--

**(BORRADOR)**



**IES PERE MARIA ORTS I BOSCH**

**CICLOS FORMATIVOS DE GRADO SUPERIOR**



**Familia Profesional: Informática y Comunicaciones**

**Sei Vegan, Sei Grün**

**Alumno: Alejandro Martínez Morillo**

**(BORRADOR)**

**IES PERE MARIA ORTS I BOSCH**

**CICLOS FORMATIVOS DE GRADO SUPERIOR**



**Familia Profesional: Informática y Comunicaciones**



**PROYECTO DE FIN DE CICLO**

**Ciclo Formativo**

**Sei Vegan, Sei Grün**

**AUTOR: Alejandro Martínez Morillo**

**TUTOR: Jose Rafael Caturla Palao**

**Benidorm, 4 de junio de 2023**

# Agradecimientos

Este proyecto ha resultado ser un reto, tanto por lo que ha sido el proyecto como por causas externas, por lo que me gustaría agradecérselo a muchas personas.

A las primeras personas que me gustaría agradecer son a mis profesores y quiero nombrar a todos, desde Antonio y Claudia, que son asignaturas algo más alejadas del ciclo, hasta a Marina, Jose, Jaume, Fran y Noe. Gracias a todos por ayudarme a crecer académicamente hablando.

También quería agradecer a esas personas que han estado conmigo y han aguantado infinidad de tonterías, buenos y malos ratos y, sobretodo, risas. Gracias a Guillermo, Salva y David por hacer que este año haya sido increíble y haber hecho que ir a clase sea especial, porque me pasase algo malo o no, estabais ahí para hacérmelo olvidar. Y, ya que hablamos de amistades, también me gustaría nombrar a mi amiga Toñi, que me ha apoyado y aguantado como una campeona. Gracias por todo chicos.

A las siguientes personas que quiero agradecer, y también dedicar, este proyecto es a mis padres, sobre todo a mi madre, que desde que empezó el año no ha tenido, ni de lejos, una buena temporada. Me han ayudado y apoyado en muchas situaciones y han soportado alguna que otra mala contestación cuando estaba estresado o ansioso y, claramente, no se la merecían. Gracias por todo, os quiero mucho y os merecéis un mundo entero.

Por último, pero no menos importante, quería agradecer, y también dedicar, este proyecto a mi pareja Angie. Ella es la única realmente que sabe lo mucho que eh trabajado durante este curso, lo mucho que he sufrido haciendo este proyecto, y más aún en el intermodular. Ella es la que siempre me ha animado y apoyado en todas y cada una de las decisiones que he tomado, ayudándome siempre a elegir la mejor. Gracias a ella he crecido mucho como persona y me ha hecho ser quien soy ahora mismo. Eres única Angie, te quiero mucho, gracias por todo lo que has hecho por mí.

Contenido

[Agradecimientos i](#_Toc136422728)

[Resumen v](#_Toc136422729)

[Abstract vi](#_Toc136422730)

[1. Introducción 2](#_Toc136422731)

[1.1. Objetivos iniciales 3](#_Toc136422732)

[1.2. Objetivos cumplidos 4](#_Toc136422733)

[1.3. Justificación del proyecto 4](#_Toc136422734)

[2. Estudio económico 6](#_Toc136422735)

[2.1. Idea de negocio 6](#_Toc136422736)

[2.2. Elección de forma jurídica 7](#_Toc136422737)

[2.3. Estudio de mercado 8](#_Toc136422738)

[2.3.1. Macroentorno 8](#_Toc136422739)

[Económico 8](#_Toc136422740)

[Sociocultural 9](#_Toc136422741)

[Político 10](#_Toc136422742)

[Legal 10](#_Toc136422743)

[Tecnológico 10](#_Toc136422744)

[Medioambientales 11](#_Toc136422745)

[Internacionales 11](#_Toc136422746)

[2.3.2. Microentorno 12](#_Toc136422747)

[Competidores 12](#_Toc136422748)

[Clientes 12](#_Toc136422749)

[Proveedores 12](#_Toc136422750)

[Productos sustitutivos 13](#_Toc136422751)

[2.3.3. Ubicación del negocio y plano del local 13](#_Toc136422752)

[2.3.4. Análisis DAFO 14](#_Toc136422753)

[2.4. Plan de marketing 14](#_Toc136422754)

[2.4.1. Producto 14](#_Toc136422755)

[2.4.2. Precio 14](#_Toc136422756)

[2.4.3. Promoción 15](#_Toc136422757)

[2.4.4. Place/distribución 15](#_Toc136422758)

[2.5. Plan de producción y recursos humanos 15](#_Toc136422759)

[2.5.1. Inversiones y gastos 15](#_Toc136422760)

[2.5.2. Origen del financiamiento 16](#_Toc136422761)

[2.5.3. Ayudas y subvenciones 17](#_Toc136422762)

[2.5.4. Distribución de costos 17](#_Toc136422763)

[2.5.5. Umbral de rentabilidad o punto muerto 17](#_Toc136422764)

[2.6. Trámites de constitución, puesta en marcha e impuestos 18](#_Toc136422765)

[3. Análisis 20](#_Toc136422766)

[3.1. Planificación temporal 20](#_Toc136422767)

[3.2. Casos de uso 21](#_Toc136422768)

[3.2.1. Aplicación administrativa 22](#_Toc136422769)

[Caso de uso – Gestionar usuarios, trabajadores y platos 22](#_Toc136422770)

[3.2.2. Aplicación trabajadores 24](#_Toc136422771)

[Caso de uso – Camarero 24](#_Toc136422772)

[Caso de uso – Cocinero 25](#_Toc136422773)

[3.2.3. Aplicación clientes 26](#_Toc136422774)

[Caso de uso – Usuario- Crear pedido online 26](#_Toc136422775)

[Caso de uso – Usuario- Ver menú 26](#_Toc136422776)

[Caso de uso – Usuario- Editar perfil 27](#_Toc136422777)

[3.3. Diagrama de clases 28](#_Toc136422778)

[3.4. Modelo relacional 29](#_Toc136422779)

[4. Entorno de programación 30](#_Toc136422780)

[4.1. Lenguajes y tecnologías que vamos a utilizar 30](#_Toc136422781)

[4.1.1. Base de datos 30](#_Toc136422782)

[Mongo 30](#_Toc136422783)

[4.1.2. Backend 31](#_Toc136422784)

[Node.js y Express 31](#_Toc136422785)

[Librerías utilizadas 31](#_Toc136422786)

[4.1.3. Frontend 33](#_Toc136422787)

[Dart 33](#_Toc136422788)

[Flutter 33](#_Toc136422789)

[Provider 34](#_Toc136422790)

[Bloc 34](#_Toc136422791)

[5. (Apartado 5: Ejemplo bibliografía) 36](#_Toc136422792)

[6. (Apartado 6) 38](#_Toc136422793)

# Resumen

Este documento representa la creación y el desarrollo del proyecto final de curso del Ciclo Superior de Desarrollo de Aplicaciones Multiplataforma.

Para la creación del mismo, se han creado distintos elementos ficticios para que el proyecto sea lo más cercano al mundo laboral. Como primer elemento, se ha creado una sociedad limitada de un único socio llamada *SoftDev*. En el apartado 2 de este mismo proyecto, observaremos el proceso completo de lo que es crear una sociedad limitada con unas características concretas. Además, el segundo ficticio al cual está ligado es el restaurante el cuál necesita nuestros servicios, *Grün.* Este restaurante aporta una oferta vegana al mercado, la cuál es escasa actualmente y refleja unos valores que el creador de dicho proyecto defiende.

Apartando el tema empresarial y económico, en este documento veremos reflejado el desarrollo de 3 aplicaciones, centradas en distintos usuarios finales.

El primer paso de todos fue analizar el proyecto a grandes rasgos y ver cómo enfocar el desarrollo del mismo. Un diagrama de *Gantt* inicial o analizar y crear los posibles casos de usos de las aplicaciones son algunos de los primeros pasos que se deben hacer.

Ya analizada la aplicación y observar las consideraciones a tener en cuenta, llega el momento de preparar el entorno de programación y sus librerías, el control de versiones de los proyectos y el estudio de nuevas tecnologías que se pondrán en uso.

Por último, podremos observar distintos apartados destacables de la aplicación, ya sean por su complejidad o por lo interesante y útil que ha resultado implementar dicho código. Además, pondremos a prueba la aplicación utilizando los casos de uso creados con anterioridad.

Abstract

This document represents the creation and development of the final project of the Higher Cycle of Multiplatform Applications Development.

For the creation of the project, different fictitious elements have been created to make the project as close as possible to the working world. As a first element, a single-partner limited company called SoftDev has been created. In section 2 of this project, we will observe the complete process of creating a limited company with specific characteristics. In addition, the second fictitious company to which it is linked is the restaurant that needs our services, Grün. This restaurant brings a vegan offer to the market, which is currently scarce and reflects the values that the creator of this project defends.

Apart from the business and economic issue, in this document we will see the development of 3 applications, focused on different end users.

The first step of all was to analyse the project in broad strokes and see how to approach its development. An initial Gantt chart or analysing and creating the possible use cases of the applications are some of the first steps to be taken.

Once the application has been analysed and the considerations to be taken into account have been observed, it is time to prepare the programming environment and its libraries, the version control of the projects and the study of new technologies that will be put into use.

Finally, we will be able to observe the different sections of the application, either for their complexity or for how interesting and useful it has been to implement the code. In addition, we will test the application using the use cases created previously.

1. Introducción

Sei Vegan, Sei Grün es un proyecto realizado por la empresa SoftDev, una empresa de desarrollo software que creará y desplegará las aplicaciones que necesita el restaurante Grün, centrado en la comida vegana.

La representación de este proyecto se divide en tres partes, una centrada en el uso del administrador, una segunda centrada en el uso de los trabajadores, tanto los cocineros como los camareros, y una tercera y última parte centrada en los clientes.

En la primera aplicación, centrada en la administración, el dueño o la persona designada en la gestión de recursos podrá encargarse de añadir, borrar y actualizar los datos del restaurante, ya sean los datos relacionados con los trabajadores como los relacionados con los productos del negocio.

Por otro lado, la segunda aplicación que se desarrollará para este negocio estará centrada en el uso de los trabajadores a la hora de atender a los clientes en el local, pensada para ser utilizada en un teléfono móvil o tablet. Por un lado, los camareros tendrán acceso a un TPV para crear y gestionar los pedidos realizados en el local, ya sean para degustarlos en el lugar o para llevar. Por otro lado, los cocineros tendrán una vista, pensada para que se utilice en una tablet o AIO (All in one). En este apartado, los cocineros podrán ver todos los pedidos, junto a sus detalles, y marcarlos como “Finalizados”.

Por último, se desarrollará la aplicación centrada en el uso del cliente. En dicho programa, los usuarios tendrán dos acciones principales: visualizar el menú y crear un pedido. La primera opción está diseñada y orientada para el uso en el local, de esta manera los clientes pueden visualizar el menú en cualquier momento y no tendrán por qué esperar a que un camarero les entregue la carta físicamente. La segunda función está centrada en el uso desde el domicilio. Desde la comodidad de su casa, podrán pedir cualquier producto de la carta para que lo entreguen en su domicilio.

* 1. Objetivos iniciales

A la hora de presentar la propuesta de proyecto, se estipularon una serie de objetivos que, como veremos más adelante, se cumplen. A grandes rasgos, el objetivo principal del proyecto es desarrollar tres tipos de aplicaciones, las cuáles han sido desarrolladas *grosso modo*. Para la parte administrativa, tenemos como objetivos desarrollar un CRUD (*Create*, *Read*, *Update* *Delete*) completamente funcional.

Por otro lado, a la hora de desarrollar la aplicación para los trabajadores, concretamente para los camareros, el objetivo es crear distintas funciones, las cuales son:

* Creación de pedidos.
* Creación de facturas.
* Diseño de un menú e implementarlo en la aplicación actual.

En cambio, en la vista de los cocineros tendremos una única función; visualizar los pedidos y todos sus detalles.

Como objetivo adicional, se desea implementar un *socket* para que dichas aplicaciones sean en tiempo real y que los cocineros no tengan la necesidad de darle un botón para recibir los nuevos pedidos

La tercera aplicación que se desea desarrollar tiene como función principal la de crear pedidos a domicilio. También se quiere desarrollar una vista con el menú completo con el objetivo de ayudar al usuario final en la elección de sus productos. Asimismo, como en la anterior aplicación, se intentará implementar un *socket* para que sea una aplicación en tiempo real, facilitando tanto la llegada de los pedidos como las actualizaciones del mismo.

Como último objetivo prioritario de este proyecto, se desarrollará un software intermediario que tiene como finalidad la comunicación entre aplicaciones y la base de datos. En ella, se desarrollarán distintas funciones que ayudarán en la comunicación del software anteriormente nombrado.

Por último, como objetivo ajeno a la programación en sí, se desea diseñar y recrear el menú del restaurante y realizar las fotografías correspondientes para utilizarlas en las distintas aplicaciones.

* 1. Objetivos cumplidos

Tras realizar el proyecto, podemos analizar todos los puntos obligatorios y observar que todos los puntos han sido cumplidos correctamente. Además de todo esto, algunos puntos opcionales también han sido añadidos, como que algunas aplicaciones sean a tiempo real, y también han sido añadidas algunas características que he creído que serían buenas para el proyecto, como la impresión de los pedidos en forma de ticket o la implementación de gráficos para que el dueño pueda observar qué productos son los más/menos vendidos.

* 1. Justificación del proyecto

La decisión de desarrollar este proyecto tiene un trasfondo totalmente personal. Mis padres, desde pequeño, me han transmitido un amor y una pasión por la comida que, a día de hoy, sigo manteniendo. Además, siempre me han ido enseñando a cocinar, desde una simple merienda hasta algún plato que pueda alimentar a una familia entera.

Por otro lado, otra parte característica de este proyecto es que mantenga una dieta vegana. Desde hace unos años, mi curiosidad sobre este tipo de dietas ha ido en auge, siendo el confinamiento uno de los momentos en los que más me centré en ello. Desde entonces, he descubierto un mundo culinario muy interesante, tanto como plantear llevar este tipo de dieta, aunque no es correcto llamarlo de tal manera.

En conclusión, he juntado dos facetas que me hacen feliz, la programación y la cocina, junto a un mundo culinario que no está para nada explotado. Además de crearme un reto en cuánto a la programación, decidí conjuntarlo con la creación de un menú propio para fusionar al máximo posible estas dos facetas propias.

1. Estudio económico
   1. Idea de negocio

Mi idea de negocio es crear una empresa en la que, como anteriormente he dicho, pueda desarrollar software que sea moderno. El término “moderno” se puede interpretar de distintas maneras. Quiero enfocar este término tanto a la estética como a la programación, para que las empresas tengan siempre una opción óptima. Como idea principal, quiero mantener siempre con una continua formación de mí mismo, o mis futuros empleados, para crear aplicaciones con los lenguajes de programación más actuales.

Actualmente, estoy centrado en el estudio y la implementación de lenguajes de programación como Dart y Flutter, lenguaje multiplataforma que nos permite, con el mismo código, exportar a otros sistemas operativos sin problema, y Kotlin con Jetpack Compose, el cual nos permite desarrollar aplicaciones en móvil y, con Desktop Compose, en escritorio. Añadir que ambos lenguajes son desarrollados por Google.

Por otro lado, en cuánto a estética, actualmente la más común en la mayoría de lenguajes es Material Design, desarrollado por Google también. A mi empresa le viene como anillo al dedo que esta estética sea muy reclamada y utilizada porque, tanto Flutter como Jetpack Compose, la utilizan dicha en todos sus componentes.

Por último, en lo que se centra justamente este proyecto dentro de la empresa es sobre una aplicación móvil para pedir comida a domicilio de un restaurante vegano. Además, centrado en el ámbito empresarial, desarrollaré una aplicación TPV (Terminal de Punto de Venta) centrada para el uso de los camareros y una aplicación centrada en el administrador del restaurante para gestionar todos los datos del mismo.

* 1. Elección de forma jurídica

Tras analizar nuestras expectativas, nuestro capital inicial y otros factores, se decide que la forma jurídica para este proyecto es una sociedad de responsabilidad limitada. Además de ser la forma social muy común a la hora de realizar proyectos, otra razón tiene que ver con lo dicho inicialmente, las expectativas. Si tuviésemos un capital mucho más alto y tuviéramos en mente salir a bolsa, podríamos optar para elegir una sociedad anónima como forma social, por ejemplo. Por otro lado, esta forma social es apropiada para las PYME, las pequeñas y medianas empresas, por lo que es la opción perfecta para nuestra empresa.

Como se comentó en el apartado Idea de negocio, este proyecto lo realizaré por mi cuenta, por lo que el número de socios de esta SL es de uno. De igual manera, la responsabilidad frente a terceros de los socios es limitada. Esto significa que, si en el peor de los casos la sociedad tiene pérdidas, estás quedan limitadas a los bienes, los derechos y el capital al nombre de la empresa, al contrario que los autónomos o personas fiscales que tienen una responsabilidad ilimitada.

Enlazando con el anterior apartado, el capital inicial de la sociedad será de 5000€. A la hora de crear una S.L., no es necesario tener un capital social elevado, al menos 1€, pero al momento de hacer el plan de tesorería es mucho más sencillo tener, al menos, 3000€. Esto es debido a que, si no superas los 3000€, deberás apartar, al menos, un 20% de las ganancias hasta que el capital social alcance los 3000€ estipulados.

Antes de comentar que debemos presentar para crear nuestra S.L., debemos comentar el aspecto fiscal de la misma. Si fuésemos personas fiscales o autónomos, nuestra forma jurídica se encontraría sometida al impuesto sobre la renta de las personas físicas, es decir, el IRPF, el cual es variable según la renta que tengamos, por lo que este impuesto aumentaría si el rendimiento de nuestra empresa aumenta. En cambio, al ser una sociedad limitada, nuestra forma judicial es sometida al impuesto de sociedades. En nuestro caso, tendremos que pagar un 15% durante el primer año en el que tengamos beneficios y, a partir de este, pagaremos un 25%

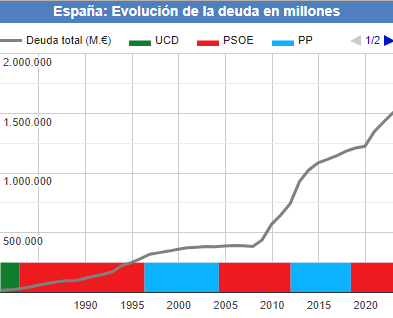
Por último, para crear la S.L. debemos aportar distintos datos, los cuales son:

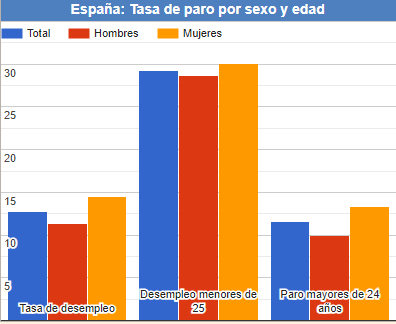
* Una escritura pública que se inscribe en el Registro Mercantil, el cual nos otorga una personalidad jurídica.
* La identidad de los socios, en este caso la identidad de la madre e hijo.
* La voluntad de constituir una S.L.
* Las aportaciones que otorgue cada socio y la numeración de las participaciones.
* Los estatutos de la sociedad.
* La manera en la que se va a administrar la sociedad.
* La identidad de las personas, ya sea única o varias, que se encargará de la administración y la representación social.
  1. Estudio de mercado

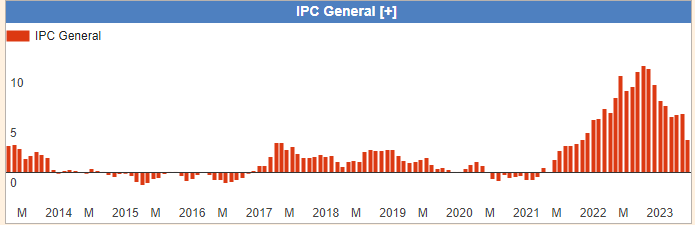
En este apartado vamos a analizar a nuestra futura empresa, tanto el entorno que la rodea como los puntos fuertes y debilidades que tiene.

* + 1. Macroentorno

Económico

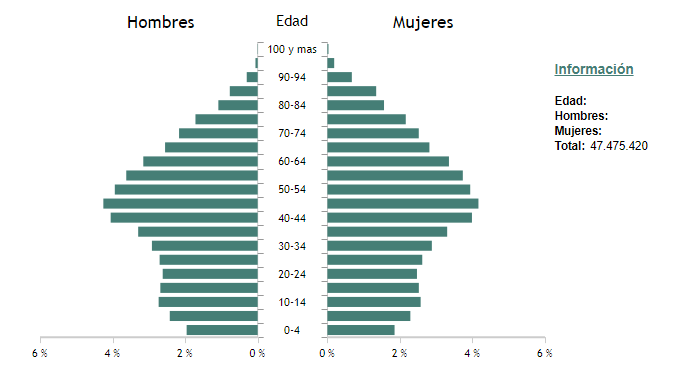
España está viviendo una situación económica complicada, la cual aumentó hace unos años con la pandemia del Covid-19. En noviembre del anterior año, la deuda pública aumento en 8.384 millones de euros respecto al mes anterior, pasando de 1.497.154 millones a 1.505.538 millones. Si recapitulamos a noviembre del 2021, observamos que la deuda ha crecido 1.484 € por habitante. Si esto se expresa mediante el PIB, observamos que, en este último trimestre, la deuda alcanzó el 115.6% del PIB en España, un 0.5% menos que el trimestre anterior. En cambio, si comparamos este último trimestre con el último trimestre de 2021, veremos que la deuda ha incrementado en 71.458 millones de euros.

Por otro lado, si hablamos del desempleo en España, observamos que, durante el último año, ha disminuido un 0.9%, posicionando el paro en un 12.4%. Actualmente, el paro está en un 12.67%, el cual ha aumentado desde un 0.27% desde noviembre del 2021. En cuantía de persona, el número de parados actualmente es de 2.980. 200 personas.

Por último, para hablar sobre el IPC, un tópico bastante recurrido en los debates y hablado por la población. La inflación en España, debido al conflicto bélico entre Ucrania y Rusia, ha aumentado un 6.4% desde diciembre del 2021. Aunque pensemos o, desde los medios de comunicación, parezca que ha incrementado mucho la inflación, si comparamos nuestro IPC interanual con los diferentes miembros de la Unión Europea, observamos que España es uno de los países con menor IPC. Por ejemplo, nuestro país vecino, Portugal, ha acumulado un 9.6% y Alemania, un país referente, ha aumentado un 8.6%.

Todos estos datos están obtenidos de [Datos Macro](https://datosmacro.expansion.com/paises/espana) y sus distintas secciones.

Sociocultural

Actualmente en España, con una población de 47.615.034 personas, se encuentra en la posición 30 de la tabla de poblaciones del mundo, con una densidad de 94 habitantes/Km2.

Si observamos el gráfico anterior, observamos que los picos más notables son de las personas entre los 40 y los 54. La conclusión clara que obtenemos de este gráfico es que tenemos más adultos, entre 40 y 54 años, que jóvenes, algo que comentaremos más adelante.

Si hablamos de nuestra sociedad y sus valores, ahora mismo estamos viviendo una situación en la que la tecnología está en auge. Tanto gente mayor como más joven tiene a su disposición un teléfono móvil, tablet u ordenador. Actualmente, pensar en un día a día sin un teléfono móvil es algo difícil de pensar, sobre todo para el público joven.

Si enlazamos los puntos anteriores con nuestra empresa, podemos ver una ventana de negocio muy rentable. La mayor cantidad de personas ronda entre los 40 y 55 años, además de haber una cantidad considerablemente de jóvenes. Esto nos va a beneficiar sin ninguna duda, ya que nuestro futuro producto, que será una aplicación de un restaurante vegano, estará orientado a gente joven-adulta.

Político

Nuestro país, España, es un estado democrático cuya forma política es una monarquía parlamentaria. Los ciudadanos participan en asuntos públicos a través de los representantes que se eligen en elecciones plurales, universales y plurales.

La situación actual en nuestro país es bastante tranquila y constante, con algunos picos de enfrentamientos políticos pacíficos entre el actual gobierno y los demás partidos. Actualmente, en España gobierna el Partido Socialista Obrero Español, con el apoyo de Unidas Podemos, el Partido de los Socialistas de Cataluña e Izquierda unida/Partido Comunista de España. Este conjunto de partidos de “izquierdas” se enfrenta, en unos meses, a unas elecciones generales en la que lucharán para mantener dicho gobierno de coalición y parar el auge de la extrema derecha.

Legal

A nivel estatal, existe un Convenio Colectivo que regula el ámbito de servicios informáticos, además de otros servicios como gestoría. Este CC fue publicado en el BOE el 6 de marzo de 2018, con resolución el 22 de febrero del mismo año. Este CC se puede encontrar [aquí](https://www.boe.es/diario_boe/txt.php?id=BOE-A-2018-3156).

Por otro lado, una ley que regula una parte de todos los servicios informáticos es la LOPD, es decir, la Ley Orgánica de Protección de Datos, que puede ser leída [aquí](https://www.boe.es/buscar/pdf/2018/BOE-A-2018-16673-consolidado.pdf).

Por último, existen distintas leyes que regulan las sociedades limitadas, como en cualquier otra forma jurídica. Por ejemplo, nuestra S.L está regulada por la ley del Impuesto sobre el Valor Añadido(IVA), que puedes encontrar un documento con la ley [aquí](https://noticias.juridicas.com/base_datos/Fiscal/l37-1992.html), la ley que regula el Impuesto sobre sociedad, que puedes encontrarlo [aquí](https://www.boe.es/buscar/act.php?id=BOE-A-2014-12328), o el RD refundido de la Ley de Sociedades de Capital, que se puede encontrar [aquí](https://www.boe.es/buscar/act.php?id=BOE-A-2010-10544).

Tecnológico

En los últimos años, la tecnología está dando pasos agigantados. Hace una década, nadie se imaginaría que, al visitar un restaurante, el camarero que te trae los platos es un robot. Este salto no solo ha sido en la informática y en inteligencia artificial, muchos ámbitos han sido beneficiados de dicho salto, como puede ser el alimenticio. Desde nuevos robots de cocina hasta, lo nombrado anteriormente, un camarero robot son unos de los muchos avances que ha traído la tecnología a nuestras cocinas.

Para mi empresa, este avance de la tecnología es algo complejo, porque a la par que comenzamos a adquirir conocimientos de nuevos lenguajes, puede que aparezcan nuevos lenguajes de los cuales aprender. La parte buena de todo esto es que, conforme pasen el tiempo, seguiremos teniendo trabajo constante y empresas a las cuales ofrecer nuestro servicio.

Medioambientales

El cambio climático, la contaminación y las distintas causas medioambientales han supuesto un cambio de mentalidad mundialmente. Aunque muchos gobiernos no están tomando medidas, otros tantos las están tomando para frenar el calentamiento global. Centrado en la informática, esta no produce una cantidad notable de residuos si lo miramos de una manera general, pero existe alguna rama de la informática que sí que es contaminante. La rama de la cuál hablo es sobre el tema de criptomonedas. Estas se consiguen, o, mejor dicho, se minan mediante el uso de tarjetas gráficas. Mucha gente mantiene habitaciones repletas de tarjetas gráficas, las cuáles necesitan una cantidad considerable de energía y, debido a su uso tan continuado y de una manera tan fuerte, estas no duran tanto como en una situación normal, por lo que deben cambiarse frecuentemente. A consecuencia de esto, se genera mucha basura electrónica, la cual contienen productos químicos que se pueden filtrar al suelo provocando una contaminación en las zonas donde penetran. Se calcula, aproximadamente, que Bitcoin, la criptomoneda más importante del momento, ha aumentado severamente las emisiones contaminantes

Internacionales

En los últimos meses, la situación internacional es bastante cruda. Ucrania lleva algo más de un año en conflicto con Rusia. A consecuencia de esto, la UE está viviendo una crisis energética que afecta a todos los países centrales y del norte, dado que Rusia era su primera fuente de gas. A causa de esto, países del sur, como España o Italia, han tenido que reducir su consumo energético para que los países del norte, como Alemania, puedan disfrutar de un invierno sin ninguna dificultad. Además de la energía, este conflicto bélico ha hecho que las materias primas, como la comida, se hayan encarecido notablemente.

* + 1. Microentorno

Competidores

Tanto en España como en el mundo, el desarrollo software es un campo solicitado y, a su vez, tiene una oferta escasa. Aun así, existen distintas empresas grandes que se encargan de dar este servicio a otras grandes empresas, siendo el presupuesto de estas mucho mayor que las PYME. Aunque existen empresas que sí proporcionan estos servicios a un presupuesto más acertado para las PYME, creo que existe una ventana muy interesante y amplia para explotar. Además, esta situación creo que es perfecta dado la situación de mi empresa, una primeriza y con poca experiencia, por lo que mis precios se ajustan mucho mejor a los pedidos de las PYME.

Clientes

Como he comentado ligeramente en el apartado anterior, mi idea es centrar mi servicio en las pequeñas y medianas empresas. Dado que soy un empresario primerizo y con nula experiencia en el ámbito laboral, concretamente en el desarrollo software, mis precios y mis servicios no serán ni tan solicitados ni con tendrán una calidad tan extraordinaria como la que puede otorgar una empresa grande.

Conforme pase el tiempo y gane experiencia, tanto la calidad como el precio subirán poco a poco y, además de ofrecer mis servicios a PYME, podré contratar un equipo mejor con el que podré dar servicios a grandes empresas.

Proveedores

Antes de hablar sobre los proveedores, para dar contexto, el local será mi propia vivienda, por lo que los proveedores serán los necesarios para vivir en ella.

A continuación, voy a mostrar una tabla con las distintas necesidades que posee mi negocio.

|  |  |
| --- | --- |
| Necesidades | Proveedores |
| Luz | Iberdrola |
| Gas | Endesa |
| Agua | Aguas de Alicante |
| Comida | Distintos supermercados (Mercadona, Carrefour, Dia…) |
| Telefonía e Internet | Jazztel |
| Servidores | Amazon Web Services |
| Software | Microsoft, Google, etc.\* |

\*Este apartado puede cambiar debido a que mi empresa estará en constante aprendizaje, por lo que los distribuidores de los nuevos lenguajes de programación cambiarán

Productos sustitutivos

En mi empresa, realmente no existen productos sustitutivos como tal, dado que creamos un producto según el cliente quiere. La única parte que podría nombrarse como producto sustitutivo es a la accesibilidad de la aplicación, la cual depende de cada cliente. Con la accesibilidad, podemos ayudar a que gente que tiene ciertos problemas, como puede ser daltonismo o deficiencia visual, para que utilicen la aplicación sin complicaciones.

* + 1. Ubicación del negocio y plano del local

Mi negocio está en el centro de Benidorm, concretamente en la Avinguda Rei Jaume I. Es una zona muy buena dado que, al ser en el centro, puedo visitar a cualquier cliente sin problema. Además, aunque viviese a las afueras, es un trabajo que puede realizarse telemáticamente sin problema.

El plano de mi oficina será el siguiente.

Ubicación de la oficina

Plano de la oficina

* + 1. Análisis DAFO

El análisis DAFO es imprescindible a la hora de crear una empresa para observar, detenidamente, sus inconvenientes (debilidades y amenazas) y sus ventajas (amenazas y fortalezas).

* 1. Plan de marketing
     1. Producto

El producto, o más bien servicio, es la creación de aplicaciones para distintos dispositivos con los lenguajes de programación más actuales. Este servicio se ajusta a los requerimientos que el cliente quiera, por lo que el diseño y la funcionalidad de la aplicación depende del cliente. A su vez, también dependerá del cliente y de la situación actual el lenguaje de programación en el cual se desarrollará la aplicación.

* + 1. Precio

El precio también será algo variable según el tamaño del proyecto y su complicidad. Actualmente, un proyecto de 6 meses en una situación normal tendría un coste de 12000€ aproximadamente. Este cálculo está realizado mediante un cálculo aproximado de lo que podría costar un sueldo mensual de un desarrollador software con un añadido en el caso de que el proyecto acabe antes de tiempo o se demore un poco. Además, se realiza un primer abono del coste total, entre un 25% y un 50%. Esta medida es así para asegurar que el cliente no cancele dicho proyecto durante el proceso del mismo y dejar al empresario, en este caso mi persona, tirado y un ingreso por ese esfuerzo.

* + 1. Promoción

Muchas empresas están buscando desarrolladores para que creen una aplicación para el uso de sus clientes o de la propia empresa. Mi manera de llegar a estas empresas va a ser de distintas formas. La primera de ellas será mediante las redes sociales, como Instagram o Facebook. En Instagram, publicaré proyectos realizados en la empresa para que futuras empresas observen el trabajo que se ha realizado y, si les gusta, se lancen a pedirnos un producto. Por otro lado, en Facebook haremos algo semejante, pero añadiendo que contactaremos con las empresas. Muchas empresas tienen Facebook en la que publican distinta información del negocio. A partir de esta información, la analizaré y contactaré con ellos para sugerir un tipo de aplicación.

Por otro lado, añadiremos nuestra empresa a las páginas amarillas, de esta manera una multitud de empresas podrán observar nuestra información y podrán contactar con nosotros.

Estas medidas, al ser sencillas y qué, según mi parecer, puede hacerlo una única persona, yo seré el encargado de todas las redes sociales y el mantenimiento de las mismas, además de añadir a la empresa en las páginas amarillas, por lo que el costo de todo esto es nulo.

* + 1. Place/distribución

Nuestra vía para poder administrar nuestros servicios a nuestros clientes es íntegramente digital. Tanto en las páginas amarillas, como en redes sociales, tendremos toda nuestra información y demostraciones de nuestros servicios para que las empresas que vean nuestros perfiles puedan contactar con nosotros para un servicio personalizado.

* 1. Plan de producción y recursos humanos
     1. Inversiones y gastos

A la hora de iniciar una actividad emprendedora siempre hace falta un capital inicial. En este proyecto, tenemos un total de 5000€ ahorrados que nos ayudarán en la creación de la S.L. y con los gastos de los mismos. A continuación, analizamos los gastos y las inversiones de nuestro proyecto para saber diferenciarlos y tener un recuento del capital que gastamos. En la siguiente tabla diferenciamos entre inversiones y gastos.

|  |  |
| --- | --- |
| Inversiones | Gastos |
| Hardware (2.400€) | Alquiler (6.000€/a) |
|  | Luz (960 €/a) |
|  | Comida (1440€/a) |
|  | Agua (240 €/a) |
|  | Gas(600€/a) |
|  | Telefonía e Internet (480 €/a) |
|  | Seguro (1.000 €/a) |
|  | Comunidad (Se encarga el casero) |
|  | Gastos de constitución (500 €) |
|  | Sueldo propio (18.000€/a) |
|  | SS administrador(960 €/a) |
|  | Servidores (100€/a) |
|  | Licencias software(100€/a) |
| TOTAL: 2400 € | TOTAL: 30280€ |
| TOTAL: 32680€ | |

* + 1. Origen del financiamiento

Como hemos comentado en apartados previos, el origen de nuestro financiamiento inicial es, en su totalidad, propio. Gracias a mis propios ahorros que he ido consiguiendo con trabajos temporales, tenemos la libertad de no pedir un préstamo bancario para comenzar nuestra actividad.

* + 1. Ayudas y subvenciones

Para el comienzo de todas las empresas, las ayudas son más que importantes. Hay de todo tipo de ayudas, como la ayuda para jóvenes emprendedores o para mujeres emprendedoras. En el caso de nuestra empresa, hemos decidido obtener la bonificación en la SS para las altas iniciales. Esta ayuda lo que nos otorga es una cuota mensual de 80€ en el apartado de SS, lo que supone una reducción más que notable del total que deberíamos pagar.

* + 1. Distribución de costos

A continuación, debemos diferenciar entre gastos fijos y variables.

|  |  |
| --- | --- |
| FIJOS | VARIABLES |
| Alquiler | Luz |
| Telefonía e Internet | Agua |
| Seguro | Comida |
| Gasto de constitución | Gas |
| Sueldo propio |  |
| SS propio |  |
| Servidores |  |
| Licencias |  |

* + 1. Umbral de rentabilidad o punto muerto

Antes de ver el umbral de rentabilidad, debemos saber qué es exactamente. El umbral de rentabilidad o punto muerto representa el momento en el que existe el equilibrio entre los ingresos y gastos totales, es decir, el punto en el que la empresa no pierde dinero, pero tampoco gana. El umbral de rentabilidad de nuestra empresa es el siguiente.

Como se puede observar en el anterior gráfico, nuestro punto muerto se encuentra entre el quinto y sexto mes, debido a que en esa fecha se realiza el segundo pago del proyecto. A partir del sexto mes, todos los ingresos pasan a ser ingresos netos.

* 1. Trámites de constitución, puesta en marcha e impuestos

|  |  |  |
| --- | --- | --- |
| **INSTITUCIÓN** | **TRÁMITES** | **IMPUESTOS/TASAS** |
| Ayuntamiento | Licencia de actividades e instalaciones | Gratuito |
| Licencia de apertura | Depende del ayuntamiento |
| Hacienda | Solicitud NIF temporal | Gratuito |
| ITPAJD → Hacienda autonómica | Gratuito |
| NIF definitivo/Inicio habitual de actividades | Gratuito |
| Declaración censal | Gratuito |
| Alta en el IAE | Gratuito solamente el primer año |
| TGSS | Alta al representate/s en el régimen correspondiente de la SS | Gratuito |
| Código de Cuenta de Cotización de la empresa | Gratuito |
| Registros oficiales | Certificación negativa del nombre en el Registro Mercantil | 14 € |
| Legalización de los libros de cuentas anuales | Gratuito |
| Inscripción en el RGPD | Gratuito |
| Notaría | Estatutos + Escritura constitución | 462 € |
| Otros | Certificación del cargo de administrador o certificado digital => FNMT | 24 € |

1. Análisis

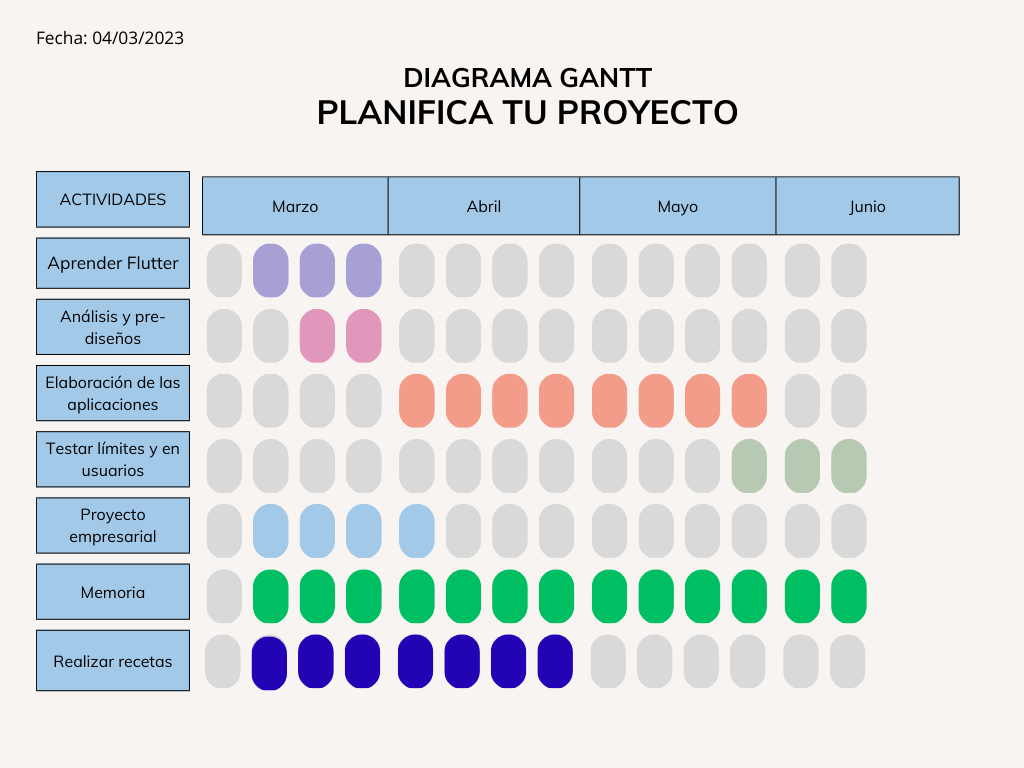
En el siguiente apartado nos centraremos en el análisis previo que ha de hacerse siempre a la hora de desarrollar un proyecto.

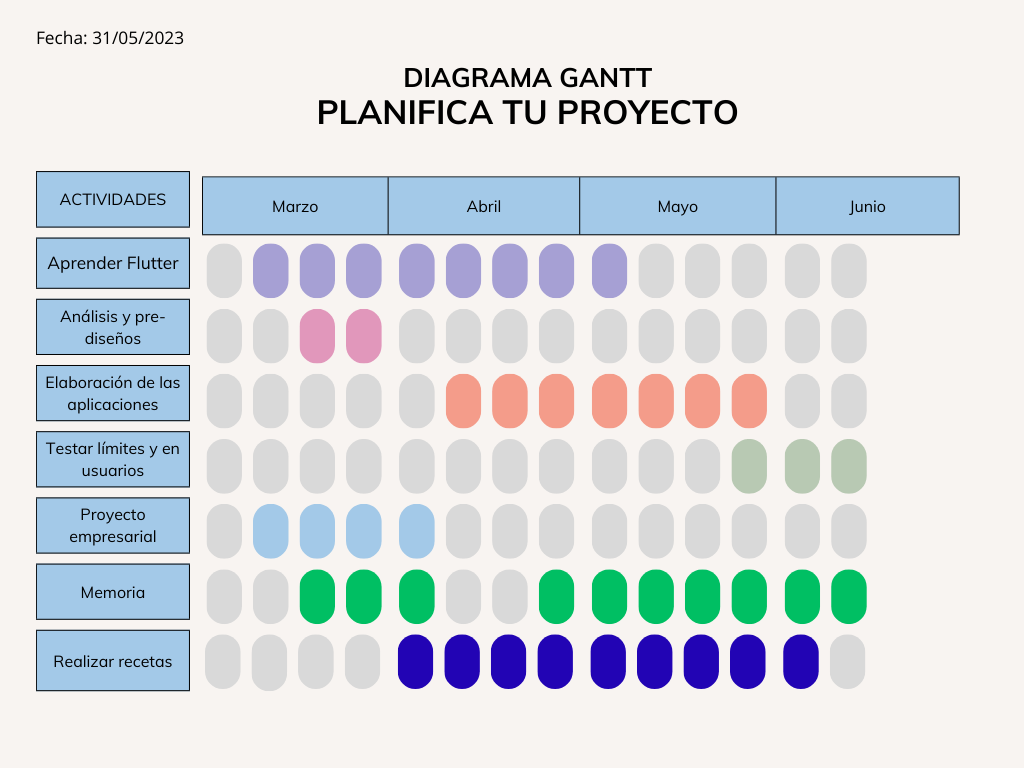
A continuación, desarrollaremos distintos puntos, desde una planificación inicial, con el Diagrama de Gantt, hasta casos de usos y diagrama de clases.

* 1. Planificación temporal

La planificación temporal es un apartado más que importante a la hora de desarrollar un proyecto de cualquier tipo. De esta manera, podemos tener un control sobre las acciones que realizamos durante el mismo. Además, realizar esta planificación nos puede ayudar en futuros proyecto a la hora de organizarlo mejor en el tiempo.

Uno de los procedimientos más conocidos y utilizados en este ámbito es el “Diagrama de Gantt”. Este diagrama te proporciona una vista general de las tareas programadas. Además, muestra datos de interés como la fecha de inicio y final del proyecto, en cuántas tareas está dividido el proyecto, una estimación aproximada de cuánto tiempo se llevará a cabo en cada tarea y la manera en la que las tareas están relacionadas entre sí.

A continuación, mostraré el diagrama de Gantt realizado al comienzo de este proyecto.

Tras finalizar el proyecto, he realizado otro diagrama de *Gantt* para poder comparar las expectativas, mostradas en el diagrama anterior, con el resultado real y final, que es el siguiente diagrama.

Se pueden observar ciertas diferencias entre diagramas. La primera diferencia, y la más grande, es sobre el aprendizaje de *Flutter.* En un principio, el aprendizaje de *Flutter* fue rodado, pero conforme pasó el tiempo fue más tranquilo.

La segunda diferencia, prácticamente igual de notable que la anterior, es la realización de las recetas del menú, las cuáles han sido movidos en el tiempo, dado que he tardado algo más de lo esperado tanto realizando las recetas como pensando el menú.

Por último, pero no más importante, ha habido ligeros cambios tanto en la memoria, con la que no he sido del todo constante, y la elaboración de las aplicaciones, que he tardado algo menos de lo esperado.

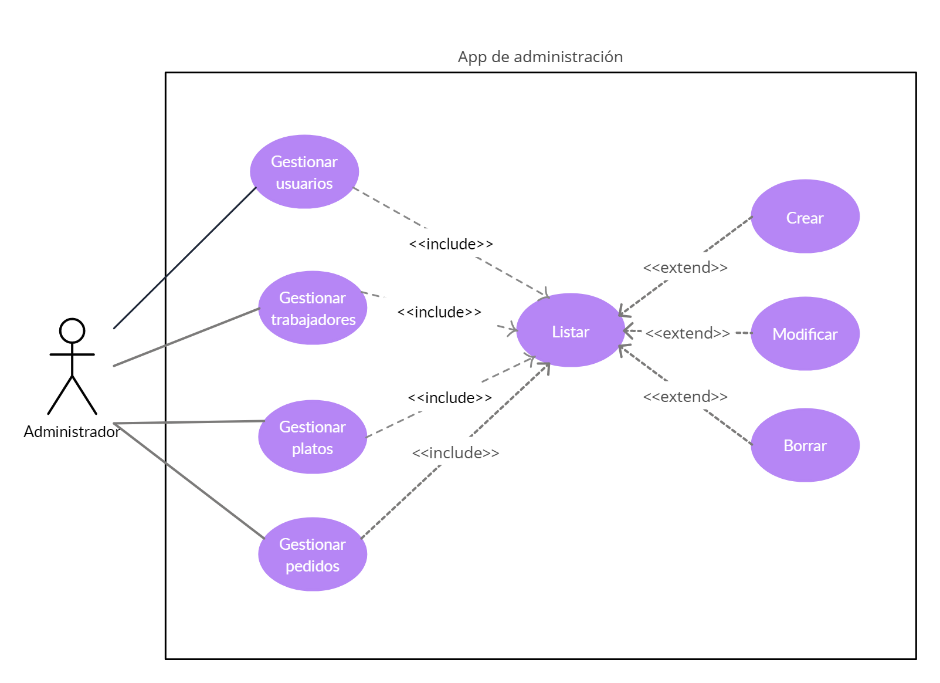
* 1. Casos de uso

Antes que nada, se debe definir correctamente que es un caso de uso. Como su nombre indica, un caso de uso es una secuencia de acciones que debe realizar alguien o algo, llamado actor, para llevar a cabo un proceso.

Como en cualquier aplicación, antes de su desarrollo se deben crear distintos casos de usos ajustado al, valga la redundancia, su uso. Por ello, he realizado los distintos casos de uso para las aplicaciones que desarrollaré.

Para comenzar, desarrollaremos los casos de uso de la aplicación orientada en el uso administrativo.

* + 1. Aplicación administrativa



Caso de uso – Gestionar usuarios, trabajadores y platos

Antes de empezar en sí con este caso de uso, agruparemos las tres acciones que puede realizar el usuario administrador en nuestra aplicación porque realmente es la misma acción, únicamente cambia la entidad a la que hace efecto.

*Descripción*: El administrador desea gestionar una entidad, ya sea para crear, borrar o modificar.

*Actor*: Administrador.

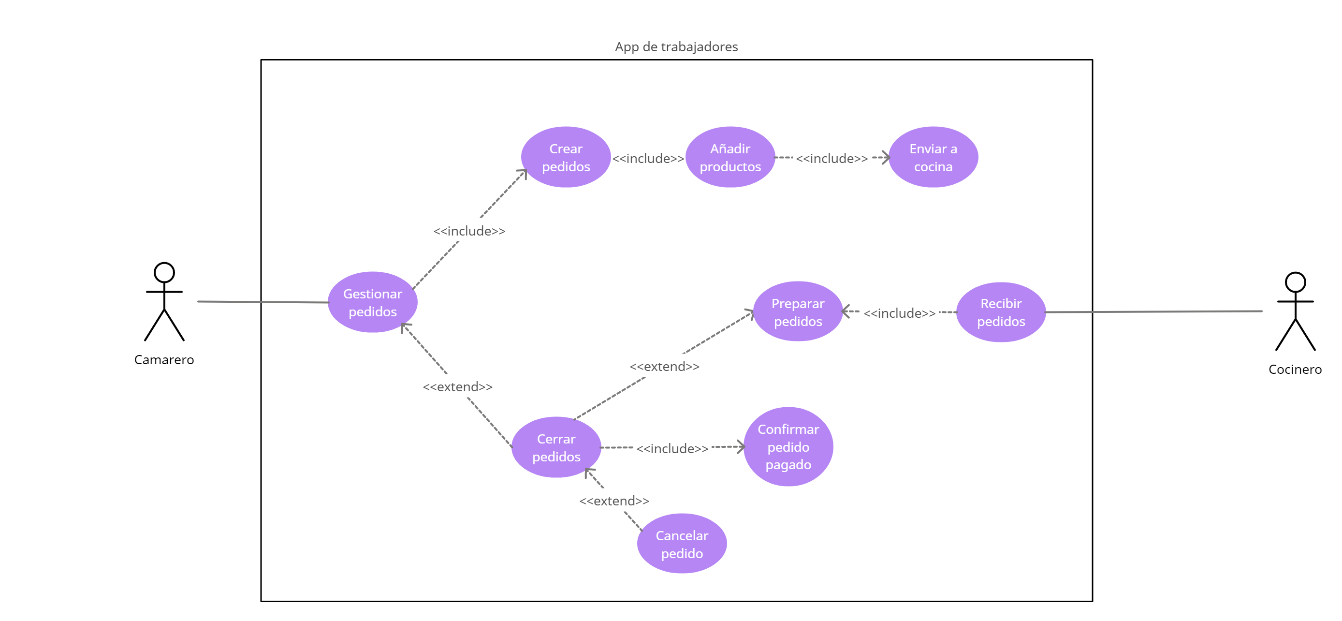
*Precondiciones*: El usuario sea ha registrado y tiene el rol “Administrador”.

*Curso normal de la acción:*

1. El administrador accede a la pestaña correspondiente a la entidad seleccionada.
2. La aplicación lista todas las entidades de la base de datos y el administrador elige una opción (Crear, Borrar o Modificar).
3. Si elige *Crear*, existirán los siguientes sucesos:
   1. El administrador introduce los datos de la nueva entidad.
   2. El sistema comprueba si los datos son correctos.
   3. Ingresa la nueva entidad en la base de datos.
4. Si elige *Modificar*, ocurrirá lo siguiente:
   1. El administrador introduce los nuevos datos.
   2. El sistema comprueba si los datos son correctos.
   3. Actualiza la entidad en la base de datos.
5. Si elige *Borrar*, los sucesos serán los siguientes:
   1. El sistema comprueba que esa entidad exista actualmente
   2. El sistema elimina toda la información relacionada con esas entidad en concreto.

*Errores/Alternativas:*

1. El sistema muestra, con una ventana emergente, que los datos son incorrectos, ya sea porque no cumplen un formato o porque ya están en uso.
2. El sistema muestra, con una ventana emergente, que los datos son incorrectos, ya sea porque no cumplen un formato o porque ya están en uso.
3. Algún dato no se borra correctamente.
   * 1. Aplicación trabajadores



Caso de uso – Camarero

*Descripción*: El camarero gestiona los pedidos.

*Actor*: Camarero

*Precondiciones*: El usuario tiene que haberse registrado y tener el rol de “Camarero”

*Curso normal de la acción:*

1. El camarero gestiona los pedidos
2. El camarero crea un pedido
3. El camarero añade productos según quiera el cliente
4. Por último, el camarero envía el correo a cocina.

*Errores/Alternativas:*

2. Si hay pedidos ya realizados, el camarero puede cerrar un pedido.

2.1 Debe comprobar si el pago está pagado correctamente.

2.2 Puede cancelar el pedido si este no ha sido pagado correctamente o el cliente desea cancelarlo.

Caso de uso – Cocinero

*Descripción*: El cocinero recibe y prepara pedidos.

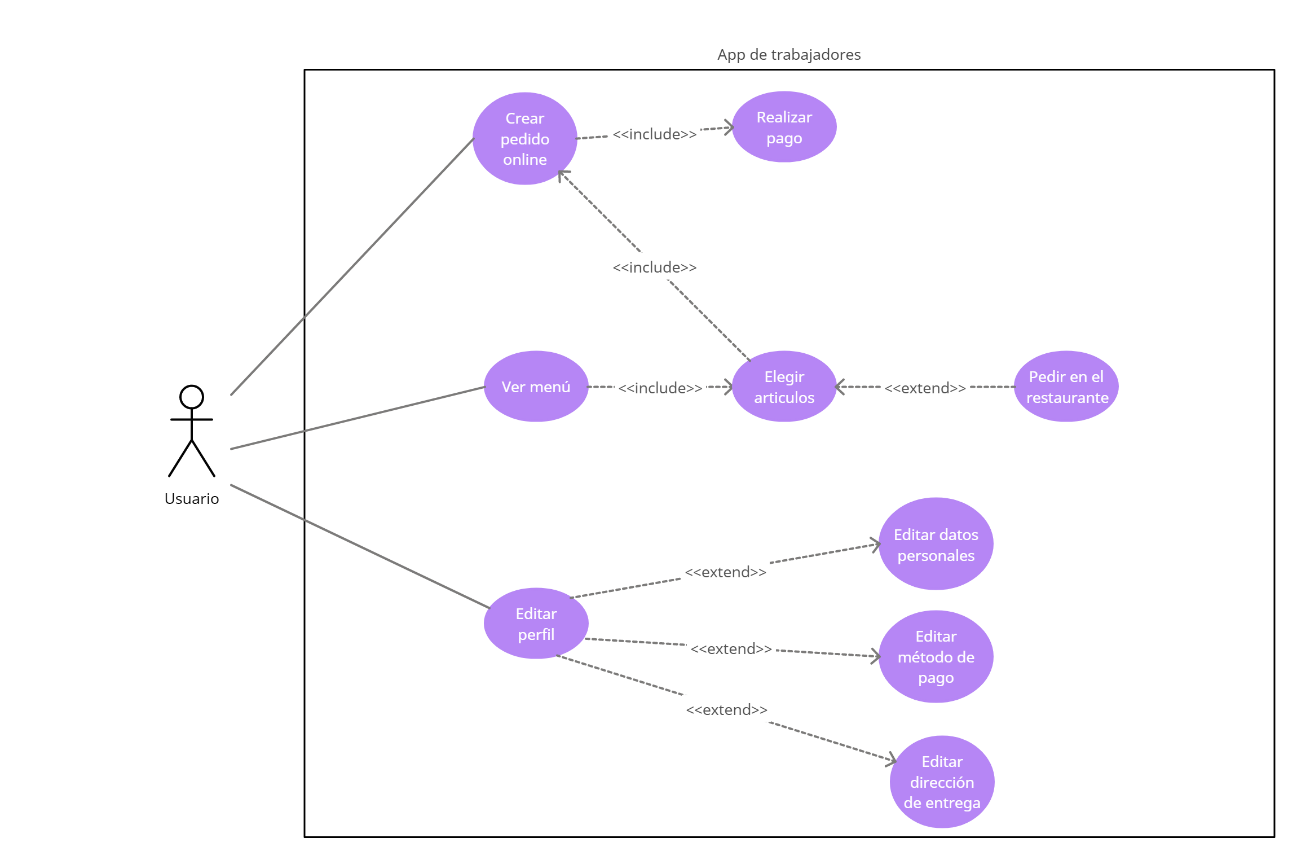
*Actor*: Cocinero.

*Curso normal de la acción:*

1. El cocinero recibe pedidos.
2. Tras esto, los prepara.
3. Cuando el pedido está acabado, lo finaliza.
4. En este caso, no es necesario comprobar si el pago es correcto, dado que cuando llega a cocina este tiene que ser correcto.

*Alternativas/Errores: Nada.*

* + 1. Aplicación clientes



Caso de uso – Usuario- Crear pedido online

*Descripción*: El usuario crea un pedido online

*Actor*: Usuario

*Precondiciones*: El usuario debe de estar registrado con el rol de “Usuario”

*Curso normal de la acción:*

1. El usuario empieza a crear el pedido
2. Elige artículos para su pedido
3. Paga dicho pedido y se envía a su domicilio.

*Alternativas/Errores:*

3. Error en el pago, por lo que se cancela el pedido

Caso de uso – Usuario- Ver menú

*Descripción*: El usuario observa el menú.

*Actor*: Usuario

*Precondiciones*: El usuario debe de estar registrado con el rol de “Usuario”

*Curso normal de la acción:*

1. El usuario observa el menú
2. A continuación, elige los productos que desea
3. Le comunica los productos que desea al camarero

*Errores/Alternativas:*

3. En el caso de comunicarle los productos al camarero, crea un pedido y los añade a él.

Caso de uso – Usuario- Editar perfil

*Descripción*: El usuario desea cambiar sus datos personales.

*Actor*: Usuario.

*Precondiciones*: El usuario debe de estar registrado con el rol de “Usuario”.

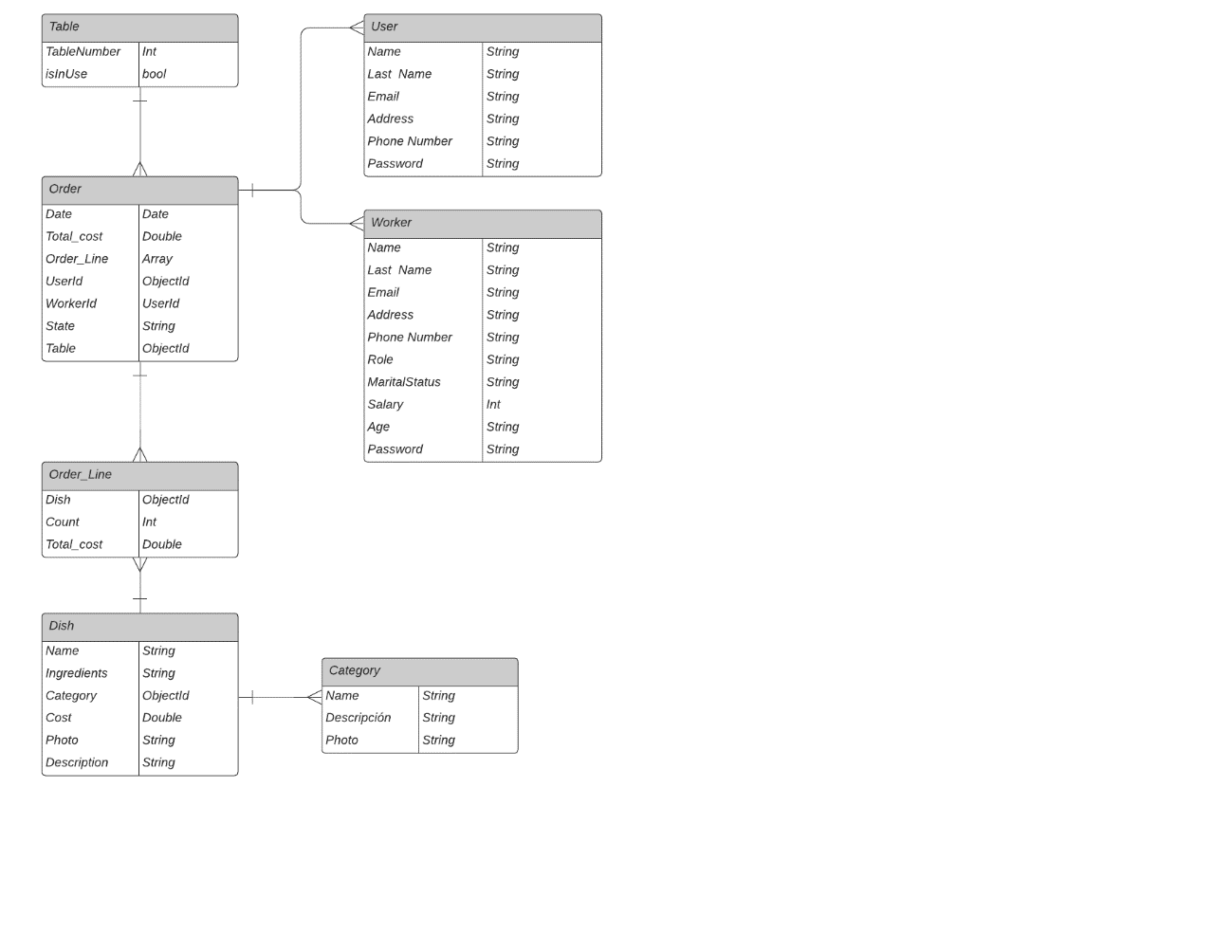
*Curso normal de la acción:*

1. El usuario selecciona su perfil.
2. Selecciona el dato que quiera cambiar.
3. Depende de lo que seleccione, podrá realizar los siguiente:
   1. Si elige datos personales, podrá cambiar datos como el correo, el teléfono móvil o su nombre y apellidos.
   2. Si elige el método de pago, podrá cambiar los datos de la tarjeta guardada o añadir una.
   3. Si elige la dirección de envío, podrá cambiar o añadir la dirección donde se enviará su pedido.

*Alternativas/Errores:*

Los datos ya están en uso o son incorrectos.

* 1. Diagrama de clases
  2. Modelo relacional

Para finalizar el apartado del análisis del proyecto, tenemos el modelo relacional de la base de datos. Como podemos observar a la derecha del documento, vemos las colecciones de la base de datos junto a sus campos y los tipos. En cada documento, además, hay un campo obligatorio en todas las colecciones, que es el id del documento, generado de manera automática por MongoDB. Sabiendo esto, las referencias que existen de otras colecciones, como puede ser en la colección *order*, son del tipo *ObjectId*.

1. Entorno de programación

En el punto 4 de esta memoria se va a exponer y explicar los distintos lenguajes y librerías que se van a emplear en el desarrollo de todo el software. Además, hablaremos ligeramente del control de versiones y cómo lo hemos empleado.

* 1. Lenguajes y tecnologías que vamos a utilizar
     1. Base de datos

Mongo

La base de datos elegida para este proyecto ha sido *MongoDB*, una base de datos no relacional. En un principio, existían dudas sobre qué tipo de base de datos se adaptaría mejor a este tipo de proyecto.

Por un lado, una base de datos relacional nos proporcionaba una estructura rígida que nos permite ser menos vulnerable a fallos. La parte negativa de utilizar este tipo de base de datos es que no tenía, a comienzo de dicho proyecto, conocimientos suficientes para enlazar el proyecto de backend, del cual se hablará más adelante, con la base de datos.

Por otro lado, las bases de datos no relacionales nos permiten tener más flexibilidad a la hora de albergar datos y que, al contrario que con las bases de datos relacionales, sí tengo conocimientos para enlazar el backend, desarrollado en *Node.js*, y una base de datos como *MongoDB*.

*MongoDB* tiene distintas diferencias frente a las bases de datos relacionales como *MySQL*. Esta base de datos, y todas las bases de datos no relacionales, se divide en “colecciones”, las cuales son equivalentes a las tablas en base de datos relaciones. Cada una de estas tablas, tiene “documentos” los cuales albergan atributos con sus respectivos datos. Al contrario que en *SQL*, en *MongoDB* generalmente no hay referencias a otras tablas, el propio documento suele albergar estos datos, por lo que para aplicaciones donde sea necesario guardar datos de manera “masiva” y sin restricciones es perfecta.

Al analizar todos estos pros y contras de ambos tipos de bases de datos, mi proyecto puede utilizar cualquiera de las dos bases de datos, incluso ambas. La razón que inclinó la balanza a favor de MongoDB es la experiencia que tenía desarrollando APIs en *Node.js* y *MongoDB* y la facilidad que te otorga desarrollar APIs en *Node.js* junto a *Express*, un entorno que se explicará en la próxima sección.

* + 1. Backend

Para comenzar esta sección, debemos saber diferenciar entre *frontend*, que definiremos más adelante, y el *backend*. El *backend* de una aplicación es la parte no visible, la cuál se encarga de gestionar los pedidos, confirmaciones de pago, etc. Esta parte, además, se encarga de la conexión del *frontend* y sus acciones con la base de datos para qué, de manera unitaria, todas las aplicaciones desplegadas tengan los mismos datos.

Este *backend* está desarrollado en *Node.js,* un entorno basado en *JavaScript,* y Express, un entorno enfocado en el desarrollo de aplicaciones web o APIs, el cuál es nuestro caso.

A continuación, se desarrollará de manera más concreta tanto los entornos anteriormente nombrados, *Node.js* y *Express*, como las librerías empleadas en el desarrollo de este *backend*.

Node.js y Express

*Node.js* es un entorno de ejecución orientado a eventos asíncronos, el cuál está enfocado para crear aplicaciones web escalables. Una de las características más destacables de este entorno es su enfoque en el modelo E/S sin bloqueo. Además, Node.js utiliza un único hilo de ejecución que maneja solicitudes de manera paralela. Por último, también cuenta con un sistema de gestión de paquetes, llamada *npm*, que facilita la instalación e incorporación de nuevos módulos y librerías a nuestro proyecto. De hecho, debemos utilizar este sistema de gestión para instalar *Express* y las demás librerías que explicaremos más adelante.

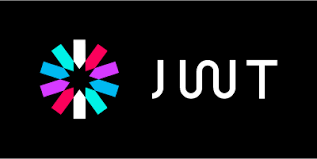
*Express*, abreviación de *Express.js*, es un *framework* el cuál se utiliza para el desarrollo de APIs de manera eficiente, flexible y modular. Permite a los desarrolladores crear APIs de manera rápida y sin una estructura rígida. Además, simplifica algunas tareas como el enrutamiento, gestión de sesiones, etc. y permite una alta compatibilidad con otro módulos y librerías de *Node.js*.

Tras explicar los dos pilares de nuestro backend, entraremos más a fondo sobre las librerías utilizadas y qué funcionalidades tiene.

Librerías utilizadas

El siguiente apartado trata de analizar las librerías que he utilizado en este proyecto. Las librerías mas destacables serían *bcrypt*, *mongoose*, *jwt-simple* y *socket.io*, dado la importancia que tienen en el proyecto, pero se analizarán todas las librerías.

La primera de ellas, y la más importante de nuestro backend, es *mongoose*. Esta librería, como parte de su nombre indica, permite la conexión entre el backend y nuestra base de datos. *Mongoose* es una librería esencial en este proyecto dado que, sin ella, no existiría una uniformidad en todas nuestras aplicaciones de manera dinámica. Además de permitirnos extraer e introducir documentos en nuestras distintas colecciones, nos facilita mucho el trabajo a la hora de hacer algunas peticiones más complejas.

Otra librería muy importante para el desarrollo del proyecto es *jwt-simple,* una librería que nos añade una capa de seguridad más que necesaria, dado que nos permite codificar y decodificar los *json web token*, los cuáles nos permite autorizar y denegar el acceso a nuestros datos. Esta librería, junto a la siguiente, son muy necesarias si queremos un *backend* seguro.

La siguiente librería, junto con la anterior, son las dos más importantes si hablamos de seguridad. En este caso, hablamos de *bcrypt*, una librería que nos permite encriptar datos sensibles para albergarlos en la base de datos, como puede ser la contraseña del usuario. De esta manera, si atacan nuestra base de datos, algo tan importante como es la contraseña no puede ser robada.

La última librería a destacar de nuestro backend es *socket.io*. Esta librería, en un principio, no iba a incluirse en el proyecto, pero al final es necesaria. *Socket.io* nos permite crear una conexión constante con un cliente mediante eventos, por lo que se puede mantener una conexión a tiempo real con los datos. Este socket lo utilizamos a la hora de la creación de pedidos para que, sin que el usuario tenga que realizar ninguna acción, se actualice el estado del pedido y para que los cocineros reciban los pedidos al momento.

Tras comentar las cuatro librerías más importantes de nuestra API, toca explicar las demás, que no son tan importantes, pero son necesarias. Son la siguientes:

* *DotEnv*. Esta primera librería nos permite tener variables de entorno, las cuales nos permiten guardar datos sensibles, como puede ser la dirección IP de la base de datos o el token secreto.
* *Nodemon*. Esta librería se podría catalogar como la prescindible, dado que únicamente nos ayuda a la hora de desarrollar a la API, permitiéndonos usar el *hot reload* para aplicar los cambios sin tener que reiniciar el programa.
* *Moment*. *Moment* es una librería que nos ayuda a trabajar con fechas, como puede ser validándolas o transformándolas.
* *Express-validator*. Esta es una de las más importantes de estás cuatro librerías. Nos permite añadir un middleware entre la autenticación, gestionada con *jwt-simple*, y la función principal de la petición con la cuál podemos crear validaciones para rechazar o aceptar dichas peticiones.

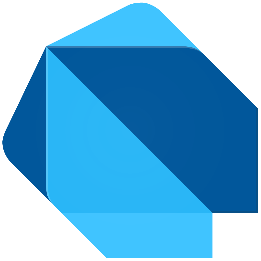
Ya explicado toda la parte del *backend*, pasamos a explicar la parte más grande del proyecto, el *frontend*.

* + 1. Frontend

El *frontend* es la capa con la que el usuario interactúa y todo lo que ello conlleva, como son los diseños, funcionalidad que se encarga de la interactividad con los usuarios, etc.

En el caso del proyecto *Sei Vegan, Sei Grün*, el *frontend* está desarrollado únicamente en el *framework* *Flutter*, con base del lenguaje de programación *Dart*, los cuales van a ser explicados en el siguiente punto.

Dart

*Dart* es un lenguaje de programación de código abierto desarrollado por Google, publicado por primera vez en 2011. Este lenguaje, en sus inicios, era una alternativa para JavaScript, siendo Dart un lenguaje orientado a objetos (POO) y de tipado estático, un estilo parecido a Java. Actualmente, Dart puede ser utilizado en distintos ámbitos, como aplicaciones web, aplicaciones multiplataforma o servidores.

Dart, como hemos comentado de manera leve anteriormente, es un lenguaje de programación con similitudes con JavaScript, Java o C++. Por lo tanto, al tener esta base tan similar con lenguajes tan conocidos, a los nuevos estudiantes, en los que me incluyo, se les hace muy sencillo aprender de él.

Flutter

Como hemos nombrado anteriormente, vamos a utilizar el *framework* *Flutter*, basado en un único lenguaje, *Dart*. A diferencia de otros frameworks de otros lenguajes o del propio Dart, Flutter compila a código nativo, consiguiendo así un rendimiento mayor que en otras aplicaciones basadas en web-views.

El mayor potencial, y unas de las razones por las que se ha elegido este SDK, es la facilidad en la que se pueden crear aplicaciones en distintas plataformas con un único código. Flutter permite crear aplicaciones nativas para *Android*, *iOS*, aplicaciones web y aplicaciones en *Windows*, *Linux* y *MacOS*. Además, también tiene integrado el *Hot Reload*, una característica muy útil a la hora de desarrollar. A continuación, indagaremos en características de *Flutter*, y a su vez de *Dart*, y veremos qué ventajas tiene.

La primera ventaja de la cual vamos a hablar es sobre su rápido desarrollo. Cómo hemos avanzado anteriormente, *Dart* compila en lenguaje 100% nativo, por lo que su velocidad de compilado es una de las más óptimas actualmente. Además de todo esto, tiene implementado la función *Stateful Hot Reload*, que te permite ver, de forma instantánea, los cambios realizados con la aplicación corriendo.

Otra de las ventajas que tiene *Flutter* es tener una amplia librería de widgets. Los widgets son los componentes que conforman nuestra aplicación. Estos componentes, mediante clases y *builders*, pueden hacerse reutilizables y personalizables, algo muy interesante a la hora de optimizar y ahorrar código. Algunos de los *widgets* más básicos pueden ser:

* *Text* Nos permite ver texto
* *Appbar/TabBar* Son la parte superior e inferior de la aplicación.
* *Button* Componente qué, al dar click, realiza una función.

Los *widgets*, además, tienen dos tipos: Con estado y sin estado. El estado es la información que se puede utilizar una vez se crea el *widget* y que cambie durante su vida útil. Dicho estado puede ser administrado/controlado de distintas maneras, como puede ser con un *Stateful* *Widget* o con gestores de estado, como son *Provider* o *Bloc*. A continuación, vamos a hablar de los más conocidos.

Provider

*Provider* es, posiblemente, el gestor de estado más conocido de *Flutter* junto a *Bloc*. Este gestor es muy fácil de implementar, lo cual no significa que sea incompleto o perfecto para aplicaciones pequeñas. *Provider* no tiene nada que envidiarles a gestores como *Bloc*, *GetX* o *Cubic*. Este gestor sirve tanto para aplicaciones pequeñas como medianas/grandes, aunque para aplicaciones grandes es más recomendado *Bloc*.

Como gestor de estado, nos permite administrar el estado de las pantallas y widgets de nuestra pantalla. Además, podemos crearlo nada más se ejecuta la aplicación, por lo que tendremos acceso a la instancia de nuestro provider en cualquier lugar de nuestro árbol de widgets. Esto, además de proporcionarnos mucha facilidad a la hora de centralizar nuestros datos, nos permite acceder a datos entre páginas sin perder el estado de las mismas. En mi opinión, y viendo la magnitud del proyecto, Provider es el gestor de estados perfecto para ello, aunque nunca se debe de descartar el cambio hacia Bloc, otro gestor de estado la mar de interesante. A continuación, hablaremos de él.

Bloc

Anteriormente, hemos nombrado Bloc como un gestor de estado, pero se puede nombrar a *Bloc* como una parte del patrón de *Clean Architecture*. Si se tiene conocimientos de algún tipo de *Clean Architecture*, *Bloc* es algo parecido a *MVVM*. El concepto principal es que exista una capa entre el modelo y la vista, justamente una de los conceptos que sustenta *MVVM*. Además, en esta capa se gestionarán, además de los datos, los estados de los widgets.

El patrón Bloc se puede dividir en cuatro importantes partes; Events, States, Bloc y UI. Del primero que vamos a hablar es de los estados, dado que es similar al estado de los widgets. Un estado en el patrón *Bloc* almacena las condiciones en las que se encuentra nuestra pantalla o aplicación, al igual que en los widgets guarda las condiciones de los mismos. Esto nos puede ayudar a la hora de mostrar un contenido u otro dependiendo del estado de la aplicación, como puede ser el mostrar un texto si ha saltado un error o una barra de carga mientras se obtienen los datos desde la API.

Otro de las partes de *Bloc* que va de la mano con los estados son los eventos. Estos eventos suelen ser lanzados cuando la pantalla o aplicación tienen un estado en concreto. Dicho de otra manera, y para enlazar con el apartado anterior, cuando un estado se establece en la pantalla, este puede lanzar un evento que ejecutará una acción. Poniendo por caso el ejemplo anteriormente nombrado sobre los errores, la pantalla debería cambiar de estado y, nada más lanzar dicho estado, se ejecutará un evento que está ligado a este estado.

Como tercer componente del patrón *Bloc*, tenemos el propio *Bloc*. Este componente es el encargado de manejar los estados de la aplicación y sus eventos, o, dicho de otra manera, es el gestor de estados de la aplicación. Dicho componente es la unión, por decirlo de alguna manera, entre los estados y los eventos y el usuario.

Para este proyecto, concretamente para la aplicación de los clientes, *Bloc* como gestor de estados hubiese sido perfecto, dado que se una aplicación mediana en la que la implementación de dicho gestor hace que el desarrollo de la misma sea más sencillo. Además, si a este patrón le añadimos la librería *injector*, librería que nos permite inyectar dependencias, obtenemos un gestor maravilloso. Aun teniendo estas ventajas, descubrí demasiado tarde este gestor de estados y esta librería, las cuales, y como he dicho antes, ahorran mucho trabajo. De todas formas, desarrollar un proyecto como este con un gestor de estados como Provider no ha resultado una aventura extremadamente difícil.

Librerías utilizadas

Tras comentar y analizar los gestores de estado que se podrían haber utilizado en este proyecto, toca dar paso a analizar y explicar qué librerías hemos utilizado en el proyecto y el porqué. Antes de comenzar a nombrarlas, me gustaría aclarar que no todas las librerías están en todas las aplicaciones, la siguiente lista es un conjunto de todas las librerías usadas.

La primera librería de la que vamos a hablar es una alternativa a las *snackbar* que vienen por defecto en *Flutter*. Dicha librería se llama [*another\_flushbar*](https://pub.dev/packages/another_flushbar) y nos permite notificar al usuario, de una manera más fluida y personalizable que si utilizamos las *snackbar* o *toast*. De todas maneras, he utilizado tanto las que vienen por defecto como las de este paquete para no tener únicamente de un tipo y poder cambiar dependiendo del usuario final.

La siguiente librería, llamada [*auto\_size\_text*](https://pub.dev/packages/auto_size_text)*,* nos ayuda a la hora de crear una aplicación *responsive*. Este paquete nos proporciona unas características que con un *widget* de texto nativo no podríamos. Por ejemplo, nos permite ajustar un texto a un número de líneas y se irá adaptando de manera dinámica a esta restricción. Además, si el texto sobrepasa estos límites, podemos añadir un widget para sustituir a dicho texto. En este proyecto, esta librería se ha utilizado junto a [*marquee*](https://pub.dev/packages/marquee)*,*  una librería que permite a texto desplazarse de manera infinita, obteniendo un efecto muy bueno.

Este tercer paquete es exclusivo de las plataformas de escritorio, como son *Linux*, *MacOS* o *Windows*. Dicho paquete es [*bitsdojo\_window*](https://pub.dev/packages/bitsdojo_window) y nos permite añadir y personalizar una barra superior y añadirle los botones deseados, como el de minimizar, maximizar y cerrar. Esto nos ayuda en el diseño de la aplicación, dado que la barra superior que viene por defecto con *Flutter* no es muy agradable a la vista.

Los dos siguientes paquetes están únicamente añadidos en la aplicación de administrador, los cuales son [*cloudinary\_sdk*](https://pub.dev/packages/cloudinary_sdk) y [*file\_selector*](https://pub.dev/packages/file_selector). Por un lado, tenemos el primer paquete que está enfocado en subir archivos a la plataforma *Cloudinary*. Esta plataforma es la que he seleccionado para la administración de imágenes. Por otro lado, *file*\_*selector* nos permite abrir un selector de archivos, en nuestro caso fotos, y almacenarlo en una variable.

Siguiendo el tema de las imágenes, tenemos el paquete [*cached\_network\_image*](https://pub.dev/packages/cached_network_image), el cuál nos ayuda a ahorrar recursos a la hora de obtener las fotos desde *cloudinary*. El widget que nos proporciona este paquete guarda en la memoria caché las fotos que sean cargadas desde la red, ahorrando así ancho de banda cuando se vuelva a cargar el *widget*.

Este ahorro de ancho de banda que acabamos de comentar, se nota sobretodo en widgets como el que nos proporciona [*carousel\_slider*](https://pub.dev/packages/carousel_slider), un widget que genera una lista deslizable automática, la cuál podremos observar en la aplicación de los clientes. Además de facilitarnos la creación de un deslizable automático, también nos permite personalizar en gran medida dichas funciones, como la duración de la animación, el estilo de animación o el tiempo de espera entre animaciones.

En lo referente a las animaciones, tenemos que hablar de las dos librerías utilizadas para esto, que son [*simple\_animations*](https://pub.dev/packages/simple_animations) y [*animate\_do*](https://pub.dev/packages/animate_do). Estas dos librerías han sido utilizadas para la mayoría de animaciones de este proyecto. Para concretar, *simple\_animations* es utilizado en la animación del inicio de sesión y del registro y *animate\_do* es utilizado en las demás animaciones.

En relación con los gestores de estado y todo lo utilizado en ellos, tenemos los paquetes [*provider*](https://pub.dev/packages/provider)y [*http*](https://pub.dev/packages/http). El primero de ellos nos provee, nunca mejor dicho, del gestor de estados *Provider*. El segundo de ellos, *http*, nos ayuda con cualquier conexión http, en nuestro caso nos ayuda con la conexión hacía la API. Además, si hablamos de peticiones, debemos mencionar como almacenamos el JWT, es decir, el token. Aquí es donde entra en acción [*flutter\_secure\_storage*](https://pub.dev/packages/flutter_secure_storage), una librería que nos permite guardar datos sensibles de manera segura.

Los próximos paquetes son del mismo creador, pero están separados para que la importación del paquete no sea muy pesada. Estamos hablando de los *widgets* de [*Syncfusion*](https://www.syncfusion.com), una empres especializada en crear componentes en distintos lenguajes de programación, como *WPF*, *Flutter* o *MAUI*. Para este proyecto, han sido necesarios tres de tantos que nos proporcionan, los cuáles son:

* [*syncfusion\_flutter\_charts*](https://pub.dev/packages/syncfusion_flutter_charts), especializado en los gráficos.
* [*syncfusion\_flutter\_datagrid*](https://pub.dev/packages/syncfusion_flutter_datagrid), especializado en tablas de datos.
* [*syncfusion\_flutter\_datepicker*](https://pub.dev/packages/syncfusion_flutter_datepicker), especializado en los calendarios.

Todos ellos han sido utilizados en la aplicación del administrador, para proporcionar al dueño del restaurante una experiencia satisfactoria. Además, junto al último paquete de estos tres, ha sido utilizada la librería [*flutter\_localizations*](https://pub.dev/packages/flutter_localization), el cuál nos permite traducir, de manera automática, el calendario.

Las siguientes librerías no son tan importantes como las últimas mencionadas, pero la funcionalidad que nos proporciona es más que correcta. Las librerías nombradas son *[font\_awesome\_flutter](https://pub.dev/packages/font_awesome_flutter)* y [*scrollable\_positioned\_list*](https://pub.dev/packages/scrollable_positioned_list). Por un lado, *font\_awesome\_flutter* nos ofrece una grandiosa cantidad de nuevos iconos que no están de manera predeterminada en Flutter, por ejemplo, iconos de empresas muy conocidas, como la G de *Google* o la F de *Facebook*. Por otro lado, tenemos la librería de *scrollable\_positioned\_list*, la cuál nos facilita un widget muy parecido al *ListView* nativo de *Flutter*, pero qué añadí a este proyecto para facilitar la creación de ciertas partes del proyecto.

Los últimos paquetes, pero no por ello menos importante, son los utilizados en la aplicación de los camareros.

path\_provider: ^2.0.15

pdf: ^3.10.4

printing: ^5.11.0

1. (Apartado 5: Ejemplo bibliografía)

[Identificador] Inicial Nombre 1er Autor. Apellido 1º Autor, Inicial Nombre 2º Autor. Apellido 2º. Autor, , ... y Inicial Nombre Último Autor. Apellido Último Autor, *Nombre Completo del Libro en cursiva*. Ciudad de edición: Editorial, Año.

1. (Apartado 6)