PROYECTO DE FIN DE GRADO
Desarrollo de Aplicaciones Multiplataformas
(DAM)
Curso 2021/2023

Aplicación web de comida rápida

Ayoub Kebab

Autor: Carlos Soler García

Tutor: Patricio Fernández Flórez

Índice

Índice de Figuras

1. Introducción

1.a. Contextualización

En la actualidad, podemos observar una creciente tendencia en numerosos restaurantes, especialmente en los de comida rápida reconocidos mundialmente, como Burger King o McDonald's. Estos establecimientos han implementado aplicaciones que permiten a los consumidores personalizar y realizar sus pedidos tanto desde kioscos virtuales en el propio local como a través de sus teléfonos móviles. Esta innovadora forma de realizar pedidos no solo ha logrado reducir los costos operativos de las empresas, sino también agilizar la carga de trabajo de sus empleados. Gracias al constante avance y perfeccionamiento de las tecnologías, así como al desarrollo de aplicaciones multiplataforma, esta práctica se ha expandido año tras año en todo el mundo. En Málaga, concretamente, diversos restaurantes como Taco Bell, Popeyes, Burger King, McDonald's y Carl's Jr, entre otros, ofrecen a sus clientes estos servicios digitales.

1.b. Motivación

La incorporación de kioscos digitales en los establecimientos de comida rápida puede responder a diferentes motivaciones por parte de las empresas. Por un lado, estas tecnologías permiten reducir el tiempo de espera de los clientes, mejorando así la calidad del servicio en general. Además, la implementación de kioscos digitales también supone una reducción de costos para las empresas, al disminuir la necesidad de contratar y capacitar a un mayor número de empleados. Asimismo, estos dispositivos pueden mejorar la experiencia de compra de los clientes al ofrecer una interfaz visualmente atractiva, intuitiva y de fácil uso. Por último, el uso de estas aplicaciones permite recopilar datos y realizar análisis que resultan valiosos para las empresas, ya que brindan información sobre las preferencias y tendencias de compra de sus clientes. Estas motivaciones pueden haber sido algunas de las razones que impulsaron a las empresas a desarrollar e implementar aplicaciones de kioscos digitales en sus locales.

1.c. Objetivos

- Diseñar una interfaz de usuario intuitiva y fácil de usar que permita al usuario realizar un pedido en pocos pasos.
- Desarrollar una base de datos robusta y escalable que permita almacenar y gestionar los datos de los pedidos de manera eficiente.
- Incluir un panel de empleados desde donde el personal del establecimiento podrá visualizar los pedidos de una forma clara.
- Garantizar la seguridad de la aplicación mediante el uso de técnicas de autenticación y encriptación de datos.

1.d. Estado del Arte

En el mercado actual, podemos encontrar diversas aplicaciones que presentan similitudes con la diseñada en este trabajo, tanto en el ámbito de la restauración como en el de la mensajería. En el sector de la restauración, se ha observado una creciente popularidad de los kioscos digitales en establecimientos de comida rápida reconocidos a nivel mundial, tales como Burger King, McDonald's y Taco Bell. Además, las aplicaciones de pedido a domicilio como Glovo o Just Eat han ganado terreno, permitiendo a los usuarios realizar pedidos en línea y recibirlos cómodamente en la comodidad de su hogar.

Esta aplicación, al igual que otras presentes en el mercado, busca satisfacer las necesidades y demandas de los clientes, ofreciendo una experiencia de compra eficiente y conveniente. Sin embargo, cabe destacar que cada aplicación puede contar con características distintivas y enfoques particulares que buscan diferenciarse y brindar un valor añadido a los usuarios. En este sentido, el diseño y desarrollo de la aplicación objeto de este trabajo se han enfocado en optimizar la interacción entre el usuario y el proceso de gestión de pedidos, ofreciendo una interfaz intuitiva y una experiencia fluida.

En conclusión, la aplicación creada en este trabajo se enmarca en un contexto donde existen diversas aplicaciones similares en el mercado, tanto en el ámbito de la restauración como en el de la mensajería. No obstante, cada una de estas aplicaciones busca aportar su propio valor y diferenciación, adaptándose a las necesidades y preferencias de los usuarios.

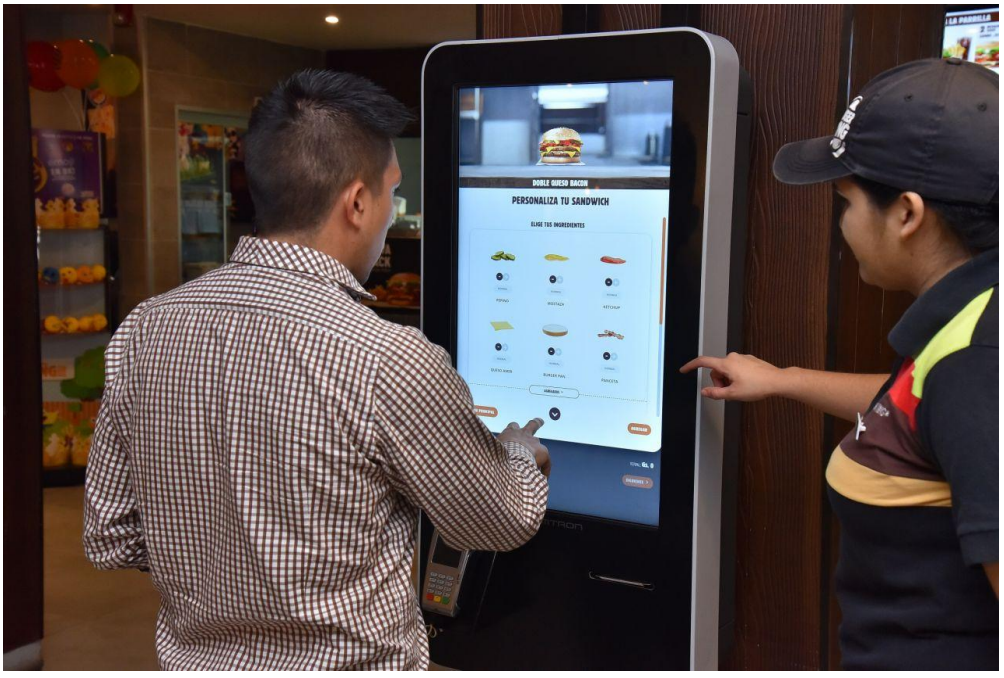

Ilustración 1 Kioscos digitales de Burger King

Ilustración 2 Kioscos digitales de McDonald's

Ilustración 3 Kioscos digitales de Taco Bell

1.e. Glosario

- **Spring Framework:** Es un framework de Java de código abierto que se utiliza para crear aplicaciones empresariales escalables y de alta calidad.

- **Spring Boot:** Es una extensión del Spring Framework que permite crear aplicaciones autónomas, autoconfiguradas y listas para usar de forma rápida y sencilla. Spring Boot automatiza muchas tareas de configuración y proporciona una configuración por defecto inteligente, lo que significa que los desarrolladores pueden centrarse en escribir código en lugar de configurar la aplicación.

- **JPA:** JPA o Java Persistence API es una especificación de Java que proporciona una forma estándar y orientada a objetos de mapear objetos Java a tablas de bases de datos relacionales. Gracias a JPA, los desarrolladores pueden realizar operaciones de persistencia de manera más eficiente y con menos código, lo que facilita la integración de aplicaciones Java con bases de datos.
- **Thymeleaf:** Es un motor de plantillas para aplicaciones web en Java. Permite a los desarrolladores definir plantillas HTML que pueden ser rellenadas dinámicamente con datos de la aplicación en tiempo de ejecución.

- **H2 database:** es una base de datos relacional escrita en Java, muy ligera y compatible con SQL. Es comúnmente utilizado para el desarrollo de aplicaciones y pruebas debido a su facilidad de uso y configuración.

- **MySQL:** Es un sistema de gestión de bases de datos relacional de código abierto y uno de los más utilizados en todo el mundo debido a su alta velocidad, escalabilidad y capacidad de manejar grandes cantidades de datos. MySQL es compatible con varios lenguajes de programación y es ampliamente utilizado en aplicaciones web, como plataformas de comercio electrónico y redes sociales.

- **REST:** Es un estilo arquitectónico utilizado para construir aplicaciones web. Permite que diferentes sistemas se comuniquen y compartan recursos entre sí a través de la web.

- **API:** Una API (Interfaz de Programación de Aplicaciones) es un conjunto de funciones y procedimientos que permite a los desarrolladores acceder a las características o datos de una aplicación, sistema operativo u otro servicio.

- **JSON:** JavaScript Object Notation (JSON) es un formato de intercambio de datos ligero y de fácil lectura para intercambiar datos entre sistemas. JSON es comúnmente utilizado en aplicaciones web y móviles para enviar y recibir datos mediante el protocolo HTTP.

- **MVC:** Modelo Vista Controlador es un patrón de diseño que separa la vista del cliente de la lógica de negocio.

- **Maven:** Es una herramienta de gestión de proyectos de software utilizada para compilar, empaquetar y gestionar las dependencias de proyectos Java.

- **Git:** Es un sistema de control de versiones de software utilizado para rastrear los cambios en el código fuente y coordinar el trabajo entre varios desarrolladores en un proyecto.

- **Persistencia:** La capacidad de una aplicación para almacenar y recuperar datos de forma duradera en una base de datos u otro sistema de almacenamiento de datos, lo que permite que los datos estén disponibles incluso después de que la aplicación se cierre o se reinicie.

- **Método handler:** Es un método en un controlador de Spring que maneja una solicitud HTTP específica y devuelve una respuesta.

- **Inyección de dependencias:** Es un patrón de diseño de software en el que los objetos no crean las dependencias que necesitan para funcionar, sino que se las proporciona un sistema externo. Esto ayuda a reducir el acoplamiento entre los componentes de una aplicación y hace que el código sea más modular y fácil de mantener. En Spring, la inyección de dependencias se realiza a través del contenedor de Spring.

- **Sprint:** En el desarrollo de software, un sprint es un período de tiempo fijo y corto durante el cual un equipo de desarrollo trabaja en la implementación de un conjunto de funcionalidades y objetivos definidos. El concepto de sprint se origina en la metodología ágil de desarrollo de software, especialmente en el marco de trabajo Scrum.

2. Planificación

2.a. Metodología de desarrollo

Se ha optado por la metodología de desarrollo incremental. Dado que se trata de un proyecto relativamente pequeño y la naturaleza del mismo puede cambiar a medida que avanza el desarrollo, este enfoque permite adaptarse a los cambios y a las necesidades del usuario final. El desarrollo se divide en cuatro pequeñas entregas funcionales llamadas "Sprints", lo que permite al desarrollador trabajar en tareas específicas en cada iteración. Además, el enfoque incremental permite una evaluación continua del software y una retroalimentación temprana por parte del usuario, lo que puede ayudar a mejorar el producto final, ya que cada Sprint contiene cuatro etapas que son Análisis, Diseño, Implementación y Pruebas. Con esta metodología se busca obtener una aplicación funcional y escalable, en la que se irán agregando nuevas funcionalidades y mejoras en cada iteración.

2.b. Planificación del Proyecto

Con el objetivo de cumplir con los plazos de entrega y los requerimientos mínimos establecidos, se ha diseñado una planificación detallada para el proyecto. Se ha decidido dividir el proyecto en varias tareas principales, las cuales serán distribuidas a lo largo de un periodo de cuatro semanas (Que a su vez representan cada Sprint). Aunque se dispone de un tiempo adicional a las cuatro semanas para la entrega final, se ha considerado prudente asignar algunos días de margen para contemplar cualquier posible retraso que pudiera surgir durante el desarrollo.

Esta estructura de planificación permitirá una adecuada distribución del trabajo y asegurará el cumplimiento de los objetivos establecidos. Cada tarea será abordada de manera secuencial,

siguiendo una secuencia lógica y priorizando los aspectos fundamentales del proyecto. Además, se establecerán hitos intermedios para evaluar el avance y realizar ajustes oportunos en caso necesario, garantizando así un desarrollo eficiente y satisfactorio del proyecto en su totalidad.

- **Sprint 1:** Se llevará a cabo un análisis de requerimientos del proyecto, la configuración de la base de datos, la creación y población de las tablas y un front-end muy básico para poder empezar a trabajar con las funcionalidades.

- **Sprint 2:** Creación de las capas lógicas de la aplicación y de sus clases. Programación de los controladores y de los Dao para plasmar los menús en la vista y registrar en la base de datos los menús escogidos por el usuario.

- **Sprint 3:** Configuración de Spring Security para diferenciar entre URLs públicas (para los clientes) y URLs privadas (para los empleados). Creación de página de error 404.

- **Sprint 4:** Desarrollar un Front-end más elegante y ameno para la experiencia del usuario final. Programación de pruebas unitarias.

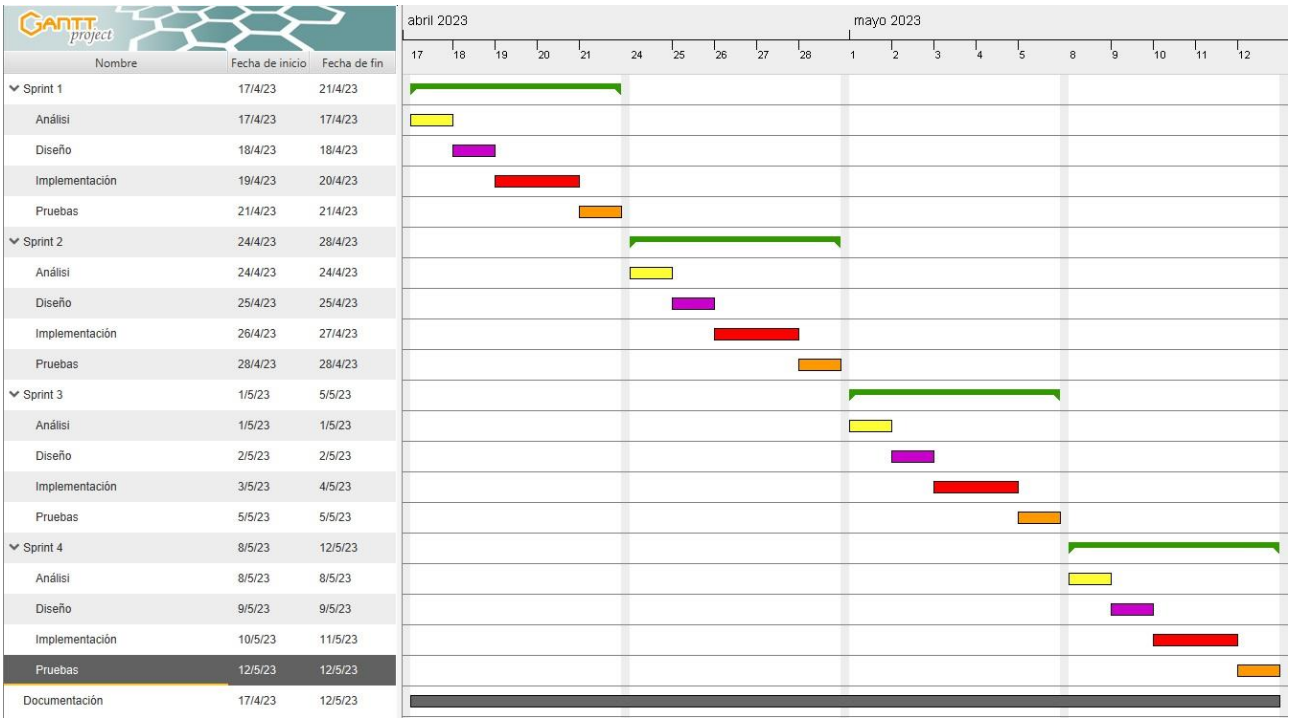

Ilustración 4 Planificación del proyecto (Diagrama de Gantt)

3. Descripción del proyecto

El apartado de Descripción del proyecto proporciona una visión general del sistema que se va a desarrollar. En esta sección, se presentan los requisitos de alto nivel que guiarán el diseño y desarrollo de la aplicación, centrándose en aspectos clave como la visualización y selección de menús, la automatización de los pedidos hacia la cocina y la optimización del tiempo de espera de los clientes. Asimismo, se detallan las tecnologías que se utilizarán en el proceso de desarrollo, asegurando una base tecnológica sólida y actualizada.

Además de los aspectos técnicos, se presta especial atención a los usuarios finales del sistema. Se destaca la importancia de proporcionar una experiencia intuitiva y agradable para los clientes, con el objetivo de facilitar la realización de pedidos y mejorar su satisfacción. De manera similar, se enfatiza la importancia de un proceso eficiente para los empleados de cocina, con herramientas que agilicen la preparación de los pedidos y optimicen la comunicación interna.

En el análisis de la descripción del proyecto también se abordan los posibles riesgos que puedan surgir durante su desarrollo. Se identifican y evalúan los riesgos potenciales, tales como retrasos en la entrega, problemas de compatibilidad tecnológica o cambios en los requisitos, entre otros. Se proponen estrategias de reducción y mitigación de estos riesgos con el fin de minimizar su impacto en el desarrollo y garantizar el cumplimiento exitoso de los objetivos del proyecto.

En resumen, la descripción del proyecto establece una sólida base para el enfoque y la planificación adecuada del desarrollo de la aplicación. Al considerar los requisitos, las tecnologías, los usuarios finales y los posibles riesgos, se sientan las bases para un diseño y desarrollo eficiente, con el objetivo de crear una aplicación que cumpla con las necesidades y expectativas de los usuarios finales y logre los objetivos establecidos.

3.a. Requisitos de alto nivel

Para el diseño y desarrollo de la aplicación, se han establecido los siguientes requisitos de alto nivel:

REQ 1: La aplicación debe permitir a los clientes visualizar de manera clara y accesible el menú completo, de modo que puedan explorar y conocer las opciones disponibles antes de realizar su pedido.

REQ 2: Se debe proporcionar a los clientes la capacidad de seleccionar los productos deseados de manera sencilla y personalizada a través de la aplicación.

REQ 3: Los pedidos realizados por los clientes deben transmitirse automáticamente a la cocina, garantizando una comunicación instantánea y precisa para agilizar el proceso de preparación de los pedidos.

REQ 4: En la pantalla principal de la aplicación, los clientes deben tener la posibilidad de visualizar un mapa detallado que muestre la ubicación precisa del establecimiento. Esto proporcionará información útil para los clientes que deseen visitar el local físicamente.

REQ 5: Se requiere un sistema de autenticación y control de acceso en el panel de administración de la cocina. Los miembros del personal de cocina podrán iniciar sesión utilizando un nombre de usuario y contraseña, lo que garantizará la seguridad y el control adecuado del panel de administración.

Estos requisitos de alto nivel han sido identificados como fundamentales para el desarrollo de una aplicación exitosa, que brinde una experiencia fluida y satisfactoria tanto para los clientes como para los miembros del personal de cocina. Su cumplimiento garantizará un proceso eficiente de

realización de pedidos, una mejora notable en los tiempos de espera y un control adecuado de la gestión de pedidos en el entorno de la cocina.

3.b. Tecnologías Usadas

El desarrollo de la aplicación web se llevó a cabo utilizando el framework Spring Boot, el cual se basa en el lenguaje de programación Java y proporciona un conjunto de herramientas para el desarrollo de aplicaciones web de manera rápida y sencilla, permitiendo a los desarrolladores centrarse en la lógica de negocio en lugar de en detalles de bajo nivel.

Para la interfaz de usuario se utilizó HTML, CSS y JavaScript por su amplia adopción y por la gran cantidad de recursos y herramientas disponibles para su desarrollo, junto con el motor de plantillas Thymeleaf, que es usado como una de las dependencias principales de Spring Boot, y Bootstrap, para facilitar el diseño y la maquetación de la página web, además de tener un diseño atractivo para el usuario.

Para la persistencia de datos se utilizó MySQL como base de datos relacional debido a su escalabilidad y rendimiento, así como su facilidad de uso y la gran cantidad de recursos y herramientas disponibles para su desarrollo. MySQL trabaja conjuntamente con Hibernate y JPA, ORMs que facilitan el mapeo de las clases Entity a las tablas, y el uso de transacciones por medio de las clases DAO.

3.c. Usuarios Finales

Aunque el cliente puede ser el empresario del establecimiento, los usuarios finales del sistema son principalmente clientes y empleados de cocina de restaurantes de comida rápida. Los clientes utilizan la aplicación para seleccionar los menús mientras que los empleados de cocina utilizan la aplicación para preparar los pedidos.

Se espera que los clientes encuentren la aplicación fácil de usar y que les permita realizar pedidos de manera rápida y sencilla, sin necesidad de interactuar con un empleado. Para los empleados de cocina, se espera que la aplicación les permita recibir y preparar los pedidos de manera eficiente, aumentando así la productividad y reduciendo los errores.

3.d. Evaluación de Riesgos

Se han identificado varios riesgos que podrían afectar al proyecto. Uno de los mayores riesgos es la complejidad técnica (R1) del proyecto, el desarrollo de una aplicación web puede presentar diversos desafíos técnicos que podrían retrasar el proyecto y aumentar los costos. Esto va ligado con la falta de planificación (R2) e incumplimiento de los plazos y la falta de presupuesto (R3). También existe el riesgo de problemas de integración (R4) con sistemas externos, como el sistema de pagos.

Todos estos posibles problemas podrían retrasar el lanzamiento de la aplicación y comprometer la calidad final del producto afectando negativamente la experiencia del usuario, además de presentar numerosos costos extras aumentando el presupuesto necesario.

- R1 (Complejidad técnica)

- **Probabilidad:** Media. Muchas tecnologías las hemos trabajado a lo largo del ciclo, y en internet hay muchísima información para enfrentar funcionalidades básicas.
- **Impacto:** Medio. Siempre se puede recortar funcionalidades menos necesarias, o buscar una alternativa más sencilla.

- R2 (Falta de planificación)
  - **Probabilidad:** Alta. Creo que al ser la primera vez que atacamos un proyecto tan grande nuestra falta de experiencia puede llegar a provocar que haya problemas de planificación.
  - **Impacto:** Medio-Bajo. Creo que se puede solventar fácilmente dejando días de margen antes del plazo final, por lo tanto si planificamos mal no creo que el impacto sea elevado.

- R3 (Falta de presupuesto)
  - **Probabilidad:** Bajo-nulo. Hay plataformas en las que puedes ejecutar el proyecto por cero costo o muy reducido, siempre que el volumen de visitas no sea elevado.
  - **Impacto:** Bajo. En caso de que hubiesen costes para este proyecto no creo que fuese nada excesivo.

- R4 (Problemas de integración)
  - **Probabilidad:** Bajo. Es muy poco probable que haya problemas de integración puesto que son tecnologías muy maduras las cuales tienen un gran soporte.
  - **Impacto:** Alto. En el caso de que se encontrasen problemas críticos podría afectar mucho a la planificación y estructura del proyecto.

3.e. Reducción de Riesgos

R1: Para minimizar el riesgo de complejidad técnica, se implementarán las siguientes estrategias:
- Se trabajará con tecnologías conocidas y se aprovechará la abundante información disponible en línea para abordar desafíos técnicos.
- Se priorizarán las funcionalidades esenciales y se considerará recortar aquellas menos necesarias.

R2: Para mitigar el riesgo de falta de planificación, se tomarán las siguientes medidas:
- Se realizará una planificación detallada del proyecto que incluirá un análisis de recursos necesarios y un cronograma de desarrollo.
- Se dejarán márgenes de tiempo antes del plazo final para anticipar posibles retrasos y ajustar la planificación en consecuencia.

R3: Para reducir el riesgo de falta de presupuesto, se considerarán las siguientes acciones:
- Se buscarán opciones de plataformas y herramientas de bajo costo o gratuitas que sean adecuadas para el proyecto.
- Se evaluarán las necesidades reales del proyecto y se evitarán gastos innecesarios.

R4: Para mitigar el riesgo de problemas de integración, se implementarán las siguientes estrategias:
- Se utilizarán tecnologías maduras con un sólido soporte y compatibilidad.
- Se llevará a cabo un riguroso proceso de pruebas y validación para garantizar una correcta integración de los sistemas.

En resumen, para reducir los riesgos identificados, se realizará una planificación detallada, se priorizarán las funcionalidades esenciales, se buscarán opciones de bajo costo y se llevará a cabo un riguroso proceso de pruebas y validación para garantizar la calidad del producto final.

4. Presupuesto

El presupuesto del proyecto se divide en diferentes categorías para abarcar los distintos costes involucrados:

4.a. Costes Directos

Los costes directos se refieren al coste del desarrollador principal. En este caso, se tiene en cuenta la contratación de un desarrollador junior para un período estimado de un mes. Considerando una tarifa acorde al mercado y la duración del proyecto, se estima que los costes directos serían de aproximadamente 1.500 euros brutos.

4.b. Costes Indirectos

Dentro de los costes indirectos, se incluyen los gastos asociados a la formación recibida tanto en Cesur como en un curso específico de Spring Boot en Udemy. La formación es fundamental para garantizar el conocimiento actualizado y las habilidades necesarias para el desarrollo del proyecto. Asimismo, se considera la inversión en el hardware y software requeridos.

- **Software:**

Software	**Precio (€)**
SpringToolSuite4	0
Visual Studio Code	0
Google Docs	0
GanttProject	0
MySQLWorkbench	0
Librerías Front-end	0
Plugins para VSC	0
Total: 0€	

Tabla 1 Costes indirectos - Software

- **Hardware:**
  En términos de hardware, se empleó un ordenador con un valor aproximado de 700 euros, que cumple con los requisitos técnicos para el desarrollo eficiente del proyecto.

4.c. Otros Costes

En esta sección se detallan los costes adicionales relacionados con el proyecto que no han sido mencionados anteriormente como costes directos o indirectos. Entre estos costes adicionales se pueden incluir los servicios de luz, el proveedor de internet, así como otros gastos operativos y administrativos necesarios para llevar a cabo el desarrollo del proyecto. Estos costes pueden variar según la ubicación, el tamaño del proyecto y las necesidades específicas del mismo.

Servicio	**Precio (€)**
Internet	70
Luz	50
Total: 120€	

Tabla 2 Otros costes

4.d. Coste Total

Para calcular el coste total del proyecto, se sumarán los costes directos, los costes indirectos y cualquier otro coste adicional mencionado en la sección de "Otros Costes". Este resultado proporcionará una estimación del coste total estimado para el proyecto. Es importante tener en cuenta que los costes pueden estar sujetos a cambios y ajustes a medida que se avanza en el desarrollo del proyecto, y es fundamental contar con un seguimiento financiero y control presupuestario adecuado para garantizar la correcta ejecución del proyecto dentro de los límites presupuestarios establecidos.

Concepto	**Precio (€)**
Costes directos	1.500
Software	0
Hardware	700
Servicios	120
Total: 2.320€	

Tabla 3 Coste total

5. Análisis de Requisitos

El análisis de requisitos es una etapa crucial en el desarrollo de cualquier sistema o aplicación. En esta sección, se detallan los actores involucrados en el sistema y sus respectivas necesidades y objetivos. Además, se presentan los requisitos funcionales y no funcionales que el sistema debe cumplir para satisfacer las expectativas de los usuarios y garantizar su eficiencia, seguridad y usabilidad. A continuación, se incluye un diagrama de casos de uso que representa visualmente las interacciones entre los actores y el sistema, identificando las principales funcionalidades y

escenarios de uso. Este análisis de requisitos sienta las bases para el diseño y desarrollo de la aplicación, asegurando que se cumplan las necesidades de los usuarios y los objetivos del proyecto.

5.a. Catálogo de Actores

En el sistema se identifican tres actores principales:

1. Cliente: Usuario final que utiliza la aplicación web para seleccionar y comprar productos. Busca una experiencia de compra rápida y eficiente.

2. Empleado de Cocina: Usuario final que utiliza la aplicación web para recibir y preparar pedidos de manera eficiente. Necesita una interfaz clara y concisa.

Cada actor tiene necesidades específicas. Por ejemplo, el cliente busca una experiencia ágil de compra y el empleado de cocina necesita recibir pedidos claros. El sistema se diseña considerando estas necesidades para ofrecer una experiencia satisfactoria.

5.b. Requisitos Funcionales

RF1. Visualización de menús: El sistema debe permitir a los clientes visualizar los diferentes menús ofrecidos por el restaurante. Satisface el requisito REQ 1.

RF2. Selección de menús: Los clientes deben ser capaces de seleccionar el menú que deseen comprar. Satisface el requisito REQ 2.

RF3. Visualización de pedidos: Los cocineros deben tener acceso a los pedidos realizados por los clientes y a los menús correspondientes para poder prepararlos. Satisface el requisito REQ 3.

RF4. Visualización del mapa de la zona: Es necesario que el cliente pueda visualizar un mapa detallado de donde se encuentra el establecimiento. Satisface el requisito REQ 4.
RF5. Seguridad en el panel de administración: Debe de haber medidas de seguridad como un formulario de login para que solo puedan acceder a material sensible personal autorizado. Satisface el requisito REQ 5.

5.c. Requisitos no funcionales

En el desarrollo de cualquier aplicación de software, los requisitos no funcionales juegan un papel fundamental para garantizar la calidad y el rendimiento del sistema. Estos requisitos se refieren a características que van más allá de la funcionalidad básica y abordan aspectos como la eficiencia, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad. Siguiendo los estándares establecidos por la norma **ISO/IEC 25010** [1], este apartado se centra en identificar y describir los requisitos no funcionales clave que han sido considerados en el desarrollo de la aplicación. Estos requisitos proporcionan un marco para evaluar la calidad del sistema y asegurar que cumpla con los estándares y expectativas requeridas por los usuarios finales.

Eficiencia de Desempeño: La aplicación proporciona tiempos de respuesta rápidos y un rendimiento eficiente incluso cuando se manejan altos volúmenes de pedidos y usuarios concurrentes. Además, los recursos del sistema, como la memoria y el procesamiento, se utilizan de manera óptima.

Compatibilidad: La aplicación es compatible con diferentes plataformas y sistemas operativos, como Windows, macOS y Linux, y funciona correctamente en distintos navegadores web. También es compatible con distintas resoluciones y tamaños de pantalla.

Usabilidad: La interfaz de usuario es intuitiva y fácil de usar, permitiendo a los usuarios navegar por las opciones de menú, seleccionar el deseado y completar el pedido sin dificultad. También los administradores pueden loguearse y visualizar los pedidos con la misma sencillez y calidad.

Fiabilidad: La aplicación es confiable y está disponible en todo momento. Es capaz de manejar situaciones de error de manera adecuada.

Seguridad: La aplicación garantiza la seguridad de los datos gracias a métodos robustos de seguridad, para de esta manera prevenir ataques maliciosos.

Mantenibilidad: El código fuente y la arquitectura de la aplicación són fáciles de entender, modificar y mantener. Se han seguido buenas prácticas de programación, utilizando estándares de arquitectura y código, para facilitar las tareas de mantenimiento y la incorporación de nuevas funcionalidades.

Portabilidad: La aplicación tiene la posibilidad de ejecutarse en diferentes entornos, tanto en local como en la nube. Es independiente de la plataforma y cumple con los estándares y protocolos relevantes para garantizar su funcionamiento en distintos sistemas operativos y configuraciones de hardware.

5.d. Diagrama de casos de uso

Un diagrama de casos de uso es una representación visual que describe la interacción entre los actores (usuarios, sistemas externos, etc.) y el sistema en cuestión. Muestra los diferentes escenarios o casos de uso en los que los actores interactúan con el sistema, ilustrando las funcionalidades y acciones que se llevan a cabo. Este diagrama permite identificar los requisitos y comportamientos clave de la aplicación, facilitando la comprensión de cómo los usuarios interactúan con el sistema y cómo este responde a esas interacciones.

El diagrama de uso propuesto plasma los dos usuarios que usan la aplicación y las funcionalidades básicas a las que tienen acceso.

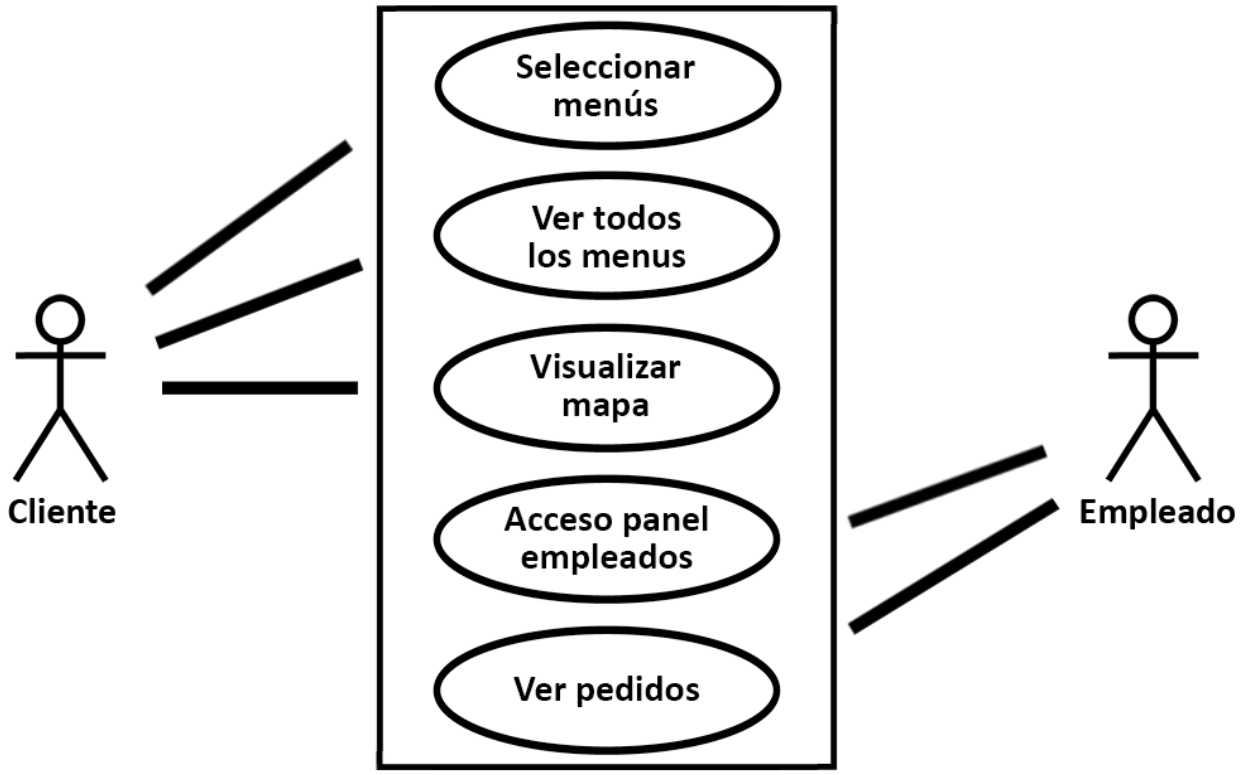

Ilustración 5 Diagrama de casos de uso

6. Diseño

En este apartado se describe el proceso de planificación y definición de cómo se estructura y funciona el sistema de software. Se plasman las decisiones tomadas sobre la arquitectura, la estructura de datos, la interfaz de usuario y otros aspectos clave para garantizar que la aplicación cumpla con los requisitos y objetivos establecidos.

El diseño de una aplicación implica diferentes niveles de abstracción, que incluyen el modelo conceptual de datos, el diseño lógico de la arquitectura y el diseño físico.

6.a. Modelo conceptual de datos

Los datos son almacenados en tres tablas. Dos de ellas tiene una relación 1:1 donde la PK de la tabla “menus” es la FK de la tabla “orders”. La tabla “employers” solo contiene las credenciales de acceso al panel de empleados.

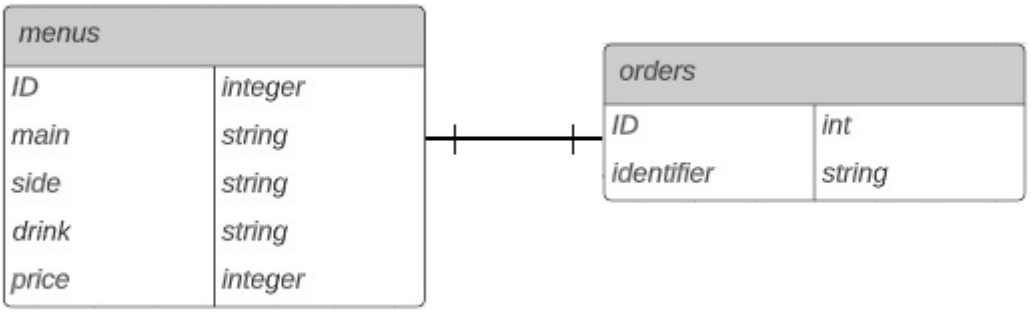

Ilustración 6 Modelo conceptual de datos

6.b. Diseño Lógico de la arquitectura

El diseño lógico de la arquitectura de una aplicación se refiere a la estructura conceptual y funcional de los componentes y su interacción en un sistema. Se enfoca en la organización de módulos, capas y componentes, definiendo las relaciones y la lógica de cómo se comunican entre sí. Este diseño busca garantizar la eficiencia, modularidad y reutilización del código, permitiendo un desarrollo y mantenimiento más sencillo y escalable de la aplicación.

La arquitectura se divide en cinco importantes módulos:

- **La vista** es la capa que ve el usuario final y con la que interactúa.

- **El controlador** es quien recibe, manda y organiza las peticiones HTTP entrantes y salientes.

- La capa **Service** o de Servicio usa el patrón de diseño Facade o Fachada. Crea una interfaz o capa de abstracción para manejar los Daos desde el controlador.

- Las clases **Dao** son las que se comunican directamente con la base de datos gracias al ORM de Hibernate y el mapeo de las clases Entity. Implementa las funciones necesarias para, por ejemplo, traer todos o uno de los menús de la tabla “menus”, escribir pedidos nuevos de los clientes, visualizar todas las comandas, entre otras funciones que sean necesarias para el CRUD de la base de datos.

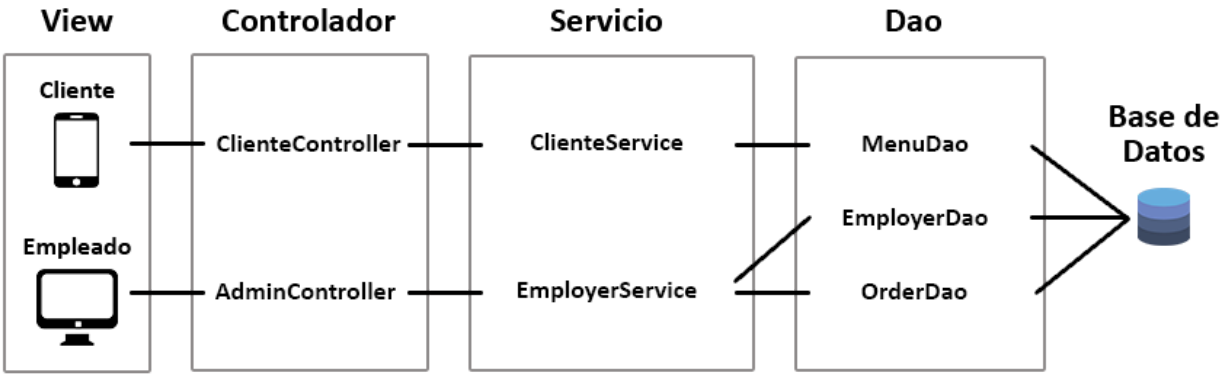

Ilustración 7 Diseño lógico

6.d. Diseño Físico

El diseño físico de una aplicación se refiere a la estructura y organización de los componentes y recursos de hardware y software necesarios para su funcionamiento. Incluye la configuración de servidores, redes, bases de datos y otros elementos tecnológicos, así como la distribución y ubicación física de los componentes.

La aplicación corre en el servidor que configuremos y será visualizada en el dispositivo del usuario final, ya sea cliente o cocina.

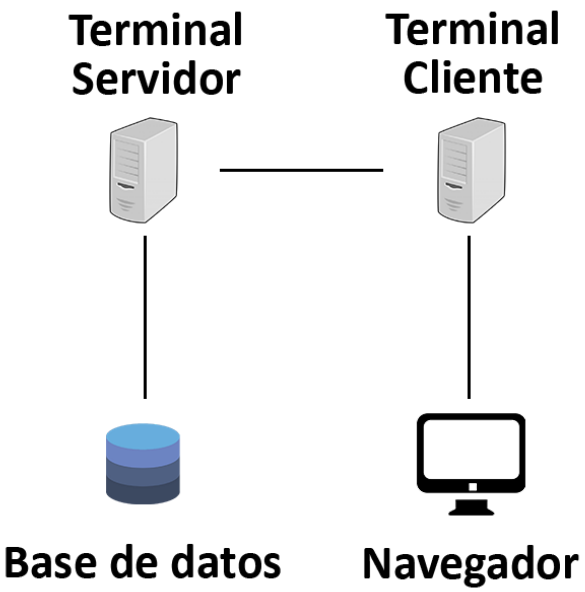

Ilustración 8 Diseño físico

7. Implementación del Sistema

Se procedió a configurar el entorno de desarrollo Spring Tool Suite 4 para asegurar un entorno de trabajo adecuado. Este entorno trae incluidas numerosas herramientas embebidas para trabajar con Spring Framework, como por ejemplo el servidor Apache Tomcat para desplegar aplicaciones webs, además nos permite incluir dependencias a nuestro proyecto de una manera muy sencilla. Se instaló y configuró el JDK (Java Development Kit) necesario para ejecutar aplicaciones Java. Se configuraron las variables de entorno pertinentes y se realizaron las configuraciones necesarias para asegurar la correcta ejecución del sistema.

Se instalaron y configuraron las dependencias necesarias, como Hibernate y JPA para la persistencia de datos con MySQL. Se utilizaron también HTML, CSS y JavaScript para la implementación de la interfaz de usuario, junto con el motor de plantillas Thymeleaf y el framework Bootstrap para facilitar el diseño y la maquetación de la página web.

Una vez configuradas todas las herramientas y el entorno de desarrollo, se procedió a la implementación del código de la aplicación. Se crearon las clases y los controladores necesarios para la lógica de negocio, la interacción con la base de datos y la gestión de las peticiones del cliente. Se implementaron las funcionalidades de visualización del menú, selección de productos por parte del cliente y envío automático de pedidos a la cocina. También se realizaron pruebas y ajustes necesarios para asegurar el correcto funcionamiento del sistema.

8. Pruebas del Sistema

En este apartado se detallarán las pruebas realizadas durante el desarrollo del sistema, incluyendo pruebas unitarias, de integración y de sistema. Se siguió un proceso metódico para evaluar el funcionamiento de cada componente y su interacción, garantizando el cumplimiento de requisitos y la identificación de posibles errores. Se registraron los procedimientos, resultados y problemas encontrados, tomando acciones correctivas. Los resultados obtenidos se presentarán de forma clara, resaltando los aspectos positivos y las áreas de mejora, con el objetivo de asegurar la calidad y el rendimiento del sistema.

El apartado de pruebas detallará los procesos realizados durante el desarrollo del sistema, incluyendo pruebas unitarias, de integración y de sistema. Se seguirá una metodología rigurosa para evaluar el funcionamiento y la interacción de los componentes, garantizando el cumplimiento de requisitos y la detección de posibles errores. Se registrarán los procedimientos, resultados y problemas encontrados, tomando las medidas necesarias para solucionarlos. Los resultados obtenidos se presentarán de forma clara, resaltando los aspectos positivos y las áreas de mejora, con el objetivo de asegurar la calidad y el rendimiento óptimo del sistema.

8.a. Pruebas Unitarias

Durante el desarrollo del sistema se llevaron a cabo pruebas unitarias para evaluar el comportamiento individual de los componentes. Estas pruebas se centran en verificar el correcto funcionamiento de cada unidad de código de manera aislada. Se diseñaron casos de prueba que abarcaban diferentes escenarios y se ejecutaron utilizando un marco de pruebas adecuado, como JUnit.

Se realizaron pruebas unitarias en las clases y métodos principales de la aplicación, validando la lógica de negocio, la interacción con la base de datos y el manejo de excepciones. Se verificaron las funcionalidades clave, como la visualización del menú, la selección de productos, el cálculo del precio total y el envío de pedidos a la cocina. Los resultados obtenidos mostraron que los componentes individuales funcionaban correctamente y se ajustaban a las especificaciones definidas.

8.b. Pruebas de Integración

Las pruebas de integración se realizaron para evaluar la interconexión y la correcta comunicación entre los diferentes componentes del sistema. Estas pruebas aseguran que los distintos módulos funcionen correctamente cuando se integran en un entorno más amplio. Se verificó la interacción entre los controladores, los servicios y la capa de persistencia de datos.

Se diseñaron casos de prueba que abarcaban diferentes escenarios de uso, incluyendo la navegación por la aplicación, la realización de pedidos y la visualización de información actualizada. Los resultados obtenidos demostraron una correcta integración de los componentes y una comunicación adecuada entre ellos.

8.c. Pruebas de Sistema

Las pruebas de sistema se llevaron a cabo para evaluar la funcionalidad global del sistema y su comportamiento en situaciones reales. Estas pruebas se centran en verificar que el sistema cumple con los requisitos establecidos y que los usuarios pueden utilizarlo de manera satisfactoria. Se probaron las funcionalidades principales, como la visualización del menú, la selección de productos, el envío de pedidos y la recepción en cocina.

Se diseñaron casos de prueba que abarcaban diferentes escenarios de uso, simulando situaciones reales. Se realizaron pruebas tanto manuales como automatizadas, siguiendo flujos de trabajo específicos y comprobando la respuesta del sistema ante diferentes situaciones. Los resultados obtenidos demostraron que el sistema funcionaba correctamente, que los usuarios podían realizar pedidos de manera eficiente y que los empleados de cocina recibían los pedidos de manera adecuada.

En resumen, se realizaron pruebas exhaustivas durante el proceso de desarrollo del sistema. Las pruebas unitarias evaluaron el comportamiento individual de los componentes, las pruebas de integración aseguraron la interconexión entre ellos, y las pruebas de sistema validaron la funcionalidad global del sistema. Los resultados obtenidos en todas las pruebas fueron satisfactorios, demostrando un sistema funcional y confiable.

9. Conclusiones

En este apartado se presentarán las conclusiones obtenidas tras el desarrollo del proyecto, incluyendo las lecciones aprendidas, los objetivos cumplidos y el trabajo futuro.

9.a. Lecciones Aprendidas

Durante el desarrollo del proyecto se han adquirido diversas lecciones que resultan valiosas para futuros proyectos. Entre las lecciones aprendidas se incluyen:

- La importancia de una planificación adecuada: Quedó claro que una planificación detallada y realista es fundamental para llevar a cabo un proyecto de manera efectiva. Esto incluye definir los requisitos, establecer los plazos y recursos necesarios, y tener en cuenta posibles obstáculos o desafíos.

- La relevancia de las pruebas exhaustivas: Se ha reforzado la importancia de realizar pruebas en todas las etapas del desarrollo, desde pruebas unitarias hasta pruebas de integración y pruebas de sistema. Esto garantiza la calidad del sistema, identifica errores y permite corregirlos antes de la puesta en producción.

9.b. Objetivos Cumplidos

Durante el desarrollo del proyecto se lograron alcanzar los objetivos establecidos. Entre los objetivos cumplidos se incluyen:

- Diseño de una interfaz de usuario intuitiva y fácil de usar: Se logró crear una interfaz de usuario amigable y accesible que permitió a los clientes realizar pedidos en pocos pasos, tal como se planteó en el primer objetivo.

- Desarrollo de una base de datos robusta y escalable: Se implementó una base de datos sólida y eficiente que almacenó y gestionó los datos de los pedidos de manera eficiente, cumpliendo así con el segundo objetivo establecido.

- Creación de un panel para empleados: Se diseñó y puso en funcionamiento un panel de para empleados intuitivo que permitió al personal del establecimiento visualizar los pedidos de forma sencilla, cumpliendo así con el tercer objetivo planteado.

- Garantía de seguridad mediante técnicas de autenticación y encriptación de datos: Se implementaron medidas de seguridad efectivas, incluyendo técnicas de autenticación y encriptación de datos, para salvaguardar la integridad y confidencialidad de la aplicación, en línea con el cuarto objetivo establecido.

9.c. Trabajo Futuro

Existen diversas posibilidades de trabajo futuro para mejorar y ampliar el proyecto en futuras versiones. Algunas de estas posibilidades son:

- Mejora de la eficiencia del sistema: Se puede realizar un análisis más profundo del rendimiento del sistema y realizar mejoras para optimizar su velocidad y eficiencia. Esto puede incluir implementar técnicas de caché o utilizar tecnologías de escalabilidad horizontal, como arquitectura de microservicios.

- Incorporación de nuevas funcionalidades: Se pueden agregar nuevas funcionalidades al sistema para enriquecer la experiencia del cliente y ofrecer opciones adicionales. Esto puede incluir la

integración de métodos de pago adicionales, la implementación de un sistema de recomendaciones personalizadas o la adición de funciones de reserva de mesas.

- Adaptación a dispositivos móviles: Se puede considerar la adaptación de la aplicación a dispositivos móviles mediante el desarrollo de una aplicación móvil o la creación de una versión responsive de la interfaz de usuario. Esto permitiría a los clientes realizar pedidos de manera conveniente desde sus dispositivos móviles

10. Referencias Bibliográficas

[1] Normativa ISO/IEC 25010: https://iso25000.com/index.php/normas-iso-25000/iso-25010

11. Anexos

11.a. Manual

A continuación, se presenta el manual de usuario que acompaña a la aplicación desarrollada.A través de este manual, se pretende facilitar la comprensión y el uso de la aplicación, permitiendo a los usuarios navegar de manera fluida y satisfactoria en todas sus funcionalidades.

Para empezar hablaremos de las funcionalidades del cliente, el cual en la pantalla principal podrá echar un vistazo de los menús disponibles, observar sus características y seleccionar el que más le interese.

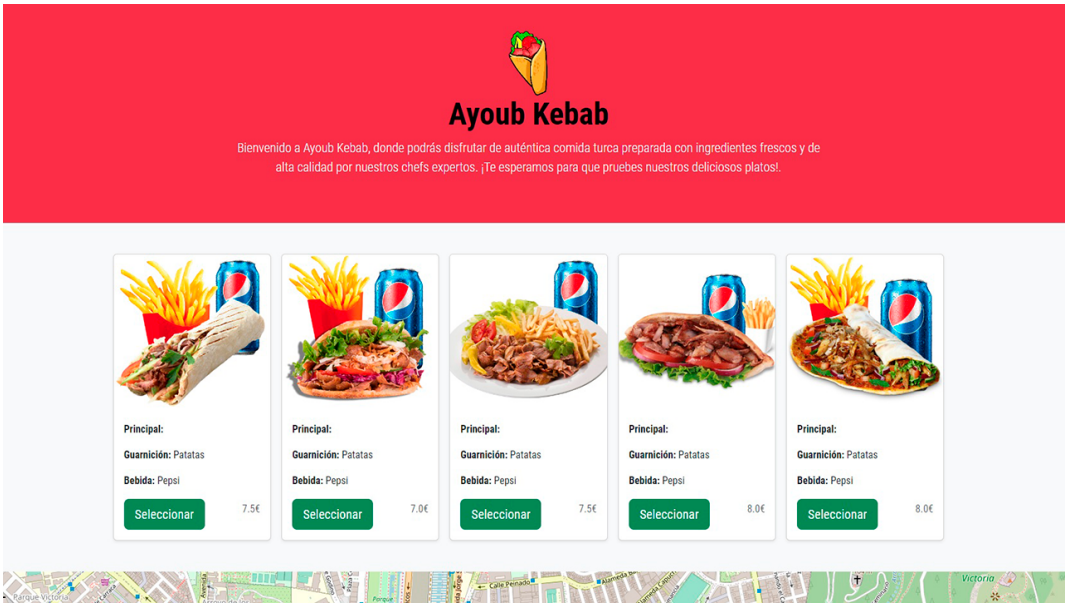

Ilustración 9 Página principal

También en la pantalla principal, en la parte inferior, podrá visualizar un mapa en el que se mostrará la localización física de la tienda.

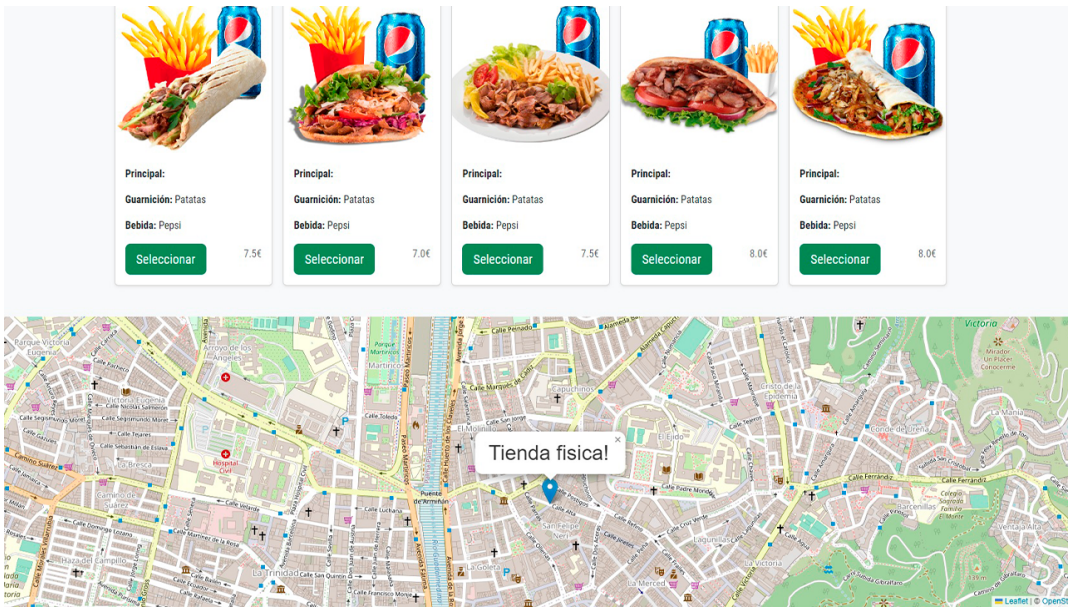

Ilustración 10 Mapa

Una vez seleccionado su menú le aparecerá una ventana de confirmación en la que si no está de acuerdo con la selección con solo pulsar el botón de “Quiero otro” le devolverá a la ventana anterior para seguir con el proceso de selección. Si por el contrario si es el menú que desea deberá hacer click en el botón “Si, estoy de acuerdo”

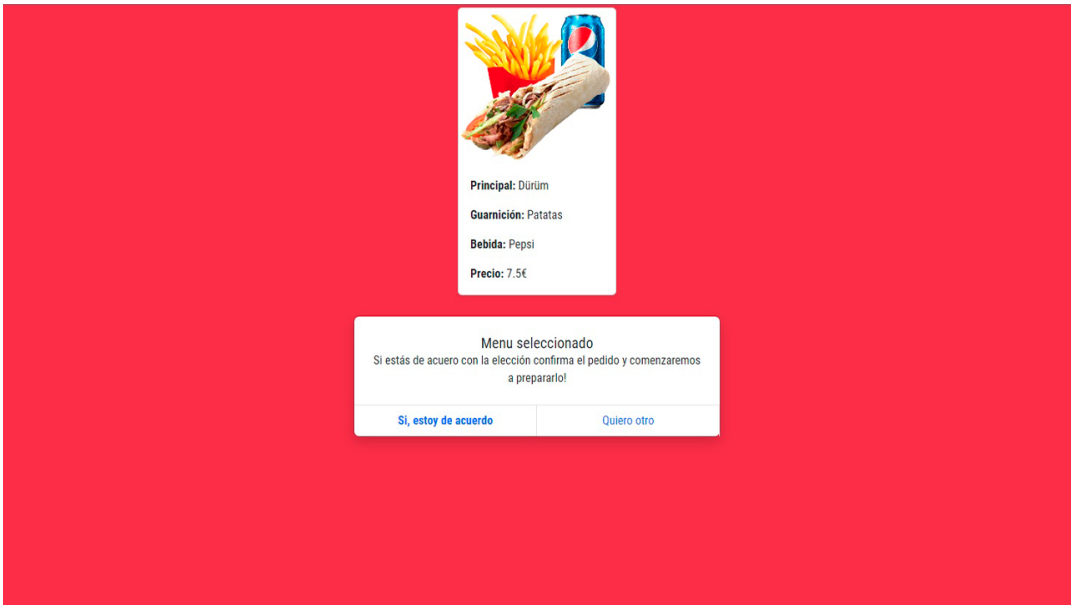

Ilustración 11 Selección de menú

Cuando el cliente ya haya confirmado su menú le aparecerá esta ventana confirmando que el menú ha sido aceptado correctamente y que su número identificatorio, con el cual podrá reconocer su pedido, es el D2059. Cada pedido realizado tendrá un número único y aleatorio.

Ilustración 12 Código de Pedido

Ahora pasemos al lado de los empleados. Lo primero que podemos observar es que todas las vistas reservadas al área de administración necesitarán pasar un filtro de login por seguridad. En la demo las credenciales serán “admin” y “admin”. La idea es que se cambien estas credenciales e incluso que cada empleado tenga las suyas propias.

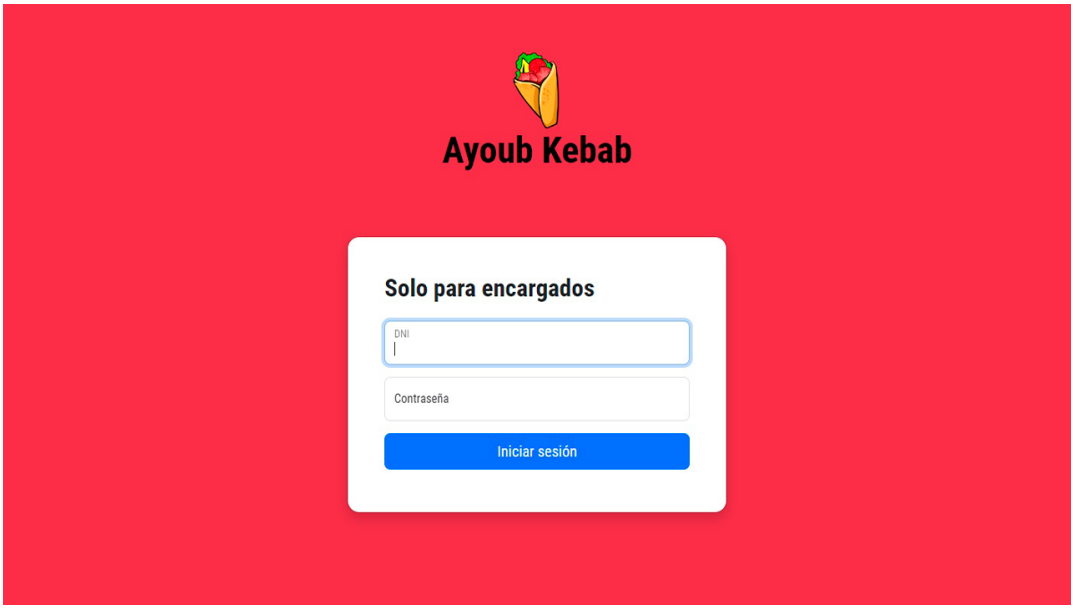

Ilustración 13 Login

Una vez dentro del panel podremos observar los pedidos solicitados por los clientes, con su número identificatorio en el que sale el del cliente del ejemplo (D2059), y también podremos ver que alimentos debemos de preparar para ese menú concreto.

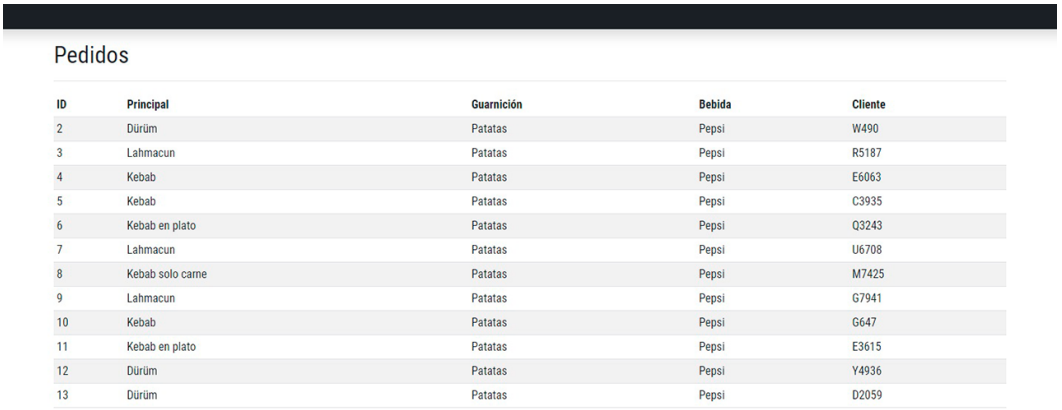

Ilustración 14 Lista de pedidos

11.b. Porciones de código explicado

Ejemplo de método handler. El método utiliza la anotación @GetMapping() para señalar que es un método get para la URL /menus. Este método retorna la plantilla menus.html y el resultado del método findAll() de la clase clienteService.

```
@GetMapping("/menus")
public String list(Model model) {
        model.addAttribute("menus", clienteService.findAll());
        return "menus";
}
```

Ejemplo de los métodos dentro de la clase de configuración de Spring Security. En el primero se crea un rol de usuario llamado Admin y se marcan sus credenciales base que son “admin” y “admin”. En el segundo método se declaran cuales son las URLs y los recursos públicos y cuales son reservados para el usuario con rol Admin.

```
@Bean
public UserDetailsService userDetailsService() throws Exception {

        InMemoryUserDetailsManager manager = new InMemoryUserDetailsManager();
        manager.createUser(
                User.withUsername("admin")
                        .password(passwordEncoder().encode("admin"))
                        .roles("ADMIN")
                        .build());
        return manager;
}

@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

        http.httpBasic().and().authorizeHttpRequests()
        .requestMatchers("/", "/menus", "/seleccionar-menu", "/peticion-aceptada", "/static/**", "/error/**").permitAll()
        .requestMatchers("/admin/**").hasRole("ADMIN")
        .and()
```

```
            .formLogin().loginPage("/login").permitAll()
            .and()
            .logout().permitAll();
            return http.build();
}
```

Clase Entity de los pedidos la cual representa la tabla del orders de la base de datos, se puede saber gracias al decorador **@Table** y su valor **name**. También se plasman cual es el campo ID de la tabla y la relación que tiene con el campo ID de la tabla menu, que es de 1:1

```
@Entity
@Table(name = "orders")
public class Order implements Serializable {

            private static final long serialVersionUID = 1L;

            @Id
            @GeneratedValue(strategy = GenerationType.IDENTITY)
            private int ID;

            @Column(name = "identifier")
            private String identifier;

            @OneToOne
            @JoinColumn(name = "menu_id", referencedColumnName = "ID")
            private Menu menu;

            public Order() {
            }
            public Order(String identifier, Menu menu) {
                        this.identifier = identifier;
                        this.menu = menu;
            }
}
```