Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

Sistema de gestión de ocupación y comandas en bares y restaurantes mediante una webapp para clientes proactivos

---

Alumno: David Curieses Fernández

Tutor: Yania Crespo González-Carvajal

A mi familia y amigos, que siempre han estado apoyando y animando.

Agradecimientos

A mi familia, que siempre me ha animado y apoyado en los momentos buenos y malos.

A mis amigos, que también han estado ahí para apoyarme cuando era necesario.

A mi tutora Yania, que siempre ha estado pendiente a todas las dudas y ha hecho que este proyecto sea muy útil para mi desarrollo como profesional.

Muchas gracias a todos.

Resumen

Se trata de un proyecto para establecimientos de restauración con un objetivo doble, ya que por una parte se facilitará el proceso de petición de comandas, permitiendo a clientes proactivos que puedan realizar dicha función por sí mismos en cuanto conozcan la funcionalidad que tendrán a su disposición. Por otro lado, se gestionará la ocupación de espacios, lo cual permitirá, tanto a empleados como a clientes, tener una información en tiempo real sobre el grado de ocupación de los establecimientos registrados en la plataforma. De esto podrá beneficiarse el potencial cliente porque saber el dato de la ocupación de un establecimiento antes de acercarse a este, le puede permitir planificarse mejor.

El sistema contará con un apartado para la gestión de cada establecimiento por parte de los propietarios del mismo, donde podrán modificar tanto los datos informativos de este, como los empleados, mesas y productos que forman parte de la carta del local. También permitirá a los trabajadores de cada local gestionar las mesas y comandas activas de cada una de estas del lugar donde trabajan.

Finalmente, el sistema contará con una supervisión de un administrador, encargado de validar las peticiones de registro tanto de propietarios como de los establecimientos de estos, si es que su petición de registro ha sido aceptada.

El proyecto se ha desarrollado como una webapp utilizando el framework Angular y HTML5 para el frontend y el framework SpringBoot en el backend, siguiendo las guías del marco de trabajo ágil Scrum.

Abstract

This is a project for catering establishments with a twofold objetive, on the one hand, the process of requesting orders will be facilitated, allowing proactive customers to perform this function themselves as soon as they are aware of the functionality they will have at their disposal. On the other hand, space occupancy will be managed, which will allow, both employees and customers, to have a real-time information about the occupancy rate of the establishments registered on the platform. Potential customers will be able to benefit from this beacuse knowing the occupancy data of an establishment before approaching it will allow them to plan better.

The system will have a section for the management of each establishments by its owners, where they will be able to modify the establishment's informative data, as well as the employees, tables and products that form part of the establishment's menu. It will also allow the employees of each establsihment to manage the active orders for each table in the establishment where they work.

Finally, the system will be supervised by an administrador, who will be in charge of validating the registration requests of both owners and their establishments, if their registration request has been accepted.

This project has been developed as a webapp using the Angular framework and HTML5 for the frontend and the SpringBoot framwork for the backend, following the guidelines of the agile framework Scrum.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

En el año 2020, las ventas en el sector de la restauración en España superaron los 21.800 millones de euros, contando con más de 270.000 establecimientos y empleando a más de un millón de trabajadores durante todo el año 2021. En España existen más de 250.000 empresas dedicadas a la restauración, lo que provoca que los ciudadanos tengan un amplio abanico de posibilidades para elegir un lugar a la hora de salir a comer, cenar o tomar algo. El gasto per cápita anual en alimentos y bebidas fuera del hogar en el año 2020 superó los 1000 euros [95].

Dados estos datos, se puede observar que el sector de la restauración es muy importante en España, por lo que el principal objetivo de este proyecto es elaborar una plataforma que pueda ayudar a los propietarios de este sector. Se quiere brindar una funcionalidad amplia al propietario, para que puedan gestionar de la mejor forma posible sus establecimientos dentro de la plataforma, brindar de forma intuitiva y sencilla el manejo de las comandas activas para aquellos trabajadores de un local y crear un proceso sencillo e intuitivo para que los clientes de un establecimiento puedan realizar las comandas en cada uno de ellos.

Con esta plataforma se quiere agilizar el proceso de realización de una comanda, otorgando todo el proceso al cliente, y desvinculando al trabajador de las tareas de anotar los pedidos. También se aporta información sobre la ocupación de los locales, por lo que los clientes podrán consultar dichos datos antes de ir a un establecimiento, evitando así que un cliente pueda acercarse a un establecimiento lleno.

1.2. Motivación

La idea principal de este proyecto surge tras observar la situación de la restauración tras la pandemia por el Covid-19, ya que la mayoría de establecimientos relacionados con la

restauración han sustituido las clásicas cartas en formato físico, por cartas online. También un aspecto a tener en cuenta para la elaboración de dicha idea principal fue la incorporación, en ciertos establecimientos, como es el caso de la franquicia de comida rápida McDonald's, de pantallas o dispositivos desde los cuales puedes realizar un pedido sin necesitar la intervención de un empleado del local para llevarlo a cabo, consiguiendo así que se agilice el proceso de realización de comandas y elaboración de las mismas.

Esta idea también se funda porque en ciertas ocasiones si un establecimiento sin dichos dispositivos para pedir sin necesitar a un trabajador está muy concurrido, existe la posibilidad de que haya una gran demora de tiempo entre que un cliente ocupa un lugar y decide qué pedir, hasta que el trabajador del local lo ve y se acerca a preguntar por la comanda, por lo que si ese tipo de establecimientos contasen con una plataforma en la cual los clientes puedan realizar dichas comandas, se reduciría esta demora de tiempo, ahorrando a los trabajadores del local ciertas tareas para que puedan ser más eficientes en otras.

Tras observar algunas otras plataformas para realizar pedidos a domicilio, donde se alojan diferentes establecimientos, se forjó la idea final, crear una plataforma que englobase los dos aspectos comentados, la posibilidad de consultar la carta de diferentes establecimientos y poder realizar pedidos en estos, aunque centrándonos exclusivamente en los pedidos de forma presencial, añadiendo a dicha idea la posibilidad de que el cliente que utiliza la plataforma tenga a su vez, y en tiempo real, el estado de ocupación de cada local. Esto ayuda al cliente a decidir previamente entre varios locales por su grado de ocupación o incluso a modificar la hora en la que se iba a acercar al establecimiento, mejorando la planificación de sus desplazamientos.

1.3. Plataformas similares

El desarrollo de esta plataforma se ha nutrido de otras plataformas ya existentes, de las cuales se han extraído ciertas funcionalidades para así construir una única plataforma que contenga en conjunto todas ellas.

- **Plataformas de entrega de alimentos a domicilio:** Este proyecto se ha nutrido de este tipo de plataformas, como es el caso de "Uber Eats", "JustEat", "Glovo", etc, ya que permite al usuario contemplar los establecimientos registrados en la plataforma y sus cartas, para así saber qué productos, y a qué precio, están disponibles en cada uno de estos establecimientos. A diferencia de estas plataformas mencionadas, el desarrollo de este proyecto se centrará en comandas presenciales en el local, y no a domicilio, aunque en un futuro podría ser un aspecto a tener en cuenta para incorporar al proyecto.
- **Establecimientos con cartas vía dispositivos inteligentes o códigos QR**: Partiendo de la base de que nos centramos en que los clientes van a realizar las comandas de forma presencial, muchos establecimientos cuentan con códigos QR en las mesas para acceder a una web con la carta del local, o incluso, de pantallas inteligentes, en caso de franquicias como "McDonald's" o de dispositivos móviles, como tablets, anclados a las mesas, donde poder consultar la carta del local y realizar un pedido, por lo

que incorporando esta idea a nuestra plataforma, el cliente podrá consultar la carta del establecimiento en cualquier momento y, desde su mismo dispositivo móvil, realizar una comanda introduciendo un código identificativo de una mesa del local en el que esta presencialmente.

- **Maybein**: Maybein [75] es una plataforma centrada en la gestión de reservar, con la intención de eliminar o paliar las cancelaciones, optimizando la rentabilidad de los restaurantes. Maybein gestiona la demanda y oferta de mesas, de tal forma que cuando una mesa queda vacía se notifica al usuario de que puede ser reservada. Nuestro proyecto se nutre de esta idea de gestión de mesas de una forma diferente, ya que, como aun no se desarrollará el sistema de reservas, damos al usuario información del nivel de ocupación de las mesas en tiempo real, permitiendo a este que pueda organizarse mejor y elegir cuando acercarse a un establecimiento.

1.4. Objetivos

1.4.1. Objetivos de desarrollo

El principal objetivo que se persigue con el desarrollo de este proyecto es crear una plataforma web, que disponga principalmente con esta funcionalidad:

- Gestión de establecimientos para los propietarios de estos, incluyendo mesas, empleados y la carta de productos de cada uno de ellos.
- Gestión de peticiones de registro de propietarios y establecimientos.
- Gestión de las mesas de un establecimiento y de las comandas de cada mesa por los trabajadores del local.
- Posibilidad de consultar el nivel de ocupación de un establecimiento en tiempo real.
- Posibilidad para los clientes de realizar ellos mismos las comandas en los establecimientos.

Otra de las muchas características que podrían ser interesantes de incluir en un futuro pero, debido al alcance del proyecto y que nos hemos centrado en el desarrollo de un producto mínimo viable, no se podrá incluir aunque formaba parte de la idea inicial:

- Posibilidad de reservar una mesa.

1.4.2. Objetivos personales

Para llevar a cabo este proyecto, al igual que he definido unos objetivos de desarrollo, también he definido una serie de objetivos personales:

- El principal objetivo de este proyecto ha sido comprender y descubrir lo que implica desarrollar un proyecto software "real" partiendo desde una idea amplia, y pasar por todas las etapas de análisis, diseño y desarrollo de cada aspecto del proyecto, hasta el despliegue del mismo.
- Poner en práctica y entender mejor el marco de trabajo Scrum elegido para llevar a cabo el proyecto.
- Mejorar mis capacidades técnicas en las diferentes tecnologías utilizadas en el desarrollo del proyecto, buscando información y profundizando en los casos necesarios.

1.5. Estructura de la memoria

Este documento se estructura por capítulos de la siguiente forma:

Capítulo 1 Introducción: En el este primer capítulo del documento se describe el origen de la idea, la motivación que hay para llevar a cabo el proyecto y los objetivos que se pretenden alcanzar.

Capítulo 2 Requisitos y planificación: En este capítulo se presenta el método de planificación y seguimiento escogido y la adecuación de este al proyecto que se va a llevar a cabo, se plantean una serie de planes de riesgos y los presupuestos asociados al proyecto. También se describen los requisitos en forma de historias de usuario.

Capítulo 3 Análisis: En este capítulo se presentan el modelado conceptual de la plataforma, diagramas de actividad de dos de las actividades más importantes del sistema y dos máquinas de estado para definir los estados por los que pasan ciertos objetos debido a la interacción de los usuarios con la plataforma.

Capítulo 4 Tecnologías utilizadas: En este capítulo explican las diferentes tecnologías, herramientas, lenguajes de programación y plataformas utilizadas durante el proceso de elaboración del proyecto, separando entre aquellas utilizadas en la gestión, el desarrollo y el despliegue del proyecto. Para cada uno de los apartados se describirán qué son y en qué parte del proyecto se han utilizado.

Capítulo 5 Diseño: En este capítulo se describirán las decisiones de diseño tomadas, y los diagramas y modelos elaborados para representar el sistema creado. Se mostrará el patrón de arquitectura elegido, los componentes utilizados, el diseño de datos del sistema, el despliegue de la aplicación y los diferentes bocetos de la interfaz de usuario.

Capítulo 6 Implementación y pruebas: En este capítulo se presentan algunos aspectos relacionados con la implementación del sistema y las distintas pruebas llevadas a cabo para la comprobación del buen funcionamiento de la plataforma.

Capítulo 7 Seguimiento del proyecto: En este capítulo se documenta cómo ha avanzado el proyecto durante su desarrollo, se tratan cuestiones relativas a problemas encontrados y cómo se han solucionado, se discute sobre diferentes aspectos y se documenta la razón

por la cual se ha escogido una u otra opción, según fuera lo más oportuno. También se detallan las historias de usuario a realizar en cada sprint y los puntos asignados a cada una de ellas, y se describe tanto al inicio como al final de cada sprint, el estado de las historias abordadas en este.

Capítulo 8 Conclusiones: En este último capítulo se exponen las conclusiones finales de la realización del proyecto, al igual que algunas ideas de posibles líneas futuras de desarrollo, con ejemplos de aspectos que se podrían implementar o mejorar de la plataforma desarrollada.

Anexo A Manuales: Incluye algunos manuales de interés, como la guía de despliegue o el manual de usuario.

Anexo B Resumen de enlaces adicionales: Incluye los enlaces de interés sobre el proyecto, como, por ejemplo, el repositorio en el que se encuentra el código fuente del sistema. También incluye el enlace a la página web donde se encuentra desplegada la aplicación.

Capítulo 2

Requisitos y Planificación

2.1. Scrum

2.1.1. Definición Scrum y pilares

El marco de trabajo seleccionado para la realización de este proyecto es Scrum [101]. Scrum se enmarca en los procesos ágiles de desarrollo de proyectos software, los cuales intentan reducir la complejidad del desarrollo, mediante la mejora de la comunicación y colaboración entre los integrantes del equipo de desarrollo. Por lo tanto, y como se declaró en el Manifiesto Ágil [60] en el año 2001, Scrum se basa en 4 valores fundamentales:

- Los individuos e interacciones por encima de procesos y herramientas.
- El software de trabajo está por encima de documentación completa.
- La colaboración del cliente por encima de negociación de contrato.
- La respuesta al cambio por encima de seguir un plan.

Scrum es un marco de referencia donde, con equipos pequeños, se desarrollan, entregan y mantienen productos complejos, siguiendo un enfoque de desarrollo iterativo e incremental en bloques de tiempo cortos y fijos.

Scrum fue diseñado principalmente para el desarrollo de nuevos productos software para un mercado competitivo, siguiendo la filosofía de que sacar un producto antes que tus competidores al mercado, puede ser más importante que tener una amplia gama de características o funcionalidades no esenciales.

Scrum se sustenta sobre tres pilares principales, **inspección**, **transparencia** y **adaptación** [101]:

- **Inspección**: Llevar a cabo una inspección del progreso del proyecto, por todos los integrantes del equipo, para detectar y corregir variaciones inesperadas que puedan haber surgido. Estas inspecciones deben realizarse frecuentemente, pero sin interferir en el ritmo de trabajo normal.
- **Transparencia**: Los responsables del trabajo a realizar deben conocer aquellos aspectos significativos para la realización del proyecto, se requiere establecer un estándar común para que todos entiendan los aspectos de la misma manera.
- **Adaptación**: Si se detecta que uno o varios aspectos del trabajo que se está realizando se desvían de los limites aceptables establecidos, deben ajustarse lo antes posible para evitar desviaciones mayores.

2.1.2. Componentes de Scrum

Dentro del marco de trabajo Scrum (Figura 2.1) nos encontramos tres componentes esenciales para su desarrollo: eventos, roles y artefactos:

Eventos de los proyectos Scrum

En los proyectos Scrum existen diversos eventos predefinidos [101] que se llevan a cabo a lo largo del ciclo de vida del proyecto, todos son bloques de tiempo, es decir, tienen una duración máxima. Una vez que se inicia cada Sprint, la duración de este es fija, no se puede aumentar ni disminuir. Todos estos eventos se establecen con el objetivo de mejorar la comunicación entre los integrantes del equipo Scrum y reducir el número de reuniones no planificadas.

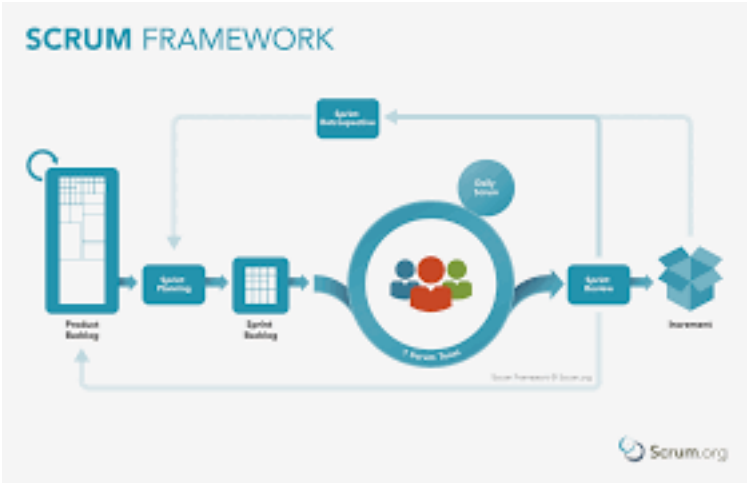


Figura 2.1: Marco de trabajo de los proyectos Scrum [99].

- **Sprint**: El Sprint se considera el corazón de Scrum, es un bloque de tiempo de entre 1 y 4 semanas durante el cual se realiza el incremento de producto "Terminado" utilizable.

Cada Sprint se inicia inmediatamente tras la finalización del anterior y consisten en un Sprint Planning, un Sprint Review, un Sprint Retrospective, el trabajo de desarrollo del producto y una serie de Daily Scrums, cuya cantidad depende de la duración del Sprint.

- **Sprint Planning**: Al comenzar un Sprint se lleva a cabo el Sprint Planning, donde se seleccionará el trabajo a llevar a cabo durante la duración de este, siendo el evento que da comienzo a un Sprint. En este participan todos los miembros del equipo y se seleccionan del Product Backlog los elementos a desarrollar que darán lugar al incremento final del Sprint, dando como resultado el Sprint Backlog.
- **Sprint Goal**: El Sprint Goal es una meta establecida para el Sprint que puede lograrse mediante la implementación de las historias de usuario seleccionadas para llevar a cabo en el Sprint Backlog.
- **Daily Scrum**: El Daily Scrum es una reunión realizada diariamente por el equipo Scrum de una duración de unos 15 minutos, en la cual cada componente del equipo informa de la situación de sus procesos para establecer una sincronización entre todos los integrantes y fabricar un plan para las siguientes 24 horas. Se realizará todos los días a la misma hora y lugar, para así reducir la complejidad y tendrá como objetivo mejorar la comunicación y solventar los diferentes impedimentos que ocurren a cada integrante, mediante la exposición de estos problemas individuales al resto del equipo.
- **Sprint Review**: Este evento se lleva a cabo al final de cada Sprint, y en él se inspecciona el incremento resultante y se adapta, si es necesario, el Product Backlog. En esta reunión participan tanto el equipo Scrum como los stakeholders y se discute el Sprint finalizado y los aspectos que se podrían tener en cuenta en el siguiente.
- **Sprint Retrospective**: Es el evento que da por concluido el Sprint y en él se lleva a cabo una inspección del Sprint finalizado para obtener aprendizajes sobre aspectos que pueden ser tenidos en cuenta en futuros Sprints, también es una oportunidad de evaluar el trabajo en conjunto del equipo en el Sprint.

Roles en los proyectos Scrum

Los equipos Scrum están formados por un Product Owner, el equipo de desarrollo y un Scrum Master, además son autoorganizados y multifuncionales [101]:

- **Product Owner**: El Product Owner es propietario de la iniciativa y el responsable de maximizar el valor del producto. Se encarga de gestionar el Product Backlog y suele actuar como representante del cliente dentro del equipo Scrum.
- **Equipo de desarrollo**: El equipo de desarrollo, de entre 3 a 9 integrantes, consiste en los profesionales que realizan el trabajo necesario para entregar un incremento de producto "Terminado", que pueda ser liberado al mercado, siendo los únicos participantes en la creación de este.
- **Scrum Master**: El Scrum Master es el responsable de asegurar que el ritmo de un proyecto Scrum se realice correctamente.

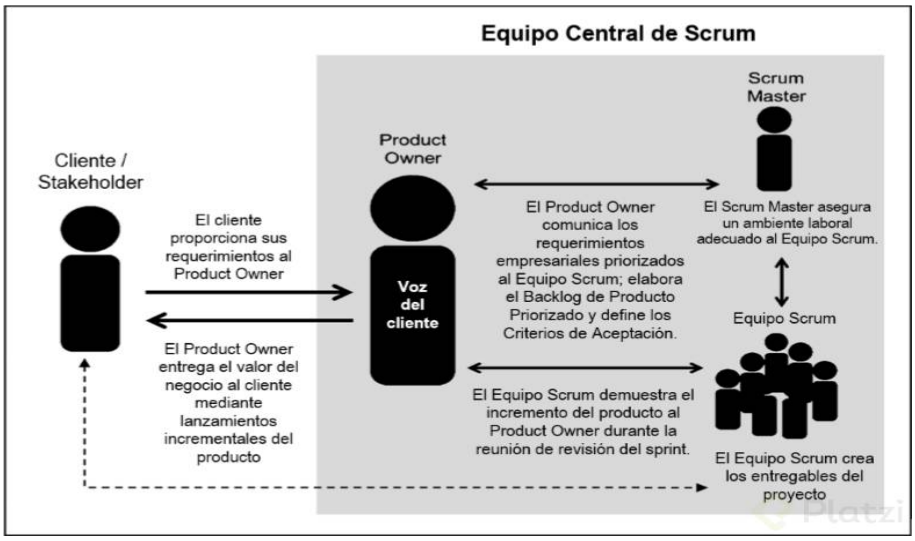


Figura 2.2: Comunicación de los diferentes roles dentro de un proyecto Scrum [29].

La comunicación entre los diferentes roles dentro de un proyecto Scrum se pueden observar en la Figura 2.2.

Artefactos de los proyectos Scrum

Los artefactos de Scrum [101] representan trabajo o valores de diferentes formas que son útiles para proporcionar una máxima transparencia de la información clave, necesaria para asegurar el entendimiento común de todos los aspectos del artefacto entre los componentes del equipo, y oportunidades para la inspección y adaptación.

- **Product Backlog**: El Product Backlog es una lista ordenada de todos los elementos que podrían ser necesarios en el producto a desarrollar, siendo esta la única fuente de requisitos para cualquier cambio que quiera realizarse sobre el producto. Registra todas las características, funcionalidades, requisitos, mejoras y correcciones que pueden constituir posibles cambios futuros a realizar sobre el producto.
  El encargado del Product Backlog es el Product Owner.
- **Sprint Backlog**: El Sprint Backlog es un conjunto de elementos, historias de usuario, seleccionados del Product Backlog que conforman el Sprint a llevar a cabo. El Sprint Backlog es responsabilidad del equipo de desarrollo y hace visible todo el trabajo que el equipo identifica como indispensable para alcanzar el Sprint Goal. Existe suficiente nivel de detalle como para que el progreso se pueda controlar mediante las Daily Scrums.
- **Incremento**: Es el resultado del trabajo realizado a lo largo de un Sprint, que, acumulado con el resto de los incrementos terminados, va proporcionando el objetivo actual del proyecto, el Product Goal. El incremento debe estar en condiciones de poder ser utilizado sin importar si el Product Owner decide liberarlo o no.

2.2. Adaptación de Scrum al proyecto

Como surge de una idea, los requisitos no tienen por qué estar muy bien definidos desde un inicio, por lo que Scrum es un buen marco de trabajo para realizar el proyecto, debido a su flexibilidad respecto a los posibles cambios que puedan producirse. También se realizará un seguimiento semanal por parte de la tutora, asegurando así retroalimentación que permitirá la mejora durante el desarrollo total de la plataforma.

Sobre los roles mencionados anteriormente también se realizará un ajuste para este proyecto. Solo habrá involucrados dos miembros en el desarrollo del proyecto, estudiante y tutora, por lo que los roles de equipo desarrollador y Product Owner, por ser una idea propia, serán del estudiante, mientras que el Scrum Master será la tutora.

Se establecerá una duración de sprint de 2 semanas, realizándose al final de cada uno de estos las pertinentes Sprint Review y Sprint Retrospective del sprint actual, y realizando posteriormente el Sprint Planning del siguiente sprint que se desarrollará, por lo que el final de cada sprint coincide con el inicio del siguiente.

Para este proyecto se cambiarán las Daily Scrum por Weekly Scrum, realizando estas, en vez de a diario, semanalmente cada miércoles, coincidiendo en aquellas semanas en las que no empieza ni finaliza un sprint. Es decir, el miércoles de cada semana, durante el desarrollo del proyecto, se realizará una reunión entre el estudiante y la tutora con un propósito diferente, dependiendo del punto temporal del sprint en el que se encuentre el desarrollo.

Por último, con la estimación de la duración en horas del TFG, establecida en aproximadamente 300 horas, y la estimación del número de sprints a realizar para la finalización del proyecto (unos 8 sprints) obtenemos unas 36 horas de trabajo en cada sprint, resultando en unas 18 horas de trabajo semanal. A mayores se establecerán 2 sprints extra de refuerzo, para así ofrecer un periodo de margen en caso de que los 8 sprints no hayan sido suficientes para la realización del proyecto, y para hacer retoques finales tanto en documentación como en el código, modelos, etc., que forman el material entregado.

En la Tabla 2.1 se puede observar una planificación y calendarización inicial de los sprints y eventos, en las diferentes fechas donde se realizarían, para el transcurso del proyecto.

2.3. Análisis de riesgos

A la hora de realizar un plan de riesgos para un proyecto, existe un componente de incertidumbre que hay que gestionar, ya que nos basamos en suposiciones y podría darse el caso de que resulten incorrectas.

Un riesgo es un evento o condición incierta que, si ocurre, tiene un efecto positivo o negativo en los objetivos de un proyecto y hacen referencia a eventos futuros que tienen una causa y un efecto relacionados.

Nº Sprint/Evento	Inicio	Final	Anotaciones adicionales
Sprint 0	09/02/2022	16/02/2022	
Sprint Review, Sprint Retrospective & Sprint Planning	16/02/2022		
Sprint 1	16/02/2022	02/03/2022	
Scrum Weekly	23/02/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	02/03/2022		
Sprint 2	02/03/2022	16/03/2022	
Scrum Weekly	09/03/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	16/03/2022		
Sprint 3	16/03/2022	30/04/2022	
Scrum Weekly	23/03/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	30/04/2022		
Sprint 4	30/04/2022	13/04/2022	
Scrum Weekly	06/04/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	13/04/2022		
-	13/04/2022	20/04/2022	Periodo de vacaciones de Semana Santa
Sprint 5	20/04/2022	04/05/2022	
Scrum Weekly	27/04/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	04/05/2022		
Sprint 6	04/05/2022	18/05/2022	
Scrum Weekly	11/05/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	18/05/2022		
Sprint 7	18/05/2022	01/06/2022	
Scrum Weekly	25/05/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	01/06/2022		
Sprint 8	01/06/2022	15/06/2022	
Scrum Weekly	08/06/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	15/06/2022		
Sprint Refuerzo 1	15/06/2022	29/06/2022	
Scrum Weekly	22/06/2022		
Límite solicitud defensa convocatoria ordinaria	24/06/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	29/06/2022		
Sprint Refuerzo 2	29/06/2022	13/07/2022	
Scrum Weekly	06/07/2022		
Sprint Review, Sprint Retrospective & Sprint Planning	13/07/2022		
Límite solicitud defensa convocatoria extraordinaria	15/07/2022		

Tabla 2.1: Planificación inicial de sprints

Para evitar que estos se materialicen, o, en caso de que estén presentes, sean un problema mayor del que podrían ser, hay que realizar un análisis e identificación de riesgos detallado.

Para cada riesgo se declararan, además del título y una breve descripción, cinco campos para su identificación y análisis:

- **Categoría**: Categoría sobre la que está enmarcado el riesgo [23].
- **Probabilidad**: Posibilidad de que el riesgo se materialice, se asignarán un valor entre los siguientes: baja, media, alta.
- **Impacto**: Nivel del efecto que produce un riesgo al materializarse, se asignará un valor entre los siguientes: bajo, medio, alto.
- **Plan de mitigación**: Serie de medidas a llevar a cabo para que un riesgo no se materialice en la medida de lo posible.
- **Plan de contingencia**: Serie de medidas a llevar a cabo cuando un riesgo se materializa, con la finalidad de minimizar lo más posible su impacto.

Los riesgos expuestos podrán ser separados en dos grupos, **riesgos generales** (Tablas 2.2 a 2.7), posibilidad de ocurrencia en cualquier contexto de trabajo y **riesgos particulares o específicos** (Tablas 2.8 a 2.10), aquellos más asociados a este proyecto en específico.

Riesgo R01	
Titulo	Enfermedad del desarrollador.
Descripción	El desarrollador enferma, lo que le impide dedicar tiempo a trabajar en el proyecto hasta que se recupera.
Categoría	Equipo de trabajo
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Llevar una vida saludable. ▪ Evitar o tener precaución al realizar actividades que puedan suponer un riesgo para la salud del desarrollador.
Plan de contingencia	▪ Tratar de recuperar las horas en los días posteriores desde la recuperación. ▪ Atrasar las tareas del sprint actual al siguiente, ▪ Hacer uso de los sprints de refuerzo. ▪ Recortar en lo posible el alcance del proyecto.

Tabla 2.2: Riesgo R01: Ausencia temporal del estudiante por enfermedad

Riesgo R02	
Titulo	Enfermedad de la tutora.
Descripción	La tutora enferma, lo que impide poder realizar el seguimiento del proyecto con normalidad.
Categoría	Equipo de trabajo
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Realizar las reuniones previstas con normalidad y mantener contacto para saber si el riesgo se materializa en otro momento en el que no haya reunión cercana.
Plan de contingencia	▪ Continuar el trabajo previsto con normalidad sin realizar reuniones hasta que la salud mejore. ▪ Establecer otro medio de comunicación para que se puedan resolver dudas puntuales sin necesidad de una reunión larga,

Tabla 2.3: Riesgo R02: Ausencia temporal de la tutora por enfermedad

Riesgo R03	
Titulo	Error en las estimaciones.
Descripción	El esfuerzo es mayor que el estimado (por líneas de código, número de puntos función, módulos, etc.), lo que provoca un retraso en ciertas actividades.
Categoría	Planificación y control
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Realizar planificaciones concienzudas procurando dar la importancia óptima a cada tarea.
Plan de contingencia	▪ Realizar las tareas posibles del sprint y posponer las menos esenciales a sprints posteriores. ▪ Si ocurre en las fechas finales, seleccionar las tareas restantes menos esenciales y no realizarlas.

Tabla 2.4: Riesgo R03: Fallo en las estimaciones.

Riesgo R04	
Titulo	Agravamiento de la situación médica actual por Covid-19.
Descripción	La situación médica actual empeora por el virus Covid-19, repercutiendo en ciertos aspectos del proyecto.
Categoría	Causa de efecto mayor
Probabilidad	Baja
Impacto	Bajo
Plan de mitigación	▪ Realizar la mayor cantidad de trabajo posible de forma telemática. ▪ Familiarizarse con las herramientas de comunicación telemática.
Plan de contingencia	▪ Establecer métodos de comunicación telemática para realizar el seguimiento del proyecto.

Tabla 2.5: Riesgo R04: Agravamiento de la situación médica por Covid-19.

Riesgo R05	
Titulo	Falta de formación en las herramientas utilizadas.
Descripción	El desarrollador necesita familiarizarse con las herramientas o el entorno de trabajo para realizar las tareas del proyecto.
Categoría	Complejidad tecnológica.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	▪ Familiarizarse con los aspectos posibles lo antes posible (idealmente en el sprint 0).
Plan de contingencia	▪ Uso de la documentación oficial de la herramienta o entorno para el manejo. ▪ Buscar una alternativa factible conocida para usar en vez de la herramienta o entorno desconocido.

Tabla 2.6: Riesgo R05: Falta de formación en las herramientas.

Riesgo R06	
Titulo	Fallos técnicos con el equipo de trabajo.
Descripción	Los equipos técnicos con los que trabaja el desarrollador del proyecto sufren algún fallo técnico lo que implica no poder trabajar con normalidad con ellos.
Categoría	Herramientas.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	▪ Realizar copias de seguridad del proyecto frecuentemente en discos duros externos o repositorios online. ▪ Trabajar, dentro de lo posible, con herramientas online.
Plan de contingencia	▪ Recuperar las copias de seguridad desde otro dispositivo sobre el cual continuar trabajando. ▪ Buscar métodos de recuperación de archivos en los diferentes programas para intentar rescatar la mayor cantidad de información posible.

Tabla 2.7: Riesgo R06: Fallos técnicos con el equipo de trabajo.

Riesgo R07	
Titulo	Cambios en los requisitos.
Descripción	El cliente (o estudiante, al hacer el papel de Product Owner) decide cambiar los requisitos del proyecto, alterando así funcionalidades realizadas o propuestas, o incluso añadiendo nuevas.
Categoría	Planificación y control.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	▪ Aclarar de la manera más precisa posible la funcionalidad requerida para el proyecto. ▪ Hacer entrega de incrementos que puedan ser validados para evaluar la trayectoria del producto.
Plan de contingencia	▪ Evaluar el cambio propuesto para optar por llevarlo a cabo, no supondría una carga de trabajo muy elevada, o desestimarlo, en caso de suponer una carga de trabajo excesiva.

Tabla 2.8: Riesgo R07: Cambios en los requisitos.

Riesgo R08	
Titulo	Desarrollo de una interfaz de usuario poco usable.
Descripción	El desarrollador realiza una interfaz de usuario para la aplicación poco amigable o entendible, impidiendo que el usuario la use con facilidad.
Categoría	Desarrollo.
Probabilidad	Media
Impacto	Medio
Plan de mitigación	▪ Realizar bocetos y pruebas de usabilidad a posibles usuarios. ▪ Obtener retroalimentación de estas pruebas para mejorar la interfaz.
Plan de contingencia	▪ Añadir tareas nuevas para abordar estos problemas con la interfaz de usuario.

Tabla 2.9: Riesgo R08: Desarrollo de una interfaz de usuario poco usable.

Riesgo R09	
Titulo	Problemas de conectividad a la red.
Descripción	Debido a cambios en el compañía surtidora de la red de conexión a Internet puede haber días sin las posibilidad de trabajar con conexión.
Categoría	Desarrollo.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Intentar buscar otro espacio de trabajo auxiliar que permita tener conectividad a Internet activa.
Plan de contingencia	▪ Desarrollar lo posible sin conectividad a la red. ▪ Retrasar tareas que requieran conexión a internet hasta disponer de esta. ▪ Buscar otro espacio de trabajo auxiliar que permita tener conectividad a Internet activa.

Tabla 2.10: Riesgo R09: Problemas de conectividad a la red.

2.4. Presupuesto

Por último, se va a realizar un presupuesto del desarrollo del proyecto, debido a estar en un contexto académico se van a realizar dos presupuestos, el primero, un presupuesto simulado, en el cual se podrá ver el costo estimado que podría tener un proyecto como el que se va a llevar a cabo en un ambiente empresarial normal y un presupuesto real, en el que se reflejaran los costos que tendrá en realidad al ser un Trabajo de Fin de Grado.

2.4.1. Presupuesto simulado

Comenzaremos por enumerar los costos referidos al desarrollador, en este caso el estudiante, que va a llevar a cabo el proyecto. Observando de varias fuentes diferentes obtenemos una media de unos 25.000€ anuales para un desarrollador software estándar, a lo que hay que añadir el porcentaje destinado por la empresa contratante del desarrollador a la Seguridad Social de este, que es aproximadamente un 35 %, lo que nos daría un coste bruto de, aproximadamente, 38.462€ anuales (siendo el coste en Seguridad Social exclusivamente de 13.462€ anuales aproximadamente).

Al año se trabaja unas 1800 horas, por lo que el desarrollador cobraría 13.89€ por hora trabajada, con un coste en Seguridad Social de 7,48€ por hora trabajada. Como vimos anteriormente el trabajo en horas estimado para la realización de este proyecto es de 300 horas, por lo que finalmente obtenemos que el coste en la totalidad del proyecto del desarrollador es de 5.667€ en términos de sueldo y 2.244€ pagados a la Seguridad Social.

El equipo del desarrollador consta de un Acer Aspire A515-51G-751G con un valor de 589€, contando con que tenga una vida útil media de 3 años, obtenemos un costo mensual de 16,36€, lo cual nos da un costo total del dispositivo en los 4 meses de duración del proyecto de 65,44€. Como se trata de un software el cual se va a utilizar principalmente desde un dispositivo móvil, utilizaremos un OnePlus Nord 2 y una tablet iPad Air 3 como dispositivos de pruebas, con un valor de 370€ y 589€, respectivamente. Si contamos con que la vida útil media de un dispositivo móvil es de 4 años, el coste mensual del dispositivo móvil es de 7,71€ y para la tablet de 12,27€ aproximadamente, para un total de 30,83€ y 48,08€, respectivamente, en los 4 meses de desarrollo del proyecto.

Por último incluiremos los costes de las licencias de las herramientas que se utilizaran para el desarrollo del proyecto. Comenzamos por la herramienta Microsoft 365, utilizaremos el plan de empresa más básico, que incluye entre otras herramientas Microsoft Teams, con un valor de 4,20€ mensuales (IVA no incluido). También se usará la herramienta Astah Professional, para la realización de los modelos necesarios, la cual tiene un plan anual para empresas de 140€, que se quedaría en 11,67€ mensuales, utilizando ambas herramientas durante los 4 meses de duración del proyecto tenemos un total de 61,48€. Para finalizar, se utilizara la herramienta Balsamiq para realizar los bocetos de las vistas en formato dispositivo móvil, con un coste de 7,94€ mensuales en su plan más sencillo, obteniendo así un total de 34,76€.

Sumando todos los valores que hemos tenido en cuenta obtenemos un coste total del proyecto de aproximadamente **8.150,59€**. Como medida de seguridad, debido a ser una

estimación y a los posibles gastos imprevistos, como licencias o equipos extra, que se puedan requerir a lo largo del desarrollo, se aumentara esa cifra en un 30 %, obteniendo así un coste total del proyecto de **10.595,77€**.

Elemento	Coste	Duración	Total
Sueldo desarrollador	13,89€/h	300 horas	5.667€
Seguridad social del desarrollador	7,48€/h	300 horas	2.244€
Ordenador del desarrollador	16,36€/mes	4 meses	65,44€
Smartphone para pruebas	7,71€/mes	4 meses	30,83€
IPad para pruebas	12,27€/mes	4 meses	48,08€
Licencia Microsoft 365	4,20€/mes	4 meses	16,80€
Licencia Astah Professional	11,67€/mes	4 meses	46,68€
Licencia Balsamisq	7,94€/mes	4 meses	31,76€
TOTAL			**8.150,59€**
TOTAL + 30 %			**10.595,77€**

Tabla 2.11: Presupuesto simulado

2.4.2. Presupuesto real

Una vez realizado un presupuesto simulado, vamos a calcular el presupuesto real adaptando los costes evaluados en el apartado anterior a la realidad del proyecto. Al tratarse de un proyecto elaborado por un estudiante como Trabajo de Fin de Grado, se eliminarán los cálculos referidos al sueldo del empleado o costes en Seguridad Social que debería tener la empresa por el contrato del desarrollador.

Respecto al coste relacionado con los dispositivos que usaría el desarrollador, se optó por realizar la simulación con el mismo equipo con el que se va a trabajar en realidad, por lo que los costes serían los mismos, 65,44€ en total por el ordenador, 30,83€ por el dispositivo móvil y 48,08€ por la tablet.

Con respecto a las licencias usadas en el proyecto, tanto con la herramienta Microsoft 365 como con Astah Professional, al ser estudiante de la Universidad de Valladolid cuento con una licencia de estudiante gratuita, por lo que se pueden suprimir esos costes. Para finalizar, se utilizará la herramienta Balsamiq en su versión de prueba para realizar los bocetos de las vistas, por lo que el coste de herramientas será de 0€.

Al ser un coste mínimo no se tendrá en cuenta el valor añadido por posibles imprevistos que podrían surgir a lo largo del desarrollo del proyecto.

Elemento	Coste	Duración	Total
Ordenador del desarrollador	16,36€/mes	4 meses	65,44€
Smartphone para pruebas	7,71€/mes	4 meses	30,83€
IPad para pruebas	12,27€/mes	4 meses	48,08€
TOTAL			**96,27€**

Tabla 2.12: Presupuesto real

2.5. Product Backlog inicial

Como ya se comentó con anterioridad, el papel de Product Owner se llevará a cabo por el estudiante en este proyecto, por lo que también será el encargado de elaborar el Product Backlog. Mediante el desarrollo inicial del concepto de la plataforma a desarrollar, se ha elaborado un Product Backlog inicial presente en la Tabla 2.13.

Al estar trabajando en Scrum, los requisitos tienen forma de historias de usuario y siguen el siguiente modelo "Como <*stakeholder*> quiero <*objetivo*> para <*razón que aporte valor al stakeholder*> ", donde el término "*stakeholder*" hace referencia a la entidad interesada en el proyecto. Al tratarse de una lista de requisitos iniciales y siendo estos muy generales, se ha optado por escribirlos en forma de épicas (Tabla 2.13), siendo estas historias de usuario muy grandes, por lo que se pueden subdividir en más historias de usuario. A su vez, se ha realizado una primera división en posibles historias de usuario que podrían albergar las diferentes épicas descritas (Tablas 2.14 a 2.18), Esta división se irá depurando durante el transcurso del proyecto, ya que existe la posibilidad de tener que añadir o modificar algunas de estas.

Número	Épica
1	Como usuario quiero poder consultar los establecimientos registrados y sus correspondientes carta y nivel de ocupación para así poder conocer la situación de cada establecimiento.
2	Como propietario quiero poder registrar y gestionar mis negocios en la plataforma para así dar a conocer mi establecimiento a los usuarios.
3	Como cliente quiero poder realizar una comanda en un establecimiento registrado en la plataforma para así facilitar el proceso de realización de comandas de forma presencial en un establecimiento.
4	Como trabajador quiero poder consultar el estado de las comandas realizadas por los usuarios y el nivel de ocupación del establecimiento para llevar un control del volumen de actividad del local.
5	Como administrador quiero poder aprobar solicitudes de registro de propietarios y establecimientos a la plataforma para llevar un control de veracidad de los establecimientos registrados.

Tabla 2.13: Product Backlog inicial

Número	Descripción
1.1	Como usuario sin registrar quiero poder consultar los establecimientos registrados en la plataforma para conocer los establecimientos a los que podría acudir.
1.2	Como usuario sin registrar quiero poder consultar la carta de productos ofrecida por un establecimiento registrado en la plataforma para conocer que productos ofrecen.
1.3	Como usuario sin registrar quiero poder consultar el nivel de ocupación de un establecimiento registrado en la plataforma para conocer la disponibilidad de sitio en este.
1.4	Como usuario registrado quiero poder reservar una mesa en un establecimiento registrado en la plataforma para asegurar tener sitio al acudir en la fecha seleccionada.
1.5	Como usuario sin registrar quiero poder consultar el nivel de ocupación de un establecimiento registrado en la plataforma mediante el dispositivo Alexa.

Tabla 2.14: División inicial en historias de usuario de la épica número 1

Número	Descripción
2.1	Como propietario registrado quiero registrar mi establecimiento en la plataforma para darlo a conocer a los usuarios que la utilicen.
2.2	Como propietario quiero poder conocer los establecimientos que he registrado en la plataforma para poder gestionarlos.
2.3	Como propietario registrado quiero poder actualizar la información de mis establecimientos en la plataforma para tener la información relacionada con este al día.
2.4	Como propietario registrado quiero poder actualizar la plantilla de mis establecimientos para tener al día los trabajadores de mis establecimientos.
2.5	Como propietario registrado quiero poder actualizar la carta de productos ofrecidos de mis establecimientos para que los clientes puedan conocer en tiempo real los productos ofrecidos.

Tabla 2.15: División inicial en historias de usuario de la épica número 2

Número	Descripción
3.1	Como cliente quiero poder realizar una comanda en un establecimiento registrado en la plataforma para disfrutar de los productos ofrecidos por un establecimiento.
3.2	Como cliente quiero poder consultar el estado de la comanda solicitada para conocer en que estado se encuentra en el proceso de realización.
3.3	Como cliente registrado quiero poder añadir productos a una comanda ya realizada sin estar completada para poder realizar modificaciones en mi comanda si fuera necesario.

Tabla 2.16: División inicial en historias de usuario de la épica número 3

Número	Descripción
4.1	Como trabajador quiero poder consultar las comandas realizadas por los clientes que no han sido completadas para conocer el nivel de comandas pendientes actuales.
4.2	Como trabajador quiero poder consultar el nivel de ocupación del establecimiento para conocer la disponibilidad de espacio en el local actualmente.
4.3	Como trabajador quiero poder establecer una mesa del establecimiento como libre para el uso de esta por otro cliente para llevar el nivel de ocupación del establecimiento en tiempo real y que otro cliente pueda ocuparla.
4.4	Como trabajador quiero poder consultar todas las comandas realizadas por los clientes para conocer el volumen de comandas realizadas en un periodo de tiempo determinado.

Tabla 2.17: División inicial en historias de usuario de la épica número 4

Número	Descripción
5.1	Como administrador quiero poder consultar las peticiones nuevas de registro de establecimiento a la plataforma para conocer los establecimientos que se quieren agregar a la plataforma.
5.2	Como administrador quiero poder consultar las peticiones nuevas de registro de propietarios a la plataforma para conocer los propietarios que quieren registrarse a la plataforma.
5.3	Como administrador quiero poder aceptar o rechazar una petición de registro de un establecimiento en la plataforma para controlar la veracidad de los datos antes de su publicación.
5.4	Como administrador quiero poder aceptar o rechazar una petición de registro de un propietario en la plataforma para controlar la veracidad de los datos antes de su publicación.

Tabla 2.18: División inicial en historias de usuario de la épica número 5

2.6. Product Backlog final

Durante el desarrollo del proyecto se han ido realizando algunos cambios al Product Backlog inicial mostrado en el anterior apartado, obteniendo así una versión final del Product Backlog mostrado en las Tablas 2.20 a 2.24.

Como se puede observar en la Tabla 2.19 con respecto a la Tabla 2.13, donde se reflejan las épicas iniciales planificadas, ha aparecido una épica más, utilizada para la realización de tareas relativas a mejoras en la interfaz de usuario.

Número	Épica
1	Como usuario quiero poder consultar los establecimientos registrados y sus correspondientes carta y nivel de ocupación para así poder conocer la situación de cada establecimiento.
2	Como propietario quiero poder registrar y gestionar mis negocios en la plataforma para así dar a conocer mi establecimiento a los usuarios.
3	Como cliente quiero poder realizar una comanda en un establecimiento registrado en la plataforma para así facilitar el proceso de realización de comandas de forma presencial en un establecimiento.
4	Como trabajador quiero poder consultar el estado de las comandas realizadas por los usuarios y el nivel de ocupación del establecimiento para llevar un control del volumen de actividad del local.
5	Como administrador quiero poder aprobar solicitudes de registro de propietarios y establecimientos a la plataforma para llevar un control de veracidad de los establecimientos registrados.
6	Como desarrollador quiero poder mejorar el aspecto de la interfaz gráfica para poder brindar a los usuarios de la plataforma de la mejor experiencia de usuario posible.

Tabla 2.19: Product Backlog final

Para las historias de usuario de la épica número 1, en la Tabla 2.20 se pueden observar la aparición de las historias de usuario 1.6 y 1.7, mientras que en las historias de usuario definidas al inicio del proyecto para esta épica, visibles en la Tabla 2.14, no existían. Por otro lado se puede observar que tanto la historia de usuario 1.4 y 1.5, de la división inicial, no se han llevado a cabo en el proyecto final.

Número	Descripción
1.1	Como usuario sin registrar quiero poder consultar los establecimientos registrados en la plataforma para conocer los establecimientos a los que podría acudir.
1.2	Como usuario sin registrar quiero poder consultar la carta de productos ofrecida por un establecimiento registrado en la plataforma para conocer que productos ofrecen.
1.3	Como usuario quiero poder consultar el nivel de ocupación de un establecimiento registrado en la plataforma para conocer la disponibilidad de sitio en este.
1.6	Como usuario quiero poder ver una imagen del establecimiento para poder diferenciar los establecimientos y que la interfaz sea más amigable.
1.7	Como usuario quiero poder registrarme en la plataforma para así poder acceder a las funcionalidades restringidas a usuarios registrados.

Tabla 2.20: Historias de usuario finales de la épica número 1

Con respecto a las historias de usuario de la épica número 2 (Tabla 2.21), se puede observar con facilidad de un gran cambio en el número de estas con respecto a la división inicial (Tabla 2.15). La historia de usuario 2.5 de la división inicial, se ha dividido en las

historias de usuario 2.8, 2.9, 2.10, 2.11, 2.5, 2.12 y 2.13, mostradas en la Tabla 2.21, siendo las 4 primeras de estas relativas a las operaciones de consultar, agregar, eliminar y editar productos de la lista general de estos registrados por un propietarios, las dos siguientes para la edición de los productos que componen la carta del establecimiento y la última para la edición del menú del local. También aparecen historias de usuario nuevas para operaciones como la gestión de mesas, aforo y trabajadores de un local, el inicio de sesión y la asignación de los establecimientos creados por un propietario a este mismo, para que pueda gestionarlos de manera óptima.

Número	Descripción
2.1	Como propietario registrado quiero registrar mi establecimiento en la plataforma para darlo a conocer a los usuarios que la utilicen.
2.2	Como propietario quiero poder conocer los establecimientos que he registrado en la plataforma para poder gestionarlos.
2.3	Como propietario registrado quiero poder actualizar la información de mis establecimientos en la plataforma para tener la información relacionada con este al día.
2.4	Como propietario registrado quiero poder gestionar la plantilla de mis establecimientos para tener un control de estos en cada establecimiento.
2.5	Como propietario registrado quiero poder añadir productos a la carta de productos ofrecidos de mis establecimientos para que los clientes puedan conocer en tiempo real los productos ofrecidos.
2.6	Como propietario registrado quiero poder iniciar sesión en la aplicación para poder acceder a las funcionalidades exclusivas de propietario registrado.
2.7	Como propietario registrado quiero poder asociar los establecimientos que cree a mi identificador de usuario para poder modificar la carta y la información sobre estos.
2.8	Como propietario registrado quiero poder añadir un producto a la lista de productos de mis establecimientos para poder incluiros en la carta de mis establecimientos.
2.9	Como propietario registrado quiero poder consultar los productos añadidos a la plataforma que podría servir en mis establecimientos para poder llevar un control de los productos que puedo servir.
2.10	Como propietario registrado quiero poder eliminar los productos añadidos a la plataforma para poder tener actualizados los productos de los que dispongo.
2.11	Como propietario registrado quiero poder editar los productos añadidos a la plataforma para poder tener la información de estos actualizada.
2.12	Como propietario registrado quiero poder eliminar productos de la carta de productos ofrecidos de mis establecimientos para que los clientes puedan conocer en tiempo real los productos ofrecidos.
2.13	Como propietario registrado quiero poder consultar los productos añadidos a la plataforma que podría servir en mis establecimientos para poder llevar un control de los productos que puedo servir.
2.14	Como propietario registrado quiero poder gestionar las mesas añadidas a mis establecimientos registrados en la plataforma para poder llevar un control de estas.
2.15	Como propietario registrado quiero poder asignar un valor al aforo máximo permitido en mis establecimientos registrados en la plataforma y poder actualizar el aforo máximo actual, siempre menor o igual al aforo máximo del local y la capacidad de las mesas activas en el momento.
2.16	Como propietario registrado quiero poder añadir trabajadores a la plantilla de mis establecimientos para tener al día los trabajadores de mis establecimientos.

Tabla 2.21: Historias de usuario finales de la épica número 2

Como se puede observar en la Tabla 2.22, las historias de usuario definidas en la división inicial de la épica número 3 (Tabla 2.16), para el rol del cliente registrado (3.2 y 3.3) no se han llevado a cabo, ya que el producto mínimo viable desarrollado para este proyecto no incluía este tipo de usuario.

Número	Descripción
3.1	Como cliente quiero poder realizar una comanda en un establecimiento registrado en la plataforma para disfrutar de los productos ofrecidos por un establecimiento.

Tabla 2.22: Historias de usuario finales de la épica número 3

Para las historias de usuario de la épica 4, la división inicial realizada (Tabla 2.17) incluía la posibilidad de que el trabajador consultase la totalidad de las comandas realizadas por los clientes, a mayores de las comandas activas por mesa, y consultar el nivel de ocupación, mientras que la división final de esta épica (Tabla 2.23), solamente incluye la posibilidad de consultar las comandas activas y la gestión de las mesas. Por otro lado la historia de usuario 4.2 de la división inicial, relativa a comprobar el nivel de ocupación del local, se ha juntado con la historia de usuario 4.3, ya que al tener, en tiempo real, el estado de las mesas puede conocer como de ocupado está su lugar de trabajo, resultando así en la historia de usuario 4.2 de la división final.

Número	Descripción
4.1	Como trabajador quiero poder consultar las comandas por cada mesa realizadas por los clientes que no han sido completadas para conocer el nivel de comandas pendientes actuales.
4.2	Como trabajador quiero poder establecer una mesa del establecimiento como libre para el uso de esta por otro cliente para llevar el nivel de ocupación del establecimiento en tiempo real y que otro cliente pueda ocuparla.

Tabla 2.23: Historias de usuario finales de la épica número 4

Las historias de usuario de la épica número 5, relativas a las operaciones realizadas por el administrador sobre las peticiones de registro tanto de propietarios, como de los establecimientos de estos, no han recibido cambios entre la división inicial y la final, mostradas en las Tablas 2.18 y 2.24, respectivamente.

Número	Descripción
5.1	Como administrador quiero poder consultar las peticiones nuevas de registro de establecimiento a la plataforma para conocer los establecimientos que se quieren agregar a la plataforma.
5.2	Como administrador quiero poder consultar las peticiones nuevas de registro de propietarios a la plataforma para conocer los propietarios que quieren registrarse a la plataforma.
5.3	Como administrador quiero poder aceptar o rechazar una petición de registro de un establecimiento en la plataforma para controlar la veracidad de los datos antes de su publicación.
5.4	Como administrador quiero poder aceptar o rechazar una petición de registro de un propietario en la plataforma para controlar la veracidad de los datos antes de su publicación.

Tabla 2.24: Historias de usuario finales de la épica número 5

Como se comentó al inicio de este apartado, la épica número 6 no era parte de la división inicial en épicas del proyecto, ya que durante el desarrollo del mismo hubo que realizar en dos momentos mejoras en la interfaz de usuario, con lo que en un principio no se contaba.

Número	**Descripción**
6.1	Como desarrollador quiero mejorar la interfaz gráfica de la plataforma para que el usuario tenga la mejor experiencia posible y sea lo más sencilla e intuitiva de usar.
6.2	Como desarrollador quiero mejorar la interfaz gráfica de la plataforma para que el usuario tenga la mejor experiencia posible y sea lo más sencilla e intuitiva de usar.

Tabla 2.25: Historias de usuario finales de la épica número 6

Capítulo 3

Análisis

En este Capítulo se presentan los resultados del análisis realizado a lo largo de los diferentes sprints durante el desarrollo del proyecto para identificar los conceptos relevantes del dominio del problema. El análisis comenzó en el sprint 0, definiendo un modelo de dominio inicial, que ha sido modificado durante el desarrollo del resto de sprints, siendo este un aspecto a tener en cuenta en cada historia de usuario.

3.1. Modelo de dominio

En la Figura 3.1 se muestra el modelo conceptual del dominio. En este se muestran todas las entidades que se han identificado tras analizar las historias de usuario de Product Backlog.

Se muestran diferentes entidades dentro del modelo de dominio en color rojo, las cuales no entran dentro del alcance de este proyecto, aunque forman parte del sistema y de ciertas historias de usuario planificadas en el inicio del proyecto. Estas entidades que han quedado fuera del alcance podrían ser desarrolladas en un futuro junto a otros aspectos que podrían entrar dentro del amplio mundo de los locales de restauración. Entre estos aspectos que quedan fuera del alcance del proyecto se pueden observar entidades como los clientes registrados y las reservas, y ciertos atributos de las diferentes entidades que si se han abordado en este proyecto, que darían mayor información a entidades como los establecimientos o productos, o en entidades como el trabajador, en la cual el rol podría definir la jerarquía interna del establecimiento.

También hay enumeraciones, como el rol del trabajador mencionado, que están descritas en el modelo como una primera aproximación, por lo que en caso de desarrollar estas, podrían tener más campos que diesen mayor información.

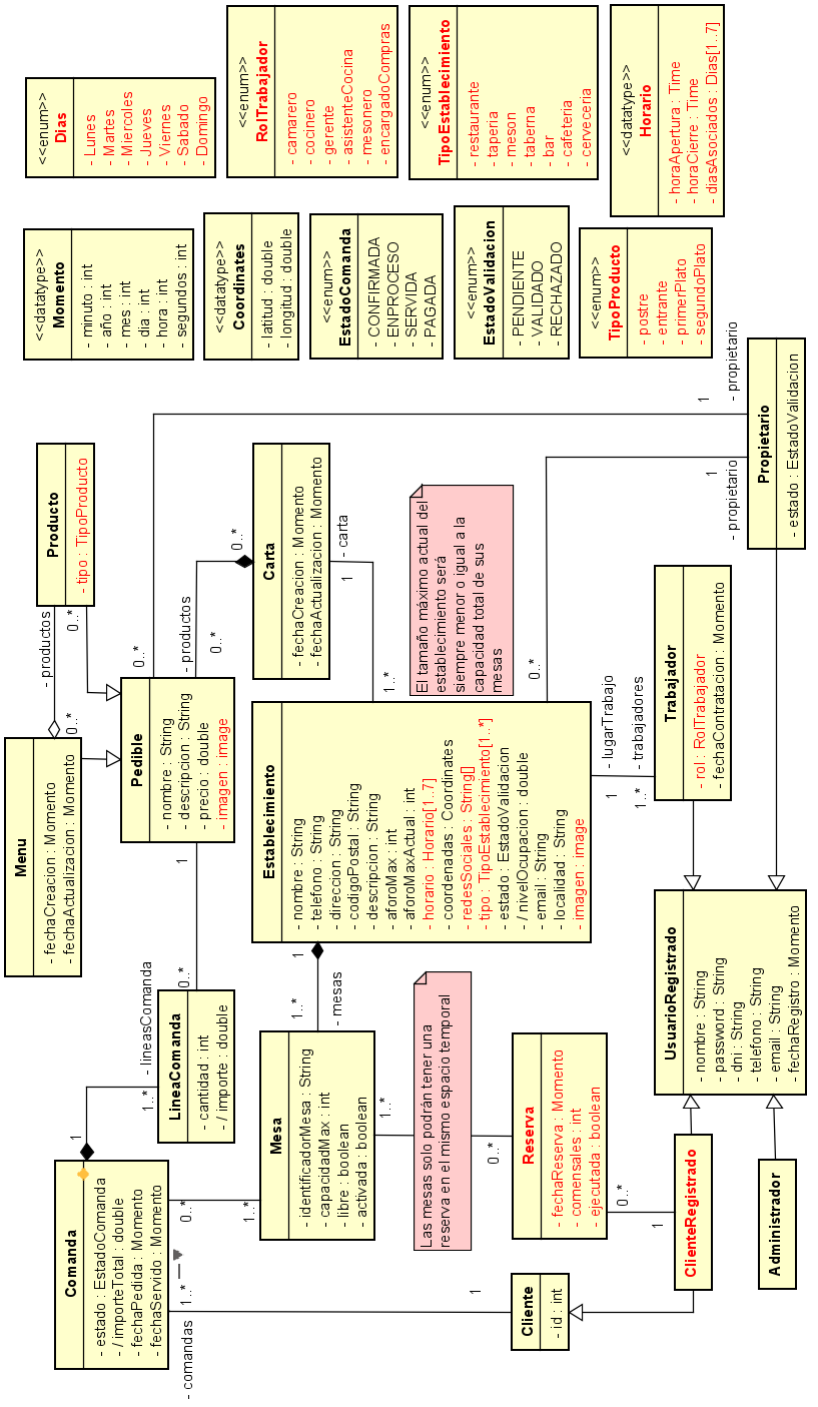


Figura 3.1: Modelo de dominio

3.2. Modelado de interacción

La interacción con el sistema de los diferentes usuarios provoca cambios en los estados de algunos de los objetos que existen dentro de la plataforma. En el caso de un usuario con rol de administrador, su principal actividad consiste en la gestión de las solicitudes de registro de propietarios y establecimientos, resultando en cambios en los estados de dichas solicitudes; por otro lado, si el usuario tiene un rol de trabajador, es el encargado de pasar de estado las diferentes comandas de las mesas de su lugar de trabajo. Estos cambios de estado se pueden observar en las máquinas de estado de las Figuras 3.2 y 3.3.

En el caso de la segunda máquina de estados, cabe destacar que es la misma tanto para la gestión de solicitudes de registro de propietarios como de establecimientos, por lo que se ha realizado una máquina de estados para solicitudes generales.

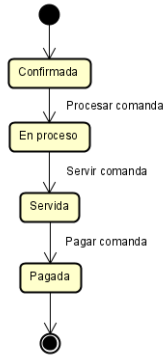


Figura 3.2: Diagrama de maquina de estados para las comandas.

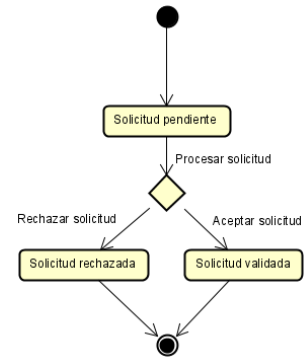


Figura 3.3: Diagrama de maquina de estados para las solicitudes.

3.3. Modelo de procesos de negocio

Para finalizar el análisis del proyecto se han elaborado dos diagramas de actividad, mostrando los dos principales procesos de negocio que se pueden llevar a cabo con el uso de la plataforma: solicitud de registro de un establecimiento por parte de un propietario (Figura 3.4) y la representación del ciclo de vida de una comanda, desde que el cliente escoge un establecimiento hasta que es marcada como pagada pasando a través de los diferentes estados (Figura 3.5).

El proceso de negocio de la Figura 3.4, que describe el proceso de solicitud de registro de un establecimiento nuevo en la plataforma, es muy similar al proceso de negocio llevado a cabo para validar una solicitud de registro de un propietario dentro de la plataforma, por lo que este último no se ha descrito.

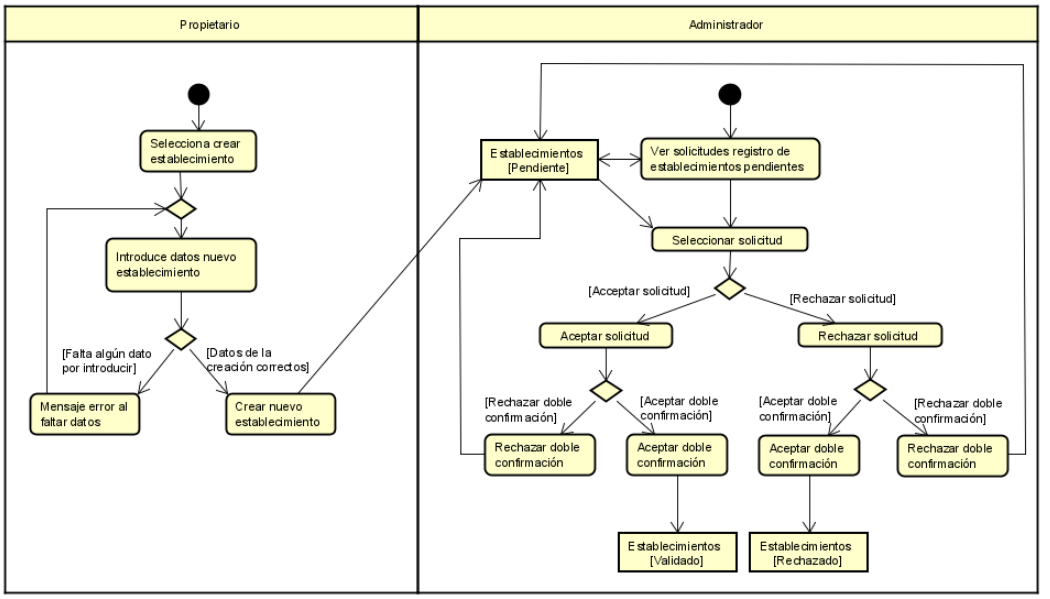


Figura 3.4: Proceso de negocio solicitud de registro de un establecimiento.

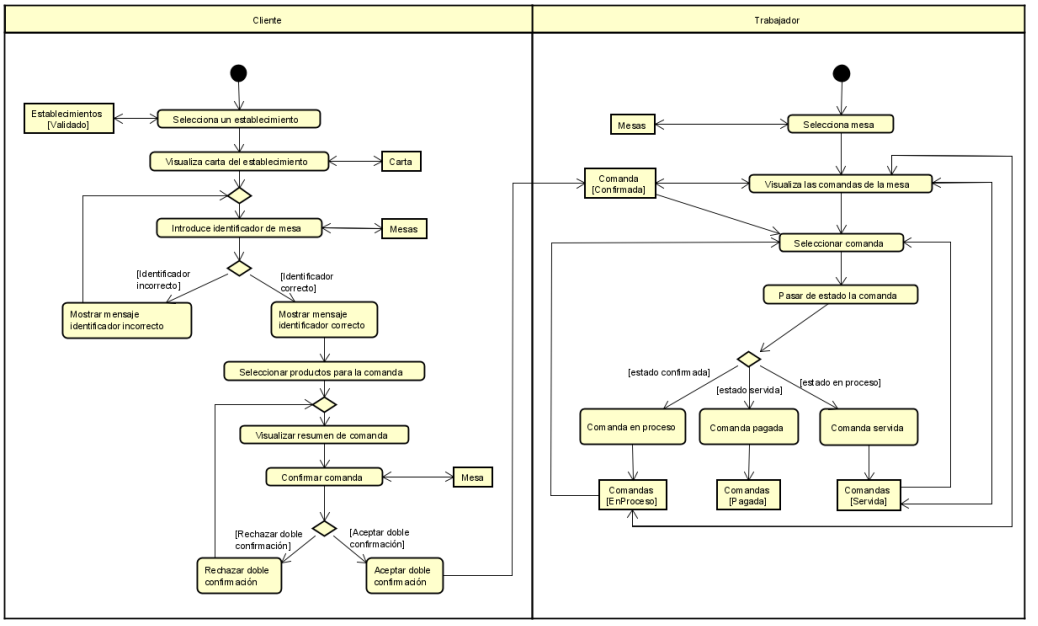


Figura 3.5: Proceso de negocio ciclo de vida de una comanda.

Capítulo 4

Tecnologías utilizadas

En este Capítulo se presenta un resumen de las tecnologías utilizadas tanto para la gestión del proyecto, como para el desarrollo, finalmente comentando la plataforma utilizada para el despliegue del sistema.

4.1. Tecnologías para la gestión del proyecto

4.1.1. Overleaf

Overleaf [6] es un editor online de textos en formato LaTeX [4] que permite la colaboración en tiempo real de los editores, así como escribir artículos científicos, técnicos, tesis, informes y cualquier documento que se quiera presentar en el formato indicado.
Es el editor de texto utilizado para la elaboración de este documento.

4.1.2. Balsamiq

Balsamiq Wireframes [1] es una herramienta rápida de wireframing que reproduce la experiencia de maquetación de interfaces de usuario en papel, pero utilizando un ordenador. Fuerza a que te centres puramente en la estructura y el contenido, evitando discusiones sobre colores y detalles que deberían abordarse en el futuro.
Se ha utilizado en este proyecto para la realización de los bocetos para las vistas.

4.1.3. Microsoft Teams

Microsoft Teams [73] es una plataforma unificada de comunicación y colaboración que combina chat persistente en el lugar de trabajo, reuniones de vídeo, almacenamiento de

archivos (incluida la colaboración en archivos) e integración de aplicaciones. El servicio se integra con el paquete de productividad de Office por suscripción y presenta extensiones que pueden integrarse con productos que no son de Microsoft.
Es el método de comunicación utilizado en el desarrollo de este proyecto, tanto para las reuniones Scrum semanales, como para la compartición de archivos y solución de dudas puntuales.

4.1.4. Git

Git [41] es un sistema distribuido, libre y de código abierto para el control de versiones diseñado para la gestión de todo tipo de proyectos con velocidad y eficiencia, manteniendo un control de los cambios que se producen en los ficheros de código fuente.

Git se basa principalmente en un sistema de repositorio y ramas. Un repositorio es un espacio de almacenamiento virtual donde se aloja el proyecto. Una rama es una copia exacta del proyecto, que se crea desde otra rama. *Git* permite tener múltiples ramas locales que pueden ser totalmente independientes entre si, siendo la creación, borrado y unión (*merge*) de estas muy rápida. Por defecto, la rama original de un proyecto es denominada *master* o *main*. Una representación de la ramificación de un proyecto se puede observar en la Figura 4.1.

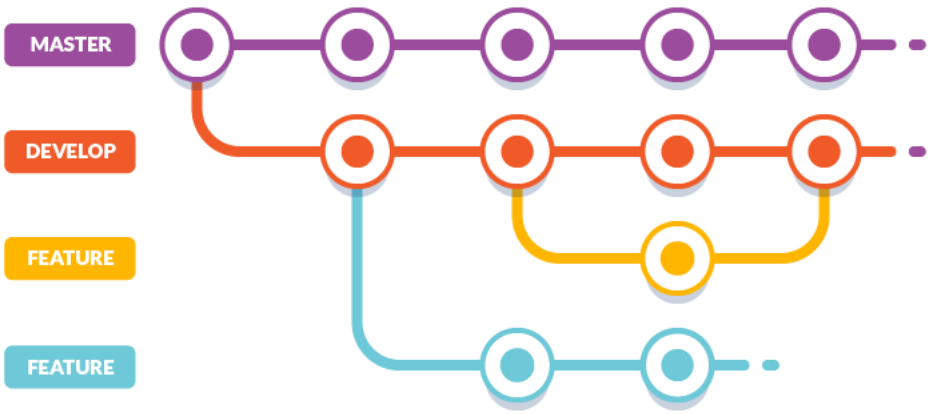


Figura 4.1: Ramificación de un proyecto [79].

Como se ha comentado anteriormente, *Git* permite el trabajo en local, mediante la clonación de repositorios en el almacenamiento local del dispositivo donde se va a trabajar, para posteriormente grabar los cambios y subirlos al repositorio remoto. Para ello se usan una serie de operaciones: `pull`, para traer a entorno local los últimos cambios de la rama en la que se encuentra; `commit`, para guardar los cambios realizados en local; y `push`, para subir al

repositorio remoto los `commits` guardados desde la última realización de un comando `push`. Existe un comando más, `add`, utilizado en la denominada zona de preparación o staging area, gracias al cual podemos pasar los cambios realizados, en entorno local, de los ficheros seleccionados al staging area, para posteriormente realizar la operación de `commit`. La existencia de esta staging area permite al usuario la posibilidad de supervisar los cambios realizados en local y corregir, en caso de ser necesario, antes de completar la operación de `commit`. El proceso de grabación de cambios en el repositorio local utilizando el staging area se puede observar en la Figura 4.2.

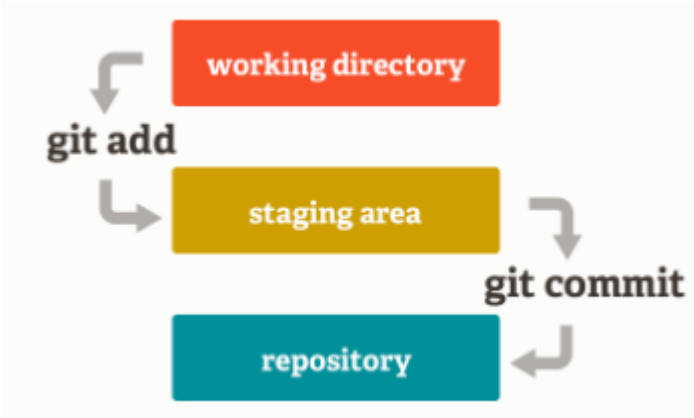


Figura 4.2: Funcionamiento del staging area de *Git* [42].

Las ramas utilizadas en este proyecto han sido:

- *master/main*: rama original del proyecto que contiene una versión estable del mismo. Esta rama no contendrá nada de funcionalidad hasta el fin del desarrollo, ya que hasta ese momento no se considera que sea una versión estable que pueda pasar a producción.
- *develop*: rama secundaria creada a partir de la rama principal en el inicio del proyecto. Desde esta rama se crearán el resto de ramas para el desarrollo de las historias de usuario, que tras finalizar se fusionarán de vuelta con develop, por lo que develop mantiene la versión actual del estado del proyecto.
  En el momento de que el proyecto se encuentre en una versión estable, esta rama se fusionará con la rama principal.
- *HU<NúmeroHU>-<descripción de la funcionalidad>*: cada una de las ramas creadas a partir de la rama develop para el desarrollo de funcionalidad han seguido esta nomenclatura, definiendo así en cada una de estas una breve descripción de la funcionalidad a desarrollar y el número de historia de usuario, para que sean mas fáciles de identificar.
- *debug-<descripción del bug>*: cada una de las ramas creadas a partir de la rama develop para el arreglo de aspectos visualizados al realizar variaciones en algún apartado del proyecto.

4.1.5. GitLab

GitLab [44] es un popular servicio web de control de versiones de código abierto utilizado para el desarrollo software, principalmente en ámbitos de desarrollo colaborativo. GitLab se basa en el software de gestión de versiones Git, mencionado anteriormente.
GitLab ha sido utilizado en este proyecto como sistema de control de versiones y como herramienta de gestión del proyecto.

4.1.6. GitLab Issue Board

Las issues [43] o incidencias son el medio fundamental que proporciona GitLab para colaborar en ideas, resolución de problemas y planificación del trabajo.

Una de las herramientas que proporciona GitLab para el manejo y gestión de issues es el GitLab Issue Board, el cual consta de varias columnas de tarjetas que representan las issues creadas para el proyecto en su estado. Cada tarjeta de issue lleva asociado un nombre, una descripción, un tiempo estimado para ser completada, un seguimiento del tiempo empleado en el desarrollo y el usuario de GitLab al que ha sido asociada.

En este proyecto se ha asociado cada historia de usuario a una issue siguiendo la nomenclatura "Historia de usuario <NumeroHU> - <descripción de la HU>", y gracias a esta herramienta se podrá llevar un control del tiempo empleado en la realización de cada una de estas.

4.2. Tecnologías para el desarrollo del proyecto

4.2.1. Spring Boot

Spring Boot [7] es un módulo del proyecto de Spring que fue creado para simplificar el desarrollo de aplicaciones con Spring Framework bajo licencia Apache 2.0, esto nos permite crear aplicaciones del lado del servidor en lenguajes como Java, Kotlin y Groovy, con esto nos podemos olvidar de la arquitectura y enfocarnos únicamente en desarrollo, delegando a Spring Boot labores como configuración de dependencias, desplegar nuestro servicio o aplicación a un servidor de aplicaciones y enfocarnos únicamente en crear nuestro código. Para esto Spring Boot utiliza internamente un servidor de aplicaciones embebido, por defecto utiliza Tomcat [14] [45].

El backend de la aplicación fue desarrollado utilizando Spring Boot y Java.

4.2.2. Java

Java [57] es un lenguaje de programación que nació en 1995 con el objetivo de tener una estructura sencilla para poder ser ejecutado en diversos sistemas operativos, se utiliza para crear aplicaciones y procesos y está basado en la programación orientada a objetos [97].

Es el lenguaje de programación utilizado para el desarrollo del backend junto al módulo SpringBoot.

4.2.3. JWT

JSON Web Token (JWT) [59] es un estándar abierto (RFC 7519 [58]) que define un método compacto y autocontenido para transmitir información entre dos partes de forma segura utilizando un objeto JSON. Esta información puede ser verificada y de confianza al ser firmada digitalmente. Los JWTs pueden ser firmados usando tanto claves secretas (con el algoritmo HMAC) como con pares de claves públicas o privadas usando los sistemas de criptografía RSA [76] o ECDSA [36].

Un token JWT consta de tres partes, separadas por puntos (.):

- **Header (Cabecera)**: consta normalmente de dos partes, el tipo de token, en este caso JWT y el algoritmo utilizado para la firma, como SHA256 [13].
- **Payload (Carga)**: contiene los datos sobre la entidad (en este caso el usuario registrado en la plataforma) y datos adicionales que se pueden añadir.
- **Signature (Firma)**: para crear la parte de la firma es necesario la encriptación de las otras dos partes del token JWT, el algoritmo definido en la cabecera y una clave secreta, y firmar todo esto junto.

En este proyecto se han utilizado los JWTs para el control de usuarios registrados en la plataforma y sus campos como el identificador de usuario, el rol con el que se han registrado dentro de la plataforma y el estado de su validación por parte de un administrador.

4.2.4. Apache Tomcat

Apache Tomcat [39] es un contenedor de servlets desarrollado con Java que puede utilizarse como servidor web donde ejecutar código Java.
En este proyecto se ha usado apache Tomcat para desplegar el backend Spring Boot.

4.2.5. JPA

La API de Persistencia de Java o JPA [82] es una especificación que indica cómo se debe realizar la persistencia (almacenamiento) de los diferentes objetos en programas Java [9].

Spring incluye soporte para JPA, pero no es una implementación propia de este, si no que necesita una implementación como Hibernate para funcionar.

SpringJPA ofrece una abstracción sobre JPA que facilita la creación del patrón repositorio, permite generar consultas JPA mediante convenciones de nombres y permite controlar los límites de las transacciones de manera declarativa.

4.2.6. MySQL

MySQL [27] es un sistema de gestión de bases de datos relacional de código abierto, que actualmente se considera como el más extendido.
En este proyecto se ha utilizado MySQL como gestor de la base de datos de la plataforma

4.2.7. Angular

Angular [11] es es un framework para aplicaciones web desarrollado en TypeScript [98], mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página (SPA[34]), en las que todas las vistas se encuentran en la misma página y todo el código se carga de una sola vez, o de forma dinámica, cuando las acciones realizadas por el usuario requieran la carga de ciertos recursos. Los proyectos Angular, contienen un único archivo HTML, denominado index.html, en el cual se van cargando el código del resto de vistas.

Las aplicaciones Angular están basadas en componentes, elementos compuestos por un archivo HTML (template del componente), una hoja de estilos (en formato CSS) y un archivo de lógica (con extensión .ts), donde se desarrollarán los métodos que realizarán las acciones del componente.

Angular ofrece la posibilidad de desarrollar la aplicación mediante TypeScript [52], un superset o superconjunto del lenguaje de programación JavaScript, esto significa que puede ejecutar programas de la tecnología, TypeScript, y del lenguaje del que es superset, JavaScript. TypeScript implementa conceptos de programación orientada a objetos y es de tipado estático.
TypeScript es el lenguaje utilizado para el desarrollo del frontend de este proyecto.

4.2.8. Angular Material

Angular Material [65] es una librería de estilos basada en la guía de diseño de Material Design, realizada por el equipo de Angular, brindando así una integración perfecta en proyectos de este tipo.
En este proyecto se han utilizado diferentes componentes que Angular Material brinda para crear la interfaz gráfica de una manera homogénea, como pueden ser las tablas, el paginador de tablas, las listas y los botones.

4.2.9. Node.js y NPM

Node.js [38] es un entorno de ejecución de JavaScript construido con el motor de Google V8 [47]. En este proyecto se ha utilizado para desplegar el frontend de la plataforma.

Node Package Manager (NPM) [51] es un gestor de paquetes de Node desarrollado en JavaScript que permite añadir nuevos paquetes y módulos a aplicaciones desarrolladas con Node.

4.2.10. Apache HTTP Server

Apache HTTP Server [39] es un servidor web gratuito de código abierto, multiplataforma y de uso muy extendido. Está desarrollado y mantenido por una comunidad de usuarios en torno a la Apache Software Fundation [31].

En este proyecto se usa para desplegar el backend de la plataforma.

4.2.11. Visual Studio Code

Visual Studio Code [74] es un editor de código fuente desarrollado por Microsoft para Windows, Linux, MacOS y Web, que da soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También cuenta con una alta cantidad de extensiones que brindan de una gran personalización del entorno a gustos del usuario. Es uno de los editores de texto más versátiles y utilizados actualmente.
Ha sido el editor de texto utilizado para el desarrollo tanto del backend como del frontend en este proyecto.

4.2.12. Astah Professional

Astah Professional [15] es una herramienta de modelado de Software que permite la creación de una gran variedad de diagramas, como diagramas UML (Unified Modeling Language) y diagramas ER (Entity-relationship). UML [49], o lenguaje unificado de modelado, es el lenguaje de modelado de sistemas de software más utilizado en la actualidad, respaldado por el Object Management Group [48].

4.3. Tecnología para el despliegue del sistema

4.3.1. Heroku

Heroku [54] es una plataforma de servicios en la nube que permite a las empresas manejar los servidores y sus configuraciones, escalamiento y la administración. Heroku proporciona unos contenedores virtuales, denominados "dynos" [53], en los cuales se despliegan las aplicaciones, y solo necesita conocer el lenguaje de programación utilizado en el proyecto, para poder realizar dicho despliegue.

Se ha utilizado Heroku para el despliegue tanto del backend como del frontend de la plataforma.

4.3.2. ClearDB

ClearDB [77] es una base de datos MySQL de alta disponibilidad como servicio que se ejecuta en diferentes entornos de computación en la nube en entornos híbridos de nube/-centro de datos. Está diseñado para eliminar la necesidad de administrar las bases de datos MySQL. ClearDB proporciona clústeres de bases de datos MySQL especializados y de alta disponibilidad que se construyen en torno a la premisa de fallas para garantizar que la base de datos se mantiene en linea y completamente disponible las 24 horas todos los días de la semana.

Es la base de datos MySQL utilizada para alojar los datos de la plataforma tras el despliegue.

Capítulo 5

Diseño

Una vez realizado el proceso de análisis y descritas las tecnologías utilizadas, es momento de realizar el diseño de la plataforma final. Durante este Capítulo se definirán el patrón de arquitectura utilizado, los diferentes patrones de diseño implementados, una descripción acerca de el desarrollo basado en componentes aplicado a Angular, incluyendo el listado de componentes utilizados en el desarrollo de la plataforma. También se definirá el modelo de datos para la base de datos utilizada, el diseño arquitectónico del backend y el diagrama de despliegue de la plataforma.

Por ultimo se mostrarán los bocetos desarrollados para la interfaz de usuario.

5.1. MVVM

El patrón Modelo-Vista-Modelo de Vista (Model-View-ViewModel, abreviado MVVM) es un patrón de arquitectura que se caracteriza por tratar de desacoplar lo máximo posible la interfaz de usuario de la lógica de la aplicación, logrando así que la parte visual sea totalmente independiente. Este patrón arquitectónico consta de tres componentes principales:

- **Modelo (Model)**: es una capa totalmente independiente de las otras, en ella encontramos la lógica del negocio.
- **Vista (View)**: Este componente tiene la interfaz de usuario, la cual se encarga de presentar la información del modelo por pantalla al usuario. También se encarga de capturar los eventos de interacción del usuario con la interfaz y puede ejecutar lógica de IU.
- **Modelo de vista (ViewModel)**: este componente tiene la finalidad de actuar como intermediario entre la vista y el modelo, mediante comandos, métodos, componentes u otros medios.

En la Figura 5.1 podemos observar las relaciones entre los distintos componentes de este patrón arquitectónico, observando el desacoplamiento entre las distintas partes, donde la Vista solo conoce al Modelo de Vista, y este solo conoce al Modelo.

Esta separación vista nos proporciona varias ventajas:

- Separar el desarrollo de la interfaz de usuario del resto del código.
- Posibilidad de testeo de la lógica de la presentación mediante test unitarios

Este patrón también nos permite establecer un data binding o enlace de datos entre la Vista y el Modelo de Vista, lo cual proporciona la posibilidad de que el Modelo de Vista notifique a la Vista cuando ocurre algo en el Modelo, de tal forma que la Vista se actualice automáticamente. Lo mismo pasa entre el Modelo y el Modelo de Vistas, siendo esta vez el Modelo quien notifica al Modelo de Vistas.

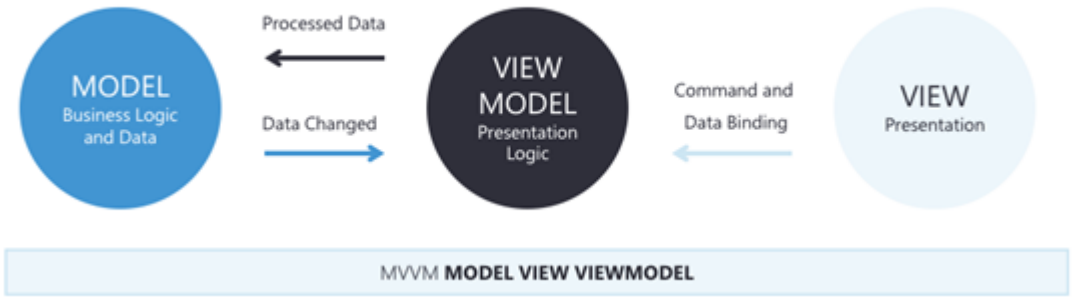


Figura 5.1: Relación entre los componentes del patrón MVVM [24]

5.2. Patrones de diseño utilizado

En este proyecto se ha utilizado varios patrones de diseño, el patrón DAO, el patrón DTO, aunque se han usado en conjunto, el patrón Repositorio y el patrón Observador en el frontend de la plataforma

5.2.1. Patrón DAO/DTO

El patrón DAO/DTO, como se mencionó anteriormente, es en realidad un conjunto de dos patrones por separado. el patrón DAO y el patrón DTO.

Por una parte, el **patrón DAO (Data Access Object)**, u Objeto de Acceso de Datos [21], propone separar por completo la lógica de negocio de la lógica de datos, proporcionando,

mediante una clase (el DAO), los métodos necesarios para insertar, actualizar, borrar y consultar la información. Mientras que por otra parte, la capa de negocio solo se preocupa por la lógica de negocio y utiliza el DAO para interactuar con la fuente de datos. En nuestro caso las clases DAO reciben el nombre de "Repositorio", encargadas de realizar las operaciones CRUD.

Por otra parte, el **patrón DTO (Data Transfer Object)**, u Objeto de Transferencia de Datos [22], es un patrón de diseño que proporciona un objeto (el DTO), el cual contiene la información deseada, pudiendo contener dicha información de múltiples fuentes o tablas, concentrándolas en una única clase simple. En nuestro caso, las clases DTO serán las "Entities".

En la Figura 5.2 se puede observar una combinación de estos dos patrones.

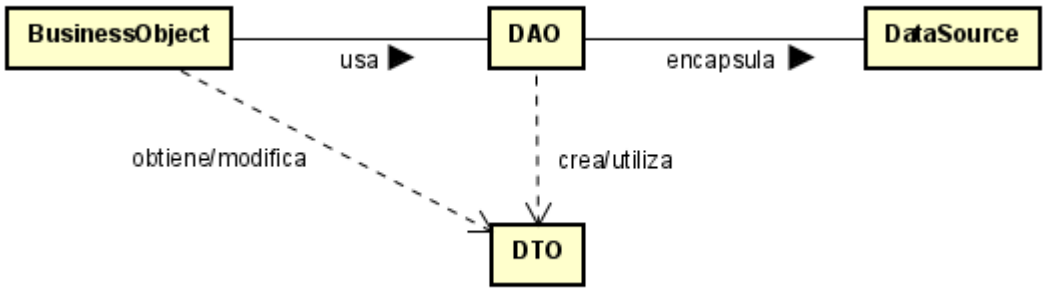


Figura 5.2: Patrón DAO/DTO [56].

5.2.2. Patrón Repository

El patrón Repository consta de una interfaz la cual contiene todas las operaciones lógicas de lectura y escritura para una entidad en específico. Esta interfaz es implementada por una o varias clases que proporcionan implementaciones específicas del almacén de datos de cada método de la interfaz.

En la Figura 5.3 se puede observar la estructura general del patrón Repository.

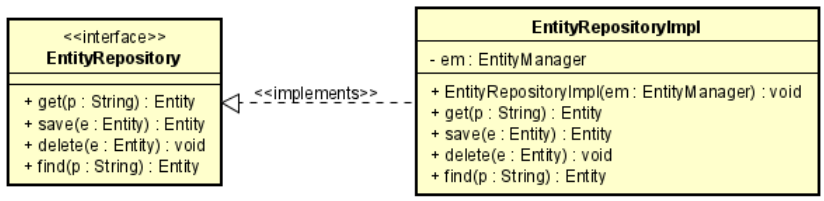


Figura 5.3: Patrón Repository [80].

5.2.3. Aplicación del patrón Observador

Antes de comentar sobre los componentes Angular, cabe destacar la aplicación del patrón Observador en el framework. Esto se lleva a cabo declarando atributos de tipo Observable y Subject, y los métodos necesarios que devuelven un valor cuyo tipo es Observable. Este patrón permite un sistema de "suscripción" sobre los objetos de tipo Subject, el cual realiza una notificación a sus "suscriptores" cuando su valor cambia.

En la Figura 5.4 se puede observar la estructura general del patrón Observador.

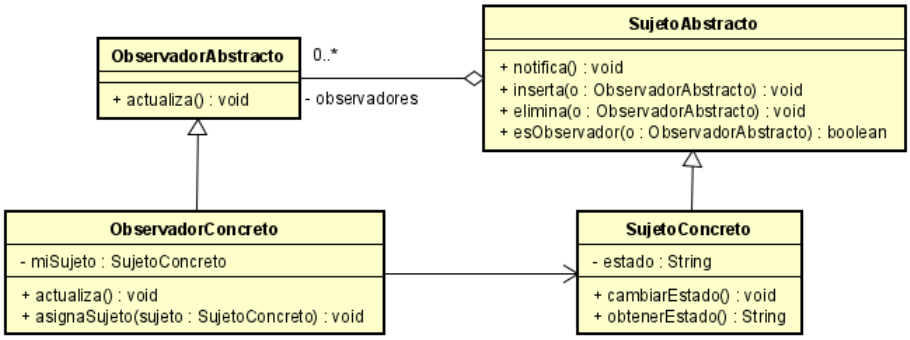


Figura 5.4: Patrón Observador [46].

5.3. Desarrollo basado en componentes

El desarrollo de software basado en componentes constituye una aproximación del desarrollo de software que describe, construye y emplea técnicas software para elaborar sistemas abiertos y distribuidos, mediante el ensamblaje de partes software reutilizables. Este enfoque beneficia la reducción de costos, tiempo y esfuerzo en el desarrollo de nuevo software, incrementando el nivel de productividad y reduciendo los riesgos, también optimiza la fiabilidad, flexibilidad, reutilización y mantenimiento de la aplicación final, ya que existe un débil acoplamiento entre los componentes que forman el software.

5.3.1. Componentes angular

Angular implementa un modelo de desarrollo basado en componentes, el cual es utilizado en este proyecto para el frontend. Los componentes dentro de un proyecto Angular están compuestos por tres partes fundamentales, un template o plantilla HTML, una clase TypeScript y una función decoradora, siendo las dos primeras partes correspondientes a la vista y el controlador, respectivamente, de un modelo MVC, y la tercera, con una función de "pegamento" entre estas dos. Dicha estructura básica de un componente Angular se puede observar en la Figura 5.5.

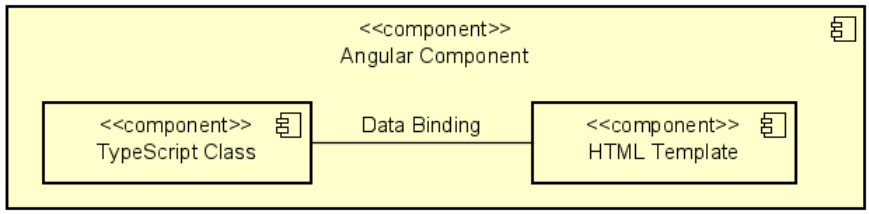


Figura 5.5: Estructura básica de un componente Angular.

La comunicación entre el template y la clase puede realizarse de cuatro maneras diferentes, gracias a estas cuatro "piezas" declarables dentro de la vista, las propiedades, valores que se pueden asignar por medio de un atributo del HTML, tienen un flujo desde el modelo hasta la vista; las expresiones, el volcado de información en el texto de la página, también tienen el mismo flujo que las propiedades, desde el modelo a la vista; el binding, un enlace entre la vista y el modelo, la información fluye en ambos sentidos, si ocurre un cambio en cualquiera de los dos se ve reflejado en el otro y los eventos, sucesos que ocurren y para los cuales se definen manejadores que se ejecutarán como respuesta, los cuales no necesariamente hacen fluir datos, pero si se consideran un flujo de aplicación, en este caso de la vista al modelo.

La forma de declarar cada uno de estas piezas y la dirección del flujo de datos se puede ver en a Figura 5.6.

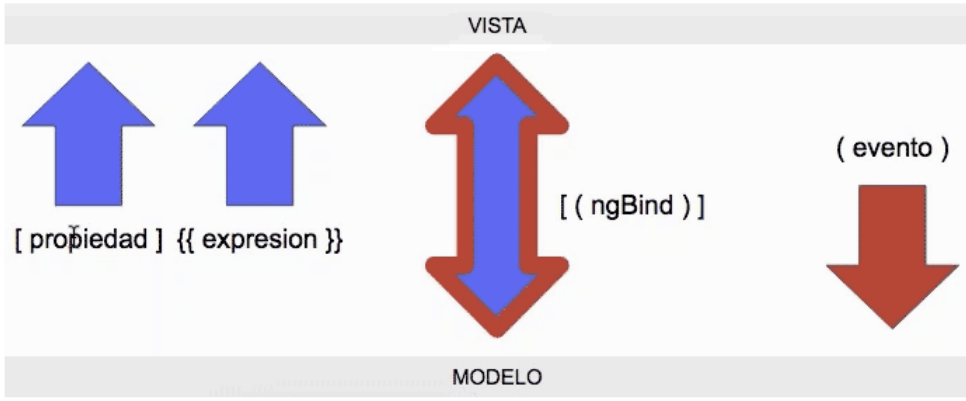


Figura 5.6: Declaración y dirección del flujo de datos entre vista y controlador [100].

Angular también proporciona la posibilidad de reutilizar componentes dentro de otros componentes, creando así una relación padre-hijo entre estos dos. Dicha estructura de componentes con un componente hijo dentro de otro se puede observar en la Figura 5.7, donde se puede ver que el componente hijo se inserta en el template del componente angular padre.

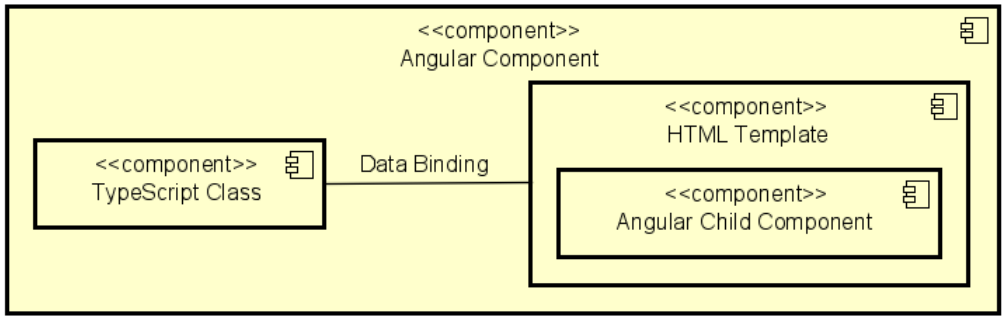


Figura 5.7: Estructura de un componente Angular con un componente hijo.

Los componentes Angular pueden requerir o no servicios para poder realizar su función de manera adecuada. El inyector de servicios se modela como una interfaz requerida por el componente, como se puede observar en la Figura 5.8

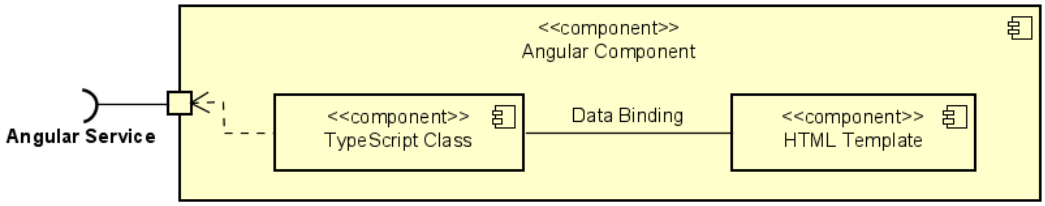


Figura 5.8: Estructura de un componente Angular que requiere un servicio para funcionar.

También existen las directivas en Angular, actúan como un componente reutilizable dentro del template proporcionando a este funcionalidades, como ocultar o mostrar partes. Tanto los servicios como las directivas son componentes, pero no componentes Angular, es decir, piezas reutilizables de software. Esto se puede observar en la Figura 5.9. Algunas de las directivas incorporadas en Angular más usadas en el proyecto son NgIf y NgFor.

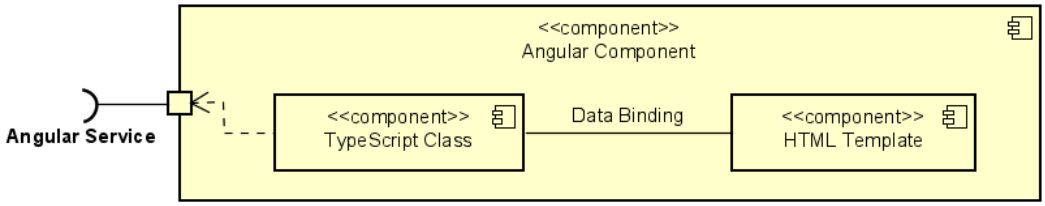


Figura 5.9: Estructura de un componente Angular con directiva.

5.3.2. Componentes creados para el frontend

Para la realización del proyecto se han creado una serie de componentes Angular siguiendo un patrón para su denominación, el cual consiste en la concatenación de dos palabras, la primera describe la acción principal que desempeña el componente (en caso de realizar más de una, se optó por nombrar con la, a primera vista, más frecuente), y la segunda el tipo de entidad sobre el que se realiza dicha acción. Existen dos excepciones, los componentes login y registro, que debido a que es una acción propia de la gran parte de las aplicaciones y plataformas, se optó por nombrar únicamente con la acción y el componente snackbar, el cual hace referencia a una ventana emergente denominada de la misma manera.

Los componentes creados se enumerarán a continuación, describiendo las operaciones que realizan cada uno de ellos. También se puede ver un diagrama en la Figura 5.10 con todos estos componentes que forman el frontend de la plataforma, mostrando las dependencias que existen entre ellos.

- **App**: Contiene la aplicación en su totalidad.
- **ListarEstablecimientos**: Página de inicio de la plataforma de manera predeterminada. Lista los diferentes establecimientos validados registrados en esta para poder seleccionar cualquiera de ellos, pudiendo observar también la ocupación en tiempo real de estos. En caso de que un usuario haya iniciado sesión y sea un propietario validado, podrá visualizar todos sus establecimientos, mostrándose en diferente color dependiendo del estado en el que estén, podrá añadir uno nuevo y editar, gestionar las mesas y trabajadores de los ya registrados. En caso de ser trabajador, podrá navegar directamente hasta las mesas del establecimiento donde trabaja.
- **ListarProductos**: Vista que únicamente pueden ver los propietarios validados, con una lista paginada de los diferentes productos que ha añadido este a la plataforma. Desde esta vista podrá editar y borrar los productos que desee, mediante un proceso de doble confirmación.
- **ListarMesas**: Vista que únicamente pueden ver los propietarios validados y los trabajadores del establecimiento, con una lista paginada de las diferentes mesas que se han añadido al establecimiento registrado en la plataforma. Desde esta vista, el propietario podrá editar y borrar las mesas que desee, mediante un proceso de doble confirmación, podrá activar las mesas oportunas y seleccionar el aforo máximo actual del local. En caso de ser un trabajador del local, solo podrá marcar como libre u ocupada las mesas activas del local y consultar las comandas activas en el momento de cada mesa.
- **ListarCarta**: Página posterior a la selección de un establecimiento donde se podrán visualizar tanto los productos de la carta, que podrán ser filtrados por nombre, como, en caso de existir, el menú que tiene dicha carta. También se podrá consultar la ocupación del local, y mediante un código de mesa, activar la posibilidad de añadir productos a la comanda.
- **ListarTrabajadores**: Vista que únicamente puede ver el propietario validado al que pertenece un establecimiento, con una lista paginada de los diferentes trabajadores que se han añadido al establecimiento registrado en la plataforma.

- **ListarComanda**: Página donde se podrán visualizar los productos de la carta, incluido el menú, que un cliente ha seleccionado para incluir en la comanda a realizar, se mostrará en forma de un resumen del pedido, con la cantidad y el precio total del pedido.
- **ListarComandasMesa**: Página donde se podrán visualizar las comandas activas, cualquier estado que no sea pagado, de una mesa en concreto, otorgando la posibilidad de observar un resumen de la comanda y de pasar de estado a esta, hasta que sea pagada y desaparezca de la vista.
- **CrearEstablecimiento**: Página para la creación y edición de establecimientos por parte de los usuarios con rol de propietario registrados y validados.
- **CrearProductos**: Página para la creación y edición de productos por parte de los usuarios con rol de propietario registrados y validados.
- **CrearMesas**: Página para la creación y edición de mesas de un establecimiento por parte de los usuarios con rol de propietario registrados y validados, a los que pertenezca cada establecimiento.
- **CrearTrabajador**: Página para la creación de los trabajadores de un establecimiento por parte del usuario con rol de propietario registrado y validado al que pertenece dicho local.
- **EditarCarta**: Vista que únicamente pueden ver los propietarios de los establecimientos donde podrán seleccionar, de todos sus productos registrados en la plataforma, los que componen la carta del establecimiento.
- **EditarMenu**: Vista que únicamente pueden ver los propietarios de los establecimientos donde podrán seleccionar, de todos sus productos registrados en la plataforma, los que componen el menú del establecimiento, y asignar tanto el nombre, como el precio y una descripción para este.
- **ValidarEstablecimientos**: Vista que únicamente pueden ver los usuarios con rol de administrador de la plataforma, donde se visualizan las diferentes peticiones de publicación de establecimientos en la plataforma. Aquí se podrán aceptar o rechazar dichas peticiones, ambas operaciones con un proceso de doble confirmación.
- **ValidarPropietarios**: Vista que únicamente pueden ver los usuarios con rol de administrador de la plataforma, donde se visualizan las diferentes peticiones de registro de propietarios en la plataforma. Aquí se podrán aceptar o rechazar dichas peticiones, ambas operaciones con un proceso de doble confirmación.
- **Login**: Página de inicio de sesión para los usuarios registrados en la plataforma.
- **Registro**: Página de registro de usuarios, se da la posibilidad de que elijan si son propietarios, aunque hasta que no sean validados no podrán acceder a las funcionalidades exclusivas de estos.
- **Snackbar**: Componente utilizado para mostrar mensajes emergentes a forma de retroalimentación en respuesta a las operaciones realizadas por los usuarios en la plataforma.

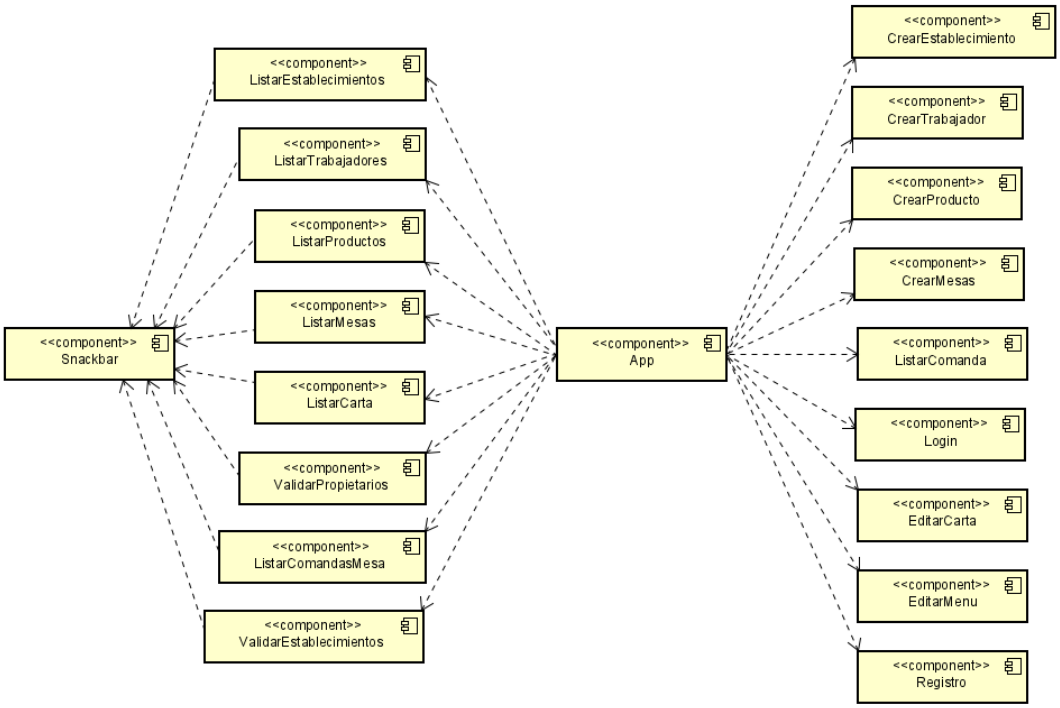


Figura 5.10: Estructura de componentes del frontend.

5.4. Diseño de datos

5.4.1. Base de datos MySQL

El diseño lógico relacional para el esquema de la base de datos MySQL utilizada en este proyecto se puede observar en la Figura 5.11

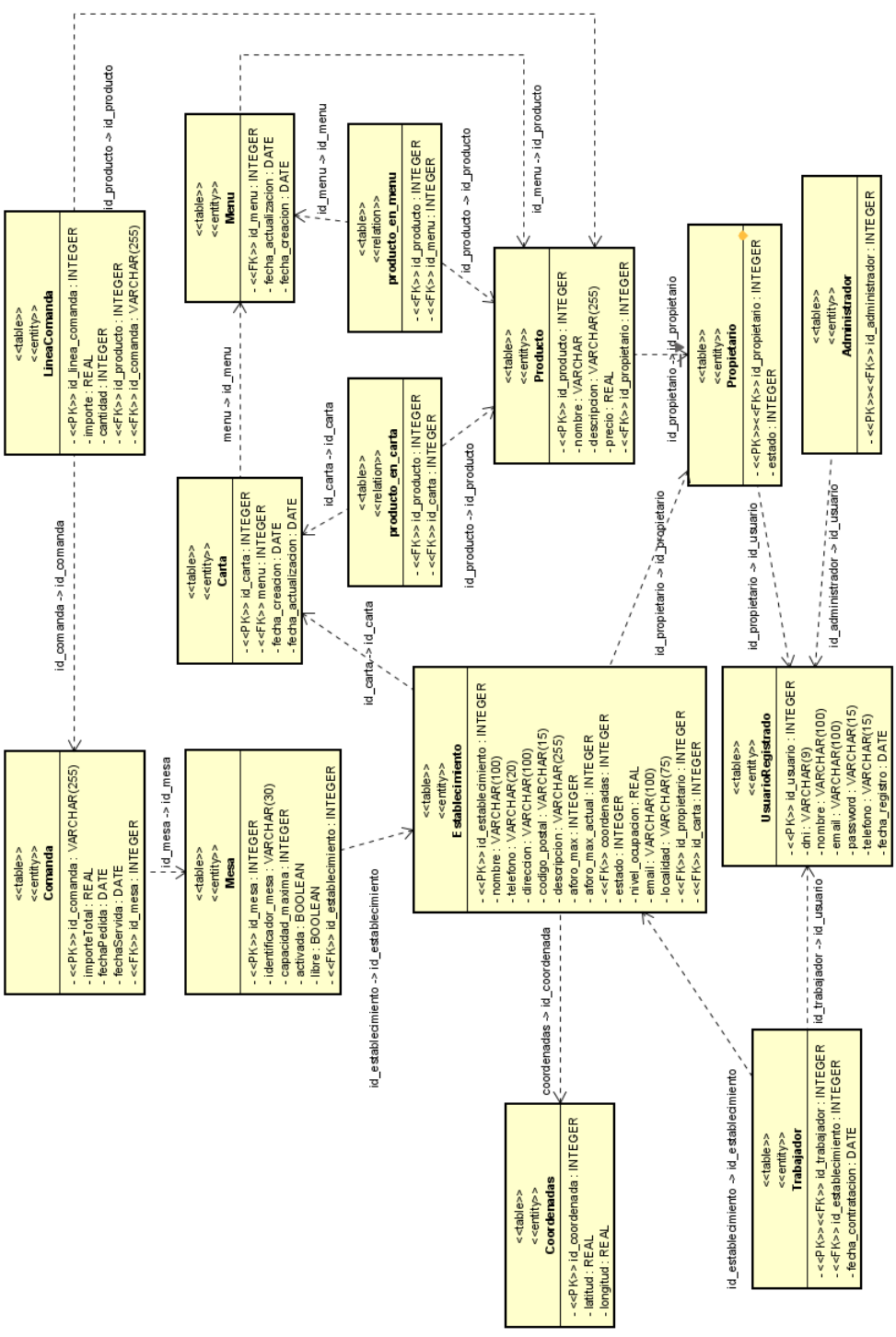


Figura 5.11: Modelo de datos de la aplicación.

5.5. Diseño arquitectónico del backend

La arquitectura general del backend de la aplicación consiste en 4 paquetes principales, aunque alguno de ellos están subdivididos en paquetes. Todos ellos bajo el paquete general *uva.tfgDavidCurieses.backend*. Los paquetes definidos son:

- **Controller**: en este paquete encontramos los controladores del backend, es decir, las clases encargadas de recibir las peticiones lanzadas por el frontend para ser gestionadas. La división en diferentes controladores se ha llevado a cabo pensando en las diferentes entidades que pueden ser solicitadas por un usuario dado el uso de la plataforma.
- **Repository**: en este paquete encontramos las clases encargadas de operar con la información de la base de datos del sistema. En este paquete encontramos una clase "repositorio" por entidad existente en la plataforma.
- **Model**: en este paquete se encuentran las clases de las entidades que forman parte del dominio. Este paquete está separado en dos subpaquetes:
  - **Entities**: en este subpaquete nos encontramos los objetos.
  - **Enum**: en este subpaquete encontramos los dos tipos de datos enumerados.
- **Services**: en este último paquete se han incluido aquellos servicios compartidos por el resto de paquetes del sistema y las clases para la gestión de las conexiones mediante websockets, en este paquete nos encontramos 3 subpaquetes:
  - **Exception**: en este paquete nos encontramos las excepciones creadas para cada uno de los casos posibles que nos podamos encontrar al operar con el sistema. Principalmente nos encontramos dos tipos de excepción:
    - **Exception**: lanzada en caso de que la entidad solicitada o la creación o edición de la misma no pueda ser realizada por alguna razón.
    - **NotFoundException**: lanzada en caso de que la información propuesta para la búsqueda en base de datos de un objeto no sea correcta, no existiendo dicha entidad solicitada.
  - **GeneradorIDs**: en este paquete encontramos las clases encargadas de la generación de identificadores personalizados para las entidades del sistema. En este caso, y dada la profundidad a la que se ha llegado en este proyecto, solo nos encontramos una clase, encargada en generar el identificador para las comandas. En caso de que fuese necesario crear más clases para realizar esta misma función para otras entidades, se alojarían en este paquete.
  - **Sockets**: en este paquete nos encontramos las dos clases encargadas de la gestión de las solicitudes de suscripción, de los diferentes usuarios mediante websockets.

En la Figura 5.12 se puede observar un diagrama presentando la arquitectura general del backend de la plataforma. Los diagramas *Modules&Uses style* de cada uno de los paquetes que forman parte de la arquitectura general del backend se pueden observar en las Figuras 5.13 a 5.16.

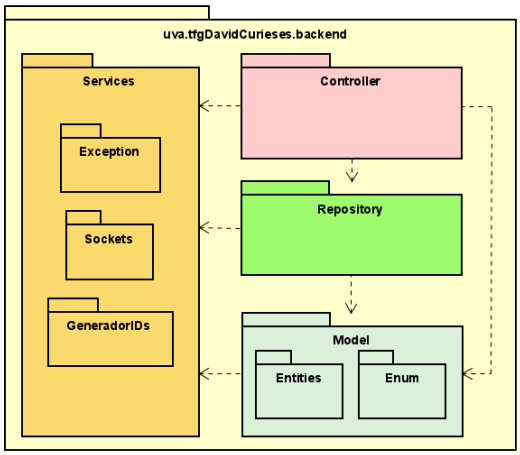


Figura 5.12: Arquitectura general del backend del sistema.

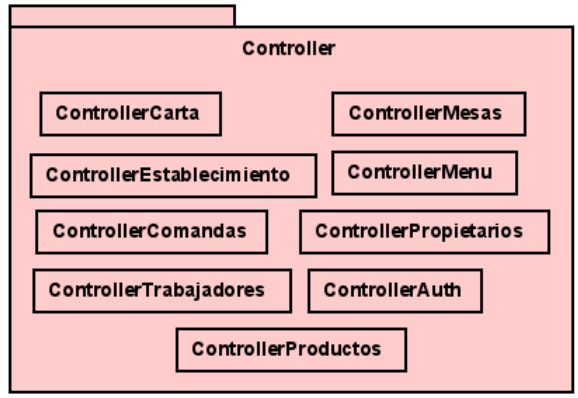


Figura 5.13: Modules&Uses style del paquete controller.

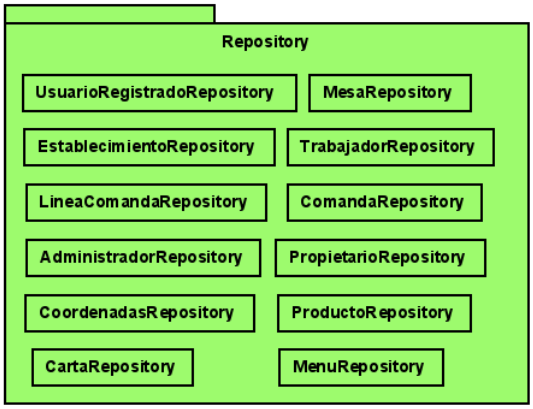


Figura 5.14: Modules&Uses style del paquete repository.

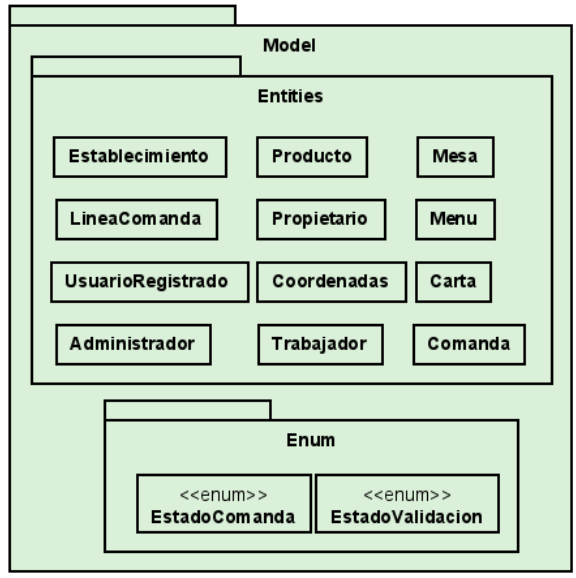


Figura 5.15: Modules&Uses style del paquete model.

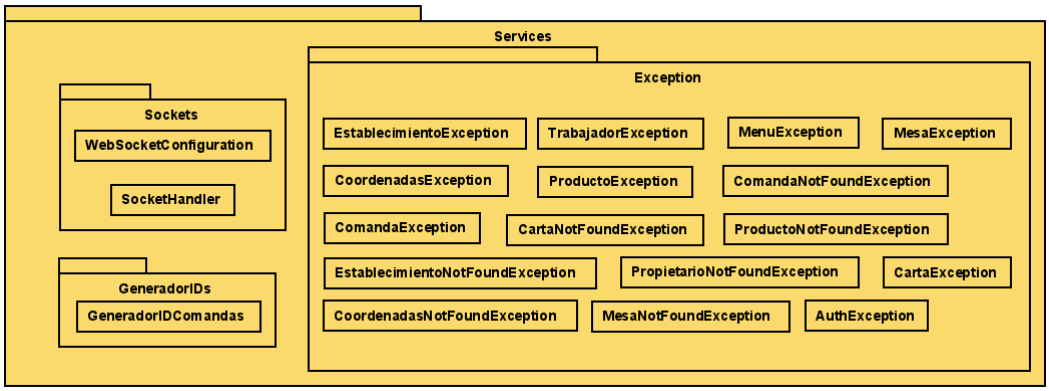


Figura 5.16: Modules&Uses style del paquete services.

5.6. Despliegue de la plataforma

Para realizar el despliegue de la plataforma se ha utilizado el servicio en la nube que proporciona Heroku [54], almacenando, debido a las necesidades del servicio utilizado, en dos proyectos diferentes cada parte del sistema. Además se ha utilizado una extensión proporcionada por Heroku para almacenar la base de datos. El resultado del diagrama de despliegue del sistema se puede observar en la Figura 5.17.

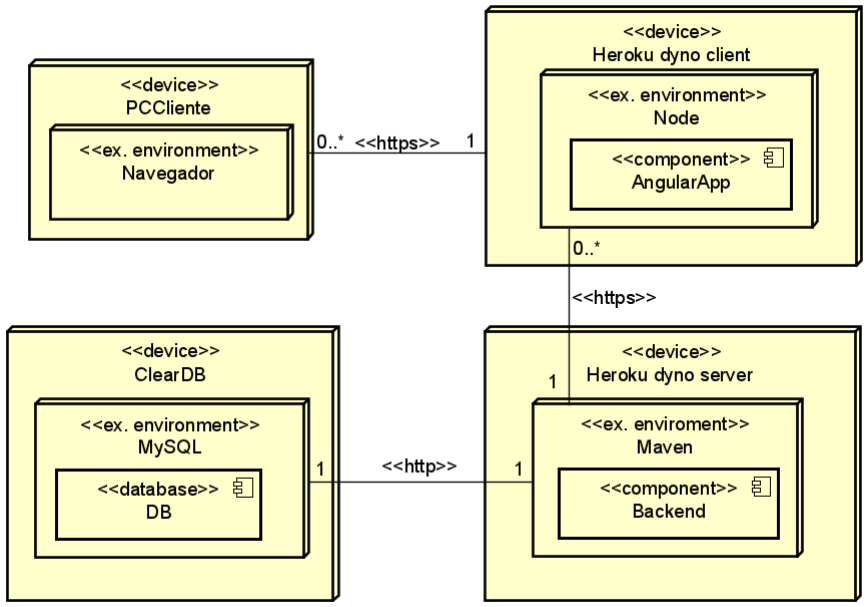


Figura 5.17: Diagrama de despliegue del sistema.

5.7. Bocetos de la interfaz de usuario

Los bocetos de la interfaz de usuario realizados para este proyecto (Figura 5.18 a Figura 5.39) se han desarrollado para dos tamaños de pantalla, todos están diseñados para una ventana del tamaño de un monitor de ordenador y un tamaño más pequeño, aquellas operaciones de gestión de establecimientos o de gestión de la propia plataforma, como las validaciones por parte de un administrador, se han realizado en un tamaño de dispositivo móvil más grande, tipo tableta, mientras que aquellas vistas que tienen que ver con la realización de una comanda, se han diseñado en un tamaño de dispositivo móvil mas pequeño, adecuándose a los dispositivos que podrían llevar la mayoría de clientes.

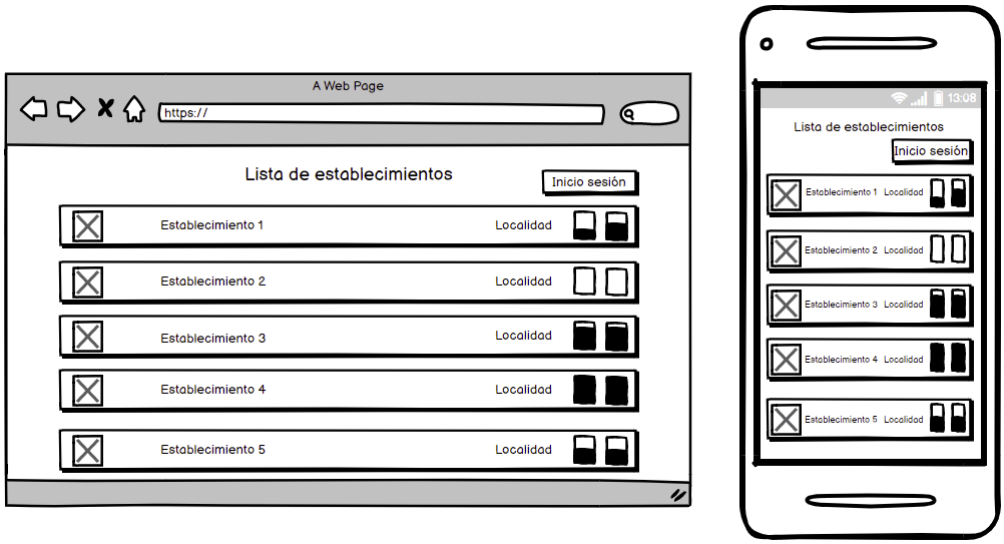


Figura 5.18: Boceto de la vista encargada en mostrar los establecimientos validados registrados en la plataforma para cualquier usuario, mostrando la ocupación con una barra en el lateral.

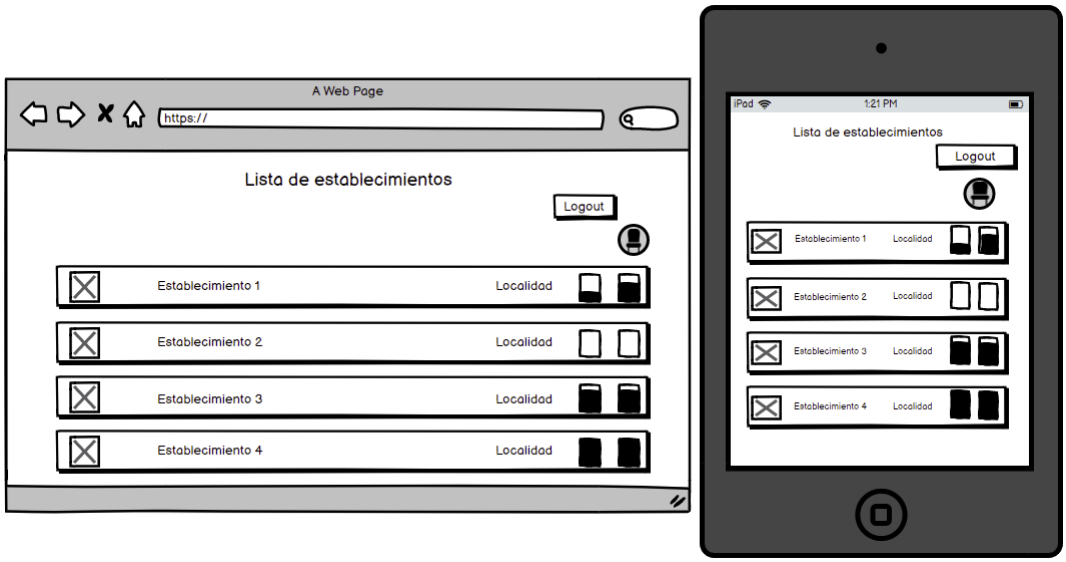


Figura 5.20: Boceto de la vista encargada en mostrar los establecimientos validados registrados en la plataforma para un usuario con rol de Trabajador, apareciendo un botón superior para ir directamente a la gestión de las mesas del establecimiento donde trabaja.

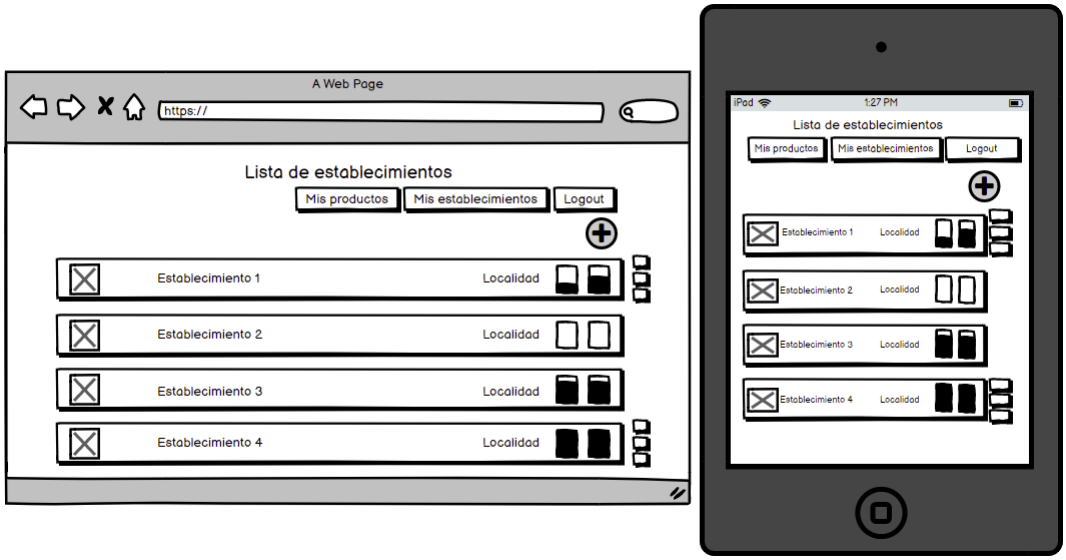


Figura 5.19: Boceto de la vista encargada en mostrar los establecimientos validados registrados en la plataforma para un usuario con rol de Propietario, apareciendo botones con funcionalidad única para este rol a la derecha de sus establecimientos, y uno superior para añadir un nuevo establecimiento.

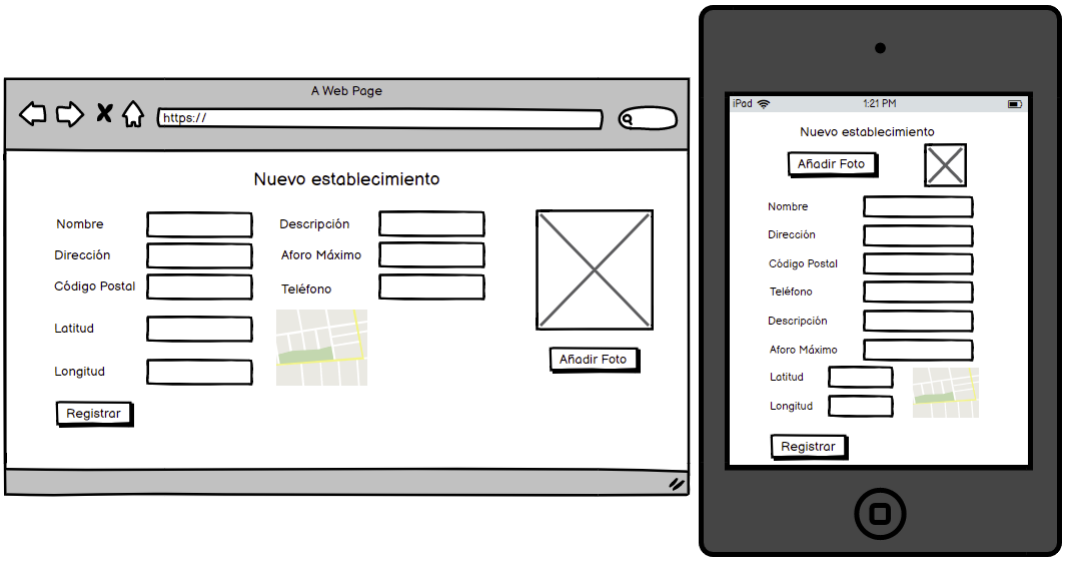


Figura 5.21: Boceto de la vista para crear o editar un establecimiento, solamente visible para usuarios con rol de Propietario.

Figura 5.22: Boceto de la vista encargada en mostrar los productos registrados por un propietario, solamente visible para los usuarios con rol de Propietario.

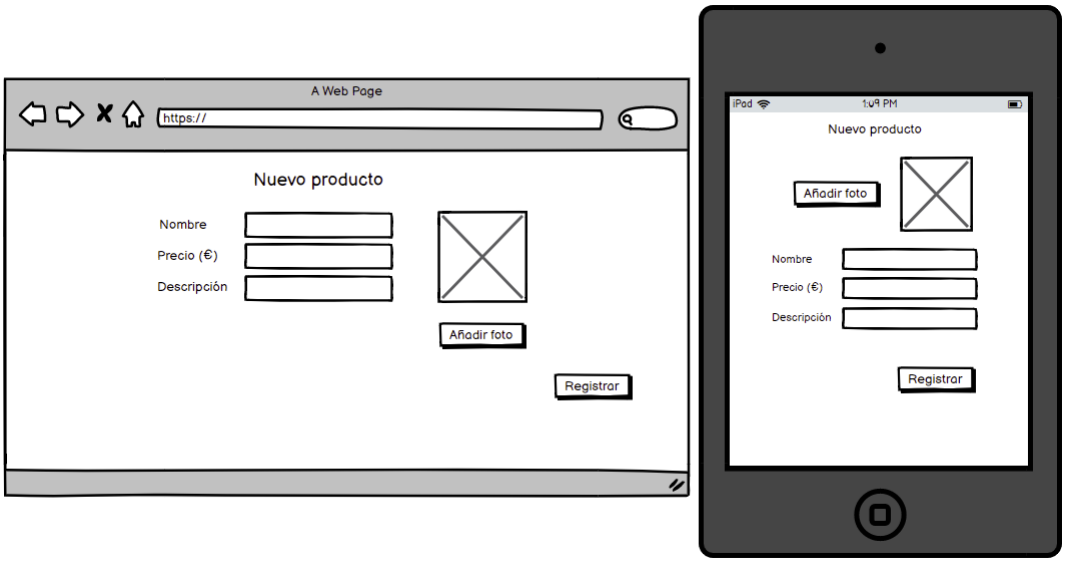


Figura 5.23: Boceto de la vista para crear o editar un producto, solamente visible para los usuarios con rol de Propietario.

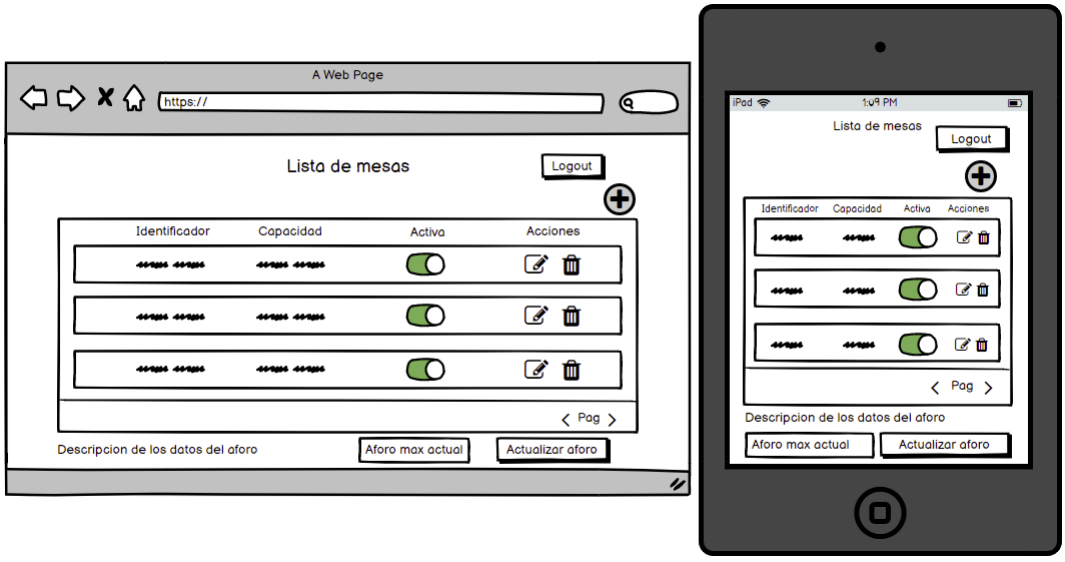


Figura 5.24: Boceto de la vista para usuarios con rol de Propietario encargada en mostrar las mesas registrados en un establecimiento por el propietario de este.

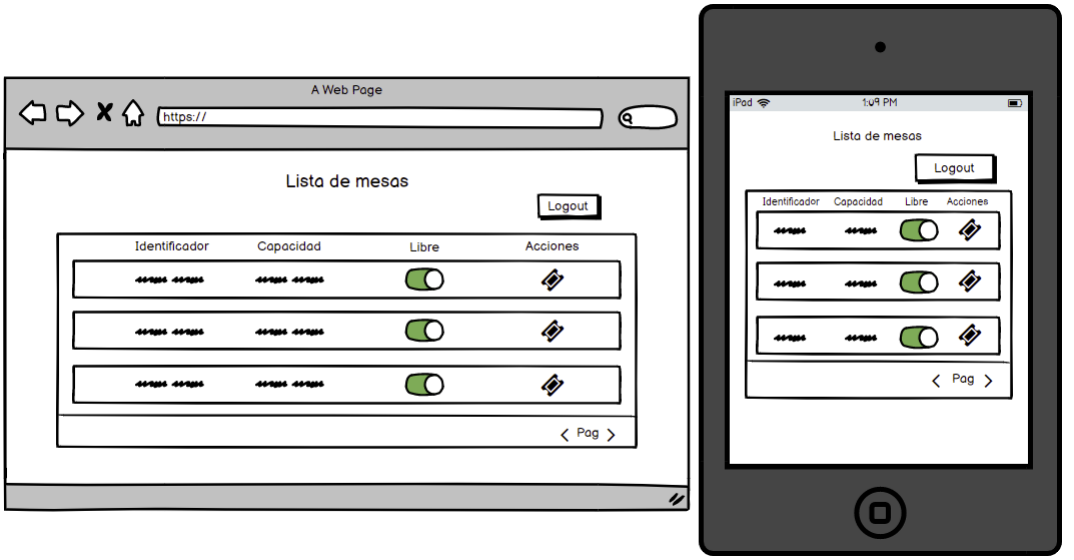


Figura 5.25: Boceto de la vista para usuarios con rol de Trabajador encargada en mostrar las mesas registradas en un establecimiento por el propietario de este.

Figura 5.26: Boceto de la vista para crear o editar una mesa, solamente visible para los usuarios con rol de Propietario.

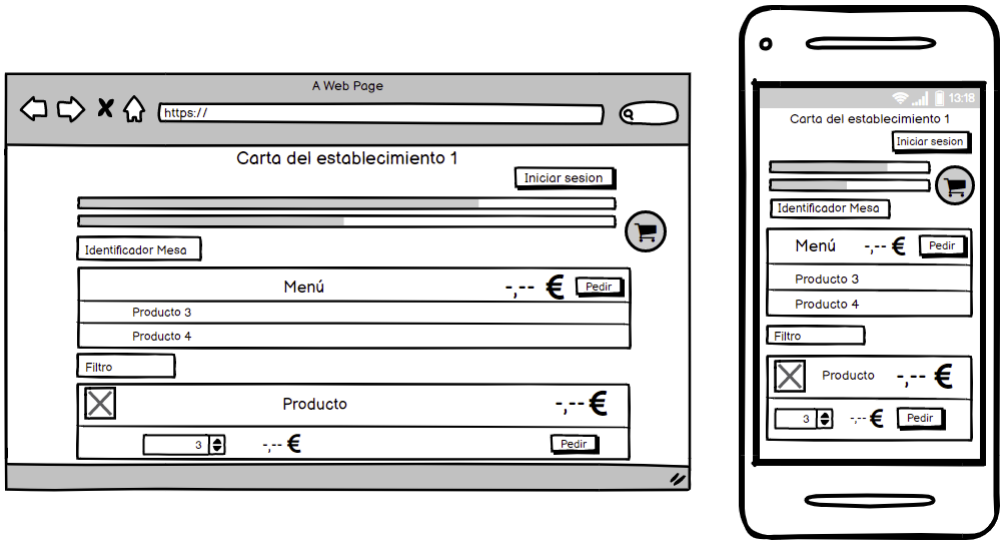


Figura 5.27: Boceto de la vista encargada en mostrar la carta de un establecimiento, marcando con dos barras de progreso la ocupación de aforo y mesas del establecimiento para no perder la información, visible por todo tipo de usuarios.

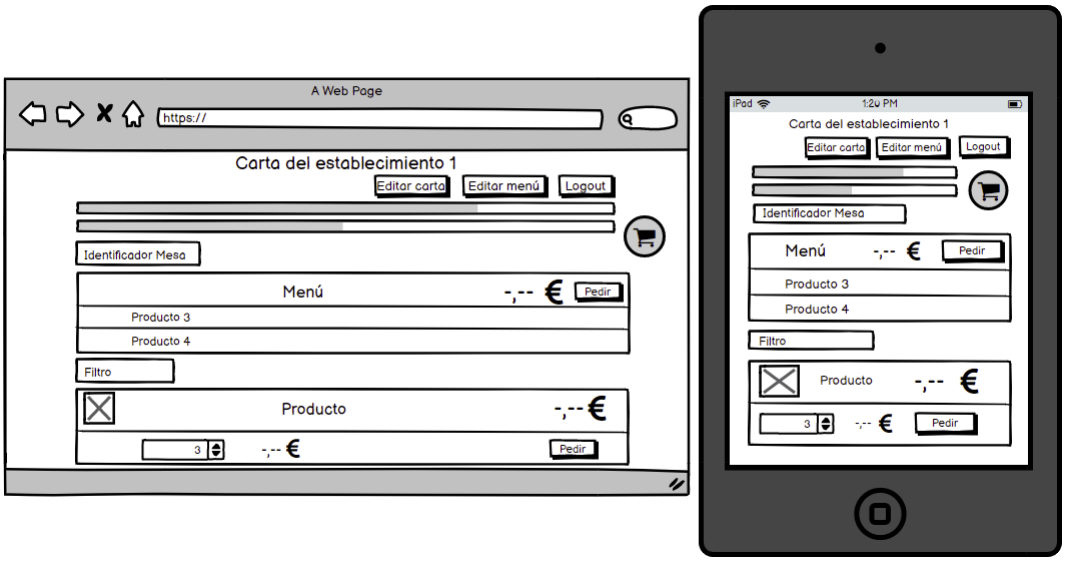


Figura 5.28: Boceto de la vista encargada en mostrar la carta de un establecimiento para el rol de Propietario, mostrando los botones para editar tanto el menú como los productos que forman la carta.

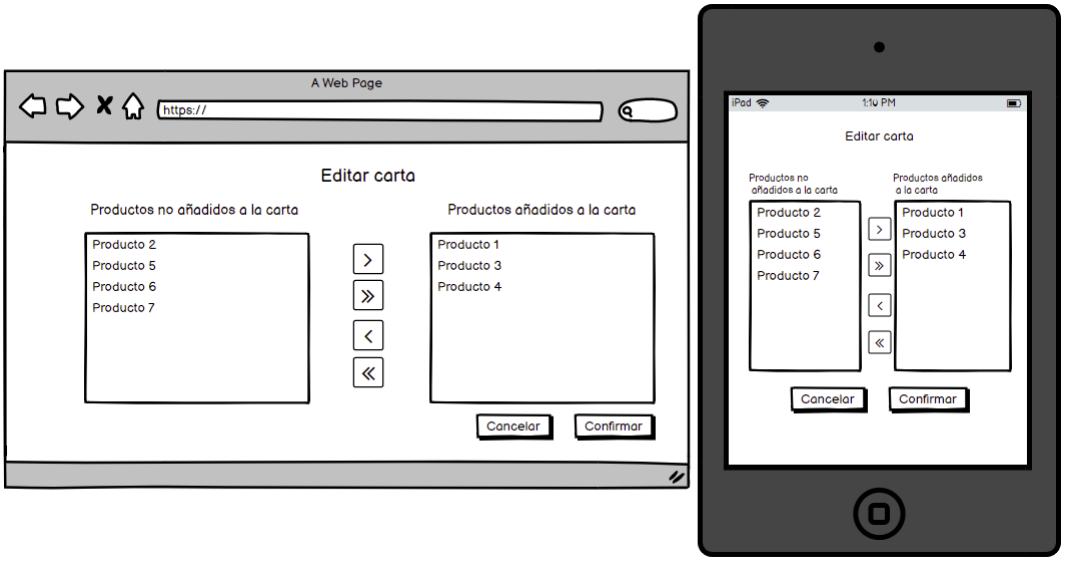


Figura 5.29: Boceto de la vista para editar los productos que componen la carta de un establecimiento, solamente visible para los usuarios con rol de Propietario.

Figura 5.30: Boceto de la vista para editar las características y los productos que componen el menú de un establecimiento, solamente visible para los usuarios con rol de Propietario.

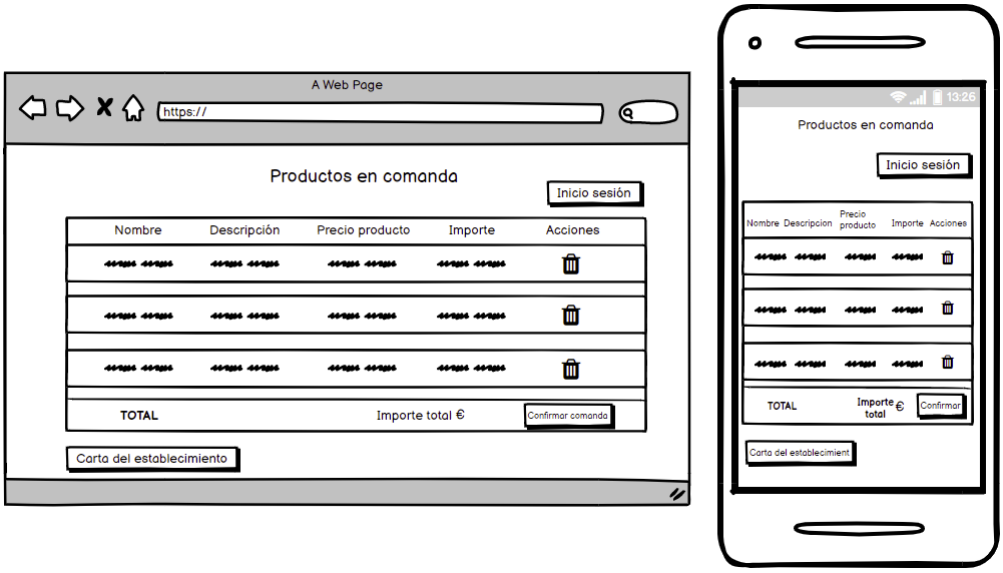


Figura 5.31: Boceto de la vista encargada en mostrar productos seleccionados por un cliente para realizar una comanda.

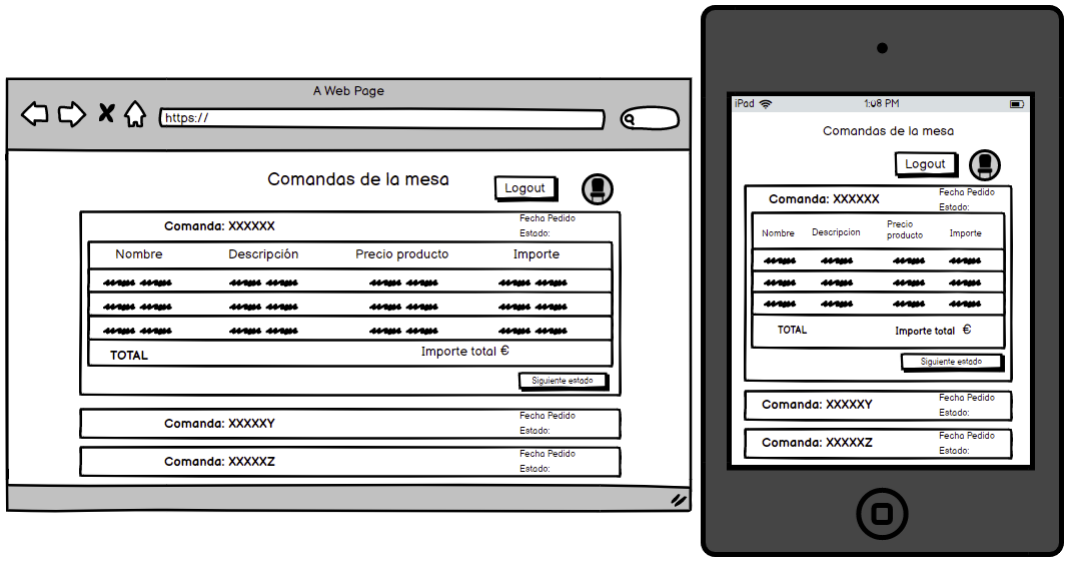


Figura 5.32: Boceto de la vista encargada en mostrar las comandas activas en el momento para cada mesa, solamente accesible por los usuarios con rol Trabajador del establecimiento.

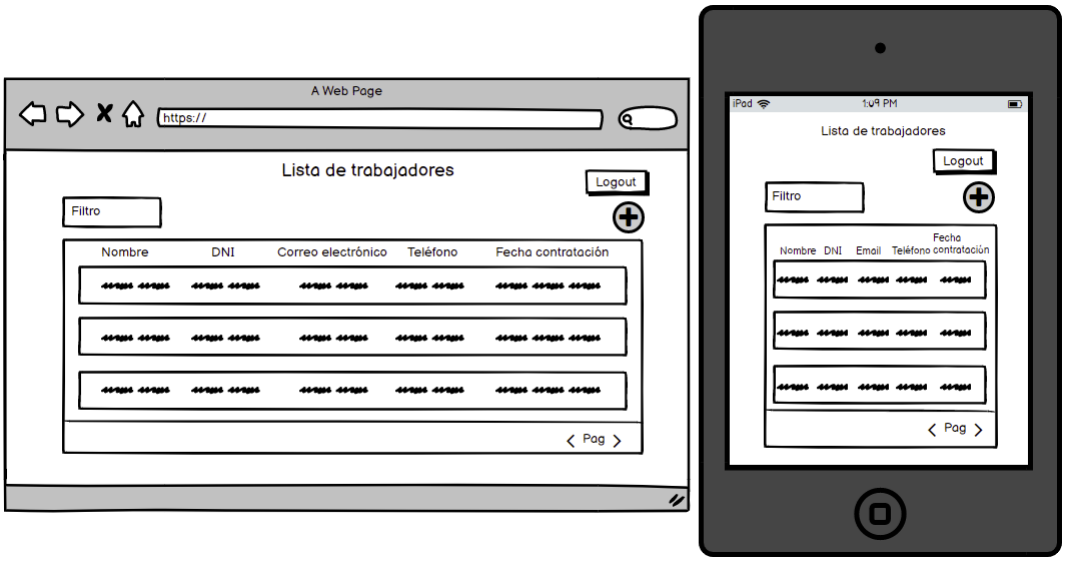


Figura 5.33: Boceto de la vista encargada en mostrar los trabajadores de un establecimiento registrados por un propietario, solamente visible para los usuarios con rol de Propietario.

Figura 5.34: Boceto de la vista para añadir un trabajador nuevo al establecimiento, solamente visible para los usuarios con rol de Propietario.

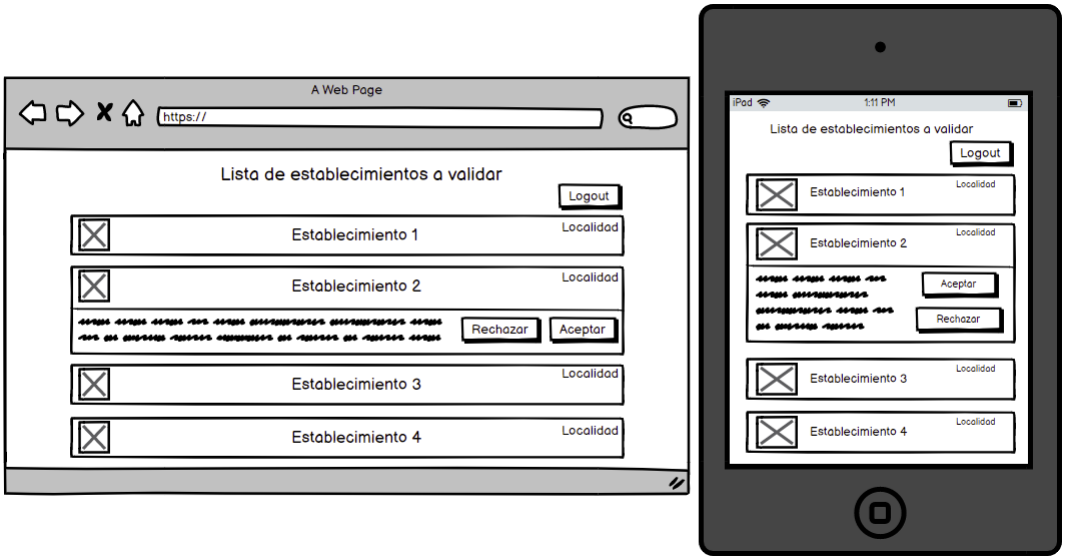


Figura 5.35: Boceto de la vista para gestionar las peticiones de registro de establecimientos, solamente visible para los usuarios con rol de administrador.

Figura 5.36: Boceto de la vista para gestionar las peticiones de registro de propietarios, solamente visible para los usuarios con rol de administrador.

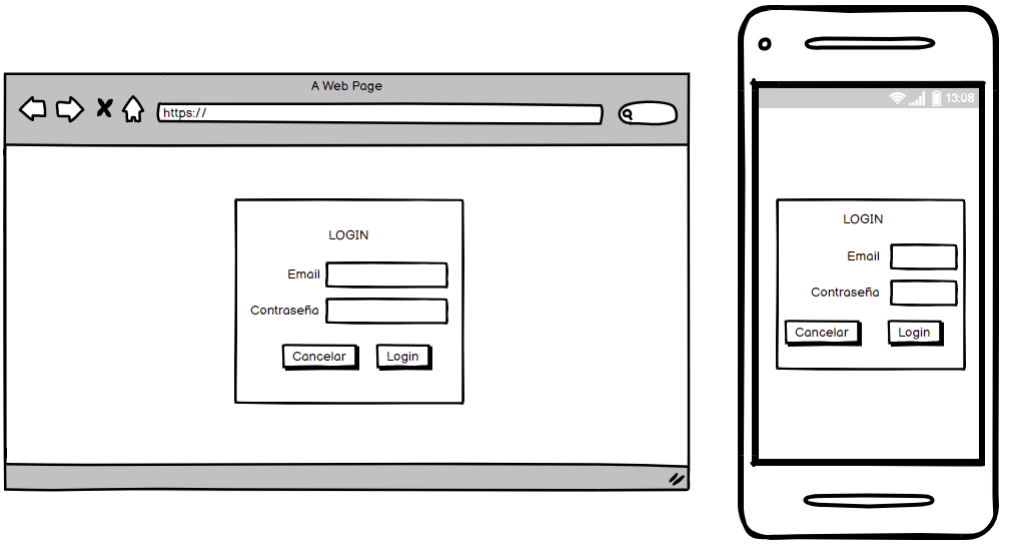


Figura 5.37: Boceto de la vista para el inicio de sesión por parte de los usuarios.

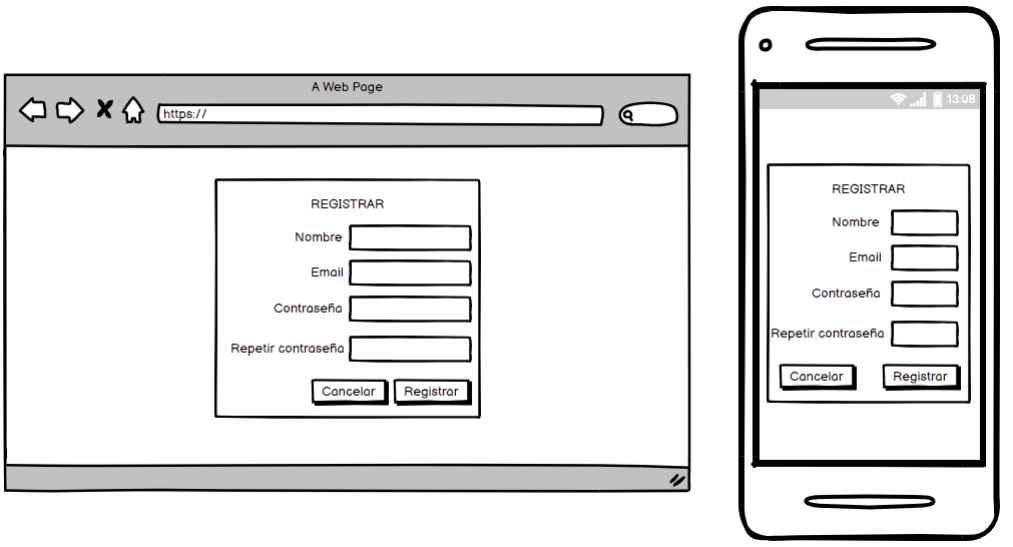


Figura 5.38: Boceto de la vista para el registro en la plataforma por parte de los usuarios.

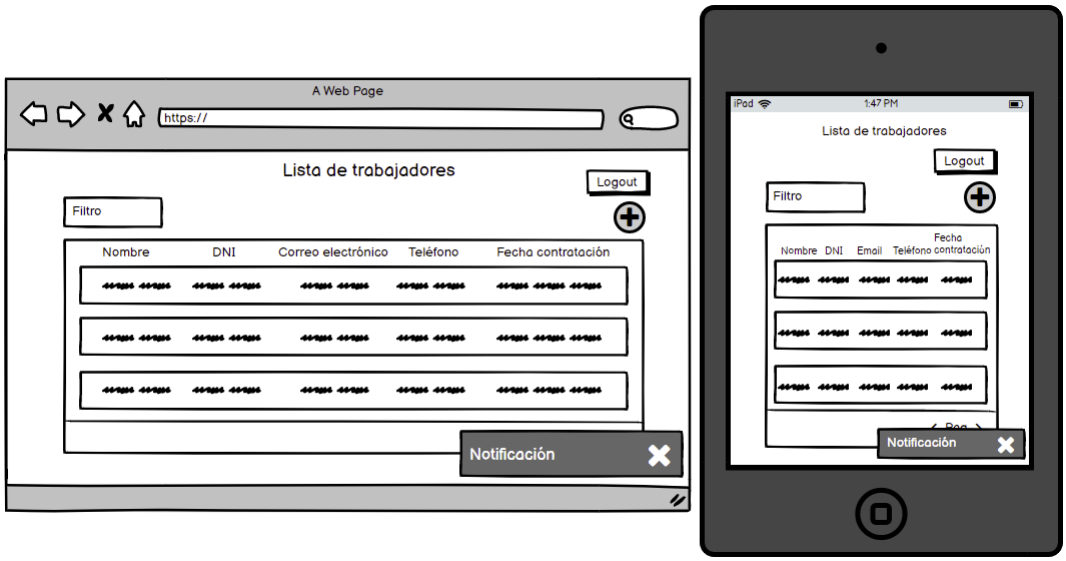


Figura 5.39: Boceto de una vista con un snackbar emergente en la esquina inferior derecha para mostrar retroalimentación sobre una operación realizada.

Capítulo 6

Implementación y pruebas

Tras haber visto tanto el análisis como el diseño de la plataforma, en esta capítulo vamos a describir la implementación de la misma. Describiremos la estructura final del código, tanto del servidor como del cliente, utilizado para el desarrollo del proyecto y algunos de los problemas más recurrentes encontrados.

Por último describiremos las pruebas llevadas a cabo para garantizar el buen funcionamiento de la plataforma.

6.1. Estructura del código del backend

Aquí se muestra la estructura final del código del backend de la plataforma. Posteriormente se va a desarrollar textualmente qué encontramos en cada directorio.

```
| - - .mvn
+ - - src
      + - - main
            + - - java/uva/tfgDavidCurieses
                  + - - Controller
                        | - - ControllerAuth.java
                        | - - ControllerCarta.java
                        | - - ControllerComandas.java
                        | - - ControllerEstablecimiento.java
                        | - - ControllerMenu.java
                        | - - ControllerMesas.java
                        | - - ControllerProductos.java
                        | - - ControllerPropietarios.java
                        | - - ControllerTrabajadores.java
                  + - - Model
                        + - - Entities
                              | - - Administrador.java
                              | - - Carta.java
```

```
                        | - - Comanda.java
                        | - - Coordenadas.java
                        | - - Establecimiento.java
                        | - - LineaComanda.java
                        | - - Menu.java
                        | - - Mesa.java
                        | - - Producto.java
                        | - - Propietario.java
                        | - - Trabajador.java
                        | - - UsuarioRegistrado.java
                + - - Enum
                        | - - EstadoComanda.java
                        | - - EstadoValidacion.java
            + - - Repository
                    | - - AdministradorRepository.java
                    | - - CartaRepository.java
                    | - - ComandaRepository.java
                    | - - CoordenadasRepository.java
                    | - - EstablecimientoRepository.java
                    | - - LineaComandaRepository.java
                    | - - MenuRepository.java
                    | - - MesaRepository.java
                    | - - ProductoRepository.java
                    | - - PropietarioRepository.java
                    | - - TrabajadorRepository.java
                    | - - UsuarioRegistradoRepository.java
            + - - Services
                + - - Exception
                        | - - AuthException.java
                        | - - CartaException.java
                        | - - CartaNotFoundException.java
                        | - - ComandaException.java
                        | - - ComandaNotFoundException.java
                        | - - CoordenadaException.java
                        | - - CoordenadasNotFoundException.java
                        | - - EstablecimientoException.java
                        | - - EstablecimientoNotFoundException.java
                        | - - MenuException.java
                        | - - MesaException.java
                        | - - MesaNotFoundException.java
                        | - - ProductoException.java
                        | - - ProductoNotFoundException.java
                        | - - PropietarioNotFoundException.java
                        | - - TrabajadorException.java
                + - - GeneradorIDs
                        | - - GeneradorIDComandas.java
                + - - Sockets
                        | - - SocketHandler.java
                        | - - WebSocketConfiguration.java
            | - - BackendApplication.java
        + - - resources
            | - - application.properties
| - - .gitignore
| - - mvnw
| - - mvnw.cmd
| - - pom.xml
```

El directorio **src** está compuesto por el subdirectorio **main**, el cual está subdividido en otros dos directorios:

- El primero de ellos recibe el nombre **java**, el cual contiene la carpeta **uva**, que a su vez contiene el directorio **tfgDavidCurieses**, el cual, finalmente, contiene el código fuente del backend de la plataforma. Este directorio está formado por 4 directorios, el primero contiene los controladores del servidor, encargados de recibir las peticiones HTTP lanzadas desde el cliente para ser gestionadas de forma correcta; el directorio **Model** contiene los modelos de datos del dominio del sistema, este está dividido a su vez en otros dos subdirectorios, el primero, **Entities**, encargado de las entidades en sí de la plataforma y el segundo, **Enum**, encargado de los tipos de datos enumerados, que proporcionan mayor información a los primeros; el directorio **Repository** donde encontramos las clases encargadas de las operaciones sobre la base de datos; el subdirectorio **Services**, con los servicios comunes para el resto de subdirectorios de la plataforma, en el podemos encontrar tres directorios, el primero, **Exception**, contiene las diferentes excepciones que pueden ser lanzadas durante la ejecución de la acciones sobre la plataforma, el subdirectorio **GeneradorIDs** que contiene los generadores de identificadores personalizados para las diferentes entidades del sistema, en el caso de este proyecto solo contaremos con el generador para las comandas; y por último el directorio encargado de manejar las suscripciones a los "temas" mediante web sockets. Por último encontramos el archivo "BackendApplication.java" encargado de ejecutar el backend del sistema.
- El segundo recibe el nombre de **resources**, el cual solo contiene el archivo application.properties, encargado de mantener las diferentes propiedades para ejecutar el sistema en diferentes entornos dentro de un solo fichero.

6.2. Estructura del código del frontend

Aquí se muestra la estructura final del código del frontend de la plataforma, como se puede observar, hay ciertos directorios de los que no se muestra el contenido por brevedad o por similitud con otros anteriores. Posteriormente se va a desarrollar textualmente que encontramos en cada directorio.

```
+ - - node_modules (módulos de Node usados, no se despliega por brevedad)
+ - - src
      + - - app
            + - - crear-establecimiento
                  | - - crear-establecimiento.component.css
                  | - - crear-establecimiento.component.html
                  | - - crear-establecimiento.component.spec.ts
                  | - - crear-establecimiento.component.ts
            + - - crear-mesas (misma estructura que el directorio crear-establecimiento)
            + - - crear-producto (misma estructura que el directorio crear-establecimiento
            + - - crear-trabajador (misma estructura que el directorio crear-establecimiento)
            + - - editar-carta (misma estructura que el directorio crear-establecimiento)
            + - - editar-menu (misma estructura que el directorio crear-establecimiento)
            + - - listar-carta (misma estructura que el directorio crear-establecimiento)
```

```
            + - - listar-comanda (misma estructura que el directorio crear-establecimiento)
            + - - listar-comandas-mesa (misma estructura que el directorio crear-establecimiento)
            + - - listar-establecimientos (misma estructura que el directorio crear-establecimiento)
            + - - listar-mesas (misma estructura que el directorio crear-establecimiento)
            + - - listar-productos (misma estructura que el directorio crear-establecimiento)
            + - - listar-trabajadores (misma estructura que el directorio crear-establecimiento)
            + - - login (misma estructura que el directorio crear-establecimiento)
            + - - registro (misma estructura que el directorio crear-establecimiento)
            + - - shared
                  | - - app-models.ts
                  | - - auth.guard.spec.ts
                  | - - auth.guard.ts
                  | - - auth.service.spec.ts
                  | - - auth.service.ts
                  | - - client-api.service.spec.ts
                  | - - client-api.service.ts
                  | - - comanda.service.spec.ts
                  | - - comanda.service.ts
                  | - - data.service.spec.ts
                  | - - data.service.ts
                  | - - jwt.interceptor.spec.ts
                  | - - jwt.interceptor.ts
                  | - - web-socket.service.spec.ts
                  | - - web-socket.service.ts
            + - - snackbar (misma estructura que el directorio crear-establecimiento)
            + - - validar-establecimientos (misma estructura que el directorio crear-establecimiento)
            + - - validar-propietarios (misma estructura que el directorio crear-establecimiento)
            | - - app-routing.module.ts
            | - - app.component.css
            | - - app.component.html
            | - - app.component.spec.ts
            | - - app.component.ts
            | - - app.module.ts
      + - - assest
            | - - .gitkeep
            | - - blue-marker.png
      + - - enviroments
            | - - enviroment.prod.ts
            | - - enviroment.ts
      | - - favicon
      | - - index.html
      | - - main.ts
      | - - polyfills.ts
      | - - styles.css
      | - - test.ts
| - - .browserslistrc
| - - .editorconfig
| - - .gitignore
| - - angular.json
| - - karma.conf.js
| - - package-lock.json
| - - package.json
| - - README.md
| - - tsconfig-app.json
| - - tsconfig.json
| - - tsconfig.spec.json
```

El directorio **src** contiene el código fuente del frontend de la plataforma (directorio **app**), así como los archivos de recursos (directorio **assets**), que en nuestro caso se compone únicamente de una imagen como marcador de coordenadas para el mapa, y las variables de entorno que cambian dependiendo de si se compila el proyecto en modo producción o en modo de desarrollo (directorio **enviroments**). Este directorio también contiene la página principal de la plataforma (fichero index.html), el archivo TypeScript para la página inicial (fichero main.ts) y el fichero de estilos general de la aplicación (styles.css), entre otros archivos.

Más en profundidad, el subdirectorio **app**, comentado anteriormente, contiene todos los componentes que forman la plataforma, que a su vez están compuestos por un template (archivo con extensión .html), una hoja de estilos (archivo con extensión .css), la clase TypeScript para la lógica del componente (archivo con extensión .ts) y un fichero para test del componente (archivo con extensión .spec.ts). También está formado por el subdirectorio **shared**, el cual contiene los archivos compartidos por los componentes, como los modelos de datos de las entidades que forman parte del dominio del sistema (app.models), aquellos usados para el control de navegación por la plataforma (auth service, auth guard y jwt interceptor), la realización de peticiones HTTP al servidor (client-api service), los servicios de datos para almacenar información compartida por componentes (comanda y data services) y los servicios para las conexiones vía web socket con el servidor (web-socket service). El resto de ficheros de este directorio **app** constituyen el componente principal de la plataforma.

6.3. Problemas y dificultades superadas

Aunque los problemas se han ido describiendo a lo largo del seguimiento del proyecto que queda reflejado en el Capítulo 7, en este apartado se van a reflejar de forma directa aquellos problemas más importantes que han surgido en el desarrollo del proyecto

- **Envío de datos desde el servidor**: Un problema muy recurrente a lo largo de la realización de este proyecto fue la forma en la cual el servidor Spring Boot respondía a las peticiones HTTP realizadas desde el cliente. Para evitar bucles infinitos en el JSON de respuesta a las peticiones, debido a las relaciones entre los diferentes objetos del modelo de dominio, Spring Boot identifica los objetos ya mapeados únicamente con su identificador único dentro de la base de datos, por lo que, en caso de que cierto objeto haya sido ya mapeado, es necesario asignarle su valor como objeto, en vez del identificador.
- **Uso de componentes no conocidos**: Debido a la extensión del proyecto y al uso de diferentes componentes, de librerías como Angular Material, muchos aspectos utilizados en el desarrollo de las diferentes funcionalidades de la plataforma eran desconocidos, ya que ciertos aspectos desarrollados o utilizados no se habían utilizado con anterioridad en las asignaturas del grado que abordan las tecnologías utilizas. Por lo que aprender el uso de estos conceptos ha requerido búsquedas en documentación oficial y los diferentes foros en Internet.
- **Uso de web sockets**: La implementación de web sockets para la comunicación de cambios a los diferentes suscriptores de cada uno de los dos "temas" existentes en el

servidor, ha supuesto una investigación profunda en el funcionamiento de esta tecnología para los lenguajes utilizados, Java en el servidor y TypeScript en el cliente. Como se creó primero uno de los dos temas, la implementación del segundo no supuso gran esfuerzo, debido a su similitud con el primero, pero debido a que hacía mucho tiempo que no usaba esta tecnología y el desconocimiento de su uso en los lenguajes mencionados, requirió una búsqueda de información sobre su aplicación, resultando en un aprendizaje nuevo.

- **Personalización del identificador de una entidad en la base de datos**: Como antes de empezar con el TFG sólo había trabajado con bases de datos que utilizan la tecnología JPA en una ocasión, en una asignatura del grado, no conocía, ni tenía práctica, sobre la asignación de identificadores personalizados a las entidades de la base de datos. En un comienzo no se iba a implementar dicha personalización, pero debido a que era una gran mejoría respecto a la idea inicial, se decidió invertir tiempo en la investigación de como llevarlo a cabo. Aunque la personalización realizada es mejorable, la posibilidad de asociar una comanda con el establecimiento, la mesa y el número de comanda dentro de esa mesa, supone una gran mejoría para la comprensión de este identificador a los usuarios de la plataforma.
- **Despliegue**: El despliegue de la plataforma supuso un problema en los últimos días de realización del proyecto, ya que nunca antes había desplegado una plataforma. El principal problema encontrado fue que la herramienta utilizada para el despliegue, Heroku, necesita conocer el lenguaje utilizado en el proyecto para funcionar correctamente, y en el repositorio de GitLab utilizado para la gestión del mismo se encontraban tanto el servidor como el cliente, por lo que no se podía desplegar. Para solucionar esto se decidió separar en dos repositorios diferentes el backend del frontend, teniendo cada uno su despliegue por separado.

6.4. Pruebas

Las pruebas llevadas a cabo para este proyecto van a ser descritas textualmente en este apartado del Capítulo 6. En dichas pruebas se van a abarcar todos los casos en los que puede desembocar la plataforma debido a su uso por un usuario, describiendo las acciones realizadas (se pondrán ejemplos de que hay que introducir en los campos de texto, o acciones sobre botones), precondiciones y resultados en cada caso, remarcando aquellas pruebas que hayan descubierto algún error en la plataforma.

Para todas las pruebas se establece como precondición general que la plataforma esté arrancada y para cada componente probado se establecen una serie de precondiciones generales para todas las pruebas de cada uno, y si es necesario, se indican en cada prueba en específico más de estas precondiciones.

6.4.1. Registro de usuarios

Precondiciones	▪ El usuario se encuentra en la pagina de registro de la plataforma */registro*.

Tabla 6.1: Precondiciones para el registro de usuarios.

Titulo	Registro de usuario con rol propietario correcto.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "password" y finalmente activará el registro como Propietario, pulsará el botón de registro, finalizando el proceso de registro de forma exitosa.
Resultado	▪ El usuario accederá a la plataforma con la funcionalidad reducida al no haber sido validado, mostrando un mensaje. ▪ El usuario ya podrá iniciar sesión en la plataforma. ▪ La petición de registro del propietario llegará a los administradores para su gestión.

Tabla 6.2: Prueba P01: Registro de usuario con rol propietario correcto.

Titulo	Registro de usuario repetido.
Precondiciones	▪ Existe un usuario ya registrado con el correo electrónico *"correoP@correo.com"*.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "password", sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará al estar el correo repetido, manteniéndose en la vista de registro y mostrando un mensaje de que el usuario ya está registrado.

Tabla 6.3: Prueba P02: Registro de usuario repetido.

Titulo	Registro de usuario contraseñas no iguales.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña "password" y en la repetición de la contraseña escribirá "wordpass", sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará al no coincidir las contraseñas, manteniéndose en la vista de registro y mostrando un mensaje de que las contraseñas no coinciden .

Tabla 6.4: Prueba P03: Registro de usuario contraseñas no iguales.

Titulo	Registro con campo nombre no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto el nombre, como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "password" y sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo al nombre en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.5: Prueba P04: Registro con campo nombre no rellenado.

Titulo	Registro con campo DNI no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto el DNI, como nombre introduce "David Curieses", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "password" y sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo al DNI en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.6: Prueba P05: Registro con campo DNI no rellenado.

Titulo	Registro con campo teléfono no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto el teléfono, en el nombre introduce "David Curieses", como DNI "12345678A", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "password" y sin importar el estado del checkbox de registro como Propietario, para pulsar el botón de registro, finalizando el proceso de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo al teléfono en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.7: Prueba P06: Registro con campo teléfono no rellenado.

Titulo	Registro con campo correo no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto el correo electrónico, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", escribirá como contraseña en ambos campos "password" y sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo al correo electrónico en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.8: Prueba P07: Registro con campo correo no rellenado.

Titulo	Registro con campo contraseña no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto la contraseña, en el nombre introduce "David Curieses", como DNI "12345678A", como correo electrónico "correoP@correo.com", escribirá como contraseña en el campo de la repetición de esta "password" y sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo a la contraseña en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.9: Prueba P08: Registro con campo contraseña no rellenado.

Titulo	Registro con campo repetición de contraseña no rellenado.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, excepto la repetición de la contraseña, en el nombre introduce "David Curieses", como DNI "12345678A", como correo electrónico "correoP@correo.com", escribirá como contraseña en el primer campo "password" y sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo a la repetición de la contraseña en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.10: Prueba P09: Registro con campo repetición de contraseña no rellenado.

Titulo	Registro con campo correo electrónico no cumple el patrón.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo", escribirá como contraseña en ambos campos "password", sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo al correo en rojo y mostrando un mensaje, indicando que algo está mal en el correo introducido.

Tabla 6.11: Prueba P10: Registro con campo correo electrónico no cumple el patrón.

Titulo	Registro con campo contraseña con menos de 8 caracteres.
Descripción	El usuario que quiere registrarse como propietario introducirá los datos pedidos en los campos del formulario, en el nombre introduce "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789", como correo electrónico "correoP@correo.com", escribirá como contraseña en ambos campos "passwor", sin importar el estado del checkbox de registro como Propietario, pulsa el botón de registro.
Resultado	▪ El usuario no se registrará, marcando el campo relativo a la contraseña en rojo y mostrando un mensaje, indicando que algo está mal con la contraseña introducida.

Tabla 6.12: Prueba P11: Registro con campo contraseña con menos de 8 caracteres.

6.4.2. Inicio de sesión de usuarios

Precondiciones	▪ El usuario se encuentra en la pagina de inicio de sesión de la plataforma */login*. ▪ Existe un usuario registrado con correo electrónico "correo@correo.com" y contraseña "password"

Tabla 6.13: Precondiciones para el inicio de sesión.

Titulo	Inicio de sesión de usuario correcto.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, como correo electrónico "correo@correo.com" y como contraseña "password" y tras pulsar el botón de inicio de sesión, finaliza el proceso de login de forma exitosa.
Resultado	▪ El usuario accederá a la plataforma.

Tabla 6.14: Prueba P12: Inicio de sesión de usuario correcto.

Titulo	Inicio de sesión con campo correo electrónico no rellenado.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, excepto el correo electrónico. Introduciendo como contraseña "password" y finalmente pulsará el botón de inicio de sesión.
Resultado	▪ El usuario no iniciará sesión, marcando el campo relativo al correo electrónico en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.15: Prueba P13: Inicio de sesión con campo correo electrónico no rellenado.

Titulo	Inicio de sesión con campo contraseña no rellenado.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, excepto la contraseña. Introduciendo como correo electrónico "correo@correo.com" y finalmente pulsará el botón de inicio de sesión.
Resultado	▪ El usuario no iniciará sesión, marcando el campo relativo a la contraseña en rojo y mostrando un mensaje, indicando que es necesario rellenarlo.

Tabla 6.16: Prueba P14: Inicio de sesión con campo contraseña no rellenado.

Titulo	Inicio de sesión de usuario con correo electrónico incorrecto.
Precondiciones	▪ No existe un usuario registrado con correo electrónico "correoIncorrecto@correo.com"
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, como correo electrónico "correoIncorrecto@correo.com" y como contraseña "password" y pulsará el botón de inicio de sesión.
Resultado	▪ El usuario no iniciará sesión, mostrando un mensaje indicando que el correo electrónico o la contraseña es incorrecto.

Tabla 6.17: Prueba P15: Inicio de sesión de usuario con correo electrónico incorrecto.

Titulo	Inicio de sesión de usuario con contraseña incorrecta.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, como correo electrónico "correo@correo.com" y como contraseña "wordpass" y pulsará el botón de inicio de sesión.
Resultado	▪ El usuario no iniciará sesión, mostrando un mensaje indicando que el correo electrónico o la contraseña es incorrecto.

Tabla 6.18: Prueba P16: Inicio de sesión de usuario con contraseña incorrecta.

Titulo	Inicio de sesión de usuario rechazado.
Precondiciones	▪ Existe un usuario registrado con correo electrónico "prop@correo.com" y contraseña "password" que ha sido rechazado.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, como correo electrónico "prop@correo.com" y como contraseña "password" y pulsará el botón de inicio de sesión.
Resultado	▪ El usuario no iniciará sesión, mostrando un mensaje indicando que el usuario ha sido rechazado.

Tabla 6.19: Prueba P17: Inicio de sesión de usuario rechazado.

Titulo	Inicio de sesión de usuario no validado.
Precondiciones	▪ Existe un usuario con correo electrónico "correoP@correo.com" y contraseña "password" cuya petición de registro aún no ha sido gestionada por un administrador.
Descripción	El usuario que quiere iniciar sesión introducirá los datos pedidos en los campos del formulario, como correo electrónico "correoP@correo.com" y como contraseña "password" y pulsará el botón de inicio de sesión.
Resultado	▪ El usuario iniciará sesión, pero al no estar validado contará con funcionalidad reducida, mostrando un mensaje para indicar esto.

Tabla 6.20: Prueba P18: Inicio de sesión de usuario no validado.

6.4.3. Validación de propietarios

Precondiciones	▪ Existe un usuario registrado con rol de administrador que ha iniciado sesión. ▪ El usuario con rol de administrador se encuentra en la pagina de validación de propietarios de la plataforma */propietarios/validar*. ▪ Existe al menos una petición de registro de un propietario.

Tabla 6.21: Precondiciones para la validación de propietarios.

Titulo	Aceptación de una solicitud de registro de un propietario.
Descripción	El usuario con rol de administrador visualiza las peticiones de registro de propietarios pendientes de gestionar, selecciona una de ellas, desplegando los datos del usuario y los botones de aceptar y rechazar la petición, presiona el botón de aceptar mostrando un popup como doble confirmación, donde presiona confirmar para aceptar la petición de registro del usuario.
Resultado	▪ La petición de registro de ese propietario seleccionado desaparecerá de la lista, habiendo gestionado dicha petición correctamente. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado. ▪ El propietario validado ya podrá iniciar sesión en la plataforma con todas las funcionalidades de su rol disponibles.

Tabla 6.22: Prueba P19: Aceptación de una solicitud de registro de un propietario.

Titulo	Rechazo de una solicitud de registro de un propietario.
Descripción	El usuario con rol de administrador visualiza las peticiones de registro de propietarios pendientes de gestionar, selecciona una de ellas, desplegando los datos del usuario y los botones de aceptar y rechazar la petición, presiona el botón de rechazar mostrando un popup como doble confirmación, donde presiona confirmar para rechazar la petición de registro del usuario.
Resultado	▪ La petición de registro de ese propietario seleccionado desaparecerá de la lista, habiendo gestionado dicha petición correctamente. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado. ▪ El propietario no podrá iniciar sesión en la plataforma.

Tabla 6.23: Prueba P20: Rechazo de una solicitud de registro de un propietario.

6.4.4. Validación de establecimientos

Precondiciones	▪ Existe un usuario registrado con rol de administrador que ha iniciado sesión en la plataforma. ▪ El usuario con rol de administrador se encuentra en la pagina de validación de establecimientos de la plataforma */establecimientos/validar*. ▪ Existe al menos una petición de registro de un establecimiento.

Tabla 6.24: Precondiciones para la validación de establecimientos.

Titulo	Aceptación de una solicitud de registro de un establecimiento.
Descripción	El usuario con rol de administrador visualiza las peticiones de registro de establecimientos pendientes de gestionar, selecciona una de ellas, desplegando los datos del local y los botones de aceptar y rechazar la petición, presiona el botón de aceptar mostrando un popup como doble confirmación, donde presiona confirmar para aceptar la petición de registro del establecimiento.
Resultado	▪ La petición de registro de ese establecimiento seleccionado desaparecerá de la lista, habiendo gestionado dicha petición correctamente. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado. ▪ El establecimiento validado ya podrá ser visible para clientes de la plataforma.

Tabla 6.25: Prueba P21: Aceptación de una solicitud de registro de un establecimiento.

Titulo	Rechazo de una solicitud de registro de un establecimiento.
Descripción	El usuario con rol de administrador visualiza las peticiones de registro de establecimientos pendientes de gestionar, selecciona una de ellas, desplegando los datos del local y los botones de aceptar y rechazar la petición, presiona el botón de rechazar mostrando un popup como doble confirmación, donde presiona confirmar para rechazar la petición de registro del establecimiento.
Resultado	▪ La petición de registro de ese establecimiento seleccionado desaparecerá de la lista, habiendo gestionado dicha petición correctamente. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado. ▪ El establecimiento rechazado ya no podrá ser visible para clientes de la plataforma, marcandose en rojo en la lista de establecimientos de un propietario.

Tabla 6.26: Prueba P22: Rechazo de una solicitud de registro de un establecimiento.

Titulo	Búsqueda por localidad de peticiones de registro de establecimientos.
Precondiciones	▪ Existe al menos una petición de registro de un establecimiento cuya localidad es “Palencia”.
Descripción	El usuario con rol de administrador visualiza las peticiones de registro de establecimientos pendientes de gestionar, optando por filtrar las peticiones por localidad, por lo que escribe “Palencia” en el buscador y presiona el botón de buscar, obteniendo así, únicamente, aquellos establecimientos de Palencia.
Resultado	▪ Las peticiones de validación de establecimientos visibles para el administrador son aquellas que sean de la localidad de “Palencia”, mostrando un botón para reiniciar el filtro a la totalidad de peticiones.

Tabla 6.27: Prueba P23: Búsqueda por localidad de peticiones de registro de establecimientos.

6.4.5. Listar productos

Titulo	Borrado de producto correcto.
Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ El usuario con rol de propietario se encuentra en la pagina para listar sus productos de la plataforma */productos*. ▪ Existe al menos un producto registrado en la plataforma por el propietario.
Descripción	El usuario con rol de propietario visualiza sus productos registrados en la plataforma, seleccionando el botón de eliminar de aquel que quiere borrar, lo que muestra un popup como doble confirmación, donde presiona eliminar para confirmar el borrado del producto.
Errores solucionados	▪ El producto eliminado no desaparecía de la lista de productos. ▪ No se avisaba al propietario de que el producto sería eliminado de las cartas y menús donde estuviera incluido.
Resultado	▪ El producto seleccionado será eliminado de la plataforma y desaparecerá de la lista de productos. ▪ En caso de que el producto seleccionado forme parte de algún menú o carta de algún establecimiento, será eliminado de estos. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.28: Prueba P24: Borrado de producto correcto.

6.4.6. Listar mesas

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por el propietario. ▪ El usuario con rol de propietario se encuentra en la pagina para listar las mesas de alguno de sus establecimientos de la plataforma */establecimientos/{idEstablecimiento}/mesas*. ▪ Existe al menos una mesa, en el establecimiento seleccionado, registrada en la plataforma por el propietario.

Tabla 6.29: Precondiciones para la lista de mesas con rol de propietario.

Titulo	Activar mesa correcto.
Precondiciones	▪ Existe al menos una mesa desactivada, en el establecimiento seleccionado, registrada en la plataforma por el propietario.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, pulsando el checkbox para activar una de las mesas desactivadas.
Resultado	▪ La mesa seleccionada será activada. ▪ Se modificará la etiqueta relativa a la capacidad total de las mesas activas. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.30: Prueba P25: Activar mesa correcto.

Titulo	Desactivar mesa correcto.
Precondiciones	▪ Existe al menos una mesa activada, en el establecimiento seleccionado, registrada en la plataforma por el propietario.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, pulsando el checkbox para desactivar una de las mesas activadas.
Resultado	▪ La mesa seleccionada será desactivada y no aparecerá en la lista de mesas de un trabajador del local. ▪ Se modificará la etiqueta relativa a la capacidad total de las mesas activas. ▪ Si la capacidad total de las mesas tras la desactivación es menor al aforo máximo actual, se mostrará un mensaje indicando que cambie dicho valor a uno correcto. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.31: Prueba P26: Desactivar mesa correcto.

Titulo	Borrado de mesa correcto.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, seleccionando el botón de eliminar de aquella que quiere borrar, lo que muestra un popup como doble confirmación, donde presiona eliminar para confirmar el borrado de la mesa.
Errores solucionados	▪ No se modificaba la etiqueta de la capacidad total de las mesas activadas. ▪ No se controlaba la condición de que el aforo máximo actual sea menor o igual que la capacidad total de las mesas.
Resultado	▪ La mesa seleccionada será eliminada de la plataforma y desaparecerá de la lista de mesas. ▪ Si esta estaba activada, se modificará la etiqueta relativa a la capacidad total de las mesas activas. ▪ Si la capacidad total de las mesas tras el borrado es menor al aforo máximo actual, se mostrará un mensaje indicando que cambie dicho valor a uno correcto. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.32: Prueba P27: Borrado de mesa correcto.

Titulo	Cambiar aforo máximo actual correcto.
Precondiciones	▪ La capacidad total de las mesas activas es igual a 20.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, decidiendo cambiar el aforo máximo actual a 20.
Resultado	▪ Se modificará el valor del aforo máximo del establecimiento a 20 personas. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.33: Prueba P28: Cambiar aforo máximo actual correcto.

Titulo	Cambiar aforo máximo actual superior a la capacidad de las mesas.
Precondiciones	▪ La capacidad total de las mesas activas es igual a 20.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, decidiendo cambiar el aforo máximo actual a 21.
Resultado	▪ No se modificará el valor del aforo máximo del establecimiento, mostrando un mensaje de que el valor debe ser positivo e inferior o igual a la capacidad total de las mesas.

Tabla 6.34: Prueba P29: Cambiar aforo máximo actual superior a la capacidad de las mesas.

Titulo	Cambiar aforo máximo actual a cero.
Precondiciones	▪ La capacidad total de las mesas activas es igual a 20.
Descripción	El usuario con rol de propietario visualiza las mesas registradas en la plataforma, en el establecimiento elegido, decidiendo cambiar el aforo máximo actual a 0.
Resultado	▪ No se modificará el valor del aforo máximo del establecimiento, mostrando un mensaje de que el valor debe ser positivo e inferior o igual a la capacidad total de las mesas.

Tabla 6.35: Prueba P30: Cambiar aforo máximo actual a cero.

Precondiciones	▪ Existe un usuario registrado con rol de trabajador que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por un propietario, donde el usuario está registrado como trabajador. ▪ El usuario con rol de trabajador se encuentra en la pagina para listar las mesas del establecimiento de la plataforma donde trabaja */establecimientos/{idEstablecimiento}/mesas*. ▪ Existe al menos una mesa activada, en el establecimiento, registrada en la plataforma por el propietario.

Tabla 6.36: Precondiciones para la lista de mesas con rol de trabajador.

Titulo	Marcar mesa como libre correcto.
Precondiciones	▪ Existe al menos una mesa activada en estado ocupada en el establecimiento.
Descripción	El usuario con rol de propietario visualiza las mesas activadas registradas en la plataforma, en el establecimiento donde trabaja, pulsando el checkbox para marcar como libre una de las mesas en estado ocupada.
Resultado	▪ La mesa seleccionada pasará a estar libre, modificando el estado del aforo actual del local, alterando la ocupación en la vista de elección de establecimientos y en la carta de cada uno de estos. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.37: Prueba P31: Marcar mesa como libre correcto.

Titulo	Marcar mesa como ocupada correcto.
Precondiciones	▪ Existe al menos una mesa activada en estado libre en el establecimiento.
Descripción	El usuario con rol de propietario visualiza las mesas activadas registradas en la plataforma, en el establecimiento donde trabaja, pulsando el checkbox para marcar como ocupada una de las mesas en estado libre.
Resultado	▪ La mesa seleccionada pasará a estar ocupada, modificando el estado del aforo actual del local, alterando la ocupación en la vista de elección de establecimientos y en la carta de cada uno de estos. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.38: Prueba P32: Marcar mesa como ocupada correcto.

6.4.7. Listar establecimientos

Precondiciones	▪ Existe al menos un establecimiento registrado en la plataforma por un propietario y validado por un administrador. ▪ El usuario se encuentra en la pagina para listar los establecimientos registrados en la plataforma */establecimientos*.

Tabla 6.39: Precondiciones para la lista de establecimientos.

Titulo	Elección de establecimiento.
Descripción	El usuario visualiza los establecimientos registrados en la plataforma que han sido validados, pudiendo visualizar la ocupación en tiempo real de cada uno de ellos. Posteriormente seleccionará el establecimiento del cual quiere consultar la carta.
Resultado	▪ El usuario de la plataforma será llevado a la vista encargada de mostrar la carta del establecimiento elegido.

Tabla 6.40: Prueba P33: Elección de establecimiento.

Titulo	Consulta de todos los establecimientos de un propietario.
Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por dicho propietario.
Descripción	El usuario visualiza los establecimientos registradas en la plataforma que han sido validados, seleccionando el botón para visualizar sus establecimientos.
Resultado	▪ El usuario podrá observar todos los establecimientos que ha registrado en la plataforma, marcando en blanco aquellos que han sido validados, en amarillo los que están pendientes de validación y en rojo aquellos que han sido rechazados.

Tabla 6.41: Prueba P34: Consulta de todos los establecimientos de un propietario.

6.4.8. Listar carta

Precondiciones	▪ Existe al menos un establecimiento registrado en la plataforma por un propietario y validado por un administrador. ▪ El propietario del local ha establecido que los productos con nombre "Pizza barbacoa" y "Pizza 4 quesos" forman parte de la carta. ▪ Existe una mesa registrada en el establecimiento seleccionado con el identificador de mesa "Mesa1". ▪ No existe una mesa registrada en el establecimiento seleccionado con el identificador de mesa "Mesa2". ▪ El usuario se encuentra en la pagina para listar la carta del establecimiento */establecimientos/{idEstablecimiento}/carta*.

Tabla 6.42: Precondiciones para la lista de los productos de una carta.

Titulo	Selección de mesa introduciendo el identificador de forma correcta.
Descripción	El usuario visualiza la carta del establecimiento. Posteriormente ingresará, en el campo del identificador de mesa, el código "Mesa1" y pulsará el botón para comprobar si es correcto y proceder a realizar un pedido.
Resultado	▪ Como el identificador existe, se da la posibilidad de pedir productos al usuario que ha introducido el identificador de la mesa. ▪ La mesa con dicho identificador será marcada como ocupada si no lo estaba, alterando el nivel de ocupación del local y el estado de las mesas para la gestión de estas por los trabajadores del local. ▪ Se muestra un icono de carrito de la compra para que pueda navegar hasta el resumen de la comanda y mostrar los productos pedidos. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que ya puede comenzar a pedir.

Tabla 6.43: Prueba P35: Selección de mesa introduciendo el identificador de forma correcta.

Titulo	Selección de mesa introduciendo el identificador de forma incorrecta.
Descripción	El usuario visualiza la carta del establecimiento. Posteriormente ingresará, en el campo del identificador de mesa, el código "Mesa2" y pulsará el botón para comprobar si es correcto.
Resultado	▪ Como el identificador no existe, se muestra un mensaje de que es necesario un identificador correcto.

Tabla 6.44: Prueba P36: Selección de mesa introduciendo el identificador de forma incorrecta.

Titulo	Pedir producto.
Descripción	El usuario visualiza la carta del establecimiento. Posteriormente ingresará, en el campo del identificador de mesa, el código "Mesa1" y pulsará el botón para comprobar si es correcto. Como dicho identificador coincide con una mesa del local, se le permite añadir productos a la comanda, por lo que el usuario pulsa sobre el producto "Pizza barbacoa", selecciona 2 unidades, observando como se modifica el precio del pedido, y pide el producto. Si el cliente desea observar un resumen de la comanda a realizar, pulsando sobre el carrito de la compra podrá observarlo.
Resultado	▪ El producto "Pizza barbacoa" se añade a la comanda, pudiendo observarlo al pulsar sobre el carrito de la compra y viendo el resumen de la comanda. ▪ Se actualiza el número de productos pedidos del carrito de 0 a 1.

Tabla 6.45: Prueba P37: Pedir producto.

Titulo	Pedir mismo producto por segunda vez.
Precondiciones	▪ El usuario ya ha introducido el identificador de mesa "Mesa1". ▪ El usuario ya ha añadido dos "Pizza barbacoa" a la comanda.
Descripción	El usuario pulsa sobre el producto "Pizza barbacoa", selecciona 1 unidad, y pide el producto, que ya había pedido con anterioridad en la misma comanda.
Resultado	▪ El producto "Pizza barbacoa" se añade a la comanda, pudiendo observar al pulsar sobre el carrito de la compra y viendo el resumen de la comanda, que se han sumado las cantidades, habiendo 3 unidades del producto en la comanda. ▪ El número de productos pedidos del carrito se mantiene en 1, ya que es el mismo producto que había pedido antes.

Tabla 6.46: Prueba P38: Pedir mismo producto por segunda vez.

Titulo	Pedir segundo producto diferente del primero.
Precondiciones	▪ El usuario ya ha introducido el identificador de mesa "Mesa1". ▪ El usuario ya ha añadido dos "Pizza barbacoa" a la comanda.
Descripción	El usuario pulsa sobre el producto "Pizza 4 quesos", selecciona 1 unidad, y pide el producto. Si el cliente desea observar un resumen de la comanda a realizar, pulsando sobre el carrito de la compra podrá observarlo.
Resultado	▪ El producto "Pizza 4 quesos" se añade a la comanda, pudiendo observar al pulsar sobre el carrito de la compra y viendo el resumen de la comanda, que están los dos productos pedidos. ▪ El número de productos pedidos del carrito pasa de 1 a 2, ya que el producto seleccionado es diferente al resto de productos de la comanda.

Tabla 6.47: Prueba P39: Pedir segundo producto diferente del primero.

6.4.9. Listar comanda

Precondiciones	▪ Existe al menos un establecimiento registrado en la plataforma por un propietario y validado por un administrador. ▪ El propietario del local ha establecido que los productos con nombre "Pizza barbacoa" y "Pizza 4 quesos" forman parte de la carta. ▪ Existe una mesa registrada en el establecimiento seleccionado con el identificador de mesa "Mesa1". ▪ El usuario se encuentra en la pagina resumen de la comanda */establecimientos/{idEstablecimiento}/carta/comanda*. ▪ El usuario ya ha introducido el identificador de mesa "Mesa1". ▪ El usuario ya ha añadido dos "Pizza barbacoa" a la comanda.

Tabla 6.48: Precondiciones para la lista de los productos añadidos para pedir.

Titulo	Eliminar producto de la comanda.
Descripción	El usuario visualiza el resumen de la comanda, observando que hay dos unidades del producto "Pizza barbacoa" por equivocación al pedir, por lo que procede a borrar dicho producto de la comanda pulsando el botón de borrado. Tras esto se despliega una ventana a modo de doble confirmación para que no sea un click por error, el usuario procede a confirmar la eliminación del producto de la comanda.
Resultado	▪ El producto borrado desaparece del resumen de la comanda, observando como se actualizan los campos cantidad e importe del total de la comanda.

Tabla 6.49: Prueba P40: Eliminar producto de la comanda.

Titulo	Confirmar comanda.
Descripción	El usuario visualiza el resumen de la comanda, observando que hay dos unidades del producto "Pizza barbacoa". Como ya ha finalizado su pedido, procede a pulsar el botón para confirmar. Tras esto se muestra una ventana a modo de confirmación, donde confirma el pedido.
Resultado	▪ La comanda está registrada en estado "confirmado". ▪ El usuario es llevado a la lista de la carta. ▪ La comanda llega a la vista de las comandas activas por mesa, solamente visible por los trabajadores del local. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.50: Prueba P41: Confirmar comanda.

6.4.10. Editar carta

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por el propietario. ▪ Existen dos productos registrados por el propietario en la plataforma con los nombres "Pizza barbacoa" y "Pizza 4 quesos". ▪ El usuario con rol de propietario se encuentra en la pagina para editar la carta de alguno de sus establecimientos de la plataforma */establecimientos/{idEstablecimiento}/carta/editar*.

Tabla 6.51: Precondiciones para la edición de una carta.

Titulo	Selección de un producto para incluirlo en la carta.
Precondiciones	▪ La carta del establecimiento está vacía.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en la carta, selecciona el producto "Pizza barbacoa", habilitando el botón "Añadir" de la parte superior de la lista, posteriormente lo pulsa para pasar el producto de lista.
Resultado	▪ El producto seleccionado "Pizza barbacoa" pasa a la lista de productos seleccionados para agregar a la carta del establecimiento. ▪ Se deshabilita el botón "Añadir" pulsado y se habilita el botón "Todos" de la lista de productos agregados a la carta

Tabla 6.52: Prueba P42: Selección de un producto para incluirlo en la carta.

Titulo	Selección de todos los productos para incluirlos en la carta.
Precondiciones	▪ La carta del establecimiento está vacía.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en la carta, selecciona el botón "Todos" de la parte inferior de la lista, lo que conlleva a que todos los productos de dicha lista se resalten, se deshabilite el botón "Todos" pulsado y se habilite el botón "Añadir" de la parte superior de la lista, posteriormente lo pulsa para pasar todos los productos de lista.
Resultado	▪ Todos los productos de la lista de productos no agregados en la carta pasan a la lista de productos seleccionados para agregar. ▪ Se deshabilita el botón "Añadir" pulsado y se habilita el botón "Todos" de la lista de productos agregados a la carta

Tabla 6.53: Prueba P43: Selección de todos los productos para incluirlos en la carta.

Titulo	Deseleccionar los productos seleccionados para incluirlos en la carta.
Precondiciones	▪ La carta del establecimiento está vacía.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, viendo que el producto "Pizza barbacoa" está seleccionado, presiona el botón "Ninguno" de la parte inferior de dicha lista para deseleccionar el producto.
Resultado	▪ El producto "Pizza barbacoa" deja de estar seleccionado. ▪ Se deshabilitan los botones "Ninguno" y "Añadir" y se habilita el botón "Todos" de la lista de productos no agregados a la carta.

Tabla 6.54: Prueba P44: Deseleccionar los productos seleccionados para incluirlos en la carta.

Titulo	Selección de un producto para eliminarlo de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto "Pizza barbacoa".
Descripción	El propietario visualiza el producto que compone la carta, selecciona el producto "Pizza barbacoa", habilitando el botón "Eliminar" de la parte superior de la lista derecha, posteriormente lo pulsa para pasar el producto de lista y quitarlo de la carta.
Resultado	▪ El producto seleccionado "Pizza barbacoa" pasa a la lista de productos no agregados a la carta del establecimiento. ▪ Se deshabilitan los botones "Añadir" y "Todos" de la lista de productos añadidos a la carta.

Tabla 6.55: Prueba P45: Selección de un producto para eliminarlo de la carta.

Titulo	Selección de todos los productos para eliminarlos de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto "Pizza barbacoa".
Descripción	El propietario visualiza el producto incluido en la carta, selecciona el botón "Todos" de la parte inferior de la lista, lo que conlleva a que todos los productos de dicha lista se resalten, se deshabilite el botón "Todos" pulsado y se habilite el botón "Añadir" de la parte superior de la lista, posteriormente lo pulsa para pasar todos los productos de lista.
Resultado	▪ Todos los productos de la lista de productos agregados en la carta pasan a la lista de productos no agregados. ▪ Se deshabilita el botón "Añadir" pulsado y se habilita el botón "Todos" de la lista de productos no agregados a la carta.

Tabla 6.56: Prueba P46: Selección de todos los productos para eliminarlos de la carta.

Titulo	Deseleccionar los productos seleccionados para eliminarlos de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”. ▪ El propietario ha seleccionado el producto “Pizza barbacoa”, resaltándolo y habilitando el botón “Ninguno” de la parte inferior de la lista derecha.
Descripción	El propietario visualiza los productos incluidos en la carta, viendo que el producto “Pizza barbacoa” está seleccionado, presiona el botón “Ninguno” de la parte inferior de dicha lista para deseleccionar el producto.
Resultado	▪ El producto “Pizza barbacoa” deja de estar seleccionado. ▪ Se deshabilitan los botones “Ninguno” y “Añadir” y se habilita el botón “Todos” de la lista de productos agregados a la carta.

Tabla 6.57: Prueba P47: Deseleccionar los productos seleccionados para eliminarlos de la carta.

Titulo	Arrastrar un producto para incluirlo en la carta.
Precondiciones	▪ La carta del establecimiento está vacía.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en la carta, selecciona el producto “Pizza barbacoa” arrastrándolo hasta la lista derecha para incluirlo en la carta del establecimiento.
Resultado	▪ El producto seleccionado “Pizza barbacoa” pasa a la lista de productos seleccionados para agregar a la carta del establecimiento.

Tabla 6.58: Prueba P48: Arrastrar un producto para incluirlo en la carta.

Titulo	Arrastrar un producto para eliminarlo de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza el producto que compone la carta, selecciona el producto “Pizza barbacoa” arrastrándolo hasta la lista izquierda para eliminarlo de la carta del establecimiento.
Resultado	▪ El producto seleccionado “Pizza barbacoa” pasa a la lista de productos no agregados a la carta del establecimiento.

Tabla 6.59: Prueba P49: Arrastrar un producto para eliminarlo de la carta.

Titulo	Cancelar edición de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza los productos que componen la carta del establecimiento, y los que no están incluidos, decidiendo que no tiene que realizar ningún cambio en esta, por lo que presiona el botón de cancelar.
Resultado	▪ La carta del establecimiento se mantiene como estaba. ▪ El propietario es redirigido a la ventana con la carta del establecimiento.

Tabla 6.60: Prueba P50: Cancelar edición de la carta.

Titulo	Confirmar edición de la carta con cambios en los productos que la componen.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza los productos que componen la carta del establecimiento, y los que no están incluidos, decidiendo que el producto “Pizza 4 quesos” tiene que formar parte de la carta, por lo que lo añade a la lista de productos seleccionados para agregar a la carta. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, confirma el cambio y termina la edición de la carta.
Resultado	▪ La carta del establecimiento cambia, incluyendo ahora el producto “Pizza 4 quesos”. ▪ El propietario es redirigido a la ventana con la carta del establecimiento. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.61: Prueba P51: Confirmar edición de la carta con cambios en los productos que la componen.

Titulo	Confirmar edición de la carta sin productos seleccionados para formar parte de la carta.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza los productos que componen la carta del establecimiento, y los que no están incluidos, decidiendo que el producto “Pizza barbacoa” ya no tiene que formar parte de la carta, por lo que lo elimina de la lista de productos seleccionados para agregar a la carta. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, donde se le avisa de que la carta está vacía, confirma el cambio y termina la edición de la carta.
Resultado	▪ La carta del establecimiento se vacía. ▪ El propietario es redirigido a la ventana con la carta del establecimiento, donde ya no aparecerán los productos al estar vacía. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.62: Prueba P52: Confirmar edición de la carta sin productos seleccionados para formar parte de la carta.

Titulo	Confirmar edición de la carta sin cambios en los productos que la componen.
Precondiciones	▪ La carta del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza los productos que componen la carta del establecimiento, y los que no están incluidos, decidiendo que la carta está formada por los productos necesarios. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, avisando de que la carta es idéntica a la carta anterior, confirma el cambio y termina la edición de la carta.
Resultado	▪ La carta se mantiene sin cambios en los productos que la componen. ▪ El propietario es redirigido a la ventana con la carta del establecimiento, donde ya no aparecerán los productos al estar vacía. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.63: Prueba P53: Confirmar edición de la carta sin cambios en los productos que la componen.

6.4.11. Editar menú

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por el propietario. ▪ Existen dos productos registrados por el propietario en la plataforma con los nombres “Pizza barbacoa” y “Pizza 4 quesos”. ▪ El usuario con rol de propietario se encuentra en la pagina para editar el menú de alguno de sus establecimientos de la plataforma */establecimientos/{idEstablecimiento}/carta/menu/editar*.

Tabla 6.64: Precondiciones para la edición de un menú.

Titulo	Selección de un producto para incluirlo en el menú.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto “Pizza barbacoa”, habilitando el botón “Añadir” de la parte superior de la lista, posteriormente lo pulsa para pasar el producto de lista.
Resultado	▪ El producto seleccionado “Pizza barbacoa” pasa a la lista de productos seleccionados para agregar al menú del establecimiento. ▪ Se deshabilita el botón “Añadir” pulsado y se habilita el botón “Todos” de la lista de productos agregados al menú.

Tabla 6.65: Prueba P54: Selección de un producto para incluirlo en el menú.

Titulo	Selección de todos los productos para incluirlos en el menú.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el botón "Todos" de la parte inferior de la lista, lo que conlleva a que todos los productos de dicha lista se resalten, se deshabilite el botón "Todos" pulsado y se habilite el botón "Añadir" de la parte superior de la lista, posteriormente lo pulsa para pasar todos los productos de lista.
Resultado	▪ Todos los productos de la lista de productos no agregados al menú pasan a la lista de productos seleccionados para agregar. ▪ Se deshabilita el botón "Añadir" pulsado y se habilita el botón "Todos" de la lista de productos agregados al menú.

Tabla 6.66: Prueba P55: Selección de todos los productos para incluirlos en el menú.

Titulo	Deseleccionar los productos seleccionados para incluirlos en el menú.
Precondiciones	▪ El menú del establecimiento está vacía. ▪ El propietario ha seleccionado el producto "Pizza barbacoa", resaltándolo y habilitando el botón "Ninguno" de la parte inferior de la lista izquierda.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, viendo que el producto "Pizza barbacoa" está seleccionado, presiona el botón "Ninguno" de la parte inferior de dicha lista para deseleccionar el producto.
Resultado	▪ El producto "Pizza barbacoa" deja de estar seleccionado. ▪ Se deshabilitan los botones "Ninguno" y "Añadir" y se habilita el botón "Todos" de la lista de productos no agregados al menú.

Tabla 6.67: Prueba P56: Deseleccionar los productos seleccionados para incluirlos en el menú.

Titulo	Selección de un producto para eliminarlo del menú.
Precondiciones	▪ El menú del establecimiento está compuesto por el producto "Pizza barbacoa".
Descripción	El propietario visualiza el producto que compone el menú, selecciona el producto "Pizza barbacoa", habilitando el botón "Eliminar" de la parte superior de la lista derecha, posteriormente lo pulsa para pasar el producto de lista y quitarlo del menú.
Resultado	▪ El producto seleccionado "Pizza barbacoa" pasa a la lista de productos no agregados al menú del establecimiento. ▪ Se deshabilitan los botones "Añadir" y "Todos" de la lista de productos añadidos al menú.

Tabla 6.68: Prueba P57: Selección de un producto para eliminarlo del menú.

Titulo	Selección de todos los productos para eliminarlos del menú.
Precondiciones	▪ El menú del establecimiento está compuesto por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza el producto incluido en el menú, selecciona el botón “Todos” de la parte inferior de la lista, lo que conlleva a que todos los productos de dicha lista se resalten, se deshabilite el botón “Todos” pulsado y se habilite el botón “Añadir” de la parte superior de la lista, posteriormente lo pulsa para pasar todos los productos de lista.
Resultado	▪ Todos los productos de la lista de productos agregados en el menú pasan a la lista de productos no agregados. ▪ Se deshabilita el botón “Añadir” pulsado y se habilita el botón “Todos” de la lista de productos no agregados al menú.

Tabla 6.69: Prueba P58: Selección de todos los productos para eliminarlos del menú.

Titulo	Deseleccionar los productos seleccionados para eliminarlos del menú.
Precondiciones	▪ El menú del establecimiento está compuesto por el producto “Pizza barbacoa”. ▪ El propietario ha seleccionado el producto “Pizza barbacoa”, resaltándolo y habilitando el botón “Ninguno” de la parte inferior de la lista derecha.
Descripción	El propietario visualiza los productos incluidos en el menú, viendo que el producto “Pizza barbacoa” está seleccionado, presiona el botón “Ninguno” de la parte inferior de dicha lista para deseleccionar el producto.
Resultado	▪ El producto “Pizza barbacoa” deja de estar seleccionado. ▪ Se deshabilitan los botones “Ninguno” y “Añadir” y se habilita el botón “Todos” de la lista de productos agregados al menú.

Tabla 6.70: Prueba P59: Deseleccionar los productos seleccionados para eliminarlos del menú.

Titulo	Arrastrar un producto para incluirlo en el menú.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto “Pizza barbacoa” arrastrándolo hasta la lista derecha para incluirlo en el menú del establecimiento.
Resultado	▪ El producto seleccionado “Pizza barbacoa” pasa a la lista de productos seleccionados para agregar al menú del establecimiento.

Tabla 6.71: Prueba P60: Arrastrar un producto para incluirlo en el menú.

Titulo	Arrastrar un producto para eliminarlo del menú.
Precondiciones	▪ El menú del establecimiento está compuesto por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza el producto que compone el menú, selecciona el producto “Pizza barbacoa” arrastrándolo hasta la lista izquierda para eliminarlo del menú del establecimiento.
Resultado	▪ El producto seleccionado “Pizza barbacoa” pasa a la lista de productos no agregados al menú del establecimiento.

Tabla 6.72: Prueba P61: Arrastrar un producto para eliminarlo del menú.

Titulo	Cancelar edición del menú.
Precondiciones	▪ El menú del establecimiento está compuesta por el producto “Pizza barbacoa”.
Descripción	El propietario visualiza la información y los productos que componen el menú del establecimiento, y los que no están incluidos, decidiendo que no tiene que realizar ningún cambio en este, por lo que presiona el botón de cancelar.
Resultado	▪ El menú del establecimiento se mantiene como estaba. ▪ El propietario es redirigido a la ventana con la carta del establecimiento.

Tabla 6.73: Prueba P62: Cancelar edición del menú.

Titulo	Confirmar edición del menú con cambios en los productos que lo componen.
Precondiciones	▪ El menú del establecimiento está compuesta por el producto “Pizza barbacoa”, tiene como nombre “Menú del día”, precio 12.50€ y descripción “Menú del día”.
Descripción	El propietario visualiza los productos que componen el menú del establecimiento, y los que no están incluidos, decidiendo que el producto “Pizza 4 quesos” tiene que formar parte del menú, por lo que lo añade a la lista de productos seleccionados para agregar al menú. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, confirma el cambio y termina la edición del menú.
Resultado	▪ El menú del establecimiento cambia, incluyendo ahora el producto “Pizza 4 quesos”. ▪ El propietario es redirigido a la ventana con la carta del establecimiento. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.74: Prueba P63: Confirmar edición del menú con cambios en los productos que lo componen.

Titulo	Confirmar edición del menú sin productos seleccionados para formar parte de este.
Precondiciones	▪ El menú del establecimiento está compuesta por el producto “Pizza barbacoa”, tiene como nombre “Menú del día”, precio 12.50€ y descripción “Menú del día”.
Descripción	El propietario visualiza los productos que componen el menú del establecimiento, y los que no están incluidos, decidiendo que el producto “Pizza barbacoa” ya no tiene que formar parte del menú, por lo que lo elimina de la lista de productos seleccionados para agregar a este. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, donde se le avisa de que el menú está vacío, confirma el cambio y termina la edición del menú.
Resultado	▪ El menú del establecimiento se vacía. ▪ El propietario es redirigido a la ventana con la carta del establecimiento, donde ya no aparecerá el menú al no tener productos. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.75: Prueba P43: Confirmar edición del menú sin productos seleccionados para formar parte de este.

Titulo	Confirmar edición del menú sin cambios en los productos que la componen.
Precondiciones	▪ El menú del establecimiento está compuesta por el producto “Pizza barbacoa”, tiene un precio de 12.50€ y descripción “Menú del día”.
Descripción	El propietario visualiza los productos que componen el menú del establecimiento, y los que no están incluidos, decidiendo que el menú está formado por los productos necesarios. Tras esto presiona el botón de confirmar, desplegándose una ventana emergente a modo de doble confirmación para la operación, avisando de que el menú es idéntico al menú anterior, confirma el cambio y termina la edición de este.
Resultado	▪ El menú se mantiene sin cambios en los productos que lo componen. ▪ El propietario es redirigido a la ventana con la carta del establecimiento, donde ya no aparecerá el menú al estar vacío. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.76: Prueba P65: Confirmar edición del menú sin cambios en los productos que la componen.

Titulo	Edición del menú sin introducir nombre.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario introduce un precio de 12.50€ y como descripción pone “Menú del día”, posteriormente visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto “Pizza barbacoa” añadiéndolo a la lista de productos a agregar al menú e intenta pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del nombre, el cual se marca en rojo destacando que es necesario.

Tabla 6.77: Prueba P66: Edición del menú sin introducir nombre.

Titulo	Edición del menú sin introducir precio.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario introduce como nombre "Menú del día" y como descripción pone "Menú del día", posteriormente visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto "Pizza barbacoa" añadiéndolo a la lista de productos a agregar al menú e intenta pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del precio, el cual se marca en rojo destacando que es necesario.

Tabla 6.78: Prueba P67: Edición del menú sin introducir precio.

Titulo	Edición del menú con precio negativo.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario introduce como nombre "Menú del día", en el campo del precio "-1" y como descripción pone "Menú del día", posteriormente visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto "Pizza barbacoa" añadiéndolo a la lista de productos a agregar al menú e intenta pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del precio, el cual se marca en rojo destacando que existe un error.

Tabla 6.79: Prueba P68: Edición del menú con precio negativo.

Titulo	Edición del menú sin introducir descripción.
Precondiciones	▪ El menú del establecimiento está vacío.
Descripción	El propietario introduce como nombre "Menú del día" y como precio pone "12.50€", posteriormente visualiza los dos productos registrados en la parte izquierda de la pantalla, en la lista de productos no incluidos en el menú, selecciona el producto "Pizza barbacoa" añadiéndolo a la lista de productos a agregar al menú e intenta pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la descripción, el cual se marca en rojo destacando que es necesario.

Tabla 6.80: Prueba P69: Edición del menú sin introducir descripción.

6.4.12. Crear trabajador

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por el propietario. ▪ El usuario con rol de propietario se encuentra en la página para añadir trabajadores al local */establecimientos/{idEstablecimiento}/trabajadores/nuevo*.

Tabla 6.81: Precondiciones para la creación de un trabajador.

Titulo	Crear trabajador correcto.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, en el nombre introduce “David Curieses”, como DNI “12345678A”, como correo electrónico “david@correo.com”, en el campo del teléfono escribe “123456789” y finalmente escribirá como contraseña “password”, pulsará el botón de confirmar, finalizando el proceso de registro de un nuevo trabajador para el local de forma exitosa.
Resultado	▪ El propietario es redirigido a la ventana con la lista de trabajadores del establecimiento, donde aparecerá el nuevo empleado. ▪ El nuevo trabajador registrado podrá iniciar sesión en la plataforma. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.82: Prueba P70: Crear trabajador correcto.

Titulo	Crear trabajador incorrecto campo nombre vacío.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, excepto el nombre, como DNI “12345678A”, como correo electrónico “david@correo.com”, en el campo del teléfono escribe “123456789” y finalmente escribirá como contraseña “password” e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del nombre, el cual se marca en rojo destacando que es necesario..

Tabla 6.83: Prueba P71: Crear trabajador incorrecto campo nombre vacío.

Titulo	Crear trabajador incorrecto campo DNI vacío.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, excepto el DNI, como nombre “David Curieses”, como correo electrónico “david@correo.com”, en el campo del teléfono escribe “123456789” y finalmente escribirá como contraseña “password” e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del DNI, el cual se marca en rojo destacando que es necesario.

Tabla 6.84: Prueba P72: Crear trabajador incorrecto campo DNI vacío.

Titulo	Crear trabajador incorrecto campo DNI con 7 números.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, como nombre “David Curieses”, en el campo DNI introduce “1234567A”, como correo electrónico “david@correo.com”, en el campo del teléfono escribe “123456789” y finalmente escribirá como contraseña “password” e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del DNI, el cual se marca en rojo destacando que existe un error.

Tabla 6.85: Prueba P73: Crear trabajador incorrecto campo DNI con 7 números.

Titulo	Crear trabajador incorrecto campo DNI con 9 números.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, como nombre "David Curieses", en el campo DNI introduce "123456789A", como correo electrónico "david@correo.com", en el campo del teléfono escribe "123456789" y finalmente escribirá como contraseña "password" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del DNI, el cual se marca en rojo destacando que existe un error.

Tabla 6.86: Prueba P74: Crear trabajador incorrecto campo DNI con 9 números.

Titulo	Crear trabajador incorrecto campo DNI sin letra final.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, como nombre "David Curieses", en el campo DNI introduce "12345678", como correo electrónico "david@correo.com", en el campo del teléfono escribe "123456789" y finalmente escribirá como contraseña "password" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del DNI, el cual se marca en rojo destacando que existe un error.

Tabla 6.87: Prueba P75: Crear trabajador incorrecto campo DNI sin letra final.

Titulo	Crear trabajador incorrecto campo correo vacío.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, excepto el correo electrónico, como nombre "David Curieses", como DNI "12345678A", en el campo del teléfono escribe "123456789" y finalmente escribirá como contraseña "password" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del DNI, el cual se marca en rojo destacando que es necesario.

Tabla 6.88: Prueba P76: Crear trabajador incorrecto campo correo vacío.

Titulo	Crear trabajador incorrecto campo correo incorrecto.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, como nombre "David Curieses", como DNI "12345678A", para el correo electrónico introducirá "correo@correo", en el campo del teléfono escribe "123456789" y finalmente escribirá como contraseña "password" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error el campo del correo electrónico, el cual se marca en rojo destacando que existe un error.

Tabla 6.89: Prueba P77: Crear trabajador incorrecto campo correo incorrecto.

Titulo	Crear trabajador incorrecto campo teléfono vacío.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, excepto el teléfono, como nombre "David Curieses", como DNI "12345678A", en el campo del correo electrónico escribe "david@correo.com" y finalmente escribirá como contraseña "password" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del teléfono, el cual se marca en rojo destacando que es necesario.

Tabla 6.90: Prueba P78: Crear trabajador incorrecto campo teléfono vacío.

Titulo	Crear trabajador incorrecto campo contraseña vacío.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, excepto la contraseña, como nombre "David Curieses", como DNI "12345678A", en el campo del correo electrónico escribe "david@correo.com" y finalmente escribirá como teléfono "123456789" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la contraseña, el cual se marca en rojo destacando que es necesario.

Tabla 6.91: Prueba P79: Crear trabajador incorrecto campo contraseña vacío.

Titulo	Crear trabajador incorrecto campo DNI con 7 números.
Descripción	El usuario que quiere agregar un nuevo trabajador a su local introducirá los datos pedidos en los campos del formulario, como nombre "David Curieses", en el campo DNI introduce "12345678A", como correo electrónico "david@correo.com", en el campo del teléfono escribe "123456789" y finalmente escribirá como contraseña "passwor" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo de la contraseña, el cual se marca en rojo destacando que existe un error.

Tabla 6.92: Prueba P80: Crear trabajador incorrecto campo contraseña con 7 caracteres.

6.4.13. Crear producto

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ El usuario con rol de propietario se encuentra en la página para añadir productos */productos/nuevo*.

Tabla 6.93: Precondiciones para la creación de un producto.

Titulo	Crear producto correcto.
Descripción	El usuario que quiere agregar un nuevo producto a su lista introducirá los datos pedidos en los campos del formulario, en el nombre introduce "Pizza 4 quesos", como precio "7.20" y como descripción pondrá "Pizza con una base de tomate y queso mozzarella, gorgonzola, fontina y parmesano", pulsará el botón de confirmar, finalizando el proceso de registro de un nuevo producto de forma exitosa.
Resultado	▪ El propietario es redirigido a la ventana con la lista sus productos, donde aparecerá el nuevo producto. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.94: Prueba P81: Crear producto correcto.

Titulo	Crear producto incorrecto nombre vacío.
Descripción	El usuario que quiere agregar un nuevo producto a su lista introducirá los datos pedidos en los campos del formulario, excepto el nombre, como precio introduce "7.20" y como descripción pondrá "Pizza con una base de tomate y queso mozzarella, gorgonzola, fontina y parmesano" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del nombre, el cual se marca en rojo destacando que es necesario.

Tabla 6.95: Prueba P82: Crear producto incorrecto nombre vacío.

Titulo	Crear producto incorrecto precio vacío.
Descripción	El usuario que quiere agregar un nuevo producto a su lista introducirá los datos pedidos en los campos del formulario, excepto el precio, como nombre introduce "Pizza 4 quesos" y como descripción pondrá "Pizza con una base de tomate y queso mozzarella, gorgonzola, fontina y parmesano" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del precio, el cual se marca en rojo destacando que es necesario.

Tabla 6.96: Prueba P83: Crear producto incorrecto precio vacío.

Titulo	Crear producto incorrecto precio negativo.
Descripción	El usuario que quiere agregar un nuevo producto a su lista introducirá los datos pedidos en los campos del formulario, como nombre introduce "Pizza 4 quesos", como precio "-2" y como descripción pondrá "Pizza con una base de tomate y queso mozzarella, gorgonzola, fontina y parmesano" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error el campo del precio, el cual se marca en rojo destacando que existe un error.

Tabla 6.97: Prueba P84: Crear producto incorrecto precio negativo.

Titulo	Crear producto incorrecto descripción vacía.
Descripción	El usuario que quiere agregar un nuevo producto a su lista introducirá los datos pedidos en los campos del formulario, excepto la descripción, como nombre introduce "Pizza 4 quesos" y como precio pondrá "7.20" e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la descripción, el cual se marca en rojo destacando que es necesario.

Tabla 6.98: Prueba P85: Crear producto incorrecto descripción vacía.

Titulo	Cancelar operación crear producto.
Descripción	El usuario accede a la ventana de creación o edición de productos, optando por cancelar la operación, por lo que pulsa dicho botón.
Errores solucionados	▪ El botón de cancelar la operación mandaba el formulario creando un producto sin datos.
Resultado	▪ El propietario es redirigido a la ventana con la lista sus productos.

Tabla 6.99: Prueba P86: Cancelar operación crear producto.

Titulo	Edición producto correcta.
Precondiciones	▪ Existe un producto registrado por el propietario con nombre "Pizza 4 quesos", descripción "Pizza con una base de tomate y queso mozarela, gorgonzola, fontina y parmesano" y precio "7.20€".
Descripción	El propietario accede a la ventana para la edición de su producto registrado con nombre "Pizza jamón y queso", optando por cambiar los datos de este, cambiando el campo de la descripción por "Pizza con una base de tomate, jamón cocido y queso mozarela" y un precio de "6.50", pulsando el botón de confirmar.
Resultado	▪ El propietario es redirigido a la ventana con la lista de productos. ▪ El producto editado cambiará sus datos.

Tabla 6.100: Prueba P87: Edición producto correcta.

Aclaración: Las pruebas relativas a la edición de un producto de forma incorrecta, ya sea por la falta de introducir alguno de los campos, o por errores en la información introducida en estos campos, son idénticos a las pruebas de la creación de productos, por lo que quedan descritas entre las tablas 6.95 y 6.98.

6.4.14. Crear mesa

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ Existe al menos un establecimiento registrado en la plataforma por el propietario. ▪ Existe una mesa en el establecimiento con el identificador "Mesa0" y ocupación máxima de 5 personas. ▪ No existe ninguna mesa en el establecimiento con el identificador "Mesa1". ▪ El usuario con rol de propietario se encuentra en la página para añadir una mesa al establecimiento */establecimientos/{idEstablecimiento}/mesas/nuevo*.

Tabla 6.101: Precondiciones para la creación de una mesa.

Titulo	Crear mesa correcta.
Descripción	El usuario que quiere agregar una mesa nueva al establecimiento seleccionado introducirá los datos pedidos en los campos del formulario, en el identificador de mesa introduce "Mesa1" y como capacidad máxima "8", pulsará el botón de confirmar, finalizando el proceso de registro de una nueva mesa de forma exitosa.
Resultado	▪ El propietario es redirigido a la ventana con la lista de las mesas del establecimiento, donde aparecerá la nueva mesa. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.102: Prueba P88: Crear mesa correcta.

Titulo	Crear mesa incorrecta identificador vacío.
Descripción	El usuario que quiere agregar una mesa nueva al establecimiento seleccionado introducirá los datos pedidos en los campos del formulario, excepto el identificador, por lo que introducirá como capacidad máxima "8", e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del identificador, el cual se marca en rojo destacando que es necesario.

Tabla 6.103: Prueba P89: Crear mesa incorrecta identificador vacío.

Titulo	Crear mesa incorrecta identificador repetido.
Descripción	El usuario que quiere agregar una mesa nueva al establecimiento seleccionado introducirá los datos pedidos en los campos del formulario, en el identificador de mesa introduce "Mesa0" y como capacidad máxima "8" y finalmente pulsará el botón de confirmar.
Resultado	▪ La acción no se realiza ya que el identificador de la mesa está repetido. ▪ Se mostrará un mensaje bajo el identificador indicando que ya está en uso.

Tabla 6.104: Prueba P90: Crear mesa incorrecta identificador repetido.

Titulo	Crear mesa incorrecta capacidad vacía.
Descripción	El usuario que quiere agregar una mesa nueva al establecimiento seleccionado introducirá los datos pedidos en los campos del formulario, excepto la capacidad, por lo que introducirá como identificador de mesa "Mesa1", e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del capacidad máxima, el cual se marca en rojo destacando que es necesario.

Tabla 6.105: Prueba P91: Crear mesa incorrecta capacidad vacía.

Titulo	Crear mesa incorrecta capacidad negativa.
Descripción	El usuario que quiere agregar una mesa nueva al establecimiento seleccionado introducirá los datos pedidos en los campos del formulario, por lo que introducirá como identificador de mesa "Mesa1" y como capacidad máxima "-1", e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo de la capacidad máxima, el cual se marca en rojo destacando que existe un error. ▪ Se mostrará un mensaje bajo la capacidad indicando que es errónea.

Tabla 6.106: Prueba P92: Crear mesa incorrecta capacidad negativa.

Titulo	Cancelar operación crear mesa.
Descripción	El usuario accede a la ventana de creación o edición de mesas, optando por cancelar la operación, por lo que pulsa dicho botón.
Resultado	▪ El propietario es redirigido a la ventana con la lista de las mesas del establecimiento.

Tabla 6.107: Prueba P93: Cancelar operación crear mesa.

Titulo	Edición mesa correcta.
Descripción	El usuario accede a la ventana para la edición de la mesa con identificador "Mesa0", optando por cambiar los datos de esta, insertando en el campo del identificador de mesa "MesaA" y cambiando la capacidad de 5 a 4 personas, pulsando el botón de confirmar.
Resultado	▪ El propietario es redirigido a la ventana con la lista de las mesas del establecimiento. ▪ La mesa seleccionada ha cambiado sus datos a "MesaA", como identificador y su capacidad a 4.

Tabla 6.108: Prueba P94: Edición mesa correcta.

Aclaración: Las pruebas relativas a la edición de una mesa de forma incorrecta, ya sea por la falta de introducir alguno de los campos, o por la repetición del identificador de la mesa son idénticos a las pruebas de la creación de la mesa, por lo que quedan descritas entre las tablas 6.103 y 6.106.

6.4.15. Crear establecimiento

Precondiciones	▪ Existe un usuario registrado con rol de propietario que ha iniciado sesión en la plataforma. ▪ El usuario con rol de propietario se encuentra en la página para añadir un establecimiento */establecimientos/nuevo*.

Tabla 6.109: Precondiciones para la creación de un establecimiento.

Titulo	Crear establecimiento correcto.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, en el nombre del establecimiento introduce "La aldea", como teléfono "123456789", en el campo correo electrónico "laaldea@correo.com", para la descripción "Establecimiento de comida casera", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, pulsará el botón de confirmar, finalizando el proceso de registro de un nuevo establecimiento de forma exitosa.
Resultado	▪ El propietario es redirigido a la ventana con la lista de los establecimientos validados. ▪ El establecimiento creado aparecerá en la lista de los establecimientos del propietario como solicitud pendiente. ▪ La solicitud de registro del establecimiento creado aparecerá en la lista de solicitudes de validación de establecimientos de los administradores. ▪ Se muestra un mensaje emergente en la esquina inferior derecha confirmando que la operación se ha realizado.

Tabla 6.110: Prueba P95: Crear establecimiento correcto.

Titulo	Crear establecimiento incorrecto nombre vacío.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto el nombre, como teléfono introducirá "123456789", en el campo correo electrónico "laaldea@correo.com", para la descripción "Establecimiento de comida casera", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del nombre, el cual se marca en rojo destacando que es necesario.

Tabla 6.111: Prueba P96: Crear establecimiento incorrecto nombre vacío.

Titulo	Crear establecimiento incorrecto teléfono vacío.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto el teléfono, como nombre introducirá "La aldea", en el campo correo electrónico "laaldea@correo.com", para la descripción "Establecimiento de comida casera", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del teléfono, el cual se marca en rojo destacando que es necesario.

Tabla 6.112: Prueba P97: Crear establecimiento incorrecto teléfono vacío.

Titulo	Crear establecimiento incorrecto correo electrónico vacío.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto el correo electrónico, como nombre introducirá "La aldea", en el campo teléfono "123456789", para la descripción "Establecimiento de comida casera", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del correo electrónico, el cual se marca en rojo destacando que es necesario.

Tabla 6.113: Prueba P98: Crear establecimiento incorrecto correo electrónico vacío.

Titulo	Crear establecimiento incorrecto correo electrónico incorrecto.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo", para la descripción "Establecimiento de comida casera", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del correo electrónico, el cual se marca en rojo destacando que existe el error.

Tabla 6.114: Prueba P99: Crear establecimiento incorrecto correo electrónico incorrecto.

Titulo	Crear establecimiento incorrecto descripción vacía.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto la descripción, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la descripción, el cual se marca en rojo destacando que es necesario.

Tabla 6.115: Prueba P100: Crear establecimiento incorrecto descripción vacía.

Titulo	Crear establecimiento incorrecto dirección vacía.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto la dirección, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", localidad "Palencia", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la dirección, el cual se marca en rojo destacando que es necesario.

Tabla 6.116: Prueba P101: Crear establecimiento incorrecto dirección vacía.

Titulo	Crear establecimiento incorrecto localidad vacía.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto la localidad, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la localidad, el cual se marca en rojo destacando que es necesario.

Tabla 6.117: Prueba P102: Crear establecimiento incorrecto localidad vacía.

Titulo	Crear establecimiento incorrecto código postal vacío.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto el código postal, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del código postal, el cual se marca en rojo destacando que es necesario.

Tabla 6.118: Prueba P103: Crear establecimiento incorrecto código postal vacío.

Titulo	Crear establecimiento incorrecto aforo máximo vacío.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto el aforo máximo, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia" y código postal "34003", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo del aforo máximo, el cual se marca en rojo destacando que es necesario.

Tabla 6.119: Prueba P104: Crear establecimiento incorrecto aforo máximo vacío.

Titulo	Crear establecimiento incorrecto aforo máximo negativo.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "-1", para las coordenadas introducirá "42,00730635614866" de latitud y "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por un error en el campo del aforo máximo, el cual se marca en rojo destacando que existe el error.

Tabla 6.120: Prueba P105: Crear establecimiento incorrecto aforo máximo negativo.

Titulo	Crear establecimiento incorrecto latitud vacía.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto la latitud, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia" código postal "34003" y aforo máximo "75", para las coordenadas introducirá "-4,521753043110412" de longitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la latitud, el cual se marca en rojo destacando que es necesario.

Tabla 6.121: Prueba P106: Crear establecimiento incorrecto latitud vacía.

Titulo	Crear establecimiento incorrecto longitud vacía.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, excepto la longitud, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia" código postal "34003" y aforo máximo "75", para las coordenadas introducirá "42,00730635614866" de latitud, e intentará pulsar el botón de confirmar.
Resultado	▪ El botón de confirmar está deshabilitado por faltar el campo de la longitud, el cual se marca en rojo destacando que es necesario.

Tabla 6.122: Prueba P107: Crear establecimiento incorrecto longitud vacía.

Titulo	Crear establecimiento click mapa.
Descripción	El usuario que quiere agregar un establecimiento nuevo introducirá los datos pedidos en los campos del formulario, como nombre introducirá "La aldea", en el campo teléfono "123456789", para el correo electrónico "laaldea@correo.com", como descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia" código postal "34003" y aforo máximo "75", para las coordenadas se desplazará sobre la posición del local en el mapa, pulsando donde esté situado.
Resultado	▪ Se actualizarán los campos de la latitud y longitud, de acuerdo al lugar donde ha pulsado en el mapa.

Tabla 6.123: Prueba P108: Crear establecimiento click mapa.

Titulo	Cancelar operación crear establecimiento.
Descripción	El usuario que accede a la vista para crear o editar un establecimiento decide que no es necesario realizar la operación, por lo que opta por pulsa el botón de cancelar.
Resultado	▪ El propietario será redirigido a la vista con los establecimientos validados en la plataforma.

Tabla 6.124: Prueba P109: Cancelar operación crear establecimiento.

Titulo	Edición establecimiento correcta.
Precondiciones	▪ Existe un establecimiento registrado por el propietario con nombre "La aldea", teléfono "123456789", correo electrónico "laaldea@correo.com", descripción "Establecimiento de comida casera", dirección "Avenida de prueba", localidad "Palencia", código postal "34003" y aforo máximo "75", como coordenadas "42,00730635614866" de latitud y "-4,521753043110412" de longitud.
Descripción	El propietario accede a la ventana para la edición de su establecimiento registrado con nombre "La aldea", optando por cambiar los datos de este, cambiando el campo de la localidad por "Valencia", el código postal por "46001" y como coordenadas "39.473429" de latitud y "-0.380062" de longitud, pulsando el botón de confirmar.
Resultado	▪ El propietario es redirigido a la ventana con la lista de establecimientos. ▪ El establecimiento editado cambiará sus datos, mostrando en la lista como su localidad ahora es "Valencia".

Tabla 6.125: Prueba P110: Edición establecimiento correcta.

Aclaración: Las pruebas relativas a la edición de un establecimiento de forma incorrecta, ya sea por la falta de introducir alguno de los campos, o por errores en la información introducida en estos campos, son idénticos a las pruebas de la creación de establecimientos, por lo que quedan descritas entre las tablas 6.111 y 6.122.

Capítulo 7

Seguimiento del proyecto

En este Capítulo se desarrollará en detalle el transcurso de los diferentes Sprints realizados a lo largo del desarrollo del proyecto, incluyendo en cada uno de ellos, las historias de usuario abordadas, los puntos de historia de cada una, las decisiones tomadas en cada momento y los problemas surgidos y las soluciones tomadas en cada caso. Al final del Capítulo se valorará el seguimiento del proyecto, comparándolo con la planificación original propuesta.

7.1. Introducción

El proyecto comenzó el día 9 de febrero de 2022, pero antes de empezar el desarrollo de este en sí, se decidió, de forma excepcional, ya que la duración establecida es de 2 semanas por sprint, establecer un periodo de una semana hasta el 16 del mismo mes como Sprint 0, para tareas iniciales.

7.2. Sprint 0 (09/02/2022 - 16/02/2022)

Con el inicio del Sprint 0, el cual se utilizará, principalmente, para la toma de contacto con las diferentes herramientas que se utilizarán en el desarrollo del proyecto, ya sea para recordar conceptos o aprendizaje de 0 de las que fuesen necesarias, se dio comienzo al proyecto.

Durante la semana que duraba este sprint, se establecieron y configuraron las diferentes herramientas y entornos necesarios para el comienzo del desarrollo del proyecto.

7.3. Sprint 1 (16/02/2022-02/03/2022)

Este primer Sprint es el que da comienzo al desarrollo del proyecto y sus historias de usuario. El tiempo estimado total a realizar en cada Sprint se estableció en 36 horas, por lo que obtenemos que cada uno de los 6 puntos de historia a distribuir en las diferentes historias de usuario corresponde a 6 horas de trabajo, siendo esta la mínima cantidad de trabajo asignable a una historia.

Durante el Sprint Planning se optó por abordar en el Sprint 1 el desarrollo de las historias de usuario 1.1 y 2.1. La relación entre los puntos de historia asignados a cada historia de usuario que se va a realizar se puede observar en la Tabla 7.1.

Historia de usuario	**Puntos de historia**
1.1	4
2.1	2

Tabla 7.1: Asignación de puntos de historia para el Sprint Backlog 1

H.U.	**Tareas**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
1.1	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	24 horas	29 horas y 25 minutos	Completada
2.1	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	7 horas y 5 minutos	Implementación del mapa para seleccionar coordenadas.
TOTAL		36 horas	36 horas y 30 minutos	1 de 2 HU completadas

Tabla 7.2: Seguimiento del Sprint 1

El Sprint 1 finalizó con un tiempo empleado de 36 horas y 30 minutos, de las 36 horas estimadas, completando la primera historia de usuario y dejando de la segunda historia

de usuario únicamente la implementación de un mapa interactivo para la selección de las coordenadas, lo que otorgaría al usuario una mejora significativa con respecto a la usabilidad de la plataforma, ya que podría indicar visualmente, al pulsar en un mapa, las coordenadas en las que se encuentra su establecimiento al crearlo.

Para llevar a cabo la interfaz gráfica de ambas historias de usuario se ha consultado la documentación de Bootstrap sobre los tipos de grids existentes [3] y los tipos de componente "collapse" [2].

La funcionalidad pendiente referida al mapa interactivo para la selección de coordenadas pasará al siguiente Sprint.

7.4. Sprint 2 (02/03/2022-16/03/2022)

Para llevar a cabo este segundo sprint se comenzó por estimar la duración de terminar la implementación pendiente de la historia de usuario 2.1, la cual no se finalizó en el anterior sprint. Una vez se estimó 1 punto para esta, se optó por determinar el resto de las historias de usuario a abordar en el presente sprint y sus puntos de historia, los cuales se pueden ver en la Tabla 7.3.

Historia de usuario	**Puntos de historia**
2.1	1
1.6	1
5.1	2
2.6	1
2.7	2

Tabla 7.3: Asignación de puntos de historia para el Sprint Backlog 2

A pesar de que en parte de la primera semana del sprint no pude dedicar las horas planificadas por enfermedad, el Sprint 2 finalizó con un tiempo empleado de 33 horas y 2 minutos, de las 42 horas estimadas, completando gran parte del sprint y dejando las historias de usuario correspondientes a consultar establecimientos a validar y la posibilidad de visualizar una imagen sin completar, aunque ambas iniciadas.

Para llevar a cabo la implementación del mapa interactivo, para seleccionar las coordenadas del establecimiento a crear por el propietario, se valoró entre dos bibliotecas gratuitas que ofrecen este servicio, Leaflet [8] y OpenLayers [5], tras analizar cada una y se optó por utilizar la primera de ellas, Leaflet.

Para la actualización de la coordenada seleccionada al hacer click sobre el mapa consulte la página Stack Overflow, donde encontré la solución a mi cuestión [83].

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
2.1	▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	4 horas y 35 minutos	Completada
1.6	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	9 horas y 55 minutos	Pendiente finalizar implementación.
5.1	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	7 horas y 36 minutos	Pendiente finalizar implementación.
2.6	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 51 minutos	Completada
2.7	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	7 horas y 5 minutos	Completada
TOTAL		42 horas	33 horas y 2 minutos	3 de 5 HU completadas

Tabla 7.4: Seguimiento del Sprint 2

Posteriormente, para primero realizar la historia de usuario 2.6, y posteriormente la historia de usuario 5.1, tuve que consultar algo más de información sobre las tokens JWT, donde me interesaba almacenar tanto el tipo de usuario (rol) que iniciaba sesión como el identificador de este. Para llevar a cabo la desencriptación de la token y obtener estos campos obtuve la solución de nuevo de Stack Overflow [90].

En relación a los campos anteriores, consulte dos de las posibilidades para almacenar estos campos en la memoria del navegador, ya que convendría tenerlos para operaciones posteriores, las dos posibilidades consultadas fueron la memoria local y la memoria de sesión del navegador [30]. Debido a que, en este caso, la información almacenada del usuario no es de vital importancia ni sensible, se podría almacenar en cualquiera de las dos posibilidades consultadas, optando finalmente por utilizar la memoria de sesión, la cual, a diferencia de la memoria local, se borra cuando el usuario opta por cerrar la pestaña o el navegador. La información guardada sobre el usuario también se borraría en caso de que este cierre sesión desde el navegador.

Para la creación de la Entity Propietario en el backend, al ser una clase que extiende de Usuario Registrado tuve que buscar información acerca de cómo crear esta relación de forma correcta, para que no haya problemas al obtener los datos posteriormente mediante la api de persistencia utilizada, JPA [17]. Para ello opté por utilizar el tipo JOINED, para que cada tipo tuviera su propia tabla. También busque información sobre las relaciones de multiplicidad entre los campos de las bases de datos con JPA [19].

Por último, tras realizar en el frontend la funcionalidad requerida para que el usuario pueda seleccionar una imagen de su almacenamiento de archivos, he encontrado un problema al recuperar dichas imágenes que el propietario subiría sobre el establecimiento al crearlo, que no he podido solucionar por el momento, consiguiendo únicamente que se muestre un cuadro blanco indicando que no carga la imagen.

7.5. Sprint 3 (16/03/2022 - 30/03/2022)

De las dos historias de usuario que quedaron pendientes del anterior sprint, se ha decidido en el Sprint Planning que la historia de usuario 1.6, correspondiente a que el cliente pueda visualizar una imagen asociada a cada establecimiento, se posponga para sprints posteriores debido a complicaciones a la hora de mostrar dicha imagen en las páginas correspondientes.

Por otro lado, la historia de usuario 5.1, correspondiente a que el administrador de la plataforma pueda consultar los establecimientos que están pendientes de validación para poder ser publicados en la plataforma, se ha decidido incluir en este sprint para finalizar su desarrollo. Los puntos de historias asignados a las historias de usuario abordadas en este sprint se pueden observar en la siguiente tabla (Tabla 7.5).

El Sprint 3 finalizó con un tiempo empleado de 15 horas y 41 minutos, de las 36 horas estimadas, completando la totalidad de historias de usuario planificadas para este sprint.

Historia de usuario	Puntos de historia
5.1	1
2.8	2
2.9	1
2.10	1
5.3	1

Tabla 7.5: Asignación de puntos de historia para el Sprint Backlog 3

Una de las razones por las cuales la historia de usuario 5.1 se tuvo que atrasar su finalización hasta este sprint consistía en que a la hora de recuperar de la base de datos los establecimientos, si dos o más de estos pertenecían al mismo propietario la respuesta era errónea, lo cual, tras consultar diversas fuentes sin éxito, y analizar en profundidad el mensaje de error devuelto por el servidor, revisé cuidadosamente las clases y descubrí que el método get del identificador de la clase Usuario Registrado, clase de la cual hereda Propietario, devolvía "Object", en vez de "Integer", lo cual solucionaba el problema.

Para llevar a cabo la historia de usuario 5.3, relativa a aceptar o rechazar las peticiones de publicación de establecimientos en la plataforma, se ha optado por cambiar el valor lógico "validado" del establecimiento, por un campo "estado", una enumeración de los diferentes estados en los que está el establecimiento, en relación a su publicación en la plataforma, cuyos valores son "pendiente", "validado" o "rechazado".

También se ha integrado que sea una validación en dos pasos, es decir, con una confirmación por si el usuario pulsase algún botón sin querer, para ello se ha elegido insertar un Modal que aparecerá en cuanto el administrador quiera validar o rechazar cualquier establecimiento, a modo de ventana de confirmación [40].

Una de las decisiones de diseño que se ha tomado en este sprint ha sido asociar los productos a un propietario, ya que en el caso de que un propietario posea una cadena o franquicia de establecimientos o que en varios de sus establecimientos se disponga de los mismos productos, será más cómodo para este el manejo de los productos, evitando que el propietario tenga que crearlos para cada establecimiento en concreto, como estaba pensado desde un principio.

En la realización de la historia de usuario 2.9, se ha optado por paginar la tabla de productos asociados al propietario que realiza la consulta, consiguiendo así que sea lo más cómodo, personalizado y sencillo de usar para este. Para llevar a cabo esta paginación se han consultado fuentes en Internet, encontrando en la página Tech Club Tajamar la funcionalidad pensada [96].

Al igual que se realizó una doble confirmación para validar un establecimiento, se ha realizado para la historia de usuario 2.10, relativa a borrar un producto registrado, ya que el usuario con rol de propietario, al visualizar sus productos podría pulsar el botón de eliminar por equivocación, lo cual da una seguridad mayor a la hora de que se realicen operaciones de alto impacto, como la eliminación de un producto o, en el caso de la otra historia de usuario citada, validar o rechazar un establecimiento.

H.U.	**Tareas**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
5.1	▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 56 minutos	Completada
2.8	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	2 horas y 37 minutos	Completada
2.9	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 17 minutos	Completada
2.10	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	1 hora y 32 minutos	Completada
5.3	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	4 hora y 19 minutos	Completada
TOTAL		36 horas	15 horas y 41 minutos	5 de 5 HU completadas

Tabla 7.6: Seguimiento del Sprint 3

7.6. Sprint 4 (30/03/2022 - 13/04/2022)

El Sprint 4 comenzó el 30 de marzo con la reunión correspondiente al Sprint Planning donde se decidió, tras haber completado todas las historias de usuario del sprint anterior, por comenzar a abordar más en profundidad el manejo de productos para su inserción en cartas de establecimientos y posteriormente la creación de menús. También se decidió abordar en este sprint un periodo de tiempo para hacer una revisión general de la interfaz de usuario, con intención de homogeneizar y depurar esta, para que sea lo más amigable, sencilla e intuitiva para el usuario que la use.

Los puntos de historias asignados a cada historia de usuario de este sprint se puede observar en la Tabla 7.7.

Historia de usuario	Puntos de historia
2.11	1
2.5	2
1.2	1
2.2	1
6.1	1

Tabla 7.7: Asignación de puntos de historia para el Sprint Backlog 4

El Sprint 4 finalizó el 13 de abril completando cuatro de las cinco historias de usuario propuestas, quedando únicamente la historia de usuario relativa a la mejora de la interfaz de usuario sin comenzar, como se puede ver en la Tabla 7.8.

En este sprint se ha decidido, en la realización de la historia de usuario 2.11, a la vez que se añade el botón de edición de un producto, cambiar la tabla en la que se mostraban los productos por una tabla "mat-table" [71], de Angular Material [65], la cual es una mejora de la interfaz y dispone de una mejor paginación y mayor usabilidad, proporcionando también un selector de productos por página dinámico.

A la hora de incorporar el tipo de tabla comentado anteriormente de Angular Material, me encontré con el problema de que la paginación no se mostraba adecuadamente, lo cual se solucionaba importando el módulo "BrowserAnimationModule", la solución a este problema se encontró en la página Stack Overflow [84].

También quise cambiar las etiquetas de texto que aparecen en la paginación de las tablas de Angular Material, ya que de forma predeterminada vienen en inglés, al igual que para el anterior problema, encontré la solución en Stack overflow [88], la cual consistía en modificar ciertos campos de MatPaginator [66], responsable de la paginación de la tabla.

Para la realización de la historia de usuario 2.5, relativa a añadir productos a la carta de un establecimiento, se han barajado dos opciones, la primera consiste en incorporar un checkbox en la lista total de productos, eligiendo así cuáles componen la carta y cuáles no,

H.U.	**Tareas**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
2.11	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	5 horas y 9 minutos	Completada
2.5	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	16 horas y 33 minutos	Completada
1.2	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	2 horas y 45 minutos	Completada
2.2	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	2 horas y 5 minutos	Completada
6.1	▪ Revisión código. ▪ Revisión IU. ▪ Testeo.	6 horas		Sin comenzar
TOTAL		36 horas	26 horas y 32 minutos	4 de 5 HU completadas

Tabla 7.8: Seguimiento del Sprint 4

y la segunda cosiste en dos listas paralelas, siendo una aquella con los productos que no están en la carta y la otra aquellos que sí, con una serie de botones para añadir o eliminar, tanto individualmente, como en grupos o en su totalidad, los productos de la carta.

Finalmente optando por realizar la segunda de las opciones, ya que es más sencilla de utilizar y de entender para el propietario.

Para esto último se utilizó Angular Dual-Listbox [28], un componente que proporciona todas las funciones que se buscan, desde selecciones de uno o varios productos, como la selección de la totalidad de los disponibles, como la eliminación de estos, que se abordará en otra historia de usuario. También se habilitó el campo "filtro" en ambas listas, lo cual proporciona la funcionalidad de poder buscar por nombres los diferentes productos que existen, que, en casos de listas muy grandes, puede ser de bastante ayuda para la gestión de la carta.

Dicho componente Angular utiliza una lista con la totalidad de los datos, que a su vez está compuesta por dos lista, una primera con aquellos datos disponibles para ser seleccionados y una segunda con aquellos datos que han sido seleccionados, que también son parte de la primera lista; al proceder a editar la carta apareció un problema en relación a los productos que ya están incluidos en la carta al cargar la página, ya que dichos datos de los productos deben aparecer en la primera lista, que contiene todos los productos, y en la segunda, que contiene los productos ya incluidos; la solución se encontró en Stack Overflow [93], donde se explica como crear la lista general de datos a partir de la creación de las dos sublistas que la componen. En dicha solución se utilizaban tres puntos antes de una variable lo cual desconocía su funcionamiento, por lo que buscando información sobre ello descubrí que recibe el nombre de "spread operator" [55], y uno de sus usos es para la inicialización de arrays a partir de otros, por lo que se usa para crear el array de datos general del componente Dual-Listbox.

7.7. Ampliación Sprint 4 (13/04/2022 - 20/04/2022)

Aunque en la planificación inicial se estimó que el periodo entre el 13 de abril y el 20 del mismo mes no se desarrollaría ninguna tarea, se ha optado por aprovechar a realizar la historia de usuario que quedó sin comenzar, por falta de tiempo, en el sprint 4, relacionada con la revisión y homogeneización de las diferentes interfaces de usuario y depuración del código desarrollado.

Como se trata de una ampliación del sprint, no se va a estimar un tiempo para la historia de usuario, pero si se contabilizará el tiempo empleado en las tareas que se realicen durante esta semana.

Historia de usuario	Puntos de historia
6.1	1

Tabla 7.9: Asignación de puntos de historia para la ampliación del sprint 4

En el desarrollo de esta ampliación del sprint 4, como se puede ver en la Tabla 7.10,

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
6.1	▪ Revisión código. ▪ Revisión IU. ▪ Testeo.		12 horas y 50 minutos	Completado
TOTAL		-	12 horas y 50 minutos	1 de 1 HU completadas

Tabla 7.10: Seguimiento de la ampliación del Sprint 4

se emplearon 12 horas y 50 minutos, que, aunque no se estimase un tiempo al ser una ampliación, si se tiene en cuenta el tiempo empleado en el desarrollo, en este caso, de la actividad realizada.

Para que toda la plataforma tenga una interfaz de usuario similar se ha optado por continuar con el uso de la librería de componentes de interfaces de usuario Angular Material [65], ya que la había usado en el desarrollo de ciertas partes de la plataforma con anterioridad. Entre los componentes que ofrece esta librería, se han usado Table [71] y Paginator [66] para añadir paginación a las tablas que no la tenían; Progress Spinner [68] en la pantalla de login, para dar una retroalimentación de espera al usuario que inicia sesión mientras se realiza el proceso; Expansion Panel [62] en vez de la etiqueta collapse de HTML y List [64] al listar establecimientos o los usuarios o establecimientos pendientes de validación.

7.8. Sprint 5 (20/04/2022 - 04/05/2022)

El Sprint 5 daría comienzo con el Sprint Planning realizado el 20 de abril, se optó por finalizar algunos aspectos de la plataforma, como son los relacionados con el registro de usuarios, tanto propietarios como clientes registrados (aunque estos no tendrán, por el momento, ninguna ventaja por estar registrados), y la validación de usuarios registrados, con rol de propietario, por parte de algún administrador de la plataforma. Como resultado obtenemos la siguiente asignación de puntos de historia para este sprint (Tabla 7.11).

Historia de usuario	Puntos de historia
2.12	1
1.7	1
5.2	1
5.3	1
2.13	2

Tabla 7.11: Asignación de puntos de historia para la ampliación del sprint 5

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
2.12	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 25 minutos	Completada
1.7	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	4 horas y 30 minutos	Completada
5.2	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	2 horas y 24 minutos	Completada
5.4	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	1 hora y 8 minutos	Completada
2.13	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	11 horas y 8 minutos	Falta por solucionar error al mostrar la carta y mostrar el menú, si hay, en la pantalla de la carta del establecimiento.

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	▪ Documentación Capítulo 2.	-	8 horas y 37 minutos	Pendiente de finalizar
-	▪ Documentación Capítulo 4.	-	2 horas y 35 minutos	Pendiente de finalizar
-	▪ Documentación Capítulo 5.	-	1 hora y 12 minutos	Pendiente de finalizar
TOTAL		36 horas	34 horas y 59 minutos	4 de 5 HU completadas

Tabla 7.12: Seguimiento del Sprint 5

Para el desarrollo de este sprint se estimaron 36 horas de trabajo, de las cuales se utilizaron 22 horas y 35 minutos en el desarrollo de la totalidad de cuatro de las cinco historias de usuario, quedando la última sin completar, y 12 horas y 24 minutos en tareas de documentación, haciendo un total de 34 horas y 59 minutos de trabajo en este sprint.

Las primeras historias de usuario se pudieron completar rápidamente gracias a que ciertas funciones llevadas a cabo en este sprint eran similares, tanto en funcionalidad como en interfaz de usuario, a funcionalidades desarrolladas en sprints anteriores, por lo que se pudo reutilizar código que agilizó la elaboración de estas historias de usuario. El seguimiento del Sprint se puede observar en la Tabla 7.12.

Como se comentó anteriormente, en este sprint se optó por finalizar con todos los aspectos relacionados con los usuarios, tanto el registro, como la validación, por parte de un administrador de la aplicación, de los usuarios con rol de propietario que se registrasen en la plataforma. Se ha decido brindar a los propietarios aún no validados la posibilidad de iniciar sesión en la plataforma, pero sin la posibilidad de realizar ninguna acción exclusiva de los usuarios con este rol que si están validados, desde la creación de establecimientos y productos, como la visualización de estos últimos.

Para indicar esta restricción de funcionalidad se ha decidido mostrar un mensaje de aviso en pantalla, para que el usuario sepa que aun no ha sido validado.

A la hora de registrar un usuario, se ha optado por utilizar un componente de Angular Material denominado Slide Toggle [69], para que el usuario que se registra elija si hacerlo como propietario o cliente.

También se decidió cambiar el campo validado del propietario de tipo booleano a "EstadoValidacion", una enumeración, y debido a que son los mismos estados en los que puede estar un establecimiento, también se adaptó el campo oportuno de esta entidad, que, aunque ya había sido cambiado en un sprint anterior (Sprint 3 (7.5)), se ha vuelto a modificar el nombre de la enumeración para que ambos tengan la misma. Ambos pueden estar en tres estados diferentes, "pendiente", hasta que un administrador decide aceptar o rechazar la

solicitud, siendo este el estado por defecto al crear la solicitud, “rechazado”, en caso de que el administrador decida rechazar la solicitud y “aceptado” en caso contrario.

Además de las historias de usuario relacionadas con el registro y validación de usuarios en la plataforma, se optó por finalizar la edición de la carta (Historia de usuario 2.12) y comenzar con la creación y edición del menú. Para este proyecto se decidió que solo existiría un menú como máximo (puede no haberlo) por carta en cada establecimiento.

Aunque en un escenario real podrían existir diferentes menús dentro de una carta, se ha decidido establecer esta cantidad para simplificar este apartado del proyecto, aunque, en un futuro, si nos encontrásemos un establecimiento con una carta más compleja, se debería implementar la posibilidad de crear varios menús por carta, como un menú para niños, menú del día, menús adaptados para gente celiaca, vegetariana, vegana, etc...

Como podemos observar en el seguimiento de este quinto sprint, la historia de usuario relacionada con la creación y edición del menú no se terminó de completar debido a que, tras la incorporación de la entidad Menu en el backend, y las correspondientes tablas y relaciones entre estas, a la hora de recuperar los datos en las peticiones realizadas desde el frontend nos encontramos con un problema. Este problema no es nuevo, ya que con anterioridad habíamos visto algo similar, pero la complejidad de este era mayor. La forma de recuperar la cadena de respuesta del servidor en formato JSON solo describe los objetos una vez, haciendo referencia al identificador de este si se repite en otro punto de la respuesta, por lo que al encontrarnos con productos que pertenecen a cartas o menús, hay que realizar una búsqueda en profundidad para obtener los datos de los objetos producto en su totalidad, y no solo sus identificadores. Una vez entendido el problema y esquematizado una solución, no resultó de gran dificultad resolverlo, pero atrasó la finalización de la historia. Esta historia pasará al siguiente sprint, con intenciones de ser finalizada.

Debido a la posibilidad de reutilización de código en este sprint se optó por comenzar también con tareas de documentación más en profundidad, comenzando el desarrollo de los Capítulos 2, 4 y 5.

7.9. Sprint 6 (04/05/2022 - 18/05/2022)

El Sprint 6 comenzó el día 4 de mayo con la reunión correspondiente al Sprint Planning, en esta se decidió finalizar la historia de usuario que quedó pendiente en el anterior sprint, asignando un punto de historia a su desarrollo, la edición de los datos de cada establecimiento de cada propietario, de otro punto de historia y se barajó la posibilidad de comenzar con el desarrollo de la creación de comandas o el manejo del aforo para obtener el nivel de ocupación del establecimiento, optando finalmente por desarrollar este último, ya que hay que crear todo desde cero y para la creación de comandas es necesario tener las mesas del establecimiento, por lo que al desarrollarlo en este sprint, ya lo tendremos cuando haga falta. Los puntos de historia correspondientes a cada historia de usuario de este sprint se pueden visualizar en la Tabla 7.13

En el desarrollo de este sprint se llevaron a cabo 26 horas y 6 minutos en el desarrollo

Historia de usuario	Puntos de historia
2.13	1
2.3	1
2.14	3
2.15	1

Tabla 7.13: Asignación de puntos de historia para el sprint 6

de las cuatro historias de usuario planificadas para este sprint, y 10 horas y 50 minutos para completar la documentación del Capítulo 2, realizando así 36 horas y 56 minutos de trabajo esta semana, sobrepasando el trabajo estimado de 36 horas para esta semana.

La historia de usuario 2.13 no se finalizó correctamente en el anterior sprint por algunos errores al recuperar los datos, ya que debido a las relaciones entre productos, menús y cartas, y la manera de mappear los datos enviados en cada respuesta desde el servidor, podría darse el caso de que ciertos objetos se encontrasen definidos en profundidades altas del JSON de respuesta, por lo que, y debido a que en este proyecto no es necesario conocer los menús y cartas a los que pertenece un producto en concreto, se ha optado por no mappear estos datos en la respuesta desde el servidor, resolviendo así el problema encontrado y finalizando el desarrollo de esta historia de usuario. Para llevar a cabo esto se ha utilizado la anotación @JsonIgnore [18] en los campos mencionados, de tal forma no se realizará una serialización de esos datos al recuperar el objeto en cuestión.

Para mostrar el menú en la carta del establecimiento se ha optado por mostrar en primera instancia el nombre del menú y su precio, y de forma desplegable, al hacer click sobre este, mostrar tanto la descripción, como los diferentes productos que componen el menú, con su correspondiente descripción.

Con respecto a la creación de nuevas mesas, se ha decidido por añadir un nuevo campo correspondiente a un nombre, o número identificativo único, que el propietario del establecimiento asignará a la mesa creada, para poder diferenciarla del resto de mesas dentro del mismo establecimiento. También se ha optado por asignar un campo activado a cada mesa, ya que, y tras las recientes experiencias vividas con el Covid-19, podría darse el caso de que ciertas mesas del establecimiento no estén operativas, dando la opción al propietario del establecimiento de marcar las mesas como desactivadas.

Debido a está decisión tomada, se ha optado por diferenciar la lista de mesas del establecimiento dependiendo del rol del usuario registrado en la aplicación, de tal manera que si el rol es propietario, este podrá visualizar todas las mesas registradas en el establecimiento y se dará la opción de poder editar, borrar y desactivar las mesas; mientras que si el rol es el de trabajador, no aparecerán los botones correspondientes a editar y borrar mesas y, en vez de aparecer el campo para activar o desactivar las mesas, aparcera un campo relacionado con el estado de la mesa, libre o ocupado, para que este pueda cambiarlo en caso de que un cliente haya abandonado el local tras su estancia en este y la mesa pueda ser ocupada por otra persona. Este último rol no se abordará en este sprint.

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
2.13	▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	6 horas y 42 minutos	Completada
2.3	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	1 hora y 55 minutos	Completada
2.14	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	18 horas	13 horas y 59 minutos	Completada
2.15	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 30 minutos	Completada
-	▪ Documentación Capítulo 2.		10 horas y 50 minutos	Completada
TOTAL		36 horas	36 horas y 56 minutos	4 de 4 HU completadas

Tabla 7.14: Seguimiento del Sprint 6

También se ha decidido añadir un Mat-Progress-Spinner [68] para dar retroalimentación al usuario al realizar la confirmación tanto de edición como de creación de mesas mientras se

está realizando la operación. Siguiendo este enfoque de dar retroalimentación, se ha optado por incluir mensajes de información al cambiar los valores del aforo máximo actual, para que así el usuario conozca al intentar confirmar los cambios si se han producido, no se ha podido cambiar o si el valor dado no es correcto por no cumplir las condiciones de menor o igual que la capacidad total de las mesas, y mayor que cero.

El aforo máximo actual de un establecimiento es una cantidad de aforo máximo permitido para la situación actual de un establecimiento, siendo siempre menor o igual que la suma de las capacidades de las mesas en estado activado, y siempre será menor o igual que el aforo máximo del establecimiento.
En caso de que no se cumpla alguna de estas condiciones, el propietario será avisado mediante un mensaje mostrado en la interfaz aclarando el problema para que lo solucione cuanto antes. También se ha añadido una condición de que, en caso de que la capacidad de las mesas en estado activado del establecimiento sea igual al aforo máximo del establecimiento, no se dará la opción al propietario de añadir más mesas, hasta que no elimine, edite la capacidad de una mesa a una cantidad menor, o desactive alguna de las mesas ya existentes en el establecimiento.

7.10. Sprint 7 (18/05/2022 - 01/06/2022)

El Sprint 7 comenzó el día 18 de mayo con la reunión correspondiente al Sprint Planning. En esta se decidió continuar con aspectos similares al sprint anterior, terminando el otro tipo de vista para la lista de mesas, correspondiente al rol Trabajador. Para ello se ha decidido abordar primero la gestión por parte del propietario del establecimiento, permitiendo la visualización, creación y borrado de estos, para después poder abordar la posibilidad de que un trabajador asigne como libre y ocupada una mesa del establecimiento donde trabaja. Una vez abordado este aspecto de la funcionalidad, ya se puede calcular el nivel de ocupación de un establecimiento, por lo que es otro de los aspectos a desarrollar en este sprint.
Los puntos de historia correspondientes a cada historia de usuario de este sprint se pueden visualizar en la Tabla 7.15

Historia de usuario	**Puntos de historia**
2.16	2
2.4	1
4.2	1
1.3	1

Tabla 7.15: Asignación de puntos de historia para el sprint 7

Durante este sprint se han dedicado 11 horas y 53 minutos al desarrollo de las cuatro historias de usuario correspondientes, y 4 horas y 43 minutos para terminar de completar la documentación correspondiente al Capítulo 4, haciendo un total de 16 horas y 36 minutos de las 30 horas propuestas para este sprint. Esta gran diferencia de horas dedicadas respecto a las estimadas viene de que gran parte de la funcionalidad desarrollada eran operaciones o implementación de aspectos de interfaz de usuario añadidas a componentes ya creados, o, en

caso de las dos primeras historias de usuario, por poder reutilizar aspectos de la interfaz de usuario. También cabe destacar que durante parte de la segunda semana de este sprint no se pudieron dedicar las horas que se querría por enfermedad del alumno.

Para comenzar el desarrollo de este sprint se optó por abordar la creación de trabajadores, para ello se barajaron diferentes opciones para la creación de la contraseña del usuario. Dichas opciones eran, la posibilidad de que el trabajador, tras iniciar sesión por primera vez con una contraseña generada, pudiese cambiar esta a una personal; que el sistema mandase un correo de confirmación al trabajador para que este decida que contraseña introducir y que el propietario otorgase una contraseña definitiva para el inicio de sesión del trabajador. Debido a que ya hay bastante funcionalidad desarrollada, para este punto del proyecto, se ha decido por utilizar la última de las opciones comentadas.

Debido a que el propietario es el encargado de decidir que contraseña tendrá el trabajador, se ha optado por incluir un botón que generará de forma aleatoria una contraseña de entre 8 y 12 caracteres, incluyendo una letra minúscula, una mayúscula y un número [37].

En este sprint se ha desarrollado la otra versión de la vista de las mesas del establecimiento, comenzada en el sprint anterior para el rol Propietario. Tras incluir el rol Trabajador y desarrollar la vista de las mesas para este rol, en la cual solo se podrán consultar aquellas que están activadas en el momento, para así dar la opción al trabajador de marcar una mesa como libre u ocupada; se ha podido abordar el cálculo y visualización del nivel de ocupación del local en tiempo real. A la hora de mostrar la ocupación a los usuarios de la plataforma se ha optado por diferenciar dos tipos de ocupación, una referida al número de mesas ocupadas y otra en relación al aforo máximo actual del local.

Tras añadir el rol del Trabajador y por la forma de recuperar datos en las peticiones, se observó un error al recuperar las mesas, por lo que se optó por añadir la anotación @JsonIgnore [18] al campo relativo al establecimiento de la entidad Mesa.

Para mostrar el nivel de ocupación del establecimiento se ha decidido utilizar “MatProgressBar“ [67], de tal forma que mediante dos barras de progreso se podrá observar de forma visual lo lleno o vacío que está un establecimiento en el momento de la consulta. Para poder mostrar de forma vertical las barras de progreso se ha tenido que consultar el funcionamiento de la función rotate de css [33], al igual que se han consultado diferentes maneras de modificar la apariencia de este tipo de componentes de Angular Material [81]. También se ha añadido mediante el componente Tooltip [72] de Angular Material, de forma textual, al colocar el cursor sobre las progress bar, el nivel de ocupación, tanto de las mesas como el aforo de personas.

Una de las decisiones tomadas al introducir el nivel de ocupación fue la de incorporar ambos niveles en la vista correspondiente a la carta del establecimiento, de tal forma que, en caso de que un usuario acceda a dicha información, pueda seguir visualizando, en la parte superior de la pantalla, la ocupación del establecimiento, manteniendo así dicha información siempre presente para el cliente.

H.U.	**Tareas**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
2.16	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	4 horas y 1 minuto	Completada
2.4	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	2 horas y 22 minutos	Completada
4.2	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	1 hora y 54 minutos	Completada
1.3	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	3 horas y 36 minutos	Aplicación de patrón Observador para mantener actualizada la ocupación. Retoques en etiqueta para la ocupación de las mesas.
	▪ Documentación Capítulo 4		4 horas y 43 minutos	Completado
TOTAL		30 horas	16 horas y 36 minutos	3 de 4 HU completadas

Tabla 7.16: Seguimiento del Sprint 7

Tras la reunión correspondiente al Scrum Weekly del 25 de mayo, otro de los cambios realizados con respecto a la primera versión de la lista de los trabajadores de un establecimiento, fue reducir los datos del campo fecha de contratación a simplemente mostrar el día de este, en formato año-mes-dia (aaaa-mm-dd), sin tener en cuenta la hora, ya que no es un dato significativo.

Por último se observó un problema al recuperar las mesas en la entidad Establecimiento, de tal manera que se recuperaban de forma duplicada. Para solucionar esto se ha encontrado información sobre como prevenir la aparición de datos duplicados [78], por lo que se decidido cambiar el atributo mesas del establecimiento de una lista (List) a un conjunto (Set), de tal forma que no existirían duplicados al recuperar los datos. Debido a que la funcionalidad estaba realizada en su totalidad mediante el uso de listas, se procedió a convertir, al recuperar y guardar los datos relativos a las mesas, de conjunto a lista y viceversa, respectivamente, para ello se consultó la página Baeldung, donde se encontró como realizar dichas conversiones [16].

En el Sprint Retrospective del día 1 de junio, se comentó la posibilidad de aplicar el patrón Observador para mantener los valores de la ocupación del establecimiento actualizados en tiempo real, y varios cambios en las etiquetas utilizadas para mostrar la ocupación del local, por ello la historia de usuario 1.3 pasa al siguiente sprint.

7.11. Sprint 8 (01/06/2022 - 15/06/2022)

En la reunión correspondiente al Sprint Planning del día 1 de junio, se decidió abordar en este sprint el desarrollo de la funcionalidad relativa a la realización de comandas por parte de un cliente. A su vez, se incluyó la finalización de la historia de usuario 1.3, que quedó sin completar en el sprint anterior.

Historia de usuario	Puntos de historia
1.3	1
3.1	2
4.1	3

Tabla 7.17: Asignación de puntos de historia para el sprint 8

Para desarrollar la funcionalidad de realizar comandas se decidió incorporar un botón con una imagen de un carrito de la compra, similar a una cesta de compra de multitud de sitios web, además de utilizar el componente “Badge“ [61] de Angular Material, que se encargará de, mediante una burbuja sobre dicha cesta de productos seleccionados, marcar la cantidad de productos diferentes que existen en la comanda, independientemente de la cantidad de unidades de cada producto pedidas.

Se añadió a cada producto un selector de cantidades [86] y un botón para añadir a la comanda, para evitar posibles cantidades erróneas, se optó por solamente poder añadir o

quitar unidades mediante los botones destinados a ello, y deshabilitando la posibilidad de reducir la cantidad o pedir, si esta es 0.

Para que la vista resumen del pedido tuviera los datos de la comanda, se optó por utilizar un servicio que actuaría de Observador [102], compartiendo así los datos entre la vista de la comanda y la vista con la carta del establecimiento. En esta vista resumen también se incorporó a la tabla un pie de tabla con el resumen del pedido, mostrando la cantidad de productos pedidos y el precio de la comanda a realizar. Se optó por que esta fila resumen se mantuviera siempre a la vista del usuario, por lo que se consultó la documentación acerca del componente MatTable [71] para contemplar si existía la propiedad buscada, observando que mediante el argumento "position: sticky" se conseguía el resultado querido. Esto también dio la idea de colocar dicha propiedad al botón del carrito de la compra, para que así, en caso de cartas con muchos productos diferentes, siempre se tenga el botón en pantalla [50].

Puesto que las lineas de comanda estaban diseñadas para contener un producto y sus cantidades, esto no permitía la posibilidad de pedir el menú de la carta, por lo que se decidió que tanto los productos como los menús pasasen a ser un tipo extendido de Pedible, lo que conllevó una modificación del diseño de datos de la aplicación. Una vez resuelto este problema, e incorporado tanto el selector de cantidades, como el botón para añadir a la comanda, al igual que con los productos que componen la carta, se finalizó la funcionalidad de añadir productos de una carta a la comanda. Para dar una mejor experiencia al usuario, se decidió incorporar en la vista resumen de la comanda, la posibilidad de eliminar alguno de los productos que componen esta.

En primera instancia, cada vez que se añadía una cantidad de un producto a la comanda, se creaba una linea de comanda nueva para este producto. Finalmente se optó por agrupar en la misma linea de comanda la cantidad total de ese mismo producto, es decir, en caso de que un producto haya sido pedido ya y se vuelva a pedir, la cantidad resultante de ese producto será la suma de ambos pedidos, al igual que se actualizará el importe a la cantidad nueva.

Otro aspecto que se abordó en este punto del desarrollo fue como asociar la comanda a la mesa, para ello se valoraron dos opciones. Una era incluir un qr que permitiría la posibilidad de realizar la comanda, y teniendo una mesa asociada al qr, se conocería el identificador de la mesa para la creación de la comanda en el backend como la posibilidad de marcarla como ocupada; la otra opción y la elegida, ha sido colocar en la vista donde se ven los productos de establecimiento un campo de texto donde el cliente escribirá el identificador propuesto por el propietario para cada mesa y, si este es correcto con una de las mesas del local, habilitará la posibilidad de añadir productos al pedido y marcará como ocupada la mesa correspondiente si no lo estaba.

Se consultó información sobre como sumar horas a un objeto de tipo Date dentro de TypeScript, debido a que al crear la comanda, utilizaba una zona horaria que no era la de España, dicha solución se encontró en Stack Overflow [87], para lo cual se sumarían dos horas en milisegundos a la hora actual.

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
1.3	▪ Análisis. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	6 horas	7 horas y 57 minutos	Completada
3.1	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	12 horas	18 horas y 27 minutos	Completada
4.1	▪ Análisis. ▪ Diseño. ▪ Implementación. ▪ Revisión IU. ▪ Testeo.	18 horas	11 horas y 25 minutos	Completada
-	▪ Documentación Capítulo 5	-	2 hora y 18 minutos	Sin finalizar
TOTAL		36 horas	40 horas y 7 minutos	3 de 3 HU completadas

Tabla 7.18: Seguimiento del Sprint 8

Para realizar diferentes asignaciones de valores booleanos se recordó el funcionamiento del operador condicional ternario [32] y su funcionamiento en TypeScript.

También se abordó un problema con las rutas restringidas a uno o varios roles. Debido a que durante otros sprints no se tuvo en cuenta este problema y solo existía una "guarda", con la función de que si el usuario había iniciado sesión podía acceder a dichas rutas restringidas, pero si un determinado rol accedía mediante URL a una vista que no debería poder visualizar, debido a su rol, este si podría acceder a ella. Para ello se han creado cuatro guardas diferentes, una para el rol de administrador, otra para los trabajadores, otra para los propietarios y una última guarda "maestra" necesaria para que dos roles puedan acceder a una misma

vista. Esta decisión de crear una guarda "'maestra" se encontró en Stack Overflow, buscando información acerca de como dar la posibilidad de acceder a una ruta determinada a dos roles con dos "guardas" diferentes [92], consiguiendo así que tanto el propietario como el trabajador puedan acceder a la misma página, en este caso concreto, a la lista de mesas de un establecimiento.

Realizando pruebas tras este último cambio se observó que se lanzaba un error al intentar acceder vía url a una ruta no existente, tras consultar la documentación de Angular sobre Router, dicho problema se soluciona añadiendo una ruta comodín [12], que redireccionará todas estas rutas inexistentes a una que se decida, en este proyecto, a la lista de establecimientos.

Para completar la historia de usuario 1.3, que no se completó en el sprint anterior, se buscó información sobre como realizar de manera dinámica cambios en una vista si desde otro cliente web se ha realizado una operación que altera algún campo en la vista actual. Para ello se comenzó incorporando un servicio a modo de Observador en el cliente angular, gracias al cual los componentes compartirían datos entre si, pero no solucionaría el problema, ya que solo comparten datos dentro de la misma instancia de la aplicación, no con otras pestañas o navegadores. Se buscó información sobre la posibilidad de realizar cambios en una vista al alterar los valores de la base de datos del servidor, y en Stack Overflow se encontraron dos soluciones [85], la primera consiste en realizar una petición al servidor cada cierto tiempo, mediante un timeout, y la segunda mediante la creación de un servicio que utiliza websockets [35] para las comunicaciones. Se optó por realizar la segunda opción, una comunicación mediante websockets, consiguiendo así optimizar la cantidad de peticiones que se realizan sobre el servidor, y realizando actualizaciones de los datos solo cuando sea necesario para aquellos componentes conectados mediante el websocket.

Para la realización de esta conexión se han consultado dos repositorios GitHub [25] [26] con un ejemplo para la realización de un chat en vivo, el cual se adaptó al problema encontrado consiguiendo el resultado deseado.

Aprovechando esta comunicación creada para la ocupación se ha decidido incorporarla también para mantener en tiempo real el estado de las mesas como libres u ocupadas, para que así si existen varios trabajadores al mismo tiempo en esta vista, y uno realiza un cambio de ocupación de alguna de las mesa del establecimiento, este cambio lo vean reflejado el resto de trabajadores, sin necesidad de recargar la página. También se ha utilizado en la lista de establecimientos, donde se muestra la ocupación, aportando así dicho dato en tiempo real al cliente.

Para llevar a cabo la ultima historia de usuario de este sprint, que consistía en poder ver en tiempo real las comandas activas, es decir, cuyo estado no es "PAGADA", se ha optado por cambiar el identificador de la comanda a una cadena personalizada por cada establecimiento y mesa, para ello se han consultado las diferentes formas de generar identificadores de JPA [20] y si existe una forma de crear este de forma personalizada, obteniendo la solución en Stack Overflow [89], donde mediante una consulta a la base de datos para obtener la cantidad de comandas por mesa actuales, y conociendo el objeto comanda al que asignar el id, se puede construir el identificador, para el cual se ha elegido un código alfanumérico siguiendo el siguiente patrón "ES<ID del establecimiento>-<Identificador de la mesa dentro del local>-

N<Número de la comanda para esa mesa>“, gracias al cual podríamos diferenciar, dentro del establecimiento, el numero de comanda para la mesa en cuestión. Este código se podría mejorar pero, por el momento, y debido a la pequeña cantidad de datos que se van a manejar en este proyecto, el identificador que se ha elegido para el establecimiento es su id único dentro de la base de datos, y el número de la comanda es generado de forma incremental dentro de la mesa a la que pertenece, en un futuro se podría también tener en cuenta la fecha, para que cada nuevo día este número comenzase en 1 para cada mesa. De esta manera las comandas creadas tendrán un identificador único personalizado por el establecimiento y la mesa en la que se han realizado.

Utilizando lo implementado anteriormente sobre web sockets, se ha adaptado un nuevo web socket para las comandas, de tal manera que las comandas realizadas por algún cliente salgan automáticamente en la vista que lista las comandas activas de una mesa, si es que algún trabajador se encuentra dentro de esta, de tal forma que pueda ver en tiempo real cuales son las comandas y en que estado están para cada mesa.

Para ello se consultó como tener varios web sockets activos para el mismo servidor, se encontró la solución en Stack Overflow [94], consiguiendo así que dependiendo de que se necesite, se conecten a un web socket u al otro, para diferenciar los “endpoints“ se ha decidido nombrar la URL para lo relativo a la ocupación de locales como “topic/ocupacion“ y el relativo a las comandas como “topic/comandas“.

Una vez todo esto implementado se observó un problema al recuperar ciertos productos de las comandas si existían en algún menú perteneciente a otra comanda. Para solucionar esto se busco información sobre como obtener el tipo de un objeto, encontrando en Stack Overflow la solución [91], consiguiendo solucionar el problema para poder diferenciar si un producto es menú o no.

El sprint 8 acabó el día 15 de junio habiendo utilizado 37 horas y 49 minutos en el desarrollo de las tres historias de usuario planificadas para realizar en este y de 2 horas y 18 minutos en continuar documentando el Capítulo 5, lo que hace un total de 40 horas y 7 minutos, superando las 36 horas estimadas para este sprint.

7.12. Sprint de refuerzo 1 (15/06/2022 - 29/06/2022)

En este primer sprint de refuerzo, de dos semanas de duración, se ha decidido únicamente estimar tiempo para las tareas pendientes para finalizar el proyecto relacionadas con mejoras en la interfaz de usuario, puesto que todas las demás tareas que se van a realizar son de depuración y documentación. Para ello se ha propuesto una única historia de usuario a llevar a cabo en este sprint. La asignación de los puntos para esta historia de usuario se puede ver en la tabla siguiente (Tabla 7.19).

Historia de usuario	Puntos de historia
6.2	2

Tabla 7.19: Asignación de puntos de historia para el sprint 8

Para llevar a cabo las mejoras en la interfaz de usuario, lo primero que se realizó fue cambiar los campos de texto de los formularios para crear productos y establecimientos a "mat-input" para seguir con la homogeneización de la plataforma, también se revisaron y añadieron, en los campos que se pensó que sería útil para el usuario, cadenas de texto informativo usando las "mat-hint" [63] de Angular Material.

Se configuraron algunos títulos de ciertas páginas con doble funcionalidad, como es el caso de aquellas que se utilizan, dependiendo del caso, para crear o editar algún establecimiento, producto o mesa, para que dependiendo cual fuera la acción a llevar a cabo, el título fuese acorde a esta.

Se ha añadido el "mat-spinner" de retroalimentación en el resto de vista en los cuales una acción podría acarrear una espera por la comunicación con el servidor, para que así el usuario no piense que su acción no se ha llevado a cabo.

Se han vinculado los cambios en los campos de texto de la creación o edición de establecimientos relativos a la latitud y longitud de las coordenadas del local al mapa, es decir, si el propietario decide introducir a mano ambos campos, a medida que va introduciendo los datos, verá como la posición del mapa varía, colocándose en dichas coordenadas.

También se ha depurado el login de usuarios, ya que en caso de que la comunicación entre el cliente y el servidor se demorase mucho, se mostraría el mensaje de error, cuando el usuario podría haber introducido bien los datos de login. Al igual que se ha añadido una condición de que si el usuario ya ha iniciado sesión en la plataforma, no se permite el acceso tanto a la página de inicio de sesión como al registro, si no que será redirigido a la lista de establecimientos.

Se han añadido crucetas en los filtros que existen en las diferentes páginas que forman la plataforma, para permitir al usuario borrar el contenido del campo de forma rápida.

Se ha controlado el acceso mediante url de los propietarios y trabajadores a las páginas de gestión de los diferentes datos alojados en la aplicación, a los cuales no deberían poder acceder, es decir, un propietario no podrá acceder vía url a la gestión de un establecimiento que no sea suyo, al igual que un trabajador no podrá acceder a la lista de mesas de un establecimiento y las correspondientes comandas activas de cada mesa, si no trabaja en dicho local.

Otro de los cambios introducidos en este punto, para potenciar la retroalimentación dada al usuario de la plataforma, fue la incorporación de "MatSnackBar" [70], un componente de Angular Material para mostrar snackbars en las aplicaciones. Este snackbar se optó por mostrarlo en la esquina inferior derecha, a modo que interfiera lo mínimo posible en la interacción del usuario con la plataforma. A estos snackbars se les asignó una duración en pantalla de 5 segundos, aunque también se da la opción al usuario de cerrarlo si lo desea, presionando el texto "Cerrar" que aparece. En este popup informativo se quiere brindar al usuario de retroalimentación tras realizar una acción dentro de la plataforma, ya sea para confirmar que dicha acción se ha llevado a cabo con éxito o si ha fallado algo en el proceso. En caso de que por algún problema en la comunicación entre cliente y servidor al obtener datos en alguna de las páginas, también se mostrará este snackbar para avisar del problema.

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
6.1	▪ Revisión código. ▪ Revisión IU. ▪ Testeo.	12 horas	13 horas y 17 minutos	Completada
-	▪ Documentación Capítulo 1	-	2 horas y 17 minutos	Completada
-	▪ Documentación Capítulo 3	-	2 horas y 32 minutos	Pendiente de finalizar
-	▪ Documentación Capítulo 5	-	2 horas y 11 minutos	Pendiente de finalizar
-	▪ Documentación Capítulo 6	-	12 horas y 14 minutos	Pendiente de finalizar
-	▪ Documentación Capítulo 8	-	35 minutos	Pendiente de finalizar
-	▪ Documentación Manuales	-	1 hora y 56 minutos	Pendiente de finalizar
TOTAL		12 horas	35 horas y 2 minutos	1 de 1 HU completadas

Tabla 7.20: Seguimiento del primer Sprint de refuerzo

También se ha añadido una gama de colores en la ventana de la lista de establecimientos de un propietario, para los cuales se ha utilizado el color amarillo para aquellas solicitudes pendientes y rojos para aquellas que han sido rechazadas, manteniendo el color blanco para las validadas. Para llevar a cabo esto se ha utilizado el punto del ciclo de vida de un componente Angular denominado AfterViewChecked [10], con su método "ngAfterViewChecked()", el cual se lleva a cabo inmediatamente después de que el detector de cambios predeterminado haya completado un ciclo de verificación de cambios para la vista del componente.

El primer sprint de refuerzo finalizó el día 29 de junio con un total de 35 horas y 2 minutos, de las cuales 13 horas y 17 minutos se dedicaron a mejoras en la interfaz de usuario y depuración de código, y las restantes 21 horas y 45 minutos para seguir realizando tareas de documentación de la memoria. La repartición de horas entre las tareas llevadas a cabo en este sprint y el estado de cada una se puede observar en la Tabla 7.20.

7.13. Sprint de refuerzo 2 (29/06/2022 - 13/07/2022)

En este último sprint de refuerzo se va a dedicar tiempo principalmente a terminar de documentar los capítulos pendientes de finalizar de la memoria y a realizar las tareas pertinentes para el despliegue de la plataforma. También se incluirán las tareas de revisión por parte de la tutora y las correcciones que se tengan que realizar.

H.U.	Tareas	Tiempo estimado	Tiempo empleado	Estado
-	▪ Documentación Capítulo 3	-	1 hora y 27 minutos	Completada
-	▪ Documentación Capítulo 4	-	42 minutos	Completada
-	▪ Documentación Capítulo 5	-	1 hora y 57 minutos	Completada
-	▪ Documentación Capítulo 6	-	2 hora y 22 minutos	Completada
-	▪ Documentación Capítulo 7	-	2 horas y 54 minutos	Completada
-	▪ Documentación Capítulo 8	-	1 hora y 32 minutos	Completada
-	▪ Documentación Manuales	-	5 horas y 56 minutos	Completada
-	▪ Revisión de toda la documentación	-	9 hora y 46 minutos	Completada
TOTAL			26 horas y 36 minutos	

Tabla 7.21: Seguimiento del segundo Sprint de refuerzo

En este sprint se ha llevado a cabo el despliegue de la aplicación. Para realizar este despliegue se ha optado por utilizar la herramienta de servicios en la nube Heroku [54], la cual proporciona las necesidades requeridas para poder realizar esta tarea con éxito. Tras una búsqueda de información a cerca de como desplegar proyectos en Heroku, descubrí que es necesario que el proyecto que se almacena en cada "dyno", contenedor virtual de Heroku, solo tenga un lenguaje de programación, por lo que el proyecto tal y como se estaba desarrollando no podía ser desplegado, ya que el mismo repositorio GitLab contenía tanto el backend como el frontend de la plataforma, resultando así en conflictos a la hora de realizar el despliegue ya que no detectaba el lenguaje de programación. Para solucionar este problema se ha decidido

mantener el repositorio original con el código fuente de ambas partes del proyecto, y crear dos repositorios a mayores para el despliegue de cada parte por separado, gracias a estos, se ha podido desplegar con éxito ambas partes del proyecto.

Para el despliegue del backend hubo que crear un archivo *system.properties* donde se alojó la versión de Java que tiene que usar Heroku para el despliegue y se borraron del archivo *application.properties* aquellas variables relacionadas con la dirección, usuario y contraseña de la base de datos

Para llevar a cabo el despliegue del frontend hubo que añadir Express al proyecto, ya que Heroku da soporte para NodeJS, y así poder desplegar el proyecto correctamente. Para ello se ejecuto el comando `npm i express` y posteriormente se creo el fichero *server.js* en el directorio raíz del proyecto, para así poder crear una pequeña aplicación en Express. También se añadieron al archivo *package.json* las versiones de node y npm utilizadas para el despliegue en Heroku, se movieron ciertas dependencias de la sección *devDependencies* a *dependencies* y se actualizó el script *"start"* por el arranque de la aplicación mediante el archivo creado `server.js`

Otro de los cambios que se realizaron para adaptar la parte dedicada al frontend de la plataforma al despliegue fue cambiar la URL sobre la que se hacían las peticiones HTTP, apuntando ahora a la dirección sobre la que se ha desplegado el backend y el cambio de usar websocket a webscoket secure, debido a que Heroku despliega las aplicaciones utilizando HTTPS, por lo que es necesario utilizar un websocket que acepte certificados SSL.

Este segundo sprint de refuerzo dedicado a la finalización de la documentación del proyecto ha llevado un total de 26 horas y 36 minutos.

7.14. Resumen del seguimiento

Durante los 5 meses de desarrollo de este proyecto se han realizado 10 sprints, sin contar el sprint 0, usado como inicialización, siendo los 8 primeros aquellos donde se han realizado, principalmente, las funcionalidades esenciales de la plataforma, y los dos sprints siguientes, utilizados como sprints de refuerzo, han servido para pulir detalles de la interfaz de usuarios y tareas principalmente de documentación y pruebas.

En cuanto a las historias de usuario realizadas podemos decir:

- **Total de historias de usuario abordadas**: Se han abordado un total de 30 historias de usuario.
- **Historias de usuario completadas**: Se han completado 29 historias de usuario del total.
- **Historias de usuario en pausa**: Solamente una historia de usuario abordada no se completo, quedando en pausa. Esta historia de usuario se abordó en el Sprint 2 (7.4).
- **Historias de usuario desplazadas**: Se han pospuesto de un sprint a otro un total de 5 historias de usuario.

La distribución de las horas dedicadas al desarrollo de la funcionalidad de la plataforma por historia de usuario se puede observar en la Tabla 7.22

H.U.	Horas estimadas	Horas empleadas
1.1	24 horas	29 horas y 25 minutos
1.2	6 horas	2 horas y 45 minutos
1.3	6 horas	11 horas y 33 minutos
1.6	6 horas	9 horas y 55 minutos
1.7	6 horas	4 horas y 30 minutos
2.1	12 horas	11 horas y 40 minutos
2.2	6 horas	2 horas y 5 minutos
2.3	6 horas	1 hora y 55 minutos
2.4	6 horas	2 horas y 22 minutos
2.5	12 horas	16 horas y 33 minutos
2.6	6 horas	3 horas y 51 minutos
2.7	12 horas	7 horas y 5 minutos
2.8	12 horas	2 horas y 37 minutos
2.9	6 horas	3 horas y 17 minutos
2.10	6 horas	1 hora y 32 minutos

H.U.	Horas estimadas	Horas empleadas
2.11	6 horas	5 horas y 9 minutos
2.12	6 horas	3 horas y 25 minutos
2.13	12 horas	17 horas y 50 minutos
2.14	18 horas	13 horas y 59 minutos
2.15	6 horas	3 horas y 30 minutos
2.16	12 horas	4 horas y 1 minuto
3.1	12 horas	18 horas y 27 minutos
4.1	18 horas	11 horas y 25 minutos
4.2	6 horas	1 hora y 54 minutos
5.1	12 horas	11 horas y 32 minutos
5.2	6 horas	2 horas y 24 minutos
5.3	6 horas	4 horas y 19 minutos
5.4	6 horas	1 hora y 8 minutos
6.1	12 horas	12 horas y 50 minutos
6.2	12 horas	13 horas y 17 minutos

Tabla 7.22: Distribución de horas por historia de usuario.

Sobre la planificación de los sprints y eventos a lo largo de los 5 meses de desarrollo, hay que comentar que se ha respetado en la forma de lo posible. Se han realizado en las fechas declaradas en la Tabla 2.1 todos los eventos marcados, a excepción de los eventos de final del Sprint 8 y comienzo del Sprint de refuerzo 1, y el Scrum Weekly de este último, ya que por motivos de disponibilidad por carga de trabajo tanto de la tutora como del alumno no se pudieron realizar. Para solucionar este problema, se intento definir con antelación ciertos aspectos sobre futuras acciones a realizar y se resolvieron las dudas que iban surgiendo mediante texto gracias a los medios de comunicación utilizados.

Con respecto al análisis de riesgos realizado en el Capítulo 2, cabe destacar la ocurrencia del Riesgo R01 (Tabla 2.2), enfermedad del desarrollador, en dos ocasiones a lo largo del proyecto, lo que complicó el poder dedicar las horas deseadas al desarrollo de los sprints donde tuvo presencia. Por otro lado el Riesgo R09 (Tabla 2.10), problemas de conectividad, tuvo mayor importancia en las primeras semanas de desarrollo, ya que un problema en el proceso de cambio de compañía que suministra el acceso a red, provocó tener que aplicar los planes de contingencia definidos. En cuanto el problema surgido fue solucionado, no volvió a tener presencia durante el resto del proyecto.

El proyecto se desarrolló desde el 9 de febrero de 2022, con el inicio del Sprint 0, hasta el 13 de julio del mismo año, con una duración de aproximadamente 5 meses y un total de 312 horas y 33 minutos empleados en el desarrollo del mismo.

Capítulo 8

Conclusiones

8.1. Conclusiones

Tras la finalización del proyecto se puede describir que el proceso de desarrollo del mismo ha sido un trabajo duro y prolongado en el tiempo durante más del que estaba acostumbrado gracias a las prácticas de las diferentes asignaturas del grado, pero también satisfactorio y gratificante, ya que durante la realización del mismo se han ido consiguiendo diferentes logros hasta concluir con la plataforma en su totalidad, añadido a los aspectos aprendidos y la mejora en el manejo tanto de los lenguajes utilizados, como el aprendizaje de las tecnologías desconocidas con anterioridad a comenzar el proyecto.

En cuanto a los objetivos de desarrollo definidos al inicio del desarrollo, en la Sección 1.4.1:

- Se ha conseguido que el propietario sea capaz de gestionar los establecimientos, tanto las mesas, como los empleados y la carta de cada uno de ellos.
- Se ha conseguido poder gestionar las peticiones de registro de propietarios y establecimientos por parte del administrador de la plataforma.
- Se ha conseguido que un usuario trabajador pueda gestionar las mesas y las comandas activas de cada mesa del establecimiento donde trabaja.
- Se ha conseguido disponer del nivel de ocupación de un local en tiempo real en diferentes partes de la plataforma, para que el usuario tenga presente el dato en ellas.
- Se ha conseguido brindar al cliente de la posibilidad de realizar comandas en el establecimiento.

Como se declaró en la sección 1.4.1, la posibilidad de reservar una mesa para un cliente no se ha abordado por el alcance del proyecto, pero si se piensa que es una muy buena

funcionalidad a poder incluir en un futuro, por lo que se ha añadido a las líneas de trabajo futuras, descritas en la sección 8.2

Por otro lado, en cuanto a los objetivos personales definidos también al inicio del desarrollo, en la Sección 1.4.2:

- Se ha llevado a cabo un proyecto de desarrollo software de principio a fin, incluyendo todas sus fases y complicaciones en cada una de ellas, aportando un gran aprendizaje durante los meses de duración de este.
- Se ha comprendido de mejor manera el marco de trabajo Scrum para el desarrollo de un proyecto, viviendo personalmente las reuniones semanales y la descripción de cada una de ellas en el documento de memoria del proyecto, en el Capítulo 7.
- Se ha desarrollado gran cantidad de código en el lenguaje Java, para la realización del servidor de la plataforma, lo que ha mejorado mis capacidades con dicho lenguaje y con las diferentes tecnologías utilizadas en el mismo, incluyendo el uso y gestión de la base de datos.
- Se ha desarrollado gran cantidad de código TypeScript para la realización del cliente de la plataforma, mejorando así mis capacidades de desarrollo con dicho lenguaje y aprendiendo el uso de diferentes componentes ya existentes que se pueden incorporar a un proyecto de este tipo.

8.2. Líneas de trabajo futuras

Debido a que el sector de la restauración es muy extenso y por la duración de este proyecto no se pueden abordar todas las funcionalidades necesarias para abarcar todos estos aspectos, hay bastantes aspectos que se podrían desarrollar en un futuro, como podrían ser:

- Terminar de desarrollar la historia de usuario 1.6, que no se finalizó, relativa a poder insertar imágenes a los establecimientos y productos.
- Asignación de horarios por día de la semana a cada establecimiento, lo cual aportaría al cliente más información acerca del establecimiento.
- Muestra de la ocupación por día de la semana y franja horaria, aportando así una posible estimación de la ocupación de los locales en otro instante.
- La utilización de la geolocalización del dispositivo que utiliza la plataforma, mediante confirmación del usuario, para poder ordenar la lista de establecimientos por proximidad, y en caso de un propietario, poder asignar las coordenadas a la hora de crear o editar un establecimiento mediante geolocalización.
- Desarrollo de una ventana de perfil para que los usuarios registrados puedan consultar sus datos y editarlos si lo ven necesario.

- Navegación a una ficha con la información del propietario de un establecimiento con una solicitud de validación activa.
- Permitir a un propietario la posibilidad de duplicar o copiar la carta de uno de sus establecimientos a otro.
- Permitir incorporar más de un menú a la carta de un establecimiento.
- Permitir a un propietario borrar alguno de sus establecimientos.
- Permitir a un propietario poder eliminar de la plantilla de uno de sus establecimientos a un trabajador.
- Permitir a un administrador eliminar de la plataforma un establecimiento o a un propietario y sus establecimientos.
- Poder cambiar, en el resumen de un pedido antes de realizarlo, la cantidad de unidades de un producto, sin tener que navegar a la carta para incrementar la cantidad.
- Ampliar el abanico de roles dentro de los trabajadores, repartiendo ciertas funcionalidades para cada uno de estos roles.
- Separar por categorías los productos.
- Permitir a los trabajadores poder agrupar todas, o la cantidad que se desee de las comandas activas de una mesa, en una sola.
- Desarrollar el sistema de pago de comandas para el cliente, permitiendo el pago fraccionado en partes iguales, el pago en su totalidad o incluso mediante pagos fraccionados personalizados.
- Utilización de un QR para permitir realizar una comanda, en vez del identificador de la mesa.
- Incluir un sistema de reserva de mesas para un rol nuevo de cliente registrado.
- Utilización de un sistema de confirmación mediante correos electrónicos para la confirmación de registro de propietarios y trabajadores, y la notificación de cambio de estado de un establecimiento o propietario por parte de un administrador.
- Mejora de la asignación de contraseñas para los trabajadores, dando la posibilidad de que sean estos quienes decidan que contraseña utilizar en la plataforma, y que no corra a cargo del propietario del establecimiento donde trabajan.
- Incluir un sistema de pedidos a domicilio si es que el establecimiento lo permite.

Bibliografía

[1] Balsamiq wireframes. https://balsamiq.com/. Accedido el 04/05/2022.

[2] Collapse. toggle the visibility of content across your project with a few classes and our javascript plugins. https://getbootstrap.com/docs/4.0/components/collapse/. Accedido el 28/02/2022.

[3] Grid system. https://getbootstrap.com/docs/4.0/layout/grid/. Accedido el 26/02/2022.

[4] Latex – a document preparation system. https://www.latex-project.org/. Accedido el 03/05/2022.

[5] Openlayers, a high-performance, feature-packed library for all your mapping needs. https://openlayers.org. Accedido el 05/03/2022.

[6] Overleaf, el editor de latex fácil de usar, online y colaborativo. https://es.overleaf.com/. Accedido el 04/05/2022.

[7] Spring boot. https://spring.io/projects/spring-boot. Accedido el 03/05/2022.

[8] Vladimir Agafonkin. Leaflet, an open-source javascript library for mobile-friendly interactive maps. https://leafletjs.com/. Accedido el 05/03/2022.

[9] José Manuel Alarcón. Campusmvp. la api de persistencia de java: ¿qué es jpa? - jpa vs hibernate vs eclipselink vs spring jpa. https://www.campusmvp.es/recursos/post/la-api-de-persistencia-de-java-que-es-jpa-jpa-vs-hibernate-vs-eclipselink-vs-sp aspx. Accedido el 03/05/2022.

[10] Angular. Afterviewchecked. https://angular.io/api/core/AfterViewChecked. Accedido el 20/06/2022.

[11] Angular. Angular. https://angular.io/. Accedido el 04/05/2022.

[12] Angular. Common routing tasks. setting up wildcard routes. https://angular.io/guide/router#setting-up-wildcard-routes. Accedido el 04/06/2022.

[13] Andrew W. Appel. Verification of a cryptographic primitive: Sha-256. https://www.cs.princeton.edu/~appel/papers/verif-sha.pdf. Accedido el 28/05/2022.

[14] Arteco. ¿por qué debes usar spring boot? https://www.arteco-consulting.com/post/por-que-debes-usar-spring-boot. Accedido el 03/05/2022.

[15] Astah. Astah professional: Uml, er, dfd & flowchart software. https://astah.net/products/astah-professional/. Accedido el 03/05/2022.

[16] Baeldung. Converting between a list and a set in java. https://www.baeldung.com/convert-list-to-set-and-set-to-list. Accedido el 29/05/2022.

[17] Baeldung. Hibernate inheritance mapping. https://www.baeldung.com/hibernate-inheritance. Accedido el 07/03/2022.

[18] Baeldung. Jackson ignore properties on marshalling. https://www.baeldung.com/jackson-ignore-properties-on-serialization. Accedido el 15/05/2022.

[19] Baeldung. @joincolumn annotation explained. https://www.baeldung.com/jpa-join-column. Accedido el 14/03/2022.

[20] Baeldung. An overview of identifiers in hibernate/jpa. https://www.baeldung.com/hibernate-identifiers. Accedido el 09/06/2022.

[21] Oscar Blancarte. Data access object (dao) pattern. https://www.oscarblancarteblog.com/2018/12/10/data-access-object-dao-pattern/. Accedido el 22/06/2022.

[22] Oscar Blancarte. Data transfer object (dto) – patrón de diseño. https://www.oscarblancarteblog.com/2018/11/30/data-transfer-object-dto-patron-diseno/. Accedido el 22/06/2022.

[23] Carlos Blanco. Lista de comprobaciÓn de riesgos en proyectos software. https://ocw.unican.es/pluginfile.php/274/course/section/196/Lista%20de%20Riesgos%20en%20proyectos%20SW.pdf. Accedido el 01/05/2022.

[24] Bravent. Xamarin.forms y mvvm. https://www.bravent.net/xamarin-forms-y-mvvm/. Accedido el 13/06/2022.

[25] codeforgeyt. Github. java-ws-chat-application. https://github.com/codeforgeyt/java-ws-chat-application. Accedido el 05/06/2022.

[26] codeforgeyt. Github. ws-chat-application. https://github.com/codeforgeyt/ws-chat-application. Accedido el 05/06/2022.

[27] Oracle Corporation. Mysql. https://www.mysql.com/. Accedido el 03/05/2022.

[28] czeckd. angular-dual-listbox. https://github.com/czeckd/angular-dual-listbox. Accedido el 06/04/2022.

[29] Pablo J. Carricondo D. Agilidad, scrum y campañas politicas. http://neurocambioyriesgo.blogspot.com/2018/03/agilidad-scrum-y-campanas-politicas.html. Accedido el 01/05/2022.

[30] Dev. localstorage vs sessionstorage. https://dev.to/caffiendkitten/localstorage-vs-sessionstorage-f9k. Accedido el 14/03/2022.

[31] Dinahosting. ¿qué es apache y para qué sirve? https://dinahosting.com/ayuda/que-es-apache-y-para-que-sirve/. Accedido el 03/05/2022.

[32] MDN Web Docs. Operador condicional (ternario). https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/Conditional_Operator. Accedido el 02/06/2022.

[33] MDN Web Docs. rotate(). https://developer.mozilla.org/es/docs/Web/CSS/transform-function/rotate. Accedido el 28/05/2022.

[34] MDN Web Docs. Spa (single-page application. https://developer.mozilla.org/en-US/docs/Glossary/SPA. Accedido el 04/05/2022.

[35] MDN Web Docs. Websockets. https://developer.mozilla.org/es/docs/Web/API/WebSockets_API. Accedido el 05/06/2022.

[36] Alfred Menezes & Scott Vanstone Don Johnson. The elliptic curve digital signature algorithm (ecdsa). https://web.archive.org/web/20160304101319/http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf. Accedido el 28/05/2022.

[37] ForoAyuda. Ejemplo de código generador de contraseña aleatoria angular. https://foroayuda.es/ejemplo-de-codigo-generador-de-contrasena-aleatoria-angular/. Accedido el 20/05/2022.

[38] OpenJS Foundation. Acerca de node.js. https://nodejs.org/es/about/. Accedido el 02/04/2022.

[39] The Apache Software Foundation. Apache tomcat. https://tomcat.apache.org/. Accedido el 03/05/2022.

[40] GeeksforGeeks. How to make a bootstrap modal popup in angular 9/8 ? https://www.geeksforgeeks.org/how-to-make-a-bootstrap-modal-popup-in-angular-9-8/. Accedido el 20/03/2022.

[41] Git. About. https://git-scm.com/about. Accedido el 28/05/2022.

[42] Git. About. staging area. https://git-scm.com/about/staging-area. Accedido el 28/05/2022.

[43] GitLab. Issues. https://docs.gitlab.com/ee/user/project/issues/. Accedido el 28/05/2022.

[44] GitLab. What is gitlab? https://about.gitlab.com/what-is-gitlab/. Accedido el 28/05/2022.

[45] Marcos Vázquez González. Codmind, ¿qué es spring boot? https://blog.codmind.com/que-es-spring-boot/#:~:text=Spring%20Boot%20es%20una%20tecnolog%C3%ADa,servidor%20de%20aplicaciones%20y%20enfocarnos. Accedido el 03/05/2022.

[46] Yania Crespo González-Carvajal. Apuntes del tema 5 de la asignatura diseño de software. Accedido el 22/06/2022.

[47] Google. What is v8? `https://v8.dev/`. Accedido el 02/04/2022.

[48] Object Management Group. Object management group: Omg. `https://www.omg.org/`. Accedido el 03/05/2022.

[49] OMG (Object Management Group). What is uml? `https://www.uml.org/what-is-uml.htm`. Accedido el 03/05/2022.

[50] Alligator.io & Andy Hattemer. How to make elements stick with css position: sticky. `https://www.digitalocean.com/community/tutorials/css-position-sticky`. Accedido el 03/06/2022.

[51] Matias Hernández. ¿qué es npm? `https://www.freecodecamp.org/espanol/news/que-es-npm/`. Accedido el 02/04/2022.

[52] Uriel Hernández. Qué es typescript. `https://codigofacilito.com/articulos/typescript`. Accedido el 04/05/2022.

[53] Heroku. Heroku dynos. `https://www.heroku.com/dynos`. Accedido el 04/07/2022.

[54] Heroku. What is heroku? `https://www.heroku.com/what`. Accedido el 04/07/2022.

[55] HowToDoInJava. Javascript spread operator. `https://howtodoinjava.com/typescript/spread-operator/`. Accedido el 12/04/2022.

[56] Thorben Janssen. Implementing the repository pattern with jpa and hibernate. `https://thorben-janssen.com/implementing-the-repository-pattern-with-jpa-and-hibernate`. Accedido el 04/07/2022.

[57] Java. ¿qué es la tecnología java y por qué la necesito? `https://www.java.com/es/download/help/whatis_java.html`. Accedido el 03/05/2022.

[58] M. Jones. Json web token (jwt). `https://datatracker.ietf.org/doc/html/rfc7519`. Accedido el 28/05/2022.

[59] JWT. What is json web token? `https://jwt.io/introduction`. Accedido el 28/05/2022.

[60] Arie van Bennekum y 14 autores más. Kent Beck, Mike Beedle. Manifesto for agile software development. `https://agilemanifesto.org/`. Accedido el 01/05/2022.

[61] Angular Material. Badge. `https://material.angular.io/components/badge/overview`. Accedido el 02/06/2022.

[62] Angular Material. Expansion panel. `https://material.angular.io/components/expansion/overview`. Accedido el 16/04/2022.

[63] Angular Material. Form field. hint labels. `https://material.angular.io/components/form-field/overview#hint-labels`. Accedido el 16/06/2022.

[64] Angular Material. List. `https://material.angular.io/components/list/overview`. Accedido el 17/04/2022.

[65] Angular Material. Material design components for agular. `https://material.angular.io/`. Accedido el 01/04/2022.

[66] Angular Material. Paginator. `https://material.angular.io/components/paginator/overview`. Accedido el 01/04/2022.

[67] Angular Material. Progress bar. `https://material.angular.io/components/tooltip/overview`. Accedido el 28/05/2022.

[68] Angular Material. Progress spinner. `https://material.angular.io/components/progress-spinner/overview`. Accedido el 19/04/2022.

[69] Angular Material. Slide toggle. `https://material.angular.io/components/slide-toggle/overview`. Accedido el 22/04/2022.

[70] Angular Material. Snackbar. `https://material.angular.io/components/snack-bar/overview`. Accedido el 20/06/2022.

[71] Angular Material. Table. `https://material.angular.io/components/table/overview`. Accedido el 01/04/2022.

[72] Angular Material. Tooltip. `https://material.angular.io/components/tooltip/overview`. Accedido el 28/05/2022.

[73] Microsoft. Microsoft teams. `https://www.microsoft.com/es-es/microsoft-teams/group-chat-software`. Accedido el 03/05/2022.

[74] Microsoft. Visual studio code. `https://code.visualstudio.com/`. Accedido el 03/05/2022.

[75] Five Flames Mobile. Caso de éxito: Maybein_. `https://fiveflamesmobile.com/maybein-caso-exito/`. Accedido el 11/07/2022.

[76] Svetlin Nakov. Practical cryptography for developers. rsa signatures. `https://cryptobook.nakov.com/digital-signatures/rsa-signatures`. Accedido el 28/05/2022.

[77] Navisite. The premier database-as-a-service for open-source & commercial applications. `https://www.cleardb.com/`. Accedido el 05/07/2022.

[78] Trevor Page. Codercampus. how to avoid duplicate records in hibernate/spring data jpa. `https://www.coderscampus.com/how-to-avoid-duplicate-records-in-hibernate/`. Accedido el 29/05/2022.

[79] Alfonso Eduardo Carrillo Paredes. Git branch y merge de cero a crack. `https://medium.com/@alfonsocarrillo_dev/git-branch-y-merge-de-cero-a-crack-80d653b05f80`. Accedido el 28/05/2022.

[80] Manh Phan. Dao pattern in java. `https://ducmanhphan.github.io/2019-02-15-DAO-pattern-in-java/`. Accedido el 22/06/2022.

[81] QueryThreads. Angular - material: Progressbar custom color? `https://www.querythreads.com/angular-material-progressbar-custom-color/`. Accedido el 28/05/2022.

[82] Spring. Spring data jpa. https://spring.io/projects/spring-data-jpa. Accedido el 03/05/2022.

[83] StackOverflow. Add new marker when i click on the map (openstreetmap, leaflet js). https://stackoverflow.com/questions/43210937/add-new-marker-when-i-click-on-the-map-openstreetmap-leaflet-js. Accedido el 07/03/2022.

[84] StackOverflow. Angular 6 matpaginator does not show mat-option for table and does not set style for next button and back button. https://stackoverflow.com/questions/53156719/angular-6-matpaginator-does-not-show-mat-option-for-table-and-does-not-set-style. Accedido el 01/04/2022.

[85] StackOverflow. Can angularjs auto-update a view if a persistent model (server database) is changed by an external app? https://stackoverflow.com/questions/11276520/can-angularjs-auto-update-a-view-if-a-persistent-model-server-database-is-chan. Accedido el 04/06/2022.

[86] StackOverflow. Customizing increment arrows on input of type number using css. https://stackoverflow.com/questions/45396280/customizing-increment-arrows-on-input-of-type-number-using-css. Accedido el 02/06/2022.

[87] StackOverflow. How to add hours to a date object? https://stackoverflow.com/questions/1050720/how-to-add-hours-to-a-date-object. Accedido el 08/06/2022.

[88] StackOverflow. How to change the text in the label in pagination? https://stackoverflow.com/questions/54103636/how-to-change-the-text-in-the-label-in-pagination. Accedido el 01/04/2022.

[89] StackOverflow. How to generate custom id in jpa. https://stackoverflow.com/questions/47259048/how-to-generate-custom-id-in-jpa. Accedido el 09/06/2022.

[90] StackOverflow. How to get user role from jwt token. https://stackoverflow.com/questions/45458408/how-to-get-user-role-from-jwt-token. Accedido el 13/03/2022.

[91] StackOverflow. Interface type check with typescript. https://stackoverflow.com/questions/14425568/interface-type-check-with-typescript. Accedido el 10/06/2022.

[92] StackOverflow. Multiple canactivate guards all run when first fails. https://stackoverflow.com/questions/40589878/multiple-canactivate-guards-all-run-when-first-fails. Accedido el 04/06/2022.

[93] StackOverflow. Not able to bind second array to angular-dual-listbox. https://stackoverflow.com/questions/63872847/not-able-to-bind-second-array-to-angular-dual-listbox. Accedido el 12/04/2022.

[94] StackOverflow. Websockets: Is it possible to add multiple endpoints using sockjs? https://stackoverflow.com/questions/26211248/websockets-is-it-possible-to-add-multiple-endpoints-using-sockjs. Accedido el 08/06/2022.

[95] Statista. El sector de la restauración en españa - datos estadísticos. https://es.statista.com/temas/6557/la-restauracion-en-espana/. Accedido el 20/06/2022.

[96] TechClubTajamar. Paginación en angular. https://techclub.tajamar.es/paginacion-en-angular/. Accedido el 14/03/2022.

[97] Tokio. El significado de java, el falso gemelo de javascript. https://www.tokioschool.com/noticias/java-significado-que-es-java/. Accedido el 03/05/2022.

[98] Typescript. Typescript is javascript with syntax for types. https://www.typescriptlang.org/. Accedido el 04/05/2022.

[99] USM. Marco de trabajo scrum para el desarrollo ágil de proyectos. https://agenciausm.com/marco-de-trabajo-scrum-para-el-desarrollo-agil-de-proyectos/. Accedido el 01/05/2022.

[100] Desarrollo Web. Sintaxis para las vistas en angular. https://desarrolloweb.com/articulos/sintaxis-vistas-angular2.html. Accedido el 14/05/2022.

[101] Ken Schwaber y Jeff Sutherland. La guía definitiva de scrum: Las reglas del juego. https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf. Accedido el 01/05/2022.

[102] Miguel Ángel Álvarez. Práctica de observables en angular. https://material.angular.io/components/badge/overview. Accedido el 01/06/2022.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

En esta sección se detallan los requisitos necesarios para desplegar y poner en marcha la plataforma.

Requisitos:

- Cuenta verificada en Heroku
- Heroku CLI instalado
- Git instalado
- Enlaces a los repositorios con el código fuente

A.1.1. Despliegue del backend

Para desplegar el backend de la plataforma es necesario clonar el repositorio de GitLab con el código fuente del backend, disponible en el resumen de enlaces adicionales (B), al dispositivo sobre el que se va a realizar el proceso de despliegue en Heroku mediante la ejecución del comando `git clone <URL_del_repositorio>` y posteriormente situarnos en el directorio del repositorio clonado.

Tras iniciar sesión en Heroku, desde una terminal lanzando el comando `heroku login`, será necesario crear un repositorio donde alojar el proyecto en la nube, para ello se ejecutará el comando `heroku create [nombre_app]`, lo cual nos mostrará por pantalla como resultado de la creación del repositorio dos enlaces, el primero es la URL sobre la que se va a desplegar el proyecto y el segundo el enlace al repositorio Git de heroku. Como se puede ver

en el comando descrito, existe la posibilidad de asignar un nombre al repositorio, ya que si no se proporciona, el nombre con el que se genera es aleatorio. Heroku proporciona la posibilidad de renombrar un proyecto con el comando `heroku apps:rename <nombre_nuevo>-a <nombre_app>`. Gracias a esto tendremos creado un repositorio local para Heroku con el nombre que hayamos definido, o el autogenerado en caso de no haberlo escrito o renombrado.

Como utilizamos una base de datos MySQL, será necesario añadir una extensión al proyecto en Heroku, para ello elegiremos la base de datos en la nube "ClearDB" la cual es proporcionada por Heroku. Para insertar esta extensión ejecutaremos el comando `heroku addons:create cleardb:<plan>[-a <nombre_app>]`, con el cual crearemos, para la app seleccionada, una extensión ClearDB con un plan, en nuestro caso usaremos Ignite ya que es el único gratuito y por el momento es suficiente para el despliegue de la plataforma. Con esto tendremos creada la base de datos para el servidor.

Para que posteriormente funcione el despliegue de la plataforma es necesario ejecutar el script sql con la creación de las tablas, que contiene también la inserción de un usuario administrador con correo "root@correo.com" y contraseña "root". Para la conexión con la base de datos ClearDB se pueden obtener los datos ejecutando el comando `heroku config`, el cual devuelve una URL de la base de datos siguiendo el siguiente patrón: `mysql://[username]:[password]@[host]/[database name]?reconnect=true`, para después seleccionar la base de datos proporcionada por Heroku al añadir la extensión.

Por último ejecutaremos el comando `git push heroku main`

Cabe destacar la importancia del archivo "system.properties" situado en el directorio raíz del proyecto, en el cual se establece la versión de Java necesaria para el despliegue.

A.1.2. Despliegue del frontend

Para desplegar el frontend de la plataforma seguiremos un proceso similar al comentado anteriormente para el despliegue del backend, para ello es necesario clonar el repositorio de GitLab con el código fuente del frontend, disponible en el resumen de enlaces adicionales (B), al dispositivo sobre el que se va a realizar el proceso de despliegue en Heroku mediante la ejecución del comando `git clone <URL_del_repositorio>` y posteriormente situarnos en el directorio del repositorio clonado.

Iniciaremos sesión en Heroku y crearemos un repositorio para el proyecto con el código del frontend al igual que se realizó para el backend, se ejecutará el comando `heroku create [nombre_app]`.

Como el frontend va a realizar peticiones sobre el servidor alojado en el apartado anterior, no necesitaremos realizar el proceso de creación e inserción en base de datos.

Finalizamos el proceso de despliegue del frontend en Heroku mediante el comando `git push heroku main`.

A.2. Manual de usuario

Cualquier usuario de la plataforma

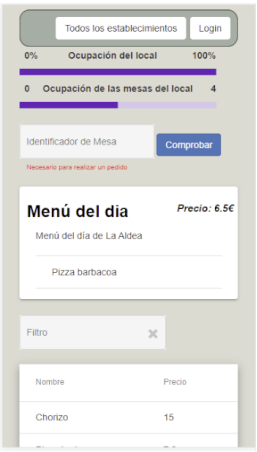
La pantalla principal de la plataforma para cualquier tipo de usuario se puede observar en la Figura A.1, en la cual se muestran los establecimientos validados en la plataforma actualmente, también dispondrán de un botón de login en la parte superior derecha de la pantalla gracias al cual podrán iniciar sesión o registrarse, en caso de no tener una cuenta. En esta misma vista con los establecimientos el usuario podrá observar información sobre el nombre del establecimiento y su nivel de ocupación, mostrado en dos columnas, siendo la primera el nivel de ocupación de personas, mostrado en porcentaje, y una segunda columna con el nivel de ocupación de las mesas, mostrado como una cantidad entera de mesas ocupadas sobre el total de mesas del local.

Figura A.1: Lista de establecimientos para un usuario cualquiera.

Una vez el usuario ha elegido un establecimiento se le mostrará la carta del mismo, tanto los productos que lo componen como, en caso de existir, el menú del establecimiento. Los productos y el menú funcionan como desplegables, mostrando, una vez clickados sobre ellos, la descripción y precio, en caso de los primeros, y los productos que lo componen, que a su vez se puede desplegar su descripción, y el precio del menú. Esta vista se puede observar en la Figura A.2.

También se puede visualizar un campo para ingresar un identificador de mesa, el cual una vez introducido, si es correcto, se desbloqueará al usuario la posibilidad de comenzar a realizar una comanda, apareciendo los campos para la cantidad de productos a seleccionar y el botón del carrito de la compra con la cantidad de productos seleccionados. Esta modificación sobre la pantalla una vez introducido el identificador correcto se puede observar en la Figura A.3.

En esta vista el usuario también puede visualizar el estado del nivel de ocupación del local en tiempo real, mediante dos barras horizontales en la parte superior de la pantalla.



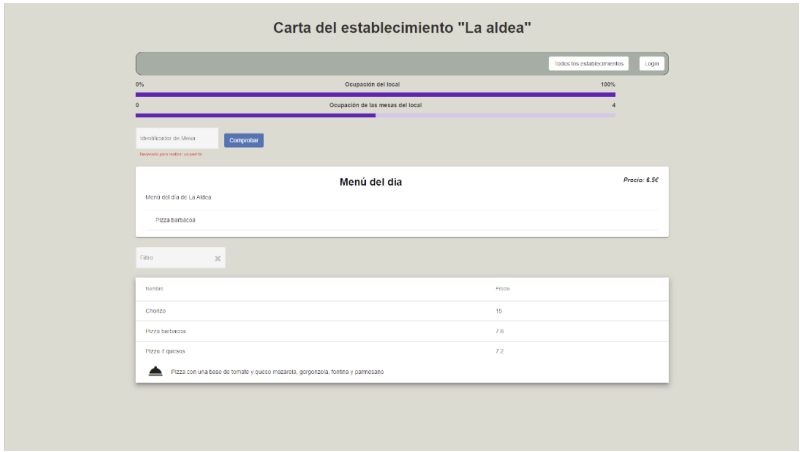


Figura A.2: Lista de productos de una carta para un usuario cualquiera.

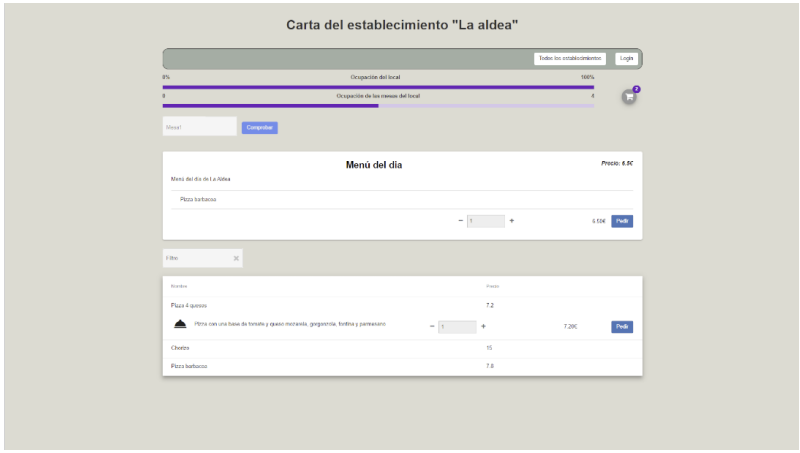


Figura A.3: Lista de productos de una carta para un usuario cualquiera identificador de mesa correcto.

Una vez el usuario ha añadido los productos que quiere pedir en el establecimiento, pulsará el icono del carrito de la compra, desplazándose a la vista con un resumen de la comanda, En esta vista el usuario podrá visualizar un resumen de los productos que ha añadido a la comanda, observando el precio por producto y el total de su pedido. También dispondrá de un botón para borrar por completo una linea del pedido, es decir, toda la cantidad de cierto producto seleccionado, de un botón para volver a la carta en caso de querer añadir otro producto y, por último, un botón para confirmar la comanda y así finalizar el proceso de realizar un pedido. Esta vista se puede observar en la Figura A.4.

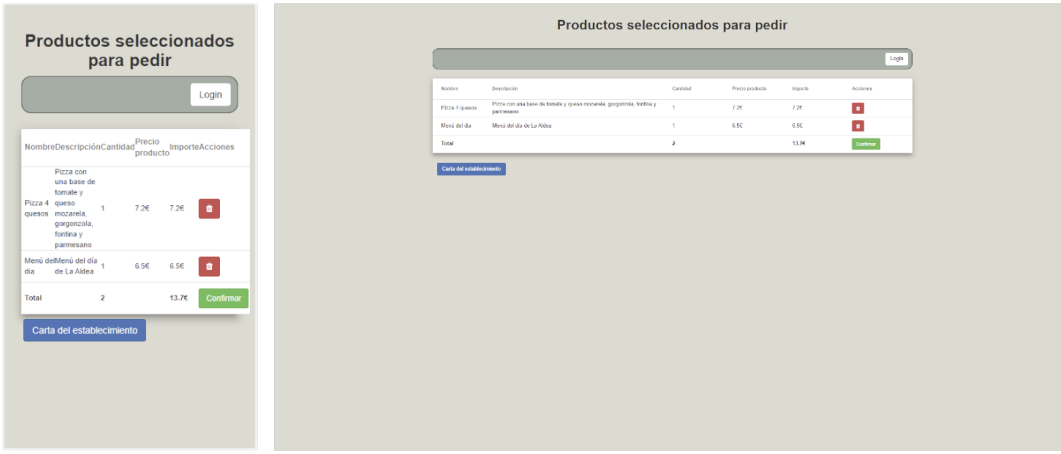


Figura A.4: Lista de productos de una comanda para un usuario cualquiera.

En cualquier momento el usuario podrá iniciar sesión o registrarse, en caso de no tener cuenta en la plataforma, gracias al botón de "Login" que aparece en la esquina superior derecha de las pantallas, tras pulsar este botón el usuario será desplazado a la vista del Login (Figura A.5), desde la cual podrá iniciar sesión en la plataforma. Desde esta, y si el usuario quiere, podrá desplazarse a la vista para registrarse en la plataforma (Figura A.6).

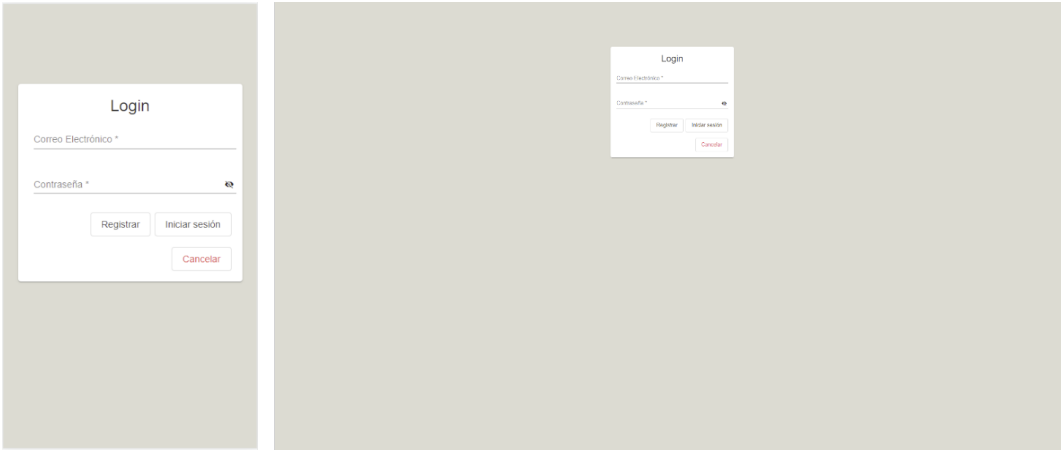
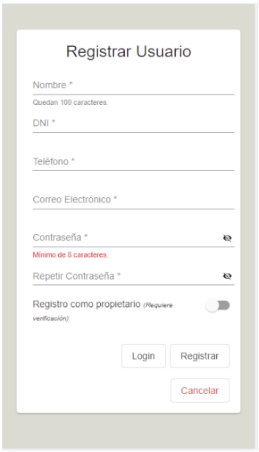


Figura A.5: Login.

Tras finalizar el inicio de sesión o registro el usuario será desplazado a la lista de establecimientos adaptada al rol con el cual ha accedido a la plataforma, Figura A.13 en caso de usuarios con rol de propietario, Figura A.10 para los usuarios con rol de trabajador, Figura A.7 en caso de usuarios con rol de administrador y por último la Figura A.1, en caso de usuarios cliente normales, esta vista es la misma que en el inicio, solo que cambiaría el botón de "Login" por "Logout", ya que no hay funcionalidad creada a mayores para este rol.



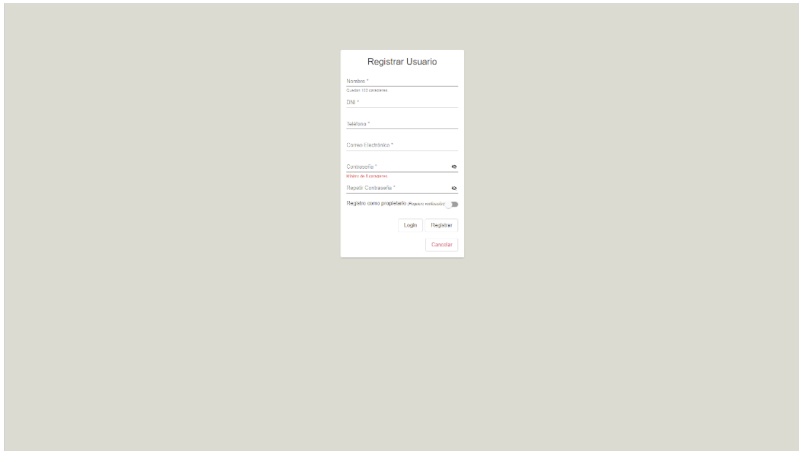


Figura A.6: Registro.

Administrador de la plataforma:

El administrador de la plataforma, tras iniciar sesión en la misma, visualiza la lista de establecimientos validados en la plataforma, al igual que el resto de usuarios tras iniciar sesión, pero este dispone de un botón con opciones desplegables desde el cual puede acceder a las vistas para gestionar solicitudes de validación de establecimientos (Figura A.8) y propietarios (Figura A.9).

Figura A.7: Lista de establecimientos para un administrador.

En caso de querer gestionar las solicitudes de registro de establecimientos, el administrador accede a dicha vista, mostrada en la Figura A.8, donde aparecerán las diferentes solicitudes de registro de establecimientos pendientes. Cuando el administrador seleccione una, se desplegará la información introducida por el propietario en la creación de este y dos botones para aceptar o rechazar la solicitud. En caso de que decida, ya sea rechazar o

aceptar la solicitud, se desplegará una ventana como doble confirmación, para asegurar que la operación no se ha realizado sin querer, proporcionando al administrador la opción de cancelar o confirmar la operación seleccionada. También se le permite realizar una búsqueda por localidad, mostrando únicamente aquellas solicitudes de establecimientos de la localidad seleccionada.

Figura A.8: Lista de establecimientos a validar.

Por otro lado, en caso de querer gestionar las solicitudes de registro de propietario, el administrador accede a dicha vista, mostrada en la Figura A.9, donde aparecerán las diferentes solicitudes de registro de propietarios pendientes. Cuando el administrador seleccione una, se desplegará la información introducida por el propietario en su registro y dos botones para aceptar o rechazar la solicitud. En caso de que decida, ya sea rechazar o aceptar la solicitud, se desplegará una ventana como doble confirmación, para asegurar que la operación no se ha realizado sin querer, proporcionando al administrador la opción de cancelar o confirmar la operación seleccionada.

Figura A.9: Lista de propietarios a validar.

Desde ambas vistas se permite al administrador la navegación de una a la otra.

Trabajador de la plataforma:

Como el resto de usuarios de la plataforma, el trabajador al iniciar sesión visualiza la lista de establecimientos validados (Figura A.10), con la peculiaridad de que observa un botón encima de la lista de establecimientos que lo redirige a la vista con las mesas de su lugar de trabajo (Figura A.11).

Figura A.10: Lista de establecimientos para un trabajador.

Una vez accedido a la lista de mesas de su lugar de trabajo, el trabajador podrá visualizar una lista con la información de las mesas registradas por el propietario del establecimiento en el que trabaja (Figura A.11), pudiendo observar el identificador de la mesa, único dentro de ese establecimiento, la capacidad máxima de cada mesa, el estado de la mesa como libre u ocupado y un botón para consultar las comandas activas asociadas a cada mesa, mostrada en la Figura A.12. El trabajador desde esta vista tiene la capacidad de modificar el estado de la mesa, de libre a ocupado y viceversa.

Figura A.11: Lista de las mesas del lugar de trabajo para un trabajador.

Si el trabajador quiere consultar las comandas activas de una mesa, seleccionará el botón en la lista de mesas para la mesa que quiere consultar, accediendo así a las comandas activas (Figura A.12), dento de la cual podrá ver un listado con las comandas activas, es decir, todas aquellas que no están pagadas. Cada comanda está identificada por un código alfanumérico que indica el identificador del establecimiento, el identificador de la mesa y el número de la comanda dentro de esa mesa, una fecha en la que ha sido solicitada y el estado en la que se encuentra. El trabajador puede seleccionar una comanda, desplegando así un resumen de la comanda y un botón para pasar la comanda al siguiente estado.

Figura A.12: Lista de las comandas activas de una mesa del lugar de trabajo para un trabajador.

Propietario de la plataforma:

Al igual que el resto de usuarios, el propietario, tras iniciar sesión en la plataforma, visualiza la lista de establecimientos validados, con alguna diferencia, ya que si alguno de

estos le pertenece, se mostrarán a la derecha del establecimiento tres botones con diferentes acciones, el botón superior es para la edición de los datos del establecimiento en cuestión (Figura A.15), el segundo para acceder a la lista de mesas del establecimiento (Figura A.19) y el último para acceder a la lista de trabajadores del local (Figura A.21). También observará un botón para la creación de un establecimiento nuevo, otro botón para acceder a la lista de sus productos registrados (Figura A.23) y un último botón para consultar solamente sus establecimientos (Figura A.14), en cualquier estado.

Figura A.13: Lista de establecimientos para un propietario.

En caso de observar la lista con sus establecimientos, el usuario con rol de propietario podrá contemplar tres diferentes colores, dependiendo del estado de la solicitud para ese establecimiento, siendo el color blanco para aquellas solicitudes validadas, el color amarillo para las pendientes y el rojo para las rechazadas.

Figura A.14: Lista de establecimientos propios para un propietario.

En caso de que el usuario decida crear un establecimiento, se le mostrará un formulario con los diferentes campos a rellenar con la información del local y un mapa desde el cual podrá seleccionar las coordenadas del establecimiento de forma más visual. En caso de que el propietario decida editar la información de un establecimiento, se le mostrará la misma vista que en la creación (Figura A.15), pero con los campos rellenados con la información actual del local. En ambos casos se le proporcionará la posibilidad de cancelar y de, en caso de que todos los campos sean correctos y estén rellenados, confirmar la operación.

Figura A.15: Vista para la creación/edición uno de sus establecimientos para un propietario.

Si el propietario accede a la vista para ver la carta de uno de sus establecimientos, mostrada en la Figura A.16, observará que es muy similar a la vista para un usuario cualquiera, con la peculiaridad de que dispone de un botón para editar la carta y otro para editar el menú.

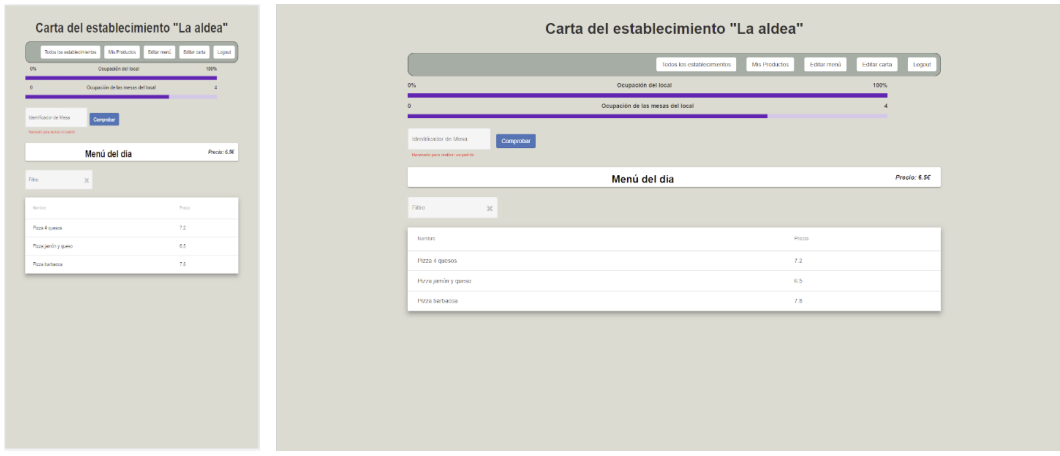


Figura A.16: Lista de productos de una carta para un propietario.

En caso de que el propietario decida editar la carta del establecimiento, accederá a la vista para ello (Figura A.17), donde observará dos listas de productos, viendo en la lista izquierda aquellos de sus productos que no están agregados en la carta del establecimiento seleccionado y en la lista derecha aquellos productos incluidos. Dispondrá de un botón para cancelar la operación y otro para confirmarla, en caso de pulsar este segundo botón, mostrará una ventana emergente como doble confirmación, donde podrá cancelar o confirmar la operación de forma definitiva. En caso de que la carta no tenga productos o sea idéntica a la carta original que ya existía para el establecimiento, la ventana emergente avisará al usuario de la situación.

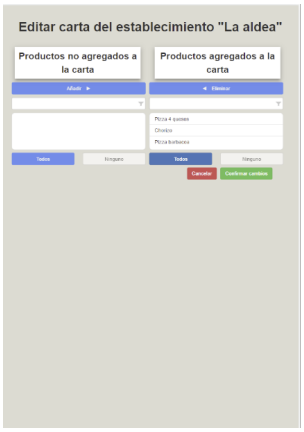


Figura A.17: Vista para la edición de la carta de uno de sus establecimientos para un propietario.

En caso de que el propietario decida editar el menú de la carta del establecimiento, accederá a la vista para ello (Figura A.18), donde observará tres campos en la parte superior de la pantalla, para introducir el nombre, precio y descripción del menú y dos listas de productos, viendo en la lista izquierda aquellos de sus productos que no están agregados al menú del establecimiento seleccionado y en la lista derecha aquellos productos incluidos. Dispondrá de un botón para cancelar la operación y otro para confirmarla, en caso de pulsar este segundo botón, mostrará una ventana emergente como doble confirmación, donde podrá cancelar o confirmar la operación de forma definitiva. En caso de que el menú no tenga productos o sea idéntica al menú original que ya existía para el establecimiento, la ventana emergente avisará al usuario de la situación.

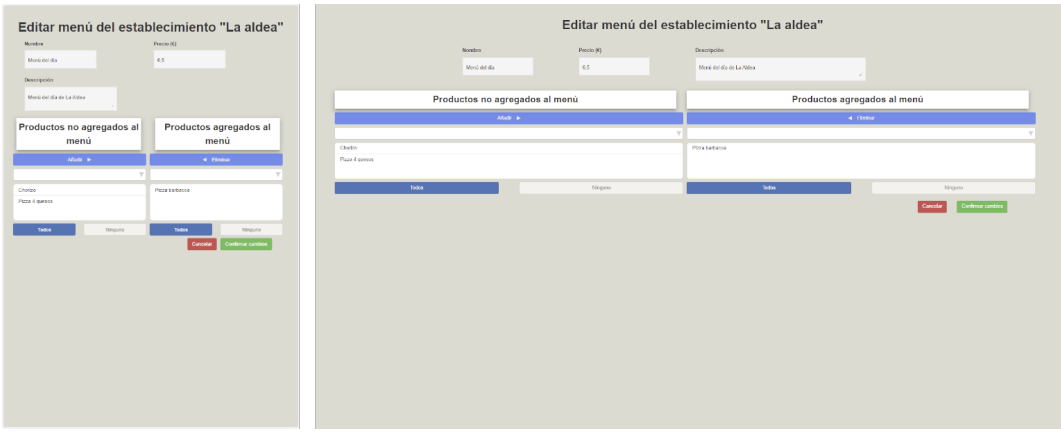


Figura A.18: Vista para la edición del menú de la carta de uno de sus establecimientos para un propietario.

Si el propietario decide consultar la lista de mesas para uno de sus establecimientos, podrá visualizar una lista con la información de las mesas registradas para el establecimiento seleccionado, mostrada en la Figura A.19, pudiendo observar el identificador de la mesa, único dentro de ese establecimiento, la capacidad máxima de cada mesa, el estado de la mesa como activada o desactivada y dos botones, uno para editar y otro para eliminar, asociados a cada mesa. El propietario desde esta vista tiene la capacidad de modificar el estado de la mesa, de activada a desactivada y viceversa, también dispone de información sobre el aforo máximo actual del local, el cual debe ser siempre igual o menor a la capacidad máxima total de todas las mesas activas del establecimiento y un campo para poder editar este valor, el cual, en caso de intentar introducir un valor erróneo mostrará un mensaje de error.

Figura A.19: Lista de las mesas de uno de sus establecimientos para un propietario.

A mayores de lo comentado, el propietario dispondrá de un botón para crear mesas, en caso de que el aforo máximo actual del establecimiento sea mayor que la capacidad máxima total de todas las mesas. En caso de realizar un cambio en el estado activado de una mesa, tendrá repercusión sobre la capacidad máxima, alterando los valores y comprobando si el aforo máximo actual es necesario cambiarlo.

Si el propietario decide crear una mesa accederá a la pantalla para ello, mostrada en la Figura A.20, donde aparecerán dos campos para introducir la información sobre la mesa, uno para el identificador de esta, el cual tiene que ser único en el establecimiento, mostrando un mensaje de error si no es así, y la capacidad máxima de la mesa. La vista utilizada para la edición de la información de la mesa es la misma, con la diferencia de que en caso de querer editar una mesa, los campos tendrán la información de la mesa seleccionada y a mayores aparecerá un botón para poder activar o desactivar la mesa.

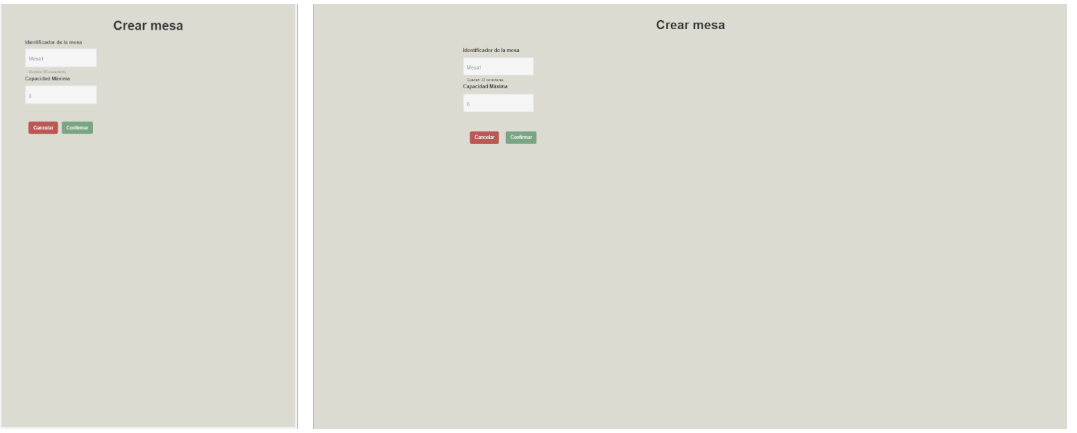


Figura A.20: Vista para la creación/edición de una mesa para un propietario.

Cada establecimiento cuenta con una lista de trabajadores (Figura A.21), a la cual puede acceder el propietario de este desde la lista de establecimientos, pulsando el tercer botón que aparece en la parte derecha. Con la lista de trabajadores presente, el propietario podrá filtrar la lista introduciendo una cadena en el campo destinado a ello.

En caso de que existan más de cinco trabajadores contratados en el establecimiento, podrá acceder a los datos del resto utilizando las flechas de paginación de la tabla para navegar por las páginas, o cambiar la cantidad de trabajadores que aparecen en cada una de ellas. También dispondrá de un botón en la parte superior de la lista para agregar un nuevo trabajador al local, en este caso será redirigido a la vista utilizada para realizar esta operación, mostrada en la Figura A.22.

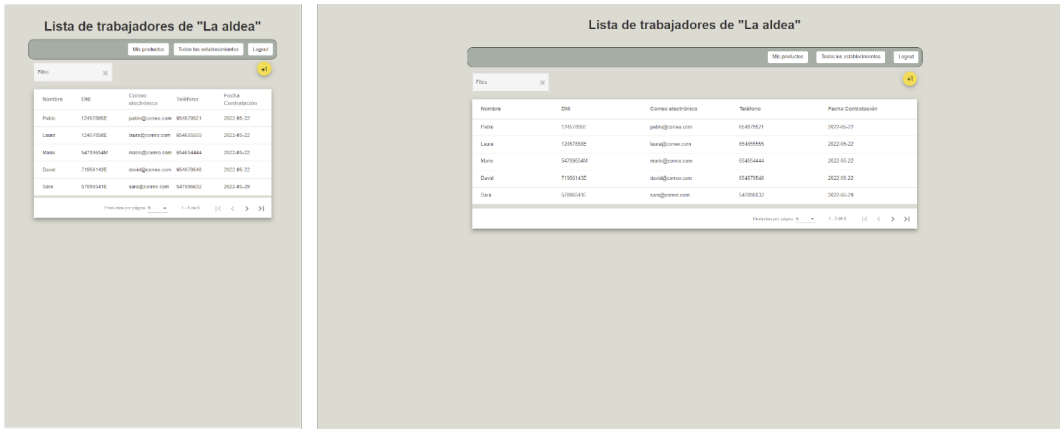


Figura A.21: Lista de los trabajadores de uno de sus establecimientos para un propietario.

Si el propietario desea añadir un nuevo trabajador al establecimiento, tendrá que rellenar los campos con la información del usuario, mostrados en la Figura A.22. Como un añadido se le ha proporcionado la posibilidad de que la contraseña sea generada por el sistema, la cual estará compuesta por una cadena alfanumérica con al menos un número, una minúscula y una mayúscula, y una longitud aleatoria entre 8 y 12 caracteres.

Figura A.22: Vista para la creación de un trabajador en un establecimiento para un propietario.

El propietario tendrá la capacidad de cancelar y, en caso de que todos los campos estén rellenados de forma correcta, confirmar la operación.

El propietario dispondrá de una vista donde podrá observar una lista con los productos que ha registrado en la plataforma, estos productos son los que posteriormente podrán ser agregados a la carta y el menú de sus establecimientos. En la vista con sus productos,

mostrada en la Figura A.23, se puede observar un listado de los productos agregados por el propietario, viendo información como el nombre, descripción y precio. En caso de que existan más de cinco productos registrados, podrá acceder a los datos del resto utilizando las flechas de paginación de la tabla para navegar por las páginas, o cambiar la cantidad de productos que aparecen en cada una de ellas.

El propietario dispondrá de un filtro, con el cual poder realizar las operaciones rápidamente sobre el producto filtrado y de un botón superior con el cual podrá agregar un nuevo producto.

También existen dos botones, asociados a cada producto por separado, desde los cuales podrá tanto editar la información del producto como eliminarlo. En caso de que el propietario desee editar uno de sus productos, accederá, tras presionar el botón, a la vista de la Figura A.24, la misma que se utiliza en caso de querer añadir un nuevo producto.

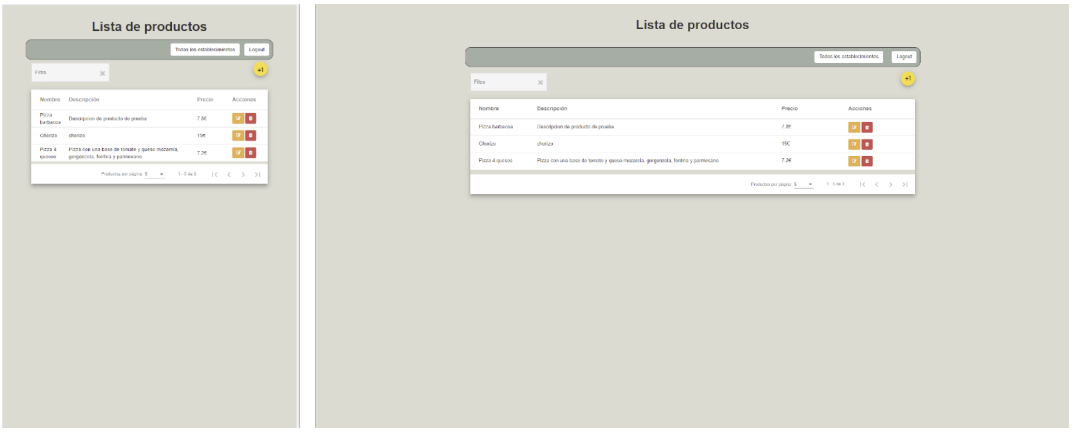


Figura A.23: Lista de productos para un propietario.

En caso de que el usuario decida crear un producto nuevo, se le mostrará un formulario con los diferentes campos a rellenar con la información del producto a crear. En caso de que el propietario decida editar la información de uno de los productos ya registrados, se le mostrará la misma vista que en la creación (Figura A.15), pero con los campos rellenados con la información actual del producto seleccionado. En ambos casos se le proporcionará la posibilidad de cancelar y de, en caso de que todos los campos sean correctos y estén rellenados, confirmar la operación.

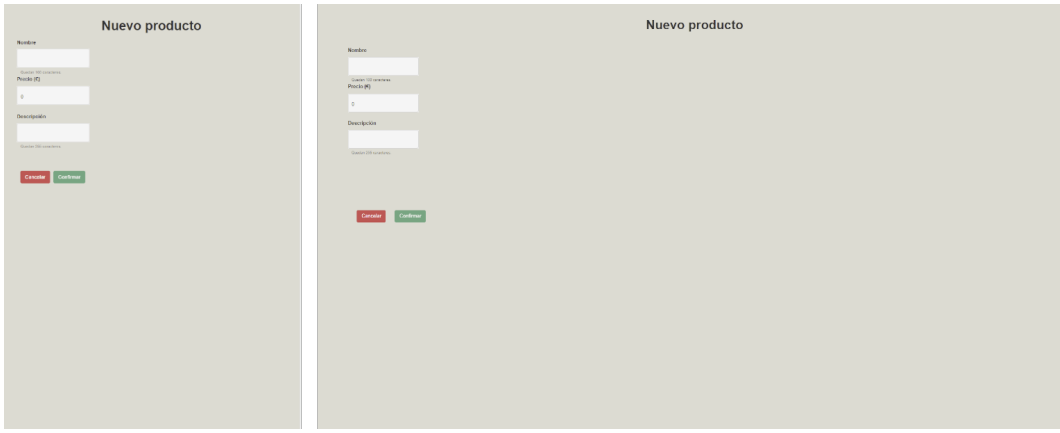


Figura A.24: Vista para la creación/edición uno de sus productos para un propietario.

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio utilizado para el desarrollo del proyecto:
  `https://gitlab.inf.uva.es/davcuri/tfg-davidcuriesesfernandez`.
- Repositorios del código fuente para el despliegue:
  - **Repositorio del código fuente para el despliegue del backend**:
    `https://gitlab.inf.uva.es/davcuri/tfg-davidcurieses-backend`
  - **Repositorio del código fuente para el despliegue del frontend**:
    `https://gitlab.inf.uva.es/davcuri/tfg-davidcurieses-frontend`
- Url del sistema desplegado:
  `https://tfg-davidcurieses-frontend.herokuapp.com/`