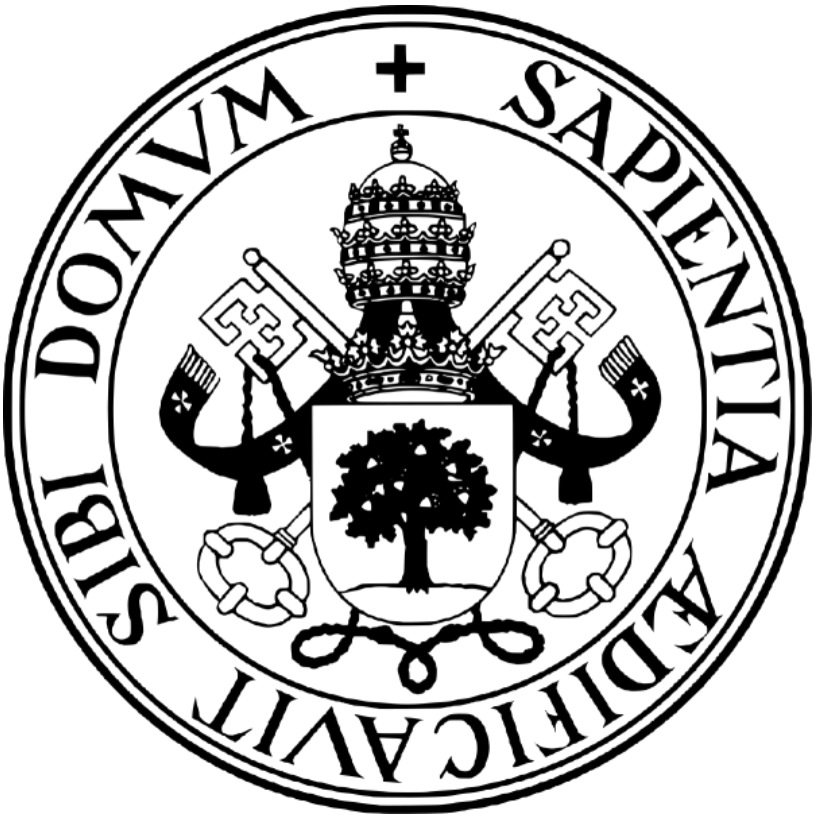


UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Restaurante Casares: planificación de reservas

Autor: Alberto Herrero Toledano

Tutor: Fernando Díaz Gómez

RESUMEN

Este trabajo de fin de grado tiene como objetivo principal una **aplicación web** para el restaurante Casares, con el objetivo de mejorar la **gestión y asignación de reservas** en los diferentes **espacios del establecimiento**. El problema actual de este restaurante radica en la necesidad de mover y cuadrar las mesas para proporcionar el servicio a las reservas, lo cual requiere la intervención y tiempo de un trabajador para organizarlas en el plano, algo que en muchas ocasiones no resulta trivial.

La aplicación pretende ser una herramienta **intuitiva** y de fácil uso, considerando usuarios de diferentes edades. Además de gestionar las reservas y los espacios, proporciona al usuario toda la información relevante sobre el turno de comida correspondiente.

Para lograr esto, la herramienta digital permite almacenar reservas y generar una lista de **posiciones disponibles**. En caso de no haber posiciones libres, realizará una **búsqueda de mesas compatibles** para combinarlas y poder atender la reserva, ofreciendo una lista de todas las **combinaciones de mesas posibles**.

Palabras clave:

Aplicación web, gestión de reservas, asignación de espacios, intuitiva, búsqueda de mesas compatibles, combinación de mesas.

ABSTRACT

The main objective of this final degree project is to develop a **web application** for the Casares restaurant, with the aim of improving the **management and allocation of reservations** in the different **spaces of the establishment**. The current problem of this restaurant lies in the need to move and square the tables to provide service to the reservations, which requires the intervention and time of a worker to organise them on the floor plan, something that on many occasions is not trivial.

The application is intended to be an **intuitive** and user-friendly tool, considering users of different ages. In addition to managing bookings and spaces, it provides the user with all relevant information about the corresponding meal shift.

To achieve this, the digital tool allows storing reservations and generates a list of available positions. In case there are no free positions, it will perform a **search for compatible tables** in order to combine them and be able to attend the reservation, offering a list of all **possible table combinations**.

Key words:

Web application, reservation management, space allocation, intuitive, search for compatible tables, table combination.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPITULO 1. INTRODUCCIÓN.

MOTIVACIÓN

El proyecto nace con el fin de eliminar un problema del restaurante Casares en lo relativo a la gestión del espacio físico que ocuparán las reservas.

El restaurante Casares es un establecimiento hostelero de renombre de la ciudad de Segovia, el cual tiene gran cantidad de reservas. Los fines de semana tiene tanta demanda, que la mayoría de las reservas ocupan el 80% de los servicios que se realizan.

Para gestionar tal demanda, el restaurante cuenta con dos turnos de comidas y uno de cenas, repartidos en tres comedores y una terraza. Entre ellos suman más de 70 mesas, con las cuales pueden prestar sus servicios a más de 250 comensales.

Las mesas tienen una colocación por defecto en cada uno de los comedores, pero el problema radica en el momento en el que hay mesas disponibles, pero no lo suficientemente grandes para abordar el número de comensales de la reserva. En este caso, antes de que comience el turno de la comida correspondiente, un trabajador deberá dibujar sobre el papel un plano de los comedores y buscar una distribución que les permita colocar las mesas para poder atender todas las reservas, y no siempre logran respetar las zonas de paso necesarias para la comodidad de los camareros durante el servicio.

A esto debe sumarse que uno de los comedores cuenta con paredes móviles, las cuales permiten separar dicho comedor en diferentes salas privadas.

Por todo esto, surge la necesidad de automatizar el proceso de asignación de posiciones y reservas con el fin de ahorrar tiempo y hacer el mínimo número de cambios de mesas posibles, y gestionar las salas privadas.

OBJETIVOS

El objetivo principal de este proyecto es desarrollar una aplicación WEB que facilite y agilice a los usuarios la gestión de reservas y posiciones dentro de los diferentes comedores, y consultar la información sobre las reservas y la distribución espacial de las mesas.

Una vez planteado el objetivo principal de la aplicación, esta también cuenta con una serie de objetivos derivados que complementan el objetivo principal y proporcionan una riqueza y una funcionalidad muy interesante para el usuario final. Estos objetivos son:

ID Objetivo	Nombre del objetivo
OBJ-01	Implementar un sistema que almacene y represente la información sobre los turnos, reservas, y distribuciones de los comedores.
OBJ-02	Permitir la creación, consulta y modificación de datos de reservas.
OBJ-03	Implementar un sistema de colocación de mesas.
OBJ-04	Implementar un sistema de búsqueda y colocación de combinaciones de mesas.

Tabla 1 - Objetivos del proyecto

Para garantizar la consecución de los objetivos, cada uno de ellos va asociado a sus criterios de aceptación, que determinarán si el objetivo se ha cumplido adecuadamente.

ID Criterio de aceptación	ID Objetivo asociado	OBJ-01
	Descripción Criterio de Aceptación	
CA-01	La aplicación almacena todos los datos sobre las reservas	
CA-02	La aplicación almacena toda la información de los turnos	
CA-03	La aplicación almacena toda la información sobre las distribuciones de los comedores.	

Tabla 2 - Criterios de aceptación para OBJ-01

ID Criterio de aceptación	ID Objetivo asociado	OBJ-02
	Descripción Criterio de Aceptación	
CA-04	La aplicación permite crear una reserva e introducir los datos sobre ésta.	
CA-05	La aplicación permite modificar los datos de las reservas	
CA-06	La aplicación dispone de una lista de reservas con un formato compacto y la información más relevante	
CA-07	La aplicación dispone de una forma de visualizar todos los datos de una reserva	

Tabla 3 - Criterios de aceptación para OBJ-02

ID Criterio de aceptación	ID Objetivo asociado	OBJ-03
	Descripción Criterio de Aceptación	
CA-08	La aplicación ofrece una lista con las mesas disponibles en cada uno de los comedores	
CA-09	La aplicación guarda los datos de la asignación mesa-reserva	

Tabla 4 - Criterios de aceptación para OBJ-03

ID Criterio de aceptación	ID Objetivo asociado	OBJ-04
	Descripción Criterio de Aceptación	
CA-10	La aplicación ofrece una lista con las mesas disponibles en cada uno de los comedores	
CA-11	La aplicación guarda los datos de la asignación mesa-reserva	

Tabla 5 - Criterios de aceptación para OBJ-04

ALCANCE

El proyecto está pensado para que puedan utilizarlo todos los trabajadores del restaurante Casares, desde los camareros, hasta el dueño y el administrativo.

Es una propuesta, que en el caso de dar los resultados esperados y tener una buena aceptación por el cliente final de la aplicación podría convertirse en la base para el desarrollo de una aplicación de gestión interna de reservas a nivel provincial, abarcando así grandes restaurantes con problemas de gestión similares.

Se desarrollará una aplicación de gestión de turnos y reservas para que los trabajadores del establecimiento puedan realizar:

- **Gestión de reservas:** incluye las características y funcionalidades asociadas a la creación, modificación y visualización de las reservas.
- **Gestión de turnos:** incluye las características y funcionalidades asociadas a la creación, modificación y visualización de los turnos.
- **Gestión de distribución de comedores:** incluye las características y funcionalidades asociadas a la creación , modificación y visualización de las distribuciones espaciales de las mesas en los comedores.

A continuación, se presenta el árbol de características de la aplicación:

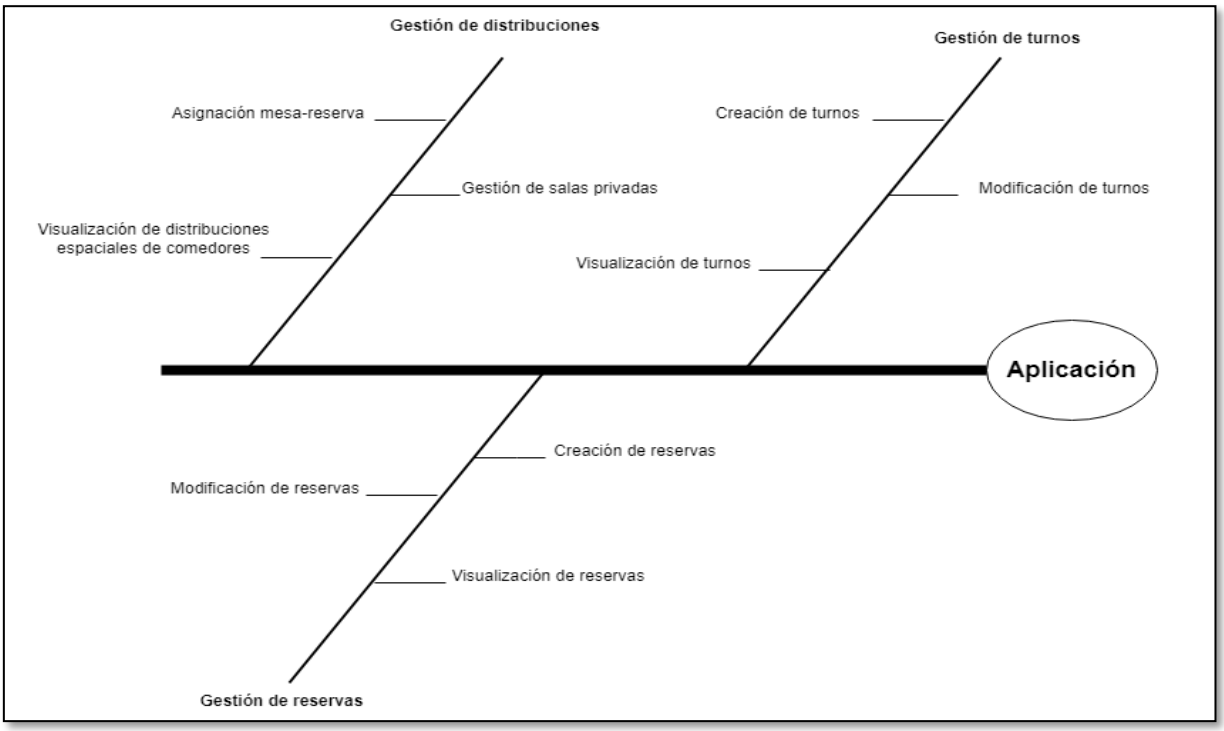


Figura 1 - Árbol de características de la aplicación

Todas las imágenes que presentan esquemas han sido elaboradas por el autor de este TFG.

ENTORNO TECNOLÓGICO

Para el desarrollo software del proyecto se han utilizado las siguientes herramientas:

- **Visual Studio Code**

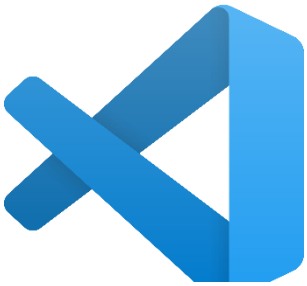


Figura 2 - Logo de Visual Studio Code

Visual Studio Code (*VS Code*) es un editor de código fuente ampliamente utilizado y altamente valorado desarrollado por Microsoft. Combina la potencia de un entorno de desarrollo integrado completo con la simplicidad y ligereza de un editor de texto. Con su interfaz intuitiva y opciones de personalización extensas, los desarrolladores pueden adaptar su entorno de programación a sus preferencias y aumentar su productividad.

VS Code admite una amplia gama de lenguajes de programación y ofrece características como resaltado de sintaxis, autocompletado inteligente e integración *Git* incorporada. También cuenta con una amplia colección de extensiones, que permiten a los desarrolladores agregar funcionalidades adicionales y mejorar su flujo de trabajo. Ya sea que seas un principiante o un desarrollador experimentado, Visual Studio Code proporciona una plataforma sólida y eficiente para escribir, editar y depurar código, convirtiéndolo en una elección habitual para muchos programadores en todo el mundo.

- **XAMPP**

Figura 3 - Logo de XAMPP

XAMPP es un paquete de software gratuito y de código abierto que proporciona un entorno de servidor web completo para el desarrollo y la implementación de aplicaciones web. Su nombre es un acrónimo que representa las diferentes tecnologías que incluye: Apache, MySQL, PHP y Perl. Está diseñado para ser multiplataforma, lo que significa que puede ser utilizado en sistemas operativos como Windows, macOS y Linux.

XAMPP simplifica el proceso de configuración y administración de un servidor web local, ya que viene preconfigurado con todos los componentes necesarios para ejecutar aplicaciones web dinámicas. Incluye el servidor web Apache, que es ampliamente utilizado en la industria, el sistema de gestión de bases de datos MySQL para almacenar y recuperar datos, el lenguaje de programación PHP para desarrollar aplicaciones web dinámicas, y el lenguaje de *script* Perl para automatizar tareas. Además, también incluye otras herramientas y componentes como *phpMyAdmin* para administrar bases de datos, *FileZilla* para transferencia de archivos y *Mercury Mail* para el envío de correos electrónicos. En resumen, XAMPP es una solución todo en uno que facilita la creación y prueba de aplicaciones web en un entorno local antes de ser desplegadas en un servidor de producción.

- **GitHub**

GitHub es una plataforma de desarrollo colaborativo de software que se basa en el sistema de control de versiones *Git*. Permite a los desarrolladores trabajar en proyectos de forma conjunta, compartir y colaborar en el código fuente. *GitHub* proporciona un repositorio centralizado donde los desarrolladores pueden almacenar, gestionar y controlar las diferentes versiones de su código, lo que facilita el seguimiento de los cambios realizados y la colaboración entre equipos.

Figura 4 - Logo de Git Hub

Además de ser una plataforma de alojamiento de repositorios, *GitHub* también ofrece diversas características y herramientas que fomentan la colaboración y la comunidad en torno al desarrollo de software. Por ejemplo, permite la creación de solicitudes de extracción (*pull requests*) para revisar y discutir los cambios propuestos, la posibilidad de generar problemas (*issues*) para realizar seguimiento de errores o tareas pendientes, y la integración con herramientas de integración continua para automatizar pruebas y despliegue. *GitHub* se ha convertido en una de las plataformas más populares y utilizadas por la comunidad de desarrollo de software, proporcionando un entorno robusto y escalable para la colaboración y el control de versiones.

CAPITULO 2. ESTADO DEL ARTE

Antes de desarrollar la aplicación web es necesario conocer y analizar las propuestas similares de herramientas ya existentes en el mercado actual, las cuales sirven para inspeccionar y marcar en cierto modo el dominio de nuestro proyecto, es decir, analizar los aspectos relevantes y características esenciales que no pueden faltar en nuestro proyecto, junto con aspectos nuevos, los cuales el resto de propuestas no son capaces de gestionar.

Existen algunos programas de gestión de mesas para restaurantes tales como *Resmio* o *Restoo*. Estos incluyen múltiples funcionales en cuanto a la gestión de reservas y mesas pero no cuentan con la automatización en la combinación de mesas y por supuesto, tampoco cuentan con la capacidad de unir o dividir a demanda las salas privadas que hacen tan dinámico y versátil al restaurante Casares.

Además incluyen características en las cuales el restaurante no está interesado, por lo cual estaría pagando por un software que no va a aprovechar.

- ***Resmio***

Resmio es una aplicación innovadora y eficiente diseñada para facilitar la gestión de restaurantes y negocios de hostelería. Con su interfaz intuitiva y funciones versátiles, *Resmio* permite a los propietarios y gerentes organizar reservas, gestionar mesas y optimizar la capacidad del local. Además, ofrece herramientas para la creación y publicación de menús, así como la gestión de pedidos y pagos online. *Resmio* también brinda la posibilidad de enviar notificaciones y recordatorios a los clientes, mejorando la comunicación y reduciendo los olvidos o cancelaciones. (Resmio, 2022)

Figura 5 – Logo de Resmio

- ***Restoo***

Restoo es una aplicación móvil diseñada para facilitar la experiencia de los usuarios al buscar, reservar y disfrutar de restaurantes. Con *Restoo*, los usuarios pueden explorar una amplia variedad de restaurantes cercanos, filtrar por tipo de cocina, ubicación y precios, leer reseñas y ver calificaciones de otros comensales. Además, la aplicación permite realizar reservas de forma rápida y sencilla, lo que evita esperas innecesarias. También ofrece la posibilidad de realizar pedidos para llevar o servicio a domicilio en algunos restaurantes. En resumen, *Restoo* brinda a los usuarios la comodidad de encontrar y disfrutar de una experiencia gastronómica placentera, ofreciendo opciones, reseñas y servicios convenientes, todo en una sola aplicación. (Restoo, 2022)

Figura 6 - Logo de Restoo

Analizando las características de los sistemas anteriores, se observa que la característica de la gestión de las salas privadas no se encuentra en ninguno de estos. Esto, junto a la combinación automática de mesas, hace que nuestra propuesta sea más atractiva que las mencionadas en esta sección.

Característica	Restoo	Resmio	Aplicación gestión de Casares
Gestiona reservas	✓	✓	✓
Gestiona turnos	✓	✓	✓
Gestiona comedores	✓	✓	✓
Maximiza espacio	✓	✓	✓
Gestiona división de comedor en salas privadas	✗	✗	✓
Combina mesas automáticamente	✗	✗	✓
Integración de canales de reservas	✓	✓	✗

Tabla 3 - Comparativa de sistemas de gestión de reservas

CAPITULO 3. PANIFICACIÓN Y PRESUPUESTOS

MODELO DE DESARROLLO

Después de examinar todas las técnicas y enfoques actuales en el desarrollo de proyectos de software, se ha decidido utilizar el enfoque iterativo-incremental para este proyecto específico. Este enfoque combina las ventajas del modelo en cascada y el modelo basado en la creación de prototipos.

Este enfoque evolutivo implica la construcción de la herramienta en bloques de trabajo planificados en varias iteraciones o etapas. Durante cada iteración, se mejora y amplía la funcionalidad y características del producto.

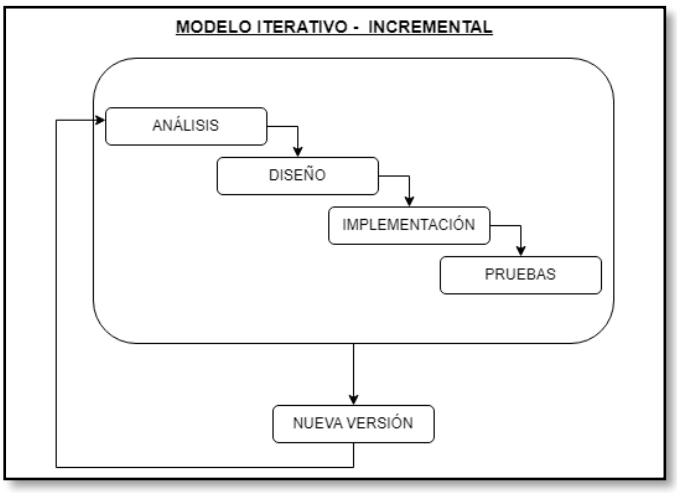


Figura 7 - Modelo de desarrollo iterativo-incremental

Cada una de esas fases o ciclos da lugar a un programa completamente funcional sobre el cual se basa la nueva fase, y es precisamente por este motivo que se necesita un desarrollo completo del requisito o característica que se ha propuesto para esa etapa, incluyendo una serie de verificaciones y una documentación detallada que permita a los programadores tener una base sólida para iniciar el desarrollo de nuevas necesidades y características que formarán parte de las futuras fases. Esto genera como resultado una serie de ventajas en el desarrollo, aunque también puede tener ciertos inconvenientes. A continuación se presentan, de manera simplificada, estas consecuencias:

Ventajas:

- Permite tener un programa completamente funcional al finalizar cada fase.
- Reduce los riesgos en el desarrollo del software, debido a la realización de verificaciones para cada una de las fases.
- Simplifica el desarrollo del proyecto, debido a la división en bloques de trabajo (etapas).
- Proporciona un alto grado de flexibilidad al incorporar nuevas necesidades y funciones al desarrollo.

- Mejora el aprendizaje y la experiencia, ya que con cada fase, el equipo de trabajo aumenta su conocimiento del software.

Inconvenientes:

- Requiere un cliente comprometido durante todo el desarrollo del proyecto.
- Las distintas fases que conforman cada ciclo son inflexibles, hasta que no se completa una de ellas, no comienza la siguiente.

Para el desarrollo se han realizado 12 iteraciones de este ciclo:

1. Etapa de análisis global de los objetivos y requisitos del proyecto.
2. Creación de la estructura de la página principal.
3. Creación y población de la base de datos.
4. Visualización de los datos en la página principal.
5. Funcionalidad básica de la página principal.
6. Automatización de creación de datos por fecha.
7. Gestión de comedores de los turnos.
8. Creación de reservas y asignación de mesa (sin combinaciones)
9. Gestión de salas privadas para la creación de reservas.
10. Búsqueda de combinaciones de mesas y asignación a nuevas reservas.
11. Página de acceso y gestión de sesiones.
12. Añadir estilos definitivos a la aplicación.

PLANIFICACIÓN INICIAL DE PROYECTO

Para la planificación de este proyecto se han tenido en consideración una serie de aspectos que repercuten de forma directa con la duración y estimación temporal del mismo como son:

- La planificación del proyecto comienza el 1 de febrero de 2022.
- También se tendrán en cuenta vacaciones en el mes de agosto y en navidades del año 2022.
- Por motivos laborales e considera que se podrá trabajar en el proyecto de 3 a 5 días laborables con un total de 10 a 15 horas semanales.
- En las primeras fases del proyecto, se prevé que las labores de análisis y diseño tengan una duración más extensa. Esto se debe a la metodología empleada en el desarrollo del proyecto, que exige un análisis detallado de los requisitos, objetivos y características esenciales de la herramienta para crear los primeros productos que satisfagan las necesidades básicas del cliente. Además, en estas etapas iniciales, el equipo de trabajo se familiariza con el proyecto y las herramientas utilizadas.
- En las fases intermedias y finales, las tareas de implementación, integración y pruebas adquieren mayor relevancia. Esto es debido al aumento en la complejidad del desarrollo de nuevas funcionalidades, lo que resulta en productos más completos cuya integración con otros componentes de la herramienta también se vuelve más complicada. Estos

productos, a su vez, necesitan pruebas más exhaustivas para asegurarse de que satisfacen todos los requerimientos propuestos por el cliente.

Etapa	Descripción	Inicio	Fin	Nº días
Etapa 1	Etapa de análisis	01/02/2022	21/03/2022	35
Etapa 2	Estructura de la página	22/03/2022	13/04/2022	15
Etapa 3	Base de datos	14/04/2022	19/05/2022	25
Etapa 4	Visualización de datos	20/05/2022	17/06/2022	20
Etapa 5	Funcionalidad básica de la página	18/06/2022	06/07/2022	20
Etapa 6	Creación datos automática por fecha	07/07/2022	04/09/2022	20
Etapa 7	Gestión de comedores	05/09/2022	02/10/2022	20
Etapa 8	Gestión de reservas y asignación mesas simples	03/10/2022	15/11/2022	30
Etapa 9	Gestión de salas privadas	16/11/2022	06/12/2022	25
Etapa 10	Búsqueda y asignación de combinaciones de mesas	07/12/2022	07/02/2023	30
Etapa 11	Página de acceso y gestión de sesiones	08/02/2023	22/02/2023	10
Etapa 12	Añadir estilos definitivos a la aplicación	23/02/2023	15/03/2023	15

Tabla 4 - Etapas de planificación del proyecto

Teniendo esto en cuenta, el proyecto debería realizarse en 12 meses aproximadamente.

COSTES INICIALES DEL PROYECTO

COSTES DE PERSONAL

Estos gastos están relacionados con todos los recursos humanos que participan en las diferentes etapas de planificación inicial establecidas en el diagrama de Gantt previamente expuesto.

Para este proyecto, solo hay un recurso humano que participa activamente en todas las fases del proyecto: el autor. Este autor desempeña diferentes roles según la fase y la tarea que esté realizando para proporcionar una estimación más precisa y realista de este tipo de gastos. Los roles asumidos en el proyecto son los siguientes:

- **Director de Proyecto**: se encarga de gestionar tanto al equipo de trabajo y sus tareas desempeñadas como el control del cumplimiento de objetivos y características del proyecto. También es el encargado de realizar las tareas de revisión y redacción de la documentación.
- **Analista**: se encarga de realizar las tareas de análisis y exploración de requisitos que debe cumplir el proyecto. También se encarga de gestionar la arquitectura del sistema.
- **Diseñador de UX/ IU**: se encarga de analizar y diseñar todos los aspectos relacionados con la experiencia e interfaces de usuario.
- **Desarrollador**: se encarga de llevar a cabo las labores de implementación del proyecto.
- **Tester**: se encarga de comprobar que se cumplen todos los objetivos y requisitos establecidos para el producto (tanto el producto completo como los distintos productos que se van generando a lo largo de las fases del proyecto).

COSTES HARDWARE

Son de los gastos asociados al uso de medios materiales durante la ejecución del proyecto. En la tabla siguiente se muestra el precio de compra de cada elemento, así como su duración promedio.

Componente	Precio	Vida útil media
Ordenador portátil	1499,00 €	6 años
Ratón y teclado	49,00 €	3 años
Monitor	119 ,00 €	6 años
Disco duro externo	45,00 €	5 años

Tabla 5 - Costes componentes Hardware

COSTES SOFTWARE

Se refieren a los gastos generados por todos los programas y herramientas informáticas empleados para llevar a cabo el proyecto. En el siguiente cuadro se muestra el desenlace de esta evaluación:

Componente	Precio	Vida útil media
Windows 10 home	120,00 €	4 años
Office 365	0,00 €	-
Visual Studio Code	0,00 €	-
XAMPP	0,00 €	-
GitHub	0,00 €	-
Google chrome	0,00 €	-

Tabla 6 - Costes componentes Software

OTROS COSTES

Son costes que intervienen en el desarrollo del proyecto, pero no se pueden clasificar en ninguna de las anteriores tipologías recogidas. Cabe destacar que después de la fase de desarrollo, en el caso de que el cliente final quiera quedarse con el sistema, deberá añadirse un gasto mensual para el alquiler de un servidor. Para este caso tomaremos como ejemplo el coste del plan de servidor virtual de IONOS En la siguiente tabla aparece reflejado el resultado de este análisis:

Otros gastos	Precio/mes
Servicios básicos	30,00 €/mes
Conexión a internet	52,50 €/mes
Alquiler servidor	1,00 €/mes

Tabla 7 - Otros costes

ESTIMACIONES INICIALES

En este momento se realizará la valoración inicial de los gastos y el tiempo del proyecto. Para esto se utilizan tres de los principales métodos de valoración utilizados en el desarrollo de software: valoración por casos de uso, método de Albretch con puntos de función y el método COCOMO.

De estos tres procedimientos, la valoración por casos de uso es el más preciso, siempre y cuando se cuente con unos casos de uso bien definidos y analizados previamente. Además, se llevarán a cabo los procedimientos de Albretch y COCOMO de manera complementaria, aunque estos no son tan precisos como el primero.

Por último, se explican y comparan los resultados obtenidos en todos los procedimientos junto con la planificación inicial propuesta para obtener una visión temporal y presupuestaria más adecuada para el proyecto.

ESTIMACIÓN POR PUNTOS DE FUNCIÓN (MÉTODO ALBRETCH)

Basándose en la perspectiva del usuario acerca del tamaño del sistema, se consideran los distintos ámbitos de información para definir el sistema, sin tener en cuenta las especificaciones técnicas y detalles de implementación o código. Puesto que esta estimación es más propia de los usuarios, se presta atención a los siguientes campos de información:

- **Entradas de usuario**: información que el usuario introduce en el sistema.
- **Salidas de usuario**: información que el sistema presenta al usuario una vez procesada la petición de este.
- **Consultas:** peticiones o entradas que requieren de una respuesta por parte del sistema.
- **Ficheros lógicos internos**: utilizados exclusivamente por el sistema de forma local.
- **Ficheros lógicos externos**: externos, los cuales el sistema utiliza para el procesamiento de datos

Una vez establecidos los campos de información, es necesario determinar la complejidad de cada una de las funcionalidades del sistema. Para ello, se utilizan las tablas siguientes:

Entradas de datos

Descripción	Complejidad
Acceso a la aplicación	Baja
Creación de reservas	Baja
Modificación reservas	Baja
Gestionar comedores	Baja
Elegir posición mesa	Baja
Elegir combinación de mesas	Baja
Cambiar fecha de visualización de datos	Baja

Tabla 8 - Entradas de datos

Salidas de datos

Descripción	Complejidad
Información turno	Baja
Distribución espacial de mesas	Media
Lista de reservas	Baja
Detalle reserva	Baja
Mensajes de estado	Baja

Tabla 9 - Salida de datos

Consultas

Descripción	Complejidad
Acceso a la aplicación	Baja
Creación de reservas	Baja
Lista de mesas (simples) disponibles	Media
Creación de datos nuevos para fecha	Baja
Lista de mesas combinadas disponibles	Alta
Asignación reserva-mesa simple	Baja
Asignación reserva-mesa combinada	Media
Creación de reserva con sala privada	Media
Modificación comedores turno	Baja

Tabla 10 - Consultas

Ficheros lógicos internos de datos

Descripción	Complejidad
Comedores	Baja
Distribuciones	Media
Mesas	Baja
Posiciones	Baja
Reservas	Baja
Turnos	Baja
Usuarios	Baja

Tabla 11 - Ficheros lógicos internos

Después de establecer todas las complejidades requeridas por parte del sistema, es momento de calcular los puntos de función sin ajustar (PFNA) en base a la tabla que se muestra a continuación:

Campos	Complejidad	Peso por complejidad	Nº de funciones	Total
Entradas de Usuario	Baja	x 3	7	21
	Media	x 4	0	0
	Alta	x 6	0	0
Salidas de usuario	Baja	x 4	4	16
	Media	x 5	1	5
	Alta	x 7	0	0
Consultas de Usuario	Baja	x 3	5	15
	Media	x 4	3	12
	Alta	x 6	1	6
Ficheros lógicos internos	Baja	x 7	6	42
	Media	x 10	1	10
	Alta	x 15	0	0
Ficheros lógicos externos	Baja	x 5	0	0
	Media	x 7	0	0
	Alta	x 10	0	0
Puntos de Función Sin Ajustar (PFNA)				127

Tabla 12 - Cálculo de PFNA

Ya definidos los puntos de función sin ajustar (PFNA) es momento de asignar el grado de complejidad de los 14 factores de ajuste que caracterizan la complejidad y la funcionalidad del sistema. Para ello se cuenta con una escala que va desde el 0 al 5, puntuando los factores de complejidad descritos en la siguiente tabla:

Factor de ajuste	Complejidad
Comunicación de datos	3
Funciones distribuidas	2
Prestaciones	3
Gran uso de la configuración	0
Velocidad de las transacciones	3
Entrada online de datos	2
Diseño para la eficiencia del usuario final	3
Actualización de datos online	3
Complejidad de procesos lógicos de la aplicación	4
Reusabilidad de código	2
Fácil instalación	2
Facilidad de operación	2
Localizaciones múltiples	3
Facilidad de cambios	1
Total	33

Tabla 13 - Factor de ajuste

Para calcular el factor de ajuste (FA) se aplica la siguiente fórmula basada en la suma total de los factores de complejidad (FC) obtenidos previamente:

FA = 0,65 + (0,01 * FC) =0,65 + (0,01 * 33) = **0,98**

Una vez obtenido el factor de ajuste, se calculan los puntos de función ajustados (PFA):

PFA = PFNA * FA = 127 * 0,98 = **124,46**

Para finalizar la estimación es necesario calcular tanto la duración del proyecto como el esfuerzo personal en horas. Para ello es necesario establecer la equivalencia puntos de horas por puntos de función contando con que 1 mes de esfuerzo (16 días laborables) equivale a 9 puntos de función.

Duración de proyecto estimada = 124.46 PFA / 9 = **13,82 meses**

ESTIMACIÓN POR COCOMO

Este tipo de cálculo se fundamenta en el tamaño en líneas de código obtenido a través de los valores obtenidos en la estimación por puntos de función realizada en la sección previa "Estimación por puntos de función (Método Albretch)". Las mediciones obtenidas indican el tiempo y esfuerzo dedicados en el desarrollo del sistema.

Antes de iniciar, es importante conocer las distintas variantes de COCOMO que se pueden utilizar en función de las características del proyecto a realizar:

- **Orgánico**: Empleado en proyectos sencillos, pequeños, con un bajo número de programadores y con requisitos bastante adaptables.
- **Empotrado**: Empleado en proyectos complejos, con una gran cantidad de requisitos caracterizados por su rigidez.
- **Semi-empotrado:** Punto intermedio entre los dos anteriores.

Una vez definidas y observadas las características principales del proyecto y los requisitos que se deben tener en cuenta, la opción escogida es la estimación a través de un modelo semi-empotrado.

El siguiente paso es determinar el número de líneas de código estimadas necesarias para la realización de un punto de función. Para los lenguajes utilizados se han considerado 38 líneas de PHP/Punto de función y 50 líneas de JS/Punto de función, dando una media de unas 45 líneas/puntos de función.

LDC: PFA * (Líneas/PF) = 124.46 * 45 = **5600 líneas de código**

Después de haber realizado el cálculo de las líneas de código y considerando que el progreso de la aplicación se está llevando a cabo en un ambiente estable y se estima que está por debajo de las 50 KLDC, se opta por emplear el modelo orgánico de evaluación. Para llevar a cabo dicho procedimiento, es fundamental aplicar un factor de esfuerzo que se basa en 15 características a tener en cuenta reflejados de la siguiente manera:

Característica	Factor	Valor
Fiabilidad requerida	Alto	1,15
Tamaño de la base de datos	Medio	1,00
Complejidad del Software	Medio	1,00
Restricciones en tiempo de ejecución	Alto	1,11
Restricciones de memoria	Medio	1,00
Volatilidad del Hardware	Bajo	0,87
Restricciones de tiempo de respuesta	Alto	1,07
Calidad de los analistas	Medio	1,00
Experiencia con el tipo de aplicación	Medio	1,00
Experiencia en el Hardware	Alto	0,9
Experiencia con el lenguaje de programación	Alto	0,95
Calidad de los programadores	Medio	1,00
Técnicas modernas de programación	Medio	1,00
Empleo de herramientas	Alto	0,91
Restricciones a la duración del proyecto	Bajo	1,08

Tabla 14 - Calificación valores de esfuerzo

A continuación, se multiplican los valores de la tabla entre sí para obtener el valor mx utilizado para el posterior cálculo del esfuerzo del proyecto (E) y el tiempo de desarrollo de este (TDEV). Para obtener estos valores, es necesario utilizar las constantes en función del modelo de COCOMO elegido, en este caso los valores son:

$$\mathbf{a} = 3 \quad \mathbf{b} = 1{,}12 \quad \mathbf{c} = 2{,}5 \quad \mathbf{d} = 0{,}35$$

Ahora, podemos realizar los cálculos utilizando dichas variables:

$$\mathbf{mx} = 1{,}15 * 1{,}11 * 0{,}87 * 1{,}07 * 0{,}9 * 0{,}95 * 0{,}91 * 1{,}08 = \mathbf{0{,}9985}$$

$$\mathbf{E} = a * (KLDC)^{b} * mx = 3 * (5{,}6)^{1{,}12} * 0{,}9985 = \mathbf{20{,}627\ persona\text{-}mes}$$

$$\mathbf{T_{DEV}} = c * E^{d} = 2{,}5 * (20{,}627)^{0{,}35} = \mathbf{7{,}2\ meses}$$

Con todo esto, podemos concluir que se necesitan 4 personas a tiempo completo para llevar a cabo el desarrollo del proyecto en 8,6 meses.

ESTIMACIÓN POR CASOS DE USO

Para realizar este tipo de estimación es necesario tener en cuenta los casos de uso definidos para el proyecto, junto con los actores que intervienen en él. El resultado es la obtención del esfuerzo necesario para completar el proyecto. Para conseguir esto, el modelo de estimación constará de 4 etapas:

ETAPA 1:

Obtener el factor de peso de los actores sin ajustar (UAW) basándose en el grado de complejidad del actor dado que la aplicación solo cuenta con uno, el cuál entra dentro de la categoría de complejo, tenemos un valor de 3.

UAW: 3

ETAPA 2:

Obtener el factor de peso de los casos de uso sin ajustar (UUAW) que describen el grado de complejidad de los casos de uso del proyecto. En este caso lo realizaremos basándonos en el número de transacciones que realiza cada caso de uso.

Tipo de Caso de uso	Descripción	Factor
Simple	Número de transacciones < 4	5
Medio	4 < Número de transacciones < 7	10
Alto	7 < Número de transacciones	15

Tabla 15 - Ponderaciones para el cálculo de UUAW basado en transacciones

Caso de uso	Transacciones	UUCW
CU-01 - Inicio de sesión	1	5
CU-02 - Cambiar fecha	8	15
CU-03 – Visualizar datos	2	5
CU-04 – Abrir/cerrar comedores	2	5
CU-05 - Crear reserva (mesa simple)	6	10
CU-06 - Crear reserva (mesas combinadas)	12	15
CU-07 - Crear reserva (sala privada)	8	15
CU-08 – Cerrar sesión	1	5
UUCW		75

Tabla 16 - Cálculo de UUCW

ETAPA 3:

Obtener los puntos de caso de uso ajustados (UCP). Dicho valor se obtiene de la multiplicación de los puntos de caso de uso sin ajustar (UAW + UUCW) con los factores técnicos (TCF) y los factores ambientales (EF).

1. Puntos de caso de uso sin ajustar (UUCP)

   UUCP = UAW + UUCW = 13 + 195 = **208**

2. Factor de complejidad técnica (TCF). Se obtienen de puntuar 13 características basadas en la complejidad del sistema y ponderarlas con sus pesos correspondientes. Las puntuaciones van de 0 a 5.

Descripción	Peso	Valor	Resultado
Sistema distribuido	2	1	2
Tiempos de respuesta	1	4	4
Eficiencia de usuario final	1	4	4
Procesamiento interno complejo	1	5	5
Código reutilizable	1	3	3
Facilidad de instalación	0,5	5	2,5
Facilidad de uso	0,5	5	2,5
Portabilidad	2	2	4
Facilidad de cambio	1	0	0
Concurrencia	1	5	5
Objetivos especiales de seguridad	1	2	2
Acceso directo a tercera partes	1	1	1
Formación de los usuarios	1	4	4
Resultado			39

Tabla 17 - Factores de complejidad técnica (TCF)

Una vez puntuados, se calcula el factor de complejidad técnica (TCF):

TCF = 0,06 + (0,01 * Resultado Total) = 0,06 + (0,01 * 39) = **0,45**

3. **Factor ambiental (EF).** Se obtienen a partir de puntuar 8 características basadas en la experiencia, conocimientos y habilidades de todo el personal implicado en el proyecto con sus pesos correspondientes. Las puntuaciones se establecen de la misma forma que las puntuaciones de los factores de complejidad técnica (TCF).

Descripción	Peso	Valor	Resultado
Familiaridad con el modelo de proyecto	1,5	3	4,5
Experiencia en la aplicación	0,5	3	1,5
Experiencia en orientación a objetivos	1	5	5
Capacidad del analista líder	0,5	3	1,5
Motivación	1	5	5
Estabilidad de los requerimientos	2	4	8
Personal a tiempo parcial	-1	0	0
Dificultad del lenguaje de programación	-1	2	-2
Resultado			22,5

Tabla 18 - Factores ambientales (EF)

Una vez puntuados, se calcula el factor ambiental (EF):

EF = 1,4 + (-0,03 * Resultado Total) = 1,4 + (-0,03 * 22,5) = **0,725**

Una vez obtenidos todos estos valores, se calculan los puntos de función ajustados (UCP):

UCP = UUCP * TCF * EF = 208 * 0,45 * 0,725 = **67,86**

ETAPA 4:

Obtener el esfuerzo horas-persona (E). Dicho valor se obtiene de la multiplicación entre los puntos de caso de uso ajustados (UCP) y el número de horas-persona (CF).

Número de horas-persona (CF):

Factor de productividad (FP). Este valor se obtiene de la suma en primer lugar de los factores ambientales (EF) comprendidos entre el E1 y E6 (ambos inclusive) con un valor inferior a 3 y los factores ambientales E7 y E8 con un valor superior a 3

FP = 2 + 0 = **2**

Con el factor de productividad obtenido se deben ajustar el número de horas-persona en base a este factor, en este caso al ser el FP <= 2, el número de horas-persona (CF) es 20.

Esfuerzo (E)

E = UCP * CF = 67,86 * 20 = **1618,2 horas-persona**

El resultado obtenido se corresponde con el esfuerzo realizado en la fase de codificación del proyecto (representa un 40% del total del proyecto). Por lo tanto, el último paso es calcular el resto de las horas del proyecto utilizando el siguiente modelo:

Fase de desarrollo	% del proyecto	Horas-Persona
Análisis	10 %	339,250
Diseño	20 %	678,500
Programación	40 %	1357,000
Pruebas	15 %	508,875
Sobrecarga	15 %	508,875
Total		3392,500

Tabla 19 - Modelo de esfuerzo total

Al final, al estimar que solo se cuenta con 1 trabajador para englobar todo el desarrollo del proyecto, la duración total del mismo asciende a 20,2 meses.

COMPARATIVA DE LAS ESTIMACIONES

La información obtenida después de realizar las estimaciones oportunas es la siguiente:

Hay que tener en consideración:

Modelo de estimación	Personas	Tiempo
Puntos de función	1	14 meses
COCOMO	4	7,2 meses
Casos de uso	1	20 meses

Tabla 20 - Comparativa estimaciones inicial

En que el proyecto solo será desarrollado por una única persona y que, por ese motivo, la traducción final de los datos obtenidos es la siguiente:

Modelo de estimación	Personas	Tiempo
Puntos de función	1	14 meses
COCOMO	1	30 meses
Casos de uso	1	20 meses

Tabla 21 - Comparativa estimaciones inicial

Observando los resultados y comparándolo con la primera planificación temporal realizada en el apartado “Planificación inicial del proyecto”, la estimación más precisa y la que más se ajusta se corresponde con la estimación de Puntos de Función (Método Albretch).

PRESUPUESTOS

Una vez realizados todos los análisis y estimaciones iniciales del proyecto, es el momento de realizar el cálculo de los presupuestos previos y posteriores al desarrollo del proyecto.

PRESUPUESTO INICIAL

Una vez calculada la estimación más precisa para el proyecto, en este apartado se lleva a cabo el cálculo de los presupuestos en base a la planificación inicial y a la estimación por puntos de función (Método de Albretch).

Costes de personal

Se considerará el número de horas que ha dedicado cada miembro del equipo de trabajo por separado, aplicándole su correspondiente tasa por cada hora de trabajo.

Rol	Tiempo	Salario	Salario / Hora	Coste en proyecto
Jefe de proyecto	150 horas	4000 €/mes	23,80 €/h	3570 €
Analista	200 horas	3400 €/mes	20,25 €/h	4050 €
Diseñador UX/UI	200 horas	1500 €/mes	8,90 €/h	1780 €
Desarrollador	350 horas	2100 €/mes	15,50 €/h	5425 €
Tester	100 horas	1600 €/mes	9,50 €/h	950 €
Total	1000 horas	-	-	15775 €

Tabla 22 - Coste inicial personal

Por otro lado, en la evaluación de estos gastos, debido a la falta de información sobre el porcentaje de trabajo efectuado por cada uno de los roles involucrados, se ha optado por emplear una tasa de horas de trabajo que corresponde al promedio salarial de todos los miembros del equipo identificados anteriormente.

Salario Medio = 4000 + 3400 + 1500 + 2100 + 1600 = 2520 € / mes = **15€ hora**

Por lo tanto, el coste de personal para la estimación asciende a:

13,82 meses * 75 h/mes = 1036 horas aproximadamente

Coste Personal = 1036 * 15 = **15540 €**

Costes hardware y software

Se ha considerado la depreciación basada en el lapso de vida útil aproximado de cada uno de los elementos, sumado al valor de su adquisición, para efectuar el cálculo. Mediante este procedimiento se determina el costo real del uso de estos componentes en el proyecto. Dado que la programación y la estimación son casi idénticas en duración, se ha redondeado el período de depreciación a 13 meses para ambos cálculos.

Componente	Precio	Vida útil media	Porcentaje de uso	Coste real en proyecto (€)
Ordenador portátil	1499 €	6 años	18 %	269,82 €
Ratón y teclado	49€	3 años	36 %	17,64 €
Monitor	119 €	6 años	18 %	21,42 €
Disco duro externo	45€	5 años	21 %	9,45 €
Total				318,33 €

Tabla 23 - Coste inicial Hardware

Componente	Precio	Vida útil media	Porcentaje de uso	Coste real en proyecto (€)
Windows 10 home	120 €	4 años	27 %	32,4 €
Office 365	0 €	-	-	-
Visual Studio Code	0 €	-	-	-
XAMPP	0 €	-	-	-
GitHub	0 €	-	-	-
Google chrome	0 €	-	-	-
Total				32,4 €

Tabla 24 - Coste inicial Software

Otros costes

Para realizar el cálculo, la duración inicial de proyecto se ha estimado en 13 meses.

Otros gastos	Precio/mes	Coste real en proyecto (€)
Servicios básicos	30 €/mes	390 €
Conexión a internet	52,50 €/mes	682,5 €
Alquiler servidor	1 €/mes	13 €
Total		1075,5 €

Tabla 25 - Coste inicial otros aspectos

Resultado coste inicial

Tipo de gasto	Coste
Personal	15775 €
Hardware	318,33 €
Software	32,4 €
Otros	1075,5 €
Total	17201,23 €

Tabla 26 - Resultado coste inicial

COSTE REAL

Tras finalizar el proyecto, se realiza el cálculo del costo efectivo en base a los factores que han influido directamente en la planificación inicial durante su desarrollo.

La fecha de finalización real del proyecto se ha retrasado hasta el 15 de junio de 2023, lo que representa un incremento de 48 días laborables respecto a la fecha prevista en la planificación original. Los principales motivos de este retraso han sido:

- Falta en la experiencia en la planificación del proyecto
- Compaginación con la vida laboral
- Incremento de 1 semana por problemas con la gestión de las distribuciones
- Incremento de horas de trabajo destinadas a la revisión de la documentación por parte del equipo de trabajo completo.

Los 48 días generan unas 144 horas que han sido repartidas entre los diferentes roles según su impacto en el área correspondiente.

Costes de personal

Rol	Tiempo	Salario	Salario / Hora	Conste en proyecto
Jefe de proyecto	150 + 5 horas	4000 €/mes	23,80 €/h	3689 €
Analista	200 + 15 horas	3400 €/mes	20,25 €/h	4353,75 €
Diseñador UX/UI	200 + 16 horas	1500 €/mes	8,90 €/h	1922,4 €
Desarrollador	350 + 72 horas	2100 €/mes	15,50 €/h	6541 €
Tester	100 + 36 horas	1600 €/mes	9,50 €/h	1292 €
Total	1000 horas	-	-	17798,15 €

Tabla 27 - Coste final personal

Costes hardware y software

Componente	Precio	Vida útil media	Porcentaje de uso	Coste real en proyecto (€)
Ordenador portátil	1499 €	6 años	20 %	299,8 €
Ratón y teclado	49€	3 años	42 %	20,58 €
Monitor	119 €	6 años	20 %	23,8 €
Disco duro externo	45€	5 años	25 %	11,25 €
Total				355,43 €

Tabla 28 - Coste inicial Hardware

Componente	Precio	Vida útil media	Porcentaje de uso	Coste real en proyecto (€)
Windows 10 home	120 €	4 años	31 %	37,2 €
Office 365	0 €	-	-	-
Visual Studio Code	0 €	-	-	-
XAMPP	0 €	-	-	-
GitHub	0 €	-	-	-
Google chrome	0 €	-	-	-
				37,2 €

Tabla 29 - Coste final Software

Otros costes

Otros gastos	Precio/mes	Coste real en proyecto (€)
Servicios básicos	30 €/mes	450 €
Conexión a internet	52,50 €/mes	787,5 €
Alquiler servidor	1 €/mes	15 €
Total		1252,5 €

Tabla 30 - Coste inicial otros aspectos

Resultado coste final

Tipo de gasto	Coste
Personal	17798,15€
Hardware	355,43 €
Software	37,2 €
Otros	1252,5 €
Total	19443,28 €

Tabla 31 - Resultado coste final

CAPITULO 4. ANÁLISIS DEL SISTEMA

En esta sección se presenta, exhaustivamente, el estudio realizado durante este proyecto y que consiste, principalmente, en la explicación detallada de todos los aspectos que deben considerarse en cada etapa del desarrollo del producto, además de los esquemas necesarios para una mejor comprensión. Estos aspectos están estructurados en formatos (tablas) para estandarizar su análisis en todo momento, junto con el diagrama de escenarios y el modelo de entidad-relación, cuyas propiedades se detallan a lo largo de esta sección.

ACTORES DEL SISTEMA

Un actor es cualquier elemento que desempeñe un rol concreto o interacción en el sistema. Pueden ser desde personas físicas hasta sistemas software externos.

En el caso de esta aplicación web privada, solo se considera un actor que será cualquier usuario del casares que acceda a la web.

Id	Nombre	Descripción
ACT-01	Usuario administrador	Usuario registrado y que posteriormente ha iniciado sesión en la aplicación.

Tabla 32 - Actores del sistema

REQUISITOS DE USUARIO

Un requisito de usuario es toda aquella necesidad propia de un usuario que el sistema deberá satisfacer.

Id	Descripción
RU-01	El usuario puede iniciar sesión en la aplicación.
RU-02	El usuario puede visualizar un resumen de información de los comedores del turno.
RU-03	El usuario puede visualizar un resumen de información de las reservas del turno.
RU-04	El usuario puede abrir y cerrar comedores.
RU-05	El usuario puede ver una distribución espacial de las mesas de los comedores
RU-06	El usuario debe diferenciar en la distribución las mesas con reserva y las mesas libres.
RU-07	El usuario puede ver una lista de reservas con el resumen de la información más importante.
RU-08	El usuario puede ver todos los datos de una reserva.
RU-09	El usuario puede modificar los datos de una reserva.
RU-10	EL usuario puede crear reservas, seleccionando la mesa donde se colocaran.
RU-11	En el caso de no disponer de una mesa lo suficientemente grande para la reserva, el usuario podrá crear una reserva colocándola en varias mesas que posteriormente se combinaran y se reflejará en la distribución de las reservas.
RU-12	El usuario podrá crear reservas en las salas privadas, esto se reflejará en el resumen de comedores y en la distribución espacial.
RU-13	El usuario podrá cancelar reservas liberando sus mesas para nuevas reservas.
RU-14	En el caso de cancelar una reserva que ha combinado diferentes mesas, estas volverán a su estado original y se reflejara en la distribución espacial.
RU-15	El usuario podrá cancelar reservas de salas privadas, volviendo estas a abrirse y reflejándose en la distribución espacial y en el resumen de comedores.
RU-16	El usuario tendrá acceso a la libre navegación entre los diferentes comedores y turnos del día seleccionado, pudiendo cambiar de fecha.

Tabla 33 - Requisitos de usuario

MODELO DE CASOS DE USO

El caso de uso se emplea para mostrar las conexiones y los procedimientos de intercambio de datos entre el programa y los usuarios. Aquí es donde se reflejan gráficamente los requerimientos del usuario descritos con anterioridad, estructurados con la colaboración de los diagramas de casos de uso.

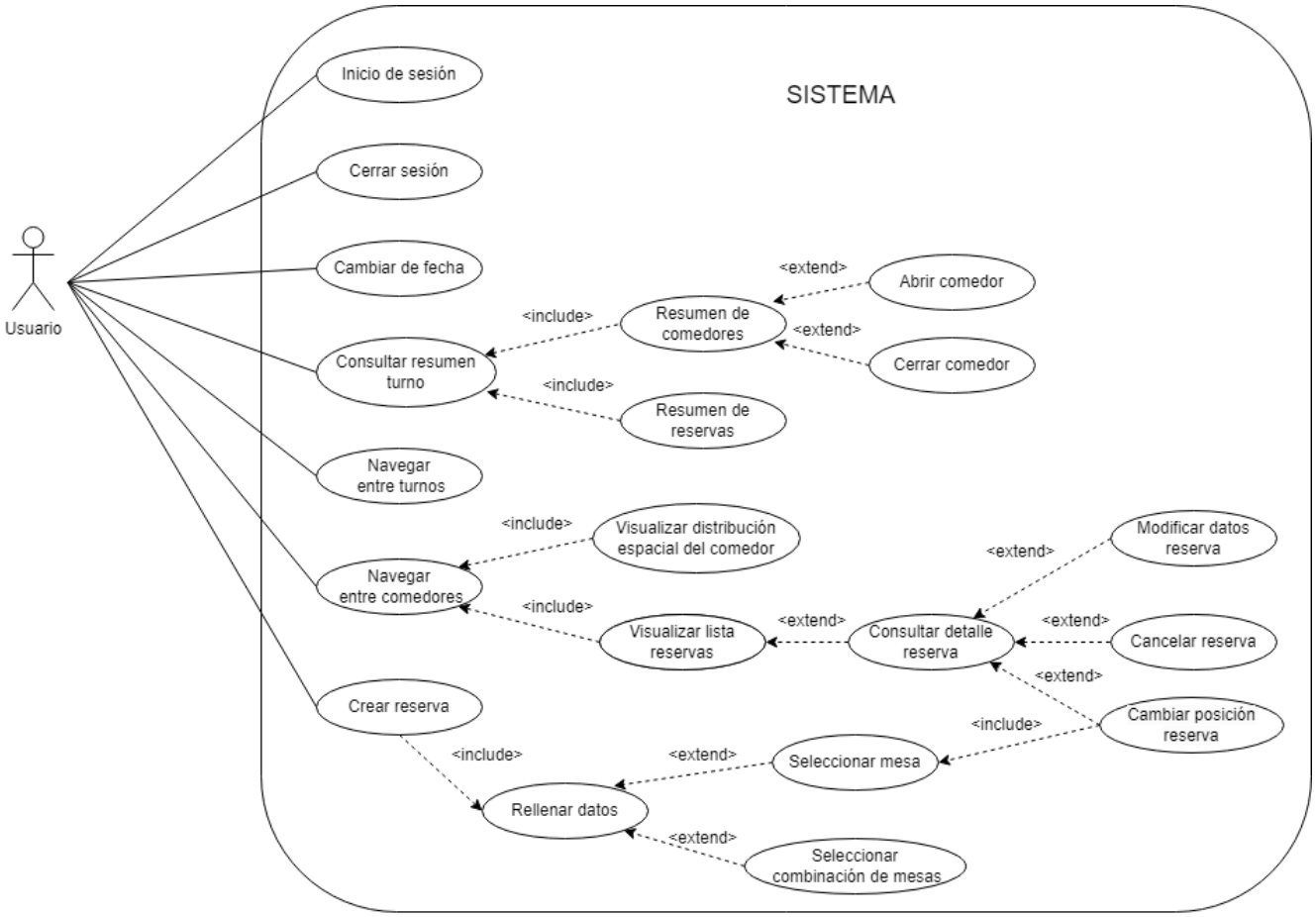


Figura 8 - Diagrama de casos de uso

A continuación, se analizará cada uno de los casos de uso con el fin de comprender la funcionalidad completa del sistema y las relaciones existentes entre requisitos.

Caso de uso	**CU-01 Inicio de sesión**		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Baja
Descripción	El usuario se acredita ante el sistema a través de su usuario y contraseña		
Precondiciones	1. El usuario debe encontrarse registrado en el sistema. 2. El usuario debe de contar con una conexión de Internet para conectarse al servidor.		
Secuencia Normal	1. EL usuario accede a la página web. (Al login o a la página principal, en este caso si no está autenticado le redirigirá al login) 2. El usuario rellena el formulario con su usuario y contraseña. 3. EL sistema comprueba que el usuario exista y su contraseña sea correcta. 4. El sistema redirige al usuario a la página principal aplicación		
Secuencia Alternativa			
Excepciones en la secuencia	3.1. Los datos de acceso son incorrectos y se redirige al usuario al paso 2.		
Postcondiciones	Acceso del usuario a la página principal de la aplicación		

Tabla 34 - CU-01 Inicio de sesión

Caso de uso	**CU-02 Cerrar sesión**		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Media	**Frecuencia**	Baja
Descripción	El usuario cierra sesión y sale de la página principal.		
Precondiciones	1. El usuario debe encontrarse e la página principal de la aplicación.		
Secuencia Normal	1. EL usuario pulsa el botón de cerrar sesión. 2. El sistema cierra la sesión y redirige al usuario al login.		
Secuencia Alternativa			
Excepciones en la secuencia	La sesión del usuario dentro de la aplicación ha sido cerrada		
Postcondiciones	Acceso del usuario a la página principal de la aplicación		

Tabla 35 - CU-02 Cerrar sesión

Caso de uso	CU-03 Cambiar de fecha		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Media
Descripción	El usuario cambia la fecha para visualizar los turnos y reservas de dicho día.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario pulsa en el input de fecha 2. El usuario cambia de fecha 3. El sistema le devuelve los datos correspondientes		
Secuencia Alternativa			
Excepciones en la secuencia	3.1. No existen datos. 3.2. Se crea la estructura básica de datos de la fecha 3.3. Se devuelven los datos creados.		
Postcondiciones	El usuario ve los datos correspondientes a la nueva fecha		

Tabla 36 - CU-03 Cambiar de fecha

Caso de uso	CU-04 Consultar resumen turno		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Media	**Frecuencia**	Media
Descripción	El usuario puede ver un resumen del estado del turno		
Precondiciones	1. El usuario debe encontrarse e la página principal de la aplicación.		
Secuencia Normal	1. El usuario visualiza el resumen del turno. 2. Puede ver el resumen de comedores del turno (**CU-05**). 3. Pude ver el resumen de reservas del turno (**CU-06**).		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario puede visualizar el resumen del turno		

Tabla 37 - CU-04 Consultar resumen turno

Caso de uso	CU-05 Consultar resumen de comedores		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Media
Descripción	El usuario puede ver el resumen del estado de los comedores de un turno. Visualizando si los comedores están abiertos o cerrados y el estado de las salas privadas.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. Dentro del resumen del turno en el apartado de los comedores visualiza: - Si cada uno de los comedores está abierto. - El estado de las salas privadas de los comedores que cuenten con ellas.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario puede visualizar el resumen de comedores del turno		

Tabla 38 - CU-05 Consultar resumen de comedores

Caso de uso	CU-06 Abrir comedor		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Baja
Descripción			
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. Algún comedor debe estar cerrado		
Secuencia Normal	1. El usuario marca como abierto un comedor cerrado. 2. El sistema cambia el estado del comedor a “Abierto” en la base de datos		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El comedor seleccionado se ha abierto para dicho turno.		

Tabla 39 - CU-06 Abrir comedor

Caso de uso	CU-07 Cerrar comedor		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Baja
Descripción			
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. Algún comedor debe estar abierto y no debe tener reservas		
Secuencia Normal	1. El usuario marca como cerrado un comedor abierto. 2. El sistema cambia el estado del comedor a “Cerrado” en la base de datos		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El comedor seleccionado se ha cerrado para dicho turno.		

Tabla 40 - CU-07 Cerrar comedor

Caso de uso	CU-08 Consultar resumen de reservas		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Media
Descripción	El usuario puede ver el resumen del estado de las reservas de un turno. Visualizando el numero de comensales en cada comedor y el total del turno. También en el nº de cuartos de cordero reservados.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. Dentro del resumen del turno en el apartado de las reservas visualiza: - El total de reservas del turno - El desglose del nº de reservas en cada comedor - EL nº de cuartos de cordero reservados		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario puede visualizar el resumen de reservas del turno		

Tabla 41 - CU-08 Consultar resumen de reservas

Caso de uso	CU-09 Navegar entre turnos		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Alta
Descripción	El usuario puede cambiar la información que visualiza entre los diferentes turnos de la fecha en la que se encuentra		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario interacciona con los botones de cambio de turno. 2. El sistema muestra la información del siguiente turno o del anterior.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario cambia fácilmente de turno para visualizar la información		

Tabla 42 - CU-09 Navegar entre turnos

Caso de uso	CU-10 Navegar entre comedores		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Alta
Descripción	El usuario puede cambiar la información que visualiza entre los diferentes comedores del turno en el que se encuentra		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario interacciona con los botones de cambio de comedor. 2. El sistema muestra la información del siguiente comedor o del anterior. 3. El usuario visualiza: - La distribución espacial del comedor (**CU-11**). - La lista de reservas del comedor (**CU-12**)		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario cambia fácilmente del comedor para visualizar la información		

Tabla 43 - CU-10 Navegar entre comedores

Caso de uso	CU-11 Visualizar la distribución espacial del comedor		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Alta	Frecuencia	Alta
Descripción	El usuario visualiza la distribución espacial del comedor en el que se encuentra.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario muestra un plano con la distribución de las mesas del comedor, señalando cuales están reservadas y cuales libres.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	EL usuario puede ver la situación de las mesas del comedor en un plano		

Tabla 44 - CU-11 Visualizar la distribución espacial del comedor

Caso de uso	CU-12 Visualizar lista de reservas		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Alta	Frecuencia	Alta
Descripción	El usuario visualiza una lista de reservas donde se muestra la información más importante.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario visualiza la lista de reservas del comedor en el que se encuentra donde visualiza la información más importante como el nombre de la persona que reservo, el nº de comensales o la hora entre otros.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	EL usuario puede ver la información mas importante en la lista de reservas		

Tabla 45 - CU-12 Visualizar lista de reservas

Caso de uso	CU-13 Consultar detalle de reserva		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Alta	Frecuencia	Media
Descripción	El usuario visualiza toda la información de una reserva.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario pulsa una ficha de reserva dentro de la lista de reservas. 2. Salta un pop-up donde se muestra la información de la reserva.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	El usuario ve toda la información de la reserva.		

Tabla 46 - CU-13 Consultar detalle de reserva

Caso de uso	CU-14 Modificar datos de reserva		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Baja
Descripción	El usuario modifica los datos de una reserva.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe encontrarse en el pop-up de detalle de la reserva		
Secuencia Normal	1. El usuario clica en el botón de modificar reserva. 2. El sistema cambia la información de la reserva por un formulario con los datos que puede modificar. 3. El usuario da a continuar y el sistema guarda en la base de datos los cambios realizados.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	La reserva ha sido modificada.		

Tabla 47 - CU-14 Modificar datos de reserva

Caso de uso	CU-15 Cambiar la posición de la reserva		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Baja
Descripción	El usuario cambia la posición de una reserva.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe encontrarse en el pop-up de detalle de la reserva		
Secuencia Normal	1. El usuario clica en el botón de cambiar posición 2. El sistema permite al usuario seleccionar una nueva posición para la reserva (**CU-19**)		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	La posición de la reserva ha cambiado.		

Tabla 48 - CU-15 Cambiar la posición de la reserva

Caso de uso	CU-16 Cancelar reserva		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Media	Frecuencia	Baja
Descripción	El usuario cancela una reserva		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe encontrarse en el pop-up de detalle de la reserva		
Secuencia Normal	1. El usuario pulsa el botón de cancelar. 2. El sistema, en base de datos, cancela la reserva y libera la mesa para que este disponible para una nueva reserva		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	La reserva ha sido cancelada y la mesa liberada.		

Tabla 49 - CU-16 Cancelar reserva

Caso de uso	CU-17 Crear reserva		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Alta
Descripción	Proceso de creación de una reserva		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación.		
Secuencia Normal	1. El usuario pulsa el botón de crear reserva. 2. Salta un Pop-up con un formulario para rellenar los datos de la reserva. (**CU-18**). 2.1. Se muestra un selector de mesas para la reserva (**CU-19**). 3. Se guarda la reserva en la base de datos.		
Secuencia Alternativa	2.2. EL nº de comensales es elevado y se necesita juntar mesas. Se muestra un selector de combinaciones mesas para la reserva (**CU-20**). 2.3. El tipo de reserva es privado y no se seleccionan mesas.		
Excepciones en la secuencia			
Postcondiciones	La reserva ha sido creada		

Tabla 50 - CU-17 Crear reserva

Caso de uso	CU-18 Rellenar datos		
Actor principal	Usuario	**Autor**	Alberto Herrero
Importancia	Alta	**Frecuencia**	Alta
Descripción	El usuario rellena los datos de la nueva reserva		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe haber comenzado el proceso de creación de reserva		
Secuencia Normal	1. El sistema muestra un formulario donde el usuario podrá rellenar los datos de la reserva. Entre estos, tenemos los siguientes campos obligatorios: - Nombre - Comensales - Teléfono de contacto - Hora - Fecha - Tipo (Normal/Salas privadas) - Nº de cuartos de cordero 2. EL usuario da a continuar para seguir con el proceso de creación.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	Los datos de la nueva reserva han sido rellenados.		

Tabla 51 - CU-18 Rellenar datos

Caso de uso	CU-19 Seleccionar mesa		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Alta	Frecuencia	Alta
Descripción	El usuario elige en que mesa quiere colocar la reserva.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe haber comenzado el proceso de creación de reserva y rellenado los datos de la reserva.		
Secuencia Normal	1. El sistema muestra una lista de las posiciones disponibles de cada comedor para la nueva reserva. 2. EL usuario selecciona una posición. 3. El usuario pulsa el botón de finalizar.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	La mesa para la reserva ha sido seleccionada.		

Tabla 52 - CU-19 Seleccionar mesa

Caso de uso	CU-20 Seleccionar combinación de mesas		
Actor principal	Usuario	Autor	Alberto Herrero
Importancia	Alta	Frecuencia	Alta
Descripción	El usuario elige que mesas combinar para la nueva reserva.		
Precondiciones	1. El usuario debe encontrarse en la página principal de la aplicación. 2. El usuario debe haber comenzado el proceso de creación de reserva y rellenado los datos de la reserva.		
Secuencia Normal	1. El sistema muestra una lista de las mesas que se pueden combinar en cada comedor para la nueva reserva. 2. EL usuario selecciona una combinación de mesas. 3. El usuario pulsa el botón de finalizar.		
Secuencia Alternativa			
Excepciones en la secuencia			
Postcondiciones	La combinación de mesas para la reserva ha sido seleccionada.		

Tabla 53 - CU-20 Seleccionar combinación de mesas

REGLAS DE NEGOCIO

Las reglas de negocio tienen una relevancia significativa en el proyecto, puesto que establecen las diversas políticas, limitaciones, ordenamientos y aspectos cruciales de una empresa, lo cual establece una serie de directrices y particularidades específicas en el proceso de desarrollo del proyecto. Con posterioridad a su estudio y análisis, se aplicarán las siguientes reglas de negocio:

ID	Descripción
RN-01	Las contraseñas se almacenan hasheadas en la base de datos (MD5).
RN-02	Los usuarios accederán a su área privada en la aplicación a través de sus credenciales correo electrónico y contraseña.
RN-03	Solo los usuarios registrados podrán visualizar y gestionar la información.

Tabla 54 - Reglas de negocio

REQUISITOS FUNCIONALES

Este tipo de requisitos describen todas aquellas características o aspectos fundamentales para el correcto desempeño de la funcionalidad del sistema, es decir, representan toda la funcionalidad que es necesaria desarrollar.

ID	Descripción
RF-01	El sistema presentará los campos necesarios para la introducción de información.
RF-02	El sistema será capaz de obtener la respuesta de los valores introducidos en los campos de inicio de sesión
RF-03	El sistema será capaz de recuperar la información de la base de datos y mostrar al usuario dicha información ordenada
RF-04	El sistema será capaz de crear una nueva reserva en base de datos con la información incorporada por el usuario y, si es necesario, actualizará los datos de la distribución y las posiciones que lo requieren.
RF-05	El sistema será capaz de devolver *feedback* al usuario en función del estado de inserción en base de datos.
RF-06	El sistema será capaz de modificar los datos de una reserva en la base de datos con la información incorporada por el usuario y, si es necesario, actualizará los datos de la distribución y las posiciones que lo requieren.
RF-07	El sistema será capaz de devolver *feedback* al usuario en función del estado de actualización en base de datos.

Tabla 55 - Requisitos funcionales

REQUISITOS NO FUNCIONALES

En este apartado se tienen en cuenta y analizan ciertos requerimientos, también denominados atributos de calidad, que completan aspectos concretos de la funcionalidad de la aplicación tales como comportamientos, restricciones o condicionantes para su correcto funcionamiento impuestos por el cliente. Estos se organizan según su tipología en diferentes categorías.

ID	Categoría	Descripción
RNF-01	Usabilidad	El tiempo requerido de aprendizaje por parte de un usuario del funcionamiento del sistema (una vez revisado el manual de usuario) no será superior a las 2 horas
RNF-02	Usabilidad	La tasa de errores cometidos por el usuario final debe de ser inferior al 1% del total.
RNF-03	Usabilidad	La aplicación debe caracterizarse por contar con un diseño *Responsive* para su correcta visualización y ejecución en ordenadores y tablets.
RNF-04	Usabilidad	El sistema debe presentar al usuario final mensajes de éxito y error.
RNF-05	Eficiencia	El sistema debe soportar una tasa de sesiones activas concurrentes de 100.
RNF-06	Eficiencia	Toda respuesta a cualquier transacción realizada por el usuario final o por el sistema no debe exceder un tiempo de respuesta de 20 segundos.
RNF-07	Eficiencia	La probabilidad de fallo de la aplicación debe ser inferior al 1%.
RNF-08	Seguridad	Las transacciones realizadas con servidor son siempre cifradas y realizadas bajo el protocolo seguro HTTPS
RNF-09	Seguridad	Las contraseñas se hashean mediante el algoritmo MD5.
RNF-10	Seguridad	La autenticación o inicio de sesión en la aplicación por parte de los usuarios se realiza a través de las credenciales de login y contraseña.
RNF-11	Disponibilidad	El sistema debe tener una disponibilidad 24/7 (24 horas al día los 7 días de la semana).
RNF-12	Disponibilidad	La disponibilidad del sistema cada vez que el usuario final intente acceder al mismo será del 99%.
RNF-13	Interfaces de comunicación	El sistema se comunica con el servidor a través del protocolo HTTPS utilizando servicios REST
RNF-14	Implementación	El desarrollo de la aplicación se realizará utilizando el entorno de desarrollo Visual Studio Code y como lenguaje de programación PHP para el *back-end* y JavaScript para el *front-end.*
RNF-15	Implementación	El diseño de la interfaz de usuario será desarrollado con HTML y CSS ampliándolo con el framework de Bootstrap.

Tabla 56 - Requisitos no funcionales

REQUISITOS DE INFORMACIÓN

ID	Descripción
RI-01	El sistema se encargará de almacenar la información de todos los usuarios de la plataforma.
RI-02	El sistema se encargará de almacenar la información de todas las reservas que se creen en el sistema
RI-03	El sistema se encargará de almacenar la información de los turnos
RI-04	El sistema se encargará de almacenar la información de las distribuciones
RI-05	El sistema se encargará de almacenar la información de las posiciones y mesas.
RI-06	El sistema se encargará de almacenar la información de los comedores

Tabla 57 - Requisitos de información

MODELO ENTIDAD-RELACION

Después de indicar todos los requisitos de información del sistema se representan a través de un diagrama entidad-relación. Este diagrama recoge los requisitos y les muestra de forma organizada indicando las principales entidades y relaciones que los caracterizan.

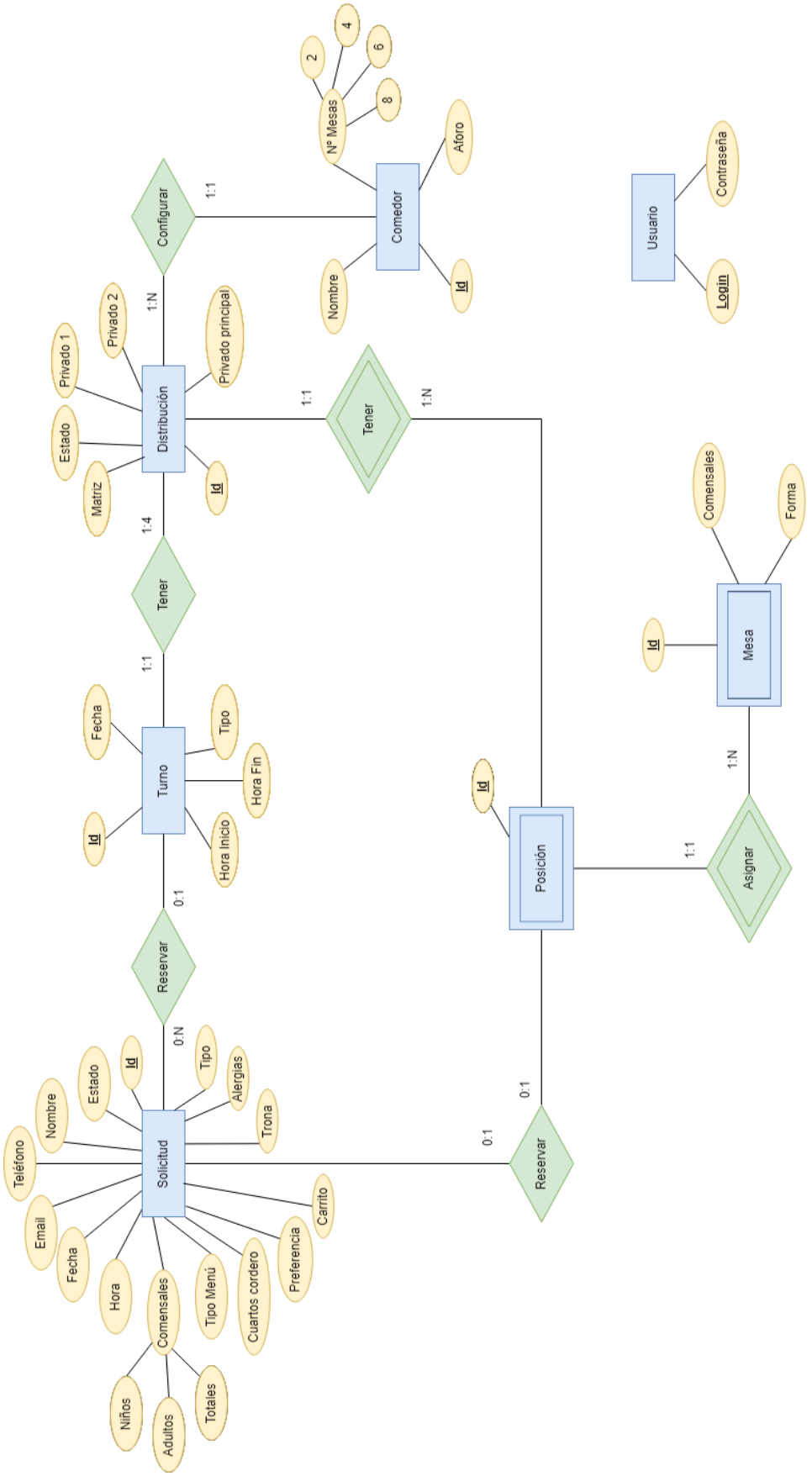


Figura 9 - Modelo Entidad - Relación

DICCIONARIO DE DATOS

Una vez definidos y representados, el último paso es analizar detalladamente estos requisitos en base al modelo entidad-relación. Cada entidad está definida por una serie de atributos que, en este apartado, son identificados, analizados y referenciados a la propia entidad de manera única.

ENTIDADES

Id	E-01	**Nombre**	**Solicitud**		
Descripción	Representa todas las reservas que se crearan en el sistema				
Campo	**Tipo**	**Null**	**Descripción**	**Único**	**Key**
Id	INT	✗	Identificador	✓	✓
Estado	[Reservada, Pendiente, Cancelada]	✗	Estado en el que se encuentra la reserva	✗	✗
Nombre	STRING	✗	Nombre de la persona que hizo la reserva	✗	✗
Teléfono	STRING	✗	Teléfono de contacto de los comensales	✗	✗
Email	STRING	✓	Correo de contacto	✗	✗
Fecha	DATE	✗	Fecha de la reserva	✗	✗
Hora	TIME	✗	Hora de la reserva	✗	✗
Comensales	INT	✗	Número de comensales	✗	✗
Niños	INT	✓	Número de niños	✗	✗
Adultos	INT	✓	Número de adultos	✗	✗
Tipo Menú	[Menú, Carta, Menú Cerrado]	✗	Tipo de menú que pedirán en la comida	✗	✗
Cuartos cordero	INT	✓	Número de cuartos de cordero que reservan	✗	✗
Preferencia	[Interior, Terraza]	✓	Si prefieren hacer la reserva en la terraza o dentro del establecimiento	✗	✗
Carrito	BOOLEAN	✓	Si traen carrito para un bebé	✗	✗
Trona	BOOLEAN	✓	Si necesitan trona	✗	✗
Alergias	STRING	✓	Alergias y comentarios	✗	✗
Tipo	[Normal, Privada]	✗	Si la reserva es privada o no	✗	✗

Tabla 58 - E-01 Solicitud

Id	E-02		Nombre	Turno	
Descripción	Representan los periodos de tiempo en los que dan las diferentes comidas				
Campo	Tipo	Null	Descripción	Único	Key
Id	STRING	✗	Identificador del turno	✓	✓
Fecha	DATE	✗	Fecha del turno	✗	✗
Hora inicio	TIME	✗	Hora de comienzo	✗	✗
Hora fin	TIME	✗	Hora de fin	✗	✗
Tipo	[Comida, Cena]	✗	Si el turno es de comida o de cena	✗	✗

Tabla 59 - E-02 Turno

Id	E-03		Nombre	Distribución	
Descripción	Representa el estado de cada comedor dentro de un turno				
Campo	Tipo	Null	Descripción	Único	Key
Id	INT	✗	Identificador	✓	✓
Matriz	STRING	✗	JSON que representa la distribución espacial de las mesas en forma de matriz	✗	✗
Estado	[Abierta, Cerrada]	✗	Indica si el comedor está abierto o cerrado	✗	✗
Privado 1	INT	✗	Indica si tiene privado 1, y en el caso de tenerlo si está abierto o cerrado.	✗	✗
Privado 2	INT	✗	Indica si tiene privado 2, y en el caso de tenerlo si está abierto o cerrado.	✗	✗
Privado principal	INT	✗	Indica si tiene privado principal, y en el caso de tenerlo si está abierto o cerrado.	✗	✗

Tabla 60 - E-03 Distribución

Id	E-04		Nombre	Comedor	
Descripción	Instancia de cada uno de los comedores				
Campo	Tipo	Null	Descripción	Único	Key
Id	INT	✗	Identificador	✓	✓
Aforo	INT	✗	Número máximo de comensales que pueden entrar	✗	✗
Nombre	STRING	✗	Nombre del comedor	✗	✗
Mesas de 2	INT	✗	Número de mesas de 2 comensales com las que cuenta el comedor	✗	✗
Mesas de 4	INT	✗	Número de mesas de 4 comensales com las que cuenta el comedor	✗	✗
Mesas de 6	INT	✗	Número de mesas de 6 comensales com las que cuenta el comedor	✗	✗
Mesas de 8	INT	✗	Número de mesas de 8 comensales com las que cuenta el comedor	✗	✗

Tabla 61 - E-04 Comedor

Id	E-05	Nombre	Posición		
Descripción	Representa la posición en la que se encuentran las mesas dentro de una distribución				
Campo	**Tipo**	**Null**	**Descripción**	**Único**	**Key**
Id	INT	✗	Identificador		✓

Tabla 62 - E-05 Posición

Id	E-06	Nombre	Mesa		
Descripción	Representa cada una de las mesas de una posición de la distribución				
Campo	**Tipo**	**Null**	**Descripción**	**Único**	**Key**
Id	INT	✗	Identificador	✗	✓
Comensales	INT	✗	Numero de comensales	✗	✗
Forma	[Square, Circle]	✗	Indica el tipo de mesa es, si redonda o cuadrada	✗	✗

Tabla 63 - E-06 Mesa

Id	E-07	Nombre	Usuario		
Descripción	Representa los datos del usuario parfa acceder a la aplicación				
Campo	**Tipo**	**Null**	**Descripción**	**Único**	**Key**
Id	INT	✗	Identificador	✗	✓
Contraseña	MD5(STRING)	✗	Contraseña hasheada	✗	✗

Tabla 64 - E-07 Usuario

RELACIONES

Id	R-01	Nombre	Reservar 1	
Descripción	Representa a que turno pertenece una reserva			
Entidad	**Nombre Entidad**	**Participación**	**Cardinalidad**	
E-01	Solicitud	0	1	
E-02	Turno	0	N	

Tabla 65 - R-01 Reservar 1

Id	R-02	Nombre	Reservar 2	
Descripción	Representa en que posición se colocará la reserva			
Entidad	**Nombre Entidad**	**Participación**	**Cardinalidad**	
E-01	Solicitud	0	1	
E-05	Posición	0	1	

Tabla 66 - R-02 Reservar 2

Id	R-03	Nombre	Tener 1	
Descripción	Representa cada una de los estados de los comedores para un turno			
Entidad	**Nombre Entidad**	**Participación**	**Cardinalidad**	
E-02	Turno	1	4	
E-03	Distribución	1	1	

Tabla 67 - R-03 Tener 1

Id	R-04	Nombre	Tener 2
Descripción	Representa las posiciones en las que las mesas están colocadas en cada distribución		
Entidad	Nombre Entidad	Participación	Cardinalidad
E-03	Distribución	1	N
E-05	Posición	1	1

Tabla 68 - R-04 Tener 2

Id	R-05	Nombre	Configurar
Descripción	Representa el estado de un comedor dentro de un turno		
Entidad	Nombre Entidad	Participación	Cardinalidad
E-03	Distribución	1	1
E-04	Comedor	1	N

Tabla 69 - R-05 Configurar

Id	R-06	Nombre	Asignar
Descripción	Representa las mesas asignadas a cada posición de una distribución		
Entidad	Nombre Entidad	Participación	Cardinalidad
E-05	Posición	1	N
E-06	Mesa	1	1

Tabla 70 - R-06 Asignar

CAPITULO 5. DISEÑO DEL SISTEMA

ARQUITECTURA DEL SISTEMA

ARQUITECTURA LÓGICA

La arquitectura lógica se define como la representación de los componentes lógicos que dan forma a un sistema junto con la relación de estos entre sí, es decir, cómo interactúan unos con otros dentro de la aplicación. La arquitectura lógica correspondiente a este sistema es la siguiente:

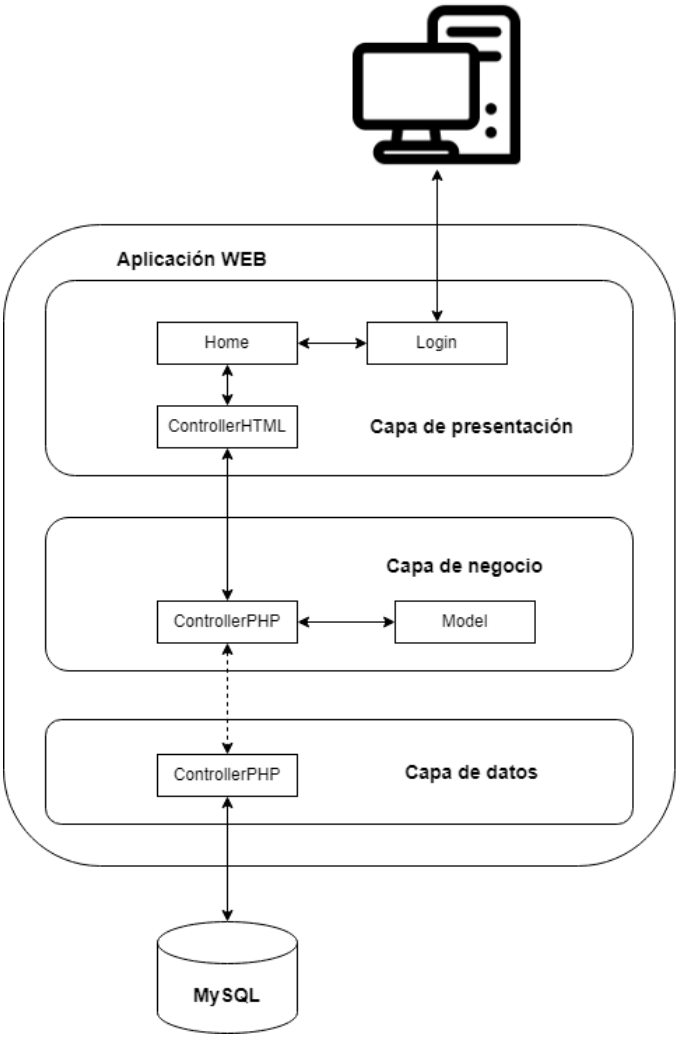


Figura 10 - Arquitectura lógica

Como podemos observar, la arquitectura lógica de la aplicación web se divide en 3 capas:

- **Capa de presentación:** formada por las vistas con las que el usuario interactúa y el controlador JavaScript que las da funcionalidad y hará de enlace con la capa de negocio Aquí también estarían incluidos en las vistas sus respectivos archivos CSS.

- **Capa de negocio:** incluye toda la lógica de negocio que se alberga en el backend de la aplicación utilizando el lenguaje PHP. Es necesaria para interpretar y realizar operaciones sobre los datos que se envían desde la capa de presentación y se recuperan de la capa de datos.
- **Capa de datos:** encargada de conectar la capa de negocio con la base de datos.

ARQUITECTURA FÍSICA

La arquitectura física trata de describir la estructura o la forma en la que se encuentra implementada la tecnología utilizada, es decir, define los elementos físicos que requiere la arquitectura lógica para lograr cumplir con sus objetivos como aparece reflejado:

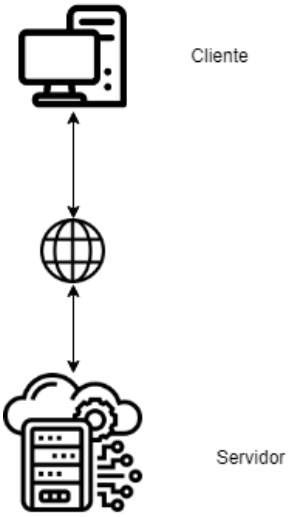


Figura 11 - Arquitectura física

En esta aplicación, la arquitectura física queda definida por un patrón cliente-servidor. El cliente estará definido por el dispositivo utilizado por el usuario. En el lado del servidor se encontrarán todos los archivos HTML, PHP y JS que se enviarán al cliente.

MODELO DE DISEÑO

Para adquirir un conocimiento exhaustivo del diseño del sistema, se emplearán un conjunto de herramientas representativas conocidas como modelos, los cuales se encuentran especificados en las secciones siguientes.

DIAGRAMA MODELO-VISTA-CONTROLADOR

Cómo se ha podido analizar en el apartado anterior, se puede concluir que la aplicación sigue un patrón MVC (Modelo-Vista-Controlador):

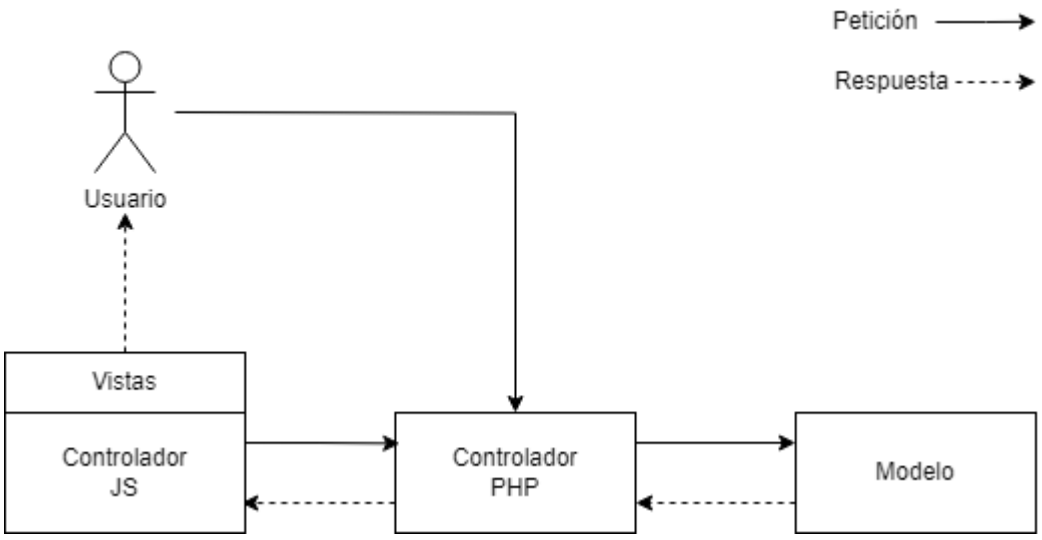


Figura 12 - Modelo Vista Controlar

DIAGRAMA DE SECUENCIA

Este tipo de diagrama tiene como objetivo mostrar las transacciones de comunicación entre los diversos elementos que participan ken una situación o conjunto específico de situaciones.

La representación de estos elementos se realiza a través de un rectángulo y una línea vertical de tiempo para cada uno de ellos. Una vez definidos, se ordenan horizontalmente y las transacciones se representan mediante líneas horizontales que conectan cada una de las líneas temporales de los elementos involucrados.

En este caso se ha decidido incluir una serie de diagramas que intentan englobar y representar la funcionalidad más relevante para la aplicación:

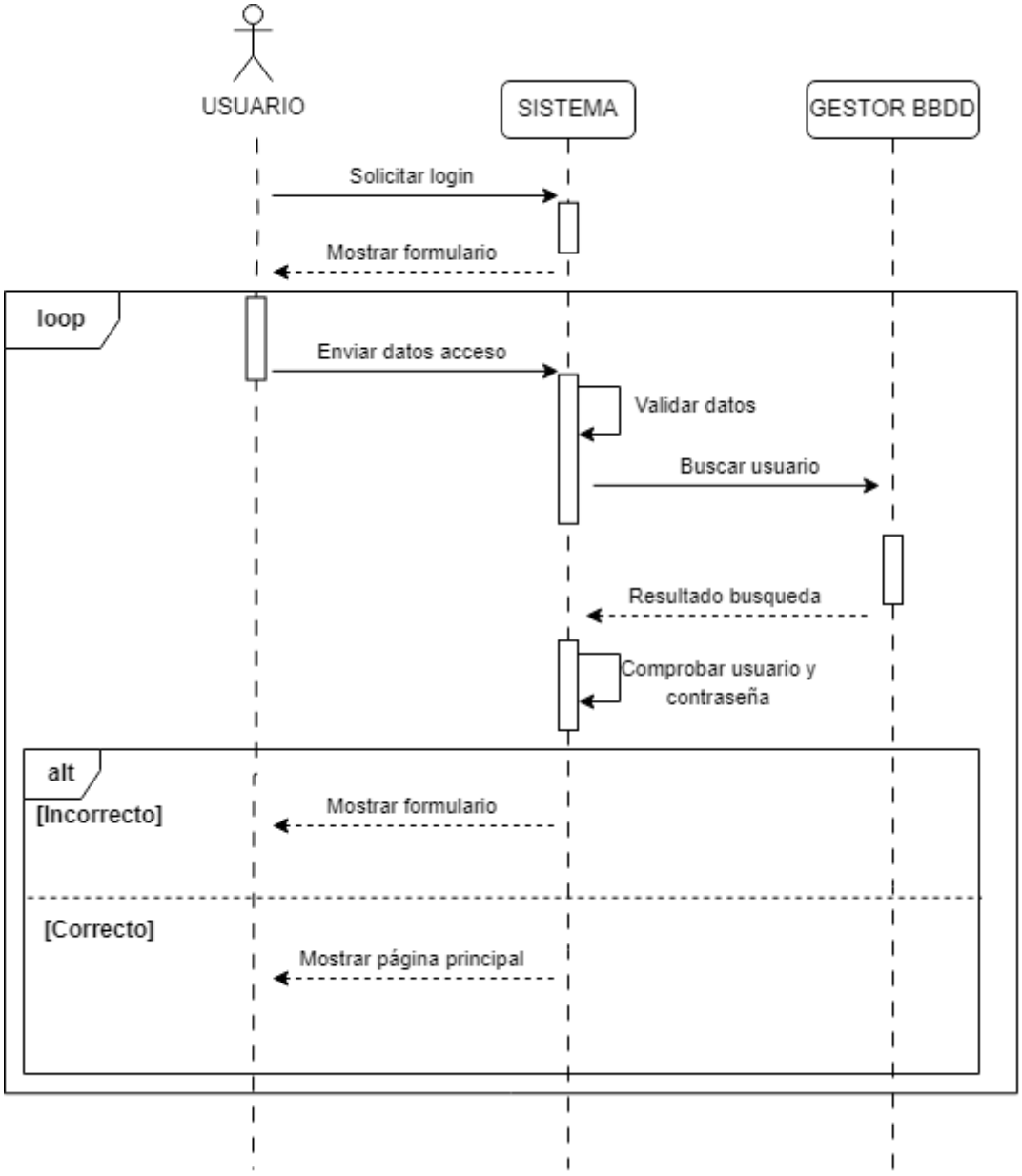


Figura 13 - Diagrama de secuencia 1

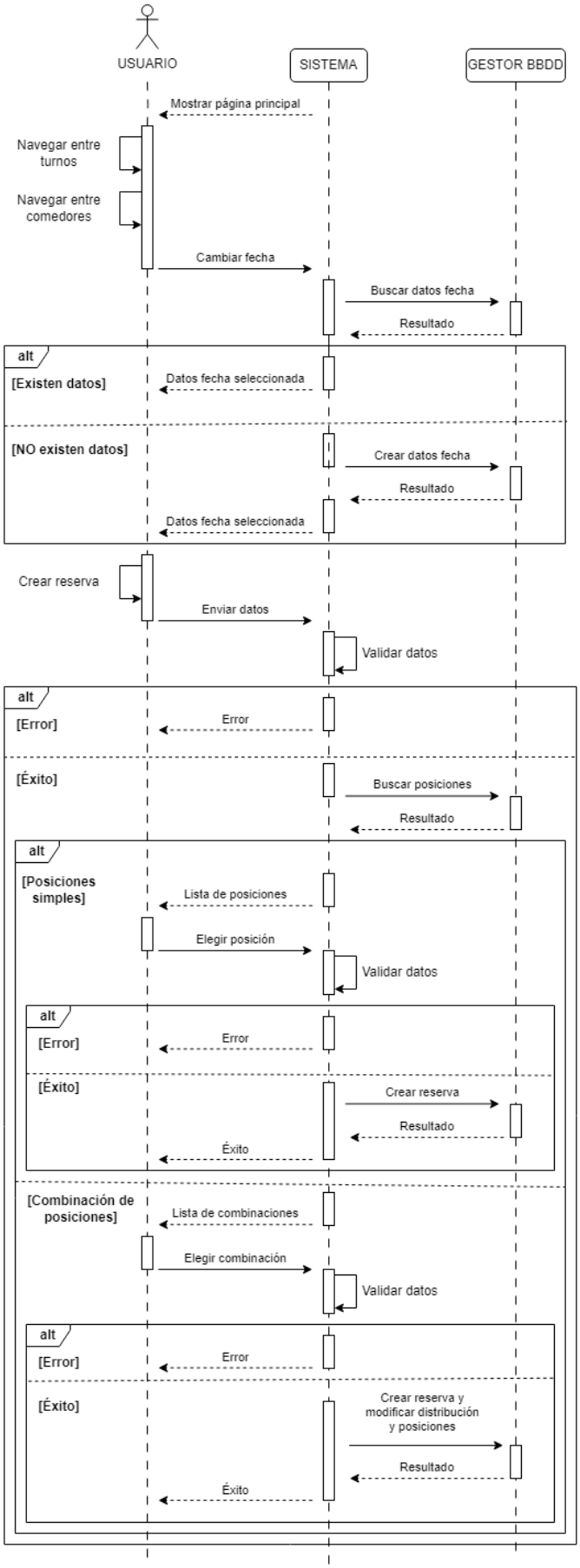


Figura 14 - Diagrama de secuencia 2

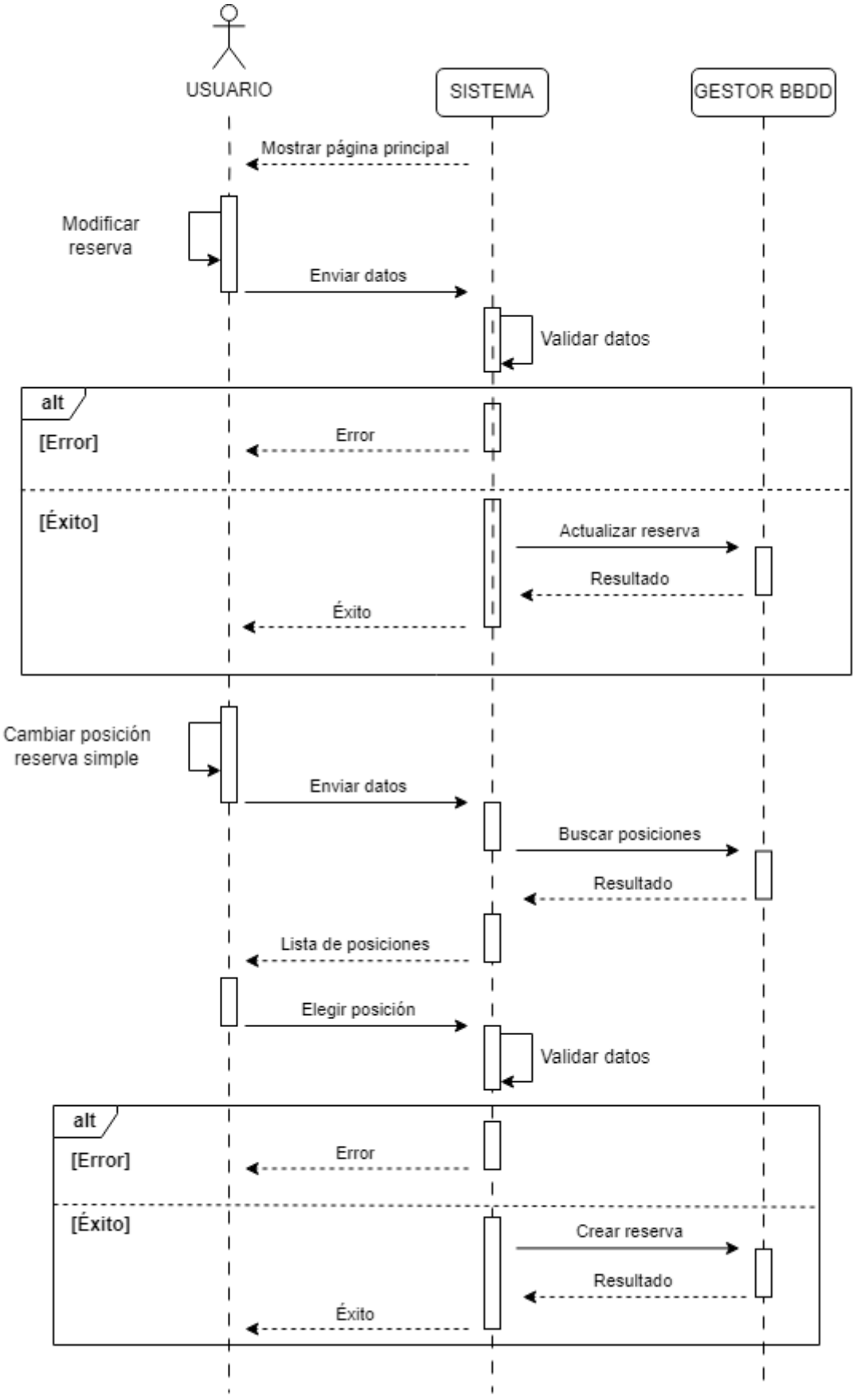


Figura 15 - Diagrama de secuencia 3

MODELO LÓGICO DE DATOS

En este apartado mostramos el modelo lógico de datos que surge de la síntesis del modelo Entidad-Relación en el apartado de requisitos de información (Figura 9).

TURNO (**Id**, fecha, Hora Inicio, Hora Fin, tipo)

COMEDOR (**Id**, Nombre, Aforo, Mesas de 2, Mesas de 4, Mesas de 6, Mesas de 8)

DISTRIBUCIÓN(**Id**, Matriz, Estado, Privado 1, Privado 2, Privado Principal, Turno, Comedor)

FK: Turno → TURNO(Id)

FK: Comedor → COMEDOR(Id)

POSICIÓN (**Id**, **Distro**)

FK: Distro → DISTRO(Id)

MESA (**Id**, Comensales, Forma, **Posición**, **Distro**)

FK: (Posición, Distro) → POSICIÓN(Id, Distro)

SOLICITUD (**Id**, Estado, Fecha, Hora, Tipo, Nombre, Email, Teléfono, Comensales, Adultos, Niños, Tipo de menú, Cuartos de cordero, Preferencia, Carrito, Trona, Alergias, Turno, Posición, Distro)

FK: Turno → TURNO(Id)

FK: (Posición, Distro) → POSICIÓN(Id, Distro)

USUARIO (**Login**, Contraseña)

Al no haber relaciones N:N, las entidades se transforman en tablas y las relaciones se representan a través de *Foreign Keys*.

Este modelo será utilizado posteriormente creando una clase por cada una de las tablas para luego crear instancias de las mismas y poder gestionar fácilmente los datos de la base de datos.

DIAGRAMA DE CLASES

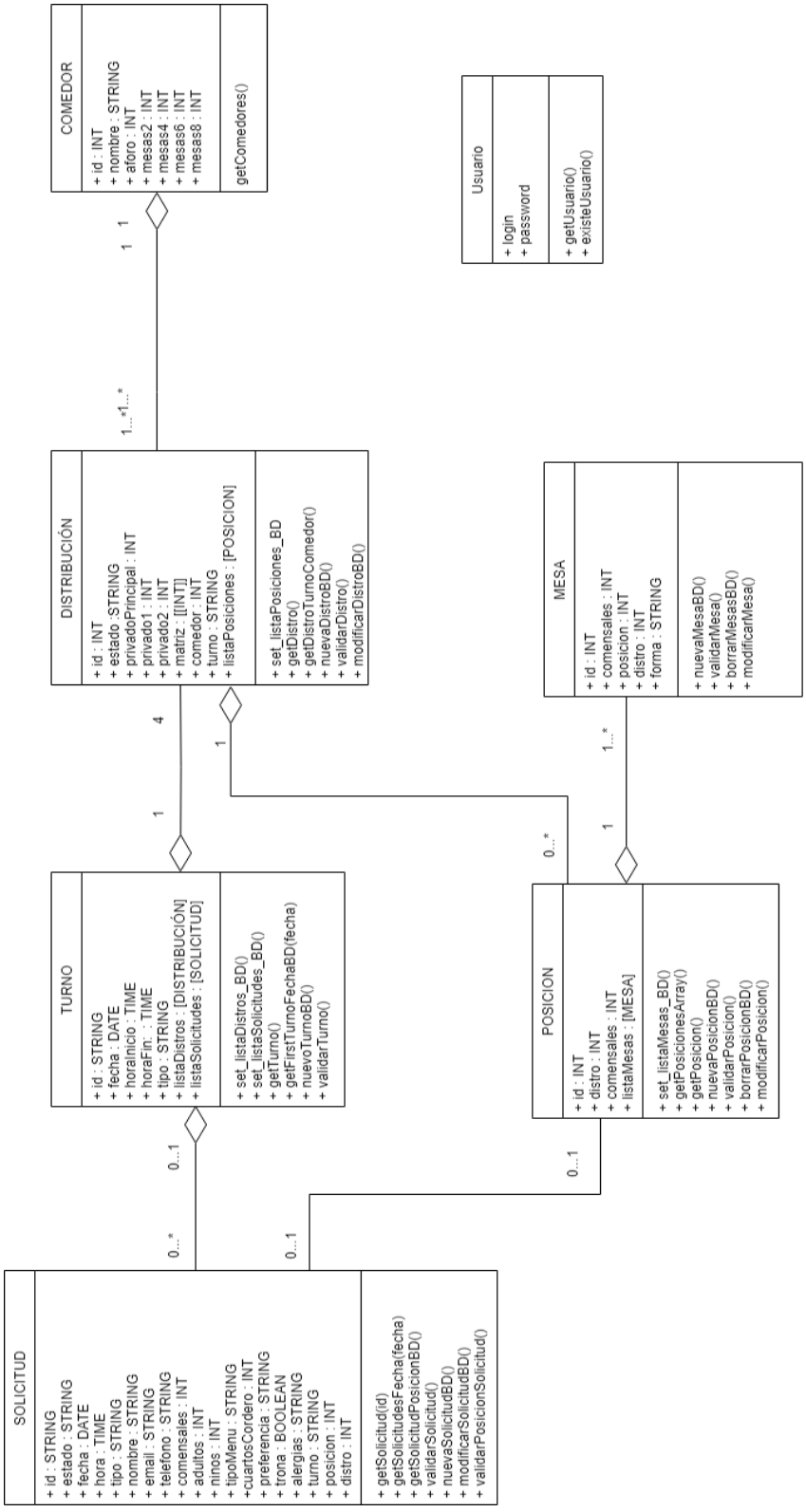


Figura 16 - Diagrama de clases

DISEÑO DE LA INTERFAZ DE USUARIO

Dado que la aplicación está destinada a funcionar como una *Single Page Aplicattion* (SPA), esta es una aplicación web que carga una única página HTML y luego actualiza dinámicamente el contenido en esa página sin tener que recargarla por completo. Esto se logra mediante el uso de JavaScript para manipular y actualizar la interfaz de usuario.

Algunas ventajas de las aplicaciones SPA son:

- **Experiencia de usuario más rápida:** al cargar solo una vez la página inicial y actualizar el contenido de forma dinámica, las aplicaciones SPA ofrecen una experiencia de usuario más rápida y fluida. No hay tiempos de carga entre diferentes páginas, lo que mejora la percepción del rendimiento.
- **Interactividad:** las SPAs permiten una interacción fluida y en tiempo real con los usuarios. Los cambios de contenido se pueden realizar de forma instantánea sin tener que cargar una página nueva, lo que proporciona una sensación más rápida y fluida al interactuar con la aplicación.
- **Menor consumo de ancho de banda:** al cargar solo los datos necesarios y actualizar el contenido de manera incremental, las SPAs reducen la cantidad de datos transferidos entre el cliente y el servidor. Esto puede resultar en un menor consumo de ancho de banda, especialmente en conexiones más lentas o dispositivos móviles.
- **Separación de responsabilidades:** las SPAs siguen una arquitectura de cliente-servidor en la que gran parte del procesamiento y la lógica de presentación se realiza en el cliente. Esto permite una mejor separación de responsabilidades entre el cliente y el servidor, lo que facilita el mantenimiento y la escalabilidad de la aplicación.
- **Reutilización de código:** al utilizar *frameworks* y bibliotecas JavaScript para el desarrollo de SPAs, es posible reutilizar componentes y módulos de código en diferentes partes de la aplicación. Esto puede acelerar el desarrollo, mejorar la consistencia y facilitar la colaboración en equipos de desarrollo.

Por lo tanto, se divide la interfaz en 2 bloques, el de acceso a la página, y la página principal.

INTERFAZ DE ACCESO

Se trata de un formulario de inicio estándar formado por el nombre de usuario y la contraseña y un botón para acceder.

Figura 17 - Interfaz de acceso

PÁGINA PRINCIPAL

Esta página servirá de centro de control para la aplicación, permitiendo al usuario acceder a toda la funcionalidad de la misma. Para la funcionalidad que no se puede abordar dentro de la página, se utilizarán ventanas emergentes, t6ambién denominadas pop-ups, dentro de la página principal.

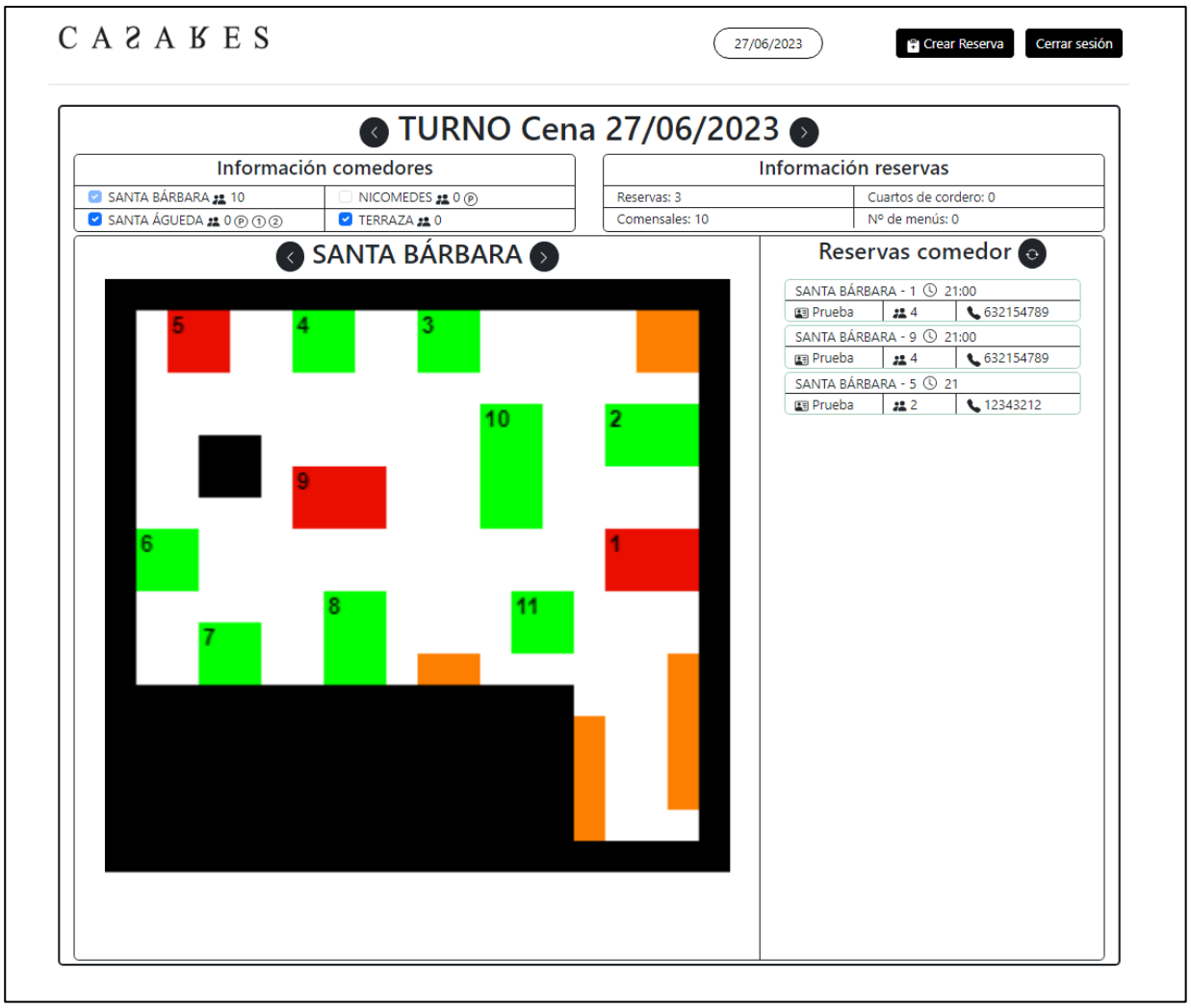


Figura 18 - Interfaz página principal

Para poder explicar la funcionalidad de la página la dividiremos en diferentes apartados.

Menú horizontal

En esta menú encontraremos:

- **Botón crear reservas:** botón que desencadenará un pop-up en el que se lleva a cabo el proceso.
- **Selector de fecha:** permite al usuario cambiar los datos que visualizará.
- **Botón de cierre de sesión**

Figura 19 - Interfaz menú horizontal

Ficha de turno

Este apartado reúne toda la información de cada turno. En él podemos encontrar:

- **Identificador del turno**
- **Botones de cambio de turno**
- **Resumen de comedores:** permite cambiar el estado del comedor siempre que no tenga reservas.
- **Resumen de reservas**
- **Ficha de comedor**

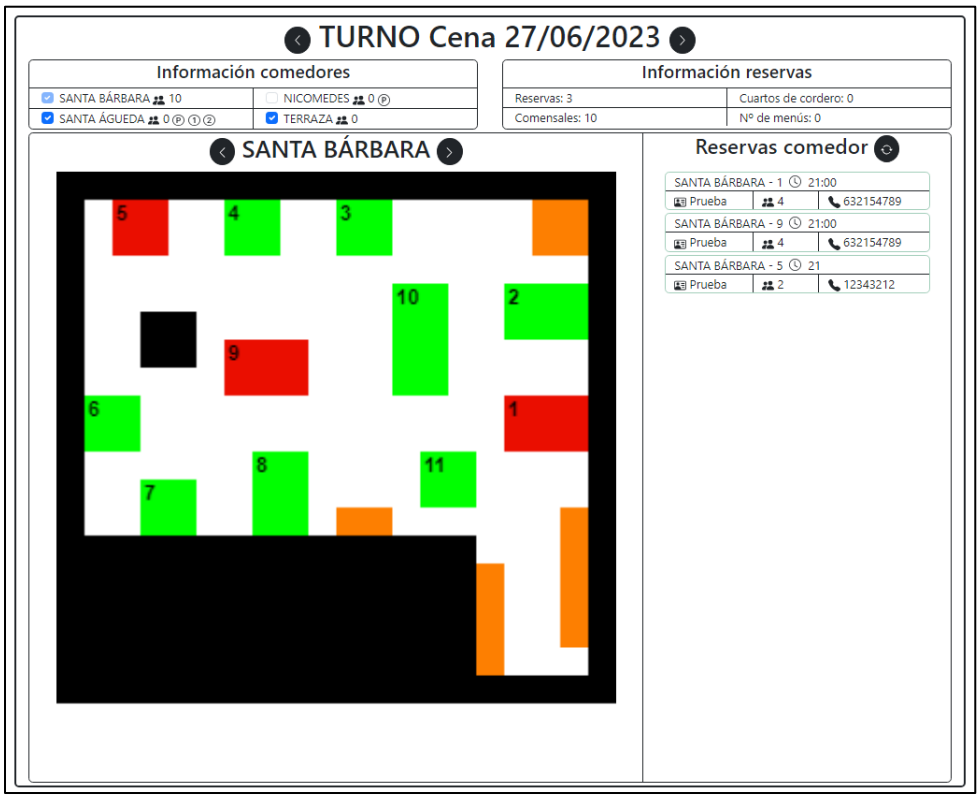


Figura 20 - Interfaz ficha turno

Ficha de comedor

Este apartado reúne toda la información del comedor. En él podemos encontrar:

- **Sección de distribución espacial**
  - **Botones de cambio de comedor**
  - **Canva:** representación gráfica de la distribución espacial de las mesas, diferenciando si están reservadas o no.
- **Sección de lista de reservas**
  - **Botón de cambio de vista:** cambia entre las reservas asignadas al comedor y todas las reservas (Incluyendo las canceladas)
  - **Lista de reservas:** formada por fichas de reserva que incluyen la información más relevante de la reserva y al pulsarla desencadena un pop-up con la información de la reserva.

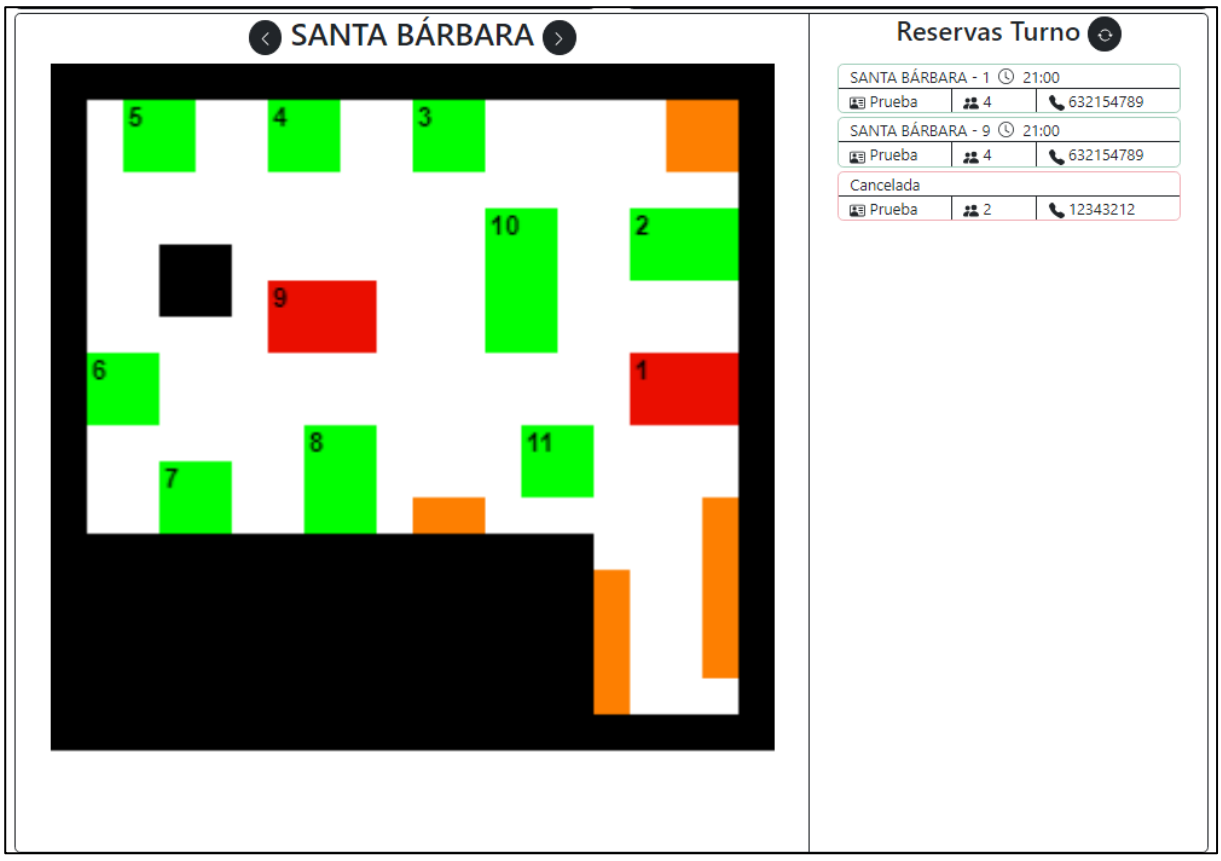


Figura 21 - Interfaz ficha comedor

Pop-up formulario creación de reserva

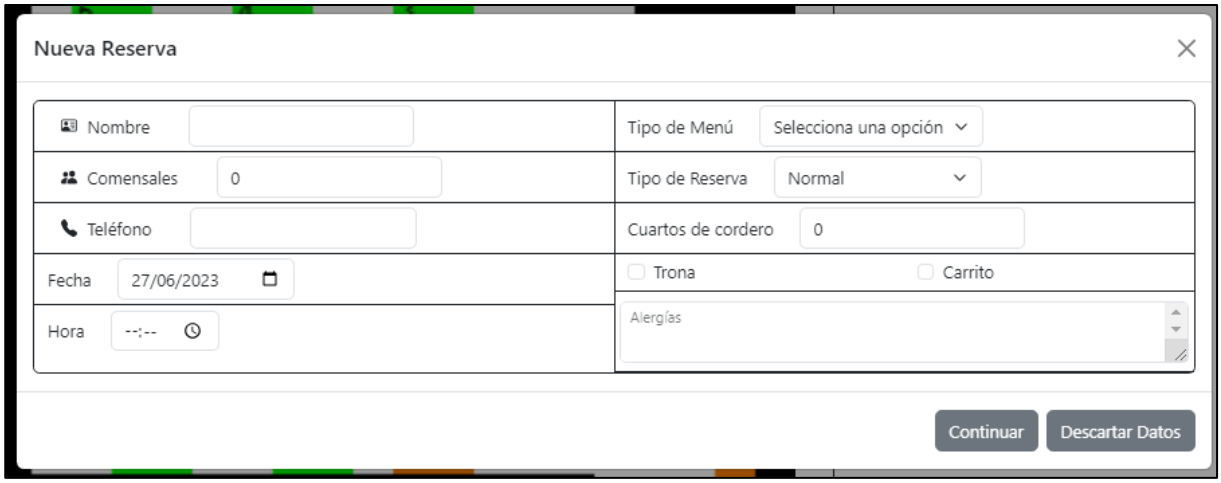


Figura 22 - Interfaz pop-up formulario de creación reserva

Pop-up selector de mesa simple

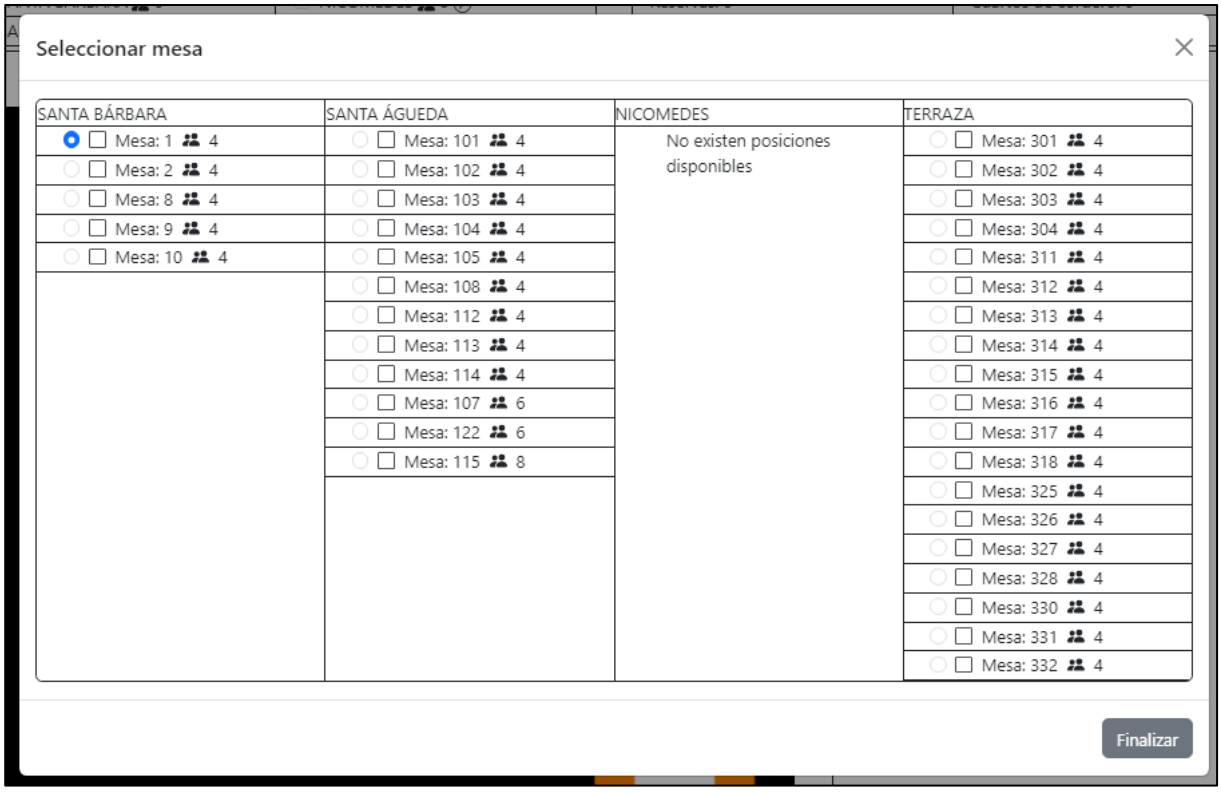


Figura 23 - Interfaz pop-up selector mesa simple

Pop-up selector de combinación de mesas

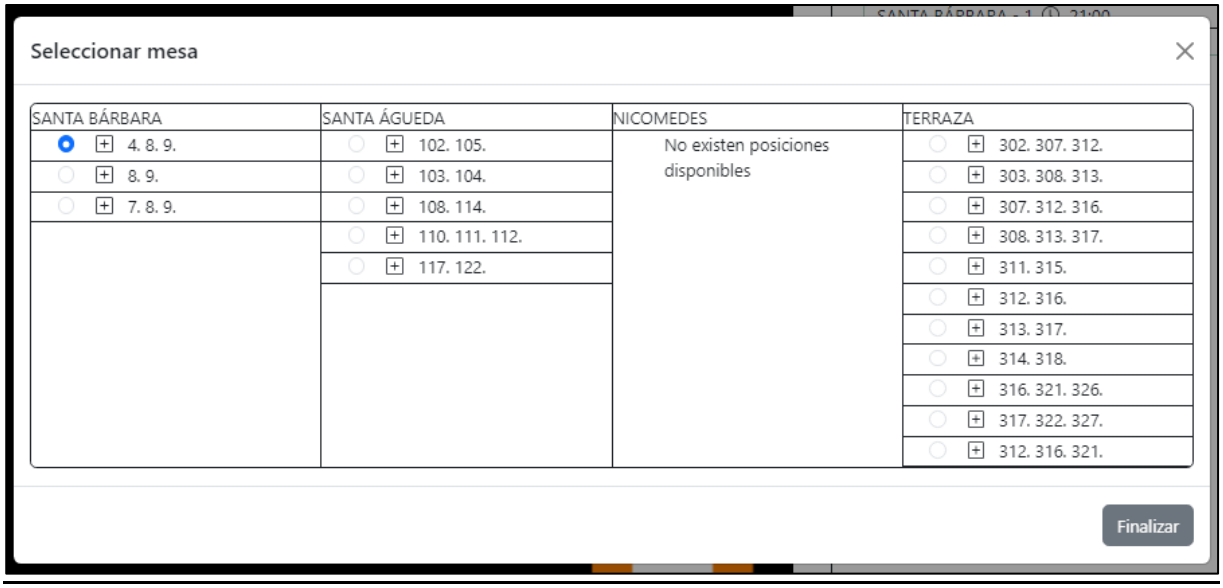


Figura 24 - - Interfaz pop-up selector combinación de mesas

Pop-up información de reserva

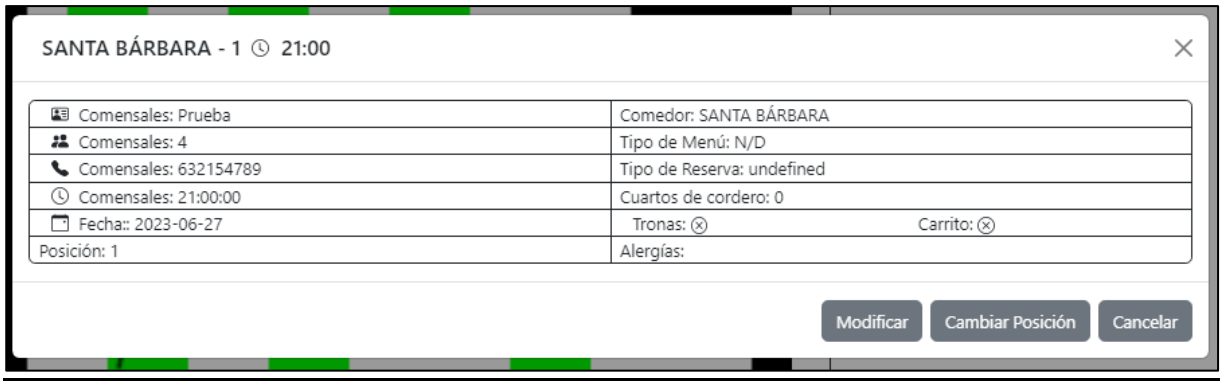


Figura 25 - Interfaz pop-up detalle de reserva

Pop-up formulario modificación reserva

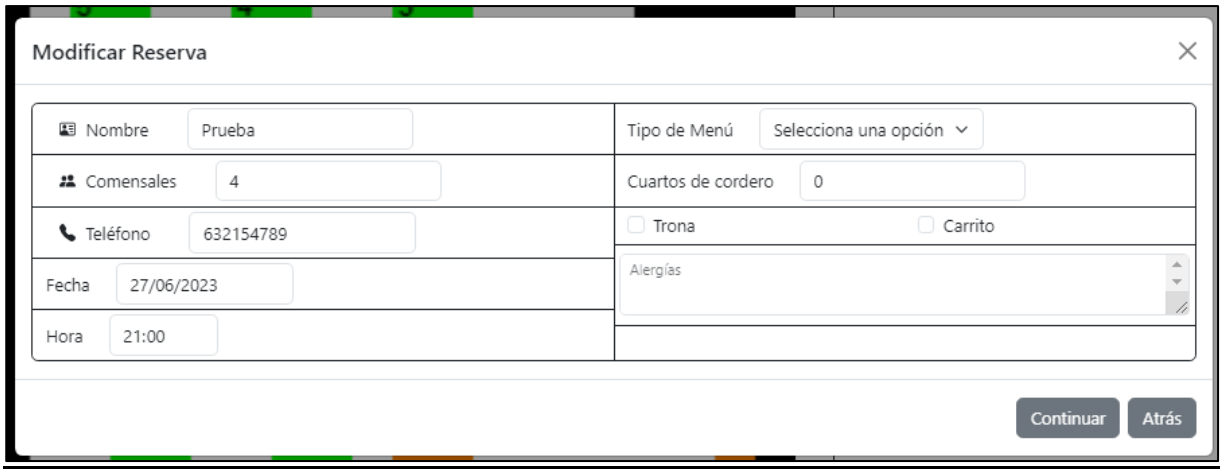


Figura 26 - Interfaz pop-up formulario de modificación de reserva

CAPITULO 6. IMPLEMENTACIÓN

ESTRUCTURA INTERNA DEL PROYECTO

En esta sección se examina la estructura interna del proyecto. Se define como estructura interna a la organización de los archivos en carpetas y paquetes, de modo que facilite al desarrollador un mejor rendimiento y fluidez al crear nuevas funcionalidades o llevar a cabo tareas de mantenimiento en la aplicación.

La estructura del proyecto consta de:

- `Index.php`

Archivo inicial desde el que arrancará el proceso de *login*.

- `View/`

Carpeta que almacena la vista de la página principal de la aplicación. En el caso de que la aplicación crezca y utilice diferentes vistas y funcionalidades, aquí se almacenarán las diferentes vistas de la aplicación web.

En ella, actualmente, sólo se encuentra la vista **`home.php`**.

- `Model/`

En esta carpeta se almacenan diferentes archivos PHP los cuales hacen referencia a cada una de las entidades de la base de datos. En cada uno de ellos se encuentra la clase que las representan con sus atributos y métodos.

Nombre	Descripción del archivo
`Comedor.php`	Clase referente a la entidad Comedor
`Distribucion.php`	Clase referente a la entidad Distribución
`Mesa.php`	Clase referente a la entidad Mesa
`Posicion.php`	Clase referente a la entidad Posición
`Solicitud.php`	Clase referente a la entidad Solicitud
`Turno.php`	Clase referente a la entidad Turno
`Usuario.php`	Clase referente a la entidad usuario

Tabla 71 - Archivos de la carpeta Model

- Controller/

Dónde se almacenan diferentes archivos que sirven de controlador en la parte *Back-End* de la aplicación.

Nombre	Descripción del archivo
controller.php	Enlace principal entre el controlador JS del Front-End y el Back-End. Se encarga de diferenciar las diferentes acciones de la llamada Ajax.
handler.php	Funcionalidad de la aplicación y sus métodos son llamados desde el controller.php
loginController.php	Controlador de inicio de sesión.
logoutController.php	Controlador de cierre de sesión.
reservus.php	Funcionalidad relacionada con la devolución de mesas y combinaciones de mesas posibles para las reservas.

Tabla 72 - Archivos de la carpeta Controller

- JS/

Carpeta que almacena los diferentes archivos JS que dan dinamismo a la página principal de la SPA.

Nombre	Descripción del archivo
banner.js	Funcionalidad de cambio entre las fichas de turno y de comedor.
canva.js	Funcionalidad que representa dentro del canva la distribución espacial de los comedores.
htmlController.js	Controlador de la página que se ejecuta en el lado del cliente. Se encarga de la generación y actualización dinámica del HTML y de las llamadas Ajax que se realizan con el Back-End.

Tabla 73 - Archivos de la carpeta JS

- CSS/

Dónde se almacenan las hojas de estilo para el HTML: **login.css** y **style.css**.

- img/

En esta carpeta se almacenan las imágenes que se vayan a utilizar en la aplicación.

- includes/

En esta carpeta se almacenan otras funcionalidades que necesita la aplicación.

Nombre	Descripción del archivo
conexionBD.php	Crea la conexión con la base de datos
include.php	Permite incluir todo lo necesario a cada uno de los archivos PHP
utils.php	Funcionalidad adicional

Tabla 74 - Archivos de la carpeta Includes

DETALLES DE IMPLEMENTACIÓN

Como se ha indicado en el apartado 3, el proyecto se ha llevado a cabo utilizando un modelo de desarrollo iterativo-incremental. En él, la primera etapa corresponde al análisis inicial de objetivos y requisitos, y las etapas consecutivas se corresponden con el desarrollo de la aplicación web.

En este apartado se explica brevemente cómo se ha desarrollado la funcionalidad de cada una de estas etapas.

ETAPAS DE CREACIÓN DE LA ESTRUCTURA PRINCIPAL Y VISUALIZACIÓN DE LOS DATOS

Aquí analizaremos la implementación de las primeras etapas del proyecto:

1. Etapa de análisis global de los objetivos y requisitos del proyecto.
2. Creación de la estructura de la página principal.
3. Creación y población de la base de datos.
4. Visualización de los datos en la página principal.
5. Funcionalidad básica de la página principal.
6. Automatización de creación de datos por fecha.

Estas etapas incluyen la instalación de XAMPP y la creación de la carpeta donde se guardan todos los archivos del proyecto.

Una vez creado el proyecto, se ha añadido un archivo `index.php` para crear la estructura HTML inicial con el fin de poder visualizar el espacio que ocupa la información e ir situando los diferentes botones y fichas de información.

```
<body>

    <header>
        <div id="logo">
            <img src="../img/logo-casares-negro.png" style="width:100%">
        </div>
        <nav id="navMenu">
            <ul>  You, 5 months ago • Init ...
                <li><a href="../Controller/logoutController.php">Cerrar sesión</a></li>
            </ul>
        </nav>
    </header>

    <div class="loading"></div>

    <div class="toast-container"></div>

    <div class="popup">
        <div class="popup-box">
            <button onclick="hidePopUp()"> <img class="boton" src="../img/cross.png" class="boton-cruz"> </button>
            <div class="popup-content">
                <div class="popup-info"></div>
                <div class="popup-formulario"></div>
                <div class="popup-posiciones"></div>
                <div class="popup-previsualizar"></div>
                <div class="popup-error"></div>
            </div>
        </div>
    </div>

    <div class="popup2">
        <div class="popup2-box"></div>
    </div>



    <div class="main">
        <div class="flex-container-row topSection"></div>
        <div class="banner"></div>
    </div>

</body>
```

Figura 27 - Código HTML de la estructura principal

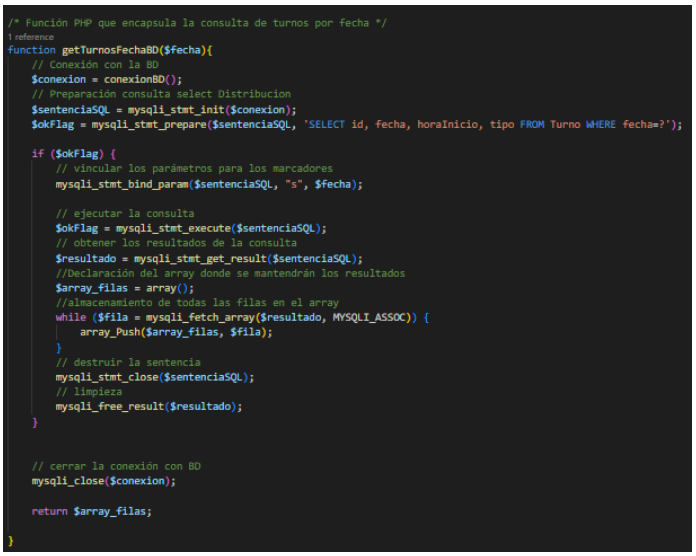
Posteriormente, se genera la base de datos en el servidor local Apache de XAMPP creando la estructura de datos necesaria para el funcionamiento de la aplicación web.

	Tabla	Acción						
☐	comedor	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	distribucion	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	mesa	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	posicion	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	solicitud	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	turno	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar
☐	usuario	☆	Examinar	Estructura	Buscar	Insertar	Vaciar	Eliminar

Figura 28 - Tablas base de datos MySQL

A continuación, se crea la carpeta `Model/` y todos los archivos que representan cada una de las tablas de la base de datos, para posteriormente, poder volcar y gestionar la información de la base de datos a la aplicación.

Una vez establecida la estructura principal de la página y la base de datos, se procede a poblar brevemente la aplicación con datos de prueba y crear los métodos para obtener las filas de las tablas de la base de datos, y así poder tener una vista de la estructura en la que se refleja la información en la web. Podemos ver un ejemplo con la obtención de turnos en la siguientes figuras.



```php
/* Función PHP que encapsula la consulta de turnos por fecha */
1 reference
function getTurnosFechaBD($fecha){
    // Conexión con la BD
    $conexion = conexionBD();
    // Preparación consulta select Distribucion
    $sentenciaSQL = mysqli_stmt_init($conexion);
    $okFlag = mysqli_stmt_prepare($sentenciaSQL, 'SELECT id, fecha, horaInicio, tipo FROM Turno WHERE fecha=?');

    if ($okFlag) {
        // vincular los parámetros para los marcadores
        mysqli_stmt_bind_param($sentenciaSQL, "s", $fecha);

        // ejecutar la consulta
        $okFlag = mysqli_stmt_execute($sentenciaSQL);
        // obtener los resultados de la consulta
        $resultado = mysqli_stmt_get_result($sentenciaSQL);
        //Declaración del array donde se mantendrán los resultados
        $array_filas = array();
        //almacenamiento de todas las filas en el array
        while ($fila = mysqli_fetch_array($resultado, MYSQLI_ASSOC)) {
            array_Push($array_filas, $fila);
        }
        // destruir la sentencia
        mysqli_stmt_close($sentenciaSQL);
        // limpieza
        mysqli_free_result($resultado);
    }


    // cerrar la conexión con BD
    mysqli_close($conexion);

    return $array_filas;
}
```

Figura 29 - Código PHP de obtención de datos de BD 1

```php
2 references
function getTurnosFecha($fecha){
    $contexto =  [
        'exito' => false,
        'mensajeError' => ''
    ];

    // Consultar valores desde la BD
    $array = getTurnosFechaBD($fecha);

    if (is_bool($array) || (is_array($array) && empty($array))) {
        $contexto['mensajeError'] = 'No se han obtenido turnos';
    }
    else{
        $listaTurnos = array();
        if (count($array) > 0) {                    You, 1 second ago • Uncommitted changes
            foreach ($array as $fila) {
                $turno = new Turno($fila["id"], $fila["fecha"], $fila["horaInicio"], $fila["tipo"]);

                $turno->set_listaDistros_BD();
                $turno->set_listaSolicitudes_BD();

                if($turno->listaDistros != null){
                    $numDistros = count($turno->listaDistros);
                    for($i=0; $i<$numDistros; $i++){
                        $turno->listaDistros[$i]->set_listaPosiciones_BD();
                    }
                }
                array_push($listaTurnos, $turno);
            }
            $contexto['exito'] = true;
            $contexto['listaTurnos'] = $listaTurnos;
        }
    }
    return $contexto;
}
```

Figura 30 - Código PHP de llamada al método de obtención de datos

Una vez se ha creado la estructura básica y se puede poblar de datos, se generan los diferentes archivos de las carpeta JS:

- **`banner.js`:** funcionalidad que permite cambiar entre las diferentes fichas tanto de turno como de comedor.
- **`canva.js`:** funcionalidad que permite pintar en pantalla sobre un elemento *canva (*etiqueta del elemento que representará el plano del comedor*)* la distribución espacial del comedor.
- **`htmlController.js`:** funcionalidad que permite la creación dinámica de HTML sobre los elementos de la estructura básica creada con anterioridad.

Este último archivo irá creciendo con el desarrollo de la aplicación. En estas etapas, se implementa principalmente lo relacionado con la generación dinámica de HTML. En las figuras que aparecen a continuación podemos ver los métodos que logran dicha generación.

```
//?-------------------------------------------------------------------------------------------------------------
//^                    GENERACIÓN HTML
//?-------------------------------------------------------------------------------------------------------------

    //~ Generar estructuras HTML Main

function main() {···
}

function topSection(){···
}

function banner(){···
}

function infoTurno(turno) {···
}

function childBannerDistros(turno, indexTurno){···
}

function childBannerReservas(turno){···
}

function fichasSolicitudes(solicitudes){···
}

function obtenerClaseFichaSolicitud(estado) {···
}


    //~ Genera estructura HTML de la ventana emergente

function viewSolicitudPopUp(id, turnoId){···
}

function modifiedSolicitudPopUp(id, turnoId){···
}

function createSolicitudPopUp(){···
}

function solicitudForm(solicitud){···
}

function selectorPosicion(){···
}

function fichaPosicionSelectorDefault(posicion, indexDistro, indexPosicion, sizePosiciones){···
}
```

Figura 31 - Código JS de generación dinámica de HTML 1

```
    //~ Generación de mensajes TOAST

function createToast(message = null, type = null) {
}


    //~ Otros

function checkHTML(aux){
}

function checkPrivados(aux, p){
}

function getNombreComedorDistro(turnoId, distroId){
}

function translateIdTurno(str) {
}

function translateTipoReserva(str){
}
```

Figura 32 - Código JS de generación dinámica de HTML 2

Así era la primera visualización de la aplicación:

Figura 33 - Visualización inicial de la página principal

Por último, dado que en la aplicación web se muestran todos los datos relacionados con un día concreto, es necesario que al seleccionar una fecha, que no tenga la estructura básica de datos, el sistema cree dicha información y la guarde en la base de datos.

Esta estructura consta de 3 turnos: 2 de comida y 1 de cena; para cada uno de los turnos, 4 distribuciones; y para cada distribución, sus correspondientes mesas y posiciones.

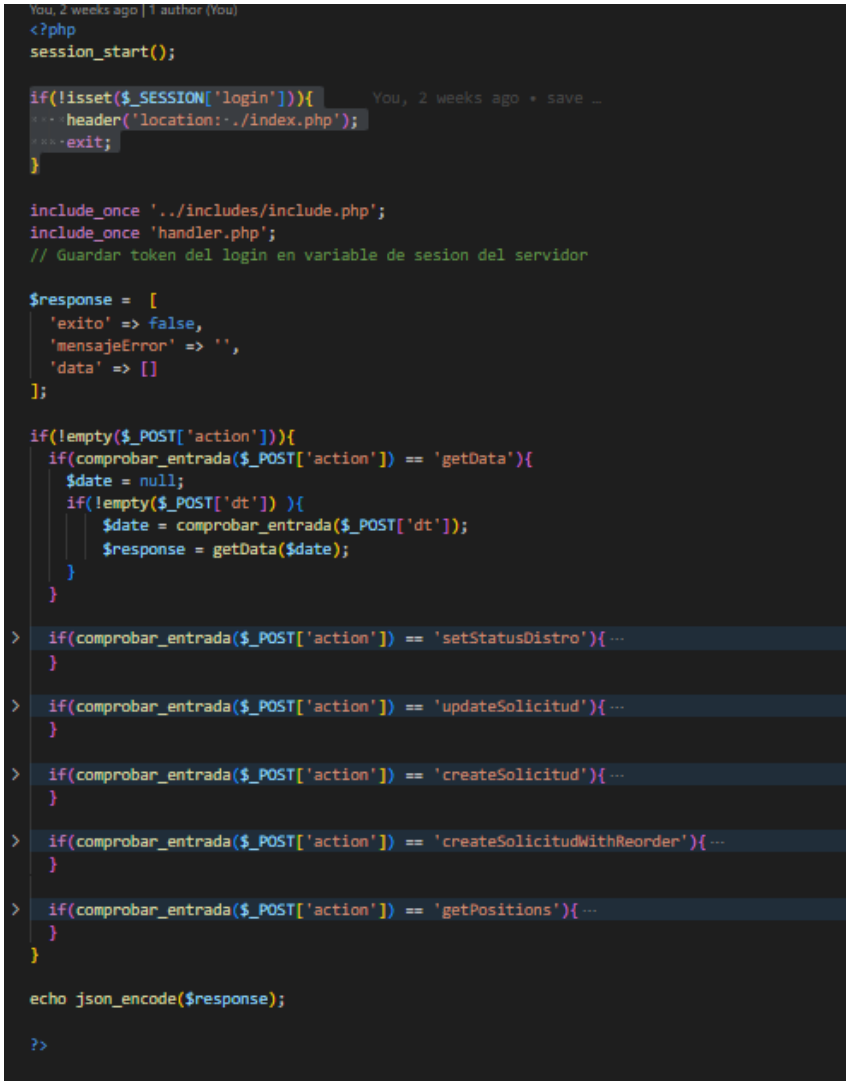
Para gestionar esto y poder recibir los datos en el controlador *Front-End* se crea la primera llamada de Ajax del JS y se crea la carpeta `Controller/` con los archivos `controller.php` y `handler.php` para gestionar el *Back-End*.

En la siguiente imagen se muestra una llamada Ajax para obtener los datos. Cabe destacar que, en este caso, el `controller.php` no devuelve una página HTML, sino que devolverá información en formato JSON, el cual será tratado en el controlador de JS.

```
//?-------------------------------------------------------------------------------------------------------------
//^                    CALL-OUTS
//?-------------------------------------------------------------------------------------------------------------

    //~ Obtención de datos

function changeDate(newDate){
    $('.main').hide();
    loadingInit()
    date = new Date(newDate);
    dateString = date.toISOString().match(/\d{4}-\d{2}-\d{2}/)[0];
    getData()
}

function getData() {
    // Mostrar la animación de carga y ocultar el div principal
    $('.main').hide();
    loadingInit()

    $.ajax({
        type: "POST",
        url: '../Controller/controller.php',
        data: {action: 'getData', dt: dateString},
        success: function(response){
            cons('Response', response); //TODO: Devolver token de sesion al JS y enviarlo con cada llamada
            contexto = JSON.parse(response);

            if(contexto.exito){
                turnosArray = contexto.data.turnos;
                comedoresArray = contexto.data.comedores;
                cons('comedoresArray', comedoresArray);

                // Pintar el contenido del div principal con los datos cargados
                main();

                // Mostrar el div principal y ocultar la animación de carga
                $('.main').show();
                loadingEnd();
            }
        }
    });

}
```

Figura 34 - Código JS de obtención de datos a través de Ajax

En la imagen que aparece a continuación se muestran los métodos para obtener los datos de los archivos PHP mencionados anteriormente.



```php
You, 2 weeks ago | 1 author (You)
<?php
session_start();

if(!isset($_SESSION['login'])){      You, 2 weeks ago • save …
    header('location:./index.php');
    exit;
}

include_once '../includes/include.php';
include_once 'handler.php';
// Guardar token del login en variable de sesion del servidor

$response =  [
  'exito' => false,
  'mensajeError' => '',
  'data' => []
];

if(!empty($_POST['action'])){
  if(comprobar_entrada($_POST['action']) == 'getData'){
    $date = null;
    if(!empty($_POST['dt']) ){
        $date = comprobar_entrada($_POST['dt']);
        $response = getData($date);
    }
  }

  if(comprobar_entrada($_POST['action']) == 'setStatusDistro'){···
  }

  if(comprobar_entrada($_POST['action']) == 'updateSolicitud'){···
  }

  if(comprobar_entrada($_POST['action']) == 'createSolicitud'){···
  }

  if(comprobar_entrada($_POST['action']) == 'createSolicitudWithReorder'){···
  }

  if(comprobar_entrada($_POST['action']) == 'getPositions'){···
  }
}

echo json_encode($response);

?>
```

Figura 35 Código del archivo controller.php

Figura 36 - Función para obtener datos del handler.php

Cabe señalar que muchos de los métodos desarrollados en PHP devuelven un objeto JSON que generalmente tiene la estructura de:

- **`exito`**: que indica si se ha realizado con éxito la función.
- **`mensajeError`**: en caso de error, indica qué lo ha podido producir. Sirve para mostrar errores al usuario final y también resulta muy útil para depurar el código.

El resto de la estructura es variable y suele contener los datos que requiera el método que llame a la función.

GESTIÓN DE COMEDORES DE LOS TURNOS

En esta etapa se implementa la funcionalidad que permite abrir y cerrar los comedores.

Mediante el controlador JS hacemos que las casillas que permiten desencadenar la llamada Ajax estén activas en el caso de que el comedor esté cerrado, o que esté abierto y no haya mesas reservadas para dicho comedor.

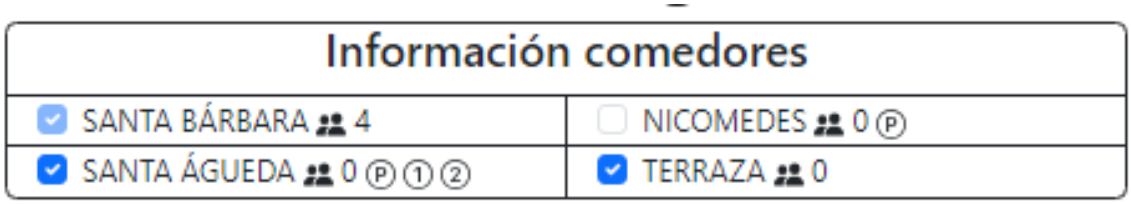


Figura 37 - Ficha información comedores

Posteriormente, al marcar o desmarcar una de estas casillas, se desencadena una llamada Ajax, la cual cambia el estado de dicha distribución en la base de datos.

```
    //~ Creación/Modificación de datos

function changeDistroStatus(idDistro, value, privado){

    let flag = true;
    let json = {
        action: 'setStatusDistro',
        id: idDistro
    };

    switch (privado) {
        case 1:
            json.priv1 = value ? 1 : 0;
            break;
        case 2:
            json.priv2 = value ? 1 : 0;
            break;
        case 0:
            json.status = value ? 'Abierta' : 'Cerrada';
            break;
        default:
            flag = flase;
            break;

    }

    cons('changeDistroStatus', JSON.stringify(json));

    if(flag){
        $.ajax({
            type: "POST",
            url: '../Controller/controller.php',
            data: json,
            success: function(response){
                cons('Response', response);      You, 3 months a
            }
        });
    }

}
```

Figura 38 - Evento JS que lanza llamada para abrir o cerrar comedores

En esta etapa se implementa por primera vez el *pop-up*. Este *pop-up* tiene diferentes elementos que se muestran y ocultan en función de la necesidad. Consta de 3 elementos. El primero es el formulario, que servirá para crear o modificar las reservas. El segundo es el selector de posiciones que nos mostrará una lista de las mesas o combinaciones de mesas en las que podrá colocar la reserva. Por último, el elemento que muestra toda la información sobre una reserva ya creada.

También se crea un elemento emergente que mostrará los mensajes del estado de las funciones para mostrar mensajes de éxito o de error.

Aquí podemos ver los métodos que los generan desde el controlador *Front-End*.

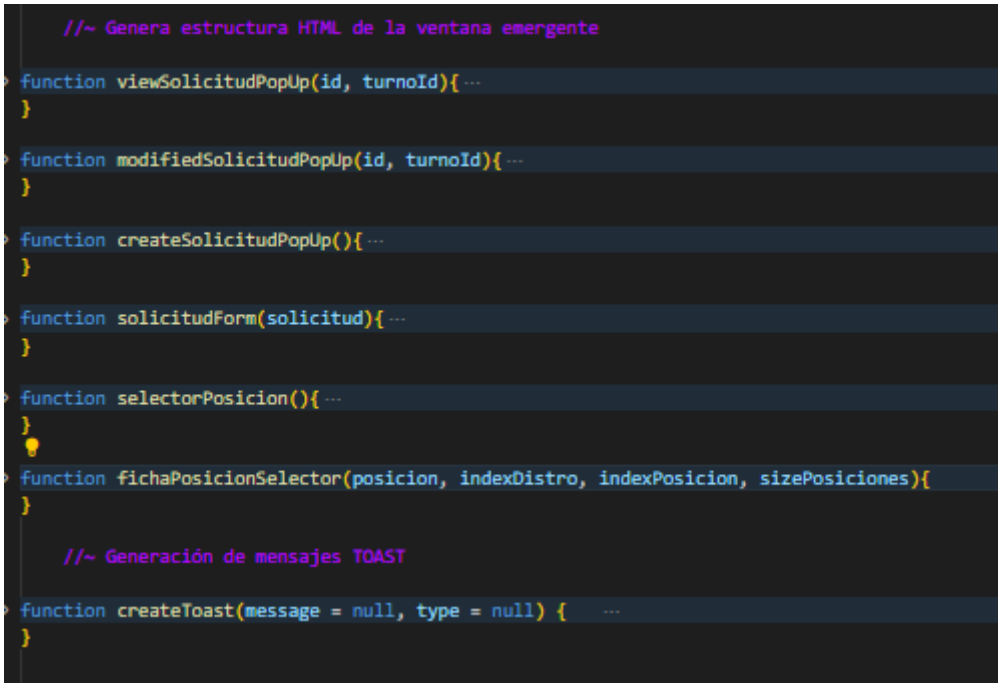


Figura 39 - Funciones JS que generan contenido HTML de las ventanas emergentes

La implementación de la funcionalidad que permite al usuario gestionar las reservas ha seguido el siguiente orden.

Creación de reservas

Para ello se desarrolla una llamada Ajax que envía los datos de la reserva. Esta llamada depende del tipo de reserva que sea. Si es una reserva "Normal" llamará a la función para buscar posiciones; si es una reserva privada, llamará a la función de creación de reserva.

Una vez se hace la llamada al *Back-End*. El controlador lanza la función correspondiente. La función de búsqueda de posiciones dependerá del número de comensales. Si existe una posición con el mismo o mayor número de comensales, devolverá un array de todas las posiciones posibles separadas por comedores. En caso contrario, realizará una búsqueda de todas las combinaciones posibles de las posiciones libres que hay en cada uno de los comedores, devolviendo un array con todas ellas. Cada una de estas combinaciones incluye todas las mesas que se juntarán y la matriz resultante para la distribución en el caso de que se junten.

Antes de esto, hay que explicar cómo está representada la distribución en la matriz. Para ello, hemos utilizado números enteros donde los números mayores que 0 representar las posiciones, los valores 0 representan espacios libres y por último, los números negativos representan paredes y/o columnas (-1), mesas y mobiliario de uso de los trabajadores (-2), zonas de paso (-3) y espacio de las posiciones (-4).

Cada una de las posiciones genera un espacio alrededor suyo ocupando un elemento más de manera horizontal y vertical, sin tener en cuenta los elementos diagonales. Esto se tendrá en cuenta a la hora de recolocar posiciones siguiendo las siguientes reglas:

- Una posición puede no tener espacio por uno de los lados menores si se encuentra en contacto directo con una pared, columna o zona de paso.
- El espacio que genera una nueva posición no podrá consumir el espacio de otra posición ya existente.

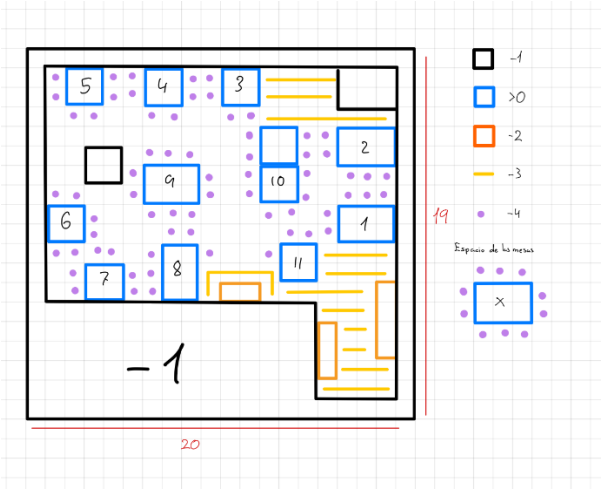


Figura 40 - Boceto inicial de matriz para distribución espacial

Para realizar esta búsqueda de las diferentes combinaciones, la función sigue una serie de pasos:

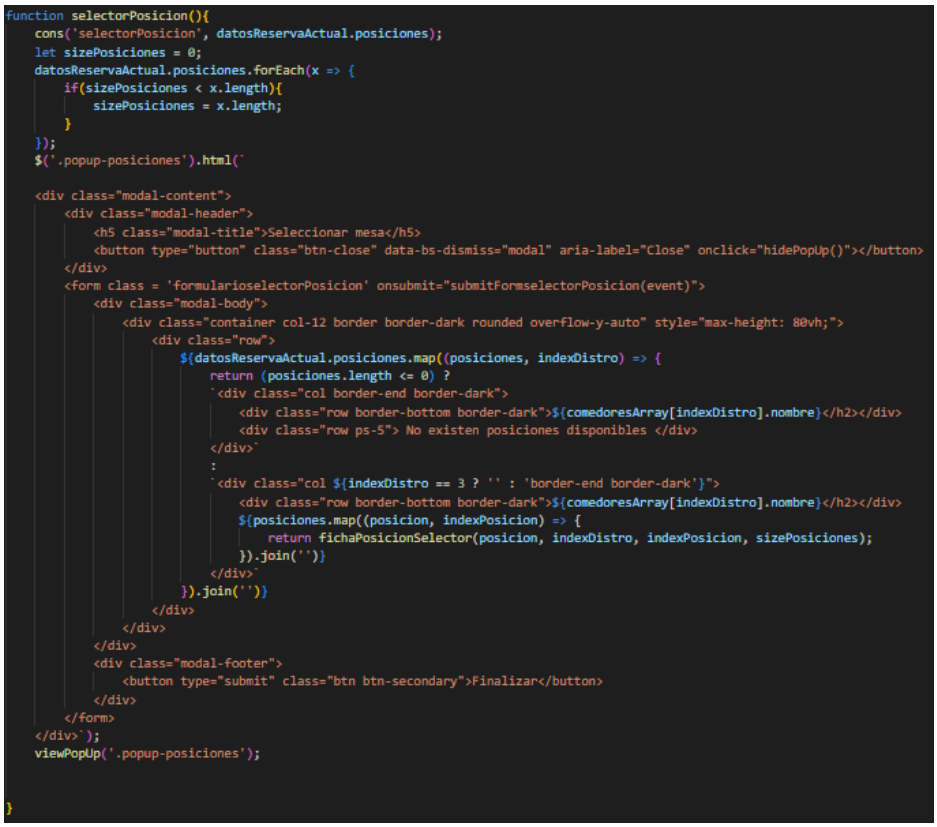
1. Recorrer la matriz en busca de una posición libre.
2. Recorrer vertical y horizontalmente desde la posición libre en busca de otra posición libre. En el caso de encontrarse una pared, columna o mesa reservada, dejará de recorrer en dicho sentido.
    a. En el caso de encontrar otra posición libre que pueda alojar a los comensales de la reserva, continuará la ejecución.
    b. En caso contrario, seguirá buscando otra posición libre hasta cumplir lo anterior.
3. Teniendo ya las posiciones para combinar, se eliminan de la matriz junto con su espacio.
4. Se calcula el largo de la nueva posición que se creará al combinar las seleccionadas.
5. Se busca un espacio donde podría colocar, ya sea vertical u horizontalmente.
    a. Si se encuentra, continúa.
    b. En caso contrario, se rompe la ejecución de los bucles.
6. Se coloca la matriz y se guarda una imagen de la matriz resultante junto un array con las posiciones a combinar.
7. Se pasa a buscar otra combinación.

Este es el método que se encarga de buscar las posiciones simples. Debido a la extensión del método que busca las combinaciones, no se mostrará a través de una imagen y se podrá acceder al mismo dentro del código proporcionado.

```php
1 reference
function getPosiciones($reserva, $turno){
    $contexto = [
        'exito' =>  false,
        'posiciones' =>  null,
        'msg' => ''
    ];
    if($reserva->comensales <= (MAX_COMENSALES_DEFAULT + 1)){
        $contexto = findPositionDefault($reserva, $turno);
    }
    if(!$contexto['exito'] && $reserva->comensales >= (MIN_COMENSALES_DEFAULT)){
        $contexto = findPositionReorder($reserva, $turno);
    }
    return $contexto;
}

1 reference
function findPositionDefault($reserva, $turno){
    $contexto = [
        'exito' =>  false,
        'posiciones' =>  null,
        'tipo' => 'Default',
        'msg' => '',
        'turno' => $turno
    ];
    $posiciones = [[], [], [], []];
    $centinela = [];
    foreach ($turno->listaDistros as $index => $distro) {
        array_push($centinela, $distro);
        if ($distro->estado == 'Abierta') {
            foreach ($distro->listaPosiciones as $posicion) {
                if((($posicion->comensales + 1) >= $reserva->comensales)){
                    $filtro = array_filter($turno->listaSolicitudes, function($x) use ($distro, $posicion, $reserva) {
                        return (($x->distro == $distro->id) && ($x->posicion == $posicion->id) && ($x->id != $reserva->id));
                    });
                    if (count($filtro) == 0) {
                        array_push($posiciones[$index], $posicion);
                    }
                }

            }
        }
    }       You, 3 months ago • GetPosicionesDefault …
    foreach ($posiciones as $array) {
        if (count($array) > 0) {
            $contexto['exito'] = true;
            $contexto['posiciones'] = $posiciones;
            break;
        }
    }

    $contexto['centTurno'] = $turno;

    return $contexto;
}
```

Figura 41 - Funciones PHP de obtención de posiciones disponibles

En las siguientes imágenes se muestra el código del selector de posiciones con sus dos posibles vistas según el tipo de posiciones obtenidas. En la primera se muestra cómo se crea la estructura interna del pop-up, y en la segunda cada una de las fichas que representan las posibles opciones que el usuario puede elegir.



```
function selectorPosicion(){
    cons('selectorPosicion', datosReservaActual.posiciones);
    let sizePosiciones = 0;
    datosReservaActual.posiciones.forEach(x => {
        if(sizePosiciones < x.length){
            sizePosiciones = x.length;
        }
    });
    $('.popup-posiciones').html(`

    <div class="modal-content">
        <div class="modal-header">
            <h5 class="modal-title">Seleccionar mesa</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close" onclick="hidePopUp()"></button>
        </div>
        <form class = 'formularioselectorPosicion' onsubmit="submitFormselectorPosicion(event)">
            <div class="modal-body">
                <div class="container col-12 border border-dark rounded overflow-y-auto" style="max-height: 80vh;">
                    <div class="row">
                        ${datosReservaActual.posiciones.map((posiciones, indexDistro) => {
                            return (posiciones.length <= 0) ?
                            `<div class="col border-end border-dark">
                                <div class="row border-bottom border-dark">${comedoresArray[indexDistro].nombre}</h2></div>
                                <div class="row ps-5"> No existen posiciones disponibles </div>
                            </div>`
                            :
                            `<div class="col ${indexDistro == 3 ? '' : 'border-end border-dark'}">
                                <div class="row border-bottom border-dark">${comedoresArray[indexDistro].nombre}</h2></div>
                                ${posiciones.map((posicion, indexPosicion) => {
                                    return fichaPosicionSelector(posicion, indexDistro, indexPosicion, sizePosiciones);
                                }).join('')}
                            </div>`
                        }).join('')}
                    </div>
                </div>
            </div>
            <div class="modal-footer">
                <button type="submit" class="btn btn-secondary">Finalizar</button>
            </div>
        </form>
    </div>`);
    viewPopUp('.popup-posiciones');


}
```

Figura 42 - Función JS que genera la estructura HTML del selector de posiciones

```
function fichaPosicionSelector(posicion, indexDistro, indexPosicion, sizePosiciones){
    html = '';
    cons(datosReservaActual.reserva.turno);
    if(datosReservaActual.tipo === 'Default'){
        let idPosicion = $.grep(datosReservaActual.turno.listaDistros , (x) => x.id === posicion.distro)[0].comedor * 100 + posicion.id;
        if(datosReservaActual.reserva.posicion === null || datosReservaActual.reserva.distro === null ||
            datosReservaActual.reserva.posicion === undefined || datosReservaActual.reserva.distro === undefined){
                datosReservaActual.reserva.posicion = posicion.id;
                datosReservaActual.reserva.distro = posicion.distro;
        }
        html = `
        <div class="fichaSolicitud row ${indexPosicion > sizePosiciones ? '' : 'border-bottom border-dark'}">
            <div class="form-check ps-5">
                <input class="form-check-input" type="radio" name="posicion" id="posicion" value="${posicion.id},${posicion.distro}"
                ${(posicion.id === datosReservaActual.reserva.posicion && posicion.distro === datosReservaActual.reserva.distro) ? 'checked' : ''}>
                <label class="form-check-label d-flex align-items-center" for="posicion">
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-square" viewBox="0 0 16 16">
                        <path d="M14 1a1 1 0 0 1 1 1v12a1 1 0 0 1-1 1H2a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1h12zM2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0 0 2
                    </svg>
                    <span>  Mesa: ${idPosicion}  </span>
                    <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-people-fill" viewBox="0 0 16 16">
                        <path d="M7 14s-1 0-1-1 1-4 5-4 5 3 5 4-1 1-1 1H7Zm4-6a3 3 0 1 0 0-6 3 3 0 0 0 0 6Zm-5.784 6A2.238 2.238 0 0 1 5 13c0-1.355.68-2.75
                    </svg>
                    <span>  ${posicion.comensales} </span>
                </label>
            </div>
        </div>`;
    } else if(datosReservaActual.tipo === 'Reorder'){
        if(datosReservaActual.reserva.posicion === null || datosReservaActual.reserva.distro === null ||
            datosReservaActual.reserva.posicion === undefined || datosReservaActual.reserva.distro === undefined){
                datosReservaActual.reserva.posicion = indexPosicion;
                datosReservaActual.reserva.distro = posicion.distroId;
        }
        html = `
        <div class="fichaSolicitud row ${indexPosicion > sizePosiciones ? '' : 'border-bottom border-dark'}">
            <div class="form-check ps-5">
                <input class="form-check-input" type="radio" name="posicion" id="posicion" value="${indexDistro},${indexPosicion}"
                ${(indexPosicion === datosReservaActual.reserva.posicion && posicion.distroId === datosReservaActual.reserva.distro) ? 'checked' : ''}>
                <label class="form-check-label" for="posicion">
                    <div class="container  d-flex align-items-center ">
                        <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-plus-square" viewBox="0 0 16 16">
                            <path d="M14 1a1 1 0 0 1 1 1v12a1 1 0 0 1-1 1H2a1 1 0 0 1-1-1V2a1 1 0 0 1 1-1h12zM2 0a2 2 0 0 0-2 2v12a2 2 0 0 0 2 2h12a2 2 0 0
                            <path d="M8 4a.5.5 0 0 1 .5.5v3h3a.5.5 0 0 1 0 1h-3v3a.5.5 0 0 1-1 0v-3h-3a.5.5 0 0 1 0-1h3v-3A.5.5 0 0 1 8 4z"/>
                        </svg>
                        <span>  ${posicion.posiciones.map((mesa, indexMesa)=> {
                                let idMesa = $.grep(datosReservaActual.turno.listaDistros , (x) => x.id === posicion.distroId)[0].comedor * 100 + mesa;
                                return indexMesa < posicion.length -1 ? ` ${idMesa},` : ` ${idMesa}.`;
                            }).join('')}
                        </span>
                    </div>
                </label>
            </div>
        </div>`;
    }{

    }

    return html;
}
```

Figura 43 - Función JS que genera el HTML de las fichas de selección de posición

Una vez obtenidas las posiciones, ya sean simples o combinadas, el *Back-End* se las envía al *Front-End*, el cual mostrará en el pop-up 4 listas de posiciones, una para cada una de las distribuciones del turno. El usuario elegirá una y el controlador JS realizará la llamada para la creación de la reserva.

Esta llamada tendrá en cuenta el tipo de reserva y, en el caso de ser "Normal", tendrá en cuenta el tipo de posición que se le ha asignado a la reserva, de tal manera que ofrece 3 posibilidades.

1. **Reserva normal con posición simple**: crea la reserva asignándole posición y distro.
2. **Reserva normal con posición combinada**: asigna todas las mesas de las posiciones a la primera posición para borrar el resto de posiciones, modifica la matriz de la distribución y crea la reserva asignándole posición y distro.
3. **Reserva privada:** durante la creación, llama al método que gestiona salas privadas para modificar la distribución si se puede, y enviará un mensaje de error en caso contrario. Si no existe error, crea la reserva asignándole la distro.

En las siguientes imágenes podemos ver el método que se encarga de gestionar las diferentes salas privadas de los comedores. En la primera muestra cómo se verifica si se puede abrir o cerrar la sala, en la segunda muestra la gestión en caso de pasar la validación.

```php
12 references
function gestionarPrivado($turno, $distro, $privado1, $privado2, $privadoPrincipal, $solicitudCanceladaId){
    $contexto = [
        'exito' => true,
        'mensajeError' => '',
        'function' => 'gestionarPrivado'
    ];      You, 2 months ago • Gestion Privados + GetPosiciones …
    foreach ($turno->listaSolicitudes as $x) {
        if(($x->estado != 'Cancelada') && ($x->distro == $distro->id) && ($x->id != $solicitudCanceladaId)){
            if($distro->comedor == 1){
                if(((!($distro->privado1 == 1) && ($privado1 == 1) && $x->posicion >= 17 && $x->posicion <= 22) ||
                (!($distro->privado2 == 1) && ($privado2 == 1) && $x->posicion >= 1 && $x->posicion <= 6) ||
                (!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1) && ($x->posicion == 1 ||($x->posicion >= 7 && $x->posicion <= 15))))){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'No se puede cerrar el privado con una reserva (Mesa:'.$x->posicion.')';
                    break;
                } else if((($distro->privado1 == 1) && !($privado1 == 1) && ($x->tipo == 'Privado1' || $x->tipo == 'Privado12')) ||
                (($distro->privado2 == 1) && !($privado2 == 1) && ($x->tipo == 'privado2' || $x->tipo == 'Privado12'|| $x->tipo == 'PrivadoPrincipal2')) ||
                (($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1) && ($x->tipo == 'PrivadoPrincipal' || $x->tipo == 'PrivadoPrincipal2'))){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'No se puede abrir el privado con una reserva privada(Mesa:'.$x->tipo.')';
                    break;
                }  else if(($distro->privado1 == $privado1) && ($distro->privado2 == $privado2) && ($distro->privadoPrincipal == $privadoPrincipal)){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'Mismo estado de privados';
                    break;
                }

            } else if($distro->comedor == 2){
                if($privado1 != 1 && $privado2 != 1 && $x->tipo == 'PrivadoNicomedes'){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'No se puede abrir el privado con una reserva (Mesa:'.$x->posicion.')';
                    break;
                }
                if($privado1 == 1 && $privado2 == 1){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'No se puede cerrar el privado con una reserva (Mesa:'.$x->posicion.')';
                    break;
                }
                if(($distro->privado1 == $privado1) && ($distro->privado2 == $privado2)){
                    $contexto["exito"] = false;
                    $contexto["mensajeError"] = 'Mismo estado de privados';
                    break;
                }
            }
        }
    }
```

Figura 44 - Función PHP de gestión de privados 1

```php
if($contexto["exito"]){
    if($distro->comedor == 1){
        if((($distro->privado1 == 1) && !($privado1 == 1)) ||
            (($distro->privado2 == 1) && !($privado2 == 1)) ||
            (($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1))){
            if(($distro->privado1 == 1) && !($privado1 == 1)) $p1 = 0; else $p1 = 1;
            if(($distro->privado2 == 1) && !($privado2 == 1)) $p2 = 0; else $p2 = 1;
            if(($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1)) $pPrincipal = 0; else $pPrincipal = 1;
            $contexto = crearPosicionesDistro($distro, $p1, $p2, $pPrincipal);
        }
        if((!($distro->privado1 == 1) && ($privado1 == 1)) ||
            (!($distro->privado2 == 1) && ($privado2 == 1)) ||
            (!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1))){

            if(!($distro->privado1 == 1) && ($privado1 == 1)) $p1 = 1; else $p1 = 0;
            if(!($distro->privado2 == 1) && ($privado2 == 1)) $p2 = 1; else $p2 = 0;
            if(!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1)) $pPrincipal = 1; else $pPrincipal = 0;
            $contexto = borrarPosicionesDistro($distro, $p1, $p2, $pPrincipal);

        }
    } else if($distro->comedor == 2){
        if(($distro->privado1 == 1) && !($privado1 == 1) && ($distro->privado2 == 1) && !($privado2 == 1) && ($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1)){
            $contexto = crearPosicionesDistro($distro, 0, 0, 0);
        } else if(!($distro->privado1 == 1) && ($privado1 == 1) && !($distro->privado2 == 1) && ($privado2 == 1) && !($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1)){
            $contexto = borrarPosicionesDistro($distro, 1, 1, 1);
        }
    }
    if($distro->comedor==1){
        if(!($distro->privado1 == 1) && ($privado1 == 1)){
            $distro->matriz = actionPrivateInMatrix('Cerrar', $distro->matriz, 'privado1');
        } else if (($distro->privado1 == 1) && !($privado1 == 1)){
            $distro->matriz = actionPrivateInMatrix('Abrir', $distro->matriz, 'privado1');
        }

        if(!($distro->privado2 == 1) && ($privado2 == 1)){
            $distro->matriz = actionPrivateInMatrix('Cerrar', $distro->matriz, 'privado2');
        } else if (($distro->privado2 == 1) && !($privado2 == 1)){
            $distro->matriz = actionPrivateInMatrix('Abrir', $distro->matriz, 'privado2');
        }
        if(!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1) && (($distro->privado2 == $privado2))){
            $distro->matriz = actionPrivateInMatrix('Cerrar', $distro->matriz, 'privadoPrincipal');
        } else if (($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1) && (($distro->privado2 == $privado2))){
            $distro->matriz = actionPrivateInMatrix('Abrir', $distro->matriz, 'privadoPrincipal');
        } else if(!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1) && (($distro->privado2 != $privado2))){
            $distro->matriz = actionPrivateInMatrix('Cerrar', $distro->matriz, 'privadoPrincipal2');
        } else if (($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1) && (($distro->privado2 != $privado2))){
            $distro->matriz = actionPrivateInMatrix('Abrir', $distro->matriz, 'privadoPrincipal2');
        }
    } else if($distro->comedor==2){
        if($distro->estado =='Cerrada'){
            $distro->estado = 'Abierta';
        }
        if(!($distro->privado1 == 1) && ($privado1 == 1)) || (!($distro->privado2 == 1) && ($privado2 == 1)) || (!($distro->privadoPrincipal == 1) && ($privadoPrincipal == 1))){
            $distro->matriz = getDistroDefault(21);
        } else if((($distro->privado1 == 1) && !($privado1 == 1)) || (($distro->privado2 == 1) && !($privado2 == 1)) || (($distro->privadoPrincipal == 1) && !($privadoPrincipal == 1))){
            $distro->matriz = getDistroDefault(2);
        }
    }
    $distro->privado1 = $privado1;
    $distro->privado2 = $privado2;
    $distro->privadoPrincipal = $privadoPrincipal;
    $contexto = modificarDistro($distro);
}
return $contexto;            You, 1 second ago • Uncommitted changes
```

Figura 45 - Función PHP de gestión de privados 2

Visualización de reservas

Una vez se ha creado la reserva, el usuario tendrá acceso a un resumen de información en la lista de reservas, pero si necesita ver más datos, puede pulsar sobre la ficha de la reserva para desplegar el *pop-up* mostrando toda la información de la reserva.

Éste mostrará botones a través de los cuales se podrá modificar la reserva, cambiar su posición o cancelar la reserva.

Modificación de reservas

En este caso, se le permite al usuario cambiar la posición de la mesa, donde volverá a mostrarse el selector de posiciones dentro del `pop-up` y realizará una llamada tras elegir la nueva posición que modificará la reserva.

Por otro lado, si el usuario quiere modificar alguno de los datos, se le presentará un nuevo formulario.

Cabe destacar que tanto para la creación y modificación de reservas se usa el mismo método que crea dinámicamente el formulario según la casuística de la reserva.

```
function modifiedSolicitudPopUp(id, turnoId){
    let turno = $.grep(turnosArray , (x) => x.id === turnoId)[0];
    let solicitud = $.grep(turno.listaSolicitudes, (x) => x.id === id)[0];
    $('.popup-formulario').html(`
            ${solicitudForm(solicitud)} `
    );
    viewPopUp('.popup-formulario')
}

function createSolicitudPopUp(){
    $('.popup-formulario').html(`
            ${solicitudForm()} `
    );
    viewPopUp('.popup-formulario')
}

function solicitudForm(solicitud){···
}
```

Figura 46 - Funciones JS que lanzan el formulario de reserva

Cancelación de reservas

Por último, para la gestión de reservas, el usuario puede cancelar una reserva donde se encontrará la misma casuística que en la creación:

1. **Reserva normal con posición simple**: cancela la reserva desasignándole posición y distro.
2. **Reserva normal con posición combinada**: al detectar que la mesa tiene varias mesas, tratará de dividir la posición en una posición por mesa y colocar dicha posición en la matriz.
3. **Reserva privada:** durante la cancelación, llama al método que gestiona salas privadas para modificar la distribución abriendo la sala privada.

En la siguiente imagen se muestran todos los métodos que se utilizan para crear una reserva que tendrá una posición, fruto de la combinación de otras o de dividir la posición de una reserva que se creó utilizando la combinación de posiciones.

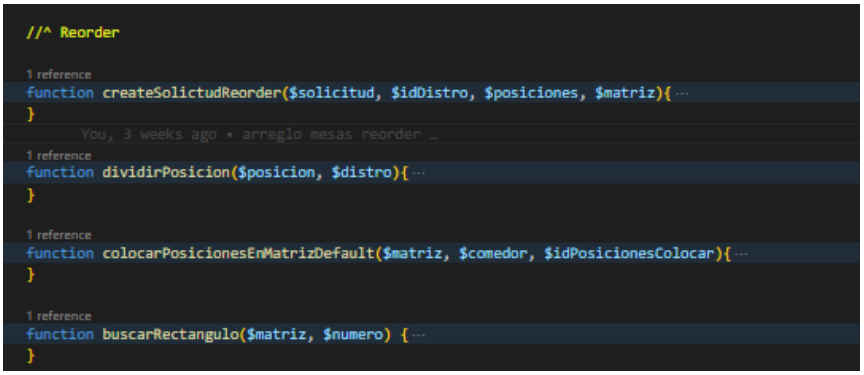
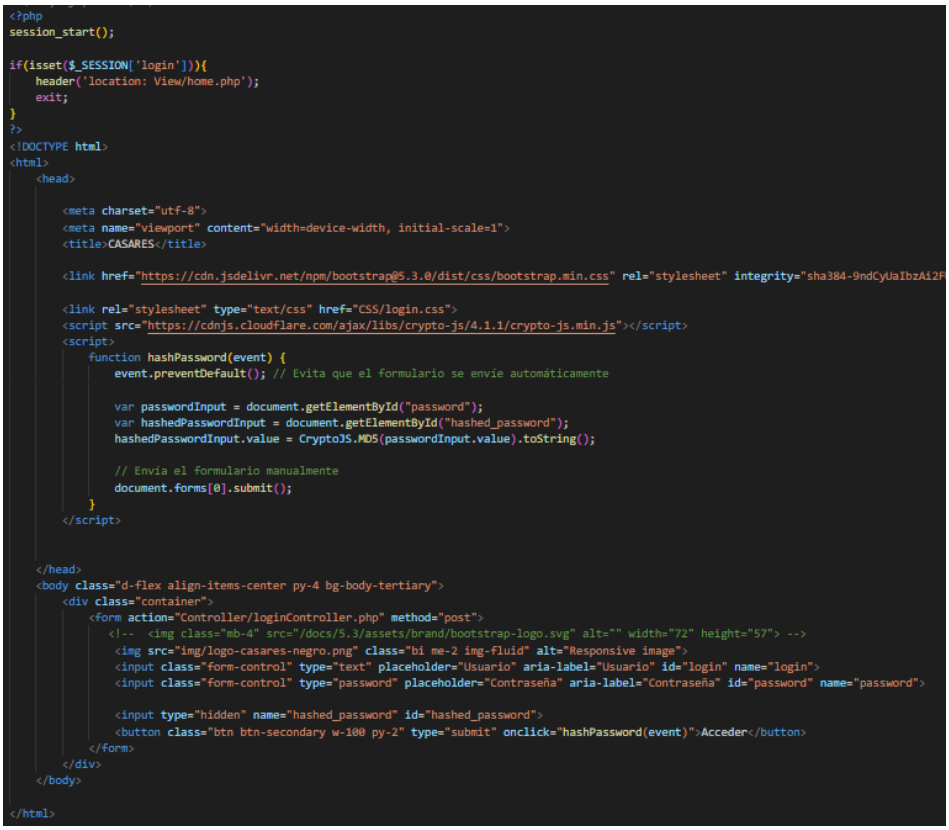


Figura 47 - Funciones PHP relacionadas con las reservas con posiciones combinadas

GESTIÓN DE SESIONES

Para la gestión de las sesiones y el acceso a la aplicación, se ha desarrollado el archivo `login.php` que ofrecerá el HTML con el formulario de inicio de sesión y los controladores de *login* y *logout*.

El primero de estos realizará la comprobación de usuario y contraseña y redirigirá a la página principal. Por otro lado, en el *Back-End* y en la página principal, se comprueba si se ha iniciado sesión y en caso de error, redirigirá a la página de inicio de sesión.



```php
<?php
session_start();

if(isset($_SESSION['login'])){
    header('location: View/home.php');
    exit;
}
?>
<!DOCTYPE html>
<html>
    <head>

        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>CASARES</title>

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2F

        <link rel="stylesheet" type="text/css" href="CSS/login.css">
        <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>
        <script>
            function hashPassword(event) {
                event.preventDefault(); // Evita que el formulario se envíe automáticamente

                var passwordInput = document.getElementById("password");
                var hashedPasswordInput = document.getElementById("hashed_password");
                hashedPasswordInput.value = CryptoJS.MD5(passwordInput.value).toString();

                // Envía el formulario manualmente
                document.forms[0].submit();
            }
        </script>


    </head>
    <body class="d-flex align-items-center py-4 bg-body-tertiary">
        <div class="container">
            <form action="Controller/loginController.php" method="post">
              <!--  <img class="mb-4" src="/docs/5.3/assets/brand/bootstrap-logo.svg" alt="" width="72" height="57"> -->
                <img src="img/logo-casares-negro.png" class="bi me-2 img-fluid" alt="Responsive image">
                <input class="form-control" type="text" placeholder="Usuario" aria-label="Usuario" id="login" name="login">
                <input class="form-control" type="password" placeholder="Contraseña" aria-label="Contraseña" id="password" name="password">

                <input type="hidden" name="hashed_password" id="hashed_password">
                <button class="btn btn-secondary w-100 py-2" type="submit" onclick="hashPassword(event)">Acceder</button>
            </form>
        </div>
    </body>

</html>
```

Figura 48 - Archivo index.php

```php
<?php
session_start();
include_once '../includes/include.php';


if(isset($_SESSION['login'])){
    header('location: ../View/home.php');
    exit;
} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if(isset($_POST['login']) && isset($_POST['hashed_password'])){
        $login = comprobar_entrada($_POST['login']);
        $hashedPassword = comprobar_entrada($_POST['hashed_password']); // Contraseña recibida en formato de hash MD5

        // Realizar las operaciones de verificación y autenticación aquí
        $correctUser = existeUsuario($login, $hashedPassword);

        if($correctUser){
            // Establecer la duración de la sesión en 30 minutos (1800 segundos)
            $sessionLifetime = 1800;

            // Actualizar la caducidad de la cookie de sesión
            session_set_cookie_params($sessionLifetime);

            // Actualizar el tiempo de expiración de la sesión
            $_SESSION['expire'] = time() + $sessionLifetime;

            $_SESSION['login'] = $login;
            header('location: ../View/home.php');
            exit;
        } else {
            header('location: ../index.php');
            exit;
        }
    }
}


?>
```

Figura 49 - Archivo loginController.php

CREACIÓN DE ESTILOS

Para añadir estilos a la página se ha utilizado *Bootstrap*. Éste es un *framework* de desarrollo *Front-end* muy popular que facilita la creación de sitios web adaptativos (*responsive*) y atractivos. Aquí se muestra una breve explicación de cómo se utiliza:

- **Configuración:** lo primero que se debe hacer es agregar los archivos de *Bootstrap* al proyecto. Se puede descargar desde el sitio web oficial de Bootstrap o utilizar una CDN (*Content Delivery Network*) para cargar los archivos directamente en tu proyecto.
- **Estructura HTML:** *Bootstrap* utiliza una estructura HTML basada en una cuadrícula (*grid system*) para organizar el contenido de la página. Debes estructurar el contenido utilizando las clases proporcionadas por *Bootstrap*, como `container` para crear un contenedor principal, y `row` para crear filas en la cuadrícula.
- **Componentes:** *Bootstrap* ofrece una amplia gama de componentes predefinidos que puedes utilizar en el sitio web, como botones, formularios, barras de navegación, carruseles, tarjetas… Se pueden añadir estos componentes utilizando las clases y atributos proporcionados por *Bootstrap*.
- **Adaptavilidad (*Responsive*):** uno de los puntos fuertes de *Bootstrap* es su enfoque en la creación de sitios web responsivos. Se pueden utilizar las clases de *Bootstrap* para ajustar el diseño y el comportamiento de los elementos en diferentes tamaños de pantalla.

También se ha utilizado, aunque en menor medida, algún estilo CSS.

PRUEBAS DE CAJA BLANCA

Las pruebas de caja blanca, también conocidas como pruebas de caja clara o pruebas de estructura, son una técnica utilizada en el desarrollo de software para evaluar el funcionamiento interno de un sistema. Se enfocan en analizar y probar el código fuente, la lógica interna y la estructura del software.

En las pruebas de caja blanca, los desarrolladores o *testers* tienen un conocimiento detallado de cómo está construido el sistema, incluyendo la arquitectura, las funciones, las rutas de ejecución y las estructuras de datos. Esto les permite diseñar casos de prueba basados en esta información interna.

Estas pruebas se llevan a cabo en la etapa de desarrollo del código, puesto que se realizan sobre éste con los siguientes objetivos:

1. **Prueba de cobertura de código:** se evalúa cuántas líneas del código fuente han sido ejecutadas durante las pruebas. Esto ayuda a identificar áreas no probadas y mejorar la cobertura del código.
2. **Prueba de bucles y condiciones:** se diseñan casos de prueba que cubren todas las posibles ramas lógicas, como bucles, condiciones if/else y casos de borde, para asegurar que todas las condiciones sean evaluadas correctamente.
3. **Prueba de flujo de datos:** se examina cómo se manejan y transforman los datos dentro del software, identificando posibles errores o inconsistencias en la manipulación de datos.
4. **Revisión del código:** se analiza el código fuente en busca de posibles errores o vulnerabilidades, siguiendo buenas prácticas de codificación y estándares establecidos.

En resumen, las pruebas de caja blanca son una técnica de prueba que se enfoca en la lógica interna y el código fuente de un sistema para garantizar su correcto funcionamiento. Ayudan a identificar errores en la implementación y a mejorar la calidad del software.

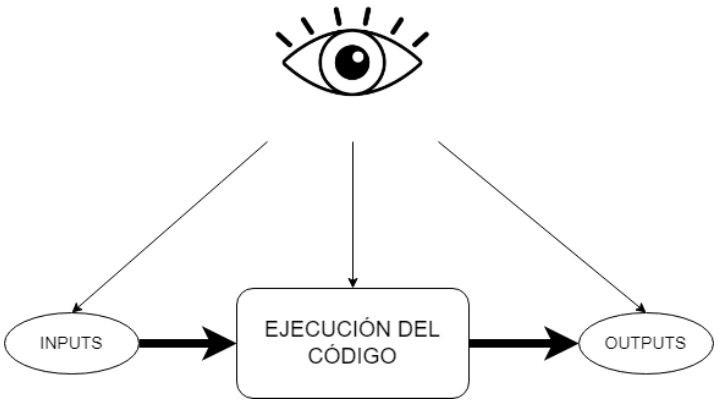


Figura 50 - Pruebas de caja blanca

Estas pruebas se realizarán a través del uso de la consola del navegador donde, gracias a las funciones de `console.log`, podremos mostrar los datos que nos llegan al *Front-End* desde el *Back-End*.

De esta manera, podemos comprobar el correcto funcionamiento del controlador JS. Para comprobar el funcionamiento del controlador PHP, también se hará uso de estas herramientas debido a que en las peticiones Ajax que realizamos, el controlador *Back-End* nos devuelve un JSON que incluye éxito de la función, mensaje de error si los hubiera, y los datos que se requieren. De esta forma, podemos depurar el código de manera sencilla.

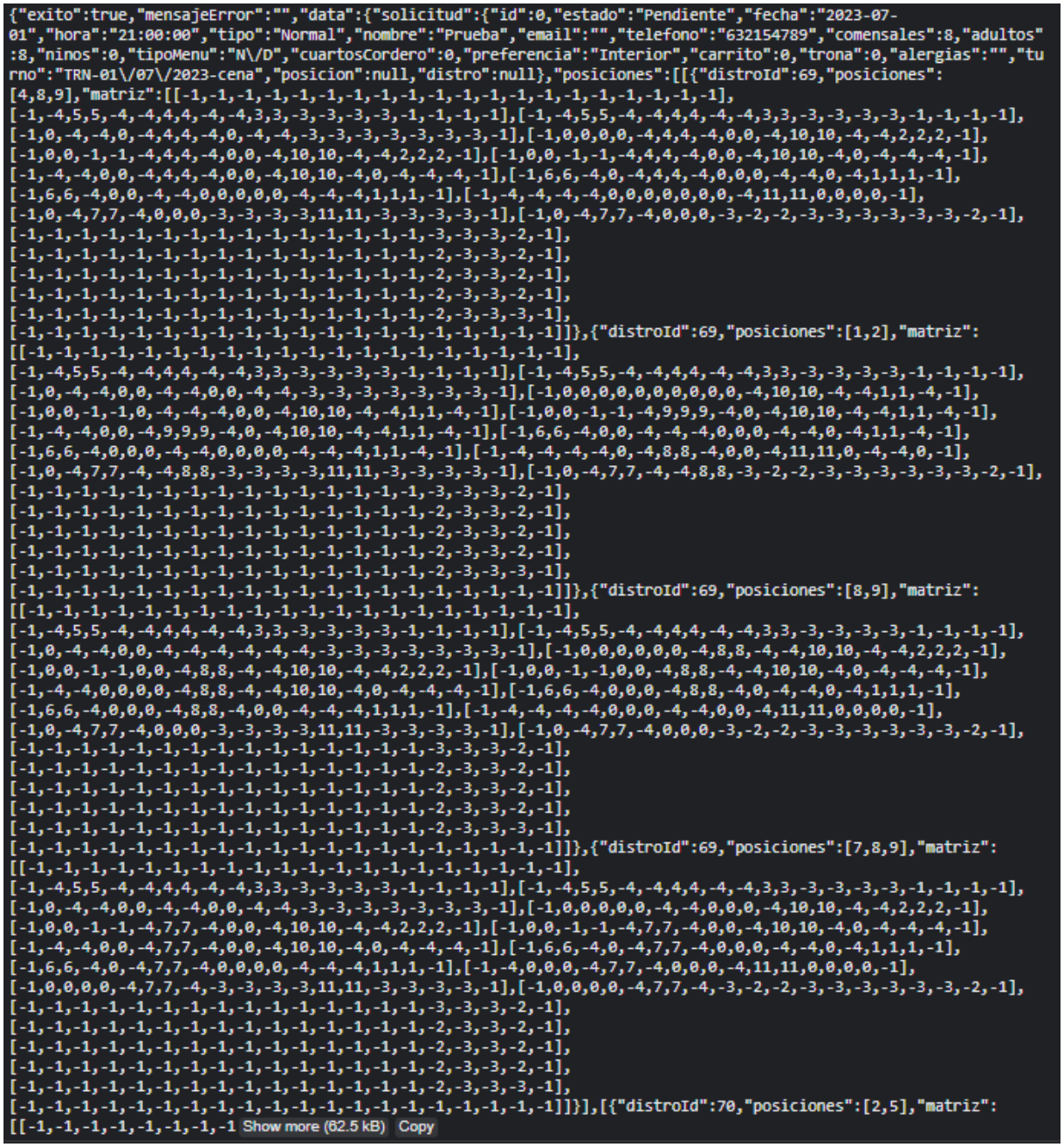


Figura 51 - Ejemplo console.log para pruebas de caja blanca

PRUEBAS DE CAJA NEGRA

Las pruebas de caja negra son una técnica utilizada en el desarrollo de software para evaluar el comportamiento externo de un sistema sin conocer su implementación interna. Se centran en probar la funcionalidad del software desde la perspectiva del usuario o cliente.

En las pruebas de caja negra, los *testers* o evaluadores no tienen conocimiento detallado de cómo está construido el sistema ni acceden directamente al código fuente. En cambio, se centran en la entrada y salida del sistema, así como en los requisitos funcionales y no funcionales del software.

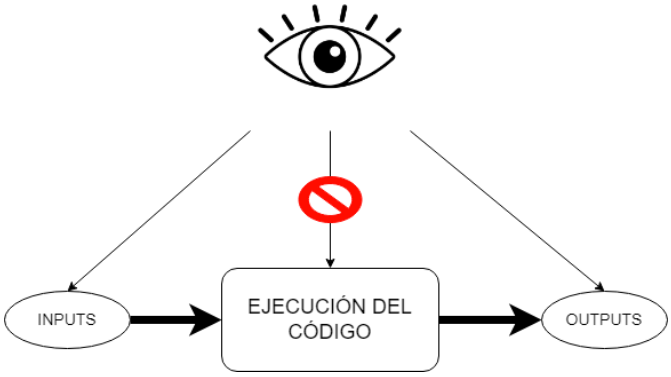


Figura 52 - Pruebas de caja negra

Debido al elevado número de casuísticas, algunas de las pruebas recopiladas engloban varias pruebas por tipología de reservas.

PCN-01: Inicio de sesión	
Objetivo de la prueba	Comprobar que el usuario puede acceder a la página principal desde la página de acceso.
Precondiciones	Debe existir el usuario y contraseña.
Datos de entrada	Usuario y contraseña correctos.
Respuesta esperada	Redirección a la página principal.

Tabla 75 - PCN-01: Inicio de sesión

PCN-02: Cierre de sesión	
Objetivo de la prueba	Comprobar que el usuario puede cerrar su sesión y no se puede acceder a la aplicación sin volver a iniciar sesión.
Precondiciones	Debe encontrarse en la página principal.
Datos de entrada	Pulsar el botón de cerrar sesión.
Respuesta esperada	Eliminación de la sesión y redirección a la página de acceso.

Tabla 76 - PCN-02: Cierre de sesión

PCN-03: Cambio de fecha	
Objetivo de la prueba	Comprobar que el usuario puede cambiar la fecha para que se muestren los datos de dicha fecha.
Precondiciones	Debe encontrarse en la página principal.
Datos de entrada	Seleccionar una nueva fecha en el selector de fechas.
Respuesta esperada	La información de la página cambia a un turno de esa fecha.

Tabla 77 - PCN-03: Cambio de fecha

PCN-04: Abrir comedor	
Objetivo de la prueba	Comprobar si el usuario puede abrir un comedor.
Precondiciones	Debe encontrarse en la página principal y haber algún comedor abierto sin comensales.
Datos de entrada	El usuario debe marcar la casilla de un comedor que este desmarcada.
Respuesta esperada	La distribución de ese comedor para ese turno queda abierta y en ella pueden colocarse reservas.

Tabla 78 - PCN-04: Abrir comedor

PCN-05: Cerrar comedor	
Objetivo de la prueba	Comprobar si el usuario puede cerrar un comedor.
Precondiciones	Debe encontrarse en la página principal y haber algún comedor cerrado.
Datos de entrada	Desmarcar algún comedor que este marcado. Solo podrá si no tiene reservas asignadas.
Respuesta esperada	Solo puede desmarcar comedores abiertos sin reservas. La distribución se queda cerrada.

Tabla 79 - PCN-05: Cerrar comedor

PCN-06: Cambiar de turno	
Objetivo de la prueba	Comprobar que el usuario puede navegar entre los diferentes turnos.
Precondiciones	Debe encontrarse en la página principal.
Datos de entrada	El usuario pulsa sobe los botones de cambio de turno.
Respuesta esperada	La ficha de turno cambia al turno siguiente o al anterior.

Tabla 80 - PCN-06: Cambiar de turno

PCN-07: Cambiar de comedor	
Objetivo de la prueba	Comprobar que el usuario puede navegar entre los diferentes comedores.
Precondiciones	Debe encontrarse en la página principal.
Datos de entrada	El usuario pulsa sobe los botones de cambio de comedor.
Respuesta esperada	La ficha de comedor cambia al comedor siguiente o al anterior.

Tabla 81 - PCN-07: Cambiar de comedor

PCN-08: Cambiar vista de lista de reservas	
Objetivo de la prueba	Comprobar que el usuario puede alternar en la lista de reservas entre la vista de reservas del turno y reservas del comedor.
Precondiciones	Debe encontrarse en la página principal y haber reservas en el turno correspondiente.
Datos de entrada	Debe pulsar el botón de cambio de vista.
Respuesta esperada	La vista de lista de reserva debe alternar entre las reservas del turno (que incluyen las canceladas) y las reservas del comedor.

Tabla 82 - PCN-08: Cambiar vista de lista de reservas

PCN-09: Formulario de creación de reservas	
Objetivo de la prueba	Comprobar que el usuario puede rellenar el formulario y enviar sus datos al backend.
Precondiciones	Debe encontrarse en la página principal y haber pulsado el botón de crear reserva.
Datos de entrada	Rellenará, como mínimo, los campos obligatorios: - Nombre - Comensales - Teléfono - Fecha - Hora - Tipo
Respuesta esperada	Dependiendo del tipo de reserva y el número de comensales, el Back-End realizara una de las siguientes acciones: - Buscará y devolverá las posiciones simples para crear la reserva. - Buscará y devolverá las combinaciones de posiciones posibles para crear la reserva. - Gestionará las matrices cerrando el privado correspondiente y en caso de éxito creará la reserva.

Tabla 83 - PCN-09: Formulario de creación de reservas

PCN-10: Selección de posición simple	
Objetivo de la prueba	Comprobar que el usuario puede seleccionar una posición simple para crear la reserva.
Precondiciones	Debe haber rellenado y enviado el formulario de creación de reserva y debe haber una o más mesas que puedan abarcar el número de comensales de la reserva.
Datos de entrada	Selección de una posición .
Respuesta esperada	La reserva se crea con éxito.

Tabla 84 - PCN-10: Selección de posición simple

PCN-11: Selección de posición combinada	
Objetivo de la prueba	Comprobar que el usuario puede seleccionar una combinación de posiciones para crear la reserva.
Precondiciones	Debe haber rellenado y enviado el formulario de creación de reserva y debe no haber una o más mesas que puedan abarcar el número de comensales de la reserva.
Datos de entrada	Selección de una combinación de posiciones.
Respuesta esperada	La reserva se crea con éxito y la distribución y posiciones se modifican con éxito.

Tabla 85 - PCN-11: Selección de posición combinada

PCN-13: Visualización detalle de reserva	
Objetivo de la prueba	Comprobar que el usuario puede ver toda la información de una reserva.
Precondiciones	Debe encontrarse en la página principal y debe existir alguna reserva en la lista de reservas.
Datos de entrada	Debe pulsar la ficha de la reserva.
Respuesta esperada	Se despliega un pop-up con la información de la reserva.

Tabla 86 - PCN-13: Visualización detalle de reserva

PCN-14: Modificación datos de reserva	
Objetivo de la prueba	Comprobar que el usuario puede modificar una reserva.
Precondiciones	Debe encontrarse en la página principal viendo el detalle de una reserva y haber pulsado el botón de modificar.
Datos de entrada	Rellenar cualquiera de los datos que permite modificar el formulario.
Respuesta esperada	La reserva debe modificarse con éxito.

Tabla 87 - PCN-14: Modificación datos de reserva

PCN-15: Cambio de posición de reserva	
Objetivo de la prueba	Comprobar que el usuario puede cambiar la posición de una reserva.
Precondiciones	Debe encontrarse en la página principal viendo el detalle de una reserva y haber pulsado el botón de cambiar posición.
Datos de entrada	Seleccionar una posición diferente.
Respuesta esperada	La reserva debe asignarse a esa nueva posición y distro.

Tabla 88 - PCN-15: Cambio de posición de reserva

PCN-16 Cancelar reserva	
Objetivo de la prueba	Comprobar que el usuario puede cancelar una reserva.
Precondiciones	Debe encontrarse en la página principal viendo el detalle de una reserva.
Datos de entrada	Debe pulsar el botón de cancelar
Respuesta esperada	Dependiendo del tipo de reserva y si la posición es simple o combinada, el Back-End realizará una de las siguientes acciones: - Cancelar la reserva. - Dividir la posición en varias y cancelar la reserva. - Gestionar los privados y cancelar la reserva.

Tabla 89 - PCN-16 Cancelar reserva

CAPITULO 8. MANUAL DE USUARIO

En este apartado se va a realizar una explicación acompañada de capturas con el fin de que el usuario final tenga conocimiento de todo lo que puede realizar dentro de la aplicación y cómo debe hacerlo para que, pasadas 1 o 2 horas, éste sea capaz de poder gestionar la información dentro de la página principal de forma autónoma.

INICIO DE SESIÓN

Nada más acceder a la aplicación, se presentará un formulario de acceso, en el que se debe introducir el usuario y contraseña facilitado por el administrador.

Figura 53 – Manual: inicio de sesión

Una vez introducidos, pulsar acceder para entrar en la aplicación. En caso de ser incorrectos, la página nos redirigirá de vuelta al formulario de acceso.

PÁGINA PRINCIPAL

Una vez que el usuario accede a la página principal, tendrá acceso a toda la funcionalidad de la aplicación. Esta página servirá al usuario de centro de control desde el que ver toda la información y poder gestionar reservas, turnos y distribuciones.

VISUALIZACIÓN DE INFORMACIÓN

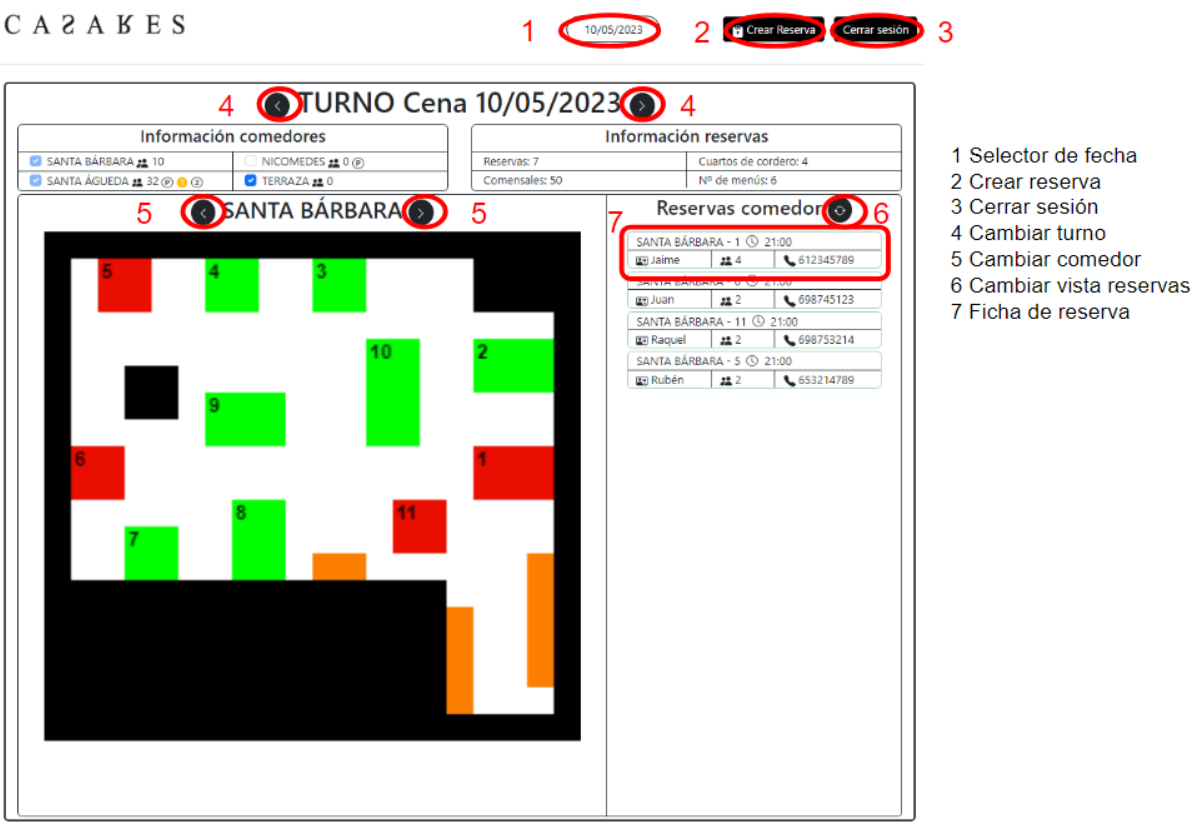


Figura 54 - Manual: página principal

La página principal está compuesta por 2 elementos:

1. **Menú horizontal**

En éste se muestra el selector de fecha (1), que permite seleccionar la fecha de la que queremos ver la información de sus turnos. También se encuentran 2 botones, el primero que inicia el proceso de creación de reserva (2), y el segundo que permite al usuario cerrar la sesión (3).

2. **Ficha de turno**

Todo la información se presenta a través de las fichas de turno. El usuario podrá navegar a través de los turnos utilizando los botones de cambio de turno (4).

Esta ficha presenta una serie de subelementos para dividir la información del turno:

2.1.Resumen de comedores

En éste se muestra el estado de los comedores a través de las casillas, el estado de las salas privadas a través de los iconos y el número de comensales.

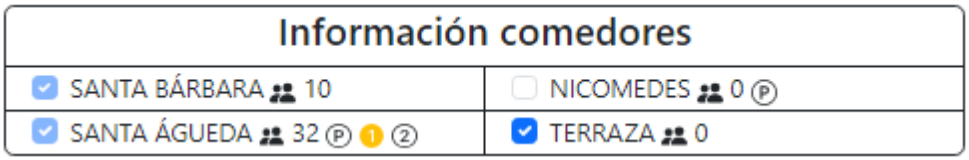


Figura 55 - Manual: resumen de comedores

2.2.Resumen de reservas

En el que se muestra el número total de reservas y comensales, el número de cuartos de cordero y el número de menús reservados.

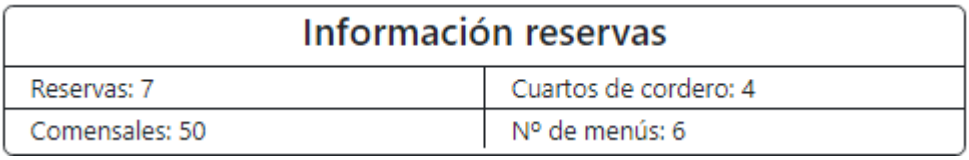


Figura 56 - Manual: resumen de reservas

2.3.Ficha de comedor

Estas fichas cuentan con la información referente a las distribuciones de los comedores dentro del turno y las reservas. El usuario puede navegar a través de los botones de cambio de comedor (5).

Dentro de la ficha de comedor, se encuentran el plano que representa la distribución espacial del comedor y la lista de reservas. El usuario podrá alternar de vista de la lista de reservas, en la que podrá visualizar a través del botón de cambio de vistas de reservas (6). Podrá elegir entre:

- Reservas del comedor: filtra las reservas por el comedor en el que se encuentra el usuario.
- Reservas del turno: muestra todas las reservas del turno incluyendo las canceladas.

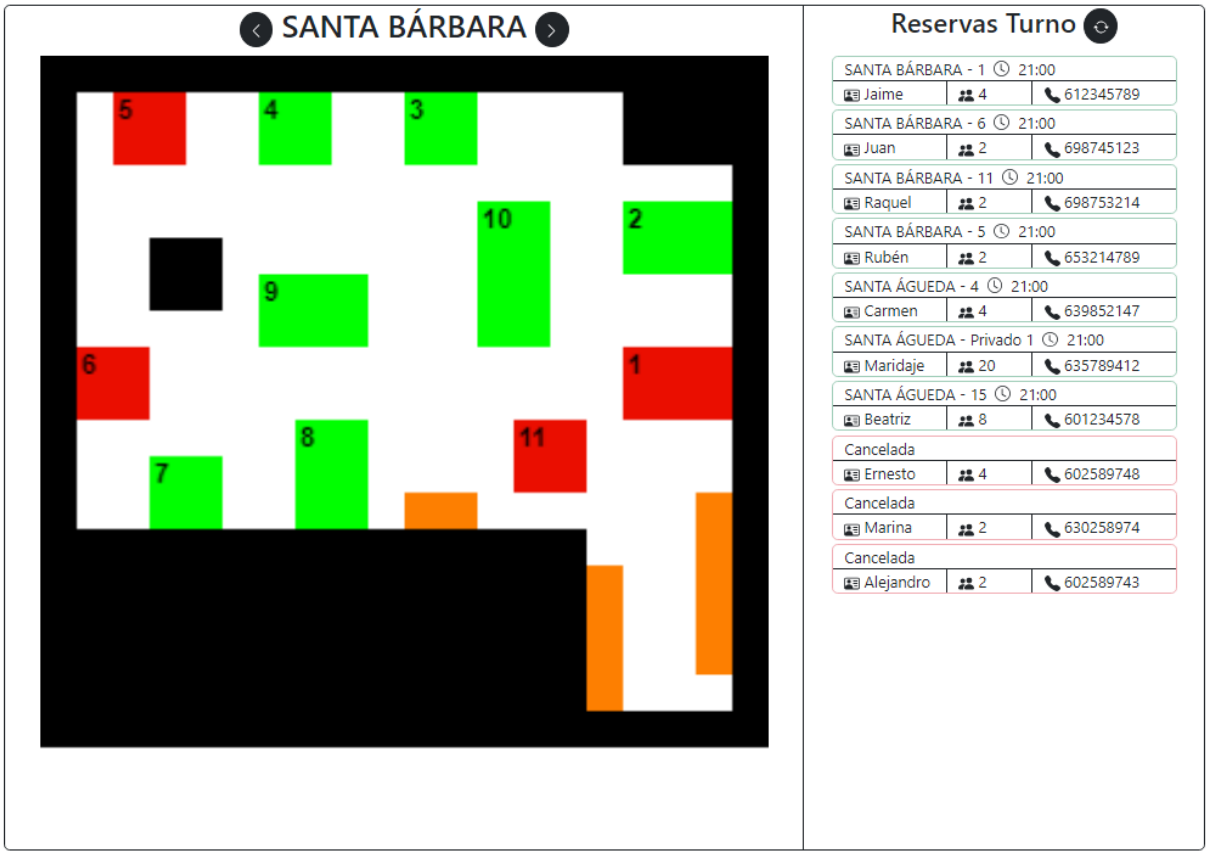


Figura 57 - Manual: ficha de comedor

GESTIÓN DE RESERVAS

Creación de reservas

Al iniciar el proceso de creación de reserva a través del botón (2), emergerá un pop-up con un formulario donde deberá rellenar los campos obligatorios:

- Nombre
- Comensales
- Teléfono
- Fecha (Por defecto día en el que se encuentra el usuario)
- Hora
- Tipo de reserva (Por defecto tipo Normal)
- Cuartos de cordero (Por defecto valor 0)

Figura 58 - Manual: formulario de creación de reserva

Pulsando el botón continuar, podrán ocurrir 3 situaciones:

1. Si el tipo de reserva es privada: el sistema comprobará si es posible cerrar la sala privada designada y comunicará si se ha creado la reserva o si no ha sido posible.
2. Si el tipo de reserva es normal:
   2.1. Si existe una posición que pueda abarcar el número de comensales seleccionado, el sistema mostrará al usuario las diferentes posiciones donde podrá colocar la reserva.

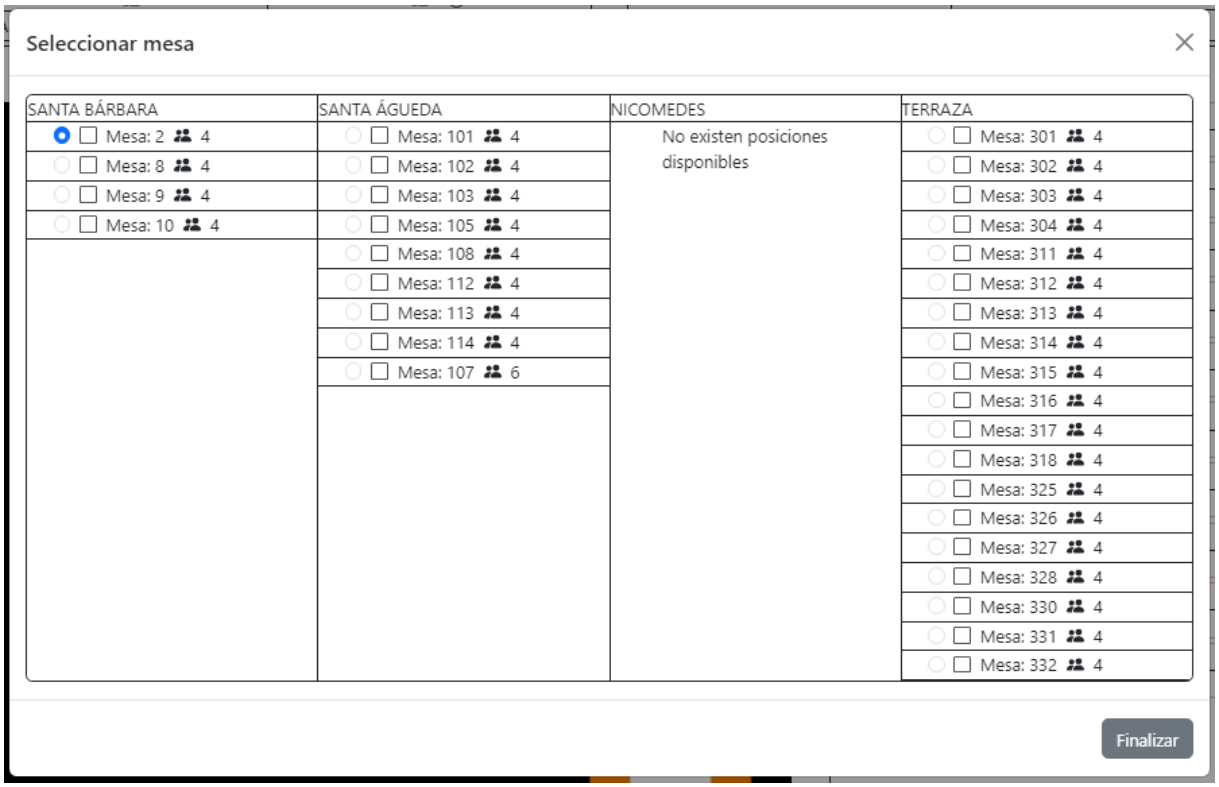


Figura 59 - Manual: selector de mesa simple

2.2. Si no existe una posición que pueda abarcar el número de comensales seleccionado, el sistema mostrará al usuario las diferentes combinaciones de posiciones donde podrá colocar la reserva.

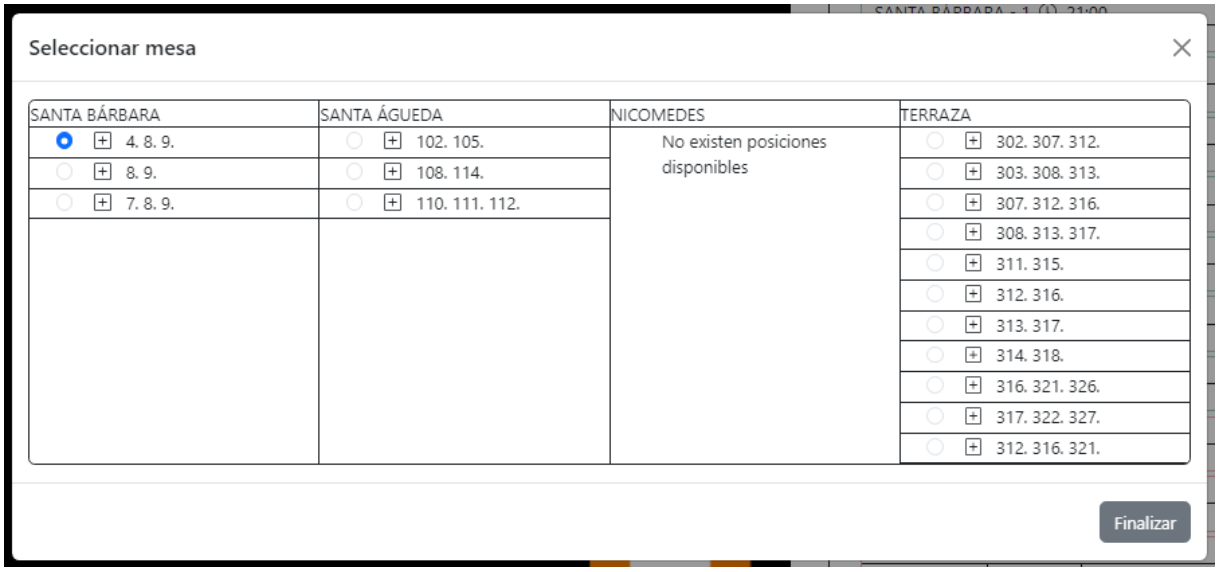


Figura 60 - Manual: selector de mesa combinada

Pulsando finalizar, se creará la reserva y el sistema comunicará si ha habido éxito en la creación o informará del error.

Visualización y modificación de reserva

Si el usuario desea visualizar todos los datos de una reserva, podrá pulsar sobre una ficha de reserva (7) de la lista de reservas, y se desplegará un pop-up con la información de la reserva

Figura 61 - Manual: vista de detalle de reserva

Dentro de esta ventana emergente, el usuario podrá pulsar los botones situados en la parte inferior.

Pulsando el botón de Modificar se desplegará un formulario con los datos que el sistema permite cambiar de una reserva. Pulsando el botón de continuar, los datos actualizados se modificarán en la base de datos.

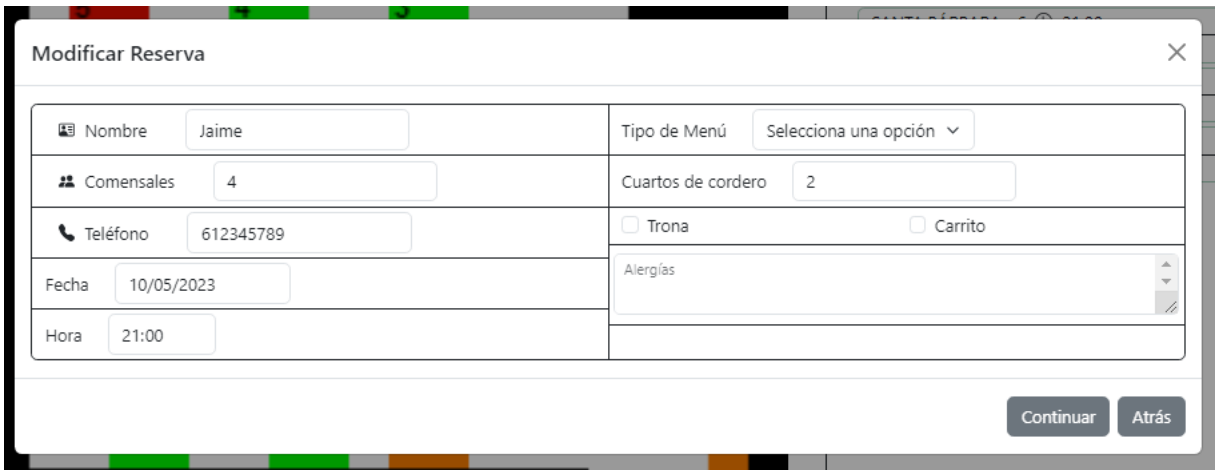


Figura 62 - Manual: formulario de modificación de reserva

Pulsando el botón de cambiar posición, se volverá a desplegar el selector de posiciones. Al elegir una nueva posición, ésta se le asignará a la reserva con la que el usuario está trabajando.

Por último, el usuario puede cancelar las reservas pulsando el botón de cancelar.

CAPITULO 9. CONCLUSIÓN Y FUTURAS AMPLIACIONES

CONCLUSIÓN

El primer aspecto que me gustaría destacar una vez acabado el proyecto, ha sido el aspecto autodidacta. He aprendido mucho informándome a través de Internet y en el ámbito laboral sobre JavaScript, el lenguaje utilizado en el controlador del lado cliente. Es un hecho que este lenguaje ha evolucionado muchísimo y cada vez se usa más en los desarrollos de tecnologías web. Es un lenguaje que se ve muy poco en grado universitario y que, desde mi punto de vista, debería verse más por la demanda que existe del conocimiento de este lenguaje en el mundo laboral.

Por otro lado, uno de los aspectos que han marcado el proyecto es la falta de experiencia en la gestión, planificación y desarrollo de proyectos reales. Estos proyectos no están sujetos y limitados por unos controles y pautas impuestos, como en el caso de un proyecto académico. Esto ha supuesto grandes retos a la hora de fijar objetivos y analizar los requisitos necesarios para satisfacerlos, debido a que se planteaba un requisito, y más adelante surgía un detalle que no se había tenido en cuenta y la solución propuesta no tenía la capacidad de cubrir, obligando a buscar un nuevo enfoque consumiendo una gran cantidad de tiempo.

También destacar que el proyecto ha sido más complejo de lo esperado, y estos cambios que han surgido durante el planteamiento han hecho que se descartaran algunos aspectos para ampliar más adelante. Este proyecto servirá de primera entrega al cliente, con el fin de que realicen pruebas y pidan correcciones y nuevas peticiones que, junto con mis ideas, formen un backlog de tareas y puedan tomar forma de nuevo proyecto en el que trabajar con ello.

Sin duda, este proyecto ha representado un desafío personal en todos los aspectos. Lo que inicialmente parecía ser algo sencillo, se convirtió en un proyecto de gran complejidad. En numerosas ocasiones, sentí que no avanzaba, que no veía el final, después de tantas horas de investigación, pruebas fallidas, modificaciones y la sensación de estar solo.

Sin embargo, al final logré superarlo. Este proyecto me ha hecho más fuerte, despertando en mí un interés por seguir aprendiendo en el campo del desarrollo de aplicaciones y transferir ese conocimiento a futuros proyectos, incluyendo éste en particular. También he comprendido la importancia de contar con un equipo sólido que brinde apoyo y alivie la carga en las diferentes etapas de desarrollo, especialmente en proyectos de esta envergadura.

A pesar de todo, he adquirido nuevos conocimientos y experiencias que me serán útiles en futuros desafíos. Hay áreas en las que deseo profundizar y estoy seguro de que, con tiempo y determinación, podré lograrlo sin dificultades en un futuro cercano.

FUTURAS AMPLIACIONES

Cómo se ha mencionado antes, existen desarrollos que no llegaron a implementarse debido al cambio de complejidad de algunas tareas y se quedaron fuera del alcance del proyecto; y también algunos que se han planteado, pero no entraron en ningún momento en el alcance del proyecto. Entre estos podemos destacar:

- Creación de perfiles con diferentes funcionalidades disponibles.
- Crear funcionalidad de recolocación automática de todas las reservas (respetando la preferencia de las mismas).
- Permitir la exportación e importación de reservas a través de csv.
- Conectar con los diferentes métodos que tiene el restaurante para crear solicitudes de reservas en estado pendiente y que posteriormente se acepten.
- Que las mesas se coloquen automáticamente. Para esto habría que negociar con el restaurante, ya que la elección de mesas era un requisito. Pero se podría dejar la posibilidad de cambiar de mesa y colocar de manera automática las reservas. De esta manera la creación de reservas siempre sería de un único paso para el trabajador.
- Crear otra página de acceso público que permita solicitar reservas y las cree en la base de datos en estado pendiente, además en estas páginas se podría adjuntar los diferentes menús y la carta.
- Hacer que se pueda interactuar con el plano, moviendo las posiciones o pulsándolas para crear en dicha posición una nueva reserva.

REFERENCIAS

Página oficial de Visual Studio Code: utilizada para descargar e instalar Visual Studio Code.

- Visual Studio Code. Disponible en https://code.visualstudio.com/ (Visitado en Febrero del 2022).

Página oficial de Apache: utilizada para descargar e instalar XAMPP.

- Apache Friends. Disponible en https://www.apachefriends.org/es/index.html (Visitado en febrero del 2022).

Página oficial de GitHub: utilizada para descargar e instalar GitHub.

- GitHub. Disponible en https://github.com/ (Visitado en abril del 2022).

Páginas de aplicaciones similares: utilizadas para coger ideas e información sobre la competencia existente.

- Restoo. Disponible en https://resto.me (Visitado en febrero del 2022)
- Resmio. Disponible en https://resmio.com (Visitado en febrero del 2022)

Página de FlatIcon: utilizada para descargar iconos e imágenes.

- FlatIcon. Disponible en https://www.flaticon.com (Visitado en junio del 2023).

Página oficial de Bootstrap: utilizada para aprender y utilizar sobre como importar Bootstrap a la aplicación web para poder darle estilos de forma sencilla.

- Bootstrap. Disponible en https://www.bootstrap.com (Visitado en junio del 2023).

Foro de StackOverflow: utilizada para consultar dudas y buscar solución de errores en el código.

- StackOverflow. Disponible en https://stackoverflow.com/ (Visitado en junio del 2023).

Manual de PHP: utilizada para aprender y consultar dudas sobre el lenguaje de PHP.

- PHP. Disponible en https://www.php.net/manual/es/index.php (Visitado en mayo del 2023).

Escuela de W3School: utilizada para aprender y consultar dudas sobre los diferentes lenguajes utilizados.

- W3School. Disponible en https://www.w3schools.com/ (Visitado en junio del 2023).