PROYECTO FINAL

DE CURSO

JORGE FULLANA ESTELA 2DAM

Índice

Índice de figuras

1 INTRODUCCIÓN

1.1 Resumen

En la actualidad, la **gestión eficiente** de documentos contables como facturas y albaranes es esencial para la operatividad de las empresas. Hoy en día las grandes empresas con recursos ya tienen optimizado estas tareas repetitivas con programas, en cambio, una persona que está empezando en el mundo laboral como autónomo o que no tiene todos los recursos suele hacer estos trabajos con Excel o delegándolo a un tercero, con el objetivo de **simplificar** este proceso, se propone el desarrollo de una **aplicación móvil** para dispositivos Android que permita la automatización de estas tareas. Esta aplicación, diseñada con un enfoque moderno, busca ofrecer **comodidad y facilidad** de uso a los usuarios, agilizando la generación de documentos contables y mejorando la eficiencia en la gestión empresarial.

En este documento consta de un profundo análisis en busca de su **publico objetivo** donde se prioriza en la app, que en 2023 constaría de **2.020.919 personas**, donde la mayoría de los autónomos representarían el **sector de servicios**.

Además se crearía una encuesta propia para el proyecto que decantaría el diseño de la app, simple y poco cargado de elementos. Este diseño estaría compuesto por los componentes *Open-Source* de **Google Material Desing**, ya que está normalizado por los usuarios de Androird y es personalizable en grandes aspectos. El patrón de diseño que se utilizó es **MVVM** (Model View ViewModel) para desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación

También se crearía un **DAFO** en busca de las debilidades, amenazas, fortalezas y oportunidades, donde las corresponden:

- **Debilidades**: Dependencia de la tecnología, Complejidad de integración, Riesgo de errores, Conexión API.
- **Amenazas**: Competencia, Cambios regulatorios, Ciberseguridad.
- **Fortalezas**: Eficiencia, Precisión, Escalabilidad, Comodidad.
- **Oportunidades**: Mercado en crecimiento, Globalización, Colaboraciones estratégicas.

Los requisitos de la app están establecidos por las **normas ISO**: ISO/IEC **27001**, ISO/IEC **25010**, ISO/IEC **20000**, ISO/IEC **29151**.

En la planificación se calculó lo que costaría el mismo proyecto en un caso ceñido a la realidad, donde se llegaría a una conclusión de un coste de entre **33.235 € - 34.375 €** solo por trabajadores y componentes de trabajo. Como el tiempo en un proyecto es una de las cosas más importantes, se planificó lo que debería durar el proyecto y se segmento por tareas en un **diagrama de Gannt**. Los **riesgos** también es una parte importante de la planificación donde se pudo detectar lo siguiente: Dependencia en herramientas y tecnologías específicas, Dificultades en la colaboración y comunicación, Capacidades técnicas insuficientes del equipo, Errores de autenticación y funcionalidad, Errores no detectados en la fase de pruebas, Demora en la preparación de la campaña de marketing.

Para finalizar el documento se describe el recorrido del desarrollo del software de la app y la explicación de la implementación. Terminando con las conclusiones se aprecia una autocrítica al proyecto y a la programación en general, donde se intenta plasmar el esfuerzo dedicado al proyecto y la capacidad infinita que tiene la programación para seguir mejorándose/ampliándose. Donde las propuestas de mejoras serían: mejorar la planificación del proyecto, probar que las funcionalidades sean compatibles con el entorno de desarrollo y en general mejorar aspectos de específicos del proyecto.

1.2 Justificación del proyecto

La justificación de este proyecto se fundamenta en la **creciente necesidad** de optimizar los procesos administrativos en las empresas, especialmente en lo que respecta a la generación de facturas y albaranes, ya que la mayoría de **autónomos** (a persona física que realiza una actividad económica y lo hace de forma habitual, directa y, como no, por cuenta propia, recibiendo una remuneración por ella) suelen dejar estos tediosos trabajos a terceros debido a que la gestión manual de estos documentos puede resultar tediosa y propensa a errores, lo que afecta la eficiencia operativa y la calidad de la información financiera. Por tanto, el desarrollo de una aplicación móvil que automatice estas tareas se presenta como una solución que contribuirá a mejorar la productividad y la precisión en la gestión contable de las empresas.

2 ANÁLISIS

El Instituto nacional de Estadística documento un censo de la población residente en España. El 1 de enero de 2023, se situó en 48.085.361 habitantes de esta cantidad son **31.343.030** (65.18 %) de 16 a 64 años. Este porcentaje de la población tienen la posibilidad de trabajar todos los mayores de 18 años, los menores de 18 y mayores de 16 legalmente emancipados o con consentimiento de sus padres o tutores y los extranjeros según la legislación que les es aplicable.

En el contexto empresarial español, los autónomos representan un segmento significativo y dinámico de la fuerza laboral, donde nuestra app tiene su público objetivo.

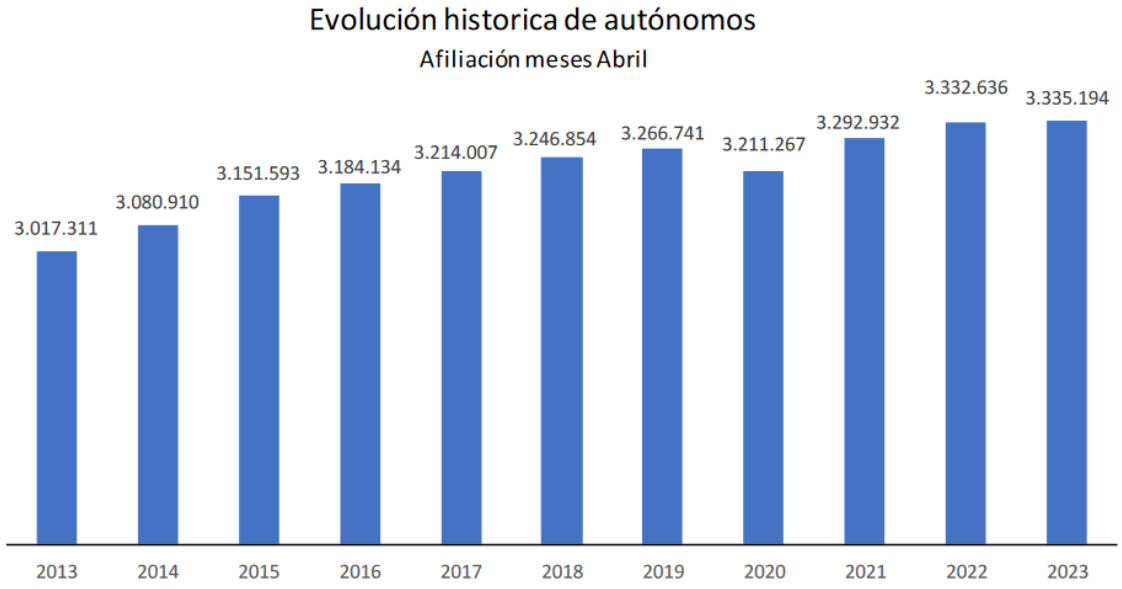

Figura 1: Gráfico cantidad de autónomos 2013-2023, de Gobierno de España, 2023. *https://www.mites.gob.es/ficheros/ministerio/sec_trabajo/autonomos/economia-soc/NoticiasDoc/NoticiasPortada/2023/Nota_Afiliacion-trabajo-autonomo_Abril.pdf.*

En la gráfica podemos contemplar que existe un **crecimiento** de la cantidad de autónomos, siendo esto beneficiente para la aplicación, ya que los posibles usuarios totales de la aplicación está en aumento.

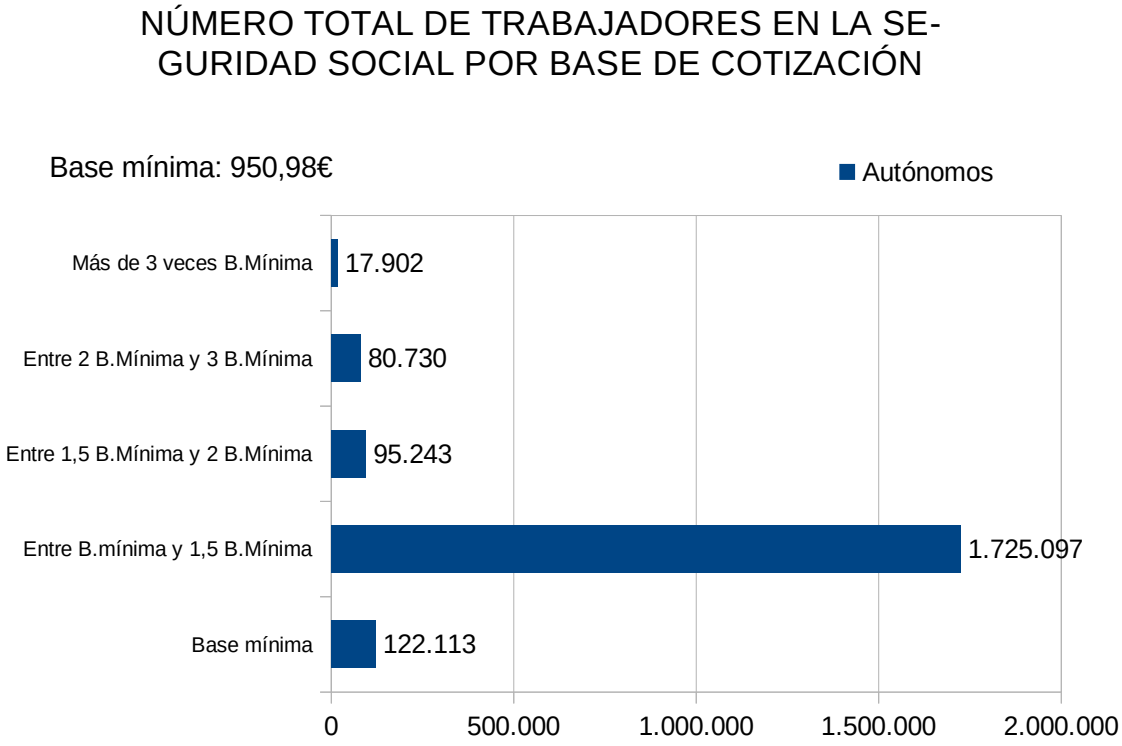

Figura 2: Elaboración propia.

La aplicación está pensada para la **comodidad del usuario** y no tanto para tener una estructura elaborada como un ERP, por esto se podría descartar a los autónomos que generan un numero grande de capital, ya que lo más probable es que estos tengan ya incorporado en sus empresas programas más completos que una de sus tantas funciones conste en gestión de facturas.

En beneficio de la aplicación, los autónomos que genera un capital 3 veces mayor a la base mínima representa un **pequeño porcentaje del total (0,88%)**.

Otra forma de segmentar y especificar a un público de autónomos es por su sector.

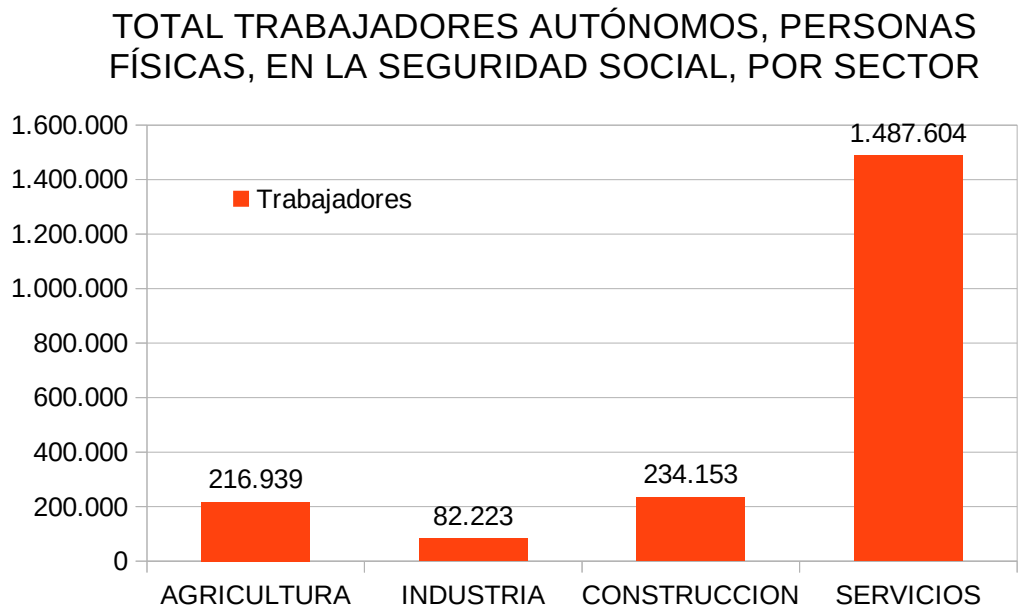

Figura 3: Elaboración propia

La aplicación debería **priorizar** las **necesidades** de los autónomos del sector de **servicios** ya que está la mayor parte de los trabajadores (73,61%).

Para terminar con el análisis se propuso hacer una encuesta a la población de Castellón de la Plana, esto con el fin de conocer las ideas y saber directamente la opinión popular.

Donde las preguntas de la encuesta corresponden:

- ¿Eres autónomo?
- ¿Realizas facturas?
- Si no hace facturas -> ¿Las delegas a un tercero?
- Si no hace facturas ¿Porque?
  a) Porque en mi trabajo se delega
  b) No me gusta hacerlas / No se hacerlas
  c) Para reducir tiempo
  d) otros
- ¿Con que frecuencia?
- diariamente / semanalmente / mensualmente
- ¿Cuantas?
- ¿Utilizas algún programa para facturar? / ¿harías las facturas tu mismo si hubiera una app para automatizarlas?
- ¿Con que te resultaría más cómodo hacer una factura móvil/Ordenador?
- ¿Porque crees esto?
- ¿Que funcionalidad te gustaría que tenga una app para la facturación?

Seccionando en grupos encuestados estos fueron los resultados:

Sector Mercadillo:

Popularmente no están interesado en una app para facturación, ya que esta es mediante tickets y no los frecuentan, ya que solo los hacen cuando los solicitan los clientes.

Comercios de comunidad asiática:

Aunque fue uno de los sectores que más fue preguntado, ninguno se presto a resolver la encuesta.

Peatones aleatorios del centro de Castellón de la Plana Autónomos:

Está fue la parte más dificultosa ya que cada 30 personas preguntadas 1 era autónoma, pero con mejores resultados desde el punto de vista de la app.

En general la mayoría de la población estaba de acuerdo con una aplicación para el móvil de facturación, unas de su mayores razones por la comodidad de poder llevar el móvil a cualquier sitio. En el tema de funcionalidad para la app, las personas encuestadas destacaron dos puntos muy importantes, sencillez y que se guarden los datos de los clientes, además hubo casos específicos que también aportaron buenas ideas, como poner fecha limite para pagar la factura sino se le sumaría un costo adicional (aunque esto sería para una app más de gestión).

Por otro lado había gente que no utilizaría la app con sus propias razones, la razón más común era que ya estaban acostumbrados a usar el ordenador (o las hacían a mano),pero no es todo malo, ya que la mayoría de este público rondaba una edad mayor, esto quiere decir que las nuevas generaciones de autónomos tienen y tendrán una mejor visión de la app.

2.1 DAFO

Figura 4: Imagen representación DAFO. https://dafo.ipyme.org/Content/images/grafico-DAFO.png

Debilidades:

- **Dependencia de la tecnología**: La aplicación funcionará a través del móvil del usuario y será necesario que este tenga las capacidades mínimas para usar la app (memoria , cpu, versión de Android, etc).

- **Complejidad de integración**: Podría resultar complicado integrar la aplicación con los sistemas de contabilidad existentes en las empresas, lo que podría disminuir su adopción.

- **Riesgo de errores**: La automatización puede llevar a errores si no se configura correctamente, lo que podría generar confusiones en los registros financieros.

- **Conexión API:** Al funcionar con Goolge es dependiente a una conexión a internet para su completo funcionamiento.

Amenazas:

- **Competencia**: La presencia de otras aplicaciones de automatización de facturas podría reducir la cuota de mercado de la aplicación.

- **Cambios regulatorios**: Cambios en las normativas fiscales podrían requerir actualizaciones frecuentes en la aplicación para mantener la conformidad legal.

- **Ciberseguridad**: La amenaza de ciberataques y robos de datos podría socavar la confianza de los usuarios en la seguridad de la aplicación.

Fortalezas:

- **Eficiencia**: La automatización de procesos puede aumentar la eficiencia al reducir el tiempo y los recursos necesarios para procesar las facturas.

- **Precisión**: Al minimizar la entrada manual de datos, la aplicación puede ayudar a reducir los errores en la facturación y en los registros financieros.

- **Escalabilidad**: La aplicación puede ser escalable para adaptarse a las necesidades cambiantes de las empresas a medida que crecen.

- **Comodidad**: Al funcionar desde el móviles da la versatilidad de usar en cualquier sitio en cualquier situación.

Oportunidades:

- **Mercado en crecimiento**: Con la creciente digitalización de los procesos empresariales, existe una creciente demanda de soluciones de automatización de facturas.

- **Globalización**: La aplicación podría expandirse a mercados internacionales, aprovechando la creciente globalización de los negocios.

- **Colaboraciones estratégicas**: Asociarse con proveedores de software de contabilidad u otros actores en el ecosistema empresarial podría ampliar el alcance y la funcionalidad de la aplicación.

2.2 Necesidades

La necesidad de automatizar la generación de facturas y albaranes surge de la creciente demanda de agilidad, comodidad y precisión en los procesos administrativos empresariales. La gestión manual de estos documentos resulta ineficiente, tediosos y propensa a errores, lo que afecta la operatividad y la fiabilidad de la información financiera.

Por esto nace la **necesidad** de una app que cumpla la demanda y criterios, para satisfacer la falta de un programa que cumpla estos requisitos.

2.3 Objetivos

El principal objetivo del proyecto es optimizar los procesos relacionados con la generación de facturas y albaranes, reduciendo el tiempo y los recursos necesarios para llevar a cabo estas tareas. Además, se busca mejorar la calidad y la fiabilidad de la información financiera, así como aumentar la accesibilidad y la flexibilidad en la gestión de documentos contables.

2.4 Requisitos

Los requisitos vienen determinados por las normativas ISO en el ámbito IT (tecnologías de la información y de la comunicación).

ISO/IEC 27001: Objetivo de Seguridad de la Información

- Establecer, implementar, mantener y mejorar un Sistema de Gestión de Seguridad de la Información (SGSI).
- Mantener la confidencialidad, integridad y disponibilidad de la información.
- Identificar, evaluar y tratar los riesgos de seguridad de la información.
- Implementar controles para mitigar los riesgos identificados.

ISO/IEC 25010: Objetivo de Calidad del Producto de Software

- Evaluar y mejorar la calidad del producto de software.
- Asegurar la funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad, portabilidad y seguridad del software.
- Proporcionar un marco para la evaluación de la calidad del producto de software.

ISO/IEC 29151: Objetivo de Protección de la Privacidad

- Establecer principios y medidas para proteger la privacidad de la información personal.
- Obtener el consentimiento del usuario para la recopilación y el uso de datos personales.
- Limitar la recopilación y el uso de datos personales a lo necesario para fines específicos.
- Implementar controles para proteger la privacidad de los usuarios de servicios en la nube.

ISO/IEC 20000: Objetivo de Gestión de Servicios de TI

- Establecer, implementar, mantener y mejorar un Sistema de Gestión de Servicios de TI (SGSTI).
- Entregar servicios de TI eficaces y eficientes.
- Planificar, proveer, controlar y mejorar los servicios de TI.
- Garantizar la disponibilidad de los servicios de TI necesarios para el funcionamiento de las aplicaciones y sistemas de la organización.

3 DISEÑO

El diseño es el proceso de configuración mental preliminar, o prefiguración, que precede a la búsqueda de soluciones para que un producto resulte útil y atractivo.

En aplicaciones móviles consiste en estructurar la navegación de la aplicación a partir de la lista funcionalidades de la solución y definir las directrices visuales que se aplicarán a los distintos elementos y pantallas.

Por lo tanto garantiza, por un lado que la información está bien estructurada y comprueba que la navegación y funcionalidad de la app fluye correctamente, asegurando una experiencia de usuario adecuada. Y por otro, se encarga de que el *look and feel* de la app vaya acorde con la guía de estilo de la marca y sea atractivo para el tipo de usuario al que se dirige.

3.1 Estudio de mercado

En general las app que rondan en el mercado aparte de hacer facturas, tienen funcionalidades de gestión, esto perjudica a su diseño al tener que sobrecargar la pantalla con opciones y pestañas.

Como se vio en el análisis la personas buscan la sencillez algo simple de usar y de visualizar, por eso en la app se buscara simplificar lo mayor posible el diseño y que sea lo más intuitivo posible.

3.2 Arquitectura de la solución

Para arquitectura se utilizó el MVVM porque ofrece una clara separación de responsabilidades, facilita las pruebas unitarias, mejora la mantenibilidad y escalabilidad del código, y permite una mejor gestión del ciclo de vida en Android.

Separación de responsabilidades

- Modelo (Model): Maneja los datos y la lógica de negocio.
- Vista (View): Presenta la información y recibe interacciones del usuario.
- VistaModelo (ViewModel): Intermediario entre la Vista y el Modelo, manejando la lógica de presentación.

Facilidad de prueba

- La separación clara facilita escribir pruebas unitarias para el Modelo y el ViewModel sin depender de la interfaz de usuario.

Mantenibilidad y escalabilidad

- MVVM organiza el código de manera que es fácil de mantener y expandir. Los componentes desacoplados permiten realizar cambios sin afectar otras partes del sistema.

Reutilización de código

- El ViewModel puede ser reutilizado en diferentes Vistas, promoviendo la reutilización de la lógica de presentación.

Data Binding

- Android Data Binding se integra perfectamente con MVVM, permitiendo que la Vista se actualice automáticamente con cambios en el ViewModel.

Mejor gestión del ciclo de vida

- El ViewModel sobrevive a cambios de configuración como rotaciones de pantalla, manteniendo los datos y la lógica de negocio intactos.

3.3 Diseño de las interfaces

El diseño será marcado por los componentes de Material Design que son los componentes *Open-Source* de Google para el diseño y construcción de la app.

Aquí hay algunas razones por las que utilizar Material Design de Google puede ser beneficioso:

- **Consistencia**: Material Design proporciona pautas claras y coherentes para el diseño de interfaces de usuario en todas las plataformas, lo que garantiza una experiencia consistente para los usuarios, independientemente del dispositivo que estén utilizando.

- **Flexibilidad:** Aunque ofrece una estructura y directrices sólidas, Material Design también permite cierta flexibilidad en términos de personalización y adaptación a las necesidades específicas de cada aplicación.

- **Facilidad de uso:** Las pautas de Material Design están diseñadas para mejorar la usabilidad y la accesibilidad, lo que facilita a los usuarios comprender la interfaz y navegar por ella de manera intuitiva.

- **Optimización para dispositivos móviles**: Material Design se basa en principios de diseño responsivo, lo que significa que las interfaces creadas siguiendo estas pautas se adaptan bien a una variedad de tamaños de pantalla y dispositivos, incluidos teléfonos móviles y tabletas.

- **Actualizaciones regulares:** Google actualiza regularmente las pautas de Material Design para mantenerse al día con las tendencias de diseño emergentes, así como para mejorar la experiencia del

usuario y abordar nuevas necesidades en el panorama digital en constante cambio.

- **Comunidad y recursos:** Dado que Material Design es un marco de diseño ampliamente adoptado, existe una gran comunidad de diseñadores y desarrolladores que comparten recursos, tutoriales y mejores prácticas, lo que facilita el proceso de diseño y desarrollo.

3.4 Anatomía de los componentes

Button

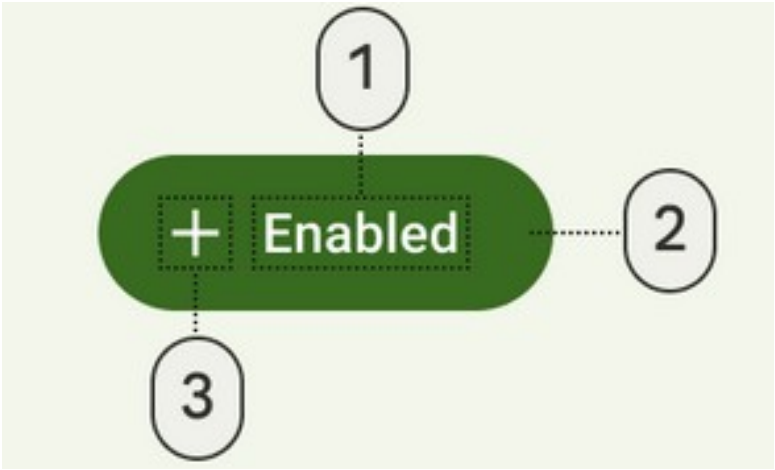

Figura 5: Buttons. https://m3.material.io/components/buttons/guidelines

1. Label text.
2. Container.
3. Icon (Opcional).

Progress indicator

Figura 6: Progress indicator.
https://m3.material.io/components/progress-indicators/guidelines

1. Indicator.
2. Track.
3. Stop indicator.

Outlined text field

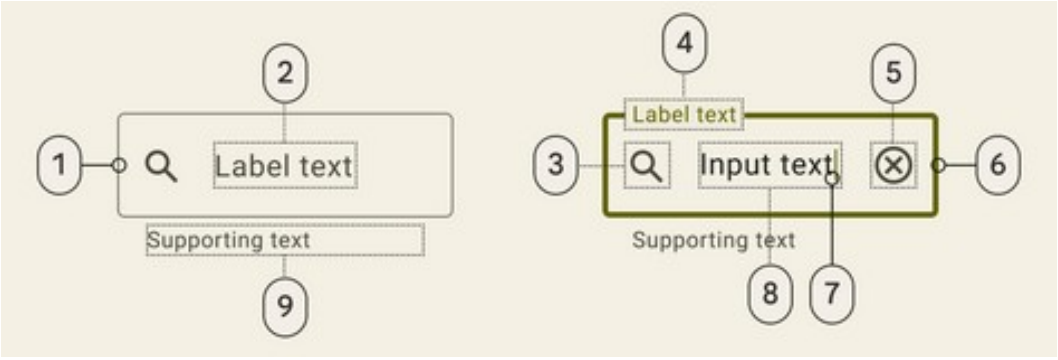

Figura 7: Campo de texto anatomía. https://m3.material.io/components/text-fields/guidelines°

1. Container outline.
2. Label text (Vacio).
3. Leading Icon (Opcional).
4. Populated (Opcional).
5. Trailing icon (Opcional).
6. Container Outline.

7. Caret.
8. Input text.
9. Supporting text (Opcional).

Navigation drawer

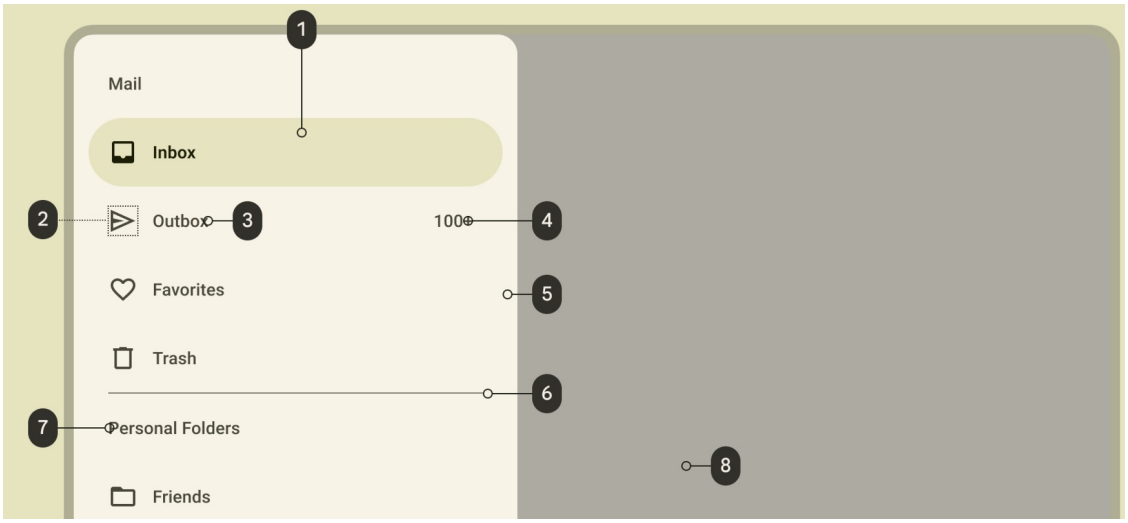

Figura 8: Navigation drawer. https://m3.material.io/components/navigation-drawer/guidelines

1. Active indicator.
2. Icon.
3. Label.
4. Badge label.
5. Sheet.
6. Divider.
7. Section label (opcional).
8. Scrim.

Dialogs

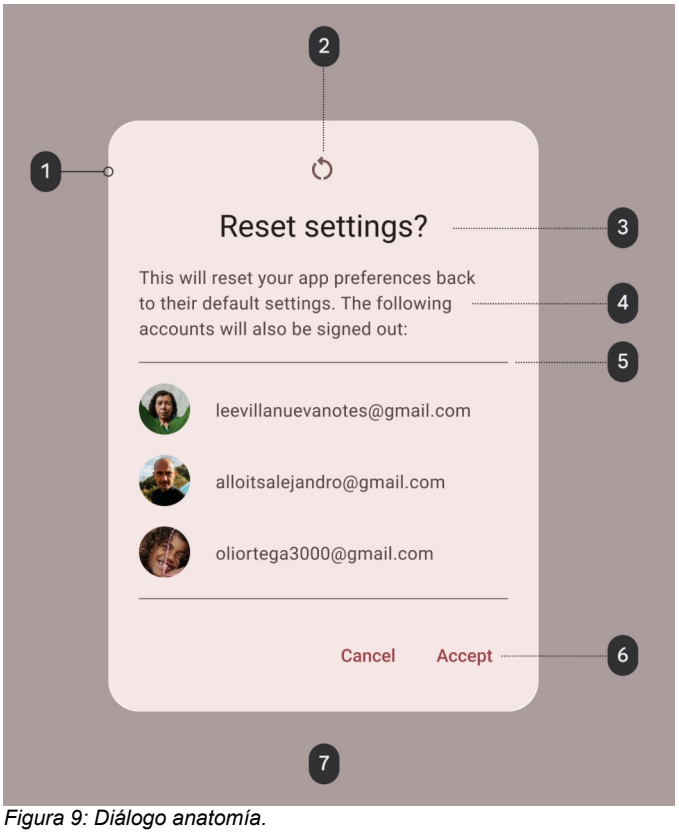

Figura 9: Diálogo anatomía.
https://m3.material.io/components/dialogs/guidelines

1. Container.
2. Icon (Opcional).
3. Headline (Opcional).
4. Supporting text.
5. Divider (Opcional).
6. Buttons (Label text).
7. Scrim.

Lists

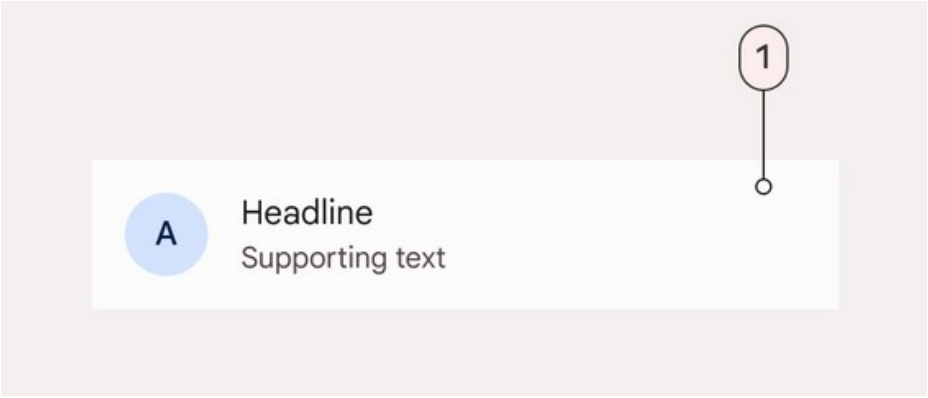

Figura 10: Lists. https://m3.material.io/components/lists/guidelines

1. List container.

4 PLANIFICACIÓN

4.1 Recursos

4.1.1.1 Materiales / Servicios

La funcionalidad de la aplicación de crear documentos está apañada por la API de Google (Spreadsheet), esto quiere decir que tiene que cargar con los gastos de uso. Los pro son la quita de carga de procesamiento al móvil ya que trabaja en la nube y la dependencia a internet.

Llamadas a la API al mes por cuenta de facturación	Costo por cada millón de llamadas a la API
De 0 a 2 millones	$ 0,00
De 2 Millones a 1,000 millones	$ 3,00
Mas de 1,000 millones	$ 1,50

Figura 11: Elaboración propia

Ordenador para Programadores (Ubuntu):

- CPU: Intel Core i5 / AMD Ryzen 5 - 200-300 €
- RAM: 16 GB DDR4 - 80-120 €
- Almacenamiento: SSD 500 GB - 50-80 €
- Tarjeta Gráfica: Integrada en la CPU - Incluida en el costo de la CPU
- Placa Base: Compatible con CPU - 100-150 €
- Caja/Torre: ATX/Micro ATX - 30-50 €
- Fuente de Alimentación: 500W - 40-60 €
- Monitor: 24" Full HD - 100-150 €
- Teclado y ratón: Conjunto básico - 20-30 €
- Total aproximado por computadora: 620-970 €

Ordenador para Director de Marketing (Windows):

- CPU: Intel Core i7 / AMD Ryzen 7 - 300-400 €
- RAM: 32 GB DDR4 - 150-200 €
- Almacenamiento: SSD 1 TB - 100-150 €
- Tarjeta Gráfica: Dedicada, gama media - 150-250 €
- Placa Base: Compatible con CPU - 150-200 €
- Caja/Torre: ATX - 50-80 €
- Fuente de Alimentación: 600W - 50-80 €
- Monitor: 27" Full HD - 150-200 €
- Teclado y ratón: Conjunto ergonómico - 50-80 €
- Licencia de Windows 10 Pro: 150-200 €
- Total aproximado por computadora: 1300-1740 €

4.1.1.2 Humanos

Para los trabajadores en un caso acercándose a la realidad, se contrataría a dos programadores junior y uno senior, además se contrataría un jefe de marketing.

4.1.1.3 Presupuesto

“Un Programador junior, con menos de 3 años de experiencia laboral, puede esperar un salario medio de alrededor de 19.700 € brutos por año. Un Programador con 4-9 años de experiencia puede tener un salario promedio de alrededor de 27.400 €, mientras que un Programador senior con 10-20 años de experiencia gana un promedio de 42.600 €. Un Programador con más de 20 años de experiencia puede esperar un salario promedio general de 49.300 €.”

Jobted (2024). Sueldo del Programador en España.https://www.jobted.es/salario/programador

“*El sueldo promedio de un Marketing es EUR 34.584 por año en Madrid, España. La remuneración promedio de efectivo adicional para un Marketing en Madrid, España es de EUR 5.206, con un rango de entre EUR 2.589 y EUR 9.955.*”

Glassdoor(2024,15,5).Sueldos para Marketing en Madrid, España. https://www.glassdoor.com.ar/Sueldos/madrid-marketing-sueldo-SRCH_IL.0,6_IM1030_KO7,16.htm

Con estos datos se calculará cuanto serían los gastos si el proyecto tendría una duración de 3 meses por un grupo formado por dos programadores Junior, un Senior y director de marketing.

Para los programadores:

- 2 Programadores Junior: 2 * 19.700 € / 12 meses * 3 meses = 9.850 €
- 1 Programador Senior: 27.400 € / 12 meses * 3 meses = 6.450 €

Para el director de marketing:

- Sueldo promedio anual: 34,584 €.
- Remuneración promedio de efectivo adicional: 5,206 €.

Entonces, el sueldo total para el director de marketing durante tres meses sería: (34.584 € + 5.206 €) / 12 meses * 3 meses = 14.395 €.

Sumando los salarios totales para el equipo:

- Programadores: 9.850 € + 6.450 € = 16.300 €.
- Director de marketing: 14.395 €.

El gasto total para el equipo durante tres meses aproximadamente sería de 16.300 € + 14.395 € = 30.695 €.

Contando sus componente estaría entre 33.235 € - 34.375 €

Resumen

Trabajadores	Coste por 3 mes	Cantidad	Valor Total
Programador Junior	4.925€	2	9.850 €
Programador Senior	6.450 €	1	6.450 €
Jefe de marketing	14.395 €	1	14.395 €

Ordenador	Coste	Cantidad	Valor Total
Ubuntu	620-970 €	3	1.860 – 2.910 €
Windows	1300-1740 €	1	1.300 – 1.740 €

Suma Total	33.235 – 34.375 €

Figura 12: Elaboración propia

4.2 Tareas

Las **tareas** se dividirían en la creación del diseño de la UI, Creación de la UI,desarrollo del software, la fase de prueba y la campaña de marketing.

En el **diseño** de la UI se utilizaría la herramienta Figma ya que contiene paquetes de componentes de Google para el diseño y el espacio de trabajo puede ser **colaborativo**. De este trabajo se encargaría el programador Senior con la colaboración del jefe de marketing.

La **creación** de la UI se desatollaría en Android Studio por los dos programadores Junior con la supervisión del programador Senior.

El **desarrollo** del **softaware** también se desarrollaría por los programadores Junior en Android Studio con las indicaciones e intervenciones necesarias del programador Senior.

Donde se priorizaría dos puntos importantes, la **autentificación** y la **funcionalidad** de creación de facturas.

Para terminar el desarrollo de la app esta pasaría a una fase de pruebas (**producción**) donde se buscarían todos los errores que se pudieran haber saltado en la fase de desarrollo.

A la vez se empezaría a preparar la campaña de marketing para la pronta salida de la app.

4.3 Planificación temporal

Mediante un **diagrama de Gannt** se planificó el tiempo que se le dedicaría a cada etapa del proyecto.

Figura 13: Elaboración propia

4.4 Riesgos

4.4.1 Identificación de riesgos y Plan de acción

Dependencia en herramientas y tecnologías específicas:

- **Riesgo**: Dependencia en Figma y Android Studio podría causar problemas si estas herramientas experimentan fallas técnicas, interrupciones en el servicio o cambios inesperados que afecten la productividad del equipo.

- **Mitigación**: Establecer planes de contingencia para alternativas en caso de fallos de las herramientas principales. Capacitar al equipo en el uso de herramientas alternativas.

Dificultades en la colaboración y comunicación:

- **Riesgo**: Problemas de comunicación y colaboración entre el programador Senior, los programadores Junior y el jefe de marketing podrían resultar en malentendidos, retrasos en la entrega y errores en el producto final.

- **Mitigación**: Establecer canales de comunicación claros y regulares entre todos los miembros del equipo. Programar reuniones periódicas para revisar el progreso y abordar cualquier problema o inquietud.

Capacidades técnicas insuficientes del equipo:

- **Riesgo**: La falta de experiencia en el uso de Figma, Android Studio o en el desarrollo de software en general por parte de los programadores Junior podría llevar a errores en el diseño de la UI, problemas de rendimiento del software o vulnerabilidades de seguridad.

- **Mitigación**: Proporcionar capacitación adicional a los programadores Junior en las herramientas y tecnologías necesarias. Asignar tareas acordes a sus habilidades y proporcionar supervisión adecuada por parte del programador Senior.

Errores de autenticación y funcionalidad:

- **Riesgo**: La falta de atención adecuada en la implementación de la autenticación y la funcionalidad de creación de facturas podría resultar en vulnerabilidades de seguridad, errores de acceso o problemas de usabilidad en la aplicación.

- **Mitigación**: Realizar pruebas exhaustivas durante el desarrollo y la fase de pruebas para identificar y corregir posibles problemas. Priorizar la revisión y validación de la autenticación y la funcionalidad de creación de facturas durante todas las etapas del proceso de desarrollo.

Errores no detectados en la fase de pruebas:

- **Riesgo**: Los errores no detectados durante la fase de pruebas podrían afectar negativamente la calidad y la reputación de la aplicación una vez lanzada al mercado.

- **Mitigación**: Implementar pruebas exhaustivas que abarquen una amplia gama de escenarios de uso. Realizar pruebas de usuario beta para obtener retroalimentación adicional antes del lanzamiento oficial. Asignar recursos adecuados y tiempo suficiente para realizar pruebas completas.

Demora en la preparación de la campaña de marketing:

- **Riesgo**: La demora en la preparación de la campaña de marketing podría resultar en un lanzamiento de la aplicación sin una estrategia de promoción efectiva, lo que afectaría su visibilidad y adopción por parte de los usuarios.

- **Mitigación**: Establecer un plan de acción claro y fechas límite para todas las actividades relacionadas con la campaña de marketing. Asignar recursos adecuados y coordinar de cerca con el equipo de desarrollo para asegurar un lanzamiento sincronizado y exitoso.

5 EJECUCIÓN

5.1 Implementación y seguimiento de tareas

Al comenzar se eligió la paleta de colores para la app, la primera opción elegida sería colores entre rosados y morados no muy saturados (pastel), los rosados pensados para el tema claro y los morados para el oscuro, más tarde se cambiarían por unos colores azulados pastel ya que transmiten **Inteligencia, confianza, seguridad, serenidad, comunicación, eficiencia, lógica, reflexión, calma.**

El inicio del desarrollo es marcado por la creación del MainActivity que es donde se comenzaría a implementar un Drawer con un menú para navegar entre la páginas principales de la app y, en la parte superior de la pantalla una app bar donde se colocaría el “Guiding” del Drawer y un menú. Por ultimo un Fragment donde se visualizaría los layouts que se crearían más tarde.

La pantalla principal del Drawer sería BillsFragment que contiene un Recycler View y un FAB (Flaoting action Button) para crear Facturas.

Luego otra pantalla donde se podría navegar desde el Drawer sería DeliveryNoteFragment, que, el layout sería una copia exacta del BillsFragment nada más que el FAB crearía albaranes.

Los Recycler View de las dos anteriores pantallas mencionadas, se mostraría la información resumida de los documentos creados en formato de lista y para esto se crearía otro layout.

Para continuar el proyecto se pensó como se crearía las facturas al pulsar el FAB de BillsFragment, la solución que se dio y se implemento fue un layout que directamente sea la factura, donde se introduciría los datos a clicar en los

campos rellenables, además el layout contendría un FAB que al ser clicado guardaría el estado actual de la pantalla y lo tranformaría de bytes a formato pdf.

Esta solución tenía dos problemas:

1. Al solo haber una factura se perdía la escalabilidad de la aplicación.
2. La factura cambiaría de tamaño según la cantidad de pixeles del móvil.

El primer problema era solventable pero el segundo no, en este punto comenzaría una cadena de errores. Para conseguir que todas la facturas se vieran del mismo tamaño sin importar el móvil, se pensó trabajar con hojas de calculo, la solución propuesta era almacenar un Exel en el proyecto y con las librerías de « Apache POI Common » editar ese excel y convertirlo en pdf con los datos introducidos por el usuario, como el testing de las librerías se hizo en InteliJ (Otro IDE) se pensó que funcionaban correctamente (así era) pero al ser implementado en Android hubo problemas de compatibilidad. Esto ocurrió porque las hojas de cálculo se modifican gráficamente y para hacer esto se utilizaban las librerías de JavaAWT que generaba una incompatibilidad con Android, ya que este tiene sus propias librerías graficas, se intento modificar el código fuente de las librerías para borrar y modificar las partes que causaban el fallo, pero fue inservible. Como Excel daba problemas se intento pasar a las hojas de calculo de Libreoficce donde se utilizaría las librerias « JOpenDocument 1.3 » , aquí se llego a la misma conclusión que con Excel.

Aunque se gasto tiempo implementando las librerías que no funcionaban, se crearon Diálogos para introducir los datos de la factura que se reutilizarían más tarde.

Tras investigar para una solución se llegó a las librerías de Google, su API de Spreadsheet (hojas de cálculo) y Drive (almacenamiento en la nube del cliente).

Para la utilización de la API de Google se tiene que cumplir unos requisitos:

- Crearse un proyecto en console.cloud.google.com.
- Crearse una credenciales para acceder a las API con los datos de tu proyecto.
- Activar las API que se utilizarán.
- Modificar la pantalla de consentimiento y en el caso de esta app agregar usuarios de prueba que podrán utilizarla (para no tener que verificar mi aplicación).
- Para Android modificar el manifest.xml.

OAuth 2.0 es un protocolo de autorización que utiliza tokens de acceso. Un **Token de acceso** es un dato que representa la autorización para acceder a los recursos.

El proceso implica que la aplicación solicite permiso al usuario, quien luego otorga acceso. Posteriormente, el servidor de autorización emite un código de autorización a la aplicación, que la aplicación intercambia por un token de acceso. Este token permite a la aplicación acceder a los recursos protegidos en nombre del usuario. Además, el servidor de autorización puede proporcionar un token de actualización, que la aplicación puede usar para obtener nuevos tokens de acceso en el futuro.

Esto asegura un flujo continuo de acceso a los recursos protegidos sin la necesidad de volver a solicitar permiso al usuario.

Flujo de 0Auth2

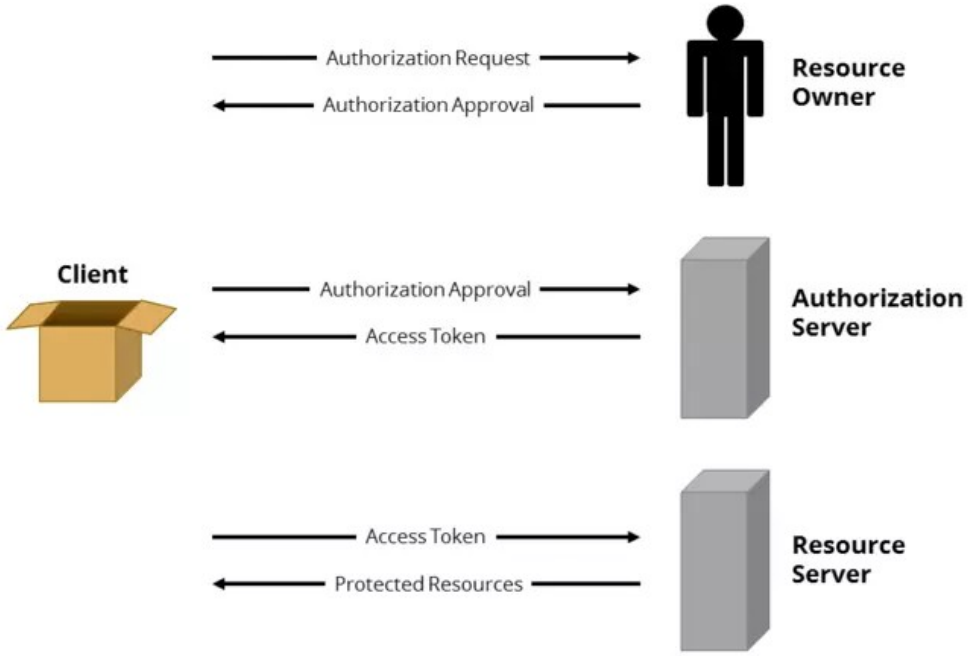

Figura 14: Flujo de protocolo abstracto en OAuth2 .2019
https://www.ionos.es/digitalguide/servidores/seguridad/oauth-y-su-version-oauth2/

El nuevo enfoque de la app provoco introducir 3 nuevos layouts:

- Pantalla de login.
- Pantalla para elegir diseño de documento.
- Pantalla para rellenar datos.

La pantalla de login pasaría a ser la nueva pantalla de inicio y esta contendría el logo de la app y un botón de inicio de sesión que implementaría las primeras dos fases de 0Auth2 y guardaría un token para la utilización de la API.

La pantalla para elegir el diseño se agregó para que haya escalabilidad, su función es simple, contiene un Switcher de imágenes y tres botones. Dos botones para cambiar de imagen (siguiente y anterior) y otro para elegir la imagen. El Switcher contendría los diseño de los documentos para el usuario,

estos serían distintos si se llegó a esta pantalla por el lado de los albaranes o por el de las facturas.

Para terminar se crearía la pantalla para rellenar datos donde se reutilizarían los Diálogos creados anteriormente, este contendría secciones:

- Cliente
- Empresa
- Datos de la Factura
- Artículos

Cada una de estas tendrían sus valores para rellenar, que se haría mediante los Diálogos utilizando el botón editar, además se agregó una funcionalidad de colapsar y expandir para la mejor visualización de todos los datos.

En el caso de los artículos se podrían editar y crear, ya que el número de estos es variable.

Por último estría el botón de crear factura, aquí se obtendría todos los datos rellenados y se comunicaría con la API para crear la factura y descargarla en el móvil del usuario.

La funcionalidad de la creación de las facturas funciona de la siguiente manera:

En Google Drive cuando se almacena un documento, a este se le otorga un ID único para todos los usuarios, para trabajar con la API utilizaremos este, en este caso da igual que API sea (Drive o Spreadsheet) ya que estos comparten ID del documento.

Cuando el usuario eligió el diseño a la vez estaba eligiendo el ID del documento de Google se iba usar(el documento está almacenado en el Drive

de la empresa) y además estaba eligiendo las instrucciones que utilizaría la API para crear el documento.

En general las **instrucciones** siguen estos pasos:

1. *__Editar el documento__ copiado (Introducir los datos anteriormente almacenados) (API Spreadsheet).
2. **Copiar el documento** (hoja de calculo) elegido en el Drive del cliente (API Drive).
3. **Descargar documento** con formato PDF(API Drive).
4. **Almacenar Documento** desde el móvil.

*La edición del documento podría variar según el diseño del documento.

Todas las comunicaciones con la API funcionan con la conexión del usuario a internet y el token almacenado al iniciar sesión.

6 CONCLUSIONES

6.1 Nivel de satisfacción

En grado de satisfacción si tendría que ponerle una puntuación del 1 al 10, sería un 9.

En este proyecto, se a enfrentado desafíos que parecían insuperables. Sin embargo, gracias a un esfuerzo constante y dedicación, se logró superar exitosamente. Durante el desarrollo de la aplicación, hemos ajustado la funcionalidad de la creación de facturas en varias ocasiones, hasta encontrar una solución que cumpliera con los requisitos del cliente y con la compatibilidad de los dispositivos móviles.

A pesar de que la aplicación aún no está completamente desarrollada y tiene margen para la mejora y la adición de nuevas funciones, enorgullece decir que no está muy por detrás de sus competidores actuales. De hecho, hay una creencia firme que, con el tiempo, podría incluso superar gracias al análisis, funciones y diseño que se ha incorporado en su desarrollo.

El proyecto no alcanza la perfección, lo cual es su principal punto negativo. No obstante, incluso si se lograra la perfección por pura ambición, la nota final permanecería invariable. Esto se debe a que una aplicación siempre tiene margen para mejoras y adiciones, reflejando la naturaleza en constante evolución de la tecnología y las demandas del usuario.

6.2 Propuestas de mejora

Para mejorar en siguientes proyectos propondría mejorar la **preparación** del proyecto.

Razones:

- Diseñar los layout antes de empezar a programar, hacerlo mientras estás programando **ralentiza** mucho la creación de estas.

- Tener las ideas claras desde el principio, **dudar** de como se va hacer algo te **quita tiempo** y concentración al la hora de programar la aplicación.

- Si se plantea **crear una funcionalidad** para una aplicación **corroborar** que se pude ejecutar en el dispositivo y no tiene contras.

- Elegir desde el inicio el **idioma** en el que se **programará** y se **documentará**.

6.3 Propuestas de ampliación

En el caso de que la app se terminara de desatollar se le podría implementar lo siguiente:

- Un sistema de plugins donde se pueda añadir funcionalidades de gestión a la app (así no pierde su esencia y solo lo activaría el que quiere).
- Un sistema de personalización de los colores de la app.
- Formas de filtrar los diseños de la app.
- Un sistema de guardado de clientes.
- Un canal de propuestas para tener *feedback* con los clientes.

7 BIBLIOGRAFÍA Y WEBGRAFÍA

- Glassdoor. (2024). Sueldos para Marketing en Madrid, España. https://www.glassdoor.com.ar/Sueldos/madrid-marketing-sueldo-SRCH_IL.0,6_IM1030_KO7,16.html.
- Jobted (2024). Sueldo del Programador en España.https://www.jobted.es/salario/programador.
- Google. (2024). *Precios de API Gateway*. https://cloud.google.com/api-gateway/pricing.
- Gobierno de España. (2023) *Datos estadísticos relativos a trabajadoras y trabajadores autónomos afiliados a 30 de abril de 2023*. https://www.mites.gob.es/ficheros/ministerio/sec_trabajo/autonomos/economia-soc/NoticiasDoc/NoticiasPortada/2023/Nota_Afiliacion-trabajo-autonomo_Abril.pdf.
- Digital Guide IONOS. (2019). *OAuth: inicio de sesión único en varias plataformas.* *https://www.ionos.es/digitalguide/servidores/seguridad/oauth-y-su-version-oauth2/.*
- Gobierno de España. *Herramienta DAFO. https://dafo.ipyme.org/Home.*
- Google. *Material Design.https://m3.material.io/.*