
**Herramienta de apoyo a la gestión de inventario
para restaurantes**
**Inventory management support tool for
restaurants**



**Trabajo de Fin de Grado
Curso 2022–2023**

Autor
Jorge Arévalo Echevarria
Jesus Martín Moraleda

Director
Mercedes García Merayo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Herramienta de apoyo a la gestión de
inventario para restaurantes
Inventory management support tool for
restaurants

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Jorge Arévalo Echevarria
Jesus Martín Moraleda

Director
Mercedes García Merayo

Convocatoria: *Junio 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

27 de junio de 2023

Agradecimientos

Quiero empezar agradeciendo a mis amigos y compañeros de carrera, en especial a Edu, Gonzalo, Dámaso y José, los cuales han sido pilares fundamentales y me han ayudado a superar los peores momentos.

También agradecer a mis padres, mi hermana y toda mi familia, por estar siempre apoyándome y confiando en mí.

Por último agradecer a Mercedes por su implicación y por habernos ayudado y guiado en este proyecto, y a Jorge, por todo el esfuerzo realizado y su apoyo durante todos estos años.

Jesús Martín Moraleda

Agradecer a mi familia y amigos por el apoyo recibido durante los años de carrera, además de a los compañeros con los que he compartido asignaturas y proyectos.

También me gustaría agradecer en especial a mis compañeros Eduardo, Fernando, Gonzalo y Dámaso por su ayuda incondicional y los días interminables en la biblioteca.

Agradecer a todos aquellos profesores que han conseguido hacer mucho más ameno las asignaturas de la carrera, y en especial a Mercedes por su guía y consejo que han sido una parte fundamental para sacar este proyecto adelante.

Por ultimo agradecer a Jesús por su dedicación y esmero durante todos estos años.

Jorge Arévalo Echevarría

Resumen

Herramienta de apoyo a la gestión de inventario para restaurantes

Este trabajo consiste en el desarrollo de un entorno que facilite la gestión de inventario y la toma de pedidos en restaurantes. Por una parte tenemos una aplicación web, TakeOrderAdmin, en la que los propietarios de los restaurantes pueden gestionar el *stock* de ingredientes, los platos que ofrecen en la carta con los ingredientes necesarios para la elaboración de los mismos, así como el menú que ofrece cada día. La aplicación facilitará la gestión del *stock* mediante la actualización automática del mismo en base al consumo diario tanto por la elaboración del menú y por los platos servidos de la carta. Por otra parte tenemos la aplicación móvil, TakeOrder, diseñada para el uso de los empleados, con el objetivo de registrar las consumiciones de las mesas de una forma rápida, sencilla y eficaz. Esta aplicación realizará una actualización de la base de datos del inventario en base a los pedidos que se realicen.

Palabras clave

Aplicación Móvil, Aplicación Web, Restaurantes, Stock, Inventario, Javascript, Android

Abstract

Inventory management support tool for restaurants

This work consists of the development of two applications that make easier the inventory management and order taking in a restaurant. On the one hand, we have a web application, TakeOrderAdmin, in which restaurant owners can manage the stock of ingredients, the dishes offered on the menu with the ingredients needed to prepare them, as well as the menu offered each day. The application will facilitate stock management by automatically updating the stock based on daily consumption both for menu preparation and dishes served during services. On the other hand, we have the mobile application, TakeOrder, designed for the use of employees, with the objective of registering table consumption in a fast, simple and efficient way. This application will update the inventory database based on the orders placed.

Keywords

Mobile Application, Web Application, Restaurants, Orders, Stock, Javascript, Android.

Índice

1. Introducción	1
1.1. Objetivos	2
1.2. Plan de trabajo	3
2. Análisis de Requisitos	5
2.1. Ingredientes	5
2.2. Platos	5
2.3. Bebidas	6
2.4. Menús	6
2.5. Mesas	6
2.6. Comandas	7
3. Entornos de Desarrollo	9
3.1. Frameworks y Librerías	9
3.1.1. Visual Studio Code	9
3.1.2. Android Studio	9
3.1.3. Maven	10
3.1.4. Bootstrap	10
3.1.5. SweetAlert	10
3.1.6. Docker	10
3.1.7. Fly.io	10
3.2. Lenguajes de Programación	10

3.2.1. HTML	11
3.2.2. CSS	11
3.2.3. Javascript	11
3.2.4. Java	11
3.2.5. Kotlin	11
3.3. Bases de Datos	11
3.4. Control de Versiones	12
3.5. Maquetación y Prototipos	13
4. Diseño e Implementación	15
4.1. Fase de Análisis	15
4.2. Fase de Investigación	15
4.3. Fase de Instalación y Configuración	16
4.4. Despliegue de la página web	18
4.4.1. Docker	19
4.4.2. Despliegue en Fly.io	20
4.5. Fase de Desarrollo	21
4.5.1. Aplicación Móvil	21
4.5.2. Aplicación Web	25
5. Conclusiones y Trabajo Futuro	31
Introduction	33
5.1. Objectives	34
5.2. Work plan	35
Conclusions and Future Work	37
Contribuciones Personales	39
Bibliografía	43

Índice de figuras

1.1.	Plan de proyecto	3
1.2.	Leyenda del plan de proyecto	4
3.1.	Diagrama entidad relación	12
3.2.	Repositorio del código de la aplicación	13
3.3.	Repositorio del código web	13
3.4.	Mockup aplicación Móvil	14
3.5.	Mockup aplicación web	14
4.1.	Creación de un nuevo proyecto Firebase	16
4.2.	Menú inicio proyecto Firebase	17
4.3.	Analíticas de un proyecto Firestore	18
4.4.	Vincular el proyecto de Firestore a las aplicaciones	18
4.5.	Aplicación de escritorio Docker Desktop	19
4.6.	Archivo fly.toml	20
4.7.	Añadir mesa en aplicación móvil y Lista de mesas asignadas	22
4.8.	Generar comanda y editar mesa	23
4.9.	Añadir bebidas y ver carta	24
4.10.	Lista de bebidas pedidas	25
4.11.	Listado de ingredientes	26
4.12.	Listado de bebidas	26
4.13.	Listado de platos	27
4.14.	Menús	28

4.15. Alertas	29
5.1. Work plan	35
5.2. Work plan legend	36

Capítulo 1

Introducción

Este documento recoge todos los aspectos fundamentales para comprender la funcionalidad de la aplicación web y móvil diseñadas como Trabajo de Fin de Grado para facilitar la gestión de *stock* de ingredientes en restaurantes.

La idea del proyecto surge de la necesidad de una gestión rápida y eficiente a la hora de dirigir un restaurante, que son unas de las cualidades más importantes para llegar a conseguir un negocio de éxito.

Hasta hace poco tiempo en cualquier restaurante, la forma de llevar a cabo el trabajo se basaba en que el camarero tomase nota de lo que quería el cliente con una libreta y un bolígrafo, y se lo llevase a cocina para su preparación. Incluso podía el camarero decirte la típica expresión: "voy a consultar en cocina a ver si está disponible ese plato", aunque este tipo de gestión la podemos seguir viendo hoy en día en cualquier pueblo o chiringuito de playa.

Nos encontramos en una nueva era digital en la que se necesitan herramientas capaces de facilitar un trabajo eficiente, acotado en tiempo y con la calidad que esperan los clientes. La idea, por tanto, es acelerar al máximo los procesos de organización de un restaurante de manera que el gestor del restaurante pueda planificar y estructurar el servicio a los clientes, teniendo acceso a la configuración de diseño del menú del día, y los platos a elaborar.

En el mundo de la gastronomía el éxito se consigue a través de una buena materia prima, una buena puesta a punto en la elaboración de sus platos y una excelente atención al cliente con un servicio rápido y de calidad. En nuestro caso, nos hemos centrado en este último punto, reduciendo tareas manuales que consumen mucho tiempo haciendo más difícil concentrarse en la atención al cliente con un servicio de calidad.

Para conseguir este objetivo nuestro proyecto se centra, por una parte, en una comunicación dinámica entre cocina y empleados de sala, proporcionando una gestión rápida y eficaz a la hora de atender a los clientes y por otra, en facilitar una gestión sencilla de la administración del *stock* de los ingredientes y productos.

1.1. Objetivos

El software desarrollado se centra en la mejora y optimización de la gestión del *stock* y la comunicación entre camareros y cocina. Los objetivos serán, por tanto, la mejora de la eficiencia, la reducción del tiempo necesario y el aumento de la calidad del servicio a los clientes mediante un sistema automatizado de tareas combinado con una amplia base de datos donde almacenar toda la información y acceder a ella rápidamente.

La utilización de una aplicación web para la gestión y administración del restaurante asegura que los platos, bebidas o menús disponibles se ajusten exactamente a los ofertados al cliente y no haya errores entre los cocineros y los camareros. Además se permite acceso inmediato a la información de la base de datos con el objetivo de gestionar el *stock* del restaurante. La actualización del *stock* resultará, por tanto, extremadamente sencilla, ya que el gestor solo tendrá que reponer los ingredientes que estén por debajo de los niveles mínimos, para los cuales se genera una alerta para su reposición. Esto va a permitir dar una idea al gestor para la planificación de que platos elaborar en el menú de ese día.

Desde el punto de vista del trabajador, en este caso los camareros, la utilización de una aplicación móvil para la comunicación interna entre los trabajadores, asegura una comunicación directa con el cliente ofreciendo e informando los productos disponibles tanto en platos como en bebidas. De esta manera se genera un servicio rápido y de calidad a la hora de gestionar las comandas de cada mesa, sin generar esperas o preguntas en cocina que pueden disgustar al cliente.

Desde el punto de vista tecnológico, los objetivos que se plantean en este proyecto son los siguientes:

- Para la aplicación web, diseñar un software con una interfaz sencilla y que facilite el trabajo al gestor de la aplicación, como ver los ingredientes que han generado una alerta por estar por debajo de los niveles mínimos de *stock* para una reposición de dichos ingredientes. Además, podrá generar un menú del día pudiendo seleccionar la cantidad de platos a elaborar para ese día, a partir de los ingredientes disponibles gestionando que platos se ofertaran de primero, de segundo y de postre.
- Para la aplicación móvil, diseñar de la misma manera una interfaz sencilla y que facilite el trabajo a los trabajadores del restaurante, tanto a camareros como a cocineros. Podrán generar una comanda asignada a una mesa pudiendo tomar nota de las bebidas y los platos ofertados por la carta, además ofertar un menú del día tomando nota de los primeros y segundos platos, y del postre.
- Diseñar una base de datos que permita el almacenamiento de una gran información, en la que guardaremos todos los ingredientes dados de alta en el restaurante, los platos con sus ingredientes asignados, las bebidas almacenadas o las mesas, cada una asignada con una comanda, y en cada comanda, almacenados los platos y bebidas pedidas por esa mesa.

1.2. Plan de trabajo

Teniendo en cuenta los objetivos descritos anteriormente, para el plan de trabajo se realizó una estimación de tiempo con una duración de 7 meses. Para poder ver el plan de trabajo de una manera clara y sencilla decidimos hacer un diagrama de *Gantt*. El plan de trabajo se puede observar en la Figura 1.1 y la descripción de las actividades en la Figura 1.2

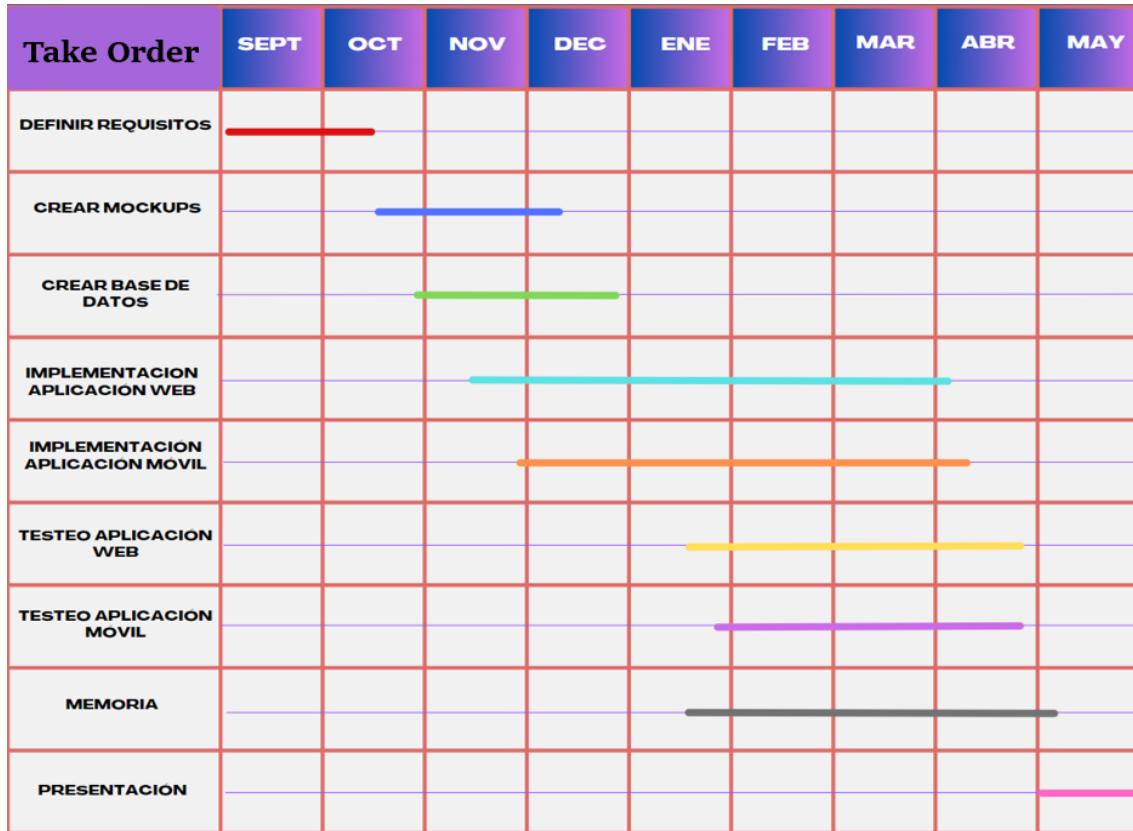


Figura 1.1: Plan de proyecto

	Definir Requisitos En esta fase diseñamos la base de nuestra aplicación, el objetivo y los requisitos técnicos.
	Crear mockups Diseñamos como queremos que sean la interacción del usuario con las interfaces, además de su apariencia.
	Crear base de datos Diseñamos una base de datos para gestionar toda la información respecto al restaurante. Los atributos que tienen que tener, relaciones y entidades.
	Implementación aplicación web Implementamos con código todos los requisitos especificados para la aplicación web en la primera fase del proyecto.
	Implementación aplicación móvil Implementamos con código todos los requisitos especificados para la aplicación móvil en la primera fase del proyecto.
	Testeo aplicación web Probamos todas las funciones que hemos implementado para garantizar el buen funcionamiento de la aplicación web.
	Testeo aplicación móvil Probamos todas las funciones que hemos implementado para garantizar el buen funcionamiento de la aplicación móvil.
	Memoria Probamos Describimos nuestro proyecto desde la idea, como se desarrolla, hasta su finalización, incluyendo un manual para el usuario.
	Presentación Fecha en la cual presentamos el proyecto ante nuestro tutor y el jurado

Figura 1.2: Leyenda del plan de proyecto

Capítulo 2

Análisis de Requisitos

El análisis de requisitos es una parte del proceso de desarrollo de software. En este capítulo, se detallarán los requisitos necesarios para el correcto funcionamiento del proyecto. El objetivo principal del análisis de requisitos es identificar las necesidades del usuario y traducirlas en especificaciones detalladas y claras para que el equipo de desarrollo pueda diseñar y construir la aplicación.

A continuación se presentan los requisitos para los ingredientes, platos, bebidas, menús, mesas y comandas, detallando los campos necesarios para cada uno de ellos y las restricciones de negocio y técnicas que deben ser consideradas en su implementación.

2.1. Ingredientes

- Al añadir un nuevo ingrediente no se puede dejar ningún campo vacío del formulario: nombre, categoría, número de unidades, tipo de medida del ingrediente y alerta mínima de existencias.
- Al editar un ingrediente se puede modificar cualquier campo.
- No puede haber dos ingredientes con el mismo nombre.
- Los ingredientes tendrán un valor mínimo de existencias, si el *stock* es inferior a ese valor, saltará una alarma al entrar a la aplicación.

2.2. Platos

- Al añadir un plato no se puede dejar ningún campo vacío del formulario: nombre, categoría, disponibilidad y lista de ingredientes.

- Al añadir un plato se mostrará una lista con todos los ingredientes disponibles, para elegir los que se requieran.
- Los platos deben tener al menos un ingrediente.
- Todos los ingredientes que se añadan deben tener una cantidad asociada, que se corresponde con la cantidad de una ración.
- Al editar un plato se puede modificar cualquier campo.
- No puede haber dos platos con el mismo nombre.

2.3. Bebidas

- Al añadir una bebida no se puede dejar ningún campo vacío del formulario: nombre, cantidad y alerta mínima de existencias.
- Al editar una bebida se puede modificar cualquier campo.
- No puede haber dos bebidas con el mismo nombre.
- Las bebidas tendrán un valor mínimo de existencias, si el *stock* es inferior a ese valor, saltará una alarma al entrar a la aplicación.

2.4. Menús

- El menú del día constará de una serie de primeros platos, segundos platos y postres.
- Por cada plato que se introduce en el menú se seleccionará un número fijo de raciones que se harán ese día, una vez se acaben no habrá más.
- Al seleccionar los platos para el menú del día se tendrá en cuenta el *stock* de los ingredientes que haya en ese momento. Cuando se guarde el menú se calculará si disponemos de los ingredientes suficientes para hacer todas las raciones que se hayan introducido. Si varios platos utilizan un mismo ingrediente, se tendrá en cuenta en el *stock* disponible.
- Solo habrá un menú disponible a la vez.

2.5. Mesas

- Al añadir una mesa no se puede dejar ningún campo vacío del formulario.
- Cada mesa tendrá un número único que la identifica.

- Al editar una mesa se puede modificar cualquier campo.
- No puede haber dos mesas con el mismo identificador.

2.6. Comandas

- Cada comanda irá asociada única y exclusivamente a una mesa.
- Una comanda podrá ser de dos tipos: un menú del día o selección de platos de la carta. Los platos de la carta estarán sujetos a la disponibilidad de ingredientes que haya en ese momento.
- Al añadir un plato del menú del día no se puede superar la cantidad que esté disponible en ese momento.
- Al servir los platos pedidos a la mesa se podrán marcar como entregado.
- Cada comanda tendrá una ventana donde se muestran todos los platos y bebidas pedidas hasta el momento, así como un botón que dirá si ya han sido entregados o no.

Capítulo 3

Entornos de Desarrollo

En este capítulo se van a enumerar una serie de librerías, herramientas, entornos y lenguajes de programación que hemos utilizado y han facilitado la realización del proyecto. También se procederá a explicar nuestro sistema de control de versiones y la base de datos empleada, además de las utilidades para prototipo de las interfaces.

3.1. Frameworks y Librerías

En esta sección se enumera la lista de entornos usados para el desarrollo de las dos aplicaciones, así como los *frameworks* que nos han ayudado en la implementación del código.

3.1.1. Visual Studio Code

Visual Studio Code (Microsoft, 2015) es un editor de código fuente desarrollado por Microsoft. Es un software libre y multiplataforma, que está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sin número de extensiones, que básicamente da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

3.1.2. Android Studio

Android Studio (Google, 2013), es el entorno de desarrollo integrado oficial del sistema operativo Android de Google, basado en el software IntelliJ IDEA de JetBrains y diseñado específicamente para el desarrollo de Android. Está disponible para su descarga en sistemas operativos basados en Windows, macOS y Linux. Además es el principal entorno de desarrollo para aplicaciones móviles, con soporte para

diversos lenguajes de programación como Java, C++ y Kotlin.

3.1.3. Maven

Maven (Apache Software Foundation, 2002) es un *Project Management Framework*, esto es, un *framework* de gestión de proyectos de *software*, que proporciona un modelo estándar de gestión y descripción de proyectos. Maven da soluciones a tareas que abarcan desde la compilación hasta la distribución, despliegue y documentación de los proyectos.

3.1.4. Bootstrap

Bootstrap (Bootstrap Team, 2011) es un *framework* CSS gratuito y de código abierto orientado al desarrollo web *front-end* responsive y *mobile-first*. Contiene plantillas de diseño basadas en HTML, CSS y JavaScript para tipografía, formularios, botones, navegación y otros componentes de la interfaz.

3.1.5. SweetAlert

SweetAlert (Tristan, 2022) es una variante atractiva y muy personalizable para gestionar los *pop-ups* emergentes en javascript.

3.1.6. Docker

Docker (Kamel Founadi, Solomon Hykes, and Sebastien Pahl, 2013) es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

3.1.7. Fly.io

Fly.io (Open Source community, 2017) es una nueva nube pública, construida sobre servidores *Bare-Metal* que se gestiona en centros de datos de todo el mundo, diseñada para facilitar el despliegue de aplicaciones distribuidas y en tiempo real cerca de sus usuarios, estén donde estén.

3.2. Lenguajes de Programación

Durante la realización del proyecto se han utilizado los lenguajes de programación presentados a continuación.

3.2.1. HTML

HTML (Tim Berners-Lee, 1993) es el lenguaje de marcado estándar para documentos diseñados para visualizarse en un navegador web. Suele estar asistido por tecnologías como las hojas de estilo en cascada (*Cascading Style Sheets*) y lenguajes de script como JavaScript.

3.2.2. CSS

CSS (Håkon Wium Lie, 1996) es un lenguaje que permite el desarrollo del estilo de una página web cuando es utilizado sobre otro lenguaje de marcado, como en este caso HTML. Aunque la mayor parte del aspecto visual de la aplicación está realizado a través de Bootstrap se han incluido algunas hojas de estilo para añadir detalles que este *framework* no contempla.

3.2.3. Javascript

JavaScript (Brendan Eich, 1995) es un lenguaje de programación que se utiliza para hacer páginas web interactivas. Desde actualizar fuentes de redes sociales a mostrar animaciones y mapas interactivos, las funciones de JavaScript pueden mejorar la experiencia del usuario de un sitio web.

3.2.4. Java

Java (Sun Microsystems, 1995) es un lenguaje de programación ampliamente utilizado para codificar aplicaciones web. Ha sido una opción popular entre los desarrolladores durante más de dos décadas, con millones de aplicaciones Java en uso en la actualidad.

3.2.5. Kotlin

Kotlin (JetBrains, 2010) es un lenguaje de programación de código abierto creado por JetBrains que se ha popularizado gracias a que se puede utilizar para programar aplicaciones Android.

3.3. Bases de Datos

Como gestor de base de datos decidimos usar *Firebase* debido a su comodidad para integrarse en aplicaciones móviles realizadas con Android Studio.

Cloud Firestore (Google, 2010) es una base de datos NoSQL orientada a documentos. A diferencia de una base de datos SQL, no hay tablas. En su lugar, se almacenan los datos en documentos, que se organizan en colecciones. Cada documento contiene un conjunto de pares clave-valor. En la Figura 3.1 podemos observar el diagrama entidad-relación

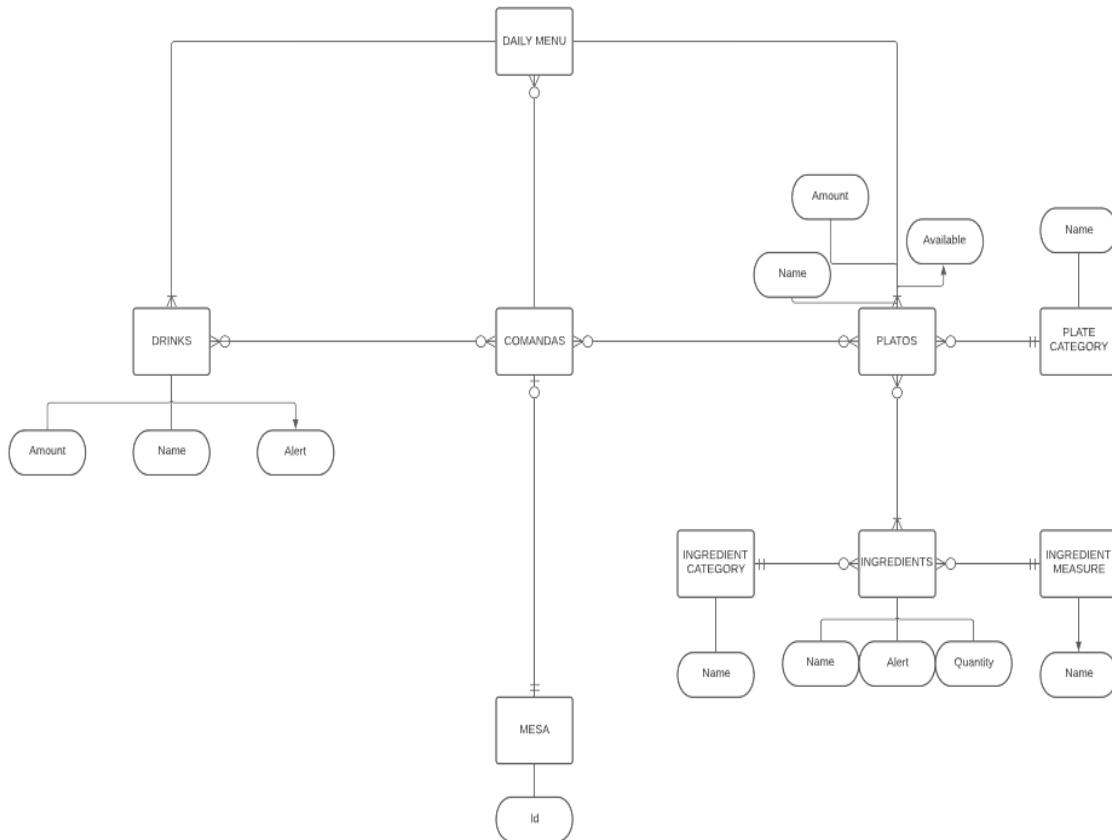


Figura 3.1: Diagrama entidad relación

3.4. Control de Versiones

GitHub (Microsoft, 2008) ofrece un servicio, tanto gratuito como de pago, para el control de versiones de proyectos usando la tecnología Git, registrando cada cambio subido al servidor gracias a su almacenamiento en la nube, y permitiendo volver a versiones anteriores o comparar los cambios realizados entre las mismas. Decidimos usar esta plataforma dada nuestra experiencia con ella y que nos viene muy bien para trabajar paralelamente los proyectos. Para ello creamos dos repositorios, uno para el desarrollo de la aplicación móvil, Figura 3.2, y otro para el de la web, Figura 3.3.

- Aplicación móvil: <https://github.com/jesusmoraleda/TakeOrder.git>.

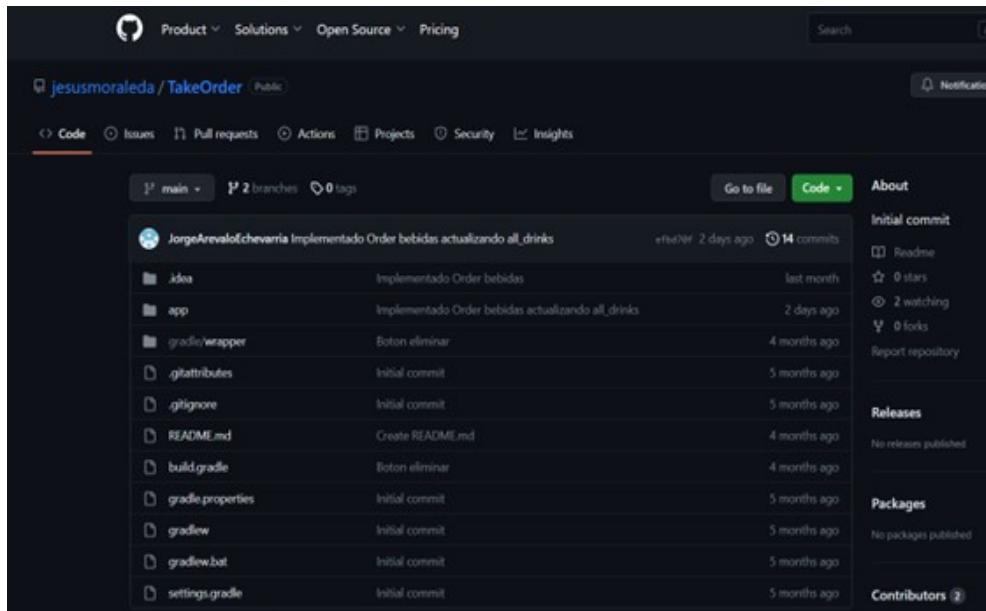


Figura 3.2: Repositorio del código de la aplicación

- Aplicación web: <https://github.com/jesusmoraleda/TakeOrderAdmin.git>.

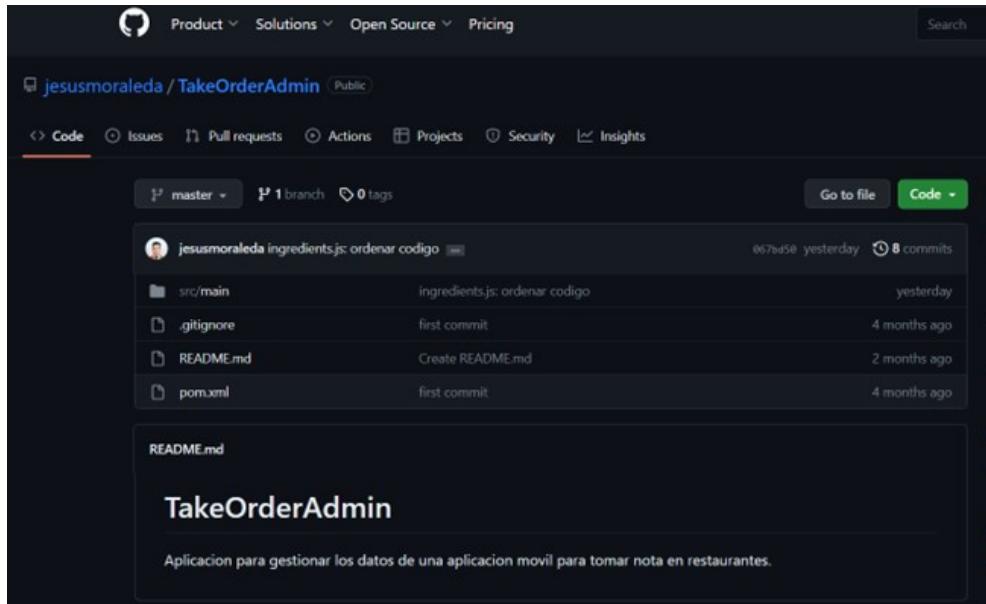


Figura 3.3: Repositorio del código web

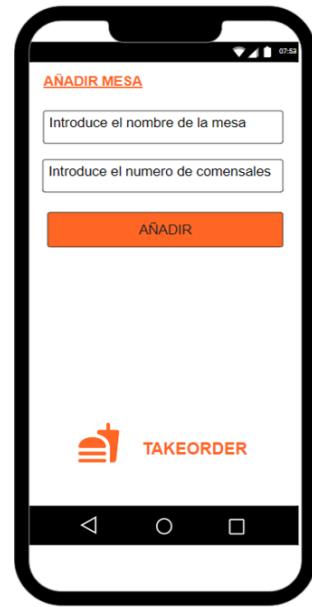
3.5. Maquetación y Prototipos

Comenzamos desarrollando unos *mockups* para que nos ayudaran a tener una idea general del aspecto que queríamos para la app móvil, como las que aparecen

en las Figuras 3.4a y 3.4b, y para aplicación web como se puede visualizar en la Figura 3.5, con el objetivo de resultar amistosa para el usuario.



(a) Mockup 1



(b) Mockup 2

Figura 3.4: Mockup aplicación Móvil

A tablet mockup showing a table of ingredients. The top bar includes the 'TAKEORDER' logo, a search bar, and a dropdown menu. The table has columns: NOMBRE, CATEGORIA, CANTIDAD, and ALERTA. Data rows include: CEBOLLA (VERDURAS, 8, 5), AJOS (VERDURAS, 10, 7), PIÑAS (FRUTAS, 8, 4), and LUBINAS (PESCADO, 6, 3).

NOMBRE	CATEGORIA	CANTIDAD	ALERTA
CEBOLLA	VERDURAS	8	5
AJOS	VERDURAS	10	7
PIÑAS	FRUTAS	8	4
LUBINAS	PESCADO	6	3

Figura 3.5: Mockup aplicación web

Capítulo 4

Diseño e Implementación

En este capítulo se presentan diferentes fases del proyecto que permiten explicar por qué utilizamos las tecnologías mencionadas y reproducir la ejecución del mismo a través de un manual de instalación y configuración.

4.1. Fase de Análisis

Una vez planteado el proyecto se analizaron diferentes tecnologías y lenguajes de programación para afrontar su implementación. La aplicación web teníamos claro que la íbamos a desarrollar con JavaScript y HTML, ya que no teníamos experiencia previa creando páginas web y estos lenguajes son los más usados y podemos encontrar mucha información sobre ellos. Para la creación de la aplicación móvil tuvimos más dudas: empezamos intentando crear un proyecto usando Xamarin (John Freddy Vega, 2014), puesto que Jesús tenía algo de experiencia previa con ese lenguaje, pero al poco tiempo vimos que nos daba algunos problemas, por lo que decidimos cambiar el proyecto a Android Studio. Jorge ya había usado este programa en otra asignatura, así que teníamos esa ventaja. Además, es un entorno más sencillo e intuitivo para crear aplicaciones móviles.

En cuanto a la base de datos, la primera idea era generar una base de datos relacional con MySql o con Oracle, pero debatiendo nos dimos cuenta de que para un proyecto de este tipo, en el que no se necesitaba una gran base de datos, el uso de una base de datos no relacional, como la que ofrece *Firebase* nos proporcionaba más seguridad y una mejor y más sencilla implementación.

4.2. Fase de Investigación

Una vez elegidas las tecnologías a usar en la fase de análisis, nos encontramos ante algunas dificultades, como los lenguajes web y el servicio de *Firebase*. Los lenguajes

web no los habíamos visto prácticamente durante la carrera, al igual que las bases de datos no relacionales, así que nos pusimos a hacer algún curso online (un curso JavaScript en Platzi (John Freddy Vega, 2014) y algunos tutoriales de *Firebase*) con los que poder empezar a trabajar en estos proyectos.

4.3. Fase de Instalación y Configuración

En esta sección se describen los pasos a seguir para recrear y ejecutar el proyecto.

Paso 1. Creación de un proyecto en GitHub

Para mantener el control de versiones y copias de seguridad constantes del proyecto, se utiliza un proyecto en GitHub, y su aplicación de escritorio GitHub Desktop para manejarlo más fácilmente.

Paso 2. Instalación de Android Studio

Se descargó la aplicación de código abierto oficial de Android para el desarrollo de la aplicación móvil.

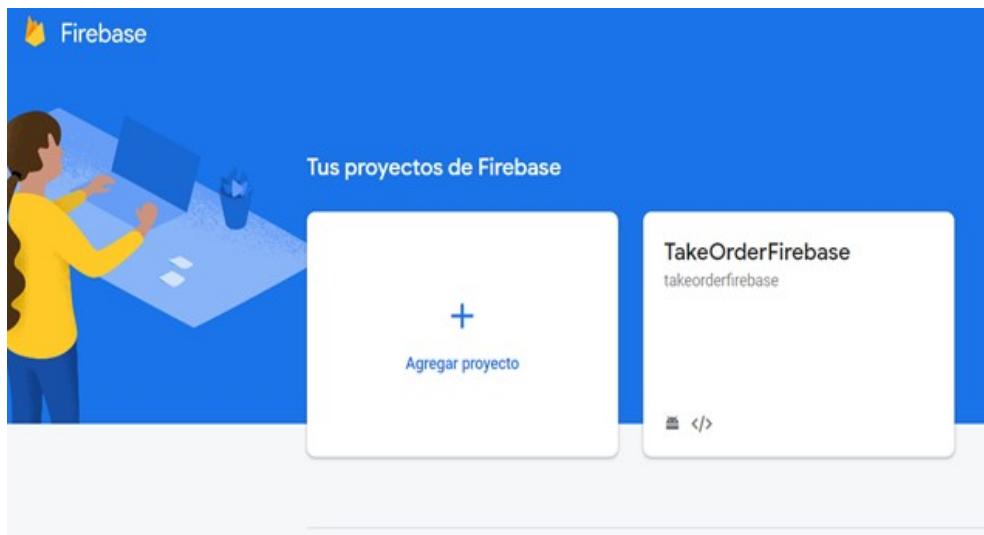


Figura 4.1: Creación de un nuevo proyecto Firebase

Paso 3. Instalación de Visual Studio Code

Se descargó la aplicación de Visual Studio Code (Microsoft, 2015) desde su página oficial.

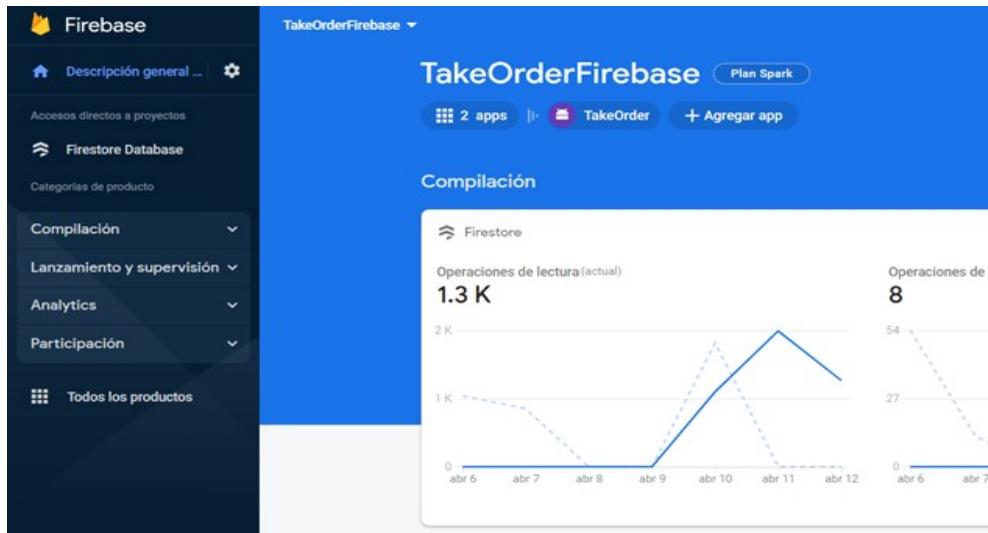


Figura 4.2: Menú inicio proyecto Firebase

Paso 4. Creación de un proyecto en Firebase

Se ha desarrollado un proyecto en *Firebase* para registrar y administrar el restaurante, el cual se integrará con las aplicaciones web y móvil.

Para llevar a cabo esto, hay que acceder a la plataforma de *Firebase* y autenticarse con una cuenta de Google. Una vez iniciada la sesión, creamos un proyecto nuevo (como se muestra en la Figura 4.1) y rellenamos el formulario con información como el nombre del proyecto y su ubicación.

Firebase nos ofrece dos tipos de base de datos: *Cloud Firestore* o *Realtime Database*. *Cloud Firestore* es la base de datos más reciente de *Firebase* y mejora en muchos aspectos (consultas más ricas y rápidas, nuevo modelo de datos más intuitivo...) a la original, *Realtime Database*, por lo que nos decidimos a usar esta nueva solución.

Cuando hayamos creado el proyecto llegamos a su página principal(Figura 4.2), desde la que podremos acceder a los diferentes menús donde nos ofrecen un montón de métricas de nuestras aplicaciones que podemos consultar(Figura 4.3).

Ya solo nos queda vincular nuestro proyecto de *Firebase* con nuestras aplicaciones. Para ello tenemos que pulsar sobre el ícono *settings* a la derecha del nombre del proyecto, y en la pestaña general nos dará la opción de vincular nuestras aplicaciones (Figura 4.4).

Seguiremos todos los pasos que nos va indicando *Firebase*, entre los cuales está añadir un trozo de código con todos los parámetros necesarios para realizar la conexión.

Una vez finalizada la conexión de la aplicación con *Firebase* podremos hacer uso de las utilidades de las que dispone, como el acceso a la base de datos de usuarios, llamada *Authentication*, y la base de datos no relacional que se encuentra en la

pestaña *Firebase Database*.

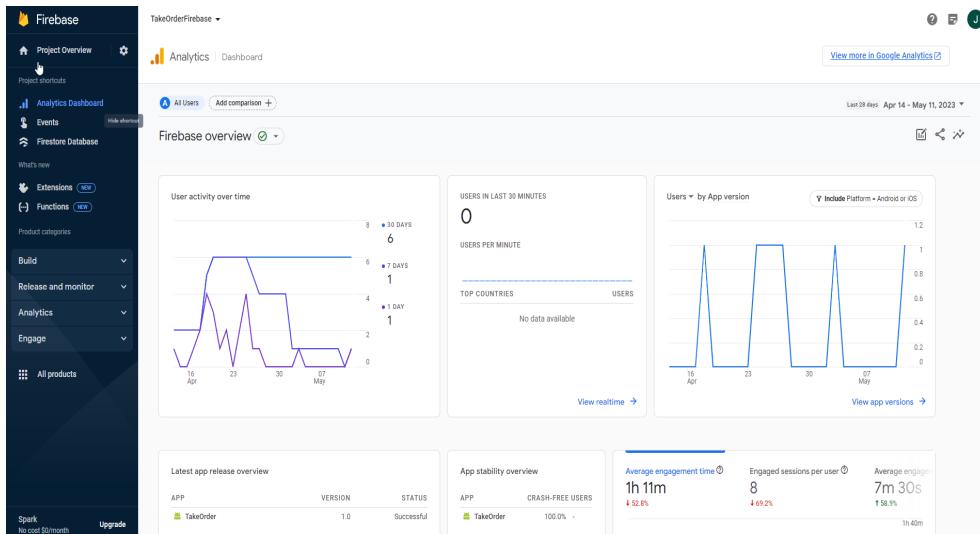


Figura 4.3: Analíticas de un proyecto Firestore

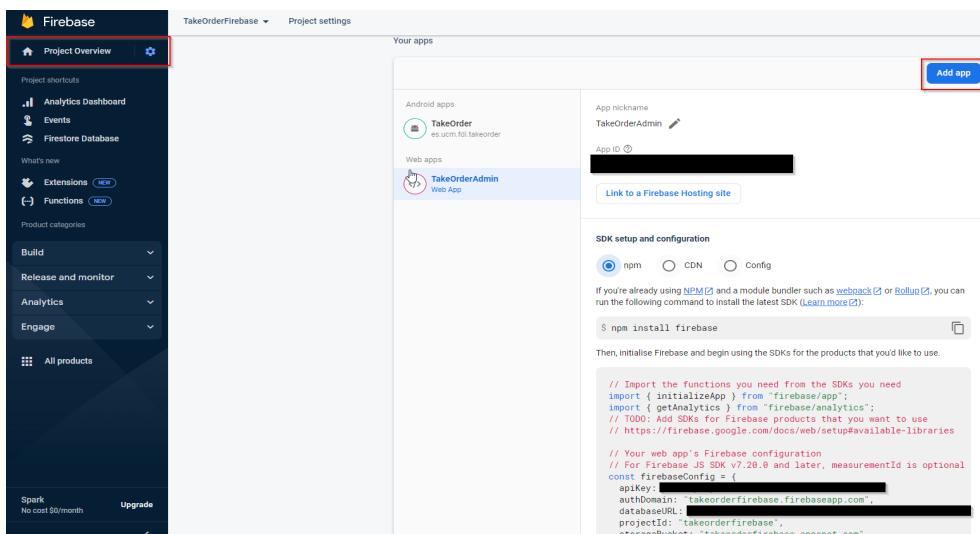


Figura 4.4: Vincular el proyecto de Firestore a las aplicaciones

4.4. Despliegue de la página web

Las aplicaciones web se ejecutan en un servidor. En nuestro proyecto se utiliza para el *back-end* la plataforma que nos ofrece Fly.io, y para el *front-end*, *Firebase*.

Fly.io es un *PaaS* (*Platform as a service*) que nos permite manejar el servidor y sus configuraciones. Su arquitectura es mucho más robusta y segura que un *hosting*.

El proceso para el despliegue ha consistido en registrarnos en Fly.io, y desplegar nuestra aplicación. Fly.io te ofrece distintas maneras para desplegar tu aplicación:

desde diferentes *frameworks* web (Django, Phoenix, Rails...) seleccionando un lenguaje de programación (Ruby, Python....) hasta despliegue con un *Dockerfile*. Como nuestra aplicación no usaba ningún *framework* de las opciones que nos daba Fly.io, decidimos crear un *dockerfile* para nuestra aplicación.

4.4.1. Docker

Para poder desplegar nuestra aplicación en Fly.io necesitamos tener un *Dockerfile* o una imagen de nuestro proyecto. Un *Dockerfile* es un archivo de texto que contiene las instrucciones necesarias para crear una nueva imagen. Una imagen de *Docker* es una plantilla de solo lectura que define su contenedor. La imagen contiene el código que se ejecutará, incluida cualquier definición para cualquier biblioteca o dependencia que el código necesite.

Para conseguir esa imagen utilizamos un *plugin* de Google: *jib-maven-plugin*. Para construir la imagen usamos el comando `mvn compile jib:dockerBuild` desde la terminal de VSCode. Ahora tenemos la imagen *docker* creada, y podemos visualizarla desde la aplicación de escritorio de *Docker* (Figura 4.5).

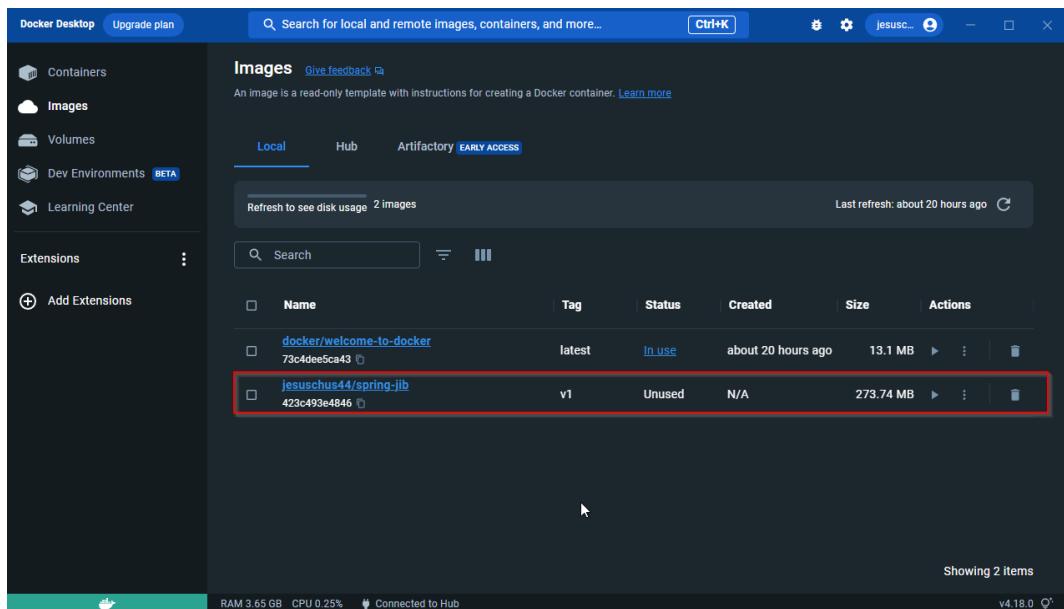


Figura 4.5: Aplicación de escritorio Docker Desktop

Para poder usar esa imagen en Fly.io tenemos que publicarla en *Docker hub*, que es un repositorio público de imágenes. Para hacer esto tenemos que ejecutar una serie de comandos desde el *Powershell* de Windows:

- Hacemos login en nuestra cuenta de *Docker*.
- Se prepara la imagen para que sea aceptada en este registro público. Todos los registros siguen una nomenclatura a la hora de almacenar los repositorios. En el caso de *Docker hub* necesitamos que nuestra imagen se llame

me nombredeusuario/nombredelrepositorio:etiqueta. En nuestro caso, jesuschus44/spring-jib:v1.

- Por último, usaremos el comando `docker push jesuschus44/spring-jib:v1` para publicar nuestra imagen en el *Docker hub*.

4.4.2. Despliegue en Fly.io

Ahora sí, con nuestra imagen publicada en nuestro repositorio de *Docker*, podemos desplegar nuestra aplicación en Fly.io. mediante los siguientes comandos:

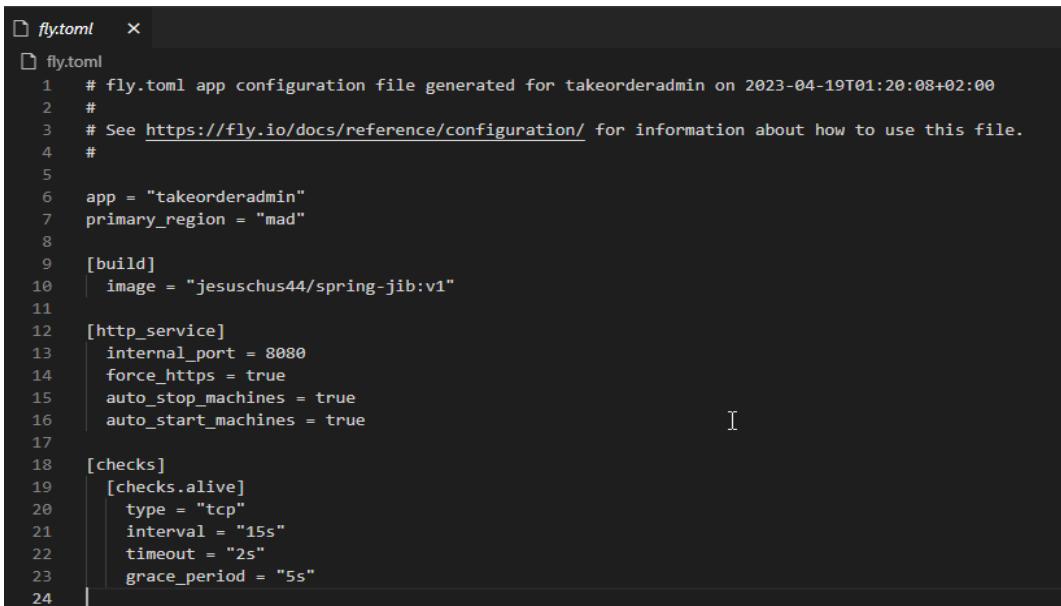
- Primero instalaremos *flyctl*:

```
powershell -Command iwr https://fly.io/install.ps1 -useb | iex
```

- Después nos *logueamos* en fly.io: `fly auth login`

- Por último, lanzamos nuestra aplicación:

```
fly launch - --image jesuschus44/spring-jib:v1
```



```

fly.toml  ×
fly.toml
1 # fly.toml app configuration file generated for takeorderadmin on 2023-04-19T01:20:08+02:00
2 #
3 # See https://fly.io/docs/reference/configuration/ for information about how to use this file.
4 #
5
6 app = "takeorderadmin"
7 primary_region = "mad"
8
9 [build]
10   image = "jesuschus44/spring-jib:v1"
11
12 [http_service]
13   internal_port = 8080
14   force_https = true
15   auto_stop_machines = true
16   auto_start_machines = true
17
18 [checks]
19   [checks.alive]
20     type = "tcp"
21     interval = "15s"
22     timeout = "2s"
23     grace_period = "5s"
24

```

Figura 4.6: Archivo fly.toml

Cada aplicación Fly.io necesita un archivo `fly.toml` (Figura 4.6) para indicar al sistema como desplegarla. El archivo `fly.toml` se puede generar automáticamente con el comando `fly launch`, que nos hará algunas preguntas para configurar todo. Nos preguntará por un nombre para nuestra aplicación, una región y si queremos una base de datos *Postgres* o *Redis*. Finalmente, nos preguntará si queremos desplegar para lo que usará la configuración que ha creado en el archivo `fly.toml`. Ahora ya podemos visitar nuestra

web. La dirección será el nombre de nuestra aplicación.fly.dev, en nuestro caso <https://takeorderadmin.fly.dev/>

4.5. Fase de Desarrollo

En esta sección se muestra como hemos generado las distintas vistas para la aplicación móvil y para la aplicación web. En cada uno de los apartados describiremos cada una de las funcionalidades de cada vista.

4.5.1. Aplicación Móvil

Lista de mesas asignadas

En esta vista el usuario, como se aprecia en la Figura 4.7a, podrá observar el listado de mesas asignadas en el restaurante.

- El botón flotante, de color verde azulado, nos permite generar una nueva ventana en la que podremos añadir una nueva mesa.
- El botón *editar*, asociado a cada mesa, nos generará una nueva ventana en la que se permitirá modificar los datos de la mesa seleccionada.
- El botón *comanda*, asociado a cada reserva, nos genera una nueva ventana, en la que se permitirá crear una comanda para la mesa seleccionada.
- El botón *eliminar*, asociado a cada mesa, borrara la misma, y la eliminará del listado de mesas asignadas.

Añadir una nueva mesa

En esta vista el usuario, como se aprecia en la Figura 4.7b, accederá a la ventana para añadir una nueva mesa.

- El primer campo nos permitirá introducir el identificador de la mesa que asignaremos.
- El segundo campo nos permitirá introducir el número de comensales.
- El botón *añadir* validará los datos y añadirá la nueva mesa asignada.

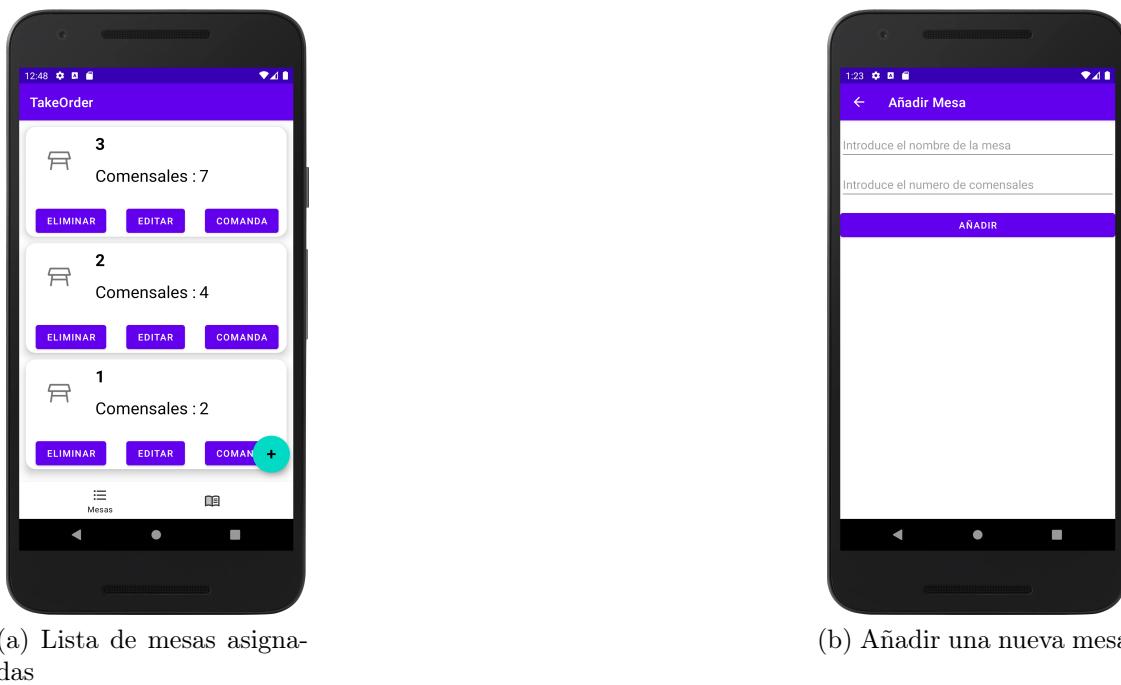


Figura 4.7: Añadir mesa en aplicación móvil y Lista de mesas asignadas

Editar una mesa asignada

En esta vista el usuario, como se aprecia en la Figura 4.8a, podrá editar una mesa ya asignada.

- El primer campo permitirá modificar el nombre de la mesa asignada.
- El segundo campo permitirá modificar el número de comensales.
- El botón *actualizar* validará los datos y modificará la mesa asignada.

Generar nueva comanda a una mesa asignada

En esta vista, el usuario, como se aprecia en la Figura 4.8b, podrá seleccionar que tipo de comanda desea el cliente.

- El botón *carta*, nos generará una nueva ventana en la que se permitirá generar una comanda pudiendo seleccionar productos de la carta.
- El botón *menú*, nos generará una nueva ventana en la que se permitirá generar una comanda pudiendo seleccionar los platos del menú diario.

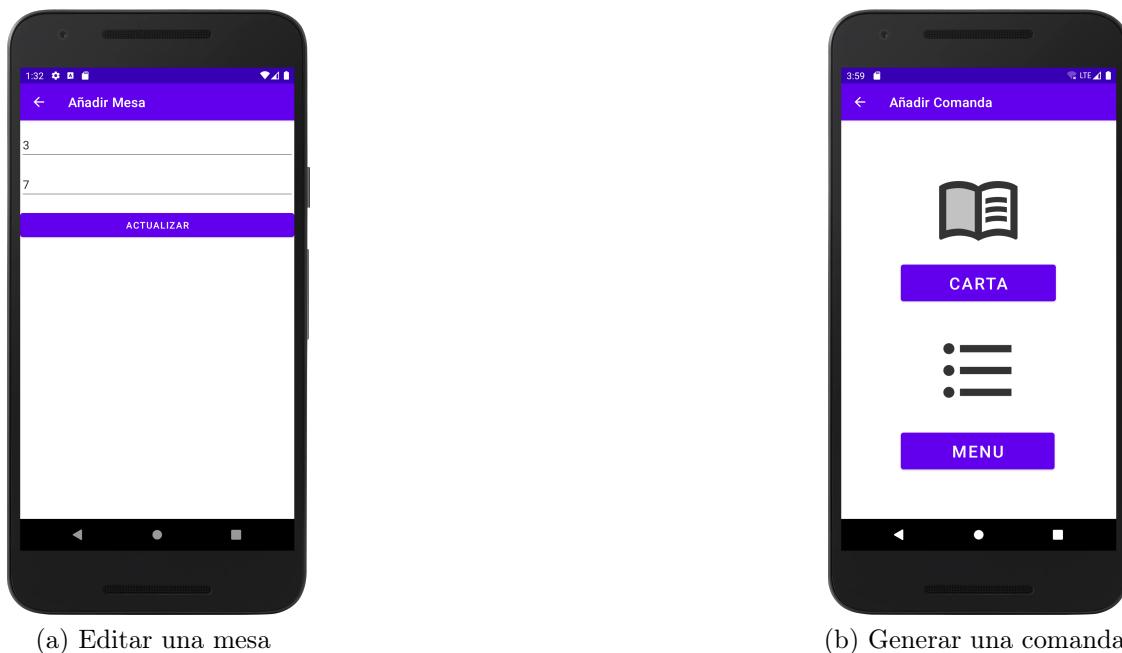


Figura 4.8: Generar comanda y editar mesa

Carta

En esta vista, el usuario, como se aprecia en la Figura 4.9a, podrá añadir los elementos de la carta deseados, además de poder ver el listado de los elementos solicitados.

- El botón *añadir bebidas*, nos generará una nueva ventana en la que se listara todas las bebidas disponibles.
- El botón *bebidas pedidas*, nos generará una nueva ventana en la que listara las bebidas pedidas hasta el momento.
- El botón *añadir platos*, nos mostrará una nueva ventana en la que se listara todos los platos disponibles.
- Por último, el botón *platos pedidos*, nos llevará a una nueva ventana en la que se listarán los platos pedidos hasta el momento.

Añadir bebidas

En esta vista el usuario, como se aprecia en la Figura 4.9b, podrá ver el listado de las bebidas disponibles, pudiendo añadir la bebida y la cantidad deseadas.

- El primer campo nos permitirá introducir el número de bebidas que deseamos añadir, debiendo ser menor que la cantidad total disponible.

- El botón *añadir*, nos generará una nueva ventana de confirmación en la que nos mostrará la cantidad introducida, y el nombre de la bebida que deseamos añadir, pudiendo seleccionar el botón *cancelar* o *aceptar* para confirmar el pedido.

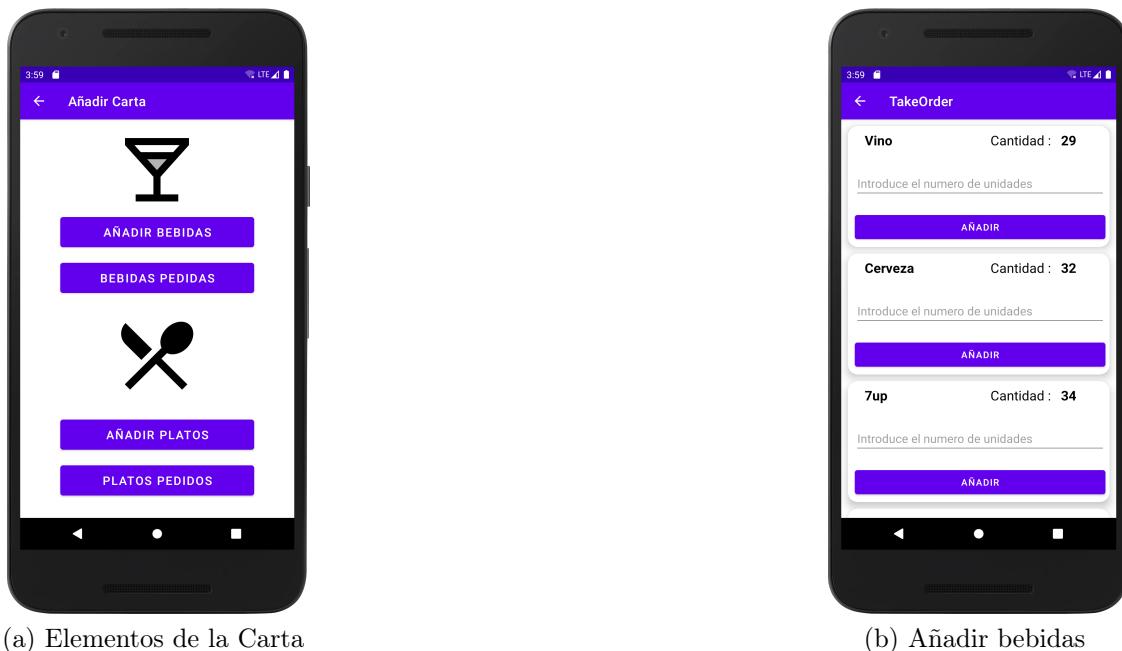


Figura 4.9: Añadir bebidas y ver carta

Lista de bebidas pedidas

En esta vista, el usuario podrá ver el listado de las bebidas pedidas Figura 4.10 con la cantidad pedida de cada bebida .

- El botón *pendiente* estará seleccionado por defecto cuando se añada una nueva bebida y permanecerá en ese estado hasta que se haga entrega de las bebidas a la mesa. El botón *pendiente* se deshabilitará cuando se seleccione el botón *entregado*.
- El botón *entregado* estará sin seleccionar por defecto cuando se añada una nueva bebida, cuando se haga entrega de las bebidas en la mesa el usuario podrá seleccionar el botón *entregado*.



Figura 4.10: Lista de bebidas pedidas

4.5.2. Aplicación Web

Listado de ingredientes

Esta vista contiene el listado de todos los ingredientes del restaurante. Por defecto, la tabla está ordenada por el nombre de los ingredientes como se aprecia en la Figura 4.11

- El botón *Añadir nuevo ingrediente* nos abrirá un *pop up* donde podemos seleccionar todos los valores y datos del ingrediente: nombre, categoría, cantidad, alerta y medida.
- También dispone de un filtro y un buscador. El filtro te selecciona los ingredientes de la categoría que selecciones. El buscador, en cambio, busca en la tabla por el nombre del ingrediente.
- Cada ingrediente dispone de dos botones: editar y eliminar. El botón editar nos muestra un *pop up*, igual que el de añadir ingrediente, pero con los datos rellenos del ingrediente seleccionado, para poder cambiar los campos que nosotros queramos. El botón eliminar abre un *pop up* para eliminar el ingrediente.

El objetivo era diseñar una interfaz intuitiva, en la que sea fácil y rápido consultar y gestionar todo el *stock* de ingredientes del restaurante.

Listado de bebidas

Esta vista nos muestra todas las bebidas disponibles en el restaurante, Figura 4.12.

NOMBRE	CATEGORIA	CANTIDAD	MEDIDA	ALERTA	MODIFICAR	ELIMINAR
Tomate natural	Vegetales	22	Kilos	25		
Tomate frito	Salsas	12	Kilos	4		
Spaghettti	Pasta	78	Kilos	4		
Sal	Especias	20	Kilos	5		
Queso para gratinar	Postres/lacteos	6	Kilos	1		
Queso en lonchas	Postres/lacteos	32	Unidades	4		
Queso de cabra	Postres/lacteos	12	Kilos	2		
Platanos	Frutas	40	Unidades	12		
Pimientos	Vegetales	56	Kilos	12		
Patatas	Vegetales	80	Kilos	10		
Pasta sin gluten	Pasta	20	Kilos	4		
Pasta integral	Pasta	12	Kilos	3		
Pan	Productos secos	123	Kilos	12		

Figura 4.11: Listado de ingredientes

- El botón *Añadir nueva bebida* nos abrirá un *pop up* donde podemos añadir una nueva bebida con sus campos correspondientes: nombre, cantidad y alerta.
- En esta ventana solo hay disponible un buscador por nombre de bebida. No hay filtro porque las bebidas no tienen una categoría asociada.
- Al igual que en la ventana de ingredientes, cada bebida dispone de dos botones: editar y eliminar. El de editar abre un *pop up* con los datos de la bebida donde podemos cambiar el que queramos. El de eliminar, elimina una bebida.

NOMBRE	CANTIDAD	ALERTA	MODIFICAR	ELIMINAR
7up	34	5		
Agua	61	100		
Cerveza	32	5		
Cocacola	44	5		
Fanta	48	5		
Nestea	5	5		
Vino	27	5		

Figura 4.12: Listado de bebidas

Listado de platos

Aquí mostramos el listado de todos los platos del restaurante, ver Figura 4.13.

- El botón *Añadir nuevo plato* nos abrirá un *pop up* para añadir un nuevo plato. En el *pop up* deberemos seleccionar un nombre, una categoría y una lista de ingredientes con su cantidad correspondiente. La cantidad es la establecida para una ración de ese plato y cada plato deberá tener al menos un ingrediente.
- Dispone de un filtro por categorías (primer plato, bocadillos, postre...) y un buscador global que busca por nombre del plato.
- Cada plato tiene un botón *Ingredientes* que al pulsar nos muestra un desplegable con los ingredientes y cantidades necesarias para elaborar ese plato.
- Al igual que los ingredientes y las bebidas, cada plato dispone de dos botones, editar y eliminar, que funcionan igual que los anteriores.

Nombre del plato	Categoría	Ingredientes	Editar	Eliminar
Sopa de fideos	Primer plato	Ingredientes	/	/
Macarrones bolognesa	Primer plato	Ingredientes	/	/
Lubina al horno	Segundo plato	Ingredientes	/	/
Hamburguesa de pollo	Hamburguesas	Ingredientes	/	/
Hamburguesa de carne	Hamburguesas	Ingredientes	/	/
Fruta	Postre	Ingredientes	/	/
Ensalada mixta	Primer plato	Ingredientes	/	/
Bocadillo de jamón y queso	Bocadillos	Ingredientes	/	/
Arroz con pollo	Segundo plato	Ingredientes	/	/

Figura 4.13: Listado de platos

Menú del día

En la Figura 4.14 podemos ver la vista donde se confecciona el menú diario que servirá el restaurante.

- Tiene una tabla donde muestra todos los primeros platos, segundos y postres disponibles, que corresponden a las categorías que compondrán el menú. Cada plato tiene un *checkbox* y una cantidad asociada. Para añadir un plato al menú debemos introducir las raciones que se van a elaborar. Este procedimiento se repetirá con todos los platos que queramos añadir.

- La tabla de la derecha nos muestra los platos añadidos al menú. Este menú se guarda pulsando el botón *guardar menú*.
- El botón *guardar menú* se encarga de comprobar que hay disponibilidad de ingredientes para realizar todas las cantidades de los platos que se han añadido al menú. Si no hay suficiente cantidad de algún ingrediente, te lo muestra en un *pop up* para que cambies las cantidades de algún plato o poner otros distintos.
- En esta vista no se pueden editar los platos ni sus ingredientes, solo se pueden añadir o quitar platos del menú del día.

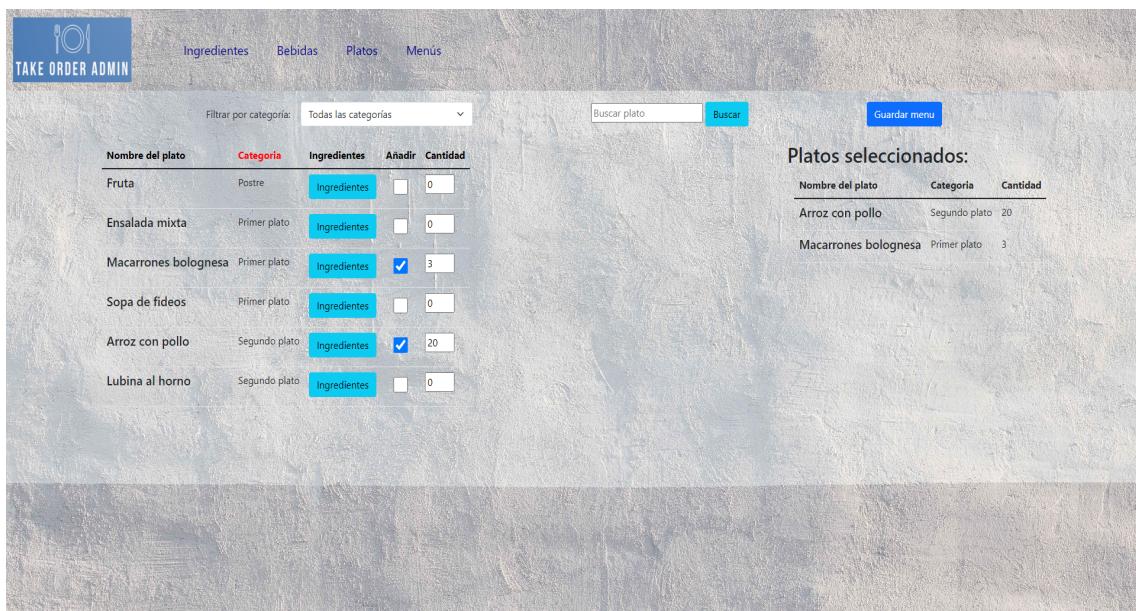


Figura 4.14: Menús

Alertas

Esta es la página inicial, Figura 4.15, donde se muestran los ingredientes y bebidas que tienen un *stock* inferior al nivel de la alerta que se estableció para cada uno de ellos.

- Al iniciar la web se muestra un *pop-up* si hay algún ingrediente o bebida bajo de *stock*.
- Hay dos tablas, una que muestra las alertas de los ingredientes y otra la de las bebidas. Cada tabla tiene un botón para reponer su stock. Al pulsar en cada botón nos mostrará un *pop-up* para poder reponer el stock de las bebidas o los ingredientes, tanto de los que están por debajo de su alerta y los que no.

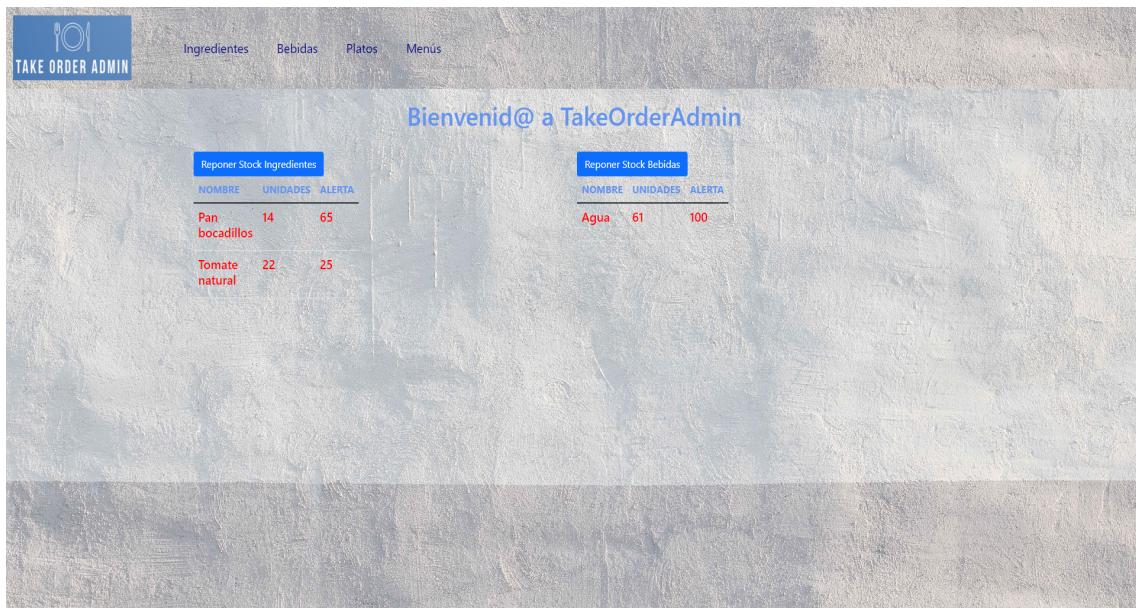


Figura 4.15: Alertas

Capítulo 5

Conclusiones y Trabajo Futuro

Una vez finalizado el proyecto, podemos decir que el objetivo de crear una aplicación para el mantenimiento del *stock* de restaurantes se ha cumplido. Se ha quedado una interfaz muy intuitiva y sencilla de utilizar, que era el objetivo principal. Los objetivos alcanzados han sido:

- Creación de distintas vistas para gestionar el *stock* de un restaurante desde la visión del propietario del local.
- Diseño de una base de datos para almacenar los ingredientes, bebidas y platos donde poder acceder y seleccionar la información.
- Implementación de la aplicación móvil para que se puedan tomar nota desde cada mesa y simultáneamente se actualice el *stock* del restaurante.
- Implementación de la aplicación web para poder gestionar el *stock* de los ingredientes, bebidas y platos del restaurante y se actualice la base de datos al momento.
- Diseñar un aspecto visual agradable que haga una mejor experiencia del usuario, tanto en la aplicación web como en la aplicación móvil.

Todas las funcionalidades se han desarrollado de manera independiente, tratando de tener un código limpio y organizado, lo que nos permitirá añadir nuevas funcionalidades de manera cómoda y sencilla.

En cuanto a la funcionalidad de las aplicaciones, se han solventado los requisitos mínimos para poder gestionar un restaurante, pero es fácil añadir nuevas características que sean requeridas en cada establecimiento en particular.

Como trabajo futuro nos gustaría enumerar una serie de ideas que se podrían implementar:

- **Limitaciones de Firebase.** Se ha usado Firebase para la implementación, ya que es una base de datos muy práctica para proyectos pequeños. Sin embargo, se ha usado un plan de datos gratuito, el cual tiene límite de consultas y de usuarios simultáneos. Para implementarlo en un restaurante se debería pasar a un plan de pago. Si se quiere usar para grandes cadenas de restaurantes, por ejemplo, habría que estudiar el rendimiento y la posibilidad de cambiar a otra base de datos
- **Mejoras en el dominio.** Al igual que en la base de datos, se ha usado una página web que nos ofrece un dominio por tiempo limitado. Habría que estudiar la posibilidad de comprar y hacernos con el dominio *TakeOrderAdmin.es*.
- **Alertas.** Se podría añadir una funcionalidad para notificar mediante un email al encargado del *stock* de las alertas asociadas a los ingredientes y bebidas.
- **Precio y proveedores.** Implementar campos para guardar los precios y los proveedores de los ingredientes y bebidas sería muy útil.
- **Menús del día.** Sería interesante tener una serie de menús del día ya predefinidos, por ejemplo para cada día de la semana, y poder cargarlos rápidamente. También se podría añadir una funcionalidad para poder acceder a estos menús del día o a toda la carta mediante un código QR o poder imprimirlos con un formato adecuado.
- **Creación de usuarios.** Se podrían integrar la creación usuarios, de tal forma que podamos restringir algunas funcionalidades a ciertos trabajadores, o poder mostrar alguna información sensible como precios y proveedores solo al administrador del *stock*.

Introduction

This document contains all the fundamental aspects to understand the functionality of the web and mobile application designed as a Final Degree Project to facilitate the management of ingredients stock in restaurants.

The idea of the project comes from the need of a fast and efficient management when managing a restaurant, which are some of the most important qualities to achieve a successful business.

Until recently, in any restaurant, the way of carrying out the work was based on the waiter taking note of what the customer wanted with a notebook and pen, and take it to the kitchen for preparation. Even the waiter could even say the typical expression: "I'm going to check with the kitchen to see if that plate is available", although we can still see this type of management nowadays in any town or beach bar.

We are in a new digital era in which we need tools to facilitate our work, with the quality that our clients expect. The idea, therefore, is to accelerate as much as possible the organizational processes of a restaurant, so that the restaurant manager can plan and structure the service to customers, having access to the design configuration of the menu of the day, and the plates to be prepared.

In the world of gastronomy, success is based on good raw materials, a good preparation of the plates and an excellent customer service with a fast and quality service. In our case, we have focused on this last point, reducing time-consuming manual tasks that make it more difficult to concentrate on customer service with a quality service.

To achieve this objective, our project focuses, on the one hand, on a dynamic communication between kitchen and dining room employees, providing a fast and efficient management when serving customers and, on the other hand, on facilitating a simple administration management of the stock of ingredients and products.

5.1. Objetives

The software developed is focused on improving and optimizing stock management and communication between waiters and the kitchen. The objectives will be, therefore, the efficiency improvement, reduction of the time required and increase of the quality of service to customers with an automated task system combined with an extensive database where all information can be stored and accessed quickly.

The use of a web application for the management and administration of the restaurant ensures that the plates, drinks or menus available are exactly the same as the offered to the customer and that there are no errors between the cooks and the waiters. Immediate access to the database information is allowed in order to manage the restaurant's stock. Stock updating will be extremely simple, as the manager will only have to replenish the ingredients that are below the minimum levels, for which an alert is generated. This will give the manager an idea for which plates to include in that daily menu.

From the point of view of the employee, in this case the waiters, the use of a mobile application for internal communication between employees ensures direct communication with the customer by offering and informing them of the products available in both plates and drinks. This generates a fast and quality service when managing the orders for each table, without generating waits or questions in the kitchen that may upset the customer.

From the technological point of view, the objectives of this project are as follows:

- For the web application, design a software with a simple interface that facilitates the work of the application manager, such as seeing the ingredients that have generated an alert for being below minimum stock levels for a replenishment of those ingredients. In addition, you will be able to generate a menu of the day, being able to select the number of plates to be prepared for that day, based on the available ingredients, managing which plates will be offered as first course, second course and dessert.
- For the mobile application, design in the same way a simple interface that facilitates the work of the restaurant workers, both waiters and cooks. They will be able to generate an order assigned to a table being able to take note of the drinks and the plates offered by the menu, in addition to offer a menu of the day taking note of the first and second plates, and the dessert.
- Design a database that allows the storage of a large amount of information, in which we will store all the ingredients registered in the restaurant, the plates with their assigned ingredients, the drinks or the tables, each one assigned with an order, and in each order, stored the plates and drinks ordered by that table.

5.2. Work plan

Considering the objectives described above, a time estimate was made for the work plan with a duration of 7 months. In order to be able to see the work plan in a clear and simple way we decided to make a Gantt diagram. We can see work plan in the Figure 5.1 and description of the activities in the Figure 5.2

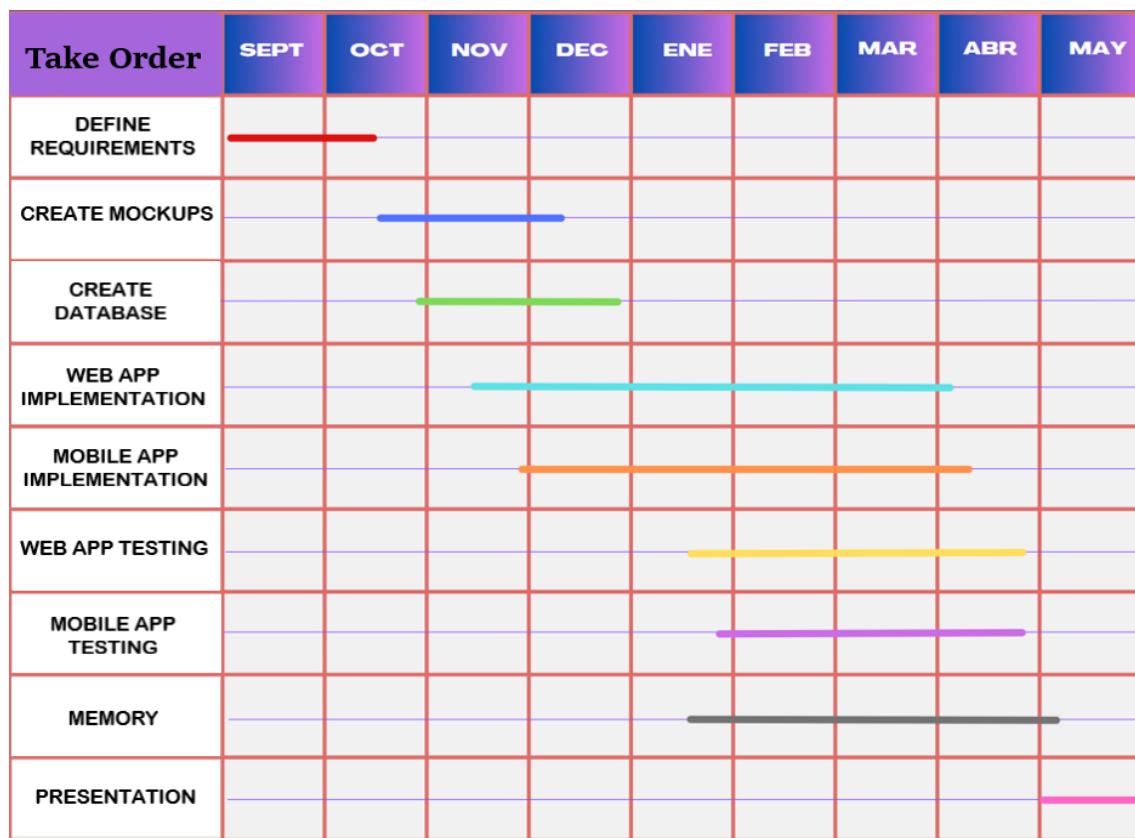


Figure 5.1: Work plan

	Define Requirements In this phase we design the basis of our application, the target and the technical requirements.
	Create mockups We design how we want the user's interaction with the interfaces to be, as well as their appearance.
	Create database We design a database to manage all the information about the restaurant. The attributes they have to have, relationships and entities.
	Web app implementation We implemented with code all the requirements specified for the web application in the first phase of the project.
	Mobile app implementation We implemented with code all the requirements specified for the mobile application in the first phase of the project.
	Web app testing We test all the functions we have implemented to ensure the smooth running of the web application.
	Mobile app testing We test all the features we have implemented to ensure the smooth functioning of the mobile application.
	Memory We describe our project from the idea, how it is developed, to its completion, including a user manual.
	Presentation Date on which we present the project to our tutor and the jury.

Figure 5.2: Work plan legend

Conclusions and Future Work

Once the project is finished, we can say that the objective of creating an application for the maintenance of restaurants has been achieved. It has resulted in a very intuitive and easy to use interface, which was the main objective. Some of the objectives have been achieved:

- Creation of different views to manage the stock of a restaurant from the owner's point of view.
- Design of a database to store the ingredients, drinks and plates where the information can be accessed and selected.
- Implementation of the mobile app so that waiters can take notes from each table and simultaneously update the restaurant's stock.
- Design a friendly visual appearance that makes a better user experience, both in the web application and in the mobile application.

All the functionalities have been developed independently, trying to have a clean and organized code, which will allow us to add new functionalities in a comfortable and simple way.

As for the functionality of the applications, the minimum requirements to manage a restaurant have been solved, but it is easy to add new features that are required in each particular establishment.

As future work we would like to list a number of ideas that could be implemented:

- **Firebase Limitations** Firebase has been used for the implementation as it is a very practical database for small projects. First of all a free data plan has been used, which has a limit of queries and simultaneous users, to implement it in a restaurant it would be necessary to change to a paid plan, if you want to use it for large restaurant chains, for example, you would have to study the performance and the possibility of changing to another database.

- **Domain improvements** As in the database, we have used a web domain that offers us a domain for a limited time. It would be necessary to study the possibility of buying and getting the domain TakeOrderAdmin.es
- **Alerts** It would be possible to add a functionality to notify by email to the owner of the restaurant when there is some ingredient or drink in alert.
- **Price and providers** Implementing fields to store prices and providers of ingredients and drinks would be very useful.
- **Menus of the day** It would be interesting to have a predefined set of menus of the day for example for each day of the week, and to be able to load them quickly.
You could also add a functionality to access these menus of the day or the entire menu via a QR code or to print them in a suitable format.
- **User creation** The creation of users could be created, so that we can restrict some functionalities to certain workers, or be able to show some sensitive information like prices and suppliers only to the restaurant owner. 5.

Contribuciones Personales

Desde un inicio decidimos repartirnos el trabajo de manera que los dos tuviéramos la misma carga. Nos hemos compenetrado muy bien fortaleciendo los puntos fuertes de cada uno. Aunque se ha dividido el trabajo de desarrollo de las aplicaciones, hemos estado siempre en contacto, consultándonos dudas y proponiendo nuevas ideas el uno al otro.

Jesús Martín Moraleda

Al principio del proyecto, consideramos invertir un tiempo en estudiar todas las tecnologías que podíamos utilizar, ya que el desarrollo web y móvil era prácticamente nuevo para nosotros. Me encargué de probar e informarme de ventajas e inconvenientes de cada una y decidir las que íbamos a usar. Del mismo modo investigué qué base de datos nos convenía usar y como conectar nuestras aplicaciones con esta.

Una vez teníamos claras las tecnologías, me dediqué a crear el proyecto de la base de datos en *Firebase* y, junto a Jorge, a conectar las aplicaciones con la base de datos. Una vez las dos aplicaciones se comunicaban correctamente con la base de datos, me encargué de crear un repositorio de GitHub para cada proyecto. Una vez hice los primeros *commits* con la estructura del proyecto y la conexión con la base de datos, Jorge empezó con el desarrollo de la aplicación móvil y yo me dediqué al desarrollo de la web.

Con todo esto listo, empecé con el desarrollo de la página web con la ayuda de los *mockups* que había realizado mi compañero. Empecé desarrollando la pantalla de los ingredientes. Mi objetivo era ir desarrollando las vistas completas y que funcionaran correctamente. Luego seguí con las pantallas de bebidas e ingredientes, y mas tarde realice las vistas del menú del día y las alertas.

Mientras hacíamos todo esto, teníamos reuniones con Mercedes, que nos fue guiando en el desarrollo del proyecto. Nos dió libertad para usar las tecnologías que quisieramos, y las funcionalidades de la aplicación las debatíamos entre los tres en estas reuniones.

En mitad del desarrollo, me puse a investigar como podíamos desplegar la página web en un dominio, para que se pudiera acceder de otra manera que desde nuestros ordenadores. Esto me llevó un tiempo ya que no habíamos tenido que desplegar una página web anteriormente.

En esta fase de desarrollo, Jorge y yo debatíamos juntos las dudas y las cosas nuevas que queríamos implementar en las dos aplicaciones. Veíamos como poder implementarlas y que tuvieran concordancia las funcionalidades de una aplicación con la otra.

La memoria la fuimos completando a medida que íbamos avanzando en las fases. Mercedes nos fue ayudando a como estructurarla e implementarla

Jorge Arévalo Echevarría

Durante la fase de análisis realizábamos reuniones con nuestra tutora, para ir dando forma y moldeando en que dirección queríamos llevar el proyecto, definiendo los requisitos y las funcionalidades que queríamos llevar a cabo. Nuestra tutora nos dio completa libertad para la selección de entornos de desarrollo o lenguajes de programación, dejándonos escoger con los que nos sintiésemos más cómodos. Además, hicimos un diseño previo de como sería nuestra base de datos que se encargaría de almacenar la información de nuestras aplicaciones. Así que por último empezamos a planificar y dividir el trabajo para investigar que entornos utilizar para el proyecto.

Durante la fase de investigación mi labor consistió en el estudio y diseño de la base de datos y de la aplicación móvil.

Respecto a la base de datos, empecé investigando los entorno conocidos y utilizados en las distintas asignaturas de la carrera como "Bases de Datos.^{en}" la que utilizábamos el entorno Oracle Database. Después vimos que era muy buen entorno para desarrollar la base de datos a nuestro gusto, pero nos surgió el problema de que los ejemplos que estábamos acostumbrados a hacer en clase trabajábamos en local, y nosotros necesitábamos una base de datos remota para poder acceder a los datos desde la página web. Jesús me informó sobre "Firestone Firebase" siendo muy fácil e intuitivo para proyectos a pequeña escala, teniendo un gran resultado en el desarrollo de sus proyectos, además de ser un entorno gratuito. Así, Jesús se encargó del diseño e instalación de la base de datos para el desarrollo de nuestro proyecto.

Respecto a la aplicación móvil, yo ya tenía experiencia respecto a entornos de desarrollo, ya que utilice Android Studio en la asignatura optativa de "Programación de aplicaciones para dispositivos móviles". Lo que más me gusto fue la libertad que proporciona a la hora de diseñar las vistas, además de usar el lenguaje de programación Java, con el que todos estamos familiarizados. Respecto a las vistas para la aplicación móvil, decidimos realizar unos *mockups* para ir diseñando y plasmando las ideas preconcebidas que teníamos en la cabeza, de como queríamos que fuese la aplicación y que funcionalidades queríamos que llevase a cabo.

En la fase de configuración y de instalación, me encargué de la instalación de la última versión de Android Studio en nuestros dispositivos. También generamos un repositorio en Github para comprobar las actualizaciones realizadas, además de controlar y guardar las versiones anteriores de nuestro proyecto de la aplicación móvil.

La fase de desarrollo se puede decir que fue la más compleja de las fases, ya que fue la que nos llevó más tiempo y esfuerzo. Durante esta fase me encargué de llevar a cabo la implementación de las ideas que habíamos preconcebido para la aplicación móvil. Lo primero que hice fue enlazar mi proyecto al repositorio de Github, para poder controlar la subida y actualizaciones del proyecto, o en caso de cometer un error, poder recuperar una versión que funcionase de forma correcta. Despues, uno de los pasos más importantes fue el de realizar la conexión del proyecto con la base de datos Firestore database, para poder obtener la información o editar campos dentro de la base de datos. A partir de aquí desarrollé todas las vistas y funcionalidades de la aplicación móvil, en colaboración con Jesús, mientras desarrollaba la página web, para ir dando forma al producto final que queríamos obtener.

La memoria del proyecto la fuimos completando mientras íbamos avanzando en las distintas fases del proyecto. Gracias a la corrección y supervisión de nuestra tutora que nos daba indicaciones de qué partes debíamos modificar o mejorar, encarrilando y dirigiendo la memoria para conseguir un nivel apto de entrega.

Bibliografía

*Y así, del mucho leer y del poco dormir, se
le secó el celebro de manera que vino a
perder el juicio.*

Miguel de Cervantes Saavedra

APACHE SOFTWARE FOUNDATION. Maven. <https://maven.apache.org/>, 2002.

BOOTSTRAP TEAM. Bootstrap. <https://getbootstrap.com/>, 2011.

BRENDAN EICH. Javascript. <https://www.javascript.com/>, 1995.

GOOGLE. Firestore. <https://console.firebaseio.google.com/>, 2010.

GOOGLE. Android studio. <https://developer.android.com/studio>, 2013.

HÅKON WIUM LIE. Css. <https://www.w3.org/Style/CSS/>, 1996.

JETBRAINS. Kotlin. <https://kotlinlang.org/>, 2010.

JOHN FREDDY VEGA. Platzi. <https://platzi.com/home/>, 2014.

KAMEL FOUNADI, SOLOMON HYKES, AND SEBASTIEN PAHL. Docker. <https://www.docker.com/>, 2013.

MICROSOFT. Github. <https://github.com/>, 2008.

MICROSOFT. Vs code. <https://code.visualstudio.com/>, 2015.

OPEN SOURCE COMMUNITY. Fly.io. <https://fly.io/>, 2017.

SUN MICROSYSTEMS. Java. <https://www.java.com/es/>, 1995.

TIM BERNERS-LEE. Html. <https://html.spec.whatwg.org/multipage/>, 1993.

TRISTAN. Sweetalert. <https://sweetalert2.github.io/>, 2022.

