


Figura 73. Interfaz menú inicial de creación de documento

F.3.1.1 Insertar partidas (Figura 74)

Pulsa el botón ‘Añadir nueva partida’ e introduce el concepto y precio. Pulsa guardar y la partida se añadirá a la lista ‘Mis partidas’.

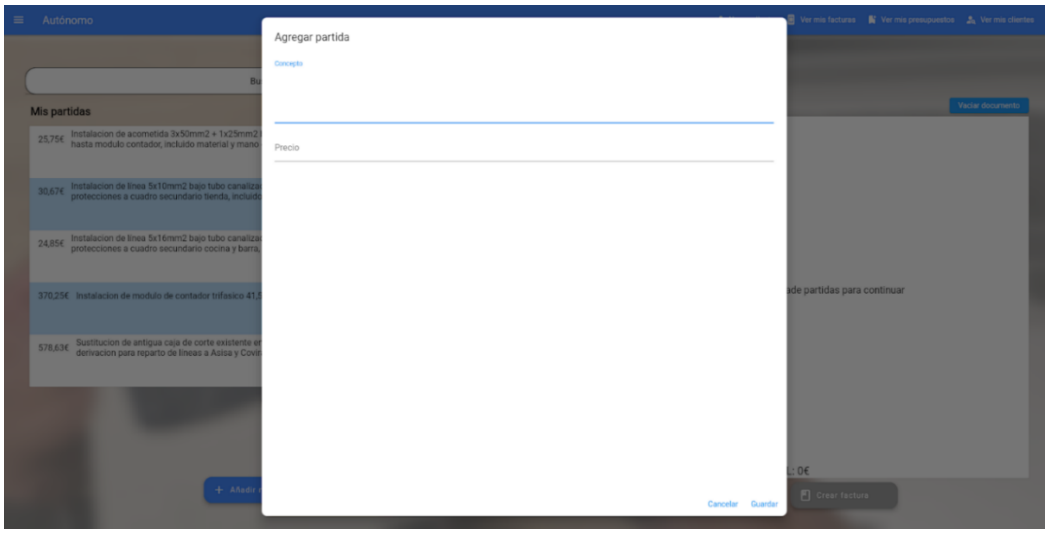


Figura 74. Interfaz agregar partida

F.3.1.2 Editar, eliminar partidas (¡Error! No se encuentra el origen de la referencia.)

Para editar una partida pulsa sobre el icono del lápiz (**¡Error! No se encuentra el origen de la referencia.**).

Se te abrirá un menú similar al de la Figura 74 con los datos actuales, modifícalos y pulsa ‘Actualizar’. Si todo ha ido bien se mostrará un mensaje indicándolo.

Para eliminar una partida pulsa sobre el icono de la papelera de la partida y confirma (Figura 75).

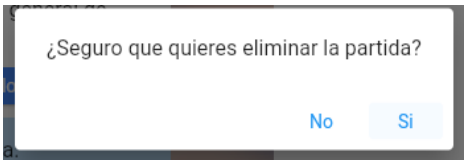


Figura 75. Interfaz mensaje de confirmación eliminar partida

F.3.1.3 Añadir a documento (Figura 76)

Pulsa el botón ‘Añadir al documento’ de la partida que quieres añadir (Figura 76).

Indica la cantidad, pulsa ‘Guardar’ (Figura 77) y se añadirá a la lista del documento. Podrás ver el total del documento y el de cada partida.

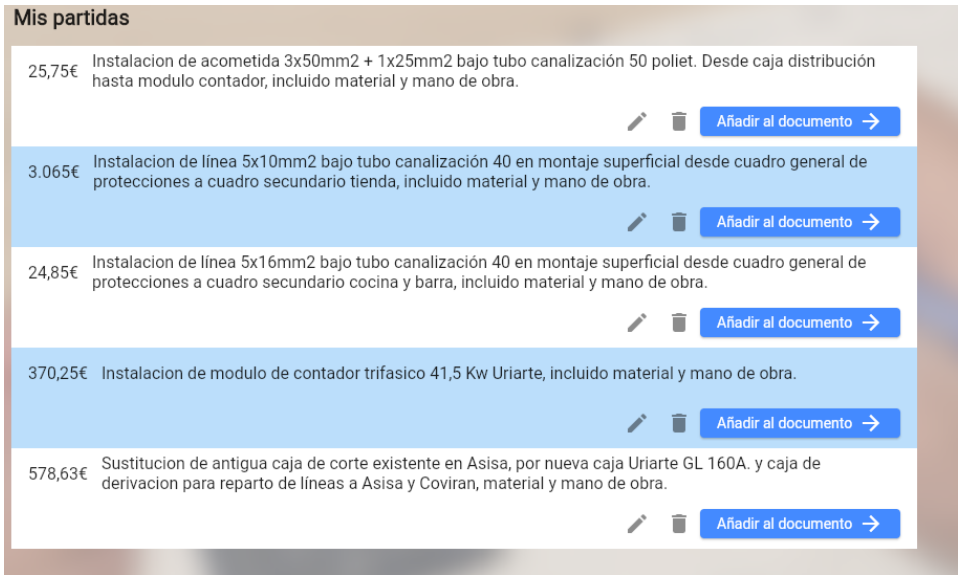


Figura 76. Interfaz lista de partidas

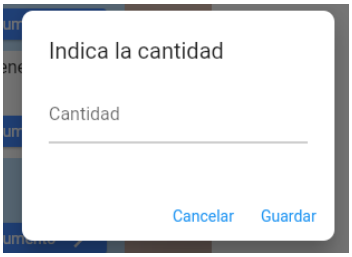


Figura 77. Interfaz indicar cantidad

F.3.1.4 Añadir clientes

Añade los datos de un cliente para poder seleccionarlo después al introducirlo a la factura. Pulsa sobre 'Nuevo cliente' (Figura 78) y escribe sus datos en el formulario (Añadir clientes

). Pulsa guardar.

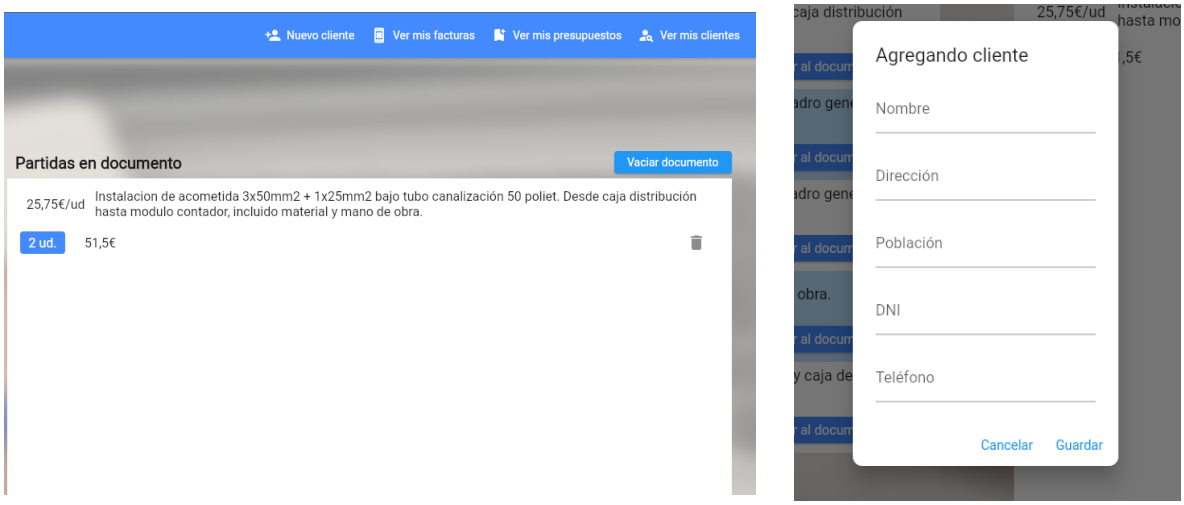


Figura 78. Añadir clientes

F.3.1.5 Vaciar documento

Si quieres eliminar todas las partidas de documento pulsa ‘Vaciar documento’ y confirma que quieres vaciarlo.

F.3.1.6 Selección factura/presupuesto (Figura 79)

Una vez tengas todas las partidas que añadirás al documento seleccionadas pulsa sobre ‘Crear presupuesto’ o ‘Crear factura’. Se te abrirá la página donde deberás introducir los datos.

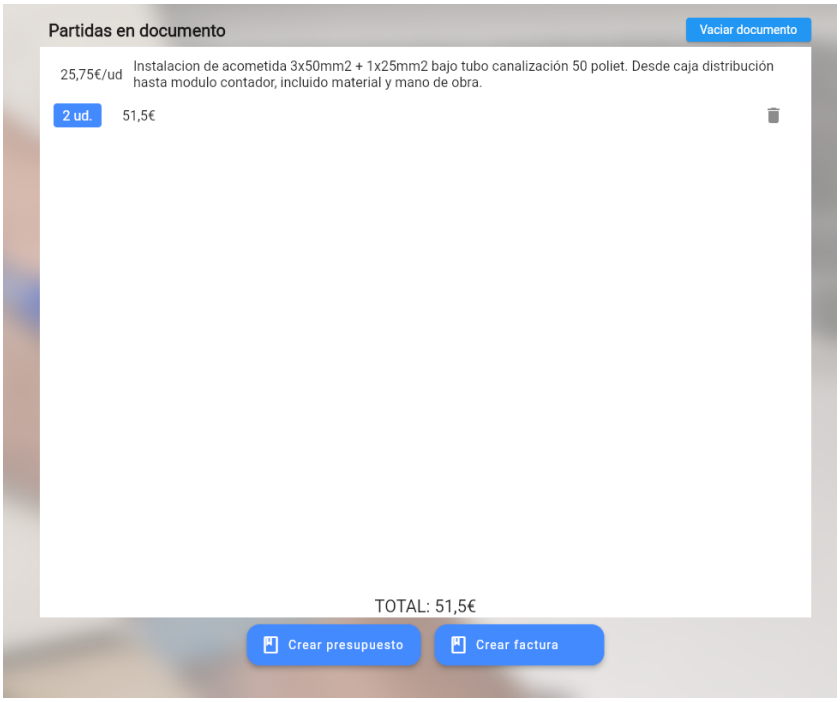


Figura 79. Interfaz elección presupuesto/factura

F.3.1.7 Insertar datos documento (Figura 80)

Introduce los datos solicitados, selecciona un cliente y pulsa ‘Crear presupuesto’ o ‘Crear factura’. Se te abrirá un PDF en una nueva ventana.

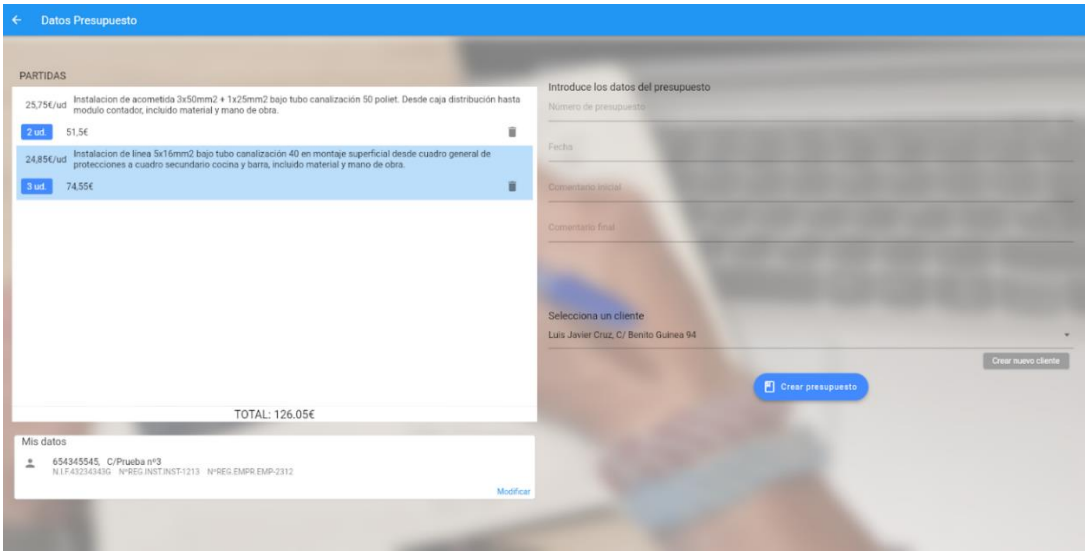


Figura 80. Interfaz insertar datos documento

F.3.1.8 PDF

Obtendrás un PDF similar al que se muestra a continuación:

J.R.DIAZ DE CERIO
SISTEMAS ELECTRICOS

654345545
C/Prueba nº3

CLIENTE
Aitor Ormazabal

DIRECCIÓN
Andre Kalea 62 4ºA

POBLACIÓN
Hernani

NºFactura
fac2022-102
NIF
4134412K
FECHA
10-09-2022
TLF
632323223

N.I.F. 43234343G NºREG.INST.INST-1213 NºREG.EMPR.EMP-2312

Factura de instalación en vivienda

CANT	CONCEPTO	PRECIO	IMPORTE
2ud.	Instalacion de acometida 3x50mm2 + 1x25mm2 bajo tubo canalización 50 poliet. Desde caja distribución hasta modulo contador, incluido material y mano de obra.	25.75€	51.50€
4ud.	Instalacion de línea 5x10mm2 bajo tubo canalización 40 en montaje superficial desde cuadro general de protecciones a cuadro secundario tienda, incluido material y mano de obra.	306€	1224.00€
3ud.	Instalacion de línea 5x16mm2 bajo tubo canalización 40 en montaje superficial desde cuadro general de protecciones a cuadro secundario cocina y barra, incluido material y mano de obra.	24.85€	74.55€
1ud.	Instalacion de modulo de contador trifasico 41,5 Kw Uriarte, incluido material y mano de obra.	370.25€	370.25€
2ud.	Sustitucion de antigua caja de corte existente en Asisa, por nueva caja Uriarte GL 160A. y caja de derivacion para reparto de líneas a Asisa y Coviran, material y mano de obra.	578.63€	1157.26€
	SUMA		**2877.56€**
	+21% I.V.A.		**604.29€**
	TOTAL FACTURA		**3481.85€**

A ingresar en C.C. IBAN 3424 4342 3423 4342, Banco Santander

Figura 81. PDF ejemplo factura

F.2.3 Gestionar datos

F.2.3.1 Gestionar clientes

Pulsa 'Ver mis clientes' (Figura 82) de la barra superior o 'Gestionar clientes' en el menú de selección inicial. Podrás buscarlos por cualquier dato, nombre, dirección, etc.

Desde esta página podrás modificarlos o eliminarlos. Si el cliente tiene algún documento asociado no te permitirá eliminarlo y te mostrará un mensaje (Figura 83).

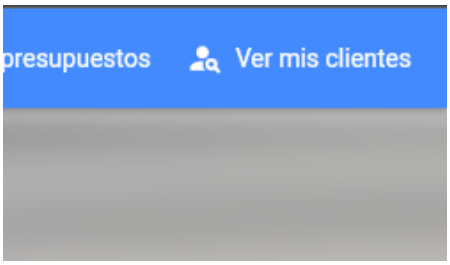


Figura 82. Interfaz botón ver clientes

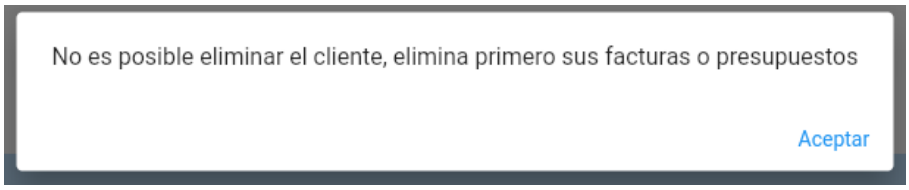


Figura 83. Interfaz no es posible eliminar cliente

F.2.3.2 Gestionar facturas (Figura 84)

Pulsa 'Ver mis facturas' de la barra superior o 'Gestionar facturas' en el menú de selección inicial.

Para descargar la factura pulsa sobre el icono de descargar, para eliminarla sobre el de la papelera y confirma que quieres eliminarla (Figura 85).

Si lo que quieres es modificarla o crear otra factura a partir de ella pulsa sobre la factura y se abrirá el menú de creación de documentos principal con todas las partidas de dicha factura.

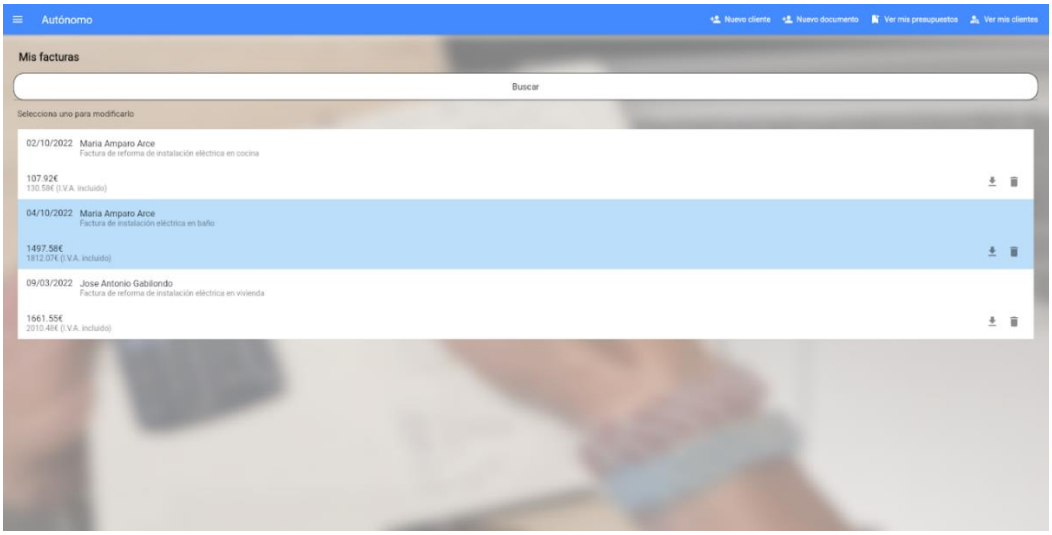


Figura 84. Interfaz gestionar facturas

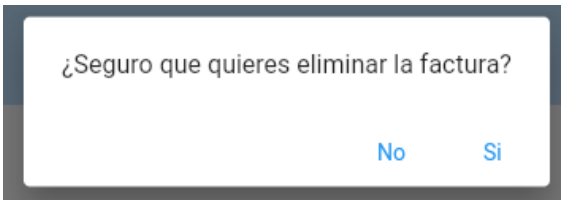


Figura 85. Interfaz eliminar factura

F.2.3.3 Gestionar presupuestos (Figura 86)

Pulsa 'Ver mis presupuestos' de la barra superior o 'Gestionar presupuestos' en el menú de selección inicial.

Para descargar el presupuesto pulsa sobre el icono de descargar, para eliminarlo sobre el de la papelera y confirma que quieres eliminarlo.

Si lo que quieres es modificarlo o crear una factura a partir de ella pulsa sobre el presupuesto y se abrirá el menú de creación de documentos principal con todas las partidas de dicho presupuesto, podrás modificar las partidas que quieras y convertirlo en factura en segundos.

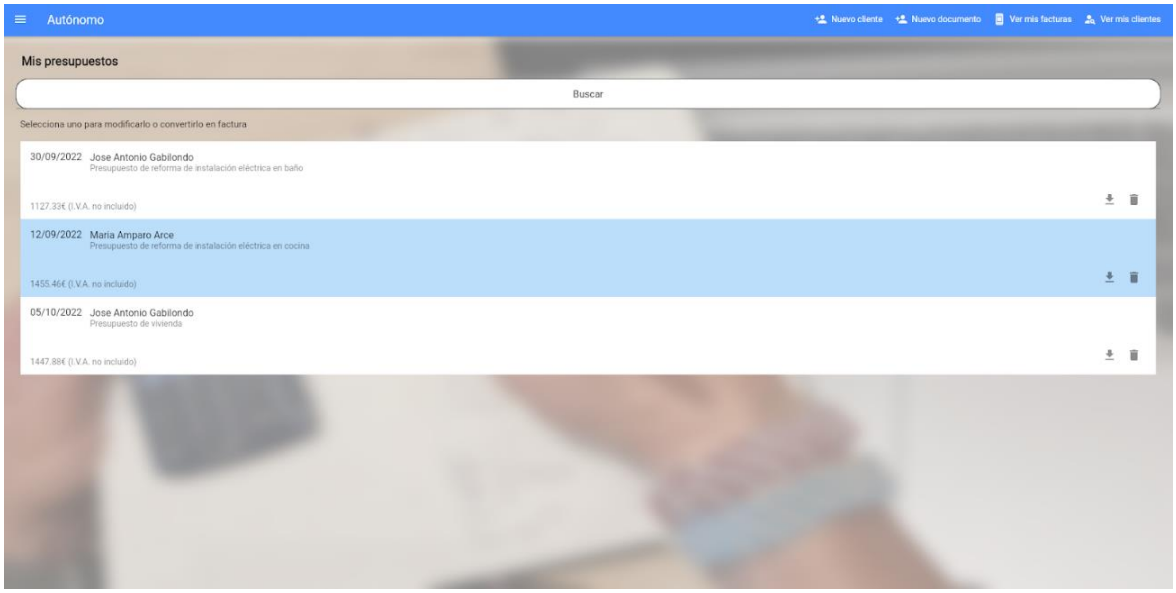


Figura 86. Interfaz gestionar presupuestos