TRABAJOS FIN DE GRADO

Universidad de Jaén

Escuela Politécnica Superior de Jaén

APLICACIÓN WEB PARA GESTIÓN DE COMANDAS

Autor: Patricio Ortega Higueruelo

Grado: Ingeniería Informática

Director: Carlos Javier Ogayar Anguita

Departamento del director: Informática

Fecha: 11/09/2024

Agradecimientos

Primero, quisiera dar las gracias a mi tutor por todos sus consejos a la hora de realizar este trabajo, por todo lo que me ha aportado y me ha ayudado en el transcurso de este trabajo. Ha sido un inmenso placer trabajar en este proyecto con él.

En segundo lugar, me gustaría dar las gracias a mi familia por darme el apoyo necesario en aquellos momentos en los que sientes que no estás capacitado para una carrera de esta magnitud y en los que sientes que la vida no te premia como mereces y tienes unos baches emocionales difíciles de superar.

En tercer lugar a la Universidad de Jaén por proporcionarme todos los recursos necesarios para formarme como estudiante, tanto personalmente como profesionalmente y por su compromiso con el estudiantado.

Tras un camino largo y en muchas ocasiones costoso, estoy orgulloso de poder finalizar mis estudios con este trabajo y poder dedicarme laboralmente a lo que siempre me ha gustado, el mundo de las tecnologías y la comunicación.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Informática Empresarial
Idioma	Español
Tipo	Específico
TFT en equipo	No
Autor/a	Patricio Ortega Higueruelo
Fecha de asignación	
Descripción corta	Este trabajo consiste en la realización de una aplicación web para la gestión de comandas en hostelería. El proceso debe ser útil y sencillo mediante una experiencia de usuario (UX)eficiente, con el objeto de facilitar el trabajo a los empleados y dar un servicio más rápido y fiable a los clientes.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 7 (Definiciones y abreviaturas).

1.1 Introducción

El trabajo se trata de una aplicación web para la gestión de comandas para hostelería, más dedicado para pequeños, medianos bares y restaurantes.

Con esta aplicación se pretende introducir la digitalización en los bares y restaurantes, facilitar el trabajo y flujo de comandas de la hostelería, ya que mediante una aplicación web, habrá transferencias rápidas de las comandas de los clientes con los administradores o dueños del establecimiento.

Además, se integrarán gráficas y datos en un panel de administrador con el objetivo de maximizar el rendimiento del restaurante y obtener el máximo beneficio posible, pudiendo analizar esos datos y, a partir de ellos, poder tomar decisiones en aquellos aspectos que no tengan tanto éxito como en aquellos que tengan éxito.

Tener un conocimiento constante de la interacción del usuario con las comandas y saber cuáles son las que tienen más fama, cuáles las que tienen menos, es fundamental para el devenir de un negocio, en este caso el devenir del restaurante.

1.2 Objetivos del trabajo

El objetivo del trabajo es implementar un sistema que pueda ayudar a los dueños de restaurantes y bares, que no tienen suficientes recursos para administrar el local y que no tengan los conocimientos adecuados de qué le puede gustar al cliente, con este sistema, al cabo de un poco tiempo de uso, podrá conocer lo que le gusta al usuario, al igual que podrá cambiar su estrategia de uso.

Otro de los objetivos finales es facilitar a los clientes con la comanda, ya que en muchas ocasiones, debido al cúmulo de personas que puede haber en ese momento almorzando o cenando, tienen que esperar durante una gran cantidad de tiempo, por lo que con este sistema, en cuánto se sienten en el restaurante, podrán registrarse en el sistema y así realizar la comanda al instante o no tener que esperar a que pueda venir un camarero a atenderles debido a la alta afluencia de clientes.

Por último, este sistema servirá para poder integrar análisis de datos en la aplicación.
El administrador podrá ver datos relacionados con las comandas realizadas por el usuario, así podrá tomar decisiones en cuánto a poder quitar un plato del menú, poder añadir uno nuevo ya que a los clientes le gusta más carne que pescado o si a un cliente le gusta más una bebida u otra.

Con esto, los dueños o administradores podrán predecir el volumen de cantidad de los determinados platos que van a necesitar y así no verse en la obligación de tirar comida o gastarse una cantidad de dinero por una comida que no se va a consumir, produciendo pérdidas en la empresa, que a lo largo podría llevar al cierre del establecimiento.

En definitiva, lo que se busca con esta aplicación es maximizar los ingresos del restaurante, minimizar las pérdidas tanto materiales como monetario y que el flujo de consumo sea lo más rápido posible, es decir, no haya pérdidas de tiempo para los clientes porque no les atienden, llevando esta situación hacia el enfado y muy seguramente a la pérdida de un cliente para posible visita con posterioridad.

1.3 Antecedentes y estado del arte

Este trabajo viene dado desde la experiencia de ir a muchos bares pequeños y restaurantes y tener un servicio deficiente, ver cómo se tira una gran cantidad de comida debido a que no se consume a lo largo del día, ver cómo debido a que son fiestas y hay mucha cantidad de gente, tardan mucho en pedirte la comanda produciendo un enfado considerable en el cliente y, por lo tanto, el cliente posiblemente no vaya una próxima vez a ese establecimiento.

Ante la frustración de tanto clientes como dueños de los bares y cocineros con la imposibilidad de poder servir y atender a sus clientes como ellos quisieran, surge la idea de producir este sistema para intentar corregir todos los problemas comentados anteriormente.

Primeramente, se estudió el hecho de hacer una aplicación simplemente para comanda, pero esto solo subsanaría el problema del posible enfado de los clientes ante la tardanza en poder pedir su comanda, por lo que no era suficiente para solucionar los problemas de las previsiones de diferentes categorías de comida que se puede gastar a lo largo de la semana.

Finalmente, se consideró tener en cuenta el problema del desperdicio de comida y se optó por integrar datos que van a ser analizados por los dueños del restaurante y a partir de ellos, tomarán las decisiones oportunas para que no haya tanto desperdicio de comida.

Existen aplicaciones que pueden suponer servir de fuente de inspiración para el desarrollo del proyecto.

Algunas aplicaciones comerciales que hay actualmente en el mercado puede ser:

- **Comandas**: Una de las opciones más sencillas que puedes encontrar. A su favor cuenta con la ventaja de solo tener que instalarla en tu dispositivo

Android y hacerla funcionar. Pero no se le puede pedir mucho, ya que no permite la variación de comandas. Es decir, si un cliente pide una pizza sin un ingrediente o desea la merluza sin gambas, la app no puede registrarlo. Pero eso no es problema para que se convierta en una propuesta interesante para cierto tipo de locales.

- **Camarero TPV**: Aquí ya nos encontramos con una app más completa, con múltiples opciones de configuración y una interfaz sencilla y amena.

  Una app de comandera efectiva, sencilla de utilizar por el personal de sala y de cocina. Incluso puedes sacar las comandas por una impresora de tickets. Lo que se echa en falta es poder tener acceso al inventario. Sin esto, el camarero puede vender platos que ya no están disponibles. Salvo ese detalle, puede ser una opción que te interese.

- **OctoTable**: Este software es quizás una de las propuestas más completas que puedas encontrar. No es una app, sino una web en la que te registras y puedes hacer casi de todo.

  Desde crear la carta y generar un QR para poner en las mesas, hasta llevar la gestión de los pedidos. Incluso el cliente puede mandar la comanda directamente a la cocina sin la intervención del personal de sala. Con OctoTable puedes además gestionar tanto comandas para tu local como para pedidos a repartir. Cuenta con 2 planes de pago, con un coste de 29 y 49 euros al mes.

1.4 Descripción de la situación de partida

El punto inicial del trabajo es de un conocimiento limitado del sector hostelero, simplemente un conocimiento adquirido por la experiencia y consumo en ellos pero, que a través de la observación y algunos problemas que fueron surgiendo en las diferentes visitas a establecimientos, sobre todo en épocas del año concretas donde suele haber mucho turismo o acumulación de personas debido a fiestas regionales o nacionales, hicieron ver como un problema general y que podría tener una solución para una satisfacción de ambas partes, tanto clientes como empresarios.

En cuanto a conocimiento humano, se parte de un conocimiento del sector pero no de las tecnologías que se pueden usar para el desarrollo de la aplicación web, por lo tanto, es un punto en contra ya que se ha tenido que realizar un aprendizaje elaborado y profundo de las tecnologías envueltas en esta aplicación.

En cuanto a la experiencia al igual que con el conocimiento humano, no se tenía una experiencia en cuánto a trabajo en el sector hostelero, ya sean de familiares o amigos.

Si me refiero a la experiencia en la creación de software sería limitada, debido a que es la primera vez que uso las tecnologías que he usado para la aplicación web.

El equipamiento hardware es asequible, no se necesita disponer de una tecnología avanzada ni dispositivos electrónicos que estén fuera del alcance, simplemente con una portátil y la instalación de las tecnologías necesarias se puede desarrollar el sistema.

No hace falta licencia software, al ser un programa para pequeños y medianos restaurantes, será un programa de licencia gratuita, no se necesitará desembolsar una cantidad de dinero por su uso.

Para su puesta en marcha en los establecimientos, sólo hará falta el uso de un ordenador para administrar los sistemas y de unos dispositivos portátiles en las

mesas para que los clientes puedan hacer su comanda desde la misma mesa, metiendo sus credenciales, sin necesidad de desplazamiento hacia otra zona.

El punto de partida en cuanto a situación hostelera en cuanto al uso de alguna ayuda digital suele ser muy limitado por lo general.

Los hosteleros se basan en una gestión de comandas básica mediante el camarero recogiendo las solicitudes mediante libreta, transmitiendo esa comanda a cocina o gente de la barra y, en función de la afluencia de clientes que tengan en el momento específico, tardan más o menos en poder gestionar esas comandas.

Por lo que, para facilitar todo el trascurso desde que el cliente entra al establecimiento hasta que sale, una ayuda digital es importante en aspectos como la satisfacción del cliente (no tiene que esperar mucho tiempo en poder recibir lo que pide), estrés por parte de los trabajadores del establecimiento (disminuye su estrés por la facilidad que le permite esta gestión).

Por último, para poder administrar el sistema, el dueño del establecimiento o la persona que se encargará de hacer el mantenimiento y administración, deberá tener un conocimiento en informática, a la vez que un conocimiento del manejo de la aplicación web, que podrá ser aprendido mediante la lectura de una manual de uso.

1.4.1 Descripción del entorno actual

El entorno hostelero, actualmente está sometido a una incertidumbre debido a la pandemia que tanto ha sacudido el sector y que, debido a las medidas a las que han estado obligados a estar sometido, ha sido muy complicado poder sacar adelante sus negocios.

El miedo de los clientes a las acumulaciones en el interior de un establecimiento, junto con las pérdidas económicas producidas por tener su establecimiento cerrado, hacen ver al empresario diferente vías de ingreso como puede ser la comida a domicilio.

Por lo tanto, este sistema podrá integrarse en el establecimiento proporcionando una ayuda a los hosteleros, ya que con este sistema se podrá recuperar pérdidas económicas producidas por la crisis del coronavirus, además de las diferentes situaciones en las que se ven involucrados estos establecimientos.

1.4.2 Resumen de las deficiencias y carencias identificadas

A partir de la experiencia de consumir en los bares, restaurantes, tabernas, he detectado como en muchos casos, los clientes están enfadados por el mal servicio, ya que puede haber problemas de falta de personal suficiente para poder servir a la gran cantidad de gente que puede visitar el establecimiento en el mismo día gracias a épocas de turismo alto como puede ser verano o fiestas nacionales, regionales de los lugares.

Otro de los problemas fundamentales es la falta de provisionamiento de los restaurantes, ya que, en muchos establecimientos, debido a la gran afluencia que tienen se agotan un producto determinado o, en el caso contrario, debido a que no tiene mucha afluencia en esa época o que el producto no ha sido muy demandado se tiene que tirar este producto, suponiendo pérdidas materiales y económicas al establecimiento.

Si ese producto a la semana se compra 30KG y sólo se consume 15KG en esa semana, al ser un producto fresco del que no se puede alargar en el tiempo, se produce unas pérdidas de 15KG, suponiendo como he indicado anteriormente, pérdidas económicas brutales y más para un establecimiento pequeño, que puede llegar a vivir de épocas de turismo o incluso, de día a día.

Otro de los problemas detectados, en un mundo en el que muchos trabajadores tienen que ir a almorzar o cenar a los restaurantes ya que tienen un tiempo para comer determinado y no puede desplazarse a su casa para comer, es el hecho de que tienen que tener un servicio rápido, un servicio que no haga esperar más del tiempo que tiene la persona.

Con este software, el cliente que tenga un tiempo determinado para alimentarse, podrá gestionar su tiempo a su propia manera, sin depender de que le tomen la comanda con un retraso que puede hacer que tenga que comer con prisas, sin poder consumir el producto de la manera que el cliente quisiera.

Con este software se busca la organización, producción, gestión de los bares y restaurantes, pudiendo conseguir una reducción en los costes, a la vez que una mejora del servicio y de la calidad final del establecimiento.

Los sistemas informáticos contribuyen al mundo empresarial una herramienta que les pueda proporcionar una ayuda considerable y que, por lo tanto, pueda contribuir a unos mejores resultados en términos de rendimiento y económicos.

1.5 Requisitos iniciales

Los requisitos iniciales del sistema los podemos diferenciar entre requisitos funcionales y no funcionales:

Los requisitos funcionales son los siguientes:

-El sistema debe proporcionar al cliente un sistema de autenticación para que pueda acceder al sistema con un usuario.

-El sistema debe dar al cliente la posibilidad de interactuar con este de manera fácil e intuitiva.

-El sistema debe recoger las comandas realizada por los clientes, de manera que se pueda hacer un posterior análisis de estas comandas.

-El sistema debería diferenciar entre diferentes categorías de comida, es decir, debería diferencias entre primeros platos, segundos platos y postres.

-El sistema debería poder diferenciar entre aquellos usuarios que son clientes normales y aquellos que van a ser los administradores.

-El sistema debe proporcionar al administrador la posibilidad de gestionar los diferentes platos que quiere servir en su establecimiento.

-El sistema debe ayudar al administrador a tener un conteo de los trabajadores que tiene en el establecimiento.

Los requisitos no funcionales son los siguientes:

-El sistema debe ser compatible con todos los ordenadores posibles del mercado.

-El sistema debería poder ser ejecutado en cualquier sistema operativo posible.

-El sistema debería tener un mantenimiento asociado ante posibles problemas de rendimiento, problemas de realización de las funciones dentro del sistema.

-El sistema debe ser usable, fácil de mantener y con un acceso visible para el usuario.

-El sistema debe tener una alta disponibilidad de, al menos, 99,9% disponible durante todo el tiempo efectivo.

-El sistema debe tener un procedimiento de DRP(Disaster Recovery Plan) ante cualquier circunstancia que pueda ocurrir natural y que pueda dejar sin disponibilidad la aplicación.

1.6 Alcance

Los contenidos entregables del trabajo serán tanto los entregables que hacen referencia a un producto final (incluyendo los resultados de los estudios teóricos y experimentales), como los entregables que hacen referencia a la gestión y control de la ejecución del proyecto o trabajo, por lo que serán los siguientes:

-Documentación del proyecto: Documentación con los aspectos técnicos del trabajo, es decir, todo lo relacionado con el trabajo cómo puede ser: Objetivo del proyecto, requisitos del programa, tecnologías utilizadas, metodología de desarrollo software, entre otros aspectos técnicos del software implementado.

-Manual para la instalación de la aplicación, junto con el código fuente suministrado.

1.7 Hipótesis y restricciones

El TFG se define como una asignatura de 12 créditos, lo que supone que la duración total del proyecto será de 300 horas, incluyendo todas las etapas del ciclo de vida, con la excepción del mantenimiento. Por consiguiente, la principal restricción aplicable es la limitación de la duración del trabajo.

En aprendizaje de las herramientas involucradas en la realización del trabajo ha sido un periodo aproximado de 6 meses.

Se parte de la base de un conocimiento muy bajo de las tecnologías usadas, por lo que el tiempo dedicado para el aprendizaje de estas ha sido fundamental para poder desarrollar todo el software, además de todos los patrones usados para las diferentes partes del software.

En cuanto al coste, en su gran mayoría se limita en el coste temporal pero también se puede añadir un coste económico asociado a la realización de los diferentes cursos para aprender las tecnologías usadas.

1.8 Estudio de alternativas y viabilidad

Las alternativas que he tenido presente a lo largo de la ejecución del proyecto han sido las siguientes:

-Hacer una aplicación dedicada simplemente a cafeterías, con la comanda de cafés simplemente, pero, el limitar a una cafetería me parecía que no cumplía lo que yo tenía pensado al principio cuando se planteó el proyecto, que como expliqué anteriormente, era mejorar el desarrollo y vida de la hostelería.

No era del todo viable ya que, las cafeterías no suele ser un lugar de mucho atasco de afluencia, ya que es un servicio rápido y eficiente, por lo que, los problemas detallados anteriormente no se ven reflejados perfectamente en este caso.

-Hacer una aplicación dedicada para pizzerías, pero, al igual que la alternativa comentada anteriormente creo que no se ajustaba a lo que yo iba buscando, ya que, las pizzerías se mueven más en el propósito de comida a domicilio, por lo que, al igual que con las cafeterías, el atasco de personas en un establecimiento y la limitación de servicio no se va a cumplir de manera tan clara como en un restaurante o bar tradicional.

Se ha tenido en cuenta para la realización del proyecto tecnologías como Tailwind CSS, el uso de bases de datos relacionales como, por ejemplo, MySQL.

Se pensó usar como framework React JS ya que está en auge, cada vez es más usado para el desarrollo web y es uno de los frameworks punteros en cuánto a elaboración de webs complejas y bien estructuradas.

En cuanto a lenguaje se pensó usar simplemente Javascript pero, la mejora de este con el lenguaje Typescript, hace que se haya desechado la opción de usar solamente Javascript.

1.9 Descripción de la solución propuesta

Ante el análisis anterior de las alternativas a la solución propuesta y la viabilidad de estas alternativas, finalmente se procedió a la elección de las comandas para restaurantes y bares.

La solución propuesta es un sistema de comandas para un restaurante en el que el modo de uso es un usuario introduce sus credenciales, previamente mediante registro para poder hacer almacenamiento de datos más exactos, tras este acceso con su cuenta al sistema, podrá directamente seleccionar las categorías deseadas e ir añadiendo esta a su cesta de pedidos y finalmente, cuando haya realizado todo, poder pagar mediante el sistema y confirmar el pedido.

Tras este paso, el pedido llegará al administrador o dueño que se dedique a esta labor, que lo procesará y enviará esta comanda a los cocineros que lo cocinará, tras haber salido la comanda para el cliente, el administrador podrá cambiar el status a finalizado y se guardará ese pedido para el posterior análisis de datos que se podrá realizar semanalmente, pudiendo establecer una metodología (esta parte se deja a gusto del administrador o dueño) para ir variando los diferentes platos que se van sirviendo por categorías.

Con esta solución, se proporciona un flujo de tiempo establecido, sin que se produzca una gran demora en el servicio, pudiendo satisfacer las necesidades del cliente y pudiendo mantener ese cliente para más tiempo.

Finalmente, las tecnologías usadas para el desarrollo del proyecto han sido las llamadas Mean Stack, formadas por el framework Angular, el framework de nodeJS llamado express, el mismo NodeJS y como base de datos no relacional MongoDB.

Los lenguajes usados han sido HTML, Bootstrap como framework CSS y typescript como lenguaje utilizado para toda la lógica de la aplicación web.

1.10 Tecnologías utilizadas

Para este proyecto se ha usado la tecnología llamada MEAN STACK.

MEAN STACK se llama al conjunto de tecnologías formadas por: MongoDB, Express, Angular y NodeJS.

Las tecnologías utilizadas para este proyecto son las siguientes:

1. El framework de desarrollo de la aplicación es Angular: es un framework Javascript para la creación de páginas web SPA (Single Page Application).

Que una web sea SPA quiere decir que cuando el usuario entra en la web se carga todo el contenido de todas las páginas a la vez. Esto quiere decir que la primera carga nada más entrar es más lenta pero luego los cambios entre páginas son instantáneos.

Características o ventajas principales que proporciona Angular:

-Sigue el patrón MVC (Modelo - Vista - Controlador), con la vista separada de la lógica de negocio.

- Basado en componentes, es decir, piezas de código con vista que puedes reutilizar en otras páginas.

- Programación reactiva, la vista se actualiza automáticamente a los cambios en las variables.

- Inyección de dependencias, un patrón de diseño que se basa en pasar las dependencias directamente a los objetos en lugar de crearlas localmente.

Dentro del lenguaje Angular, se ha usado las siguientes tecnologías:

-**HMTL:** es un lenguaje de marcado que define la estructura de tu contenido.

HTML consiste en una serie de elementos que usarás para encerrar diferentes partes del contenido para que se vean o comporten de una determinada manera. Las etiquetas de encierre pueden hacer de una palabra o una imagen un hipervínculo a otro sitio, se pueden cambiar palabras a cursiva, agrandar o achicar la letra, etc.

-**Bootstrap**: es un Framework CSS y Javascript para dar estilo y apariencia a las páginas web.

Además, ofrece un amplio abanico de herramientas y funciones, de manera que los desarrolladores podemos crear prácticamente cualquier tipo de sitio web haciendo uso de los mismos.

-**Typescript**: es un lenguaje de programación de alto nivel que implementa muchos de los mecanismos más habituales de la programación orientada a objetos, pudiendo extraer grandes beneficios que serán especialmente deseables en aplicaciones grandes, capaces de escalar correctamente durante todo su tiempo de mantenimiento.

En definitiva, estas tecnologías han sido las usadas para hacer la parte Frontend del proyecto, es decir, las del lado del cliente.

2. La tecnología usada para la Base de Datos es MongoDB:

   MongoDB es una base de datos orientada a documentos. Esto quiere decir que, en lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.

   Una de las diferencias más importantes con respecto a las bases de datos relacionales, es que no es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla de una base de datos relacional -, pueden tener esquemas diferentes.

3. Para la implementación del backend he usado tanto NodeJS como Express:

   -**NodeJs**: Node.js utiliza un modelo de entrada y salida sin bloqueo controlado por eventos que lo hace ligero y eficiente (con entrada nos referimos a solicitudes y con salida a respuestas). Puede referirse a cualquier

operación, desde leer o escribir archivos de cualquier tipo hasta hacer una solicitud HTTP.

La idea principal de Node.js es usar el modelo de entrada y salida sin bloqueo y controlado por eventos para seguir siendo liviano y eficiente frente a las aplicaciones en tiempo real de uso de datos que se ejecutan en los dispositivos.

-**Express**: Es el framework web más popular de Node, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).

Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.

Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.

Al principio del proyecto, durante su análisis y diseño se pensó si usar Angular u otros frameworks como React.

Finalmente, se decidió a utilizar Angular debido a la gran demanda que tenía en ese momento por parte de las empresas contratantes y esto me hizo ver como una buena oportunidad para aprender esta tecnología y tener ese plus ya ganado en el futuro para poder encontrar un empleo más fácilmente relacionado con el desarrollo web.

Otro de los puntos a favor del uso de angular es su documentación, al igual que su fácil instalación en el ordenador y su sencilla entendimiento de cómo se divide su estructura.

Para la realización del proyecto se utiliza el entorno de desarrollo Visual Studio Code.

-**Visual Studio Code**: es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, 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 sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Las características de Visual Studio Code son las siguientes:

- Multiplataforma.
- Uso de control de versiones.
- Extensiones.
- Intellisense.

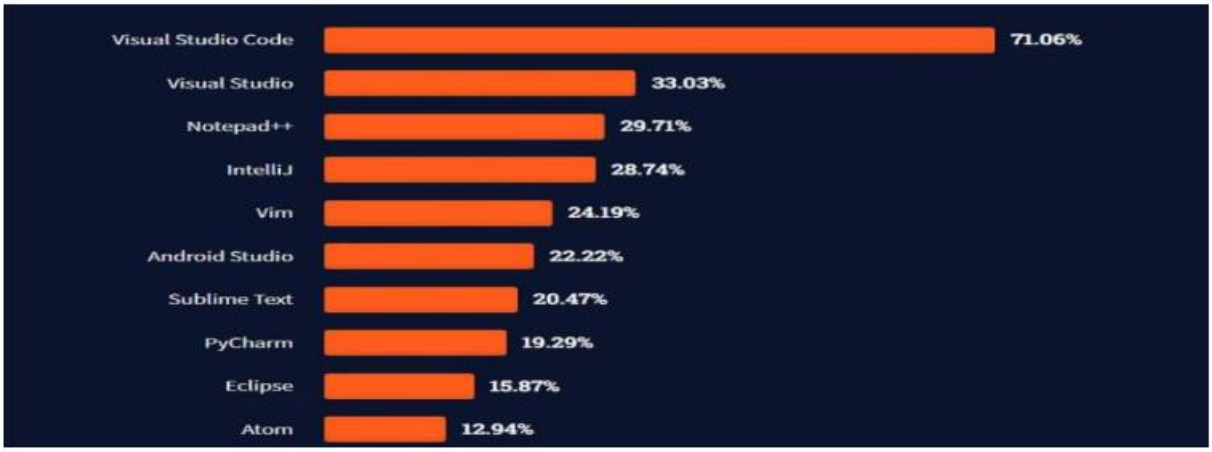


Ilustración 1.1: Gráfico de IDE más usados

Como se puede ver en la utilización, es el editor de código más utilizado por los desarrolladores, según la encuesta realizada en 2021 por la página StackOverflow.

Una herramienta adicional usada como programa soporte para las operaciones relacionadas con el backend es Postman:

-**Postman**: Se trata de una herramienta dirigida a desarrolladores web que permite realizar peticiones HTTP a cualquier API.

Postman es muy útil a la hora de programar y hacer pruebas, puesto que nos ofrece la posibilidad de comprobar el correcto funcionamiento de nuestros desarrollos.

Con esto no queremos decir que Postman sea una herramienta exclusiva para profesionales del entorno web, de hecho, va a ser muy útil para todo aquel que tenga que interactuar con una API.

Fuera de su objetivo principal, hacer peticiones a servicios, nos ofrece un conjunto de funcionalidades que nos ayudarán a organizar las peticiones en colecciones, hacer y automatizar pruebas, mantener equipos sincronizados y crear Mocks de APIs.

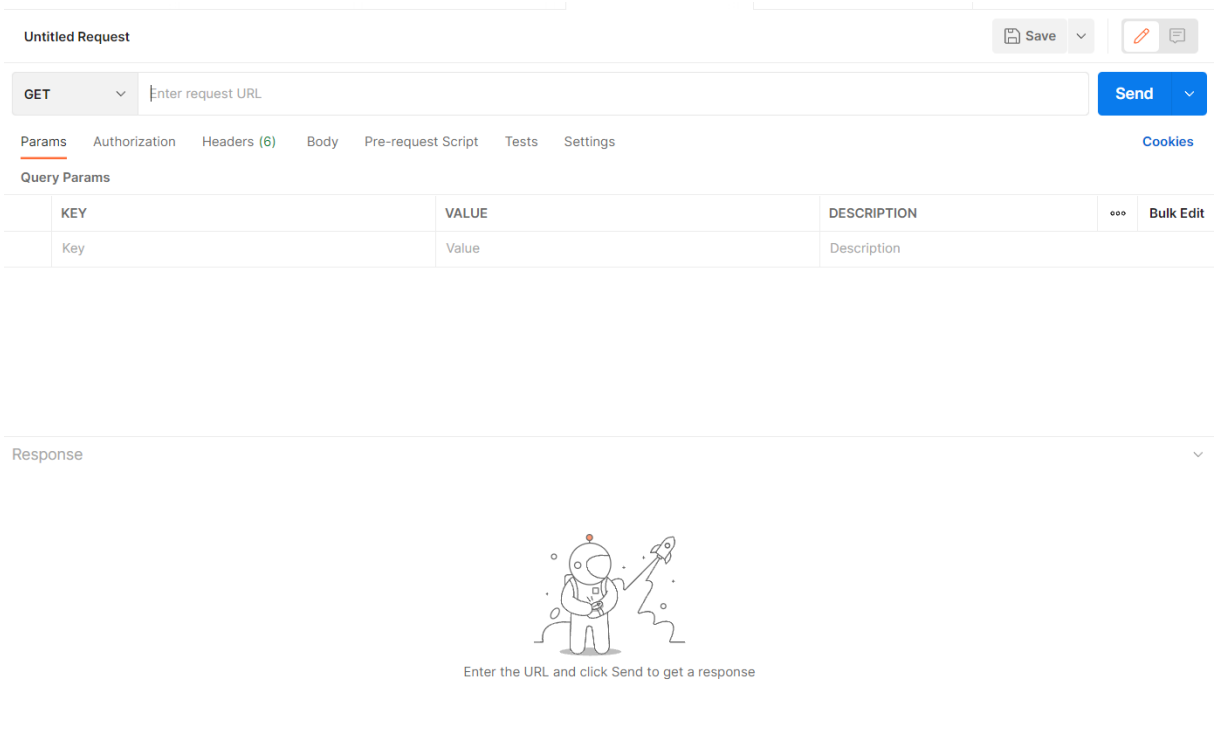


Ilustración 1.2: Interfaz de Postman para la prueba de la API

-**Sweet Alert 2**: Librería utilizada para la creación de notificaciones dinámicas en la parte del frontend.

Nos permite tener una gran variedad de notificaciones pop up con las que jugar y dar información al usuario de los diferentes pasos que va dando dentro de la aplicación web.

Se puede introducir para partes como registro del usuario, login del usuario, creación de diferentes objetos dentro del frontend como puede ser Plato, Categoría, entre otros…, además de otras funcionalidades extras que nos proporciona la librería.

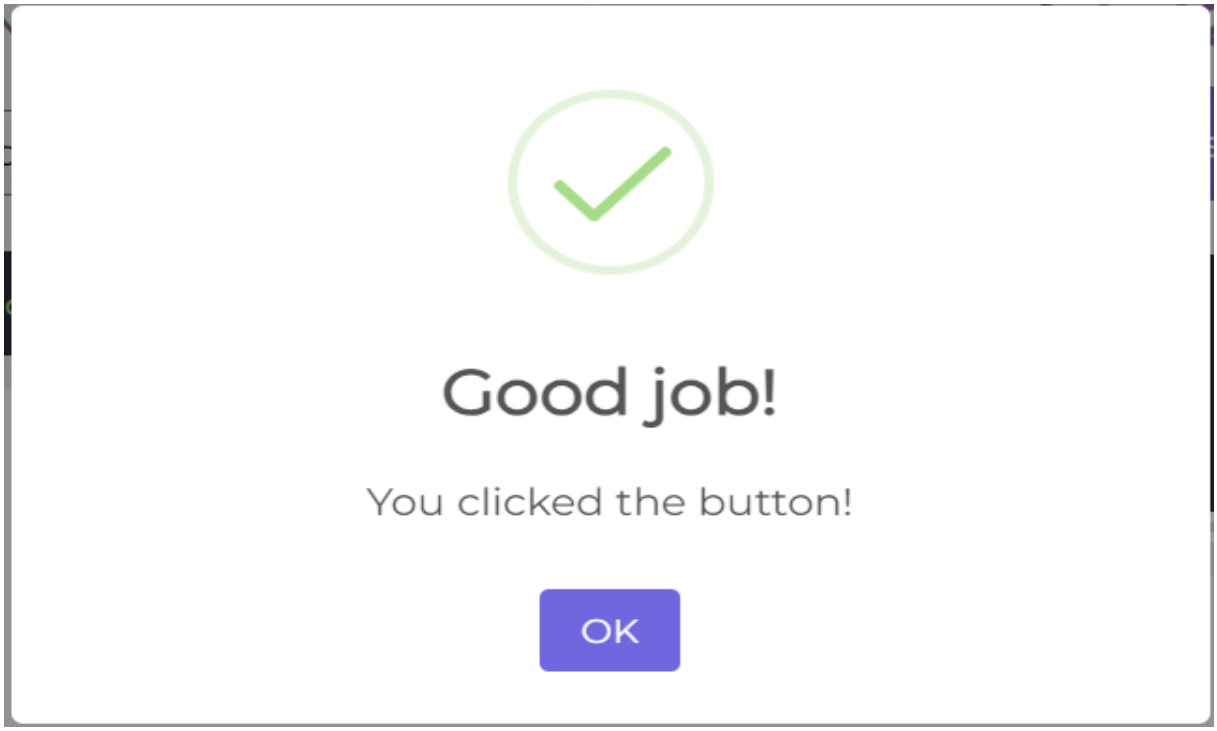


Ilustración 1.3: Ejemplo de la librería Sweet Alert 2

-**PDFkit**: Es una librería de generación de documento PDF para NodeJS que la hace muy completa y fácil de usar. Se pueden crear tanto documentos simples o complejos, en función de la finalidad para la que la queramos tener.

-**Nodemailer**: Es un módulo de NodeJS que permite el envío de correos de forma sencilla y automatizada.

Ejemplo de envío de correo:

Ilustración 1.4: Ejemplo de la librería PDFkit

-**Multer**: Es una librería de NodeJS para la implementación de subidas de imágenes al servidor y poder hacer interacción con el frontend.

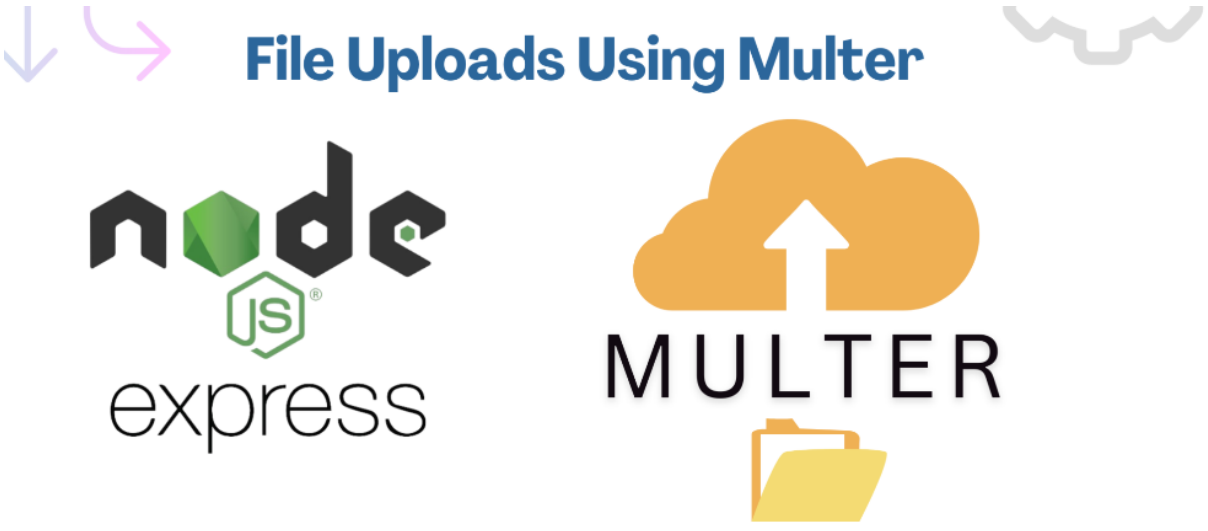


Ilustración 1.5: Ejemplo de la interacción Multer

1.11 Metodología de desarrollo de software

Uno de los aspectos más importantes para el desarrollo de software es la metodología a usar.

Existe una gran variedad de alternativas en cuánto a metodologías para el desarrollo de software, pero, en mi caso, la opción elegida es la metodología de desarrollo incremental o progresiva.

Esta metodología es iterativa y creciente, es decir, en cada iteración se va añadiendo nueva funcionalidad al proyecto, conforme se tiene establecido en el análisis del proyecto.

El modelo incremental de gestión de proyectos tiene como objetivo un crecimiento progresivo de la funcionalidad, es decir, el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario.

Este enfoque, que se usó inicialmente para proyectos de software, aunque más tarde se aplicó a otros sectores, establece entregas parciales mediante un calendario de plazos. En cada una de ellas, el producto debe mostrar una evolución con respecto a la fecha anterior; nunca puede ser igual.

La principal diferencia del modelo incremental con los modelos tradicionales es que las tareas están divididas en iteraciones, es decir, pequeños lapsos en los cuales se trabaja para conseguir objetivos específicos.

Con los modelos tradicionales no pasaba esto; era necesario esperar hasta el final del proceso.

El modelo de gestión incremental no es un modelo necesariamente rígido, es decir, que puede adaptarse a las características de cualquier tipo de proyecto, existen al menos 7 fases que debemos tener en cuenta a la hora de implementarlo:

- **Requisitos**: son los objetivos centrales y específicos que persigue el proyecto.

- **Definición de las tareas y las iteraciones**: teniendo en cuenta lo que se busca, el siguiente paso es hacer una lista de tareas y agruparlas en las iteraciones que tendrá el proyecto. Esta agrupación no puede ser aleatoria. Cada una debe perseguir objetivos específicos que la definan como tal.

- **Diseño de los incrementos**: establecidas las iteraciones, es preciso definir cuál será la evolución del producto en cada una de ellas. Cada iteración debe superar a la que le ha precedido. Esto es lo que se denomina incremento.

- **Desarrollo del incremento**: posteriormente se realizan las tareas previstas y se desarrollan los incrementos establecidos en la etapa anterior.

- **Validación de incrementos**: al término de cada iteración, los responsables de la gestión del proyecto deben dar por buenos los incrementos que cada una de ellas ha arrojado. Si no son los esperados o si ha habido algún retroceso, es necesario volver la vista atrás y buscar las causas de ello.

- **Integración de incrementos**: una vez son validados, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto en su conjunto. Cada incremento ha contribuido al resultado final.

- **Entrega del producto**: cuando el producto en su conjunto ha sido validado y se confirma su correspondencia con los objetivos iniciales, se procede a su entrega final.

1.12 Análisis de riesgos

Los riesgos establecidos a la hora de realizar el trabajo son varios, ya que, en un primer momento se establecen unas pautas o unas tecnologías para usar pero que, cuando empiezas a usarlos o a intentar aprender buscando información y cursos detallados sobre la tecnología, no se adecúan a las necesidades que se va buscando para el proyecto.

-La primera aproximación para la realización del proyecto era con la tecnología Angular como framework pero como BBDD la idea era usar Firebase.

- ¿Qué es Firebase?

Es una plataforma digital diseñada para facilitar el desarrollo de aplicaciones web y móviles de calidad de una forma rápida y eficiente, con el objetivo de mejorar el rendimiento de las mismas a través de la implementación de sus distintos módulos que harán que la aplicación sea mucho más manejable, segura y fácil de utilizar para los usuarios.

Empecé buscando información para aprender a usar esta BBDD pero me di cuenta de que la información era escasa o no se ajustaba a lo que yo necesitaba aprender para poder usar en mi proyecto, por lo que deseché esta opción, decidiendo usar la tecnología MEAN STACK que he explicado anteriormente.

Estudiando MEAN STACK me di cuenta de que había más información de este grupo de tecnologías, sobre todo de Mongodb.

El impacto sobre el proyecto fue considerable, ya que me hizo trastocar todos los planes y aprender otro conjunto de tecnologías nuevas, por lo que se puede

considerar que fue un impacto en término temporal, me llevó un tiempo que desperdicié y no pude sacar provecho.

-Otro segundo impacto, que me llevó una pérdida temporal fue el hecho de primero querer separar los proyectos del panel de administrador del propio restaurante, en proyectos totalmente diferente, llevándome a finalmente realizarlo todo en un mismo proyecto, separándolo en los diferentes componentes que nos da la posibilidad de hacer Angular.

1.13 Estimación del tamaño y esfuerzo

Ya que el presente proyecto es un TFG, no existen restricciones de tipo económico, sino de tipo temporal (un número aproximado de horas). Por consiguiente, los cálculos de tamaño del proyecto están supeditados el tiempo disponible. En cuanto al esfuerzo, se dispone de tan un solo efectivo (la persona autora del trabajo).

Para hacer una estimación del tamaño y esfuerzo en el proyecto, podemos basarnos en COCOMO para establecer qué puntos son los más importantes en cuanto al objetivo final.

1.14 Planificación temporal

Para la planificación temporal, usaré un diagrama de Gantt para representar cómo se ha estructurado temporalmente el proyecto.

El proyecto ha sufrido un retraso temporal de lo estipulado inicialmente debido a las dificultades para poder seguir el proyecto debido a circunstancias personales que han conllevado retraso no deseados inicialmente.

Para hacer el diagrama de Gantt nos apoyaremos de una herramienta llamada **Tomsplanner**, que es un software gratuito, fácil de usar y con funcionalidades básicas para la creación del diagrama.

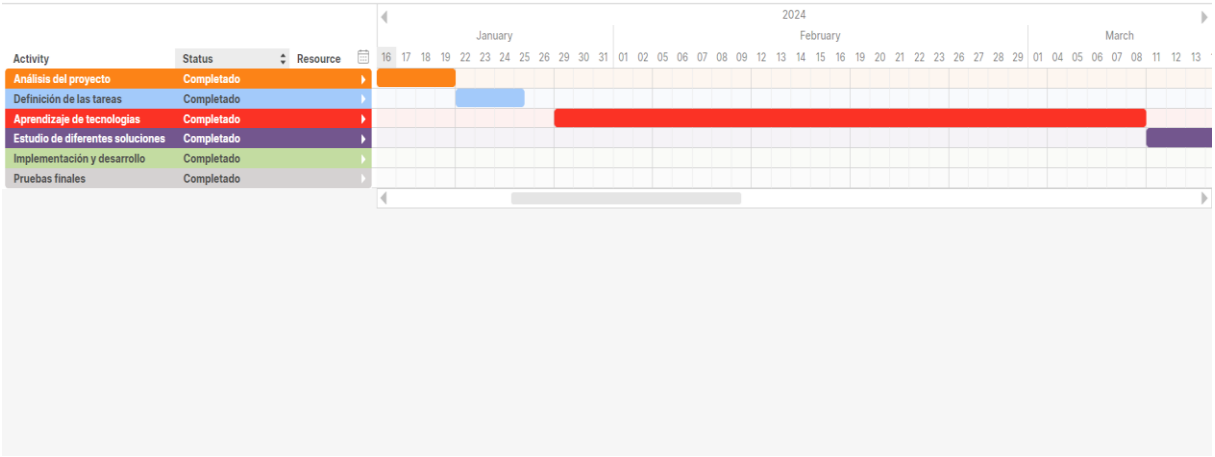


Ilustración 1.6: Diagrama de Gantt (PARTE 1)

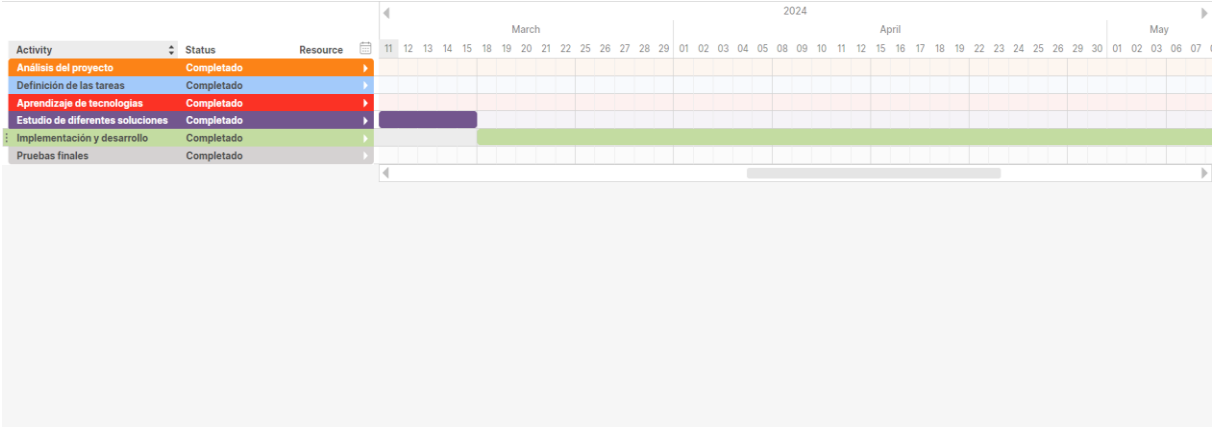


Ilustración 1.7: Diagrama de Gantt (PARTE 2)

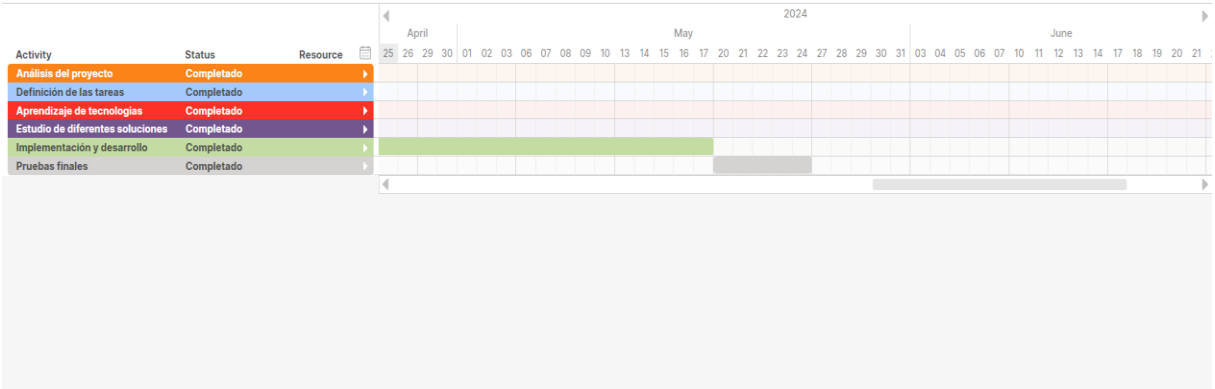


Ilustración 1.8: Diagrama de Gantt (PARTE 3)

Visto anteriormente el diagrama de Gantt, voy a explicar en qué consiste cada fase:

- **Análisis del proyecto**: Fase inicial del proyecto donde estudié y planifiqué qué quería elaborar y hacia donde quería llevar el proyecto.

- **Definición de las tareas**: Una vez realizado el análisis del proyecto, tenía que definir cuáles iban a ser las tareas a realizar a lo largo del proyecto.

- **Aprendizaje de las tecnologías**: Una vez decidido cuáles iban a ser las tecnologías usadas para el proyecto, como no tenía un conocimiento desarrollado de ellas, tenía que aprender mediante cursos y tutoriales su funcionamiento para poder elaborar la aplicación.

- **Estudio de diferentes soluciones**: Una vez aprendida las tecnologías, era momento de aplicar ese aprendizaje y elaborar qué soluciones eran las mejores para el desarrollo de la aplicación.

- **Implementación y desarrollo**: Conlleva la gran parte del proyecto, consistiendo en la implementación y en el desarrollo del proyecto.

- **Pruebas Finales**: Una vez realizada la aplicación web, se realizan una serie de pruebas finales para comprobar el correcto funcionamiento.

1.15 Presupuesto

El proyecto consta de un número duración en cuanto a horas de 300 pero al ser un proyecto mucho más elaborado, se ha estimado que, para implementar todas las funcionalidades pensadas al inicio, estamos hablando de una duración aproximadamente de 10 meses.

Con estas 300 horas establecidas, se ha decidido cuáles eran las funcionalidades más importantes, las cuáles dan esa solución pensada a los clientes finales.

El proyecto se estructura en unas 15 semanas en lo que se refiere a las etapas estrictamente específicas del proyecto.

Si sumamos el tiempo indirecto invertido en la realización de cursos y tutoriales para la realización del TFG, el trabajo se puede ir aproximadamente a unas 30 semanas.

El presupuesto vendrá dividido entre las siguientes partes:

- **Aplicación**: En esta parte se determina el coste en cantidad de horas realizadas para el proyecto, como he comentado anteriormente, al estar establecidas en 300 horas, el coste del trabajo será de aproximadamente 15 semanas simplemente para el proyecto en sí y si sumamos lo indirecto como son los cursos, se puede ajustar un coste temporal de 30 semanas.

- **Hardware**: El coste por hardware viene dado por el ordenador donde se ha realizado la aplicación, ya que no se ha realizado compra de ningún sistema aparte de hardware como puede ser algún servidor.

- **Software**: Para la realización del programa se han utilizado softwares de uso gratuito como son: principalmente Visual Studio Code, Postman y MongoDB atlas.

A continuación, se detalla mediante una tabla, todos estos apartados bien detallados con el coste de las partes involucradas en la realización del proyecto, además de partes indirectas como son los cursos realizados, los costes indirectos dados por el coste de la luz, coste de reparación de algunos componentes electrónicos.

En definitiva, estos costes indirectos pueden suponer un coste adicional del 10% de los costes ya previamente estipulados.

En esta tabla se va a tomar en cuenta que la amortización es a 5 años

Presupuesto		
Estructura	**Componente**	**Coste**
Aplicación	Coste analista-programador	4000 euros
Hardware	Ordenador utilizado Coste de luz Servidor Web	1000 euros 300 euros 90 euros
Software	Office BBDD	50 euros 50 euros
Total		5490 euros

Tabla 1.1: Presupuesto de coste

En definitiva, el coste sin aplicar % en IVA ni otros costes indirectos viene dada por la tabla anterior.

A continuación, se calculan estos costes añadiendo los costes indirectos.

Presupuesto	
Añadidos	**Coste**
Coste	5490 euros
Sobrecoste indirecto (10%)	549 euros
Total	**6039 euros**

Tabla 1.2: Presupuesto de coste amortizado

Con esto se ha calculado el coste que ha habido a la hora de desarrollar la aplicación web.

2 DISEÑO INICIAL

El diseño inicial del proyecto se basa en un modelo muy usado en la actualidad, aunque actualmente se usa más la tecnología MERN STACK pero muchos desarrolladores actualmente sigue usando la tecnología full stack basada en MEAN STACK.

Este diseño lo podemos dividir en dos partes bien diferenciadas:

1. **Parte del servidor**: En esta parte se utilizan las tecnologías de NodeJS, Express y MongoDB.

   Para realizar la parte del servidor, me he basado en una arquitectura modelo-vista-controlador.

   La parte del modelo se puede definir como las clases, que albergará la información de las diferentes entidades que tendrá el proyecto, todo aquello que va a suponer algún tipo de información para almacenar en la BBDD.

   La parte de la vista se puede denominar a lo llamado rutas, ahí haremos llamadas a los controladores para realizar las diferentes partes de un CRUD en la base de datos, ya sea un créate, read, update o delete de las entidades previamente nombradas.

   Por último, los controladores acogerán toda la lógica del backend, toda la lógica de la parte del servidor, es decir, tendrá las implementaciones del manejo de la información en el lado del servidor que después usará las rutas.

   Esta parte es fundamental para el desarrollo y diseño del proyecto, ya que es la parte que se va a encargar de hacer la comunicación con el Frontend para el intercambio de información y de datos y para una buena estructuración de la información.

2. **Parte del cliente**: En esta parte se va a utilizar el framework de Angular junto con los lenguajes de Typescrypt, Bootstrap y el lenguaje de hipermarcado HTML.

   Angular se basa en una arquitectura de componentes basado también en una arquitectura Modelo-Vista-Controlador pero, ¿Qué es esta arquitectura?

   Para crear nuestras aplicaciones con Angular, generamos templates, también llamado plantillas, con HTML y controlamos estos mismos templates con lógica creada en nuestros componentes, que serán exportados como clases. Así mismo, agregamos lógica en nuestros servicios para manejar la data que nuestra aplicación tendrá y finalmente “encapsulamos” nuestros componentes y servicios en módulos.

   Cada aplicación de Angular tiene al menos un componente. Al igual que el root module, existe el root component que conecta una jerarquía de componentes con el DOM. Cada componente define una clase que contiene data y lógica, y está vinculada a nuestro template HTML.

   Luego, iniciamos la aplicación arrancando el módulo raíz. Angular toma el control y muestra el contenido de la aplicación en el navegador y responde a las interacciones del usuario de acuerdo con las instrucciones de funcionamiento que proporcionamos.

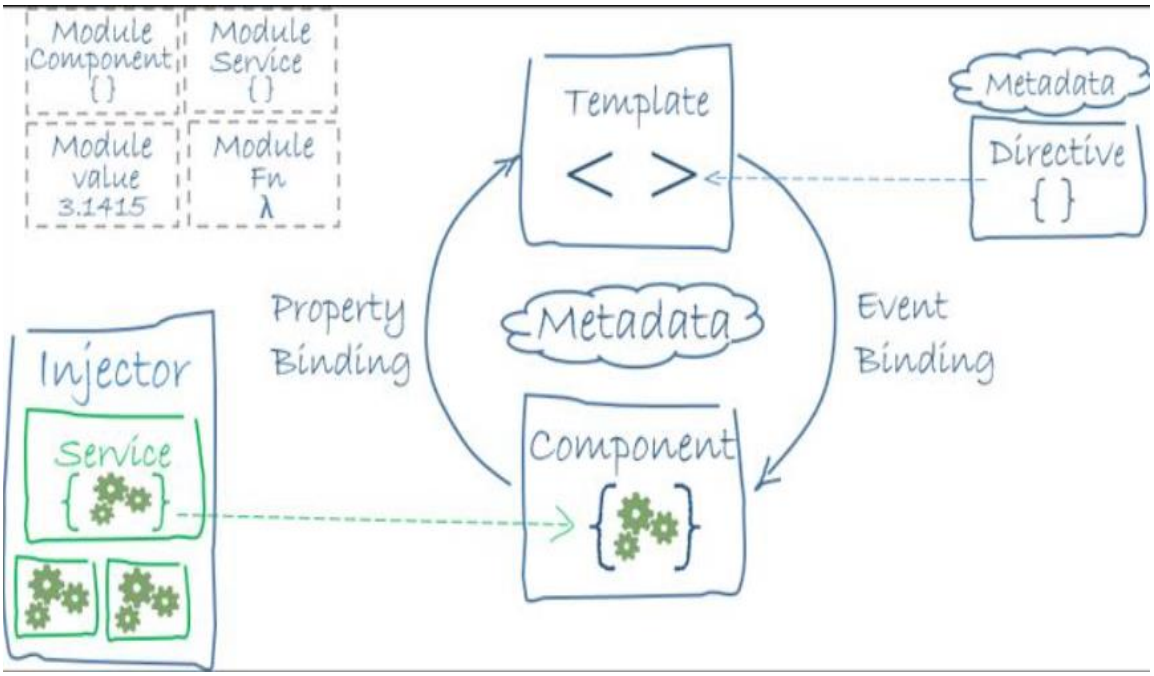


Ilustración 2.1: Funcionamiento del framework Angular

En esta ilustración se puede observar cómo se produce la comunicación entre el componente HTML que es el template y el component que es la parte lógica del programa, la que se encarga de procesar los datos.

Property Binding es usado para pasar los datos del componente al template, para que lo muestre en la interfaz, por ejemplo, si necesitamos los usuarios que se han registrado en el sistema, desde el componente lo extraemos y lo mandamos al template para que lo visualice.

Event Binding permite a una aplicación Angular ejecutar código y acciones cuando se produce un evento.

Es habitual tener un flujo donde el componente tendrá impacto en la vista, de acuerdo con su estado. Esta vez, se trata de un flujo donde la vista va a notificarle al componente una interacción.

La parte de cliente es la que se va a encargar de ponerse en contacto con el servidor, la BBDD para acoger esos datos y mostrarlos o procesarlos mediante tablas, formularios para hacer alguna de las operaciones deseadas.

Estas dos partes son las que, en conjunto, nos va a dar forma a la arquitectura completa de la aplicación web.

A continuación, se puede ver un esquema gráfico de cómo se produce esta interacción completa entre la parte del cliente y servidor.

Ilustración 2.2: Desarrollo de comunicaciones entre agentes

Primero se hace una llamada, en este caso mediante el sistema implementado por Angular de HTTPCLIENT, que simula las peticiones AJAX que siempre se ha conocido en el desarrollo web.

Tras esto, hacemos una consulta a la base de datos, ya sea como he comentado anteriormente para hacer alguna de las operaciones CRUD también conocidas en el mundo del desarrollo web.

En tercer lugar, la base de datos genera una respuesta que es procesada mediante NodeJS y tras esto, se devuelve un objeto en formato JSON con la información requerida por Angular en la primera etapa.

2.1 Especificaciones del sistema

El sistema consta de dos participantes fundamentales en la gestión y desarrollo del flujo de la aplicación: Administradores y clientes.

Cada uno tendrá un panel diferente dentro de la aplicación, los administradores tendrán un panel para administrar todos los productos, pedidos, categorías y datos que se producirán por la acción de los clientes mientras que los clientes, simplemente tendrán acceso a la aplicación del restaurante, pudiendo simplemente acceder mediante su cuenta, añadir diferentes productos a su carro y, finalmente, realizar el pedido deseado, sin poder manipular ningún producto, categoría que sale dentro de la aplicación.

A continuación, se mostrará mediante un diagrama de uso, cuáles son las funciones que realiza cada grupo dentro de la aplicación.

Todos estos casos, nos lleva a diferenciar entre los requisitos funcionales y no funcionales:

-Los requisitos funcionales son declaraciones de los servicios que prestará el sistema, en la forma en que reaccionará a determinadas acciones.

Los requisitos funcionales de esta aplicación son:

1. El usuario debe poder registrar y loguearse en el sistema, ya que esta acción le proporcionará el poder para realizar las acciones principales de la aplicación.
2. El usuario debe poder navegar entre las diferentes categorías expuestas en la aplicación.
3. El usuario debe poder añadir productos a su cesta para poder finalizar el pedido, siendo este perfil de cliente.

4. El usuario debe poder formalizar su pedido y ser procesado por el administrador.
5. La contraseña del usuario esta encriptada con SHA-512 para mayor seguridad.
6. El usuario debe poder añadir o quitar productos o categorías, siendo este el que cumple el perfil de administrador.
7. El usuario debe poder procesar los pedidos y recoger los datos, siendo este el que cumple el perfil de administrador.

-Los requisitos no funcionales son aquellos requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario), sino a las propiedades del sistema: rendimiento, seguridad, disponibilidad.

Los requisitos no funcionales de esta aplicación son:

1. El sistema debe ser capaz de aguantar una cantidad de clientes al mismo tiempo, sin que esta se vea con problemas de rendimiento.
2. El sistema debe ser de fácil aprendizaje para todos los usuarios que lo usen.
3. El sistema debe ser capaz de procesar toda la información administrada en una cantidad de tiempo pequeña, con un máximo de 3 segundos para su procesamiento.
4. Se debe proporcionar mensajes de alerta o error al usuario cuando quiera realizar una acción, pero no consiga hacerlo, ya que será intuitivo y sabrá qué ha sucedido en sus acciones.
5. Se debe tener el sistema totalmente actualizado en cuanto a las novedades y problemas que puedan surgir en la aplicación.
6. La aplicación debe ser segura, es decir, que los datos comprometidos del usuario no se vean vulnerados por algún procedimiento erróneo.
7. La aplicación debe poseer interfaces gráficas bien formadas.

8. La aplicación debe ser compatible con todas las versiones de Windows, desde Windows 7.

2.2 Análisis y diseño del sistema

Para determinar un correcto flujo del sistema, las tecnologías nos ofrecen una amplia gama de posibilidades para poder realizar un análisis, y un posterior diseño del sistema.

En cuanto al apartado de análisis, primero lo que he realizado es un estudio de los bares y restaurantes de los alrededores de Jaén, en los que se ha comprobado, que, en épocas de turismo o épocas de festivos o ferias, entre otras fiestas, se produce una gran congestión en cuánto al servicio de la comida a los comensales.

Por lo que, una vez realizado un amplio análisis del sistema, se procede a hacer una breve encuesta sobre el por qué los dueños de los restaurantes creen que ocurre este problema.

La mayoría concluye que esto es producido por la acumulación de gente y la falta de tecnologías implementadas en la hostelería para poder esclarecer este problema.

Si tuvieran un sistema en el que el usuario directamente conforme llegue pueda pedir todo lo que desea para el servicio completo, esto producirá una mayor rapidez en la toma de decisiones, un mejor servicio ya que el proceso entre el pedido y la producción de los platos no será tan retrasado en el tiempo y, en definitiva, un resultado más satisfactorio con el que el mundo de la hostelería se ve reconfortado y los clientes ven sus necesidades resueltas cuando ellos quieren.

En cuanto al diseño del sistema, la ingeniería del software nos proporciona una serie de herramientas que nos permiten obtener un resultado más claro de lo que queremos construir, pudiendo establecer todas las relaciones que queremos que haya entre los actores participantes en las diferentes interacciones.

Entre estas herramientas están:

- Un diseño arquitectónico del sistema.
- Diseño de la base de datos.
- Diagrama de clases.
- Diagrama de casos de uso.
- Diagramas de secuencia.
- Storyboards.

2.2.1 Diseño arquitectónico del sistema

El diseño de una arquitectura de software utiliza los conocimientos de programación para planear el diseño general del software de modo que puedan agregarse detalles más adelante, lo cual permite a los equipos de software delimitar el panorama general y comenzar a elaborar un prototipo.

Para el diseño arquitectónico del sistema, nos basamos en que se va a tener dos estructuras principales: Backend y Frontend.

Para el backend tendremos una serie de modelos que pertenecerán a los objetos con los que se interactúa para añadir en la base de datos.

Para el frontend tendremos la lógica de la aplicación y todo el sistema.

A continuación, en las siguientes secciones podremos profundizar más en los diversos aspectos del sistema mediante diferentes diagramas.

El sistema tiene una serie de requisitos funcionales y no funcionales descritos en anteriores apartados.

2.2.2 Diseño de la base de datos

El diseño de base de datos es un proceso fundamental a la hora de modelar nuestros conjuntos de datos y definir las operaciones que queremos realizar sobre ellos.

Los datos son el activo más importante de nuestra organización y una base de datos bien diseñada influye de forma directa en la eficiencia que obtendremos a la hora de almacenar, recuperar y analizar nuestros datos.

Un diseño de base de datos realizado de forma correcta nos proporciona unas ventajas fundamentales:

- Nos permite ahorrar espacio, mediante el diseño de base de datos optimizadas y sin datos duplicados.
- Nos ayuda a que se preserve la precisión e integridad de los datos y que no se pierda información.
- Agiliza de forma extrema el acceso y el procesamiento de los datos.

Como cada proceso, el diseño de base de datos está compuestos por distintas etapas secuenciales.

Recopilación y análisis de requisitos:

Esta primera fase consiste en un paso previo obligatorio, para asegurarnos de que nuestra base de datos cumplirá con nuestros objetivos. Para ello, deberemos analizar distintos factores, entre los cuales:

Los datos que necesitamos almacenar y de dónde provienen, la información que los datos describen, los usuarios de la base de datos y sus necesidades a la hora de acceder a los datos.

Diseño conceptual

En esta fase se representan una descripción a alto nivel del contenido de la base de datos, independientemente del sistema de gestión de base de datos que se utilizará a continuación. Se definen en un dibujo las entidades, sus atributos y las relaciones entre ellas.

Elección de un sistema de gestión de base de datos

Es en esta fase donde elegiremos el sistema de gestión de bases de datos (SGBD) concreto que mejor se adapta a nuestro proyecto, como, por ejemplo, Oracle, MySQL, Microsoft SQL Server y PostgreSQL, MongoDB.

Diseño lógico

En esta fase, se traduce el modelo conceptual obtenido anteriormente a un esquema lógico, que describe la estructura de la base de datos. Se trata de la fase en la cual se diseñan las tablas propiamente dichas, con sus filas, columnas y relaciones. El modelo lógico depende del SGBD que se utilizará.

Diseño físico

En esta fase se definen las estructuras de almacenamiento de la base de datos de forma física. Es cuando se escribe el código (por ejemplo, SQL) para concretar el diseño en el motor de base de datos que hemos escogido.

Implementación

Finalmente, se crea y se compila el esquema de la base de datos, se generan los ficheros y las aplicaciones que implementan las transacciones.

El diseño de la base de datos del proyecto viene dado por el siguiente diagrama:

Se ha realizado el diseño de la base de datos mediante la herramienta **drawSQL**.

La herramienta proporciona una interfaz intuitiva y de fácil uso para realizar diagramas sencillos, para proyectos pequeños medianos.

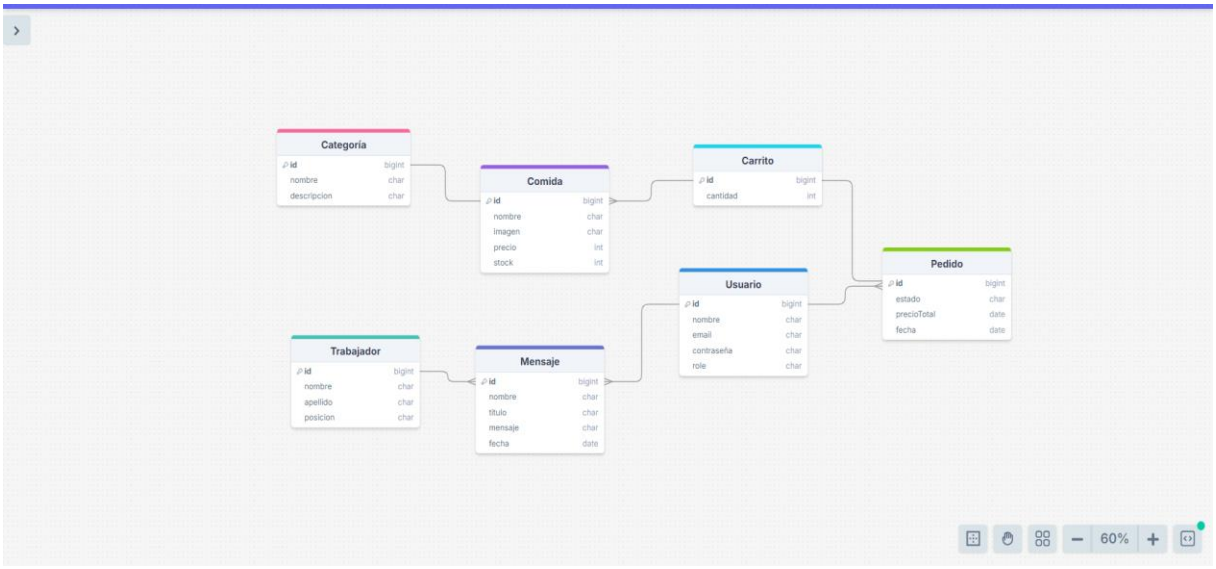


Ilustración 2.3: Diagrama de base de datos

2.2.3 Diagramas de casos de uso

En el diagrama de casos de uso, las funciones del sistema en cuestión se representan desde el punto de vista del usuario (llamado “actor” en UML). Este actor no tiene que ser necesariamente un usuario humano, sino que el rol también puede atribuirse a un sistema externo que accede a otro sistema.

De este modo, el diagrama de casos de uso muestra la relación entre un actor y sus requisitos o expectativas del sistema, sin representar las acciones que tienen lugar o ponerlas en un orden lógico.

En la práctica, esta estructura es adecuada para representar claramente las funciones u objetivos más importantes de un sistema. Por esta razón, a la hora de desarrollar un software o planificar nuevos procesos empresariales, crear un diagrama de casos de uso suele ser uno de los primeros pasos, ya que permite visualizar clara y fácilmente qué casos de uso deben tenerse en cuenta durante el desarrollo para que los actores del sistema logren identificar los pasos en su flujo en el sistema.

Estos tipos de diagramas también pueden ser más complejos y describir muchos tipos de funciones que podrían o no ocurrir siempre en el curso de la relación de un individuo con un sistema.

En el sistema tenemos dos actores principales:

- **Administradores**: Los administradores tendrán su propia función dentro del sistema, que será la de analizar y gestionar los productos, pedidos, categorías, al igual que analizar los pedidos que se han realizado.

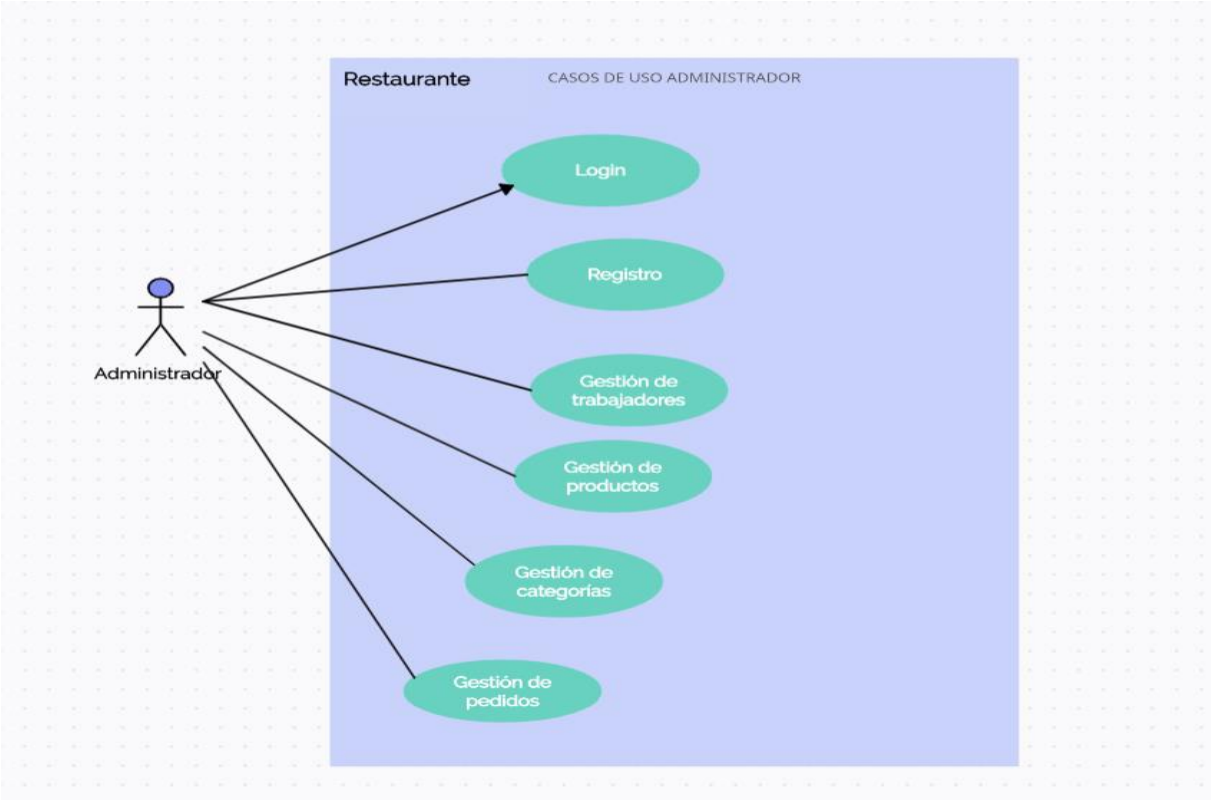


Ilustración 2.4: Diagrama de casos de uso administrador

Ahora voy a crear diagramas de casos de uso específico para cada uso.

Gestión de categorías.

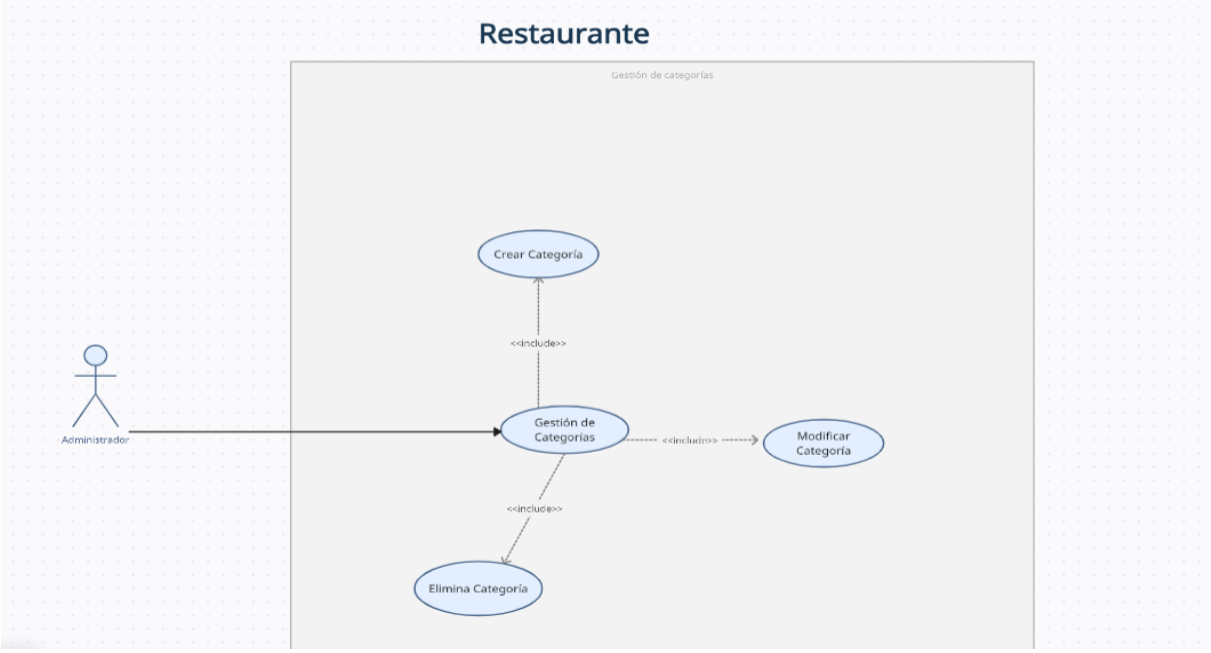


Ilustración 2.5: Diagrama de casos de uso administrador categorías

Gestión de platos:

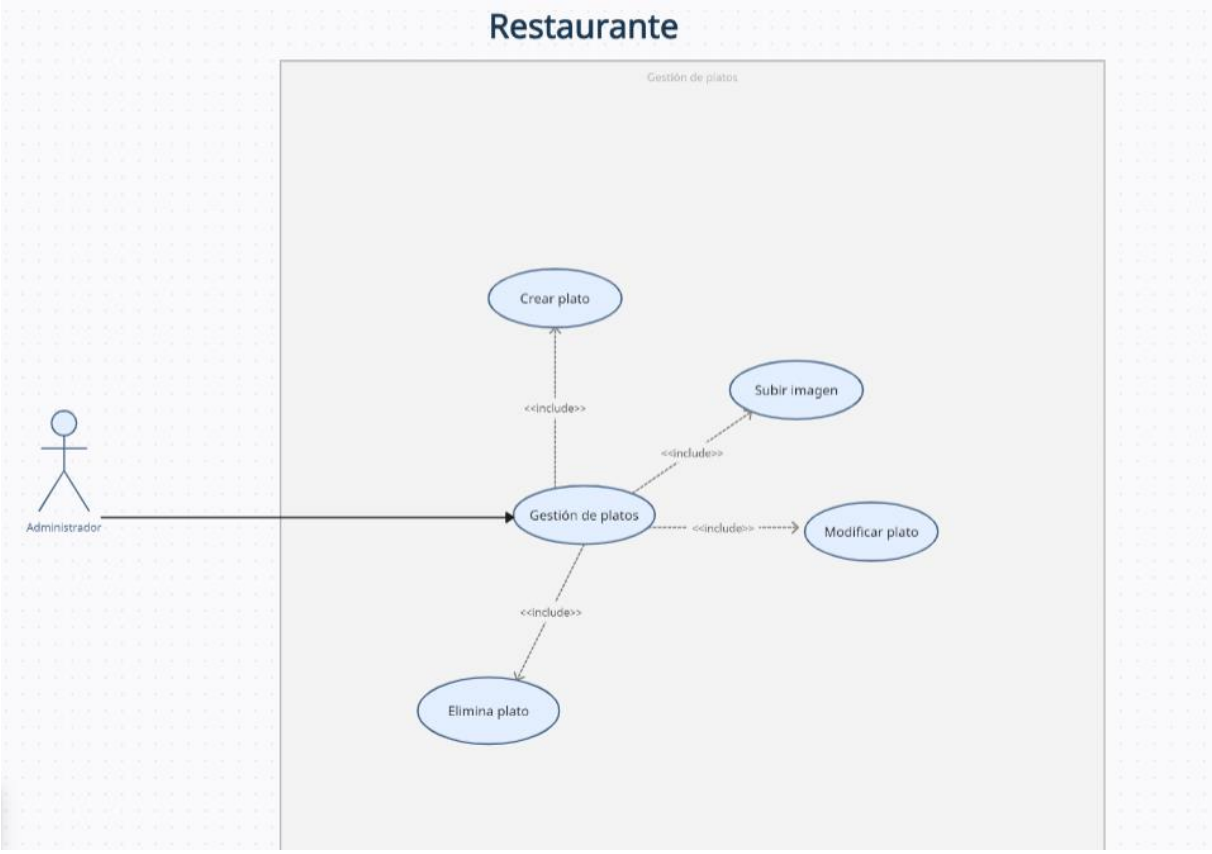


Ilustración 2.6: Diagrama de casos de uso administrador platos

Gestión de trabajadores:

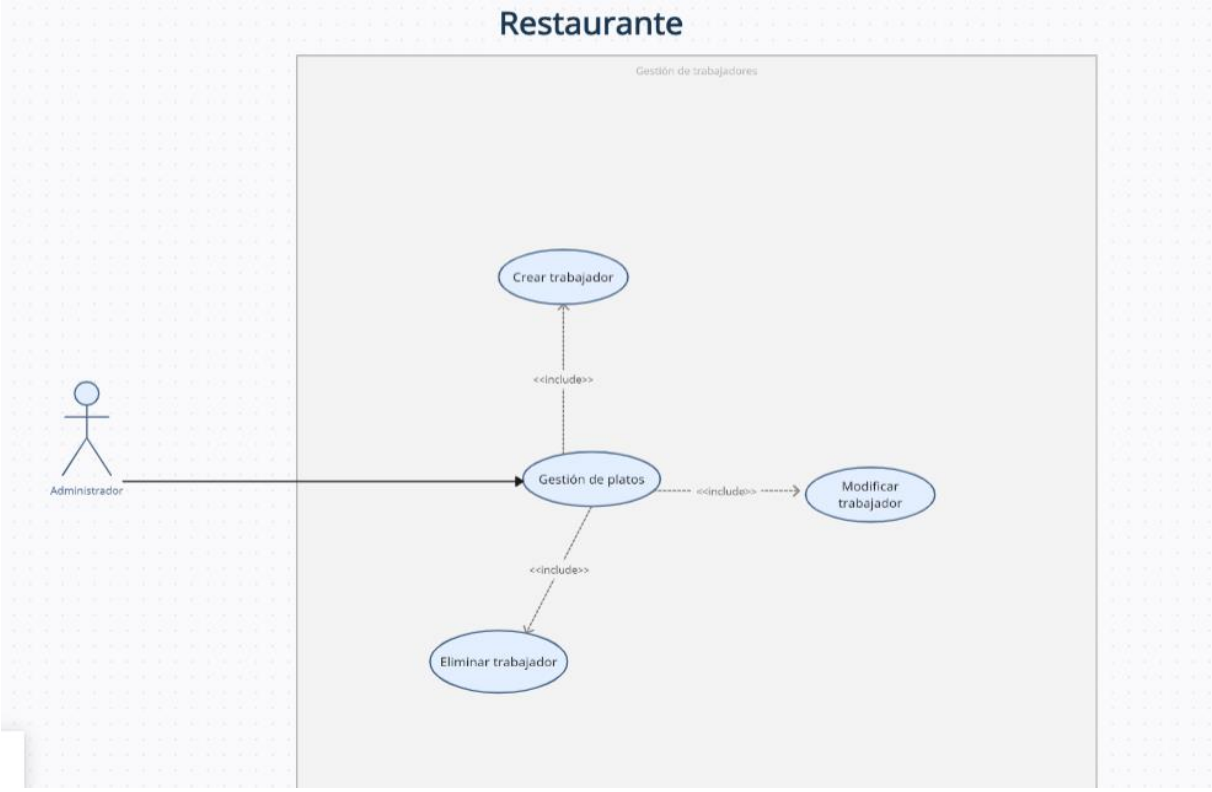


Ilustración 2.7: Diagrama de casos de uso administrador platos

Gestión de pedidos:

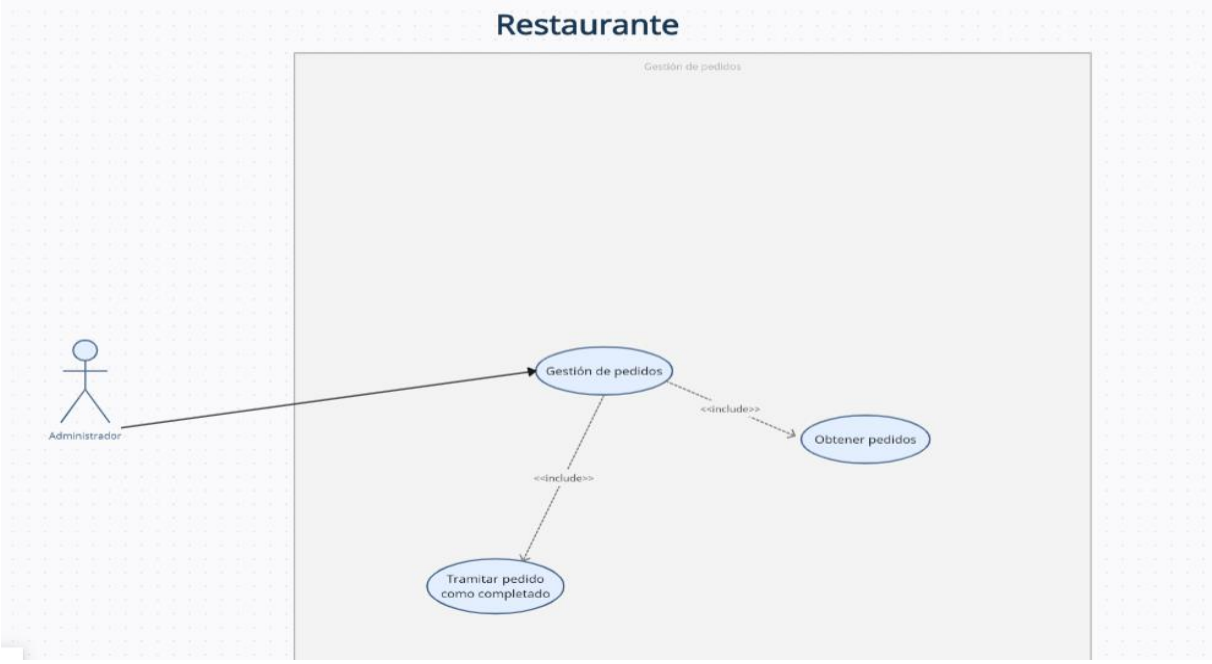


Ilustración 2.8: Diagrama de casos de uso administrador pedidos

Gestión de mensajes:

La parte de gestión de mensajes simplemente tenemos la opción de revisar los mensajes y eliminarlos conforme hayamos leído o procesado la información que nos van dando.

En un futuro se puede añadir una funcionalidad más concreta como puede ser un foro o chat en vivo para interacción clientes y administrador.

- **Clientes**: Los clientes tienen la función de mirar entre las diferentes categorías, seleccionar los platos que le gustan y realizar su pedido para que se lo sirvan en su mesa.

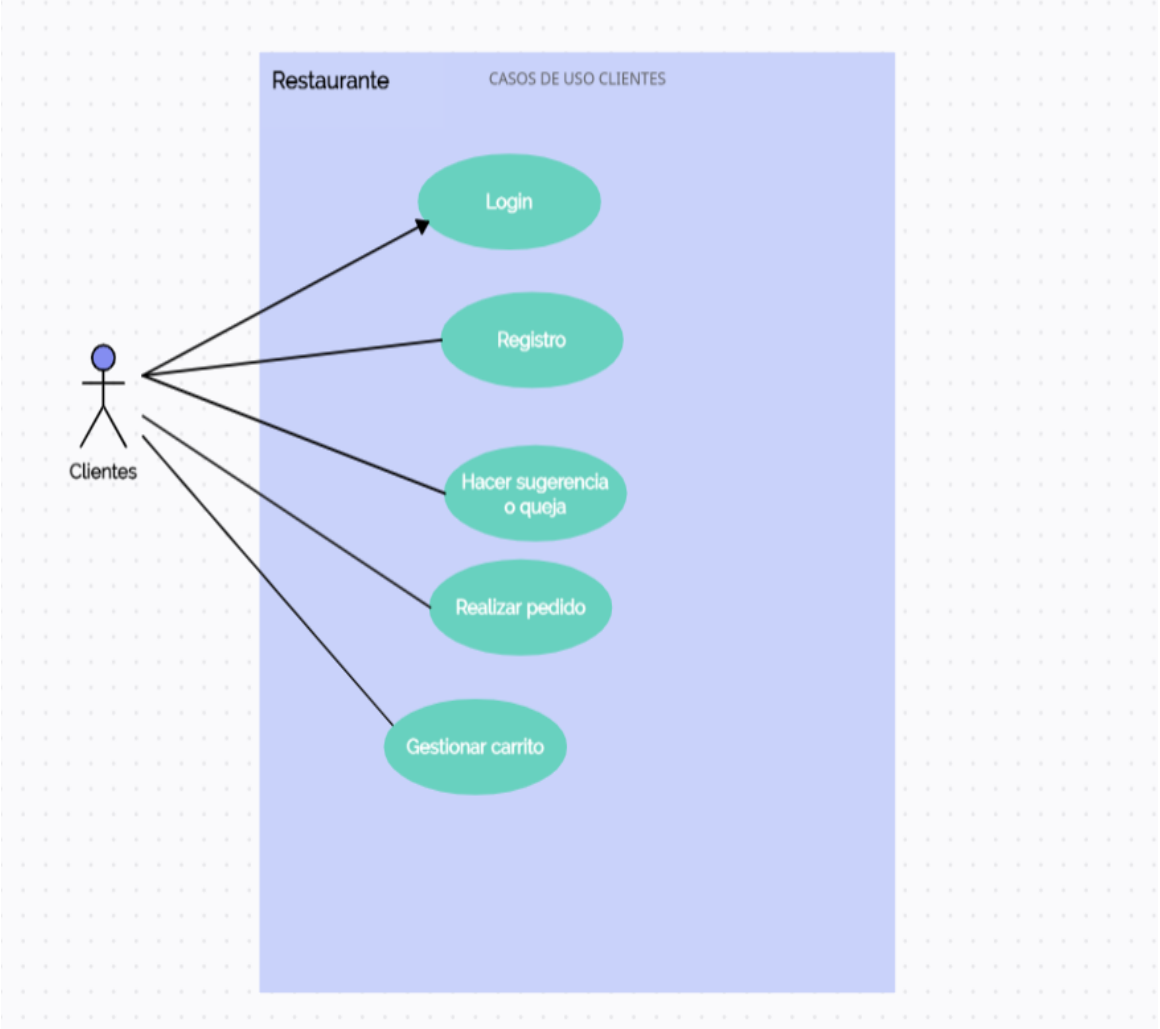


Ilustración 2.9: Diagrama de casos de uso clientes

En detalle de cada uno de las entidades, para el usuario tenemos los siguientes casos de uso:

Navegación categorías:

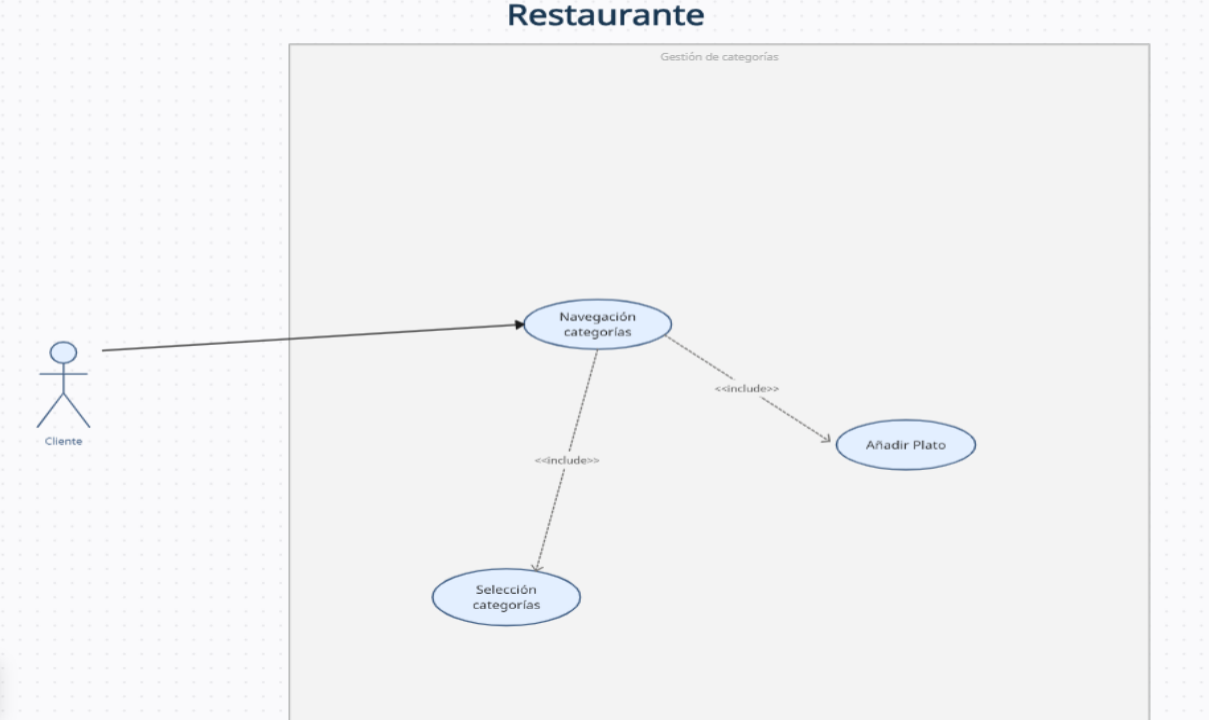


Ilustración 2.10: Diagrama de casos de uso clientes categorías

Gestión carrito

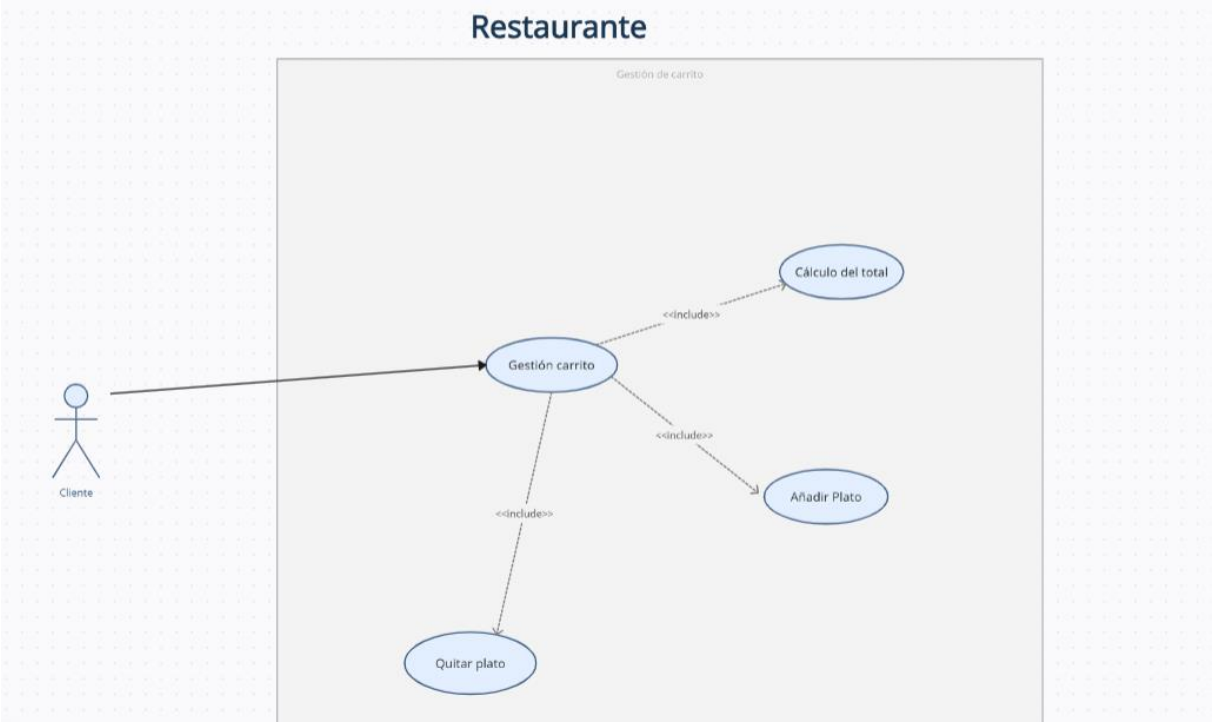


Ilustración 2.11: Diagrama de casos de uso clientes carrito

Realizar pedido: Para realizar el pedido simplemente hay un uso que es crear el pedido procesando el carrito y tramitando la ejecución del pedido.

Contactar mediante mensaje: Para el crear un mensaje de queja o sugerencia para el administrador sería un único caso de uso, ya que sería el administrador el que luego se ocuparía de procesarlo y eliminarlo o aceptar sugerencia.

El diagrama genérico del funcionamiento de las dos clases de entidades relacionadas con el proyecto se puede dar por el siguiente diagrama:

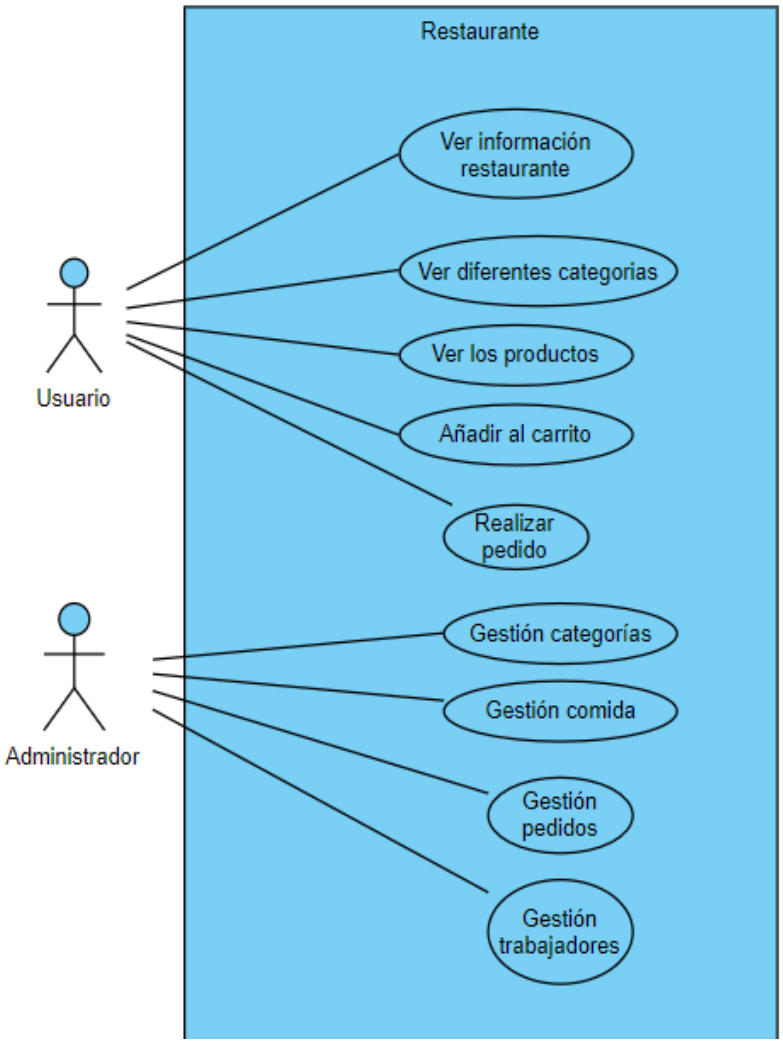


Ilustración 2.12: Diagrama de genérico

El papel del administrador es el de simplemente gestionar todo lo que ocurre en la aplicación, como son los diferentes platos que se sirven en el restaurante, los pedidos que se van realizando, la gestión de los trabajadores.

El papel del usuario es el de navegar entre las diferentes pestañas de la aplicación y realizar el pedido deseado.

2.2.4 Diagramas de secuencia

Los diagramas de secuencia, comúnmente utilizados por los desarrolladores, modelan las interacciones entre los objetos en un solo caso de uso. Ilustran la forma en que las diferentes partes de un sistema interactúan entre sí para llevar a cabo una función, y el orden en que se producen las interacciones cuando se ejecuta un caso de uso concreto.

En palabras más sencillas, un diagrama de secuencia muestra diferentes partes de un sistema trabajando en una “secuencia” para conseguir algo.

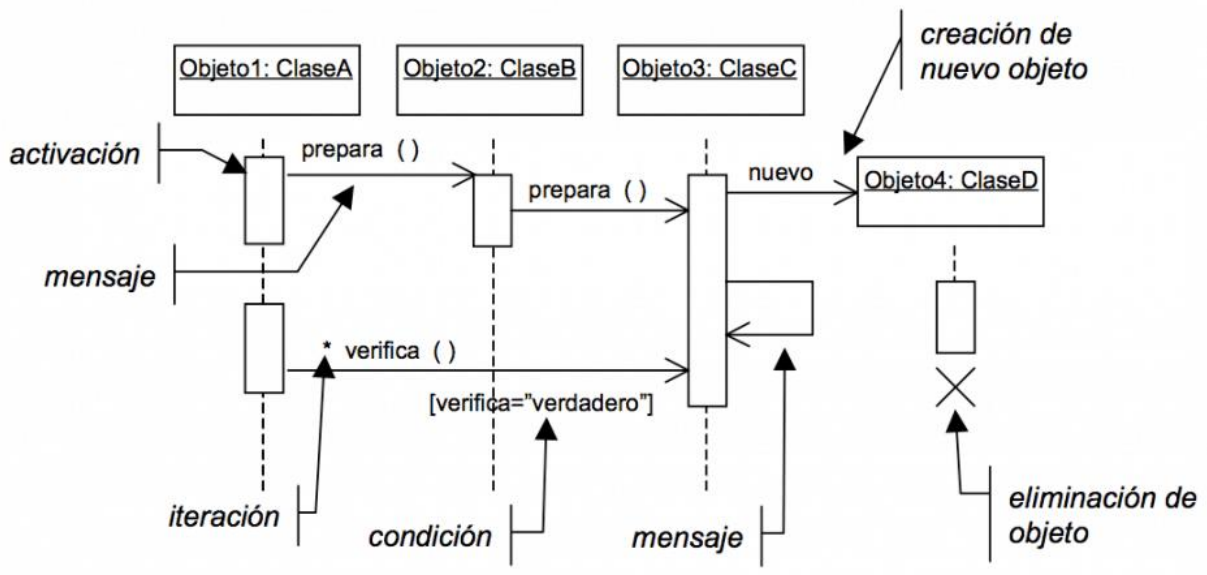


Ilustración 2.13: Ejemplo de diagrama de secuencia

Por lo que, para representar el diagrama de secuencia de esta aplicación, tomaremos al usuario con un rol neutro, es decir, ni se va a determinar como un cliente normal o como el administrador del establecimiento.

Flujo normal del sistema desde que el usuario entra en el sistema hasta que ya realiza su función u objetivo.

- **Primera introducción del usuario al sitio web:**

El usuario accede a la página, encuentra en ella un menú con diferentes apartados, una de las opciones es saber más cosas del sitio, puede pinchar sobre él para obtener más información, representado mediante el siguiente diagrama de secuencia.

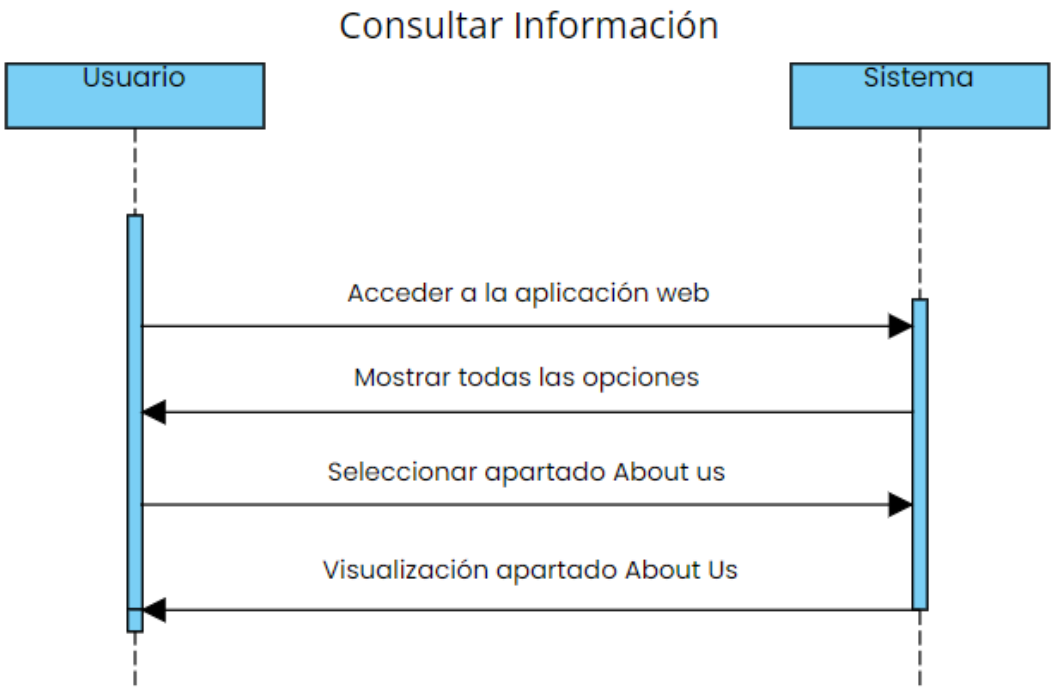


Ilustración 2.14: Consultar infomración

- **Consultar las diferentes categorías y productos de la aplicación web:**

El usuario accede al apartado de platos donde podrá visualizar todas las categorías pertenecientes a la aplicación, al igual que los productos proporcionados para cada categoría.

Puede navegar encontrando los diferentes productos de las diferentes categorías que se van presentando en la aplicación.

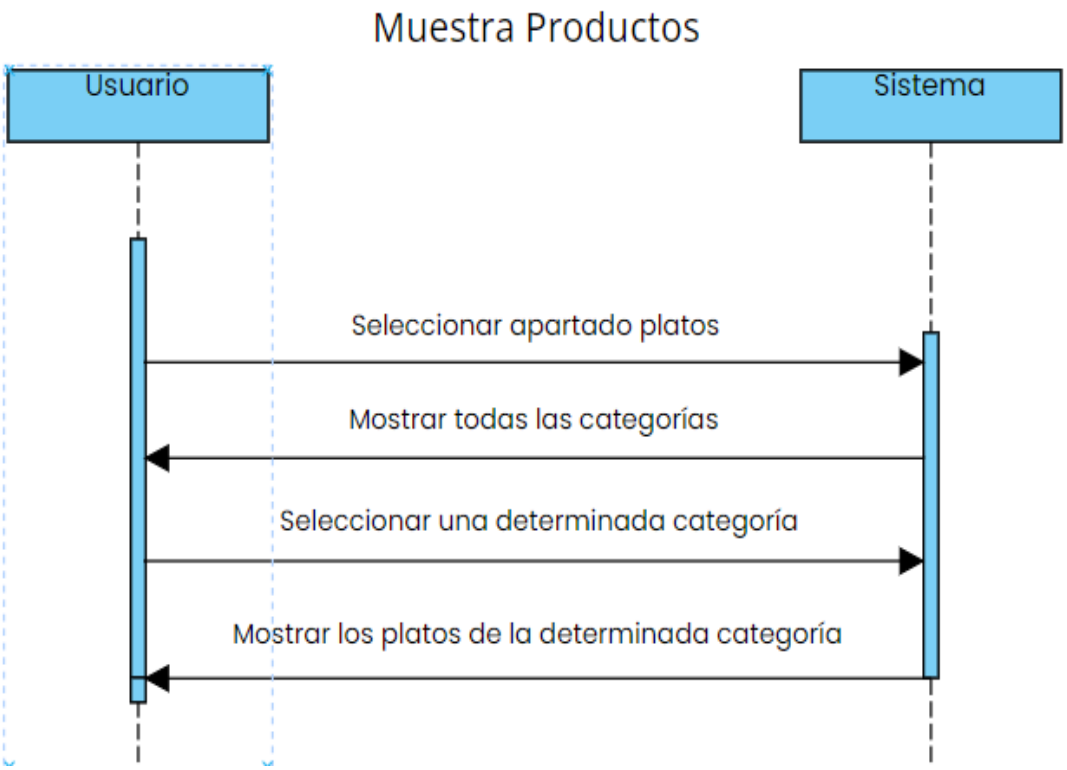


Ilustración 2.15: Muestra de los diferentes platos

- **Añadir producto al carrito:**

El usuario accede a los productos y va añadiendo los diferentes platos a su carrito de compra.

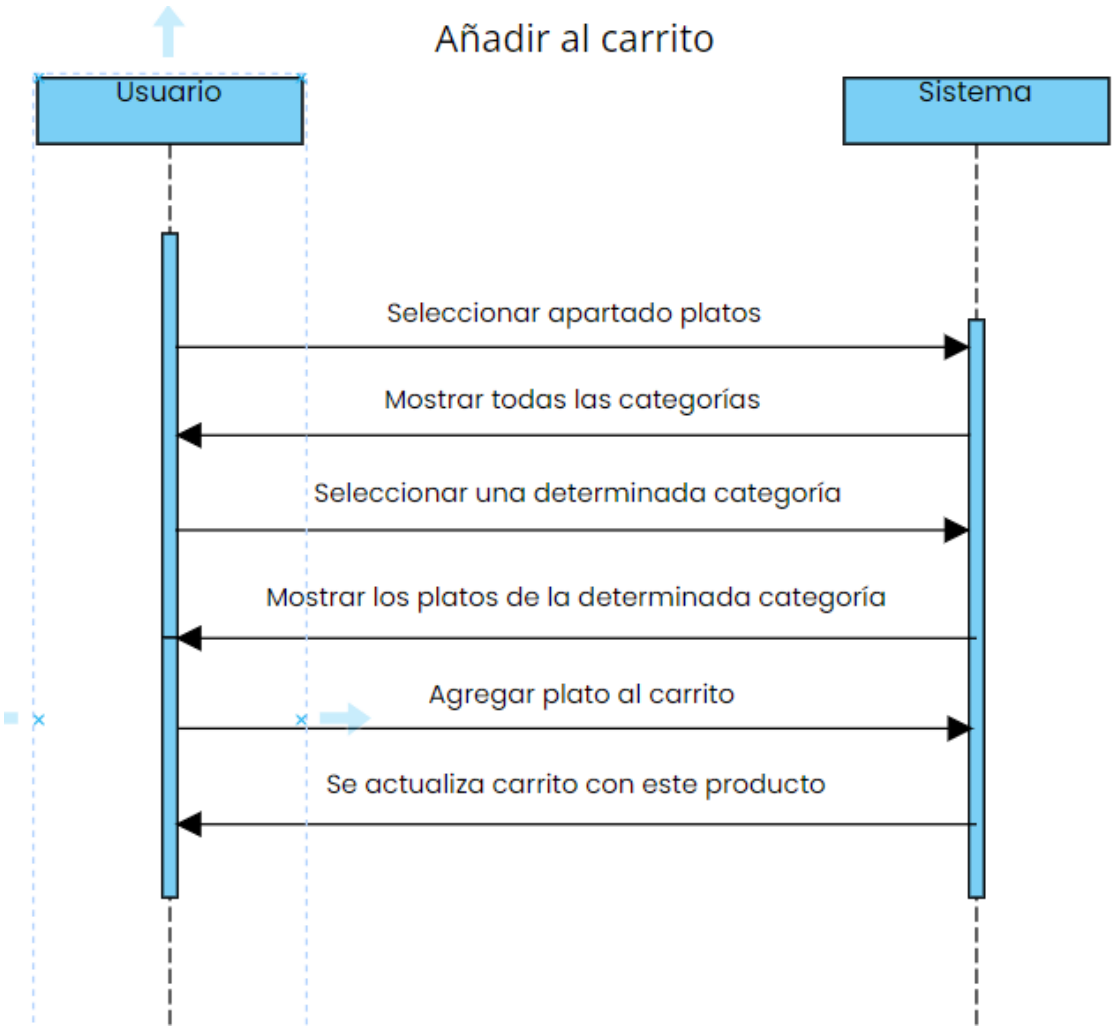


Ilustración 2.16: Añadir platos al carrito

- **Realizar el pedido:**

Una vez añadido todos los productos en el carrito, el usuario puede proceder a hacer el checkout de su carrito y procesar su pedido.

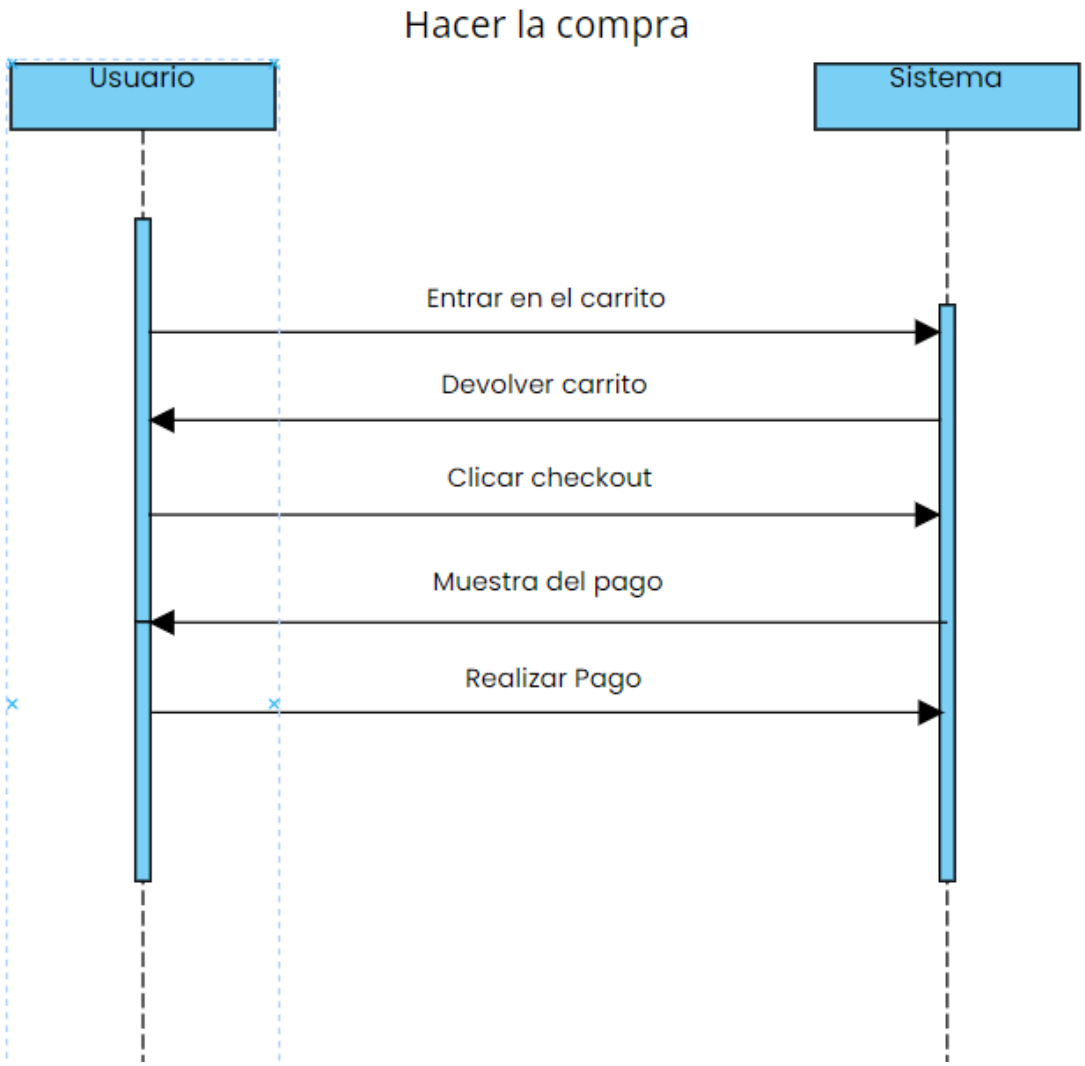


Ilustración 2.17: Realizar la confirmación del pedido

2.2.5 Storyboards y diseño de la interfaz

El storyboard es un concepto que puede aplicarse a distintos usos. Se utiliza, por ejemplo, en cine.

Con esta herramienta se representan, en forma de esquema, cómo se va a desarrollar una escena.

En el caso del desarrollo de un software, es una fase muy importante ya que podemos plasmar mediante una serie de viñetas las diferentes vistas que va a tener nuestra página y como se puede ir disponiendo, además se puede apreciar los diferentes cambios de pestaña que podemos realizar al presionar un botón o al presionar un determinado link.

Es una fase importante con la que podemos enseñar al cliente, en este caso el dueño del establecimiento, cómo va a ser, más o menos, la interfaz de la aplicación que va a usar.

El concepto de interfaz es muy amplio y se refiere a todo sistema que permite el contacto y la funcionalidad entre dos sistemas diferentes.

Por ejemplo, los botones y la pantalla del teléfono celular conforman la interfaz, ya que permite que el usuario pueda emplear las funciones que este aparato ofrece.

En definitiva, la interfaz web es el conjunto gráfico que permite la presentación y la navegación del sitio.

Esto se consigue con la inclusión de elementos comunes a toda la web que son estándares, haciendo que los usuarios tengan completo control sobre las funcionalidades del sitio desde el momento mismo de entrar a él sin que para ello deba tener amplios conocimientos ni preparación anterior alguna.

Una página web puede contar con los mejores contenidos en el género que se desarrolla, pero indefectiblemente fracasará si su interfaz no permite un rápido y cómodo acceso a los mismos por parte de los usuarios.

Por el contrario, una página web cuyos contenidos sean de menor calidad (sin que éstos sean malos, por supuesto) pero cuya interfaz permite que sus usuarios naveguen en forma sencilla, tengan acceso en forma inmediata al contenido que desean e interactúen de forma fluida, tendrá un mayor éxito.

Para la realización de storyboard voy a hacer uso de la herramienta Balsamiq.

Primera pantalla – Registro en el sistema.

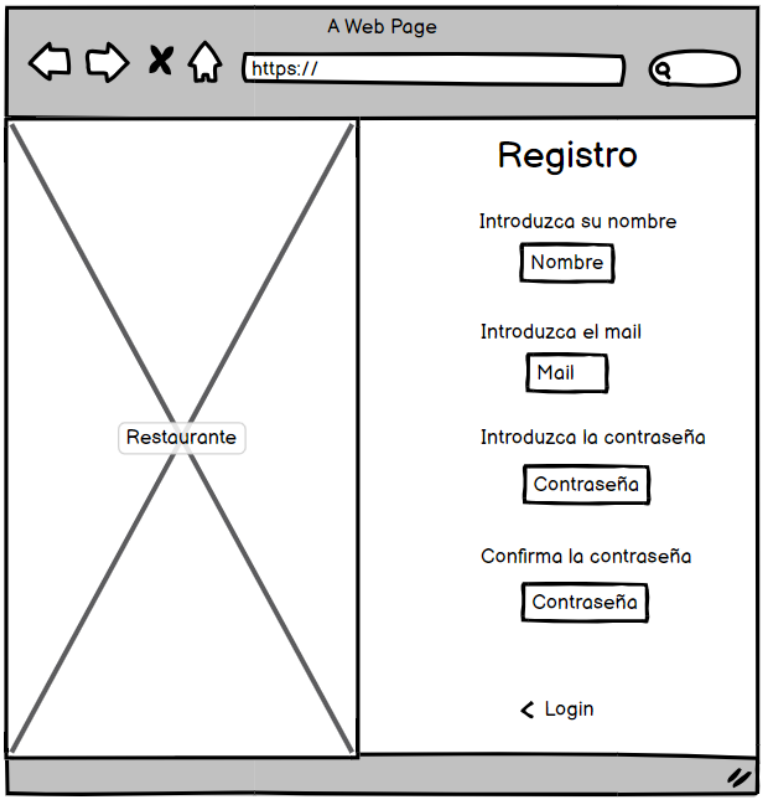


Ilustración 2.18: Storyboard 1: Registro en el sistema

Seguna pantalla – Login en el sistema.

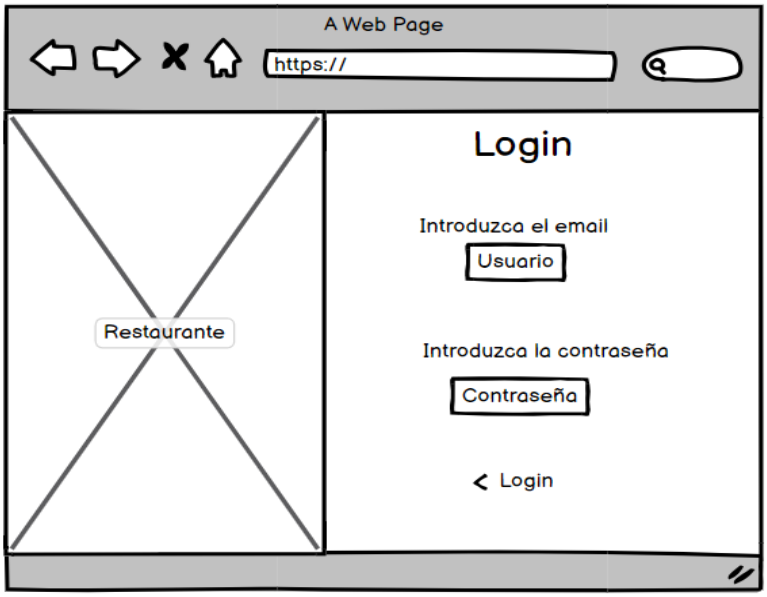


Ilustración 2.19: Realizar el login del sistema

Tercera pantalla – Interfaz cliente (Si es role de user)

Ilustración 2.20: Interfaz de cliente

Cuarta Pantalla – Interfaz del cliente (About US)

Ilustración 2.21: Interfaz de About US

Quinta pantalla – Interfaz del cliente.

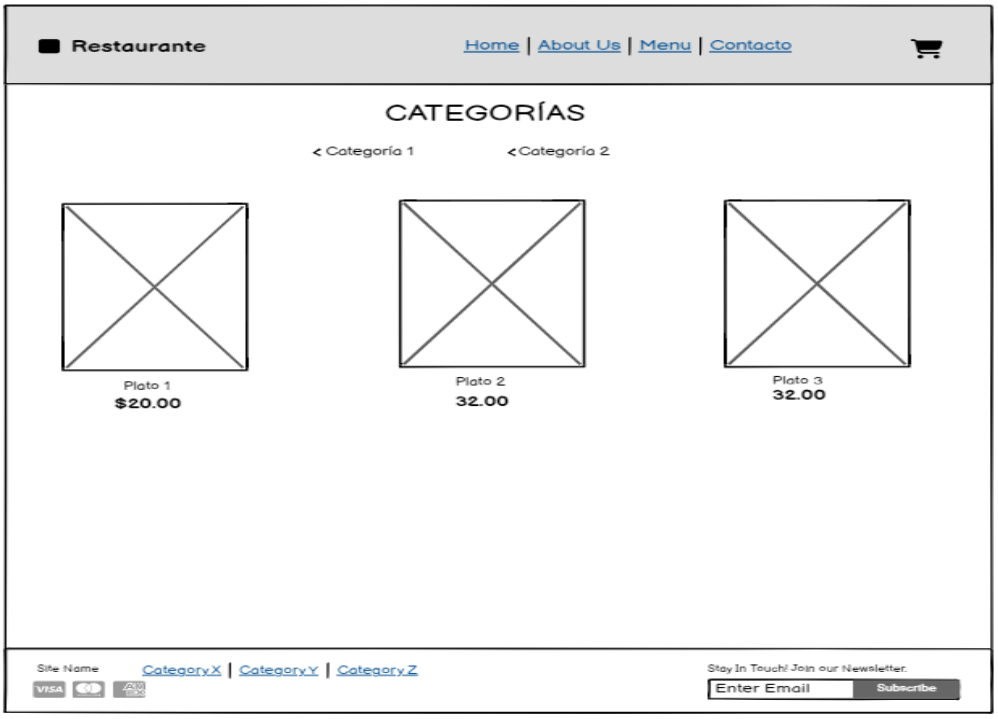


Ilustración 2.22 : Navegación categorías

Sexta pantalla – Interfaz del cliente.

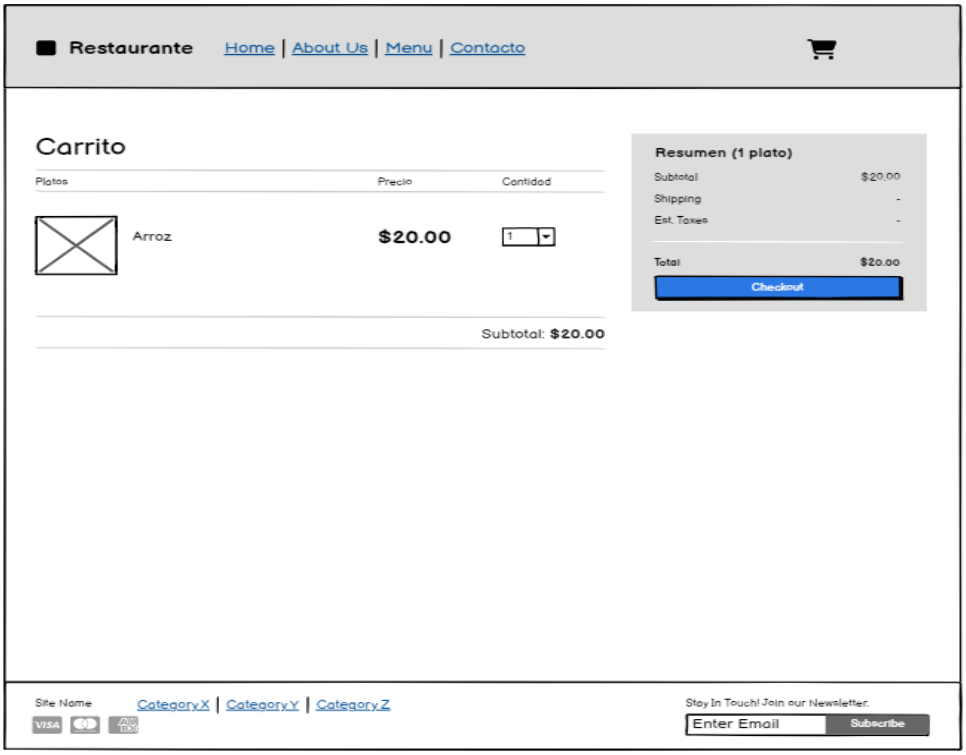


Ilustración 2.23: Realizar la confirmación del pedido

Séptima pantalla – Interfaz del cliente.

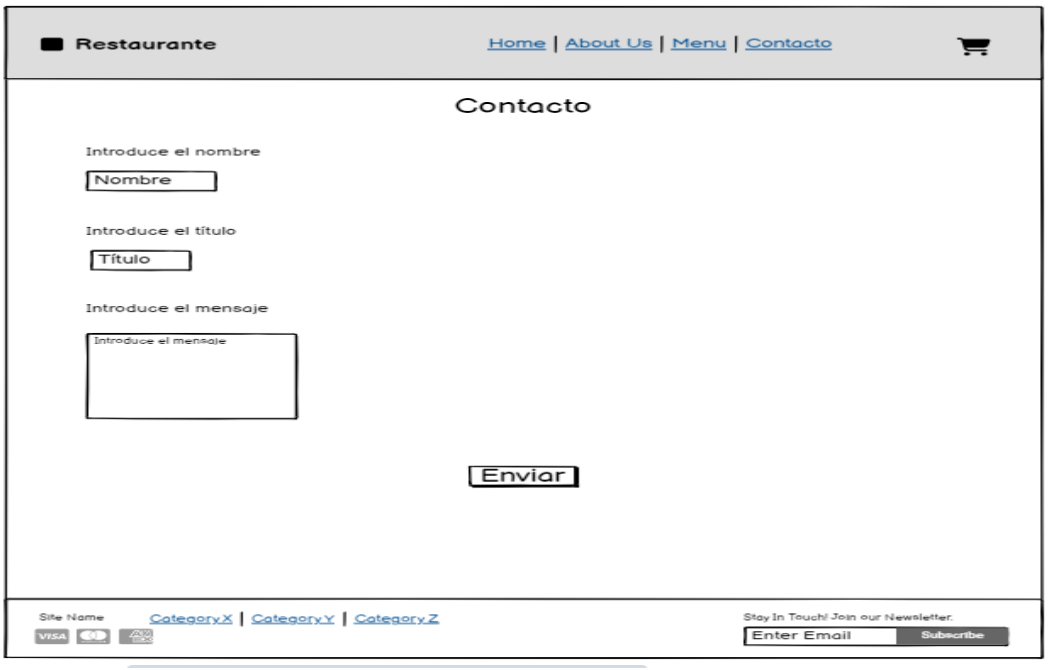


Ilustración 2.24: Realizar el contacto con el admin

Octava pantalla – Interfaz panel admin (Dashboard)

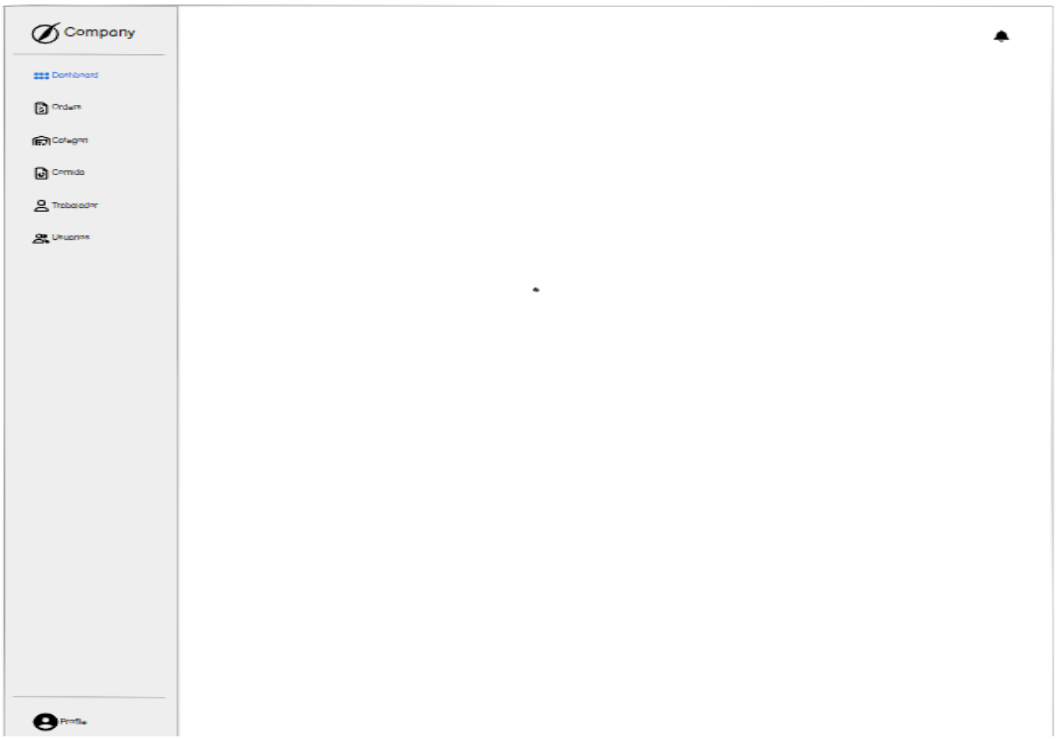


Ilustración 2.25: Dashboard admin

Novena pantalla – Interfaz panel admin

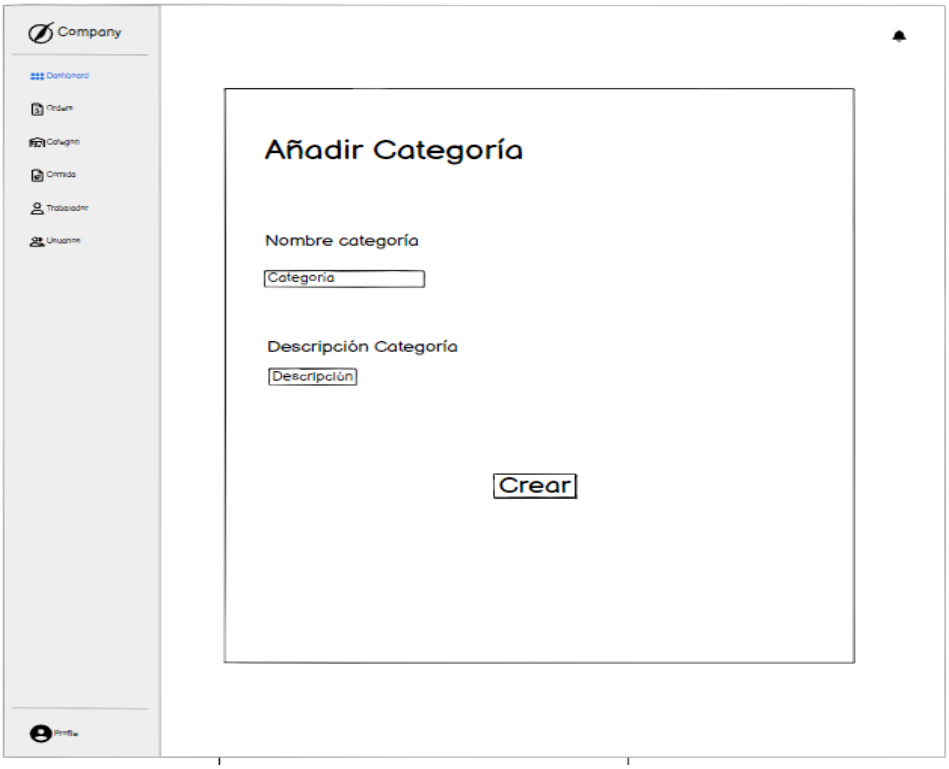


Ilustración 2.26: Añadir categoría

Décima pantalla: Interfaz de Admin (Listar Categoría)

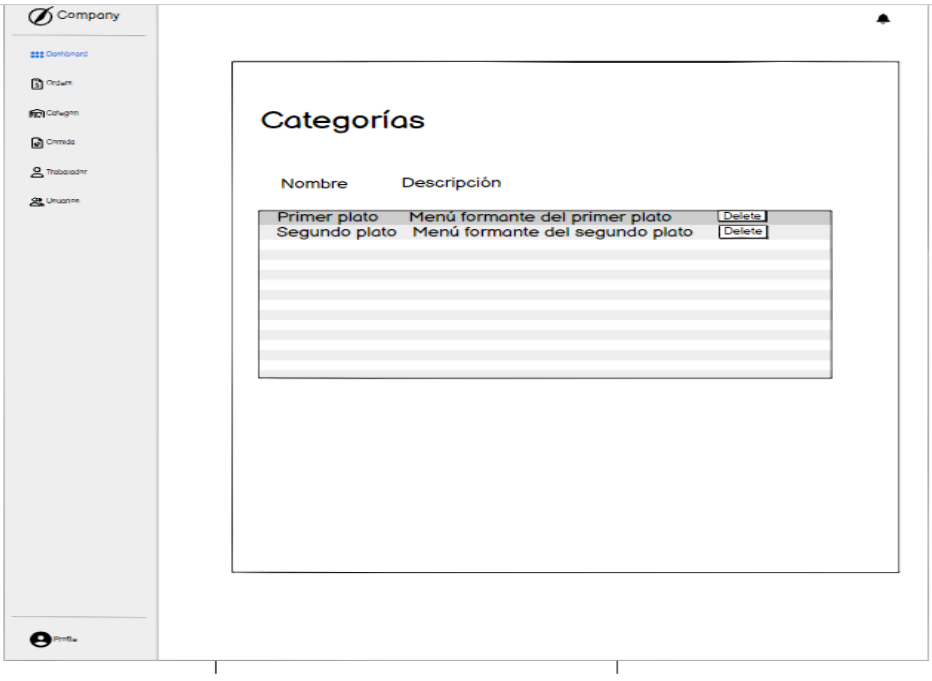


Ilustración 2.27: Listar categoría

Undécima pantalla: Interfaz de Admin (Añadir Trabajador)

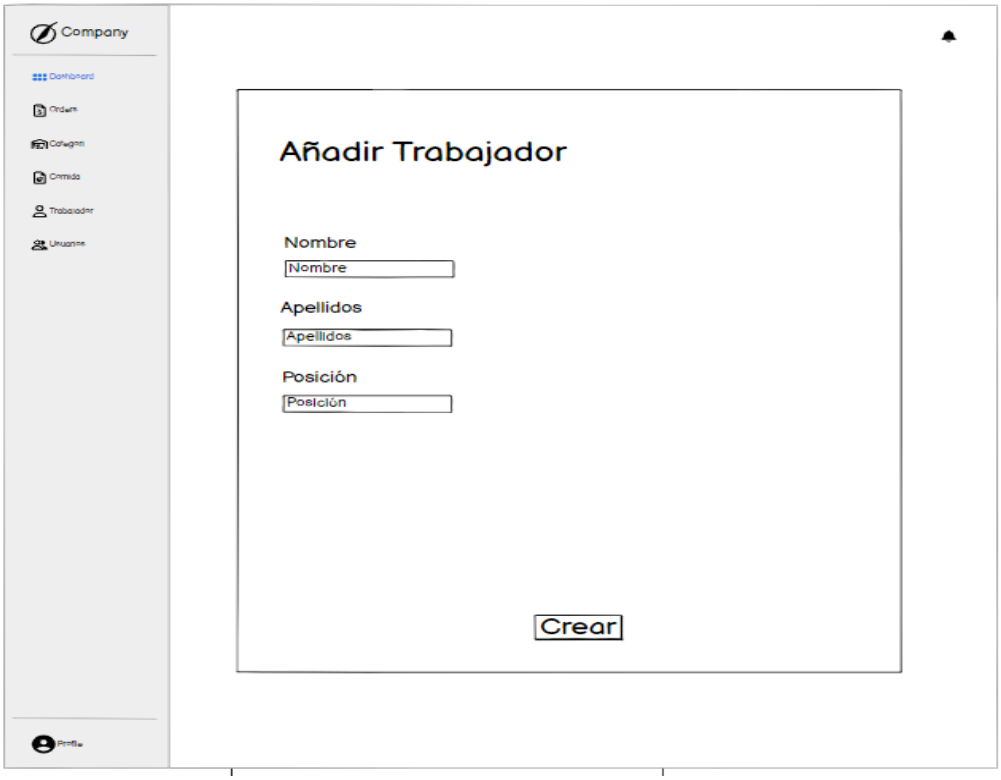


Ilustración 2.28: Añadir trabajador

Duodécima pantalla – Interfaz del Admin (Listar Trabajador)

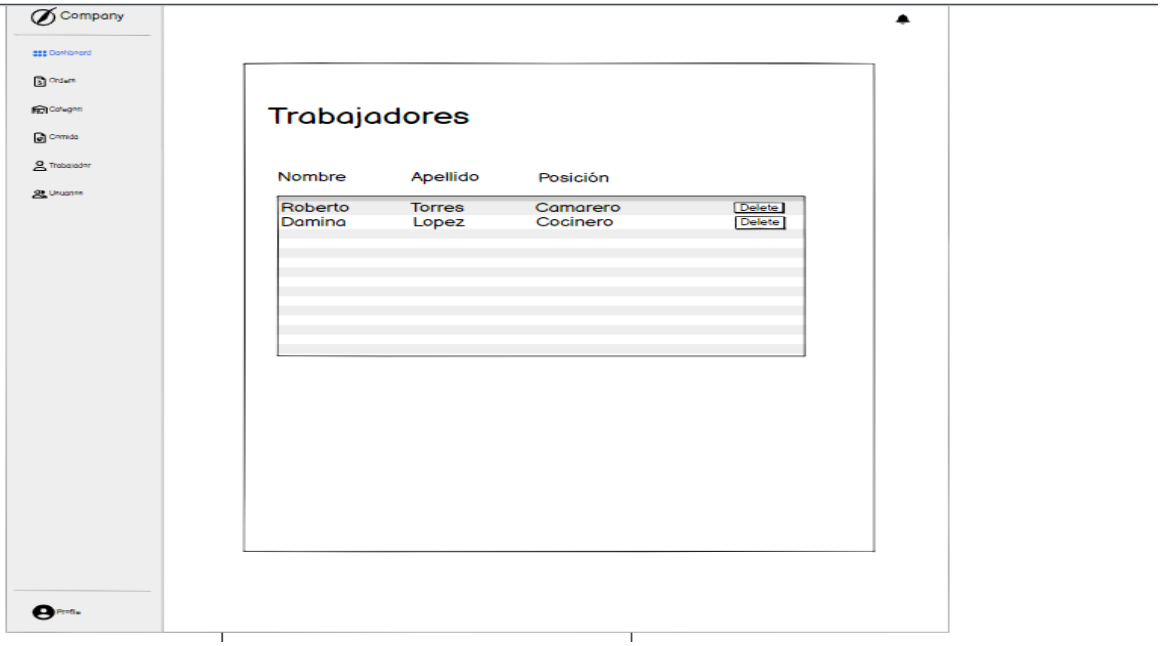


Ilustración 2.29: Listar trabajador

Decimotercera pantalla: Interfaz de Admin (Añadir Plato)

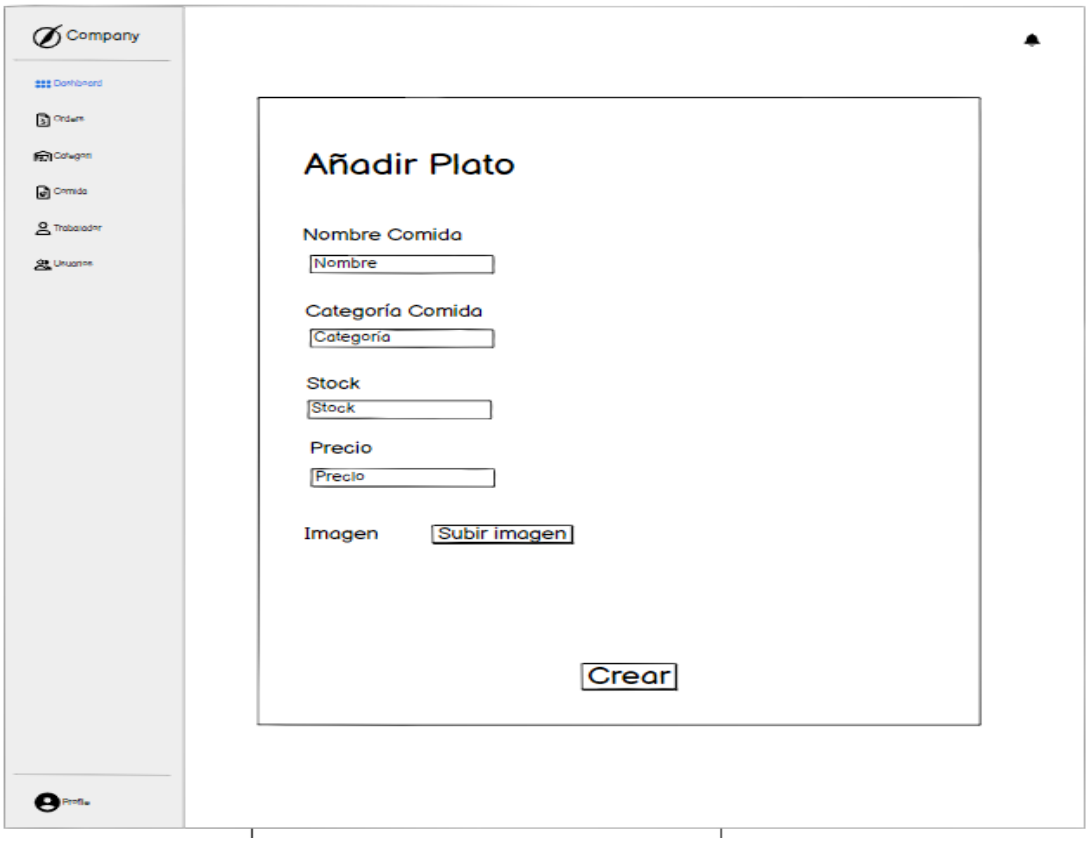


Ilustración 2.30: Añadir plato

Décimocuarta pantalla: Interfaz del Admin (Listar platos)

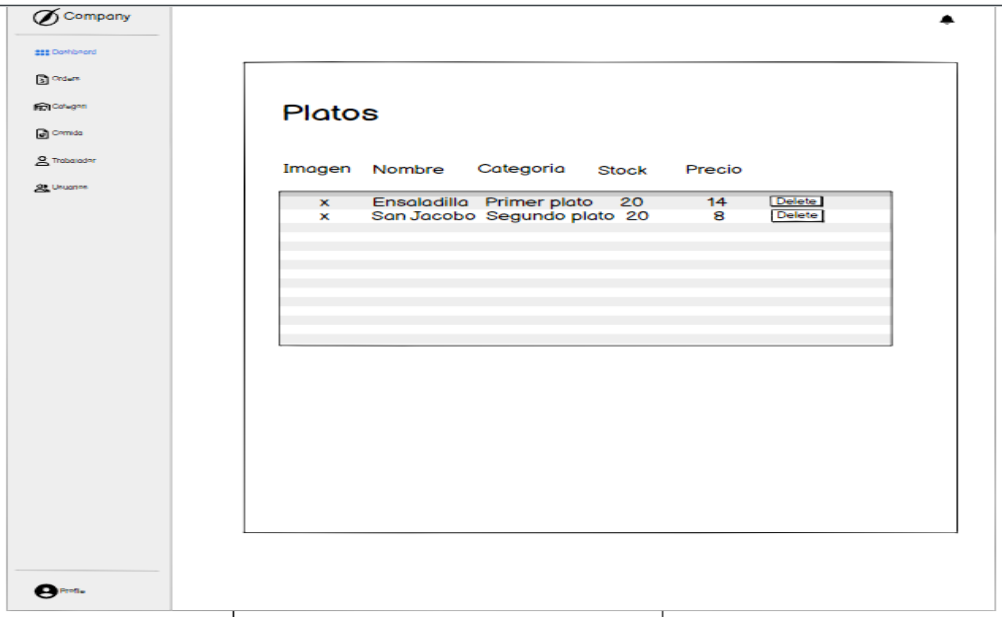


Ilustración 2.31: Listar plato

Décimaquinta pantalla: Interfaz del Admin (Listar mensajes)

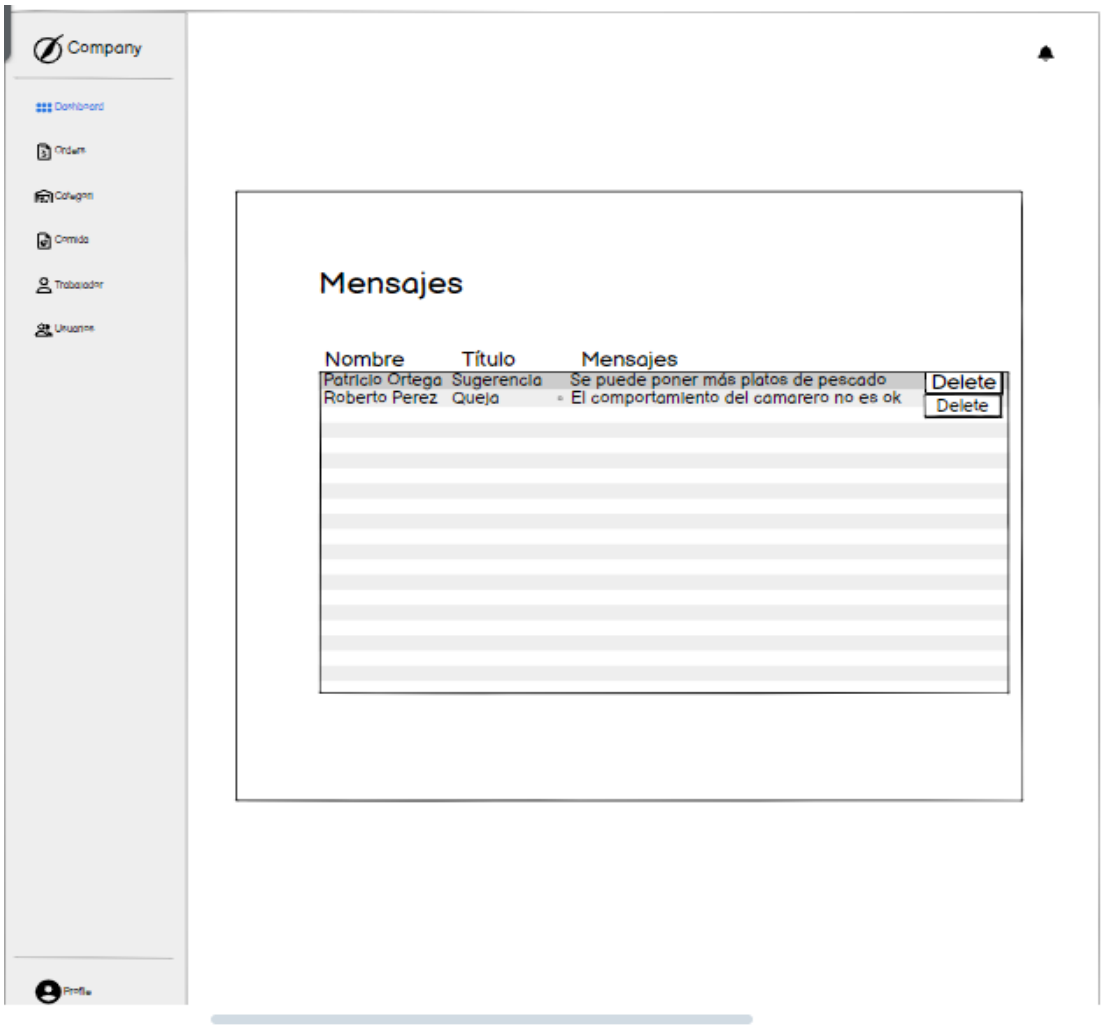


Ilustración 2.32: Mensajes

Décimosexta pantalla: Interfaz del Admin (Gestión de pedidos)

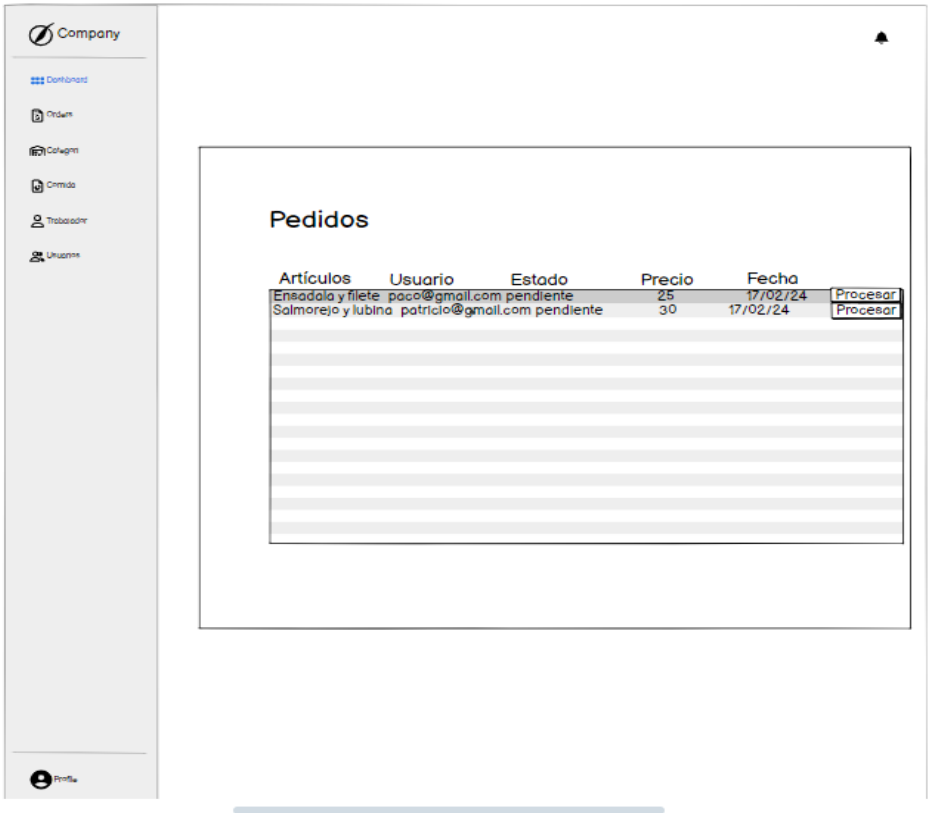


Ilustración 2.33: Gestión de pedidos

Para hacer los storyboards se han planteado alternativas como Canva, MarvelApp pero Balsamiq la recomiendo debido a su facilidad de uso, su gran cantidad de plantillas que posee con una gran personalización para añadir o eliminar navbar, sidebar, entre otros elementos de una aplicación web.

3 DESARROLLO

En esta parte se va a hacer un viaje por lo que ha sido todo el proyecto en cuanto a las iteraciones que he ido desarrollando y cómo lo he ido planificando.

3.1 Primera Iteración

El proyecto, al seguir un proceso iterativo incremental, se ha seguido una serie de iteraciones, una serie de pasos que han ido formando en pequeñas porciones, el proyecto completo.

En la primera iteración se produjo un acercamiento con diferentes empresas hosteleras que forman parte del lugar donde resido.

Estas empresas fueron proporcionando información sobre los problemas que sufren el sector hostelero en esta situación tan complicada que vivimos actualmente y el gran atraso que sufren en épocas dónde se producen grandes masificaciones de personas y hay que proporcionar el mejor servicio posible, en el cuál, en muchas ocasiones no es posible proporcionar.

Tras esta primera toma de contacto con las personas, me surge la idea de crear un sistema que les sirva a los hosteleros más cercanos a intentar corregir estos problemas que surgen y que tanto clientes como hosteleros terminen esta relación contentos.

Estos hosteleros, al explicarles cuál era la idea a implementar, veían que era una buena iniciativa que les podría suponer un apoyo fundamental para el poder mejorar los servicios y poder incluir nuevas iniciativas que hicieran atractivo el visitar los establecimientos.

La primera iteración siguió con un primer análisis sobre las diferentes funcionalidades que puede tener el proyecto, al igual que las posibles características

que se podría implementar para conseguir el objetivo final, mejorar la satisfacción tanto de clientes como de hosteleros.

Para crear una perspectiva real de cómo gestionar el proyecto y qué elementos a añadir, realicé un proceso de historias de usuario o entrevistas, donde fui por los distintos establecimientos de mi sitio de residencia, preguntando e interesándome sobre sus mayores problemas.

Esta labor junto con mi experiencia en estos establecimientos, sacamos como conclusión el cansancio tanto de trabajadores como de dueños los días de Semana Santa, Navidad, Ferias, etc.

Esto provoca que considere que, con este sistema, se aumentará la velocidad de atención a los clientes, produciendo menos hartazgo en estos y menos estrés y agobio para los camareros y trabajadores, ya que, esto supondrá que los clientes puedan entrar al establecimiento, coger mesa y mediante ello pedir directamente lo que necesitan sin pasar por el periodo de tiempo de que los trabajadores puedan atenderle y tardar por la gran cantidad de personas.

3.1.1 Segunda Iteración

La segunda iteración se compone de un análisis más profundo del proyecto.

Se realiza los diagramas adjuntados en el apartado anterior como es el diseño de la base de datos, diseño de los diagramas de casos de uso y diagramas de secuencia.

Se empieza a plantear la planificación del proyecto, es decir, cómo se va a realizar el backend y el frontend.

La primera parte a realizar será la parte de backend, parte fundamental en el proyecto ya que se va a encargar de manejar las diferentes operaciones enviadas por el frontend con la Base de Datos.

Se empiezan a pensar los detalles de la implementación, los diferentes modelos que se deben implementar en el backend, al igual que todo lo relacionado con las diferentes rutas y controladores que debe tener la parte del servidor.

Se empiezan a diseñar las primeras vistas de los storyboards que va a tener la página, una primera muestra de lo que se quiere conseguir, sin entrar en profundidad sobre la interfaz al detalle, ni la apariencia deseada pero sí un primer contacto con la aplicación con la que se realiza el storyboard final.

3.1.2 Tercera iteración

Esta iteración se podría definir como la implementación de todo el backend.

Se instalan las tecnologías necesarias como es NodeJS, MongoDB y Express.

La primera clase que se crea es la clase de categoría.

Esta clase nos embarca dos atributos principales: nombre e icono.

A través de mongoose podemos crear la clase de forma sencilla para hacer la interacción con la base de datos de MongoDB.

```
const mongoose = require("mongoose");

const categoriaSchema = new mongoose.Schema({
  nombre: {
    type: String,
    required: true,
  },
  icono: {
    type: String,
  },
});

exports.Categoria = mongoose.model("Categoria", categoriaSchema);
```

Ilustración 3.1: Modelo de categoría

La siguiente clase importante es Comida o como en algunos otros aspectos nombro como Plato.

Tiene los atributos de nombre, imagen, precio, categoría a la que pertenece y stock.

```
const comidaSchema = new mongoose.Schema({
  nombre: {
    type: String,
    required: true,
  },
  imagen: {
    type: String,
    default: "",
  },
  precio: {
    type: Number,
    required: true,
  },
  categoria: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Categoria",
    required: true,
  },
  stock: {
    type: Number,
    required: true,
    min: 0,
    max: 100,
  },
});
```

Ilustración 3.2: Modelo de comida o plato

La tercera clase es la de usuario, encargada de realizar las operaciones dentro de la Página.

Se compone de un nombre, email, password, imagen que no se utiliza por ahora y y un rol que es el que va a hacer dentro de la aplicación.

```
const usuarioSchema = new mongoose.Schema({
  nombre: {
    type: String,
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true,
  },
  password: {
    type: String,
    required: true,
  },
  img: {
    type: String,
  },
  role: {
    type: String,
    required: true,
    default: "USER_ROLE",
  },
});
```

Ilustración 3.3: Modelo de usuario

Las dos últimas tienen relación ya que una se compone de otra y son el carrito que es donde se van a añadir los diferentes platos y el pedido que es la confirmación de este carrito en forma de comanda.

El carrito se compone de los atributos cantidad y comida, mientras que la clase pedido se compone del carrito en sí con los diferentes platos y cantidad y también se compone del estado del pedido, precio total, usuario que ha realizado ese pedido y el día que se ha pedido.

```
const mongoose = require("mongoose");

const carritoSchema = new mongoose.Schema({
  cantidad: {
    type: Number,
    required: true,
  },
  comida: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Comida",
  },
  precio: {
    type: Number,
    required: true,
  },
});

exports.Carrito = mongoose.model("Carrito", carritoSchema);

```

Ilustración 3.4: Modelo del carrito

```
const mongoose = require("mongoose");

const pedidoSchema = new mongoose.Schema({
  articulo: [
    {
      type: mongoose.Schema.Types.ObjectId,
      ref: "Carrito",
      required: true,
    },
  ],
  estado: {
    type: String,
    required: true,
    default: "Pendiente",
  },
  precioTotal: {
    type: Number,
  },
  usuario: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Usuario",
  },
  diaPedido: {
    type: Date,
    default: Date.now,
  },
});
```

Ilustración 3.5: Modelo del pedido

Estos son los modelos sobre los que se van a modelar los datos, pero, para crear estas entidades necesitamos de una lógica en forma de CRUD.

Estos se realizan mediante la lógica que nos proporciona la tecnología MEAN STACK.

Las operaciones CRUD se basan en la creación, lectura, actualización y eliminación de las diferentes entidades que van a formar parte del proceso de flujo dentro de la aplicación web.

La primera, al igual que al explicar los modelos, será la entidad categoría.

Para su creación, eliminación, actualización y lectura se ha realizado el siguiente código:

```
router.get("/", async (request, response) => {
  const listaCategoria = await Categoria.find();

  if (!listaCategoria) {
    response.status(500).json({
      success: false,
    });
  }
  response.send(listaCategoria);
});
```

Ilustración 3.6: Método GET para obtención de categoría

```
router.put("/:id", async (request, response) => {
  const categoria = await Categoria.findByIdAndUpdate(request.params.id, {
    nombre: request.body.nombre,
    icono: request.body.icono,
  });

  if (!categoria) {
    return response.status(404).send("La categoria no puede ser creada");
  }

  response.send(categoria);
});

router.post("/", async (request, response) => {
  let categoria = new Categoria({
    nombre: request.body.nombre,
    icono: request.body.icono,
  });
  categoria = await categoria.save();

  if (!categoria) {
    return response.status(404).send("La categoria no puede ser creada");
  }

  response.send(categoria);
```

Ilustración 3.7: Método POST y PUT para creación de categoría

```
router.delete("/:id", (request, response) => {
  Categoria.findByIdAndRemove(request.params.id)
    .then((categoria) => {
      if (categoria) {
        return response
          .status(200)
          .json({ success: true, message: "La categoria está eliminada " });
      } else {
        return response
          .status(404)
          .json({ success: false, message: "Categoria no encontrada" });
      }
    })
    .catch((error) => {
      return response.status(400).json({ success: false, error: error });
    });
});
```

Ilustración 3.8: Método DELETE para borrado de categoría

Estos son las operaciones elementales en el manejo de los datos que se va a realizar con el modelo Categoría.

Para el resto de modelos sería algo muy parecido ya que se basa en un funcionamiento de comprobar si ese modelo existe pues crearlo o eliminarlo gracias a las funcionas auxiliares que nos proporciona Express.

Las pruebas realizadas en el desarrollo del backend se han hecho gracias a la herramienta Postman, que nos proporciona una interfaz muy amigable, donde mediante la introducción de la URL de nuestro backend, podemos ir probando las diferentes funciones CRUD de los modelos establecidos en el proyecto.

Por lo tanto, para probar esta parte se ha ido procediendo con cada modelo, sus diferentes operaciones y viendo en la salida que nos proporciona Postman, si los datos eran correctos o había algún error HTTP en el proceso.

Para hacer las pruebas pertinentes y enriquecer la API, se han hecho las siguientes pruebas en Postman:

1. CRUD para las categorías:

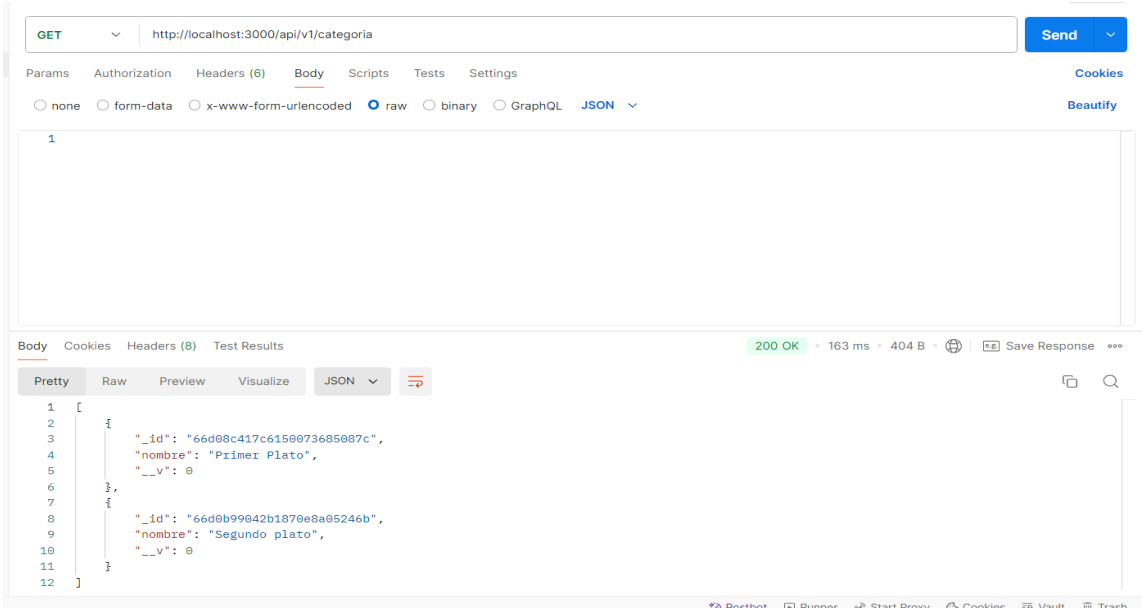


Ilustración 3.9: Get Categoria desde Postman

2. CRUD para los platos:

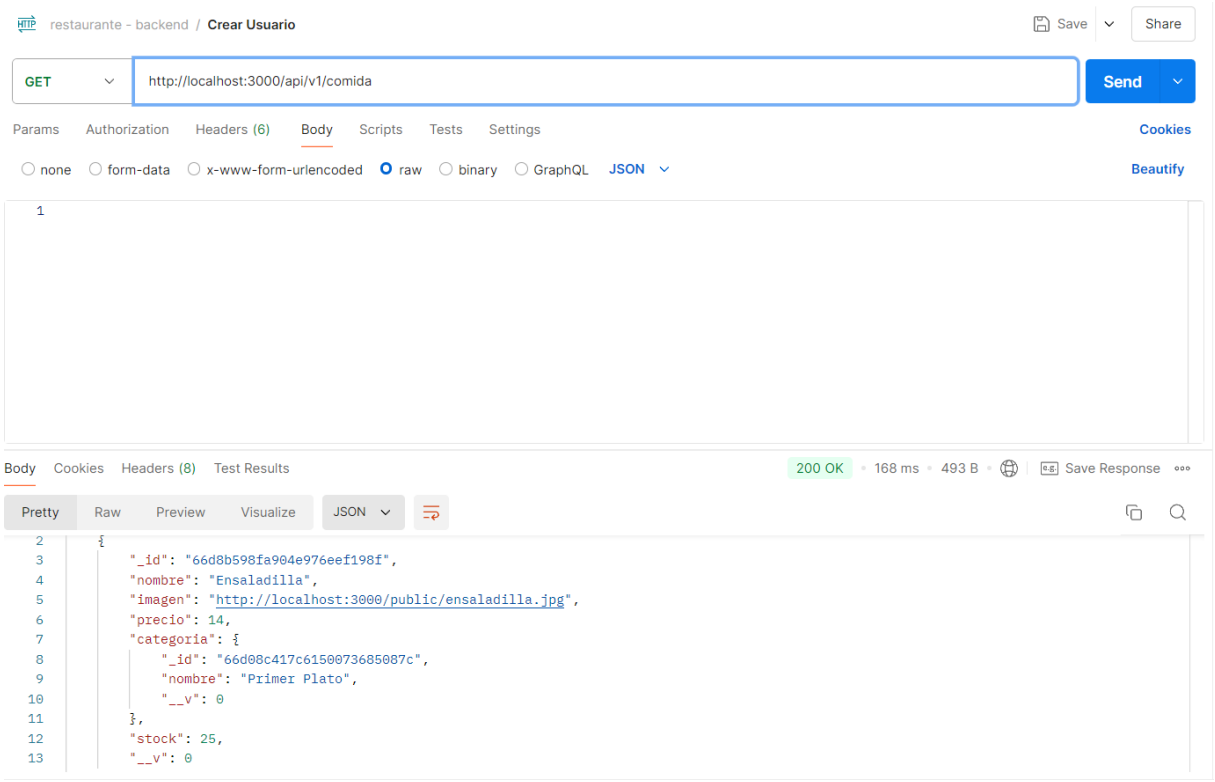


Ilustración 3.10: Get Comida desde Postman

3. CRUD para los trabajadores:

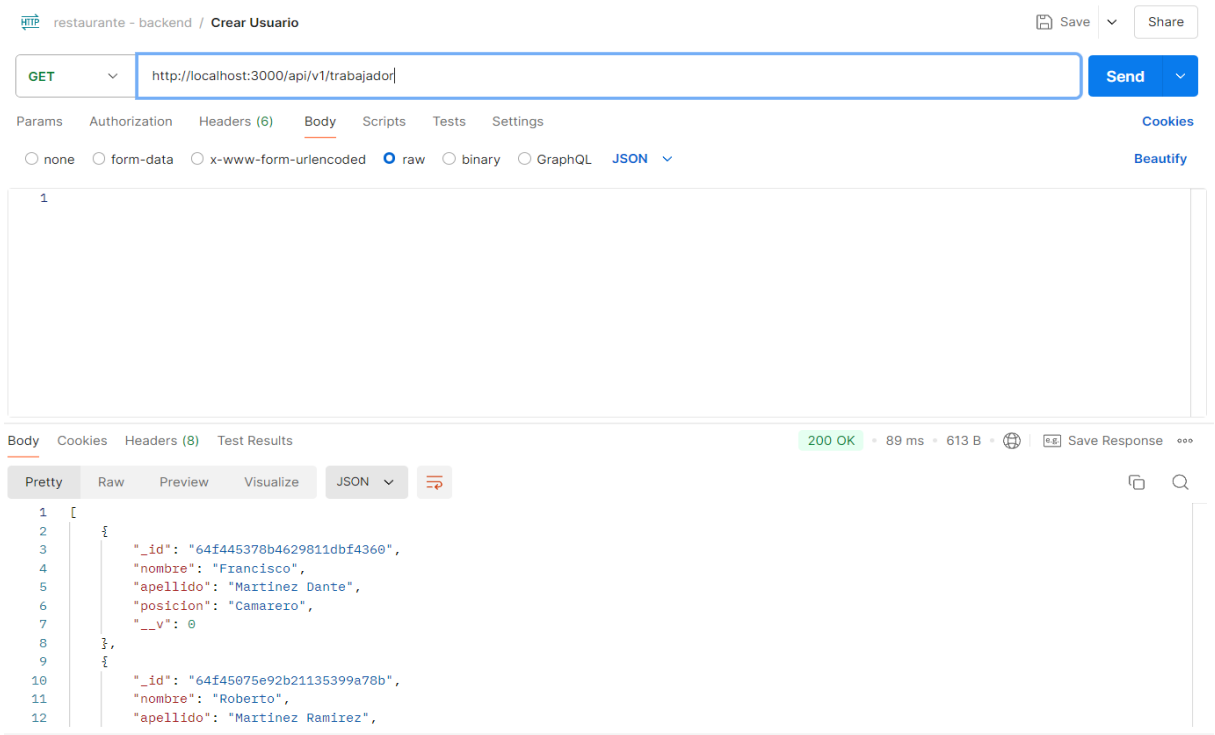


Ilustración 3.11: Get Trabajador desde Postman

Se ha realizado lo mismo para las demás entidades presentadas en el proyecto.

Problemas surgidos durante el enriquecimiento de la API:

- Se podían añadir platos repetidos con el mismo nombre, siendo una incongruencia ya que no pueden hacer dos platos con el mismo nombre. Para solucionarlo se añade una condición en el backend en tanto si el objeto ya existe en la BBDD, de una respuesta de que ese objeto ya ha sido creado anteriormente.

- No se podían eliminar platos ya que no estaba bien configurado la función findIdAndRemove que nos proporciona NodeJS.

- No se podía hacer login del usuario ya que no se creaba correctamente el Token para el acceso. Se ha utilizado JWT para solucionar este aspecto, proporcionándonos herramientas necesarias para la creación del token para el usuario específico.

3.1.3 Cuarta Iteración

En esta iteración, se empezó a formalizar todo lo que es el frontend, es decir, todo lo que va a formar a ser la aplicación web, empezando por la interfaz sin una funcionalidad muy concreta.

En el frontend habrá dos partes, una dedicada completamente para el administrador del establecimiento hostelero y otro para el cliente, dónde se va a producir toda la dinámica de visualización de diferentes platos, lectura de la carta de presentación del establecimiento, y, sobre todo, la realización del pedido.

En esta parte de frontend tenemos dos diferentes partes: la parte que verán los clientes para realizar toda la gestión del pedido de la comanda y la parte del

administrador donde verá toda la parte relacionada con la gestión de la parte del restaurante.

Para la parte del cliente, se ha usado una plantilla encontrada en la plataforma W3LAYOUTS con algunos cambios que se han ido realizando de apariencia de la estructura y de la interfaz.

Para poner en funcionamiento la plantilla, he tenido que añadir las dependencias que tienen tanto de JS, CSS, entre otros. Esto se añade en la carpeta assets del proyecto y en el archivo index.html se añaden estas referencias mediante la etiqueta link.

```
<html lang="en">
  <head>
    /> -->
    <!-- slider css -->
    <link href="assets/css/bootstrap.css" rel="stylesheet" type="text/css" />
    <!-- bootstrap css -->
    <link href="assets/css/style.css" rel="stylesheet" type="text/css" />
    <!-- custom css -->
    <link href="assets/css/font-awesome.min.css" rel="stylesheet" />
    <!-- fontawesome css -->
    <!-- //css files -->

    <!-- google fonts -->
    <link
      href="//fonts.googleapis.com/css?family=Lato:100,100i,300,300i,400,400i,700,700i,900,900i"
      rel="stylesheet"
    />
```

Ilustración 3.12: Index.html con los assets añadidos

Para la parte del admin se ha realizado desde 0 con tecnologías como CSS, HTML, Bootstrap, haciendo una interfaz sencilla de entender, sencilla de manejar para que el usuario administrador no tenga problemas para realizar sus funciones de manera sencilla.

Empecemos mostrando la parte del restaurante:

En esta primera captura se puede ver el header y el menú asociado a la página donde se puede ver varios botones como son el about us (Información detallada del restaurante con un poco de información del mismo.

Ilustración 3.13: Interfaz Home de la interfaz

En esta primera captura se puede ver el header y el navbar asociado a la página donde se puede ver varios botones como son el about us (Información detallada del restaurante con un poco de información del mismo).

Parte de menú donde el usuario podrá realizar la mayor función de su apartado que es interactuar con todos los platos que hay para poder añadirlos en el carrito para su posterior gestión del pedido.

Hay dos botones relacionados con el sistema de autenticación del usuario donde podrá loguearse si ya tiene una cuenta previamente creada o registrarse si todavía no tiene una cuenta.

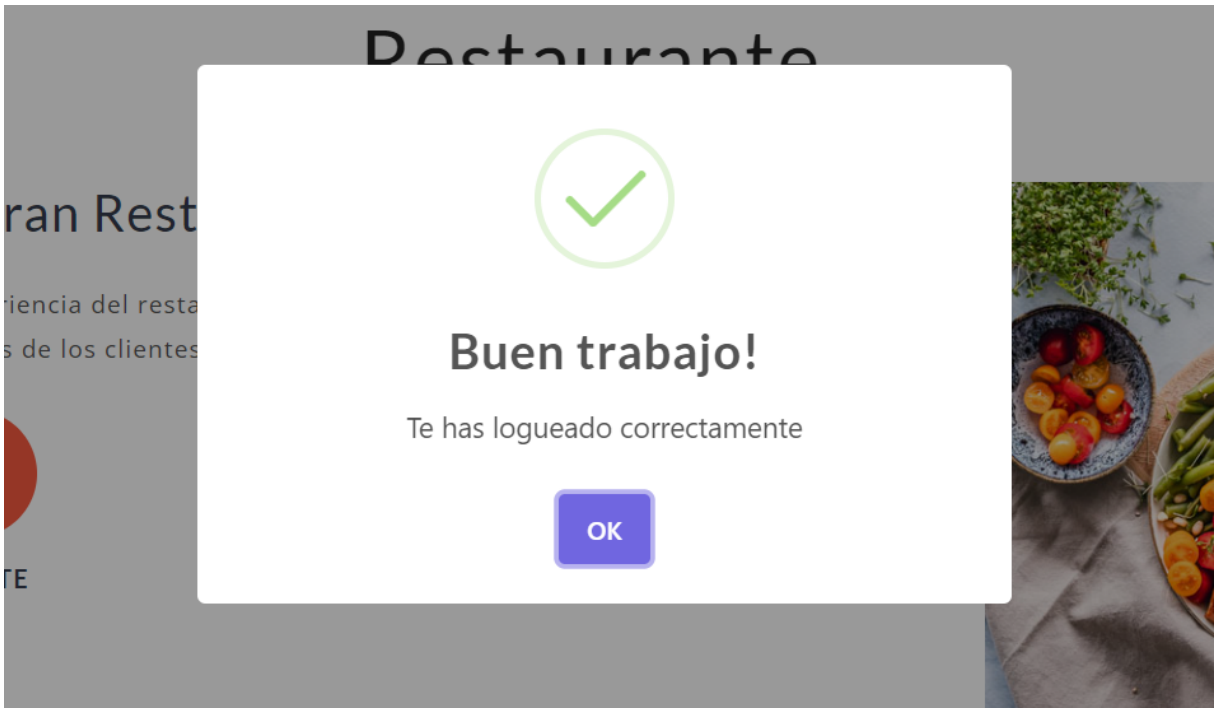


Ilustración 3.14: Login correcto en el sistema.

Para la parte de registro, dejo captura de cómo se interactúa con Sweet Alert 2 para notificar que se ha registrado correctamente.

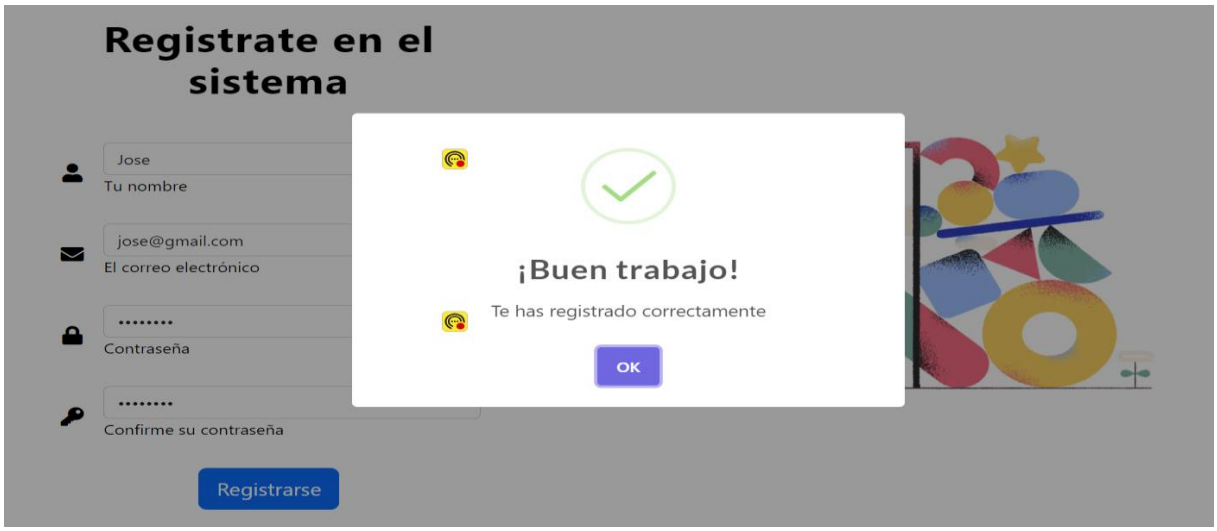


Ilustración 3.15: Registro correcto en el sistema.

En el apartado de contacto, podemos dejar un comentario que posteriormente en el panel del admin, el propio administrador podrá revisarlo. La idea de esto es que el papel del cliente sea importante ya que mediante su crítica pueda el administrador o

gestor del restaurante tomar medidas o aceptar sugerencias de los clientes para mejorar en el devenir del establecimiento.

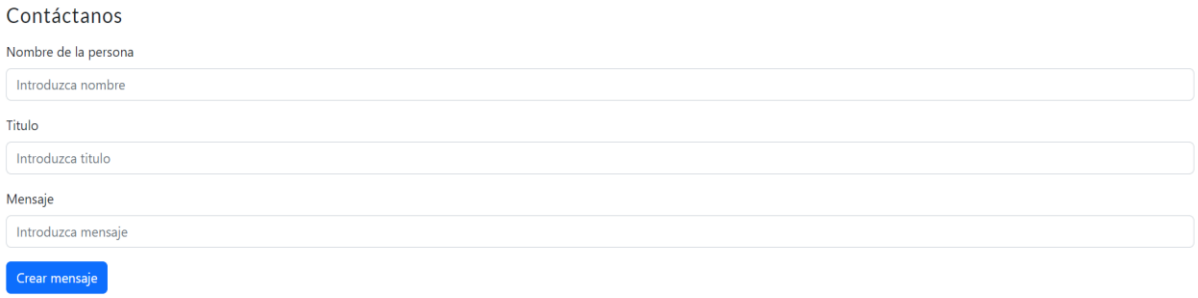


Ilustración 3.16: Formulario para enviar mensaje a admin

En la parte final hay un footer con información más sencilla del restaurante como es la ubicación, número de teléfono, etc…

Pruebas realizadas por parte del cliente:

1. Se ha realizado el registro de diversos usuarios para comprobar que se registra correctamente en el sistema.
2. Se realiza el login de los usuarios creados para ver que, en función del rol establecido para cada usuario, se redirige correctamente hacia un panel u otro.
3. Se revisa que los platos aparezcan correctamente en la interfaz.
4. Se envía mensajes con quejas o sugerencias respecto al funcionamiento del restaurante.

A continuación, pasamos a la parte del admin:

El panel de admin viene estructurado por un sidebar donde en diferentes menús podemos seleccionar para elegir qué queremos revisar (categorías, comidas, trabajadores, mensajes, etc…)

Aquí un ejemplo de la interfaz como la podemos ver con el apartado de los trabajadores:

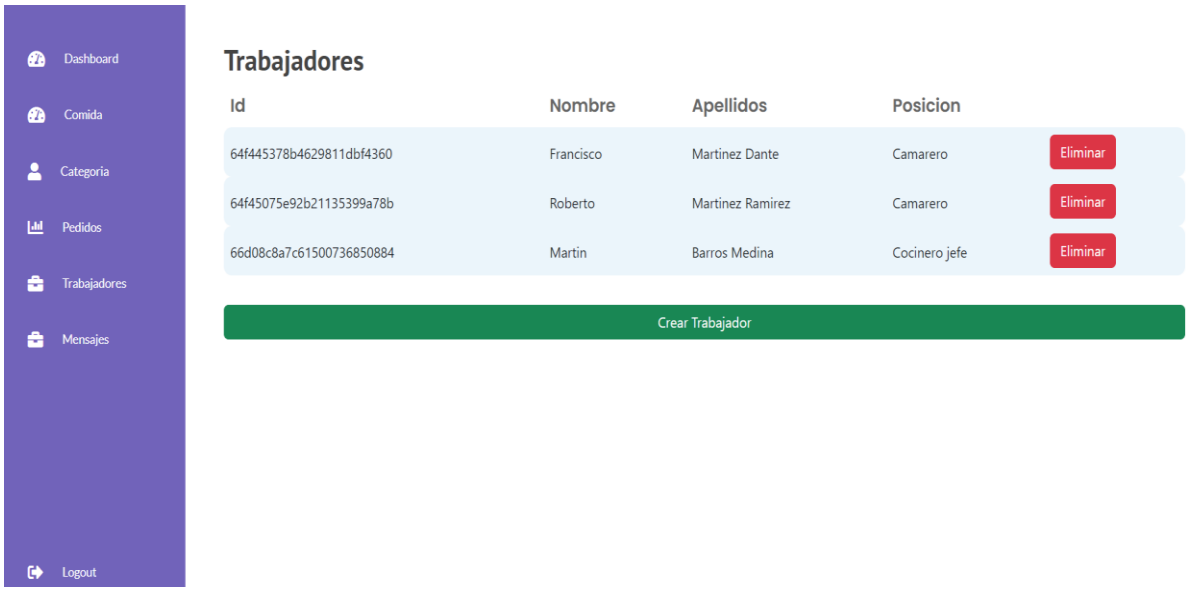


Ilustración 3.17: Interfaz del panel de administrador

Al igual que se puede visualizar en este ejemplo, también se puede revisar en los demás apartados.

Para poder añadir tanto categorías, como comidas, trabajadores, se ha realizado mediante un formulario clicando en el botón de Crear categoría/comida/trabajador.

Ilustración 3.18: Formulario para añadir platos

Con este formulario, el administrador puede gestionar sus platos en función de las necesidades de diferentes fechas, necesidad de cambio o algún motivo extra que le pueda llevar a realizar el cambio.

El formulario es muy similar para los otros apartados pero rellenado con los campos necesitados por cada clase.

Pruebas realizadas en el panel de admin:

1. Se han creado diferentes categorías, platos, trabajadores y se ha realizado la gestión de estos para comprobar que existe consistencia en los datos.
2. Se han eliminado trabajadores, mensajes, para comprobar que funciona correctamente el apartado del botón creado para Delete.

4 MODELO DE NEGOCIO

El auge de la tecnología en los últimos años en todos los sectores, hace que la hostelería sea uno de los valores más importantes en los que poder innovar y mejorar la calidad que actualmente pueden ofrecer.

Debido a la poca innovación que ha sufrido el sector hostelero en los últimos años, se puede plantear un surgimiento de nuevas tecnologías que mejoren el servicio que nos plantean y el resultado final de cara a los clientes.

Con la implementación de nuevas tecnologías y herramientas, se espera que el sector hostelero sume puntos en cuánto a la inversión en tecnología, que en los últimos años no iba de la mano, ya que la tecnología avanza a pasos agigantados mientras la hostelería sigue el modelo tradicional que se lleva viendo a lo largo de los años.

Modelo establecido por un dueño, que en muchas ocasiones hace las labores de camarero y tiene que lidiar con todos los problemas que van surgiendo durante las comandas; esto sumado a un día tras otro, hace que nos podamos plantear introducir herramientas de ayuda para mejorar en todos los sentidos.

4.1 Objetivos

Los objetivos que nos planteamos cuando fundamos una empresa, deben basarse en los siguientes dos: Objetivos cualitativos y objetivos cuantitativos.

Los objetivos cualitativos son aquellos objetivos que fija la empresa para conseguir un mejor posicionamiento e imagen en el mercado. Al plantearlos se establecen metas más genéricas. Son más subjetivos, por lo tanto, menos tangibles, más complicados de medir que los objetivos cuantitativos.

La recomendación es que se planteen unos objetivos claros, coherentes y alcanzables y que vayan en concordancia unos con otros.

Los objetivos cualitativos que podríamos lograr podrían ser:

- **Imagen**: la imagen es la idea que los consumidores tienen de mi empresa. Las empresas a veces se marcan como objetivo crear una imagen en la mente de los consumidores.

- **Consolidarse en el mercado**: La consolidación en el mercado en el que competimos es muy importante para poder sentar unas bases que pueden ser muy importantes para un futuro empresarial. Seremos duros competidores si estamos asentados en el sector del que participamos como nicho.

- **Calidad**: La calidad es un punto de vista muy importante para una empresa, ya que puede ser un escaparate perfecto para subir muchos objetivos de los nombrados anteriormente como puede ser la imagen.

  Con una calidad reconocible, nos haremos muy fuertes en el sector.

- **Responsabilidad social**: La responsabilidad social es un punto muy importante en el día a día, ya que en una sociedad que cada vez mira más por nuestro entorno, es importante cumplir una serie de estándares que pueden hacernos tener una imagen importante en cuánto a la posible publicidad de nuestra empresa.

Los objetivos cuantitativos, por su propio carácter, son los más sencillos de medir, suelen estar ligados a cantidades, al beneficio en sí de la empresa u otras variables similares y son apreciables en el corto plazo.

Los objetivos cuantitativos que podríamos lograr podrían ser:

- **Aumentar el beneficio**: Cuando una empresa no tiene un rendimiento bueno, es augurio de mal futuro. Toda empresa trata de ganar dinero, sino gana dinero, está abocada a la desaparición.

- **Conseguir un aumento del número de clientes**: Consiste en hacerse poco a poco con un número alto de clientes, ya sea en el principio ofreciendo recompensas que pueden ser no tan beneficiosas para la economía de la empresa como otras, pero que de primeras da una imagen muy buena y nos ayudará, a la larga, a seguir mejorando y aumentando números.

- **Crecer**: Crecer puede implicar muchos resultados, como puede ser crecer a nivel económico, a nivel de fama, crecer empresarialmente entre otros posibles resultados que pueden devenir de este crecimiento.

- **Mejorar su valor de mercado**: Si la empresa cotiza en bolsa, esta aspira a tener la mejor valoración posible para ganar socios o accionistas que quieran invertir en ella a través de la participación por acciones para poder vender esa participación a mayor precio después, pero mientras, quien se beneficia es la empresa.

4.2 Análisis del entorno

El análisis del entorno es uno de los puntos más importantes en los que analizar en cuanto a la implantación de una empresa, ya que, realizar un análisis del entorno de tu empresa consistirá en la evaluación profunda de tu entorno económico, normativo, social, tecnológico o cultural que pueda afectarte de manera directa o indirecta.

4.2.1 Factores políticos

Los factores políticos nos pueden llevar a muchas conjeturas, a tomar diferentes decisiones según el devenir de la política.

Entre alguno de los factores políticos, uno de los importantes es el devenir de un país políticamente, es decir, nos debemos preguntar si es un país estable en términos políticos, si este país sufre de constantes cambios políticos en pocos años, este período de incertidumbre puede llevarnos a no depositar la suficiente confianza en este país.

Además, estos cambios pueden llevar a cambios en materia fiscal, los impuestos que los ciudadanos pagan y los impuestos que las empresas deben pagar por su actividad, debe ser un factor fundamental a analizar para poder tomar decisiones.

El panorama político también puede venir influenciado por la situación de incertidumbre en muchos ámbitos nacionales como puede ser inestabilidad de algunas regiones, discrepancias entre diferentes gobiernos autonómicos con el gobierno nacional.

4.2.2 Factores económicos

Otro de los aspectos importantes de análisis es el tema económico, ya que esto puede influir directamente en nuestra empresa.
El saber las condiciones económicas del sector en el queremos emprender, al igual que conocer las diferentes tendencias de la economía mundial, es importante para decidir si es buen momento de emprender.

Uno de los factores a analizar económicamente es la inflación, ya que este nivel nos dará a conocer el precio de los productos al igual que es una tendencia fiable de la situación de la economía. Actualmente, en la situación en la que se encuentra el mundo con la guerra de Rusia y Ucrania y las tensiones ocasionadas con el gas ruso, las economías sufren una gran inflación, haciendo este ser un período de difícil apuesta para emprender.

En otra parte, es importante analizar el sector en el que queremos emprender; el sector hostelero, muy golpeado por la crisis del coronavirus y por los problemas económicos de las familias, se convierte en un sector en el que emprender es una incertidumbre total, siempre entendiendo que el sector hostelero es uno de los grandes motores del país, movido por el turismo nacional, y sobre todo, internacional, por lo que, en cuánto la situación económica mejore, será uno de los motores del país; emprender puede llegar a ser un valor seguro.

4.2.3 Factores sociales

Uno de los puntos positivos del sector de hostelería es la socialización.

El ser humano, por naturaleza es un ser que necesita estar en permanente contacto con sus homónimos, necesita intercambiar opiniones, ideas, sensaciones, en definitiva, hablar con sus familiares, amigos o, incluso, en algunas ocasiones, con extraños.

Uno de los sitios en los que se ha realizado esto de forma más natural es en un restaurante o en un bar.

Las personas, en un ambiente coloquial, se siente a gusto y junto, con el poder de los alimentos, consigue sumar uno de los placeres más preciados para una persona.

Por lo que, este factor ayuda a emprender en un sector que ayuda a las personas a mejorar su día, a encontrar nuevos amigos, encontrar nuevas personas que pueden cambiarte la vida.

4.2.4 Factores tecnológicos

La evolución de la tecnología en un sector donde no se ha producido muchos avances significativos, puede suponer un gran nicho de mercado de emprendimiento.

El uso de la energía es un aspecto importante actualmente, debido a la subida del precio de la luz y el uso cada vez más acentuado de las energías verdes o energías renovables, hace que nuestra empresa pueda aportar sostenibilidad con el uso de estas energías renovables.

La irrupción de tecnologías como Machine Learning, nueva maquinaria o dispositivos electrónicos, el cada vez más uso de las tecnologías para tareas sencillas, hace que la empresa tenga un valor añadido en el posible desarrollo de herramientas que puedan solucionar problemas para los clientes.

4.2.5 Factores ecológicos

El factor ecológico es un ámbito cada vez más importante a la hora de analizar el entorno de una empresa, ya que, debido a las contaminaciones de algunos sectores, añadido con el cambio climático que parece cada vez más presente en nuestras vidas, debemos tener muy presente este factor.

En nuestro caso, será una empresa que no emitirá gases contaminantes, ya que su único uso será mediante energías, y siempre en la medida de lo posible, mediante energías renovables.

Por lo tanto, nuestro uso del medio ambiente será sostenible, siempre mediante el reciclaje de los posibles problemas que tengamos a nivel tecnológico, ya sea por rotura de ordenadores o cualquier complemento como teclados, ratones, monitores, que ayudan en la acción diaria de los trabajadores.

4.2.6 Factores legales

Otro factor que suele pasar un poco desapercibido en el análisis de entorno de una empresa es el factor legal.

Las empresas deben estar atentas a las diferentes leyes que debe cumplir una empresa para no caer en posibles fraudes de ley o delitos que pueden desembocar en sanciones graves a nivel empresarial.

Las empresas deben conocer perfectamente las leyes de propiedad intelectual, la ley de protección de datos o las diferentes licencias de uso de su producto.

Otro aspecto importante es conocer factores legales-políticos como puede ser el salario mínimo interprofesional, por lo que, debe saber los posibles cambios en estas medidas y adecuarse a ellas, al igual que las bajas por maternidad/paternidad o los diferentes códigos que deben regir una empresa.
Un aspecto importante para analizar los aspectos positivos y negativos de la empresa en el determinado sector es realizar una matriz DAFO.

- ¿Qué es una matriz DAFO? [Cómo hacer un análisis DAFO (con ejemplos y plantilla)]

La matriz de análisis DAFO, es una conocida herramienta estratégica de análisis de la situación de la empresa.

La matriz de análisis DAFO permite identificar tanto las oportunidades como las amenazas que presentan nuestro mercado, y las fortalezas y debilidades que muestra nuestra empresa.

Esta matriz DAFO deriva en dos análisis: interno y externo.

En el análisis externo de la empresa se identifican los factores externos claves para nuestra empresa, como por ejemplo los relacionados con: nuevas conductas de clientes, competencia, cambios del mercado, tecnología, economía.

En este análisis externo, analizamos las oportunidades y las amenazas.

- **Oportunidades**: Representan una ocasión de mejora de la empresa. Las oportunidades son factores positivos y con posibilidad de ser explotados por parte de la empresa.
- **Amenazas**: Pueden poner en peligro la supervivencia de la empresa o en menor medida afectar a nuestra cuota de mercado. Si identificamos una amenaza con suficiente antelación podremos evitarla o convertirla en oportunidad.

Por otra parte, tenemos en análisis interno; en el análisis interno de la empresa se identifican los factores internos claves para nuestra empresa, como por ejemplo los relacionados con: financiación, marketing, producción, organización.

Se pueden dividir en dos partes: las fortalezas y las debilidades.

- **Fortalezas**: Son todas aquellas capacidades y recursos con los que cuenta la empresa para explotar oportunidades y conseguir construir ventajas competitivas.
- **Debilidades**: Son aquellos puntos de los que la empresa carece, de los que se es inferior a la competencia o simplemente de aquellos en los que se puede mejorar.

Esta matriz DAFO es fundamental para identificar bien la situación en la que puede encajar nuestra empresa en el sector y poder identificar cuáles son los aspectos positivos y negativos y a partir de ahí, saber cuáles de ellas nos hacen más fuerte para poder tomar esa ventaja competitiva respecto otras empresas.

Ilustración 4.1: Matriz DAFO detallada

Con este análisis DAFO, podremos establecer las diferentes estrategias a seguir y el plan de mercadotecnia que podemos emplear para poder crear una empresa solvente.

4.3 Plan de mercadotecnia

En este apartado vamos a detallar cuál es el plan de la empresa en cuanto a la estrategia de marketing que se va a desarrollar. [Qué es un plan de marketing y cómo crearlo (incluye plantillas].

- **Estrategia de producto**: El producto a desarrollar por la empresa será, principalmente el software destinado a la gestión y análisis de las comandas de la hostelería, pudiendo en un futuro incluir innovaciones en cuánto al sector tecnológico que pueda suponer un avance o un paso más allá a lo que actualmente forma el sector hostelero.

  Este software será de uso en los diferentes ordenadores que tengan el establecimiento en las mesas.

  Este producto se podrá ir aclimatando según las necesidades de los clientes hosteleros a los que se les suministra el producto.

- **Estrategia de precios**: El sistema se podrá conseguir mediante la compra de la aplicación web, es decir, se les proporcionará la aplicación para instalarlo en un servidor.

  Esta compra se recomienda, inicialmente para restaurantes y bares pequeños y medianos, que necesiten mejorar su servicio y que no tenga los conocimientos empresariales suficientes para analizar las posibles mejorar a añadir en su servicio.

El establecimiento se podrá poner en contacto con la empresa para un mantenimiento de la aplicación, es decir, subsanar los posibles problemas que pueda tener la aplicación junto con los posibles bugs que se vayan encontrando en el uso.

Además, este mantenimiento puede suponer la posibilidad de incluir nuevas funcionalidades a la aplicación web, consensuadas con la empresa desarrolladora.

- **Estrategia de promoción y comunicación**: La estrategia a usar para dar a conocer el producto será principalmente mediante las redes sociales.

  Debido al gran auge de las redes sociales, podemos publicitarnos de una manera totalmente gratuita y darnos a conocer, primeros por las zonas cercanas a nuestra localización, y cuando hayamos conseguido esa fama, podamos establecernos en zonas más lejanas a nuestra localización central.

  Se podrá poner en contacto con radios locales para pedir que el producto sea publicitado, aunque es un método que cada vez está mas en desuso debido al uso cada vez menos frecuente de las radios.

  En el lugar natural de la empresa podemos ir publicitándonos mediante pequeños folletos repartidos en los distintos establecimientos hosteleros, ofreciendo una pequeña prueba gratis para que el establecimiento pueda ver su uso y cómo les mejora su productividad y eficiencia.

  Otra opción es mediante una campaña de marketing, administrada por una empresa especializada en el sector, para darnos a conocer entre los mayores establecimientos posibles.

Con estas estas estrategias se espera conseguir una campaña de comunicación exitosa, que dé a conocer el producto y que pueda hacernos crecer en nuestro futuro como empresa.

- **Estrategia de venta y distribución**: Una posible estrategia de venta y distribución puede ser la búsqueda de nuevos incentivos que puedan ser un foco de atención sobre nuestra empresa y nuestros productos, esto puede llevarnos a la búsqueda de productos más novedosos, mejorar los plazos de entrega, una atención personalizada con los posibles clientes.

  Una búsqueda de proveedores puede llevarnos hacia una estrategia de distribución correcta.

4.4 Forma jurídica de la empresa

Para poder ser una empresa, debemos decidir qué tipo de sociedad queremos formar [Tipos de sociedades: comparativa de formas jurídicas].

Existe una gran cantidad de formas jurídicas en el país, por lo que vamos a analizar cuál es la que mejor se adecúa a nuestras necesidades y a lo que queremos conseguir como empresa.

La forma jurídica de una empresa es la identidad que tiene legalmente, teniendo en cuenta la responsabilidad que tendrán sus socios frente a la Ley.

Es la modalidad que tendrá una empresa a la hora de desarrollar una actividad empresarial o profesional, y qué, por tanto, definirá sus obligaciones tributarias, sus responsabilidades frente a terceros y su régimen de funcionamiento interno.

4.4.1 Empresario individual

Es la persona física que realiza en nombre propio y por medio de una empresa, una actividad comercial, industrial o profesional.

- No tiene una regulación legal específica y está sometido en su actividad empresarial a las disposiciones generales del Código de Comercio en materia mercantil y a lo dispuesto en el Código Civil en materia de derechos y obligaciones.
- Control total de la empresa por parte del propietario, que dirige su gestión.
- La personalidad jurídica de la empresa es la misma que la de su titular (empresario), quien responde personalmente de todas las obligaciones que contraiga la empresa.

Las ventajas de usar esta forma es que es perfecta para empresas de un tamaño muy reducido, puede ser más económico que otras formas jurídicas.
Las desventajas pueden ser que responde con su patrimonio personal de las deudas generadas en su actividad, si su volumen de beneficio es importante, puede estar sometido a tipos impositivos muy elevados.

4.4.2 Sociedad Anónima

Es una sociedad mercantil de tipo capitalista en la que el capital social está dividido en acciones.

La ventaja más evidente es que los accionistas no responden de las deudas sociales con su patrimonio personal (bienes particulares), y, por otro lado, existe la posibilidad de atraer capitales ajenos por medio de la emisión de obligaciones.

La ventaja más evidente es que los accionistas no responden de las deudas sociales con su patrimonio personal (bienes particulares), y, por otro lado, existe la posibilidad de atraer capitales ajenos por medio de la emisión de obligaciones.

La convocatoria deberá hacerse por anuncio publicado en el Boletín Oficial del Registro Mercantil y en uno de los diarios de mayor circulación en la provincia con quince días de antelación a la fecha fijada para la celebración de la Junta.

4.4.3 Sociedad Limitada

Es una sociedad de naturaleza mercantil, con un capital determinado, integrado por las participaciones sociales de los socios, con la gran ventaja de que éstos no responderán personalmente de las deudas sociales

El capital social no podrá ser inferior a 3.000 euros, y se desembolsará íntegramente desde su origen.

Aunque lo normal es que haya dos socios o más, también existe la modalidad de Sociedad Limitada Unipersonal. Surge como respuesta del empresario individual para ejercitar su industria o comercio con responsabilidad limitada frente a sus acreedores.

Pueden darse dos tipos de sociedades unipersonales:

La constituida por un único socio, sea persona natural o jurídica.

La constituida por 2 o más socios cuando todas las participaciones hayan pasado a ser propiedad de un único socio.

Necesariamente habrán de constar en escritura pública que se inscribirá en el Registro Mercantil.

Estas son las principales formas jurídicas que pueden crearse para una empresa, por lo que para decidir cuál es la mejor o la peor, tenemos que tener clara cuál de ellas es la que nos va a suponer un mayor beneficio.

En este caso, para la empresa se podría elegir una sociedad limitadas ya que permiten una gestión interna sencilla, y el hecho de que los socios solo respondan con el capital aportado frente a terceros constituye una gran ventaja.

5 CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo ha supuesto una innovación en cuanto a la gestión y análisis del sector hostelero, ya que se le ha proporcionado una herramienta a los bares y restaurantes, sobre todo orientados a los más pequeños, para poder mejorar su política como establecimiento hostelero, llevando a cabo posibles cambios si ven que hay cosas que fallan o manteniendo la estrategia si el rendimiento actual de la empresa es bueno.

Este trabajo ha servido para entender cómo los hosteleros tienen que luchar día a día con los problemas que van surgiendo en cuánto al servicio.

Estos problemas pueden ser, falta de almacenamiento de productos, desecho de productos debido a su no consumo, enfado de clientes por su mal servicio, pérdidas económicas severas pudiendo llevar al cierre del establecimiento.

Los planes futuros son seguir ayudando y apostando por un sector que es muy importante para nuestra vida.

En cuanto a futuros desarrollos pueden venir de la mano del análisis, incluir muchos más parámetros de los presentados actualmente, que les dé una información mucho más detallada del funcionamiento de su establecimiento y qué es lo que les gusta y no a los clientes.

Otra posible mejora futura es la incorporación de un foro entre clientes y dueños, para expresar las opiniones sobre el establecimiento, ya no sólo en cuanto al servicio y productos, sino al establecimiento en sí, es decir, si el establecimiento por dentro es acogedor, hay problemas en cuánto al mobiliario o cualquier posible queja relacionado con el disfrute del tiempo de los clientes en el establecimiento.

Otro de los posibles trabajos futuros puede ser el contenerizar la aplicación para hacerla más eficiente, más ligera.

Aspectos posibles a añadir: una gestión de reservas para que el usuario desde el mismo establecimiento pueda hacer una reserva de una mesa para un día específico.

Además de esta posible mejora, se puede gestionar el añadir un sistema multilenguaje para aquellos establecimientos turísticos que requieran la posibilidad de varios idiomas.

Mejora necesaria es añadir una pasarela de pago como Stripe o Paypal al proyecto ya que no he sido capaz de poner en funcionamiento la pasarela, por lo que el proceso de pago se haría de forma manual.

6 APÉNDICES

6.1 Guía original del Trabajo Fin de Título

PROPUESTA DE TRABAJO DE FIN DE GRADO

GRADO EN:

Grado en Ingeniería Informática

ESPECIALIDAD:

Tecnologías de la Información

MENCIÓN:

General

TÍTULO DEL TRABAJO:

Aplicación web de gestión de comandas en hostelería

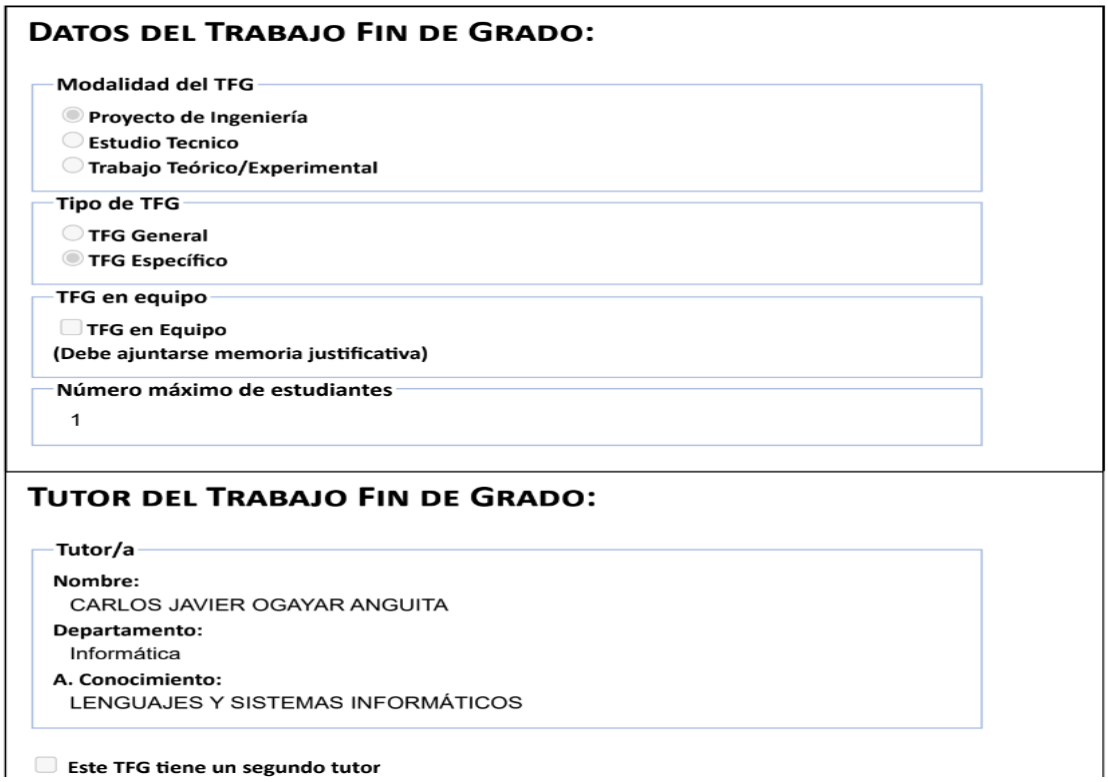
Idioma:

Castellano

DESCRIPCION CORTA DEL TFG:

Este trabajo consiste en la realización de una aplicación web para la gestión de comandas en hostelería. El proceso debe ser útil y sencillo mediante una experiencia de usuario (UX) eficiente, con el objeto de facilitar el trabajo a los empleados y dar un servicio más rápido y fiable a los clientes.

Ilustración 6.1: Guía Original del trabajo (PARTE 1)



DATOS DEL TRABAJO FIN DE GRADO:

Modalidad del TFG

- (•) Proyecto de Ingeniería
- ( ) Estudio Tecnico
- ( ) Trabajo Teórico/Experimental

Tipo de TFG

- ( ) TFG General
- (•) TFG Específico

TFG en equipo

- [ ] TFG en Equipo

(Debe ajuntarse memoria justificativa)

Número máximo de estudiantes

1

TUTOR DEL TRABAJO FIN DE GRADO:

Tutor/a

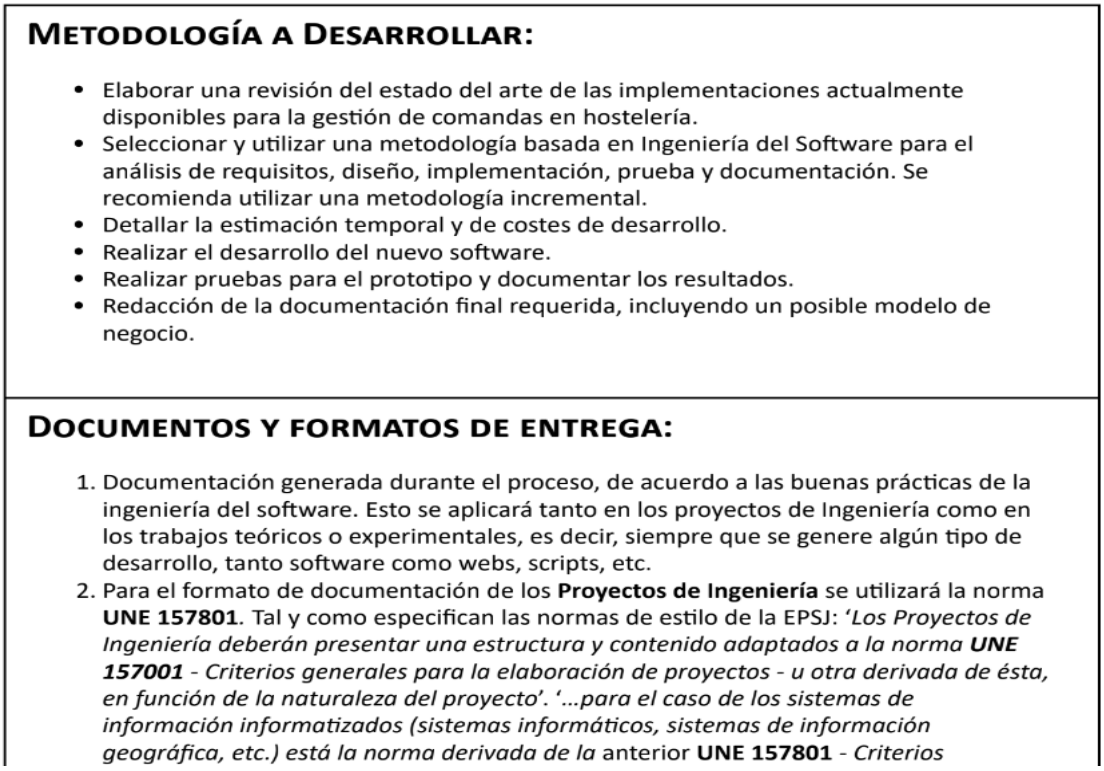
Nombre:
CARLOS JAVIER OGAYAR ANGUITA

Departamento:
Informática

A. Conocimiento:
LENGUAJES Y SISTEMAS INFORMÁTICOS

- [ ] Este TFG tiene un segundo tutor

Ilustración 6.2: Guía original del trabajo (PARTE 2)



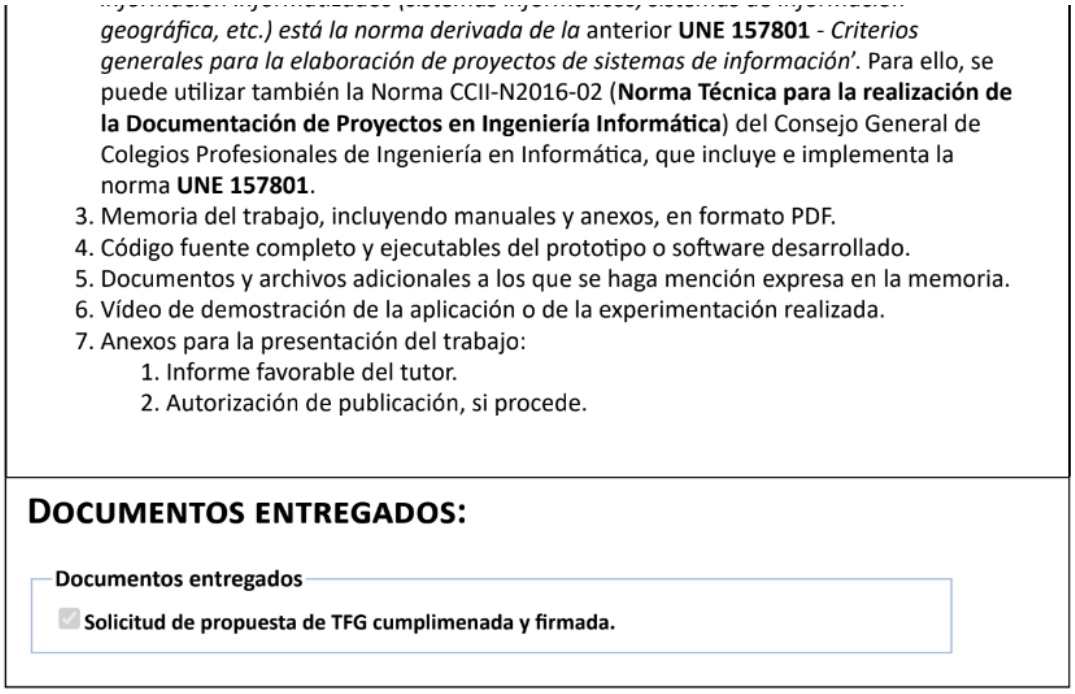
METODOLOGÍA A DESARROLLAR:

- Elaborar una revisión del estado del arte de las implementaciones actualmente disponibles para la gestión de comandas en hostelería.
- Seleccionar y utilizar una metodología basada en Ingeniería del Software para el análisis de requisitos, diseño, implementación, prueba y documentación. Se recomienda utilizar una metodología incremental.
- Detallar la estimación temporal y de costes de desarrollo.
- Realizar el desarrollo del nuevo software.
- Realizar pruebas para el prototipo y documentar los resultados.
- Redacción de la documentación final requerida, incluyendo un posible modelo de negocio.

DOCUMENTOS Y FORMATOS DE ENTREGA:

1. Documentación generada durante el proceso, de acuerdo a las buenas prácticas de la ingeniería del software. Esto se aplicará tanto en los proyectos de Ingeniería como en los trabajos teóricos o experimentales, es decir, siempre que se genere algún tipo de desarrollo, tanto software como webs, scripts, etc.
2. Para el formato de documentación de los **Proyectos de Ingeniería** se utilizará la norma **UNE 157801**. Tal y como especifican las normas de estilo de la EPSJ: '*Los Proyectos de Ingeniería deberán presentar una estructura y contenido adaptados a la norma **UNE 157001** - Criterios generales para la elaboración de proyectos - u otra derivada de ésta, en función de la naturaleza del proyecto*'. '*...para el caso de los sistemas de información informatizados (sistemas informáticos, sistemas de información geográfica, etc.) está la norma derivada de la* anterior **UNE 157801** - *Criterios*

Ilustración 6.3: Guía original del trabajo (PARTE 3)



geográfica, etc.) está la norma derivada de la anterior **UNE 157801** - *Criterios generales para la elaboración de proyectos de sistemas de información*'. Para ello, se puede utilizar también la Norma CCII-N2016-02 (**Norma Técnica para la realización de la Documentación de Proyectos en Ingeniería Informática**) del Consejo General de Colegios Profesionales de Ingeniería en Informática, que incluye e implementa la norma **UNE 157801**.

3. Memoria del trabajo, incluyendo manuales y anexos, en formato PDF.
4. Código fuente completo y ejecutables del prototipo o software desarrollado.
5. Documentos y archivos adicionales a los que se haga mención expresa en la memoria.
6. Vídeo de demostración de la aplicación o de la experimentación realizada.
7. Anexos para la presentación del trabajo:
    1. Informe favorable del tutor.
    2. Autorización de publicación, si procede.

DOCUMENTOS ENTREGADOS:

Documentos entregados

☑ **Solicitud de propuesta de TFG cumplimenada y firmada.**

Ilustración 6.4: Guía original del trabajo (PARTE 4)

6.2 Instalación y configuración del sistema

Para poder instalar y configurar el proyecto en otro equipo y ponerlo en funcionamiento, necesitamos tener lista las tecnologías usadas MEAN STACK.

Primero tenemos que tener listo la tecnología NodeJS para el funcionamiento del backend.

Para tenerlo instalado debemos descargarlo desde su página web e instalarlo.

Para comprobar que esté todo listo en el lado de NodeJS, se puede ejecutar en la terminal: **node -v.**

Por la parte del frontend necesitamos tener listo Angular, por lo que debemos instalarlo mediante el siguiente comando: **npm install -g @angular/cli**.

Una vez tenemos las tecnologías necesarias para el frontend y el backend, descargamos el proyecto.

Para la parte del backend, hacemos cd backend para ubicarnos sobre la carpeta del backend e instalamos todas las dependencias que tiene mediante: **npm install.**

Realizamos lo mismo con la parte del frontend, nos ubicamos sobre la carpeta de frontend y volvemos a ejecutar npm install para ejecutar todas las dependencias necesarias en el lado del frontend.

Una vez instaladas las dependencias tanto del backend como del frontend, ya simplemente lo que debemos hacer es ejecutar e iniciar los servicios.

Creamos dos terminales diferentes, una nos ubicamos en el frontend y la otra en el backend.

Sobre la del backend: **node server.js**

Sobre la del frontend: **ng serve -o**

6.3 Manuales de usuario

Para facilitar el uso de los usuarios para la aplicación web, se crea un manual de usuario para que tengan una guía sobre qué hacer dentro de la aplicación y cómo poder acceder a todas las funcionalidades que tiene.

Podemos diferenciar entre los dos tipos de usuario: Administrador o cliente.

- Si eres Administrador, al acceder mediante el login verás la siguiente interfaz:

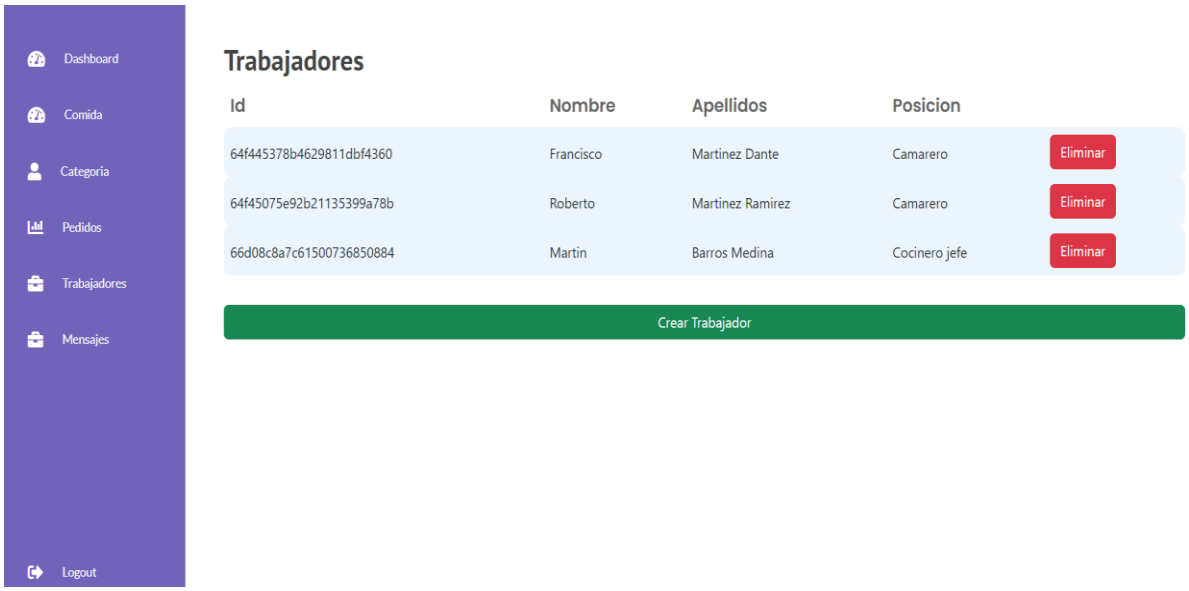


Ilustración 6.5: Interfaz Panel Admin

Mediante esta interfaz podrás navegar por el menú que es fundamental para este rol moviéndose entre los menús de Comida, Categorías, Pedidos, Trabajadores o Mensajes.

También puede hacer Logout para cerrar sesión y volver a la interfaz asociada al cliente.

-Si eres cliente, sin el rol de Admin, al crear una cuenta y hacer login en la aplicación, la primera pantalla que verás será la siguiente:

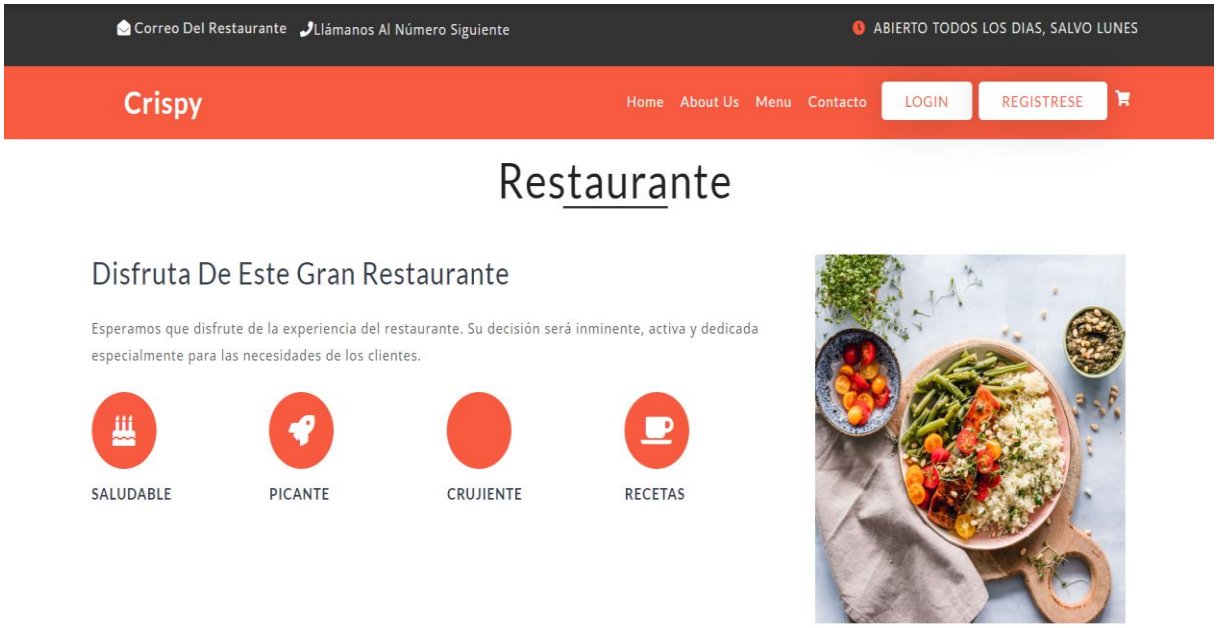


Ilustración 6.6: Interfaz Panel Cliente

Podrás navegar entre diferentes opciones como es el Carrito, el menú donde podrás pedir los diferentes platos, contacto donde podrás dejar un mensaje con lo que desees en forma de Queja o Sugerencia entre otros aspectos.

7 DEFINICIONES Y ABREVIATURAS

A continuación, se introducen las definiciones o términos que pueden ser más complicados de entender o que puede requerir algún tipo de aclaración adicional.

- **MEAN STACK**: Es un conjunto de tecnologías usadas para la creación e implementación de aplicaciones web. Viene dado por MongoDB (Base de datos), ExpressJS (entorno de trabajo para aplicaciones web para Node.js), Nodejs (Interacción con la parte del servidor) y Angular (Framework de desarrollo de aplicaciones).

- **Framework**: Conjunto de técnicas, prácticas desde una base que se utiliza para la creación de un proyecto.

- **Matriz DAFO**: Matriz utilizada en el ámbito empresarial para analizar diferentes variables de un entorno como puede ser creación de una empresa basada en un sector.

- **Postman**: Software utilizado para probar la consistencia y robustez del backend de una aplicación.

- **Hostelería**: Sector relacionado con el alojamiento o ligados con la alimentación para los clientes.

- **Comandas**: Las comandas son los encargos que un usuario realiza en un establecimiento.

- **Footer**: El footer es la parte inferior de una página web donde suele venir información del establecimiento como suele ser la calle, número de teléfono, entre otros aspectos.

- **Sidebar**: Es una barra lateral vertical que viene con diferentes opciones para la interacción del usuario con la aplicación web.

- **CRUD**: Siglas de Create/Read/Update/Delete, utiizado para hacer referencia a las funciones básicas en una BBDD.

- **HTTP**: Protocolo orientado a transacciones entre cliente y servidor.

- **Frontend**: Parte de una aplicación web que se encarga de la interacción con el usuario.

- **Backend**: Parte de una aplicación web que se encarga de la interacción con el servidor, todo lo relacionado con la Base de Datos y su ejecución.

- **TypeScript**: Es un lenguaje de programación utilizado para la implementación de la lógica del proyecto en la parte frontend. Angular usa este lenguaje.

- **JSON**: Formato de texto específico que contiene una serie de datos estructurados.

- **Navbar**: Son barras de navegación donde está compuesto por un menú para interactuar con los servicios que nos ofrece la aplicación.

- **Dependencias**: Son las tecnologías introducidas en el proyecto para el correcto funcionamiento de la aplicación.

- **LocalStorage**: Atributo de los navegadores que permiten guardar información de la aplicación en él.

- **BSON (JSON Binario)**: Serialización de documentos JSON en binario.

8 BIBLIOGRAFÍA

Marcia Ramos (2023). Mean Stack Explained. URL. https://kinsta.com/blog/mean-stack/

Chema Dyaz (2023) Problemas de los restaurantes y soluciones. URL. https://avocaty.io/problemas-de-los-restaurantes/

Yanira Muradas (2019) Qué es Postman y primeros pasos. URL. https://openwebinars.net/blog/que-es-postman/

Maria Cappola (2023) ¿Qué es Angular? Características y ventajas. URL. https://blog.hubspot.es/website/que-es-angular

Carlos Azaustre (2014) Desarrollo web ágil con Angular JS. URL. https://www.amazon.es/Desarrollo-ágil-Angular-js-Carlos-Azaustre/dp/B0BQ9J8ZR8/ref=sr_1_2?__mk_es_ES=ÅMÅŽÕÑ&crid=3O1PNNJNMAC6B&dib=eyJ2IjoiMSJ9.6fGHEeo8BHF3gvtdxmBr5M4IV4mte6DW7TRXkQcT9N5Yy-qC75vR_JXNYOaZPlLWHLw3hwRt8Em-tZqze_807RNwO5z8LSDt1bHmLL5dt5cnI6RSlmHt2ZYrtfuPXVqdweXeOSSCqcQLh6IV3cEMAapEjGUosPmgYSM32PuatrCJZ_ZotSvOsAC1np8w3DUVCF89qYgza3FEyiMyCl6R8kZiyibEJkdOGDmGkfspZCIHmnbPK49BVkMZKT-Mtvacl3t4imKnYfa-c_2WeN04STjJ4SFxL7j5abPflYU5I88.mpHM1ITOKAzeSvnFxSWjx01Drtp-HBUXehWjC0xrAeU&dib_tag=se&keywords=libro+angular&qid=1725813294&sprefix=libro+angula%2Caps%2C192&sr=8-2

Gustavo Morales (2020) Creando API con Node.js, express y MongoDB. URL. https://www.amazon.es/Creando-API-Node-js-express-MongoDB/dp/B08H6RVTDP/ref=sr_1_5?__mk_es_ES=ÅMÅŽÕÑ&crid=1JN1I0WVYMZGH&dib=eyJ2IjoiMSJ9.g9nyaIin0MYWnywkQFL0U2FD8SdSmjxzFcFXsflVdi_CwVLHixzS2hpvyh3mDDvgREXY-lrVchk87ss9ysgMEylaskmKc5DJjzgiWKDkWPigKYPSJlipwuQ0fKz9WmA5kPIcRBIdJOdu73ZzrRpeRWUEfRRp1JT0BMR7Ip7WKjMUuBLbebDtsxk79nlJYO4oNxY-

K54KmcJv-2TrJsQ5cfypUppUoMPcFHbo7ZO2-aSMRV63JXh3AbCPZbE_RM4w-gebI5eT0If3H8cPisFZU1XTqm8SKq77VDVBYOKfJO0.RrhhblcvdoBQfcey-QxOqziD2XxsNZcY1FU-cQX7RJQ&dib_tag=se&keywords=LIBRO+NODEJS&qid=1725813364&sprefix=libro+nodejs%2Caps%2C188&sr=8-5

Fernando Herrera (2023) Curso Udemy: Legacy - Node: De cero a experto. URL. https://www.udemy.com/course/node-de-cero-a-experto/

Fernando Herrera (2023) Curso Udemy: Angular: De cero a experto. URL. https://www.udemy.com/course/angular-fernando-herrera/

Fernando Herrera (2024) Curso Udemy: Angular Avanzado: Lleva tus bases al siguiente nivel – MEAN. URL. https://www.udemy.com/course/angular-avanzado-fernando-herrera/

Camilo Clavijo (2024) Qué es un modelo de negocios: definición, tipos y cómo crearlo. URL. https://blog.hubspot.es/sales/modelo-negocio

Camilo Clavijo (2024) Cómo hacer un análisis DAFO (con ejemplos y plantilla). URL. https://blog.hubspot.es/sales/como-hacer-analisis-dafo

Shelley Pursell (2024) Qué es un plan de marketing y cómo crearlo (incluye plantillas). URL. https://blog.hubspot.es/marketing/generador-plan-de-marketing#:~:text=Un%20plan%20de%20marketing%20es%20un%20documento%20que%20describe%20las,de%20medios%20y%20el%20presupuesto.

Tipos de sociedades: comparativa de formas jurídicas (2024) URL. https://www.infoautonomos.com/tipos-de-sociedades/crear-una-sociedad-comparativa-de-formas-juridicas/

Sweet Alert 2 – Tutorial con ejemplos (2024) URL. https://parzibyte.me/blog/2019/12/16/sweet-alert-2-tutorial-ejemplos/

ANÁLISIS DEL ENTORNO DE UNA EMPRESA, POR QUÉ ES IMPORTANTE (2023) URL. https://www.mastermarketing-valencia.com/ventas-y-gestion-comercial/blog/entorno-de-una-empresa/