UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"La Carta Mia. Conjunto de aplicaciones para la gestión de carta y pedidos de un restaurante"

TRABAJO DE FIN DE GRADO

Diciembre – 2025

AUTOR: Víctor López Fornés

DIRECTOR: Antonio Peñalver Benavent

Agradecimientos

Hace 8 años, cuando empecé a estudiar informática, con muchas dudas y sin saber claramente lo que me apasionaba o a lo que me quería dedicar en mi carrera profesional. Después de cuatro años de ciclos formativos y cuatro años de este grado, quiero agradecer a todo el personal docente por guiarme en este proceso de descubrimiento de lo que es hoy en día, tanto mi profesión, como mi afición.

Quiero agradecer en especial a mi madre, que ha estado siempre presente en mis momentos de estrés académico, por estar siempre a mi lado después de cada jornada de estudio o de trabajo, por ayudarme económicamente cuando me ha hecho falta para poder seguir estudiando.

Quiero agradecer también a mi pareja, por ayudarme en cada bajón, por apoyarme antes de cada examen o tarea difícil y por estar conmigo en todos mis triunfos, por estar siempre ahí para mí a pesar de todo.

Quiero agradecer a mis compañeros de grado, por esas largas tardes en las que nos juntábamos para estudiar o resolver problemas en común, por las risas entre ejercicios que aliviaban el cansancio, por el apoyo mutuo en los momentos de más presión, y por demostrarme que el esfuerzo compartido no solo hace más llevadero el camino, sino que también lo llena de recuerdos.

Quiero agradecer a todo el cuerpo docente de la universidad, por las asignaturas que tanto me han interesado y por todo su entusiasmo a la hora de enseñar tu temario.

Por último, pero no más importante quiero agradecer a mi tutor Antonio Peñalver por aceptarme bajo su tutela en este proyecto, por su orientación y dedicación constante.

Resumen

En este trabajo de fin de grado he desarrollado un sistema compuesto por tres aplicaciones destinadas a la gestión de la carta online y los pedidos de un restaurante ficticio llamado *La Carta Mia.* El objetivo principal ha sido diseñar una solución tecnológica que cubre tanto la administración interna del restaurante como la experiencia de los clientes a la hora de consultar la carta y realizar pedidos.

Para ello, se han implementado un conjunto de tres aplicaciones, las cuales tienen las siguientes características:

- La primera de ellas es una página web desarrollada en Spring Boot, encargada de gestionar la carta del restaurante. Esta aplicación permite el acceso como usuario anónimo para la consulta de platos disponibles y, a su vez, incorpora un sistema de autenticación que habilita a los administradores la modificación de dicha carta. Además, esta aplicación funciona como servicio RESTful, proporcionando los datos necesarios a las demás aplicaciones.
- La segunda aplicación, muestra la carta de manera interactiva y permite la realización de pedidos. Estos pedidos se guardan online en Firebase, comunicándose con la tercera aplicación.
- La tercera aplicación consume los datos generados por la segunda aplicación y los gestiona, mostrando los actuales, pudiendo modificar sus estados (entregado, pagado, etc.). Además, tendremos un histórico de todos los pedidos realizados, para una posible contabilidad.

Índice

CAPÍTULO 1: INTRODUCCIÓN 14
1.1. Contexto y motivación del proyecto 15
1.2. Objetivos generales y específicos 16
1.3. Metodología de trabajo 16
1.4. Estructura del documento 18
CAPÍTULO 2: ANTECEDENTES Y SOLUCIONES SIMILARES 20
2.1. Sistemas actuales de gestión de pedidos 21
2.2. Tecnologías empleadas en el sector 22
2.3. Comparativa con soluciones existentes 23
2.4. Justificación de la propuesta 24
CAPÍTULO 3: OBJETIVOS 26
3.1. Objetivos generales 27
3.2. Objetivos didácticos 27
3.3. Objetivos personales 28
CAPÍTULO 4: ANÁLISIS DE REQUISITOS 30
4.1. Requisitos funcionales 31
4.1.1. Aplicación web 31
4.1.2. Aplicación de pedidos (cliente) 33
4.1.3. Aplicación de gestión de pedidos (interna) 36
4.2.1. Aplicación web 37
4.2.2. Aplicación de pedidos (cliente) 39
4.2.3. Aplicación de gestión de pedidos (interna) 40
4.3. Casos de uso 42
4.3.1. Aplicación web 42
4.3.2. Aplicación de pedidos (cliente) 57
4.3.3. Aplicación de gestión de pedidos (interna) 66
4.4. Diagramas UML 72

4.4.1. Aplicación web 72
4.4.2. Aplicación de pedidos (cliente) 77
4.4.3. Aplicación de gestión de pedidos (interna) 83
CAPÍTULO 5: DISEÑO DE LA SOLUCIÓN 87
5.1. Arquitectura general del sistema 88
5.2. Descripción de las aplicaciones 89
5.2.1. Aplicación web en Spring Boot (gestión de carta y servicio RESTFUL) 89
5.2.2. Aplicación para clientes (realización de pedidos) 91
5.2.3. Aplicación de gestión de pedidos (pedidos actuales e histórico) 93
5.3. Modelado de datos y base de datos 95
5.3.1. Modelo relacional (carta y usuarios web) 95
5.4 Diseño de interfaces gráficas 98
5.4.1. Aplicación web 98
5.4.2. Aplicación de pedidos (cliente) 103
5.4.3. Aplicación de gestión de pedidos (interna) 106
5.5. Seguridad y gestión de usuarios 109
CAPÍTULO 6: DESARROLLO E IMPLEMENTACIÓN 111
6.1. Tecnologías y herramientas utilizadas 112
6.1.1 Entornos de desarrollo 112
6.1.2. Tecnologías de backend 112
6.1.3. Tecnologías de frontent web 113
6.1.4. Tecnologías de desarrollo móvil 114
6.1.5. Infraestructura y despliegue 114
6.1.6. Herramientas de control y colaboración 115
6.2. Desarrollo de la aplicación web 115
6.2.1. Tecnologías empleadas 115
6.2.2. Estructura del proyecto 116
6.2.3. Seguridad y gestión de usuarios 117
6.2.4. Servicio REST 117

6.2.5. Despliegue y configuración del entorno 118
6.3. Desarrollo de la aplicación cliente 118
6.3.1. Arquitectura general 118
6.3.2. Comunicación con el backend 119
6.3.3. Gestión del carrito y patrón Singletone 120
6.3.4. Persistencia de pedidos en Firebase Firestore 120
6.3.5. Renderizado de imágenes con Picasso 121
6.3.6. Diseño visual y experiencia de usuario 121
6.4. Desarrollo de la aplicación de gestión de pedidos 121
6.4.1. Diseño visual y experiencia de usuario 122
6.4.2. Flujo principal 122
6.4.3. Actualización en tiempo real 122
6.5. Integración entre las tres aplicaciones 123
6.6. Problemas encontrados y soluciones adoptadas 123
6.6.1. Aplicación web 123
6.6.2. Aplicación pedidos cliente 124
6.6.3. Aplicación gestión de pedidos 125
CAPÍTULO 7: VALIDACIÓN DEL SISTEMA 126
7.1. Estrategia de pruebas manuales 127
7.2. Escenarios de prueba 127
7.2.1. Aplicación web 128
7.2.2. Aplicación pedidos (cliente) 129
7.2.3. Aplicación gestión de pedidos (interna) 130
7.3. Resultados de la validación 131
CAPÍTULO 8: CONCLUSIONES Y TRABAJO FUTURO 132
8.1. Conclusiones principales 133
8.2. Aportaciones del proyecto 133
8.3. Limitaciones del sistema 134
8.4. Posibles mejoras y líneas de trabajo futuro 134

8.4.1. Aplicación web 134
8.4.2. Aplicación pedidos (cliente) 135
8.4.3. Aplicación gestión de pedidos (interna) 136
CAPÍTULO 9: BIBLIOGRAFÍA 137
ANEXOS 143
Anexo I. Aplicación web (Spring Boot - Gestión de la carta y servicio RESTful) 144
Manual de usuario 144
Anexo II. Aplicación cliente (visualización de carta y realización de pedidos) 155
Manual de usuario 155
Anexo III. Aplicación de gestión de pedidos 161
Manual de usuario 161

Índice de Ilustraciones

Ilustración 1. Diagrama de Gantt del proyecto 72
Ilustración 2. Diagrama de clases - Aplicación web - Paquete principal 73
Ilustración 3. Diagrama de clases - Aplicación web - Paquete controladores rest 73
Ilustración 4. Diagrama de clases - Aplicación web - Paquete de servicios y sus implementaciones 74
Ilustración 5. Diagrama de clases - Aplicación web - Paquete de entidades 75
Ilustración 6. Diagrama de clases - Aplicación web - Paquete de controladores 75
Ilustración 7. Diagrama de secuencia - Aplicación web - Creación de plato 76
Ilustración 8. Diagrama de secuencia - Aplicación web - Modificación de plato 76
Ilustración 9. Diagrama de secuencia - Aplicación web - Eliminación de plato 77
Ilustración 10. Diagrama de clases - Aplicación cliente - Paquete principal 77
Ilustración 11. Diagrama de clases - Aplicación cliente - Paquete de modelos 78
Ilustración 12. Diagrama de clases - Aplicación cliente - Paquete managers 78
Ilustración 13. Diagrama de clases - Aplicación cliente - Paquete REST 78
Ilustración 14. Diagrama de clases - Aplicación cliente - Paquete fragments 79
Ilustración 15. Diagrama de clases - Aplicación cliente - Paquete adapters 80
Ilustración 16. Diagrama de clases - Aplicación cliente - Paquete dialogs 80
Ilustración 17. Diagrama de clases - Aplicación cliente - Paquete interfaces 81
Ilustración 18. Diagrama de secuencia - Aplicación cliente - Conexión con servicio REST 81
Ilustración 19. Diagrama de secuencia - Aplicación cliente - Inicio de sesión como administrador 82
Ilustración 20. Diagrama de secuencia - Aplicación cliente - Flujo normal de la aplicación 82
Ilustración 21. Diagrama de clase - Aplicación gestión de pedidos - Paquete principal 83
Ilustración 22. Diagrama de clase - Aplicación gestión de pedidos - Paquete de interfaces 84
Ilustración 23. Diagrama de clase - Aplicación gestión de pedidos - Paquete de fragments 84
Ilustración 24. Diagrama de clase - Aplicación gestión de pedidos - Paquete de enums ... 85
Ilustración 25. Diagrama de clase - Aplicación gestión de pedidos - Paquete de adapters 85
Ilustración 26. Diagrama de secuencia - Aplicación gestión de pedidos 86
Ilustración 27. Diagrama BBDD aplicación Web Spring Boot 96
Ilustración 28. Modelo de datos de pedidos en Firebase 98
Ilustración 29. Wireframe página principal aplicación web 99
Ilustración 30. Wireframe listado de categorías aplicación web 100

Ilustración 31. Listado de platos aplicación web 100
Ilustración 32. Wireframe detalle de plato aplicación web 101
Ilustración 33. Wireframe detalle categoría aplicación web 101
Ilustración 34. Wireframe detalle plato aplicación web..... 102
Ilustración 35. Wireframe nueva categoría aplicación web..... 102
Ilustración 36. Wireframe inicio de sesión aplicación web 103
Ilustración 37. Wireframe listado de categorías aplicación móvil clientes 104
Ilustración 38. Wireframe listado de platos aplicación móvil clientes 105
Ilustración 39. Wireframe detalle de plato aplicación móvil clients 105
Ilustración 40. Wireframe carrito aplicación pedidos clientes 106
Ilustración 41. Wireframe página de inicio de sesión aplicación gestión de pedidos 107
Ilustración 42. Wireframe página de listado de pedidos aplicación de gestión de pedidos 108
Ilustración 43: Wireframe detalle pedido aplicación gestión de pedidos 108
Ilustración 44. Página principal - Aplicación web..... 144
Ilustración 45. Listado de platos (usuario anónimo) – Aplicación web 144
Ilustración 46. Página de inicio de sesión - Aplicación web 145
Ilustración 47. Inicio de sesión fallido - Aplicación web 145
Ilustración 48. Inicio de sesión correcto - Aplicación web 146
Ilustración 49. Lista de platos como usuario común - Aplicación web..... 146
Ilustración 50. Detalle plato - Aplicación web 147
Ilustración 51. Detalle de una categoría - Aplicación web 147
Ilustración 52. Desplegable para cerrar sesión - Aplicación web 147
Ilustración 53. Inicio de sesión como administrador - Aplicación web 148
Ilustración 54. Listado de platos como usuario administrador - Aplicación web 148
Ilustración 55. Formulario creación de un plato 149
Ilustración 56. Creación nuevo plato (información errónea) - Aplicación web..... 149
Ilustración 57. Comprobación existencia nuevo plato - Aplicación web 150
Ilustración 58. Formulario editar plato - Aplicación web 150
Ilustración 59. Listado de platos después de modificar - Aplicación web 151
Ilustración 60. Detalle plato de prueba - Aplicación web..... 151
Ilustración 61. Resultado eliminación de plato - Aplicación web 152
Ilustración 62. Creación de categoría - Aplicación web..... 153
Ilustración 63. Listado con categoría de prueba creada - Aplicación web..... 153
Ilustración 64. Modificación categoría de prueba - Aplicación web..... 154
Ilustración 65. Comprobación en listado de modificación - Aplicación web 154
Ilustración 66. Detalle de categoría después de ser modificada 154
Ilustración 67. Comprobación eliminación de categoría - Aplicación web..... 155

Ilustración 68. Listado de categorías - Aplicación cliente 156
Ilustración 69. Listado de platos de una categoría - Aplicación cliente 156
Ilustración 70. Modal añadir plato - Aplicación cliente 157
Ilustración 71. Detalle de cliente - Aplicación cliente 157
Ilustración 72. Carrito lleno – Aplicación cliente 158
Ilustración 73. Comprobación número en el carrito – Aplicación cliente 158
Ilustración 74. Carrito después del borrado 159
Ilustración 75. Modal borrado de plato - Aplicación cliente 159
Ilustración 76. Modal inicio de sesión - Aplicación cliente 160
Ilustración 77. Menú para iniciar sesión 160
Ilustración 79. Menú de administrador - Aplicación cliente 160
Ilustración 78. Modal cambio número de mesa - Aplicación cliente 160
Ilustración 80. Pantalla de inicio de sesión - Aplicación gestión de pedidos 161
Ilustración 81. Listado de pedidos actuales - Aplicación gestión de pedidos 162
Ilustración 82. Modal confirmación confirmar pago - Aplicación gestión de pedidos 162
Ilustración 83. Detalle de pedido - Aplicación gestión de pedidos 163
Ilustración 84. Menú lateral - Aplicación gestión de pedidos 164
Ilustración 85. Histórico de pedidos - Aplicación gestión de pedidos 164

Índice de tablas

Tabla 1. Requisito funcional RF-WEB-1: Acceder a la página principal 31
Tabla 2. Requisito funcional RF-WEB-2: Autenticación de usuarios 31
Tabla 3. Requisito funcional RF-WEB-3: Alta de nuevos platos 31
Tabla 4. Requisito funcional RF-WEB-4: Modificación de platos 31
Tabla 5. Requisito funcional RF-WEB-5: Eliminación de platos 32
Tabla 6. Requisito funcional RF-WEB-6: Alta de nuevas categorías 32
Tabla 7. Requisito funcional RF-WEB-7: Modificación de categorías 32
Tabla 8. Requisito funcional RF-WEB-8: Eliminación de categorías 32
Tabla 9. Requisito funcional RF-WEB-9: Almacenamiento en BBDD 32
Tabla 10. Requisito funcional RF-WEB-10: Obtención de categorías en formato JSON 32
Tabla 11. Requisito funcional RF-WEB-11: Obtención de platos en formato JSON 33
Tabla 12. Requisito funcional RF-CLI -1: Visualización de categorías 33
Tabla 13. Requisito funcional RF-CLI -2: Visualización de platos por categoría 33
Tabla 14. Requisito funcional RF-CLI-3: Visualización del detalle del plato 33
Tabla 15. Requisito funcional RF-CLI-4: Añadir plato al carrito 33
Tabla 16. Requisito funcional RF-CLI-5: Visualización del carrito 34
Tabla 17. Requisito funcional RF-CLI -6: Modificación del carrito 34
Tabla 18. Requisito funcional RF-CLI-7: Autenticación administrativa 34
Tabla 19. Requisito funcional RF-CLI -8: Selección de número de mesa 34
Tabla 20. Requisito funcional RF-CLI-9: Cierre de sesión administrador 34
Tabla 21. Requisito funcional RF-CLI-10: Realización del pedido 35
Tabla 22. Requisito funcional RF-CLI-11: Verificación de pedido activo 35
Tabla 23. Requisito funcional RF-CLI-12: Creación de un nuevo pedido 35
Tabla 24. Requisito funcional RF-CLI-13: Creación de comanda 35
Tabla 25. Requisito funcional RF-CLI-14: Actualización de estado del pedido 35
Tabla 26. Requisito funcional RF-GES-1: Autenticación de administradores 36
Tabla 27. Requisito funcional RF-GES-2: Menú principal 36
Tabla 28. Requisito funcional RF-GES-3: Visualización de pedidos actuales 36
Tabla 29. Requisito funcional RF-GES-4: Visualización de histórico de pedidos 36
Tabla 30. Requisito funcional RF-GES-5: Modificación del estado del pedido 36
Tabla 31. Requisito funcional RF-GES-6: Cambio de estado mediante interacción visual . 37
Tabla 32. Requisito funcional RF-GES-7: Actualización en tiempo real 37
Tabla 33. Requisito funcional RF-GES-8: Cerrar sesión 37
Tabla 34. Requisito no funcional RNF-WEB-1: Escalabilidad 37
Tabla 35. Requisito no funcional RNF-WEB-2: Usabilidad 38

Tabla 36. Requisito no funcional RNF-WEB-3: Compatibilidad 38
Tabla 37. Requisito no funcional RNF-WEB-4: Seguridad 38
Tabla 38. Requisito no funcional RNF-WEB-5: Rendimiento 38
Tabla 39. Requisito no funcional RNF-WEB-6: Mantenibilidad 38
Tabla 40. Requisito no funcional RNF-CLI-1: Usabilidad 39
Tabla 41. Requisito no funcional RNF-CLI-2: Rendimiento 39
Tabla 42. Requisito no funcional RNF-CLI-3: Compatibilidad 39
Tabla 43. Requisito no funcional RNF-CLI-4: Seguridad 39
Tabla 44. Requisito no funcional RNF-CLI-5: Robustez 40
Tabla 45. Requisito no funcional RNF-CLI-6: Escalabilidad 40
Tabla 46. Requisito no funcional RNF-GES-1: Seguridad 40
Tabla 47. Requisito no funcional RNF-GES-2: Usabilidad 40
Tabla 48. Requisito no funcional RNF-GES-3: Rendimiento 40
Tabla 49. Requisito no funcional RNF-GES-4: Robustez 41
Tabla 50. Requisito no funcional RNF-GES-5: Escalabilidad 41
Tabla 51. Requisito no funcional RNF-GES-6: Compatibilidad 41
Tabla 52. Requisito no funcional RNF-GES-7: Mantenibilidad 41
Tabla 53. Requisito no funcional RNF-GES-8: Disponibilidad 41
Tabla 54. Caso de Uso C.U.WEB-1: Acceder a la página principal 42
Tabla 55. Caso de Uso C.U.WEB-2: Acceder al listado de platos 43
Tabla 56. Caso de Uso C.U.WEB-3: Acceder al listado de categorías 43
Tabla 57. Caso de Uso C.U.WEB-4: Iniciar sesión 44
Tabla 58. Caso de Uso C.U.WEB-5: Cerrar sesión 45
Tabla 59. Caso de Uso C.U.WEB-6: Acceder al detalle del plato 46
Tabla 60. Caso de Uso C.U.WEB-7: Acceder a detalle de categoría 47
Tabla 61. Caso de Uso C.U.WEB-8: Gestión de platos 48
Tabla 62. Caso de Uso C.U.WEB-9: Creación de platos 49
Tabla 63. Caso de Uso C.U.WEB-10: Modificación de platos 50
Tabla 64. Caso de Uso C.U.WEB-11: Eliminación de platos 51
Tabla 65. Caso de Uso C.U.WEB-12: Gestión de categorías 52
Tabla 66. Caso de Uso C.U.WEB-13: Creación de categorías 53
Tabla 67. Caso de Uso C.U.WEB-14: Modificación de categorías 54
Tabla 68. Caso de Uso C.U.WEB-15: Eliminación de categorías 55
Tabla 69. Caso de Uso C.U.WEB-16: Acceso a servicios RESTful 56
Tabla 70. Casos de uso C.U.CLI-1: Visualizar listado de categorías (Pantalla principal) 57
Tabla 71. Casos de uso C.U.CLI-2: Visualizar platos de una categoría 58
Tabla 72. Casos de uso C.U.CLI-3: Visualizar detalle de un plato 59
Tabla 73. Casos de uso C.U.CLI-4: Añadir plato al carrito 60

Tabla 74. Casos de uso C.U.CLI-5: Consultar carrito 61
Tabla 75. Casos de uso C.U.CLI-6: Confirmar pedido 62
Tabla 76. Casos de uso C.U.CLI-7: Iniciar sesión como administrador 63
Tabla 77. Casos de uso C.U.CLI-8: Cambiar número de mesa 64
Tabla 78. Casos de uso C.U.CLI-9: Cerrar sesión como administrador 65
Tabla 79. Caso de uso C.U.GES-1: Iniciar sesión 66
Tabla 80. Caso de uso C.U.GES-2: Visualizar pedidos actuales 67
Tabla 81. Caso de uso C.U.GES-3: Visualizar pedidos históricos 68
Tabla 82. Caso de uso C.U.GES-4: Marcar pedido como pagado / no pagado 69
Tabla 83. Caso de uso C.U.GES-5: Visualizar detalle del pedido 70
Tabla 84. Caso de uso C.U.GES-6: Marcar comanda como entregada 71

CAPÍTULO 1: INTRODUCCIÓN

1.1. Contexto y motivación del proyecto

La pandemia mundial de COVID-19 nos mostró la necesidad de minimizar el contacto directo entre clientes y personal de hostelería, tanto por motivos de seguridad sanitaria como por eficiencia [1].

En este contexto, los restaurantes se vieron obligados a adaptarse para garantizar un servicio seguro y ágil, a la vez que buscaban reducir sus costes derivados del personal, ya que el número de clientes posibles era menor por restricciones asfixiantes de aforo [2].

Este proyecto surge como respuesta a esa necesidad, proponiendo un sistema de aplicaciones para la gestión de carta y pedidos para un restaurante, en este caso uno ficticio llamado *La Carta Mia*. Este sistema combina tres aplicaciones que permiten:

- Administrar la carta de manera digital, evitando la necesidad de contacto físico con menús impresos (además de ahorrarnos el coste de imprimir nuevas cartas cada vez que cambia un plato, precio o cualquier dato del plato) [3].
- Facilitar a los clientes la realización de pedidos a través de una aplicación sencilla e intuitiva [25][26], reduciendo al mínimo cualquier interacción directa con el personal del restaurante [4].
- A su vez, al restaurante le permite gestionar los pedidos de forma eficiente, permitiendo controlar tanto los pedidos actuales como el historial de manera digital.

El desarrollo de esta solución tiene como motivación principal mejorar la seguridad y la eficiencia en el servicio del restaurante, al mismo tiempo que ofrecer una experiencia cómoda y moderna para los clientes y una reducción de costes al empresario [5].

1.2. Objetivos generales y específicos

Objetivo general

El objetivo general de este proyecto es desarrollar un sistema tecnológico que permita la gestión digital de la carta y de los pedidos de un restaurante, optimizando al máximo la experiencia del cliente, minimizando el contacto físico y mejorando la eficiencia.

Objetivos específicos

- Desarrollar una aplicación web con Spring Boot que gestione la carta y funcione como servicio RESTful para las demás aplicaciones.
- Implementar una aplicación cliente que permita consultar la carta y realizar pedidos de forma clara y sencilla [6].
- Crear una aplicación de gestión de pedidos que permita visualizar los pedidos actuales, consultar los pedidos del histórico y controlar el flujo de los pedidos, cambiándolos de estado, etc.
- Integrar las tres aplicaciones de manera que la información se comparta entre ellas en tiempo real.
- Validar el conjunto de aplicaciones a través de pruebas manuales que garanticen la correcta funcionalidad y experiencia de usuario.

1.3. Metodología de trabajo

El desarrollo de este proyecto se ha llevado a cabo siguiendo un enfoque progresivo y estructurado, basado en la definición de responsabilidades, diseño modular, implementación por fases y validación mediante pruebas manuales. Esta metodología de trabajo me ha permitido garantizar la coherencia entre el conjunto de las tres aplicaciones del sistema y asegurar cumplir mis objetivos [8], siguiendo principios de usabilidad y experiencia de usuarios para garantizar aplicaciones intuitivas y fáciles de usar [25][26].

El desarrollo se ha basado en los principios de la ingeniería del software definidos por Sommerville [37] y Pressman [38], siguiendo un enfoque iterativo e incremental.

Fase 1: Análisis de requisitos

Antes de comenzar la implementación, definí las funcionalidades principales de cada aplicación, separando claramente las responsabilidades de cada una de ellas, para así asegurar un correcto funcionamiento y seguridad.

- La aplicación web se encarga de la gestión de la carta y proporciona un servicio RESTful que mantiene la información centralizada y actualizada en todo el conjunto de aplicaciones.
- La aplicación cliente permite consultar la carta y realizar pedidos.
- La aplicación de gestión de pedidos permite visualizar los pedidos y gestionarlos.

Fase 2: Diseño del sistema

Se definió la arquitectura general del sistema en tres aplicaciones conectadas entre sí: la web con el servicio REST como núcleo, la aplicación para realizar pedidos y la de gestión de estos (conectadas a través de una base de datos NoSQL en Firebase para almacenar los pedidos) [9]. Además, la disposición y navegación de las interfaces fue diseñada siguiendo principios de usabilidad para facilitar la interacción de los usuarios finales [27][28].

Se definieron ciertos diagramas de clase UML para la creación de los platos, categorías, pedidos, comandas, etc.

Fase 3: Implementación

La implementación se realizó de manera progresiva:

1. Aplicación web: desarrollo del sistema de gestión de carta y del servicio RESTful en Spring Boot, ya que las demás dependen de esta.
2. Aplicación cliente: Desarrollo de la aplicación para realizar pedidos utilizando los datos proporcionados por el servicio REST.
3. Aplicación de gestión de pedidos: desarrollo del sistema para visualizar los pedidos, integrando la información de los pedidos alojada en Firebase.

Se ha utilizado Firebase para el almacenamiento de pedidos debido a su flexibilidad al ser una base de datos no relacional, lo cual nos permite modificar la estructura de los pedidos en un futuro sin afectar los registros anteriores [10].

Fase 4: Validación y pruebas

Las pruebas se realizaron de forma manual siguiendo un orden progresivo con el objetivo de garantizar que la experiencia de usuario fuese fluida e intuitiva, validando los flujos de navegación, botones e interacción, cumpliendo los criterios de usabilidad [29][30]:

1. Se verifica el correcto funcionamiento de la aplicación web, comprobando la gestión de la carta y el funcionamiento correcto de los endpoints del servicio.
2. Se realizan las pruebas de la aplicación de pedidos, asegurando que los datos recibidos desde la web son correctos y que los pedidos se registran correctamente en Firebase.
3. Finalmente, se prueba la aplicación de gestión de pedidos, revisando tanto los actuales como los del historial.

Durante estas pruebas se identificaron y resolvieron problemas con las relaciones entre platos y categorías y se redefinió el concepto de "comanda" para los pedidos.

Fase 5: Documentación

Como parte del proyecto, se ha planificado la creación de manuales de usuario y técnicos para cada aplicación, incluyendo:

- Manual de instalación: pasos necesarios para poner en funcionamiento cada aplicación
- Manual de uso: instrucciones para los distintos tipos de usuarios (clientes, administradores y empleados).
- Manual técnico: detalles sobre la configuración y conexión entre las aplicaciones.

1.4. Estructura del documento

El documento se organiza en los siguientes capítulos:

1. **Introducción:** Presenta el contexto, la motivación, los objetivos y la metodología de trabajo, así como la organización general del TFG.

2. **Antecedentes y soluciones similares:** Analiza los sistemas existentes de gestión de pedidos en restauración, las tecnologías utilizadas y la justificación de la solución propuesta.
3. **Objetivos del proyecto:** Definimos los objetivos a cubrir realizando este proyecto, justificando la no inclusión de otros objetivos.
4. **Análisis de requisitos:** Define los requisitos funcionales y no funcionales del sistema, los casos de uso y los diagramas conceptuales que guían el diseño y desarrollo.
5. **Diseño de la solución:** Detalla la arquitectura del sistema, la descripción de las tres aplicaciones, el modelado de datos, el diseño de interfaces y aspectos de seguridad y gestión de usuarios.
6. **Desarrollo e implementación:** Explica las tecnologías y herramientas utilizadas, el desarrollo progresivo de cada aplicación, la integración entre ellas y los problemas encontrados con sus respectivas soluciones.
7. **Validación del sistema:** Describe la estrategia de pruebas manuales, los escenarios de uso elevados, los resultados obtenidos y las observaciones para posibles mejoras.
8. **Conclusiones y trabajo futuro:** Recoge las conclusiones principales, las aportaciones del proyecto, limitaciones detectadas y futuras ampliaciones y mejoras.
9. **Bibliografía:** Contiene referencias utilizadas durante el proyecto.
10. **Anexos:** Incluye, además de manuales técnicos y de usuarios, material complementario con fragmentos de código y diagramas adicionales.

CAPÍTULO 2: ANTECEDENTES Y SOLUCIONES SIMILARES

2.1. Sistemas actuales de gestión de pedidos

Debido a la digitalización del sector de hostelería, se ha impulsado el desarrollo de numerosas soluciones para la gestión de pedidos. Estas herramientas buscan optimizar la relación entre clientes y hosteleros, reduciendo tiempos de espera, evitando errores humanos y disminuyendo el contacto físico entre ambas partes [11].

Existen actualmente varias modalidades:

- **Tablets o kioscos de autoservicio:** Cadenas de restaurantes como McDonald's o Burger King han popularizado el uso de kioscos digitales en sus locales, permitiendo que el cliente seleccione los productos directamente en pantalla. En restaurantes más pequeños, también se han implementado soluciones parecidas, colocando tablets en las mesas que permiten consultar la carta y realizar el pedido [12].
- **Cartas digitales con código QR:** Tras la pandemia, se generalizó el uso de códigos QR que enlazan a una carta digital. Algunas de estas cartas solo incluyen información, mientras que otras permiten realizar los pedidos que se envían directamente a cocina [13].
- **Pedidos mediante Whatsapp:** Otra modalidad que se ha extendido, especialmente en negocios pequeños, los cuales no tienen tanto poderío adquisitivo para comprar un software específico. El cliente escanea un código QR o recibe un listado de platos numerados y envía un mensaje con los productos deseados. Plataformas como WhatsFood o MenuKite han aprovechado esta tendencia para ofrecer cartas digitales vinculadas a pedidos por el chat [14].

Inspiración personal

La motivación de este proyecto aparece precisamente por la observación de un bufé libre de mi zona, que había implementado un sistema de pedidos a través de WhatsApp. Los clientes enviaban los números de los platos y los camareros los servían en mesa.

Aunque se trataba de una solución sencilla y económica, presentaba varias limitaciones como la posibilidad de errores en la comunicación, dificultad de mantener la carta siempre actualizada y falta de un registro histórico de pedidos. Además, es fácil olvidar comandas o no tenerlas en cuenta, ya que no hay forma de ver visualmente si ya has entregado un pedido o no.

El sistema propuesto en La Carta Mia toma como referencia ese modelo, pero lo mejora con un enfoque más robusto y profesional:

- Una carta digital centralizada y gestionable por el administrador sin necesidad de habilidades técnicas.
- Aplicaciones móviles que permiten al cliente consultar la carta y realizar pedidos desde su mesa (con diseño para teléfono y para Tablet).
- Un sistema de gestión de pedidos para el hostelero, que facilita el seguimiento de comandas actuales y análisis del historial, además de tener la posibilidad de estadísticas de todo tipo, como los platos más vendidos.

2.2. Tecnologías empleadas en el sector

La digitalización de este sector ha impulsado el uso de diferentes tecnologías para mejorar la gestión de pedidos, optimizar procesos y mejorar la experiencia a nuestros clientes. Entre ellas, existen:

Terminales de Punto de Venta (TPV)

Los TPV se utilizan en la mayoría de los restaurantes actualmente. Permiten registrar pedidos, gestionar pagos y controlar inventario. Sin embargo, en muchos casos se trata de sistemas cerrados, dependientes de hardware específico y con costes de mantenimiento elevados [15].

Cartas digitales mediante códigos QR

Tras la pandemia, el uso de códigos QR se extendió a muchos restaurantes como alternativa a las cartas físicas. Estos permiten al cliente consultar los platos desde su propio dispositivo móvil, reduciendo el contacto directo [16].

En este tipo de casos, la carta se limita a la visualización, sin integrar ningún tipo de proceso de pedido o gestión avanzada [17].

Aplicaciones móviles de pedidos

Algunas cadenas han desarrollado aplicaciones móviles propias que permiten a los clientes realizar pedidos directamente desde su propio dispositivo, tanto en sala como en modalidad de recogida o a domicilio (estos dos últimos, estilo Glovo o JustEat).

Aunque ofrecen una experiencia mucho más completa, requieren de un desarrollo costoso y muchas veces no es viable para pequeños restaurantes [18].

Kioscos y tablets en mesa

Los kioscos de autoservicio y las tablets instaladas son tecnologías muy usadas. Estos dispositivos permiten al cliente visualizar la carta, personalizar su pedido y enviarlo directamente a cocina sin intermediarios. Mejoran la eficiencia del servicio, pero su implementación supone una inversión inicial elevada en hardware y mantenimiento [19].

Sistemas basados en mensajería instantánea

En algunos establecimientos, entre ellos el bufé libre que me inspiró a mí, han adoptado soluciones más improvisas y económicas, como la gestión de pedidos a través de aplicaciones de mensajería como WhatsApp. Un ejemplo común se observa en ciertos bufés libres, donde los clientes envían por chat el número de los platos deseados y el personal los sirve en mesa [20].

Esta solución no tiene un control del estado de las comandas ni ningún seguimiento del histórico de los pedidos [21].

2.3. Comparativa con soluciones existentes

Al analizar las diferentes soluciones existentes en el mercado de hostelería, he podido identificar ventajas y limitaciones que sirven de referencia para el desarrollo de sistemas alternativos. A continuación, presento una comparativa de los principales enfoques.

TPV tradicionales

- **Ventajas:** Estables, probados en el sector, permiten control de inventario y facturación.
- **Limitaciones:** Suelen ser costosos, poco flexibles y dependen de hardware específico. En muchos casos no integran el contacto con el cliente, solamente gestión interna.

Cartas digitales con QR

- **Ventajas:** Económicas, fáciles de implementar, reducen el contacto físico y el gasto en impresión de cartas.

- **Limitaciones:** Se limitan a la visualización de la carta, sin integración con pedidos ni gestión de cocina.

Aplicaciones móviles propias

- **Ventajas:** Mayor personalización y control de la experiencia del cliente.
- **Limitaciones:** Coste elevado de desarrollo y mantenimiento, barrera para los clientes que no deseen instalar una app (a no ser que sea una webapp).

Kioscos y tablets en mesa

- **Ventajas:** Experiencia interactiva, reducción de tiempos de espera y del contacto con el personal y pedidos enviados directamente a cocina.
- **Limitaciones:** Alta inversión en hardware y mantenimiento, difícil para restaurantes pequeños y medianos.

Pedidos por mensajería instantánea

- **Ventajas:** Solución rápida, sin necesidad de hardware adicional y familiar para los clientes.
- **Limitaciones:** No tenemos control sobre los pedidos, existen errores en la comunicación y no obtenemos ningún beneficio para la gestión del restaurante.

2.4. Justificación de la propuesta

Tras el análisis de los sistemas actuales de gestión de pedidos y las tecnologías utilizadas en restauración, se identifican limitaciones comunes relacionadas con los costes de desarrollo/implementación de soluciones avanzadas [22]. Estas barreras hacen que los pequeños y medianos restaurantes no puedan acceder a soluciones tecnológicas avanzadas que mejoren su eficiencia.

Este proyecto surge como respuesta a estas limitaciones, con el siguiente enfoque:

1. Accesibilidad y simplicidad

El sistema no requiere inversiones elevadas en hardware específico. Aprovecha tecnologías ampliamente conocidas (Spring Boot, MySQL, Firebase, servicios REST) para garantizar una curva de aprendizaje reducida tanto para clientes como para gestores del restaurante.

2. Integración entre aplicaciones

A diferencia de soluciones fragmentadas, la propuesta incluye tres aplicaciones interconectadas:

- **Página web con gestión de carta y servicio REST**
- **Aplicación de pedidos para clientes.**
- **Aplicación de gestión de pedidos para el personal**

Esto asegura coherencia en los datos y asegura que la información que ve el cliente y el hostelero siempre será la misma.

3. Flexibilidad tecnológica

El uso de Firebase como base de datos para los pedidos permite manejar estructuras cambiantes sin comprometer registros anteriores, aportando escalabilidad y posibilidad de adaptarse a futuros cambios en la lógica de los pedidos del negocio.

4. Respuesta a la pandemia y tendencias del sector

La propuesta está inspirada en la necesidad de reducir el contacto físico entre cliente y hostelero, impulsada por la pandemia [23]. Al mismo tiempo, recoge ideas vistas en modelos improvisados como los pedidos por Whatsapp en bufés, pero las transforma y eleva a un sistema más estructurado, ordenado, profesional y sobre todo escalable [24].

5. Reducción de costes para el empresario

La solución evita inversiones elevadas en kioscos o tablets específicas para cada mesa [22], al permitir un despliegue flexible y económico que puede adaptarse tanto a tablets propias como a otros dispositivos disponibles.

CAPÍTULO 3: OBJETIVOS

3.1. Objetivos generales

Estos son los objetivos que busco con la realización de este TFG:

- Diseñar y desarrollar un sistema integral que facilite la gestión de pedidos en restaurantes mediante una aplicación web y dos aplicaciones móviles.
- Optimizar la comunicación entre los distintos roles del restaurante (personal de sala, cocina y administración) en tiempo real.
- Susituir los sistemas tradicionales basados en papel o aplicaciones independientes por un sistema centralizado, moderno y escalable, apoyándonos en tecnologías punteras como Spring Boot, Firebase y Android Studio
- Garantizar la integridad y trazabilidad de los pedidos, desde su creación hasta su pago, reduciendo errores humanos.
- Aplicar conocimientos adquiridos durante el Grado de Ingeniería Informática, integrando desarrollo web, desarrollo móvil, bases de datos y diseño de arquitecturas de software.
- Demostrar la viabilidad del sistema en un entorno de producción real, desplegado en un servidor VPS.
- Diseñar un sistema adaptable, que permita incorporar en el futuro nuevas funcionalidades sin cambios drásticos de código.

3.2. Objetivos didácticos

Estos objetivos se centran en lo que se espera aprender y consolidar durante la realización de este trabajo.

- Aplicar metodologías de software para analizar, diseñar e implementar un sistema completo, siguiendo un proceso estructurado y documentado.
- Diseñar y desarrollar servicios RESTful completos utilizando Spring Boot, estableciendo la comunicación entre la aplicación web y las aplicaciones mediante JSON.
- Dominar el uso de BBDD relacionales y no relacionales, combinando MariaDB para la persistencia de la carta y Firebase Firestore para la gestión de pedidos.
- Aplicar principios de usabilidad y experiencia de usuario en el diseño de las interfaces web y móviles, mejorando la interacción y facilidad de uso.

- Profundizar en el uso de herramientas y frameworks actuales, como Gradle, Docker, Firebase Authentication, etc. Todos ellos fundamentales para el desarrollo de software.
- Fomentar la capacidad de planificación, análisis y resolución de problemas técnicos, enfrentando decisiones de diseño y desarrollo del sistema.
- Consolidar competencias transversales, como la documentación técnica, la gestión del tiempo y el trabajo autónomo en entornos de desarrollo profesional.
- Aprender a realizar pruebas funcionales y de integración, validando el correcto funcionamiento de la aplicación.
- Desarrollar habilidades en el despliegue y mantenimiento de aplicaciones en servidores.

3.3. Objetivos personales

A nivel personal, el desarrollo de este proyecto ha supuesto un reto y una oportunidad de crecimiento a nivel profesional y formativo. Se busca mejorar la capacidad de planificación, seguimiento y evaluación del trabajo siguiendo el enfoque del Personal Software Process (PSP) propuesto por Humphrey [39], que fomenta la mejora continua del desarrollador.

Entre los principales objetivos destacan:

- Desarrollar una aplicación útil y funcional que aporte valor al sector de la restauración, permitiendo optimizar procesos reales dentro de un negocio.
- Ampliar los conocimientos técnicos en el ámbito del desarrollo completo, tanto backend como frontend y tecnologías móviles.
- Fortalecer la capacidad de autoaprendizaje y adaptación tecnológica, adquiriendo dominio en herramientas no vistas en profundidad durante la carrear, como Spring Boot o Firebase.
- Mejorar la organización y gestión del tiempo, planificando las diferentes fases del proyecto y cumpliendo los objetivos de tiempo propuestos.
- Incrementar la confianza personal y la autonomía, enfrentándome al ciclo completo de desarrollo y despliegue de una aplicación real de este calibre.
- Consolidar la capacidad de documentación técnica y redacción, presentando un trabajo académico claro, riguroso y bien estructurado que refleje el documento adquirido a lo largo del grado.

- Desarrollar una mentalidad orientada a la mejora continua, aprendiendo de los errores anteriores y aplicando soluciones más eficientes.
- Fomentar la creatividad e iniciativa personal, buscando nuevas formas de interactuar con el sistema y mejorar la experiencia del usuario final.

CAPÍTULO 4: ANÁLISIS DE REQUISITOS

Los requisitos son la base para definir qué debe hacer un sistema y en qué condiciones debe operar [34][35]. Se dividen en funcionales y no funcionales, cada uno con un papel fundamental en el diseño y desarrollo de software [34][36].

4.1. Requisitos funcionales

4.1.1. Aplicación web

RF-WEB-1	Visualización de carta pública
Descripción	El sistema debe permitir a los clientes anónimos visualizar la página web y visualizar la carta completa del restaurante, incluyendo platos, categorías, precios e imágenes.

Tabla 1. Requisito funcional RF-WEB-1: Acceder a la página principal

RF-WEB-2	Autenticación de usuarios
Descripción	El sistema debe contar con un sistema de autenticación basado en Spring Security, que permita diferenciar entre clientes y administradores.

Tabla 2. Requisito funcional RF-WEB-2: Autenticación de usuarios

RF-WEB -3	Alta de nuevos platos
Descripción	El administrador debe poder añadir nuevos platos a la carta indicando toda su información y asignarlo a una categoría.

Tabla 3. Requisito funcional RF-WEB-3: Alta de nuevos platos

RF-WEB -4	Modificación de platos
Descripción	El administrador debe poder modificar la información de los platos existentes (nombre, precio, categoría, imagen, descripción, etc.).

Tabla 4. Requisito funcional RF-WEB-4: Modificación de platos

RF-WEB -5	Eliminación de platos
Descripción	El administrador puede eliminar platos de la carta, eliminándose de la BBDD.

Tabla 5. Requisito funcional RF-WEB-5: Eliminación de platos

RF-WEB -6	Alta de nuevas categorías
Descripción	El administrador debe poder añadir nuevas categorías a la carta indicando toda su información, para poder asignar platos a esta.

Tabla 6. Requisito funcional RF-WEB-6: Alta de nuevas categorías

RF-WEB -7	Modificación de categorías
Descripción	El administrador debe poder modificar la información de las categorías existentes (nombre, imagen, descripción, etc.).

Tabla 7. Requisito funcional RF-WEB-7: Modificación de categorías

RF-WEB -8	Eliminación de categorías
Descripción	El administrador debe poder eliminar categorías de la carta, eliminándose de la BBDD (si hay platos en esa categoría se borrarán).

Tabla 8. Requisito funcional RF-WEB-8: Eliminación de categorías

RF-WEB -9	Almacenamiento en base de datos
Descripción	Toda la información de la carta debe almacenarse en una BBDD SQL relacional.

Tabla 9. Requisito funcional RF-WEB-9: Almacenamiento en BBDD

RF-WEB-10	Obtención de categorías en formato JSON (RESTful)
Descripción	El sistema debe exponer un servicio RESTful que proporcione todas las categorías en formato JSON.

Tabla 10. Requisito funcional RF-WEB-10: Obtención de categorías en formato JSON

RF-WEB-11	Obtención de platos en formato JSON (RESTful)
Descripción	El sistema debe exponer un servicio RESTful que proporcione todos los platos en formato JSON. Además, se puede filtrar por categoría.

Tabla 11. Requisito funcional RF-WEB-11: Obtención de platos en formato JSON

4.1.2. Aplicación de pedidos (cliente)

RF-CLI-1	Visualización de categorías
Descripción	La aplicación debe mostrar al cliente la lista de categorías disponibles de la carta con su imagen representativa.

Tabla 12. Requisito funcional RF-CLI -1: Visualización de categorías

RF-CLI -2	Visualización de platos por categoría
Descripción	Al seleccionar una categoría, la aplicación debe mostrar los platos asociados a ella, incluyendo la fotografía y su nombre.

Tabla 13. Requisito funcional RF-CLI -2: Visualización de platos por categoría

RF-CLI-3	Visualización del detalle del plato
Descripción	Al seleccionar un plato, la aplicación debe mostrar su información detallada, incluyendo su imagen ampliada, descripción completa y precio junto a una opción para añadir al carrito.

Tabla 14. Requisito funcional RF-CLI-3: Visualización del detalle del plato

RF-CLI-4	Añadir plato al carrito
Descripción	Al seleccionar el botón de añadir al carrito, la aplicación debe abrir un modal que nos permita seleccionar la cantidad del plato (si queremos varios platos iguales) y un comentario (por ejemplo, poco hecho un bistec).

Tabla 15. Requisito funcional RF-CLI-4: Añadir plato al carrito

RF-CLI-5	Visualización del carrito
Descripción	La aplicación debe permitir acceder al carrito en cualquier momento para visualizar los platos añadidos y el subtotal.

Tabla 16. Requisito funcional RF-CLI-5: Visualización del carrito

RF-CLI -6	Modificación del carrito
Descripción	La aplicación debe permitir al cliente modificar el carrito, eliminando productos si es necesario o modificando su comentario.

Tabla 17. Requisito funcional RF-CLI -6: Modificación del carrito

RF-CLI-7	Autenticación administrativa
Descripción	La aplicación debe permitir iniciar sesión únicamente a administradores mediante autenticación con correo autorizado para operaciones especiales como asignar el número de mesa.

Tabla 18. Requisito funcional RF-CLI-7: Autenticación administrativa

RF-CLI-8	Selección de número de mesa
Descripción	La aplicación debe permitir a los administradores asignar el número de mesa, para ello deberá iniciar sesión previamente como administrador y acceder a su opción a través de una opción del menú.

Tabla 19. Requisito funcional RF-CLI -8: Selección de número de mesa

RF-CLI-9	Cierre de sesión administrador
Descripción	El administrador debe poder cerrar la sesión después de asignar el número de mesa, para que el cliente no pueda cambiarlo.

Tabla 20. Requisito funcional RF-CLI-9: Cierre de sesión administrador

RF-CLI-10	Realización del pedido
Descripción	La aplicación debe poder permitir al cliente confirmar el pedido, enviándolo a la BBDD NoSQL almacenada en Firebase.

Tabla 21. Requisito funcional RF-CLI-10: Realización del pedido

RF-CLI-11	Verificación de pedido activo
Descripción	La aplicación debe comprobar en Firebase si existe un pedido abierto para el número de mesa asignado (no pagado).

Tabla 22. Requisito funcional RF-CLI-11: Verificación de pedido activo

RF-CLI-12	Creación de un nuevo pedido
Descripción	La aplicación debe crear un nuevo pedido en Firebase si no existe un pedido abierto o si el último pedido está pagado.

Tabla 23. Requisito funcional RF-CLI-12: Creación de un nuevo pedido

RF-CLI-13	Creación de comanda
Descripción	La aplicación, en caso de que exista un pedido activo sin pagar, añadir una comanda a dicho pedido en vez de crear uno nuevo.

Tabla 24. Requisito funcional RF-CLI-13: Creación de comanda

RF-CLI-14	Actualización de estado del pedido
Descripción	La aplicación debe actualizar en Firebase el estado de un pedido y sus comandas según los cambios realizados.

Tabla 25. Requisito funcional RF-CLI-14: Actualización de estado del pedido

4.1.3. Aplicación de gestión de pedidos (interna)

RF-GES-1	Autenticación de administradores
Descripción	La aplicación debe mostrar una pantalla de inicio de sesión al acceder, permitiendo únicamente el acceso a usuarios administradores identificados.

Tabla 26. Requisito funcional RF-GES-1: Autenticación de administradores

RF-GES-2	Menú principal
Descripción	Una vez iniciada la sesión, la aplicación debe mostrar un menú lateral desplegable de tipo hamburguesa con las opciones “Pedidos Actuales” e “Histórico de pedidos”, junto a demás opciones interesantes sobre la gestión.

Tabla 27. Requisito funcional RF-GES-2: Menú principal

RF-GES-3	Visualización de pedidos actuales
Descripción	La aplicación debe mostrar la lista de pedidos en curso (no pagados), indicando información básica como número de mesa, estado, platos y hora de creación.

Tabla 28. Requisito funcional RF-GES-3: Visualización de pedidos actuales

RF-GES-4	Visualización de histórico de pedidos
Descripción	La aplicación debe permitir acceder al detalle de un pedido, mostrando el listado completo, comentarios, cantidad y demás.

Tabla 29. Requisito funcional RF-GES-4: Visualización de histórico de pedidos

RF-GES-5	Modificación del estado del pedido
Descripción	La aplicación debe permitir cambiar el estado de un pedido entre sus posibles valores, entregado, pagado, etc.).

Tabla 30. Requisito funcional RF-GES-5: Modificación del estado del pedido

RF-GES-6	Cambio de estado mediante interacción visual
Descripción	La aplicación debe permitir modificar ciertos estados del pedido de manera intuitiva, pulsando sobre una fotografía asociada a dicho estado.

Tabla 31. Requisito funcional RF-GES-6: Cambio de estado mediante interacción visual

RF-GES-7	Actualización en tiempo real
Descripción	Los cambios de estado realizados en la aplicación deben reflejarse en base de datos y ser visibles en las demás aplicaciones conectadas.

Tabla 32. Requisito funcional RF-GES-7: Actualización en tiempo real

RF-GES-8	Cerrar sesión
Descripción	Esta aplicación debe permitir el cierre de sesión, para evitar usuarios no identificados que puedan manipular los datos.

Tabla 33. Requisito funcional RF-GES-8: Cerrar sesión

4.2. Requisitos no funcionales

4.2.1. Aplicación web

RNF-WEB-1	Escalabilidad
Descripción	Esta aplicación está diseñada en módulos para poder añadir elementos nuevos a la página sin modificar en exceso el sistema. Por ejemplo, si en un futuro queremos separar completamente las bebidas de los platos y crear una sección solo para ellas, los cambios son mínimos. Además, podemos añadir nuevos tipos de roles con permisos diferentes de forma sencilla.

Tabla 34. Requisito no funcional RNF-WEB-1: Escalabilidad

RNF-WEB-2	Usabilidad
Descripción	Debemos diseñar una interfaz de usuario intuitiva y fácil de usar para que los usuarios naveguen con facilidad y mantenerlos el máximo tiempo posible.

Tabla 35. Requisito no funcional RNF-WEB-2: Usabilidad

RNF-WEB-3	Compatibilidad
Descripción	Debemos asegurarnos de que la aplicación sea compatible con el máximo posible de navegadores web (Safari, Chrome, Firefox, Edge, etc.) y dispositivos, además de adaptarse a varios tamaños de pantalla.

Tabla 36. Requisito no funcional RNF-WEB-3: Compatibilidad

RNF-WEB-4	Seguridad
Descripción	El sistema debe implementar sistemas de autenticación segura para proteger las funciones de administrador. Se deberá añadir un sistema de autenticación para proteger el sistema REST.

Tabla 37. Requisito no funcional RNF-WEB-4: Seguridad

RNF-WEB-5	Rendimiento
Descripción	La aplicación debe ser capaz de mostrar la carta de platos en menos de 2 segundos en condiciones normales de carga (en casos en los que no existan demasiados usuarios concurrentes).

Tabla 38. Requisito no funcional RNF-WEB-5: Rendimiento

RNF-WEB-6	Mantenibilidad
Descripción	El código debe estar organizado y documentado de forma que facilite futuras modificaciones, corrección de errores y futuros CR.

Tabla 39. Requisito no funcional RNF-WEB-6: Mantenibilidad

4.2.2. Aplicación de pedidos (cliente)

RNF-CLI-1	Usabilidad
Descripción	La interfaz debe ser extremadamente sencilla e intuitiva para garantizar que cualquier cliente pueda navegar, seleccionar plato y realizar pedidos sin necesidad de asistencia.

Tabla 40. Requisito no funcional RNF-CLI-1: Usabilidad

RNF-CLI-2	Rendimiento
Descripción	La aplicación debe ser capaz de cargar las categorías y los platos en menos de 2 segundos en condiciones normales de red.

Tabla 41. Requisito no funcional RNF-CLI-2: Rendimiento

RNF-CLI-3	Compatibilidad
Descripción	La aplicación debe ser totalmente funcional en tablets con sistema operativo Android y pantallas de distinto tamaño. Además, para futuros cambios se ha implementado la vista para teléfonos móviles.

Tabla 42. Requisito no funcional RNF-CLI-3: Compatibilidad

RNF-WEB-4	Seguridad
Descripción	La aplicación debe restringir la gestión del número de mesa y la sesión del administrador, de modo que el cliente no pueda modificarlo.

Tabla 43. Requisito no funcional RNF-CLI-4: Seguridad

RNF-CLI-5	Robustez
Descripción	El sistema debe manejar errores de red o de sincronización con Firebase mostrando mensajes claros al usuario y evitando pérdidas de datos.

Tabla 44. Requisito no funcional RNF-CLI-5: Robustez

RNF-CLI-6	Escalabilidad
Descripción	La arquitectura debe permitir añadir nuevas funcionalidades en el futuro, como encuestas de satisfacción sin necesidad de cambios masivos.

Tabla 45. Requisito no funcional RNF-CLI-6: Escalabilidad

4.2.3. Aplicación de gestión de pedidos (interna)

RNF-GES-1	Seguridad
Descripción	El acceso debe estar protegido mediante autenticación, garantizando que solo los administradores registrados puedan gestionar los pedidos.

Tabla 46. Requisito no funcional RNF-GES-1: Seguridad

RNF-GES-2	Usabilidad
Descripción	La interfaz debe estar diseñada para permitir una gestión ágil y sencilla, con una curva de aprendizaje mínima para el personal.

Tabla 47. Requisito no funcional RNF-GES-2: Usabilidad

RNF-GES-3	Rendimiento
Descripción	La aplicación debe mostrar y actualizar los pedidos en menos de 1 segundo tras recibir cambios desde Firebase.

Tabla 48. Requisito no funcional RNF-GES-3: Rendimiento

RNF-GES-4	Robustez
Descripción	El sistema debe ser capaz de manejar errores de conexión con Firebase mostrando mensajes claros, sin pérdida de datos y manteniendo la coherencia entre los pedidos.

Tabla 49. Requisito no funcional RNF-GES-4: Robustez

RNF-GES-5	Escalabilidad
Descripción	La arquitectura debe permitir la inclusión de nuevas funcionalidades como reportes estadísticos, exportación de datos o integración con sistemas de facturación.

Tabla 50. Requisito no funcional RNF-GES-5: Escalabilidad

RNF-GES-6	Compatibilidad
Descripción	La aplicación debe ser funcional en distintos dispositivos Android, tanto tablets como teléfonos usados por los empleados, adaptándose a diferentes tamaños de pantalla.

Tabla 51. Requisito no funcional RNF-GES-6: Compatibilidad

RNF-GES-7	Mantenibilidad
Descripción	El código debe estar modularizado y documentado, facilitando futuras modificaciones o ampliaciones por parte de otros desarrolladores.

Tabla 52. Requisito no funcional RNF-GES-7: Mantenibilidad

RNF-GES-8	Disponibilidad
Descripción	La aplicación debe garantizar el funcionamiento continuo durante la jornada laboral del restaurante, minimizando caídas y recuperándose automáticamente ante fallos temporales de red.

Tabla 53. Requisito no funcional RNF-GES-8: Disponibilidad

4.3. Casos de uso

Los casos de uso se han definido siguiendo la metodología de ingeniería del software orientada a objetos y UML, basada en [35][36][37]. Cada tabla recoge actores, descripción, secuencia normal, frecuencia, importancia, urgencia y comentarios.

4.3.1. Aplicación web

C.U.WEB-1	*Acceder a la página principal*
Actores	Usuario anónimo, usuario común, usuario administrador
Descripción	Al acceder a la página web cargará el Home de la página
Secuencia normal	P1 - Acceder a la página
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Cualquier persona puede acceder a la misma, además, después de iniciar sesión nos redirige a esta pantalla.

Tabla 54. Caso de Uso C.U.WEB-1: Acceder a la página principal

C.U. WEB-2	*Acceder a listado de platos*
Actores	Usuario anónimo, usuario común, usuario administrador
Descripción	A través de la pantalla de inicio podemos acceder al listado de platos
Dependencias	*C.U.1*
Precondición	

Secuencia normal	P1 - Acceder a la página principal P2 - Seleccionar Acceder al Listado de platos o acceder a través de la URL
Postcondición	La aplicación muestra el listado de platos
Excepciones	• Si al acceder no existen platos, mostramos el listado vacío.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 55. Caso de Uso C.U.WEB-2: Acceder al listado de platos

C.U. WEB-3	*Acceder al listado de categorías*
Actores	*Usuario anónimo, usuario común y usuario administrador*
Descripción	A través de la pantalla de inicio podemos acceder al listado de categorías
Dependencias	*C.U.1*
Secuencia normal	P1 - Acceder a la página principal P2 - Seleccionar Acceder al Listado de platos
Postcondición	La aplicación muestra el listado de categorías
Excepciones	• Si no existen categorías mostramos el listado vacío
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta

Tabla 56. Caso de Uso C.U.WEB-3: Acceder al listado de categorías

C.U. WEB-4	*Iniciar sesión*
Actores	Usuario anónimo
Descripción	Opción de la aplicación que permite iniciar sesión, ya sea como usuario administrador o usuario común.
Dependencias	*C.U.1*
Precondición	No tener sesión iniciada
Secuencia normal	P1 - Acceder a cualquier página de la web P2 - Seleccionar la opción del menú “Iniciar sesión” P3 - Rellenar los datos P4 - Seleccionar “iniciar sesión”
Postcondición	La aplicación guardará tu sesión, mostrará los datos en el footer y tendrás más permisos.
Excepciones	• Si los datos son incorrectos se notifica al usuario
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Comentarios	

Tabla 57. Caso de Uso C.U.WEB-4: Iniciar sesión

C.U. WEB-5	*Cerrar sesión*
Actores	*Usuario común, usuario administrador*
Descripción	Opción que nos deja cerrar la sesión actual
Dependencias	*C.U.4*
Precondición	Debemos tener la sesión iniciada previamente
Secuencia normal	P1 - Acceder a cualquier página de la web P2 - Seleccionar la pestaña del perfil P3 - Apretar "Cerrar sesión"
Postcondición	La aplicación te redirige a la página principal y eliminar todos los datos de sesión anteriores
Excepciones	
Frecuencia	Media
Importancia	Media
Urgencia	Baja
Comentarios	Existe un mensaje de éxito cuando hace la redirección a la página principal en la parte superior.

Tabla 58. Caso de Uso C.U.WEB-5: Cerrar sesión

C.U. WEB-6.	*Acceder a detalle de plato*
Actores	Usuario común, usuario administrador
Descripción	Opción que nos deja visualizar toda la información acerca de un plato
Dependencias	*C.U.2*
Precondición	
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de platos P3 - Seleccionar el ID del plato para acceder al detalle
Postcondición	La página web nos mostrará toda la información acerca del plato
Excepciones	• Si intentan acceder sin permisos se lo impediremos • Si el plato no existe, avisamos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 59. Caso de Uso C.U.WEB-6: Acceder al detalle del plato

C.U. WEB-7.	*Acceder a detalle de categoría*
Actores	Usuario común, usuario administrador
Descripción	Opción que nos deja visualizar toda la información acerca de una categoría
Dependencias	*C.U.3*
Precondición	
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de categorías P3 - Seleccionar el ID de la categoría para acceder al detalle
Postcondición	La página web nos mostrará toda la información acerca de la categoría
Excepciones	• Si intentan acceder sin permisos se lo impediremos • Si la categoría no existe, avisamos al usuario
Rendimiento	Alta
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 60. Caso de Uso C.U.WEB-7: Acceder a detalle de categoría

C.U. WEB-8.	*Gestión de platos*
Actores	Usuario administrador
Descripción	Opción del menú principal, la cual nos lleva al listado de platos y nos permite hacer todas las gestiones pertinentes, como crear, modificar o eliminar platos
Dependencias	*C.U.2, C.U.4*
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 - Acceder al listado de platos P2 - Seleccionar la opción que queramos
Postcondición	
Excepciones	• Si no es un usuario administrador, no podrá acceder a estas funcionalidades
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 61. Caso de Uso C.U.WEB-8: Gestión de platos

C.U. WEB-9.	*Creación de platos*
Actores	*Usuario administrador*
Descripción	Opción de la pantalla de listado de platos, la cual nos permite acceder a un formulario para crear un plato nuevo.
Dependencias	*C.U.2, C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de platos P3 - Seleccionar “Crear nuevo plato” P4 - Rellenar datos P5 - Guardar nuevo plato
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá un nuevo plato.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 62. Caso de Uso C.U.WEB-9: Creación de platos

C.U. WEB-10.	*Modificación de platos*
Actores	Usuario administrador
Descripción	Opción de la pantalla de listado de platos, la cual nos permite acceder a un formulario para modificar un plato existente.
Dependencias	*C.U.2, C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de platos P3 - Seleccionar “Modificar plato” en el plato pertinente P4 - Modificar datos P5 - Guardar cambios
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá el plato actualizado.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta

Tabla 63. Caso de Uso C.U.WEB-10: Modificación de platos

C.U. WEB-11.	*Eliminación de platos*
Actores	Usuario administrador
Descripción	Opción de la pantalla de listado de platos, la cual nos permite eliminar el plato que deseemos.
Dependencias	*C.U.2, C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de platos P3 - Seleccionar “Eliminar plato” en el plato pertinente P4 - Aceptar en la alerta de confirmación
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá el plato actualizado.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 64. Caso de Uso C.U.WEB-11: Eliminación de platos

C.U. WEB-12.	*Gestión de categorías*
Actores	Usuario administrador
Descripción	Opción del menú principal, la cual nos lleva al listado de categorías y nos permite hacer todas las gestiones pertinentes, como crear, modificar o eliminar categorías
Dependencias	*C.U.3,C.U.4*
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 - Acceder al listado de categorías P2 - Seleccionar la opción que queramos
Postcondición	
Excepciones	• Si no es un usuario administrador, no podrá acceder a estas funcionalidades
Rendimiento	Alta
Frecuencia	Alta
Importancia	Alta
Comentarios	

Tabla 65. Caso de Uso C.U.WEB-12: Gestión de categorías

C.U. WEB-13.	*Creación de categorías*
Actores	*Usuario administrador*
Descripción	Opción de la pantalla de listado de categorías, la cual nos permite acceder a un formulario para crear una categoría nueva.
Dependencias	*C.U.3,C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de categorías P3 - Seleccionar “Crear nueva categoría” P4 - Rellenar datos P5 - Guardar nueva categoría
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá una nueva categoría.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 66. Caso de Uso C.U.WEB-13: Creación de categorías

C.U. WEB-14.	*Modificación de categorías*
Actores	Usuario administrador
Descripción	Opción de la pantalla de listado de categorías, la cual nos permite acceder a un formulario para modificar una categoría existente.
Dependencias	*C.U.3, C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de categorías P3 - Seleccionar “Modificar categoría” en el plato pertinente P4 - Modificar datos P5 - Guardar cambios
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá la categoría actualizada.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 67. Caso de Uso C.U.WEB-14: Modificación de categorías

C.U. WEB-15.	*Eliminación de categorías*
Actores	Usuario administrador
Descripción	Opción de la pantalla de listado de categorías, la cual nos permite eliminar la categoría que deseemos.
Dependencias	*C.U.3, C.U.4*
Precondición	Tener sesión iniciada como administrador
Secuencia normal	P1 - Acceder a la página principal P2 - Acceder al listado de platos P3 - Seleccionar “Eliminar” en la categoría pertinente P4 - Aceptar en la alerta de confirmación
Postcondición	La aplicación nos redirigirá al listado, el cual tendrá la categoría actualizada.
Excepciones	• Si el usuario no ha iniciado sesión, no lo permitiremos • Si hay algún dato incorrecto avisaremos al usuario
Frecuencia	Alta
Importancia	Alta
Comentarios	

Tabla 68. Caso de Uso C.U.WEB-15: Eliminación de categorías

C.U. WEB-16.	*Acceso a servicios RESTful*
Actores	Usuario administrador
Descripción	Opción de la aplicación web en la que podemos gestionar todo lo relacionado con platos y categorías e incluso obtenerlo sin necesidad de la interfaz gráfica.
Dependencias	*C.U.4*
Precondición	Tener un token de administrador (Implementación a futuro)
Secuencia normal	P1 - Enviar petición P2 - Recibir respuesta
Postcondición	Si hacemos una petición de consulta, recibimos un JSON
Excepciones	• Si no tiene permiso se niega la petición
Rendimiento	
Frecuencia	Media
Importancia	Media
Urgencia	Baja
Comentarios	

Tabla 69. Caso de Uso C.U.WEB-16: Acceso a servicios RESTful

4.3.2. Aplicación de pedidos (cliente)

C.U.CLI-1	Visualizar listado de categorías (Pantalla principal)
Actores	Cliente
Descripción	El cliente abre la aplicación y accede directamente a la interfaz de uso en modo cliente.
Dependencias	
Precondición	Tener la aplicación instalada y conexión a internet.
Secuencia normal	P1 – Abrir la aplicación
Postcondición	La aplicación carga la página principal con el listado de categorías.
Excepciones	Error de conexión con servicio Rest – Mostrar mensaje de error Error de conexión con Firebase – Mostrar mensaje de error
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Existe un diseño distinto para Tablet o teléfono móvil.

Tabla 70. Casos de uso C.U.CLI-1: Visualizar listado de categorías (Pantalla principal)

C.U.CLI-2	Visualizar platos de una categoría
Actores	Cliente
Descripción	El cliente accede a una categoría y se le muestran los platos que contiene.
Dependencias	C.U.1
Precondición	Seleccionar una categoría existente
Secuencia normal	P1 – Seleccionar una categoría P2 – La aplicación muestra todos los platos asociados a esa categoría.
Postcondición	Visualización correcta de los platos
Excepciones	Error de conexión con servicio Rest – Mostrar mensaje de error Error de conexión con Firebase – Mostrar mensaje de error
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Existe un diseño distinto para Tablet o teléfono móvil, además si está vacía lo indicamos.

Tabla 71. Casos de uso C.U.CLI-2: Visualizar platos de una categoría

C.U.CLI-3	Visualizar detalle de un plato
Actores	Cliente
Descripción	Permite consultar información completa de un plato (foto, descripción, precio, etc.).
Dependencias	C.U.2
Precondición	Seleccionar un plato existente
Secuencia normal	P1 – Seleccionar un plato P2 – La app muestra toda la información asociada
Postcondición	El cliente puede añadir el plato al carrito.
Excepciones	Error de conexión con servicio Rest – Mostrar mensaje de error Si el plato no existe – Mostrar mensaje de error
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Existe un diseño distinto para Tablet o teléfono móvil, en el que se muestra más o menos información dependiendo.

Tabla 72. Casos de uso C.U.CLI-3: Visualizar detalle de un plato

C.U.CLI-4	Añadir plato al carrito
Actores	Cliente
Descripción	Permite añadir un plato seleccionado al carrito de compra, con la cantidad a elegir y un comentario acerca del plato (sin salsa, sin tomate, etc.).
Dependencias	C.U.3
Precondición	Haber visualizado un plato
Secuencia normal	P1 – Seleccionar un plato P2 – Pulsar el botón "Añadir al carrito" P3 – Selecciona la cantidad y añadir el comentario P4 – Seleccionar "Aceptar"
Poscondición	El plato aparece en el carrito con toda su información relacionada.
Excepciones	Error de sincronización – Mostrar aviso
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 73. Casos de uso C.U.CLI-4: Añadir plato al carrito

C.U.CLI-5	Consultar carrito
Actores	Cliente
Descripción	Permite al cliente visualizar el carrito actual con los platos añadidos.
Dependencias	C.U.4
Precondición	
Secuencia normal	P1 – Pulsar el icono del carrito P2 – La aplicación muestra los platos añadidos o una pantalla informativa que avisa en caso de estar vacío.
Postcondición	Carrito visualizado correctamente.
Excepciones	Si está vacío mostramos una pantalla específica para ello, diciéndole al cliente que el carrito está vacío con un botón con una posible redirección a la pantalla principal.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 74. Casos de uso C.U.CLI-5: Consultar carrito

C.U.CLI-6	Confirmar pedido
Actores	Cliente
Descripción	Permite confirmar el pedido actual. El sistema revisa si existe un pedido abierto asociado a la mesa.
Dependencias	C.U.5
Precondición	Tener platos en el carrito
Secuencia normal	P1 – Acceder al carrito P2 – Pulsar "Confrimar pedido" P3 – El sistema comprueba en Firebase si la mesa tiene un pedido abierto en ese momento • Si el pedido ya está pagado o no tiene pedido asociado – Crea un nuevo pedido • Si no está pagado – Añade una comanda al pedido existente.
Poscondición	Pedido registrado en Firebase y carrito vacío.
Excepciones	Error de conexión con Firebase – Mostrar mensaje de error
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	

Tabla 75. Casos de uso C.U.CLI-6: Confirmar pedido

C.U.CLI-7	Iniciar sesión como administrador
Actores	Administrador
Descripción	Permite al personal iniciar sesión como administrador para habilitar funciones adicionales en el menú.
Dependencias	C.U.1
Precondición	No tener la sesión iniciada
Secuencia normal	P1 – Abrir el menú de hamburguesa P2 – Seleccionar “Iniciar sesión” P3 – Introducir credenciales de administrador
Postcondición	Se desbloquean las opciones de cambio de mesa y cerrar sesión en el menú de hamburguesa.
Excepciones	Credenciales incorrectas – Mostrar aviso Error con el sistema de autenticación a través de Firebase – Mostrar aviso
Frecuencia	Baja
Importancia	Alta
Urgencia	Media
Comentarios	

Tabla 76. Casos de uso C.U.CLI-7: Iniciar sesión como administrador

C.U.CLI-8	Cambiar número de mesa

Actores	Administrador
Descripción	Permite seleccionar el número de mesa que se guarda en el dispositivo.
Dependencias	C.U.7
Precondición	Tener sesión de administrador iniciada
Secuencia normal	P1 – Abrir el menú de hamburguesa P2 – Seleccionar opción “Cambiar número de mesa” P3 – Se abre un modal P4 – Introducir nuevo número de mesa P5 – Guardar configuración
Postcondición	El nuevo número de mesa queda registrado en el dispositivo.
Excepciones	Número inválido – Mostrar aviso (se queda el número anterior)
Frecuencia	Media
Importancia	Alta
Urgencia	Media
Comentarios	

Tabla 77. Casos de uso C.U.CLI-8: Cambiar número de mesa

C.U.CLI-9	Cerrar sesión como administrador
Actores	Administrador
Descripción	Permite cerra la sesión de administrador y devolver la aplicación al modo cliente.
Dependencias	C.U.7
Precondición	Tener sesión de administrador iniciada.
Secuencia normal	P1 – Abrir el menú de hamburguesa P2 – Seleccionar “Cerrar sesión” P3 – Seleccionar “Aceptar” en el modal de confirmación
Postcondición	La aplicación vuelve al modo cliente
Excepciones	Fallo en autenticación Firebase – Mostramos aviso
Frecuencia	Media
Importancia	Media
Urgencia	Media
Comentarios	

Tabla 78. Casos de uso C.U.CLI-9: Cerrar sesión como administrador

4.3.3. Aplicación de gestión de pedidos (interna)

C.U.GES-1	Iniciar sesión
Actores	Usuario administrador
Descripción	Permite a los administradores acceder a la aplicación mediante un sistema de autenticación seguro. Solo los usuarios autorizados pueden iniciar sesión.
Precondición	No tener sesión iniciada
Secuencia normal	P1 – Abrir la aplicación P2 – Introducir usuario y contraseña P3 – Pulsar sesión
Postcondición	El usuario accede a la aplicación y se muestra la interfaz con los pedidos actuales.
Excepciones	Datos incorrectos – Notificar al usuario y denegar acceso Error con conexión a sistema de autenticación de Firebase – Avisar al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	La sesión activa permite al usuario interactuar con todos los pedidos

Tabla 79. Caso de uso C.U.GES-1: Iniciar sesión

C.U.GES-2	Visualizar pedidos actuales
Actores	Usuario administrador
Descripción	Permite consultar los pedidos que aún no han sido pagados y se encuentran activos.
Dependencias	C.U.1
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 – Iniciar sesión P2 – Selecciona “Pedidos actuales” en el menú de hamburguesa P3 – Visualizar el listado de pedidos ordenado por fecha de última comanda
Postcondición	Se muestra el listado completo de pedidos actuales
Excepciones	Si no hay pedidos, muestra un listado vacío
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	La información incluye número de mesa, hora de la última comanda, el total de todo el pedido, su estado de entregado y pagado, además de un botón para marcarlo como pagado.

Tabla 80. Caso de uso C.U.GES-2: Visualizar pedidos actuales

C.U.GES-3	Visualizar pedidos históricos
Actores	Usuario administrador
Descripción	Permite consultar los pedidos que ya han sido pagados y forman parte del histórico, también permitimos su modificación por posible equivocación.
Dependencias	C.U.1
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 – Iniciar sesión P2 – Selecciona “Pedidos actuales” en el menú de hamburguesa P3 – Visualizar el listado de pedidos ordenado por fecha de última comanda
Postcondición	Se muestra el listado completo de pedidos históricos ordenador por fecha de la última comanda
Excepciones	Si no hay pedidos históricos, se muestra una pantalla avisando de que no existen pedidos en ese momento.
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Los pedidos pueden ser revisados y corregido en caso de error.

Tabla 81. Caso de uso C.U.GES-3: Visualizar pedidos históricos

C.U.GES-4	Marcar pedido como pagado / no pagado
Actores	Usuario administrador
Descripción	Permite marcar un pedido completo como pagado o, en su caso, devolverlo a los pedidos actuales desde el listado del histórico, trasladando el pedido entre histórico a actuales o viceversa.
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 – Acceder al historial de histórico o actuales P2 – Pulsar el botón "Marcar como pagado" o "Marcar como no pagado" respectivamente P3 – Confirmar la acción en el modal de confirmación
Postcondición	El pedido queda registrado como pagado o no pagado y se mueve al listado correspondiente (actual o histórico)
Excepciones	Usuario no autorizado – Acceso denegado
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	El modal de confirmación sirve para evitar cambios accidentales, proporcionando una seguridad mayor a un cambio tan drástico, evitando perder el pedido en caso de no quererlo.

Tabla 82. Caso de uso C.U.GES-4: Marcar pedido como pagado / no pagado

C.U.GES-5	Visualizar detalle del pedido
Actores	Usuario administrador
Descripción	Permite acceder al detalle de un pedido para ver el listado de comandas, que a su vez contienen platos con su cantidad y comentarios.
Dependencias	C.U.2, C.U.3
Precondición	Sesión iniciada como administrador
Secuencia normal	P1 – Seleccionar un pedido del listado P2 – Acceder al detalle
Postcondición	Se muestra el detalle del pedido con todas las comandas, comentarios, cantidad de cada plato, subtotal de cada comanda y total del pedido.
Excepciones	Pedido no encontrado – Se notifica al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Paso necesario para poder marcar las comandas como entregadas.

Tabla 83. Caso de uso C.U.GES-5: Visualizar detalle del pedido

C.U.GES-6	Marcar comanda como entregada
Actores	Usuario administrador
Descripción	Permite actualizar el estado de entrega de cada comanda dentro de un pedido.
Precondición	Sesión iniciada como administrador y acceso al detalle del pedido.
Secuencia normal	P1 – Acceder al detalle del pedido P2 – Seleccionar la comanda que se ha entregado P3 – Pulsar "Marcar como entregada"
Postcondición	Comanda marcada como entregada y actualización reflejada en el listado.
Excepciones	Usuario no autorizado – Acceso denegado Error con conexión a Firebase – Notificar al usuario
Frecuencia	Alta
Importancia	Alta
Urgencia	Alta
Comentarios	Si todas las comandas están entregadas aparece como entregado el estado del pedido completo.

Tabla 84. Caso de uso C.U.GES-6: Marcar comanda como entregada

4.4. Diagramas UML

Aunque el desarrollo completo de este proyecto se ha prolongado a lo largo de aproximadamente un año, debido a la compatibilidad con el trabajo y los estudios, el tiempo efectivo del trabajo se estima en torno a unos siete meses.

En el siguiente diagrama de Gantt se representa la planificación temporal aproximada correspondiente a dicho periodo efectivo del desarrollo y documentación.

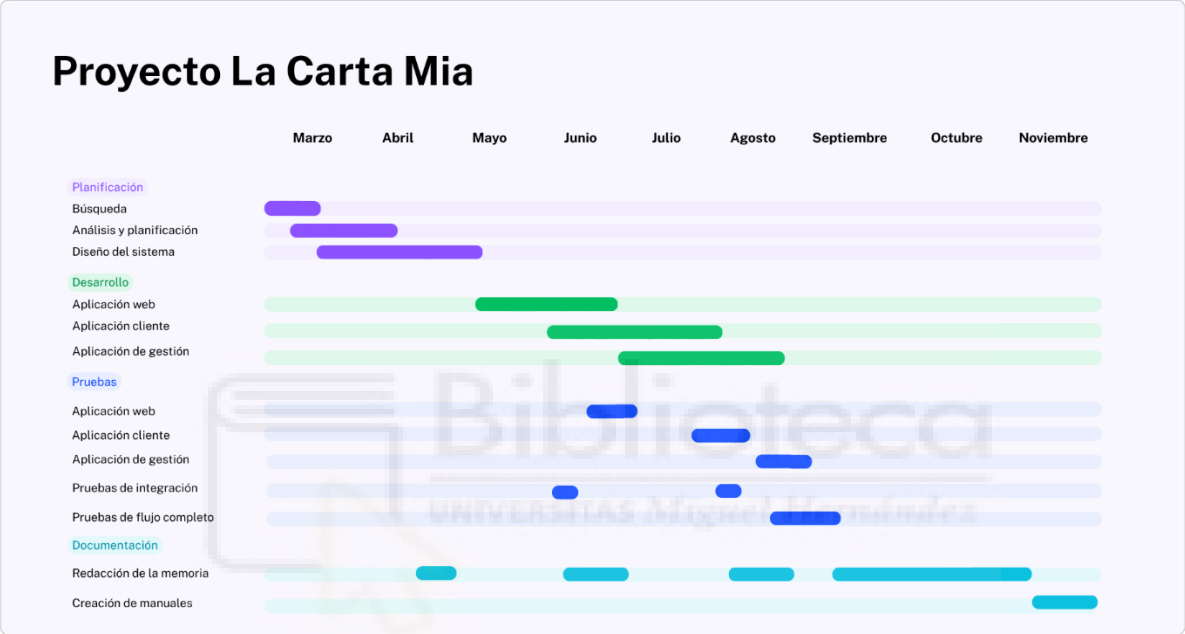

Ilustración 1. Diagrama de Gantt del proyecto

4.4.1. Aplicación web

Diagramas de clases

Para representar gráficamente la estructura y el comportamiento del sistema se han empleado diagramas UML, siguiendo las convenciones y buenas prácticas descritas por Fowler y Scoot [40] y Piattini [41], quienes destacan la utilidad del modelado orientado a objetos para el análisis y diseño de aplicaciones de gestión.

Para la generación de diagramas de clase, he aprovechado la herramienta “Sketch It!”, la cual genera un diagrama por cada paquete y sus dependencias.

Para empezar, el paquete principal de la aplicación, el cual contiene la clase de configuración, la de seguridad y la principal de arranque.

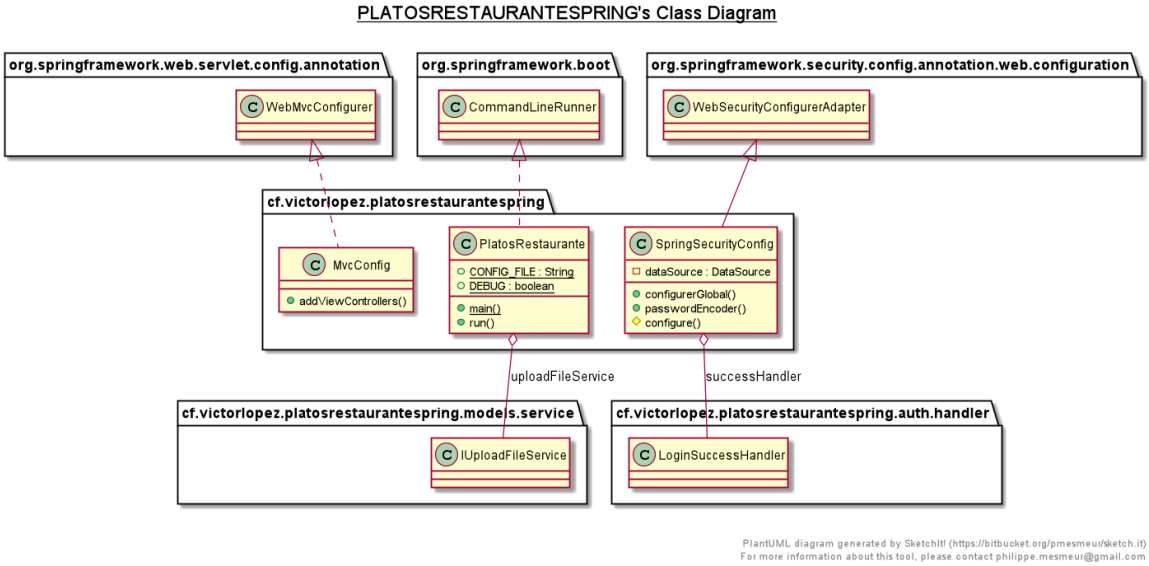

Ilustración 2. Diagrama de clases - Aplicación web - Paquete principal

Para continuar, veremos el paquete de Rest Controllers, el cual tiene las clases que nos dejan comunicarnos con el sistema a través de servicios RESTful. Este diagrama muestra tanto las clases como los métodos públicos que podemos llamar.

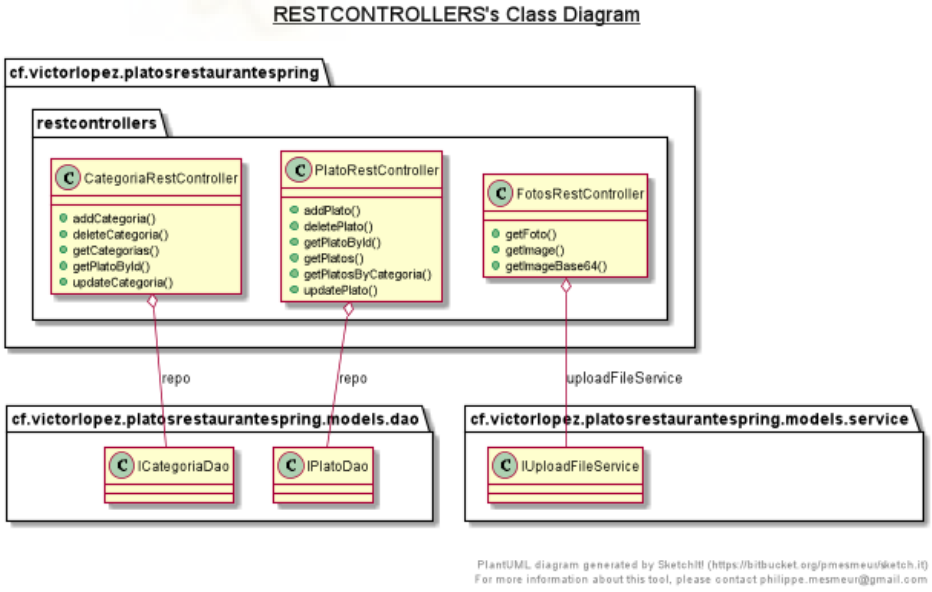

Ilustración 3. Diagrama de clases - Aplicación web - Paquete controladores rest

El siguiente diagrama son los servicios, estos actúan como intermediario entre la lógica de negocio y la base de datos, proporcionando los métodos necesarios para interactuar a través de los DAO (Data Access Object).

Los DAO son los únicos que tienen permiso para comunicarse directamente con la BBDD. Su responsabilidad es traducir las peticiones del servicio en operaciones concretas de la base (crear, leer, actualizar, borrar).

De esta manera, los servicios pueden concentrarse en una lógica de negocio y reglas de la aplicación, sin preocuparse por los detalles de conexión o manipulación de datos a bajo nivel.

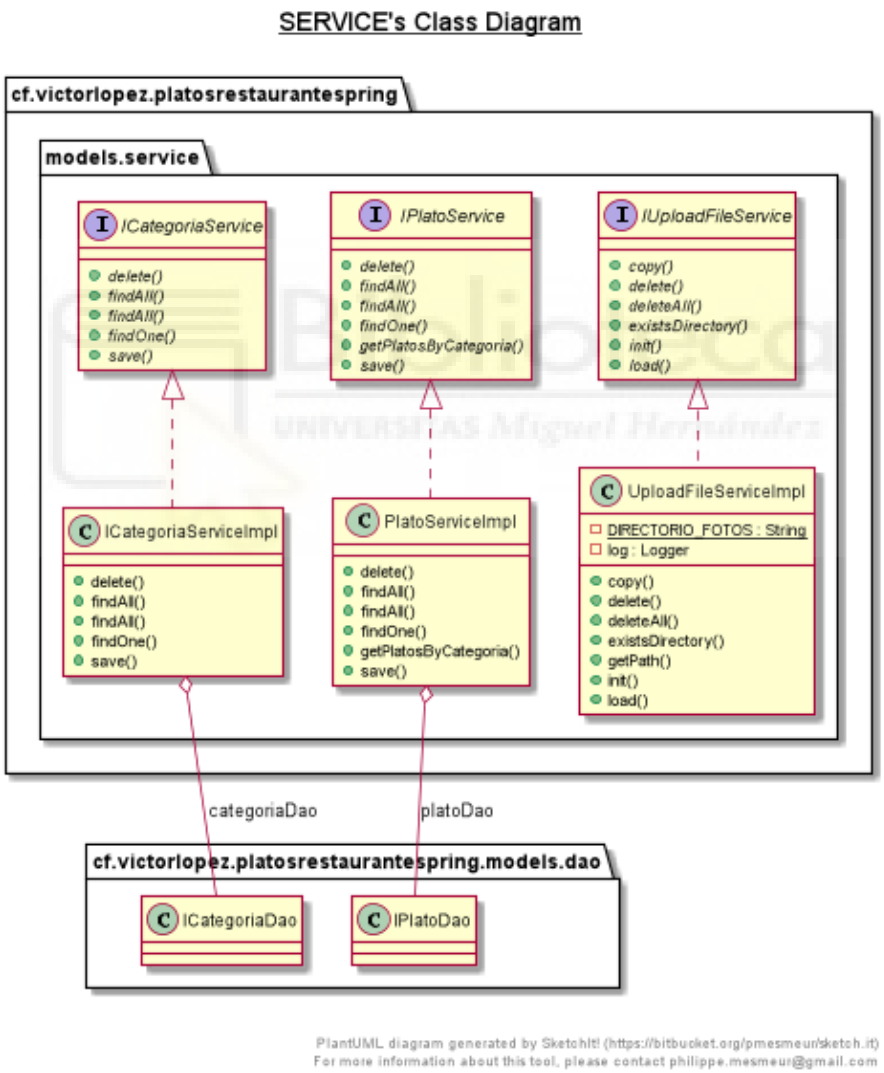

Ilustración 4. Diagrama de clases - Aplicación web - Paquete de servicios y sus implementaciones

Además, los objetos que se manejan en la aplicación son las entidades, las cuales tienen este formato:

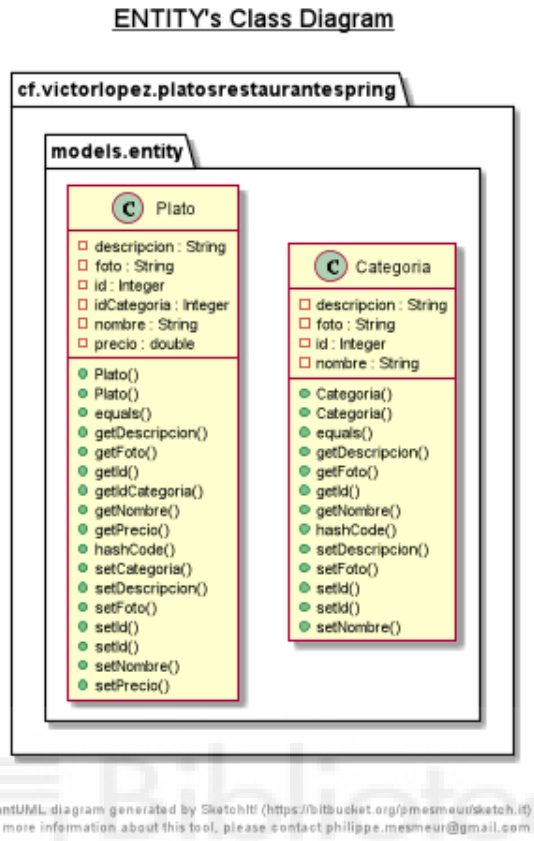

Ilustración 5. Diagrama de clases - Aplicación web - Paquete de entidades

Además de los controladores para el servicio REST, tenemos controladores diferentes para la propia página, para las acciones dentro de ella.

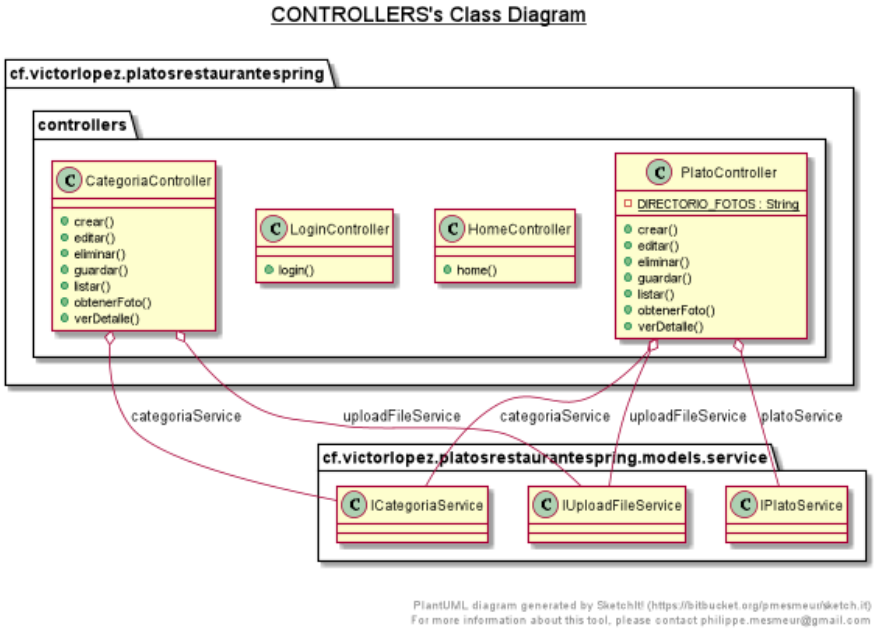

Ilustración 6. Diagrama de clases - Aplicación web - Paquete de controladores

Diagramas de secuencia

Empezamos con el diagrama de secuencia para crear un plato, el procedimiento para crear una categoría es el mismo, pero usando las clases pertinentes para ello.

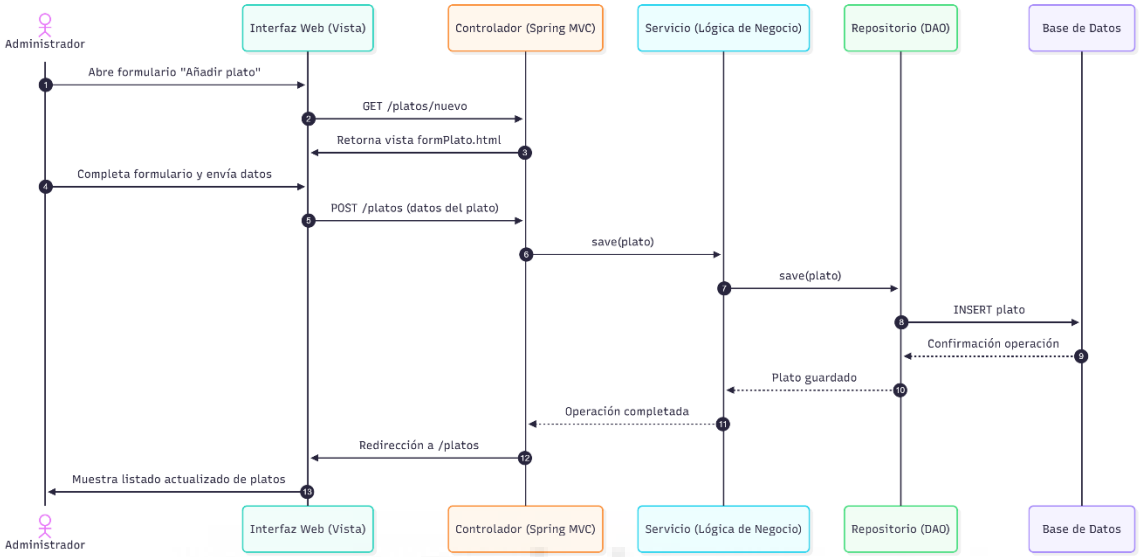

Ilustración 7. Diagrama de secuencia - Aplicación web - Creación de plato

Seguimos con el proceso a seguir para editar un plato, el cual es muy parecido pero llamando a los métodos de editar.

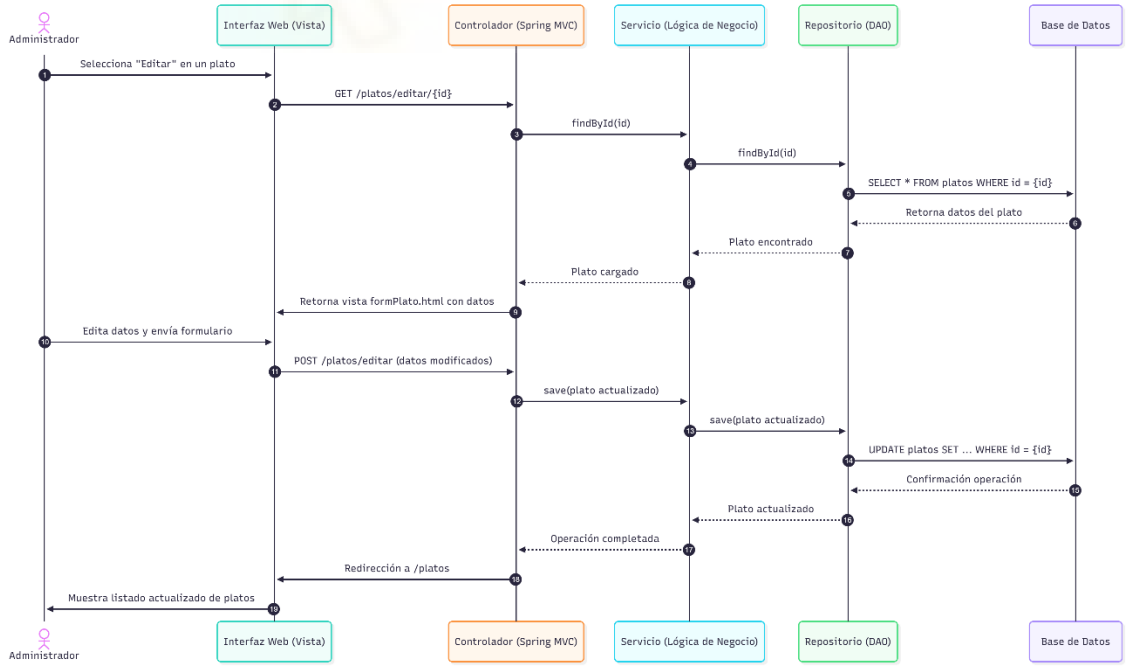

Ilustración 8. Diagrama de secuencia - Aplicación web - Modificación de plato

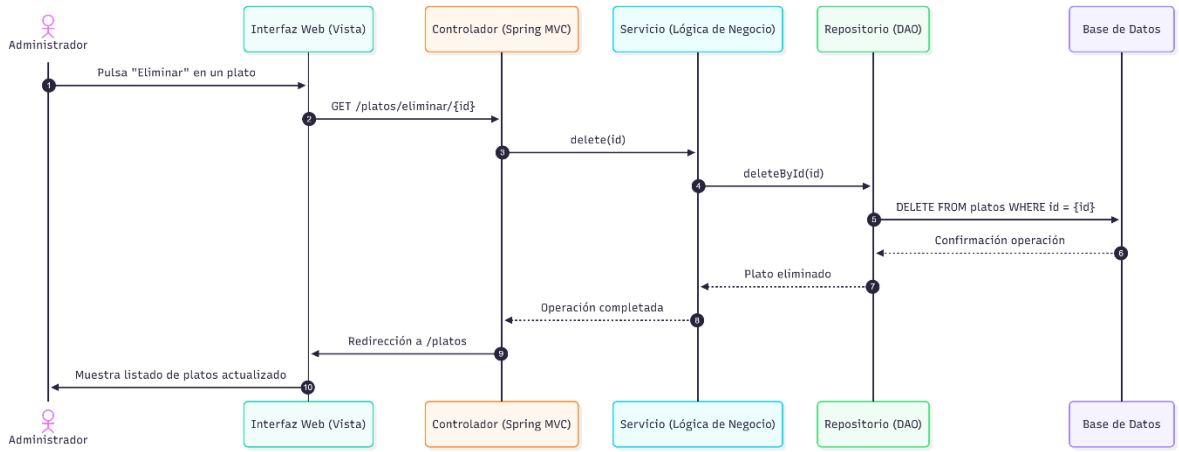

Ilustración 9. Diagrama de secuencia - Aplicación web - Eliminación de plato

4.4.2. Aplicación de pedidos (cliente)

Diagramas de clases

Para empezar, mostraremos los diagramas de clases del paquete principal, el cual contiene la actividad principal y sus dependencias.

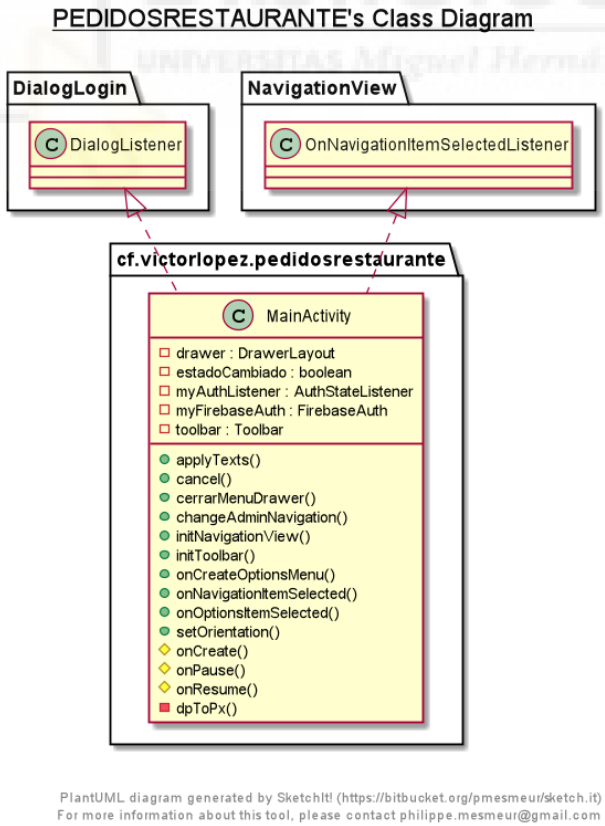

Ilustración 10. Diagrama de clases - Aplicación cliente - Paquete principal

A continuación, veremos los diagramas de la carpeta de modelos, los cuales son las clases Java de nuestro modelado de datos, tanto platos y categorías como pedidos.

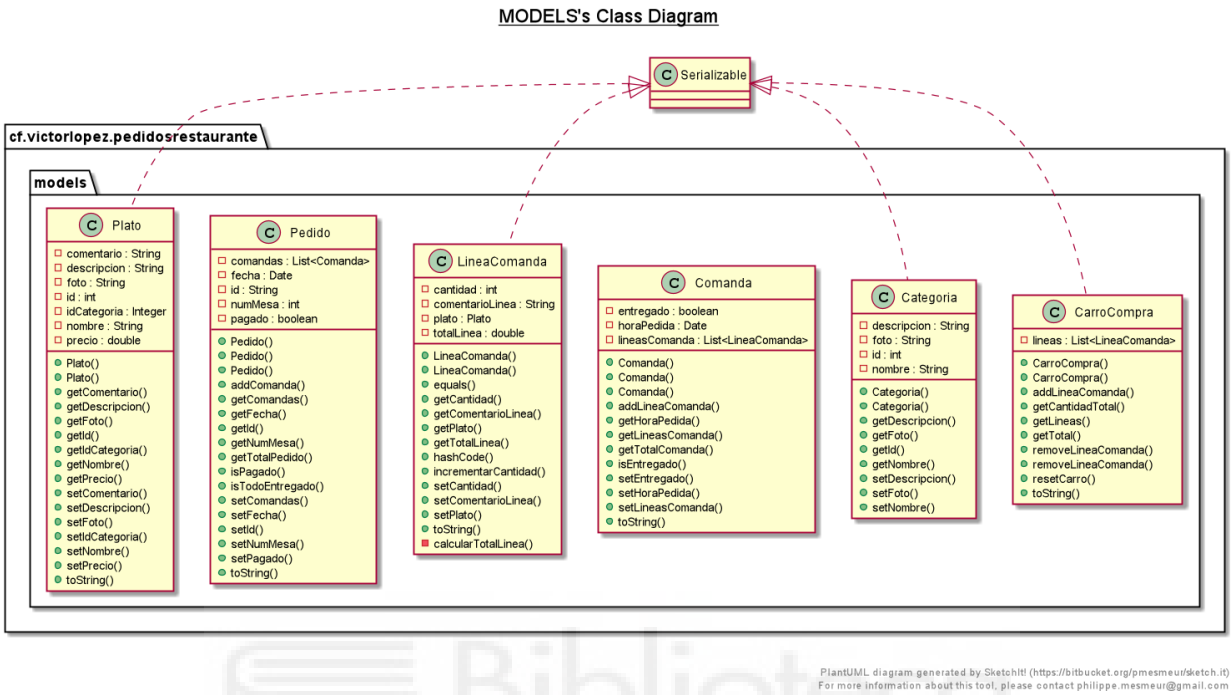

Ilustración 11. Diagrama de clases - Aplicación cliente - Paquete de modelos

Me parece interesante señalar el paquete de los managers, el cual actualmente contiene la clase que implementa el patrón Singleton para la gestión del carrito de la aplicación. Este patrón también se utiliza en el paquete rest, el servicio que nos permite hacer peticiones a la primera aplicación.

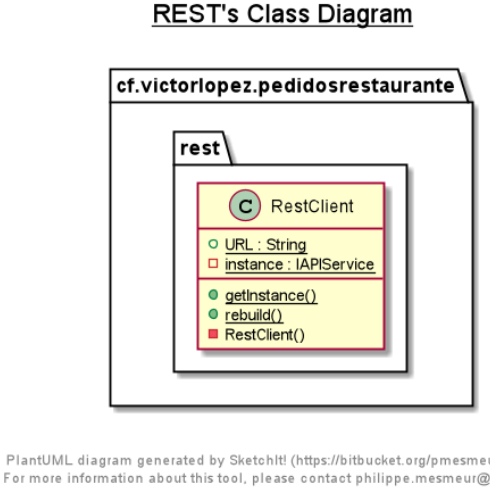

Ilustración 13. Diagrama de clases - Aplicación cliente - Paquete REST

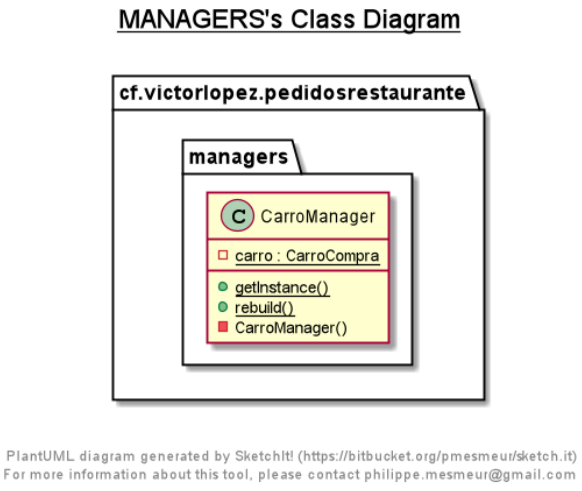

Ilustración 12. Diagrama de clases - Aplicación cliente - Paquete managers

Seguimos con los diagramas del paquete de fragments, los cuales son los que interactúan con los listados y con los servicios, contienen el grueso de la lógica de la aplicación.

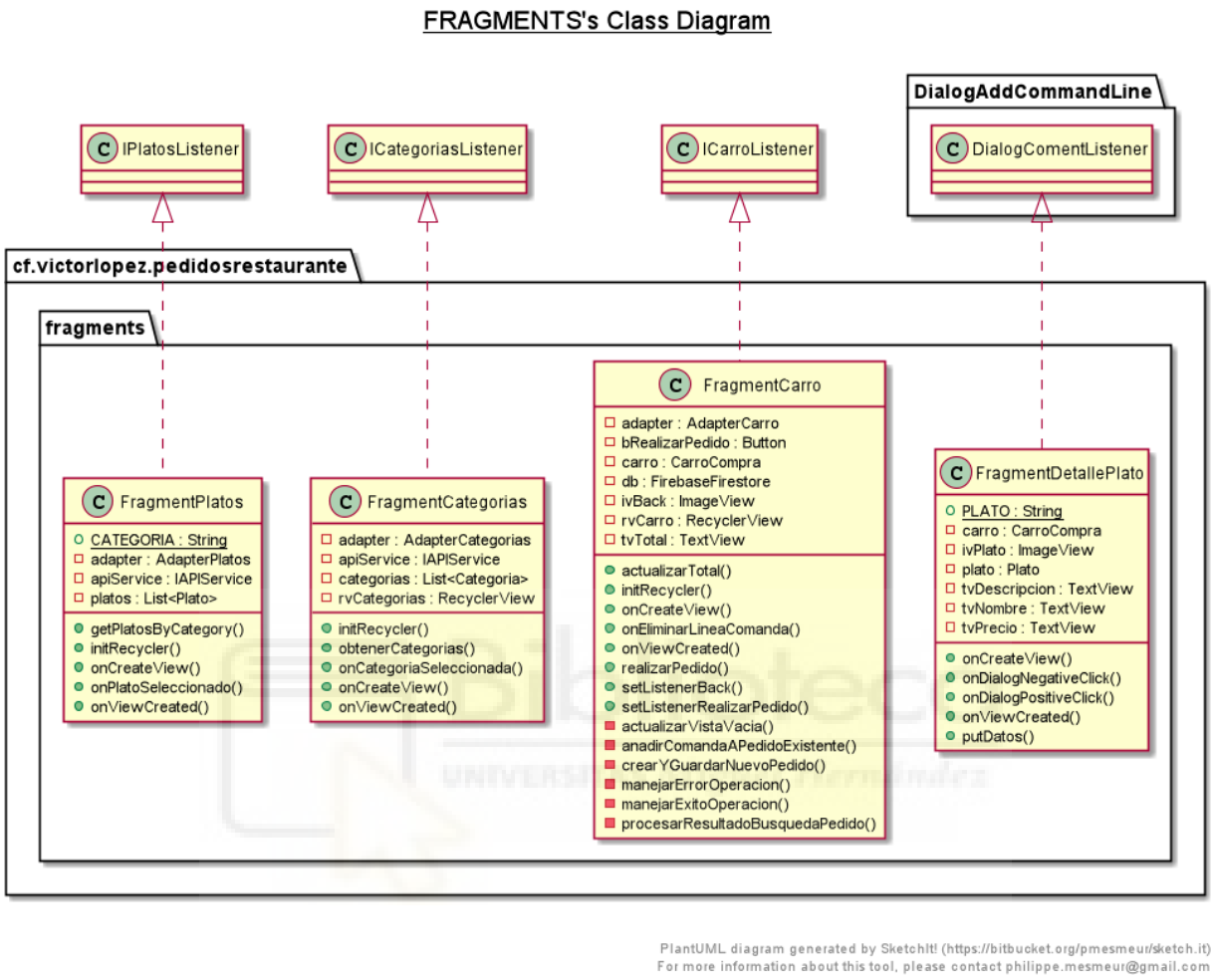

Ilustración 14. Diagrama de clases - Aplicación cliente - Paquete fragments

Los listados son manejados por los adaptadores, en este caso RecyclerViews, para ir pintando los elementos sin necesidad del borrado del elemento.

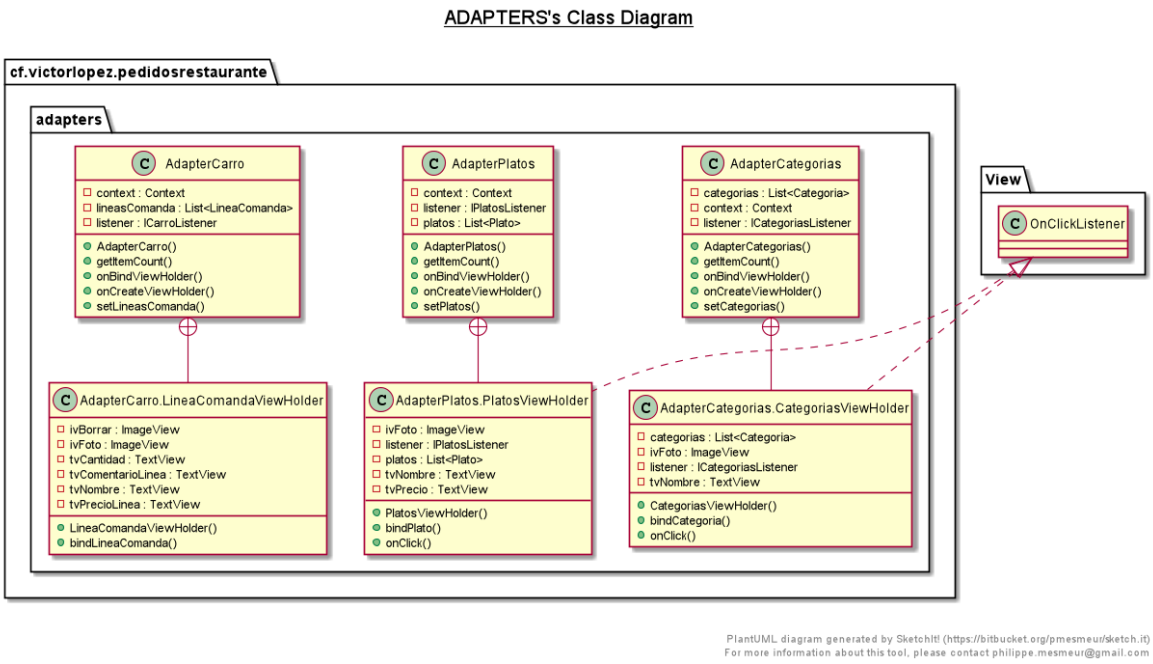

Ilustración 15. Diagrama de clases - Aplicación cliente - Paquete adapters

Como parte importante de esta aplicación, tenemos el paquete de los diálogos, son pestañas para interactuar con el usuario y pedir confirmación o información de cualquier tipo al usuario.

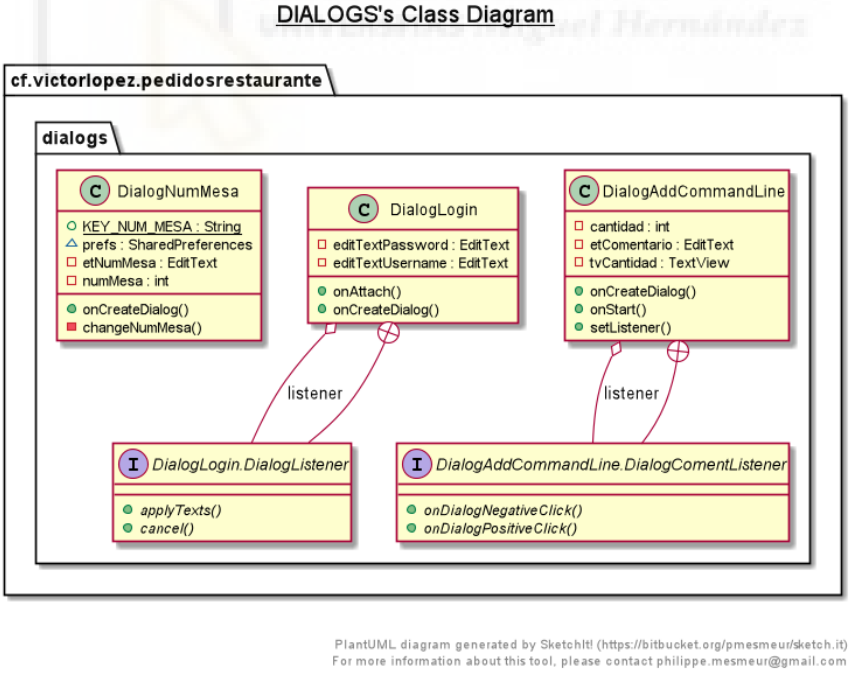

Ilustración 16. Diagrama de clases - Aplicación cliente - Paquete dialogs

Por último, me parece importante recalcar el uso de interfaces para definir un contrato común entre varias clases y usarlos para realizar callbacks cuando ocurren ciertos eventos.

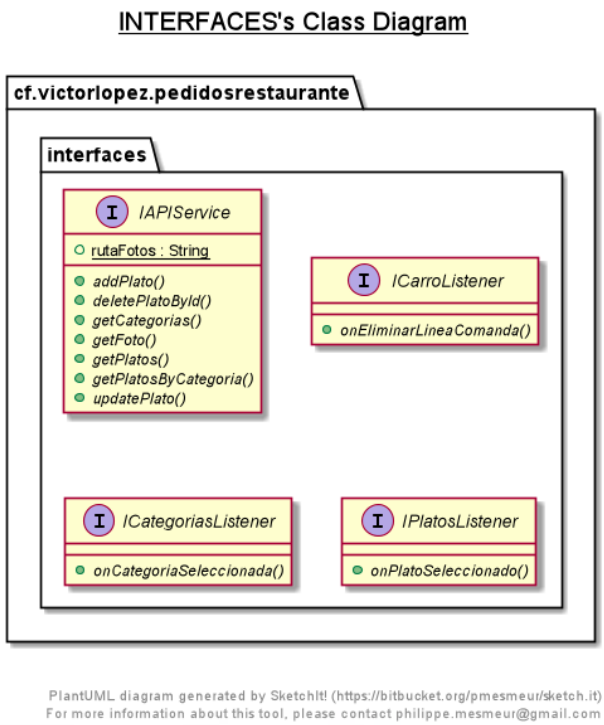

Ilustración 17. Diagrama de clases - Aplicación cliente - Paquete interfaces

Diagrama de secuencia

Para empezar con los diagramas de secuencia, vamos a ver el flujo de consulta de la aplicación cliente a la aplicación web.

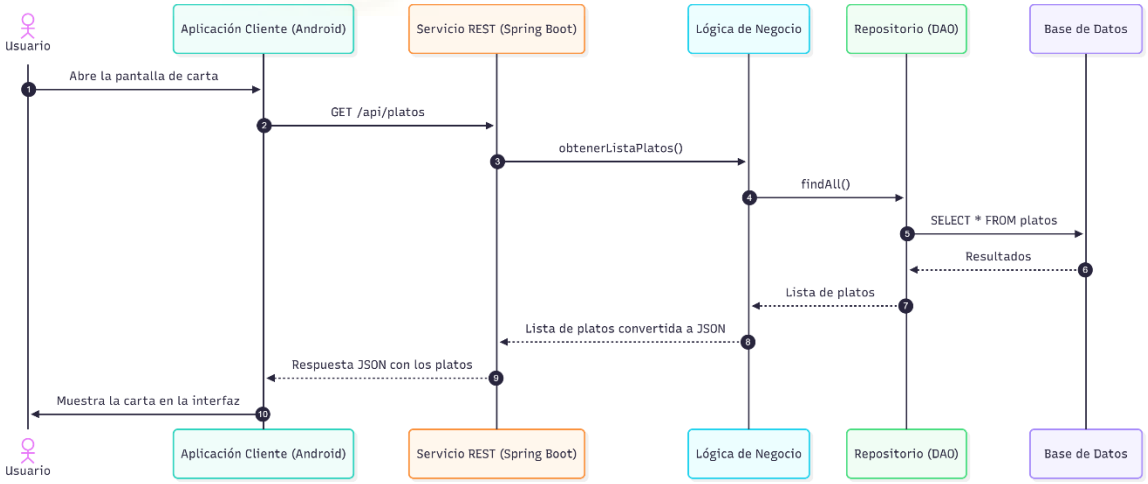

Ilustración 18. Diagrama de secuencia - Aplicación cliente - Conexión con servicio REST

Continuamos con un diagrama de secuencia que contempla el inicio de sesión de un administrador para seleccionar el número de mesa.

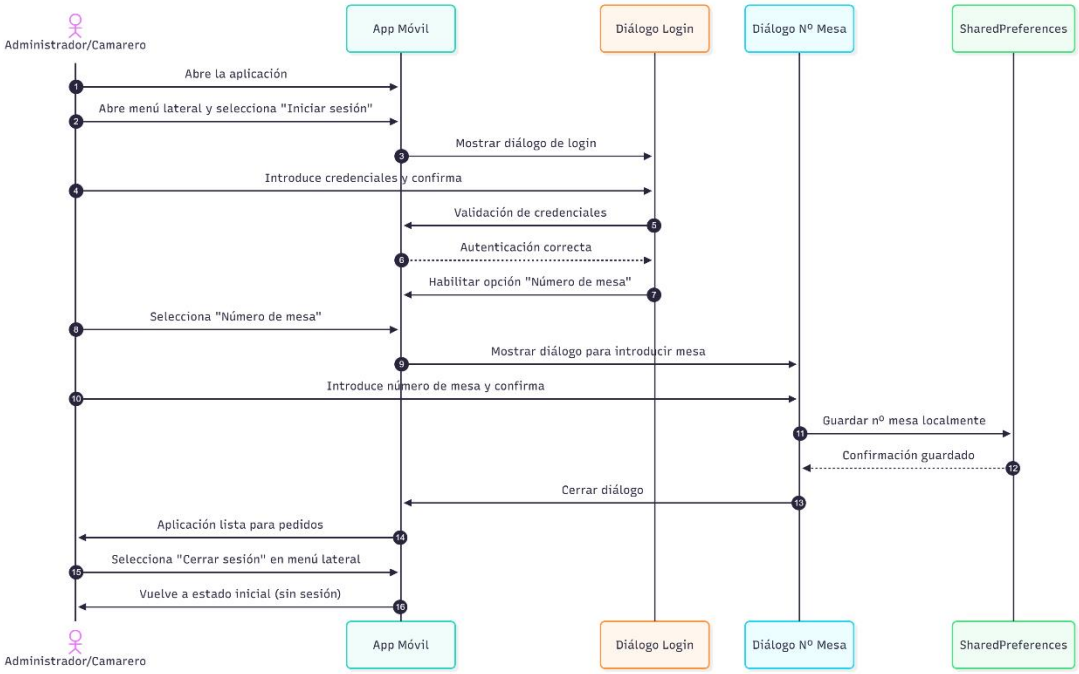

Ilustración 19. Diagrama de secuencia - Aplicación cliente - Inicio de sesión como administrador

Por finalizar, vamos a ver un diagrama de secuencia del uso principal de la aplicación, añadiendo platos al carrito y realizando el pedido.

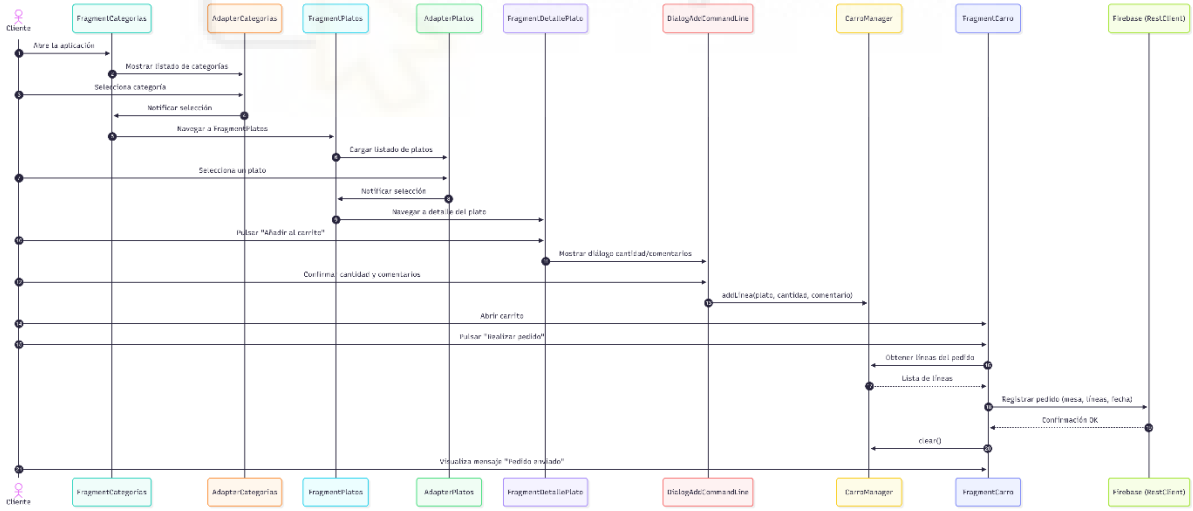

Ilustración 20. Diagrama de secuencia - Aplicación cliente - Flujo normal de la aplicación

4.4.3. Aplicación de gestión de pedidos (interna)

Diagramas de clases

Este proyecto sigue una estructura similar, pero en el paquete principal tenemos una activity más, ya que si no iniciamos sesión, no podemos usar la aplicación, por lo que prefiero separar responsabilidades y cambiar de activity completamente.

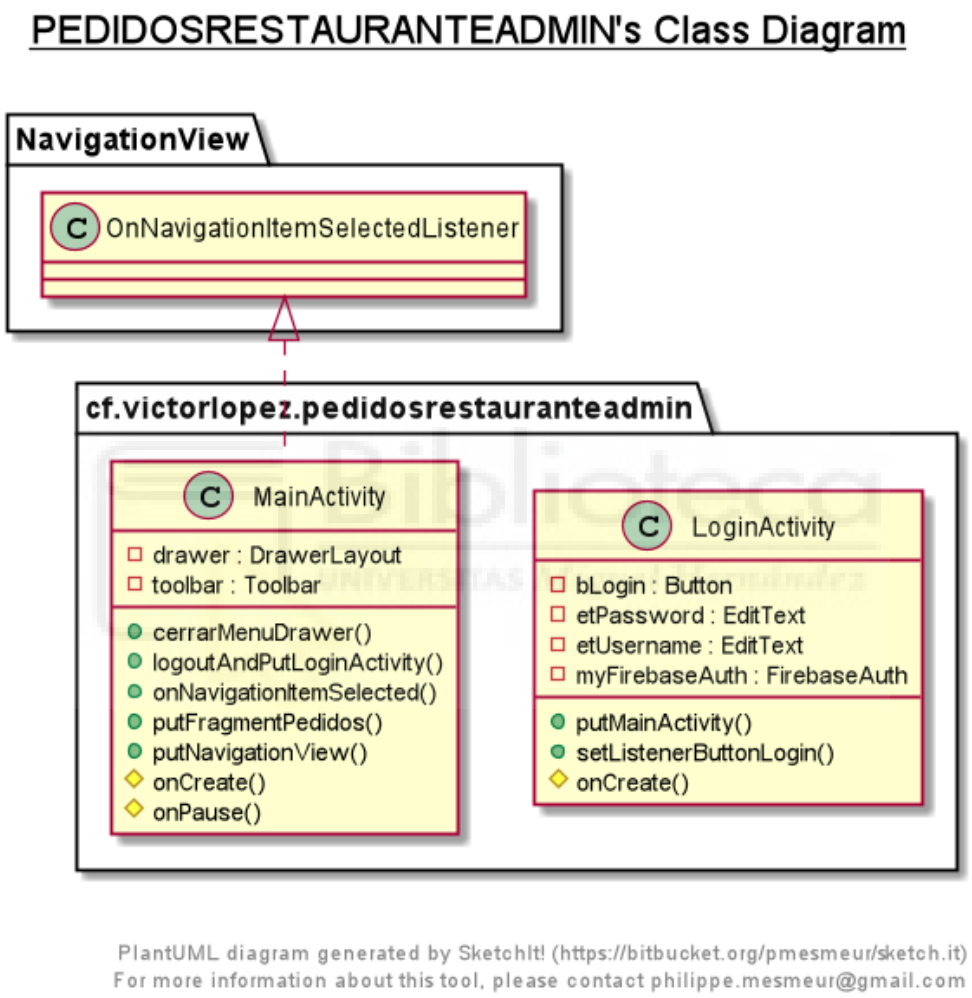

Ilustración 21. Diagrama de clase - Aplicación gestión de pedidos - Paquete principal

Este proyecto comparte las clases de los paquetes modelo y rest de la aplicación anterior, ya que trabajan con los mismos tipos de datos.

El próximo paquete que veremos es el de las interfaces, encargadas de notificar modificaciones en los pedidos y del servicio RESTful.

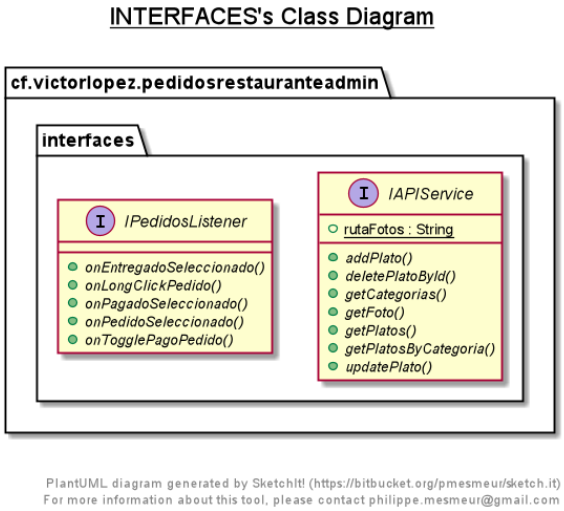

Ilustración 22. Diagrama de clase - Aplicación gestión de pedidos - Paquete de interfaces

Continuamos con los fragments de esta aplicación, como podemos observar se comparte el fragment para los del histórico y para los actuales, cambiando la consulta.

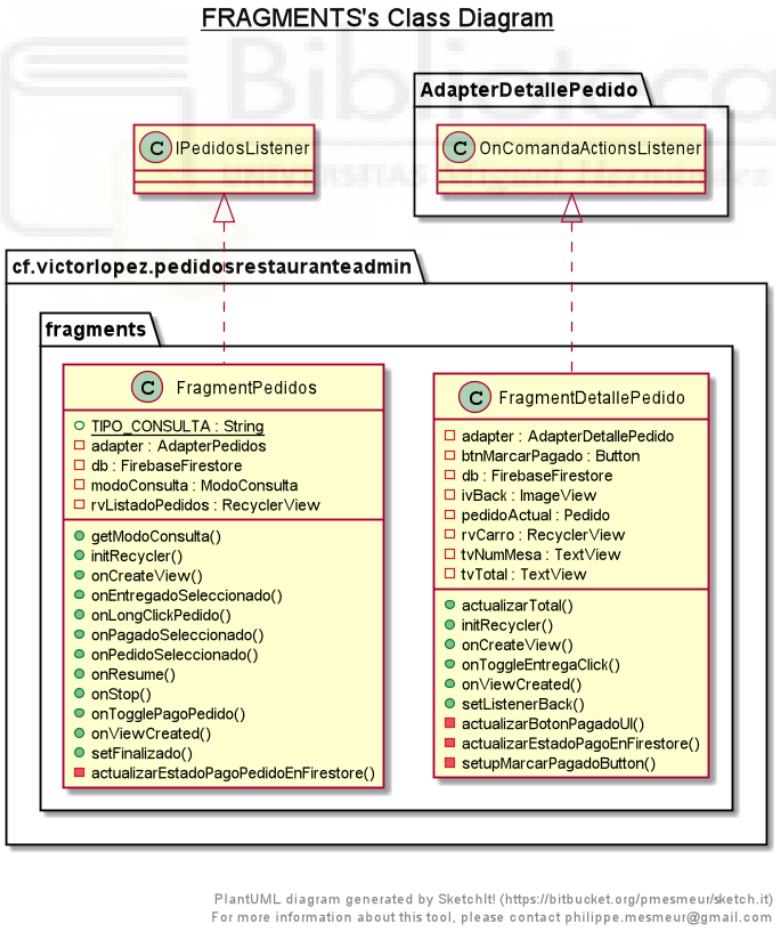

Ilustración 23. Diagrama de clase - Aplicación gestión de pedidos - Paquete de fragments

Esta distinción se hace mediante un enumerado que pasamos como parámetro desde la activity al cambiar la opción del menú.

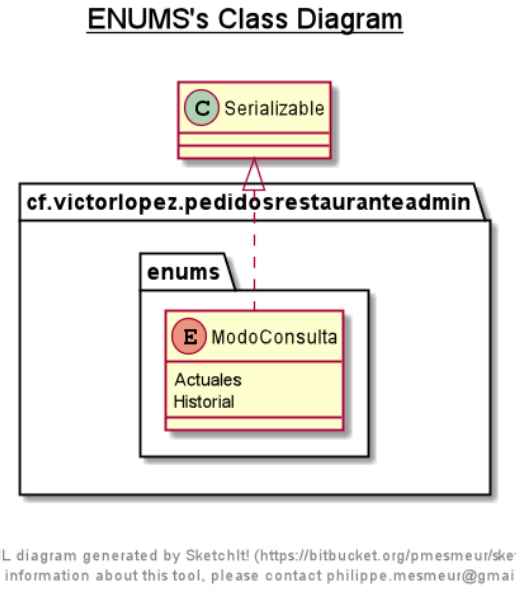

Ilustración 24. Diagrama de clase - Aplicación gestión de pedidos - Paquete de enums

Por último, tenemos el grueso de la lógica de negocio para mostrar los pedidos e interactuar con ellos.

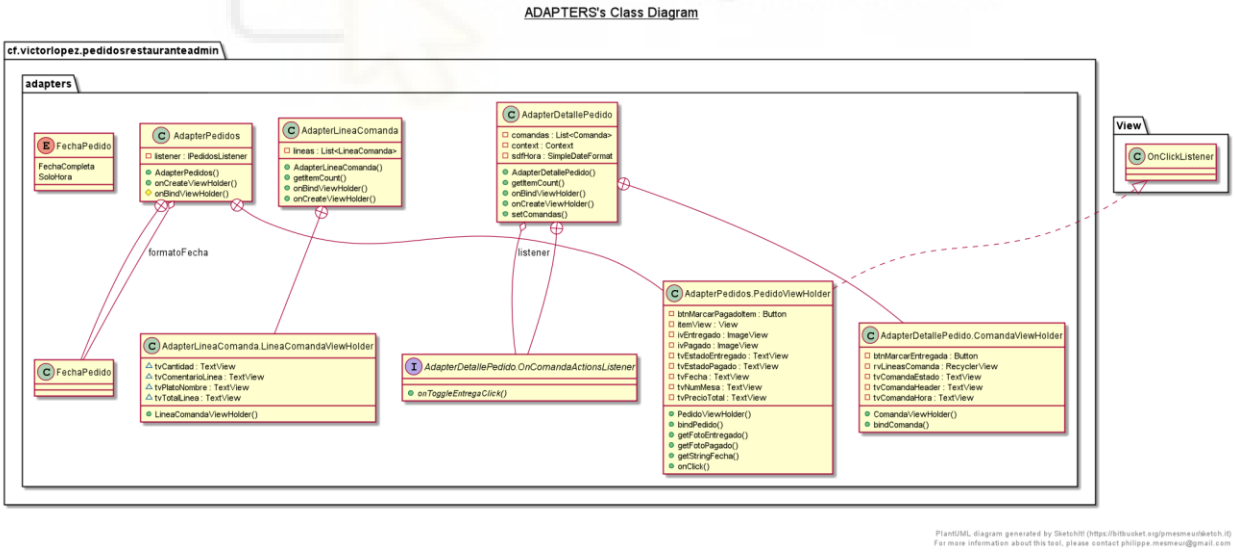

Ilustración 25. Diagrama de clase - Aplicación gestión de pedidos - Paquete de adapters

Diagrama de secuencia

En este último diagrama podemos ver cómo interactúan los diferentes elementos de la aplicación para todas las acciones que podemos realizar.

Se puede ver el flujo de inicio de sesión, los cambios de estado de los pedidos y los cambios de opción entre actuales e histórico.

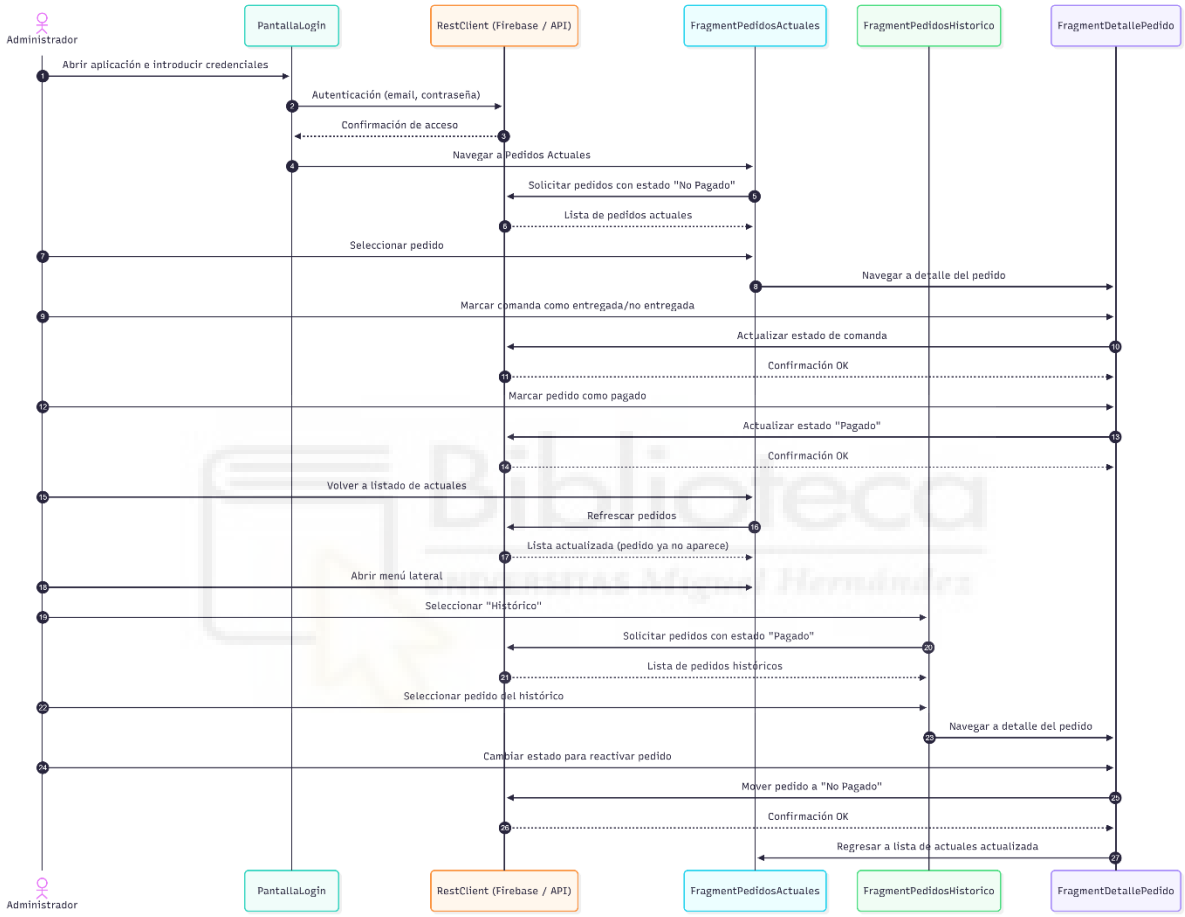

Ilustración 26. Diagrama de secuencia - Aplicación gestión de pedidos

CAPÍTULO 5: DISEÑO DE LA SOLUCIÓN

5.1. Arquitectura general del sistema

El sistema de aplicaciones desarrollado contiene tres aplicaciones completamente independientes, las cuales interactúan para poder gestionar la carta, los pedidos de los clientes y la administración de estos. A continuación, describiremos la arquitectura y las decisiones de diseño más importantes:

Aplicación web (gestión carta)

Esta aplicación sigue una arquitectura MVC (Modelo-Vista-Controlador) utilizando Spring Boot como framework principal. Para el renderizado de vistas se utiliza Thymeleaf, lo que permite generar páginas HTML dinámicas basadas en los datos de la carta que tenemos almacenados en la base de datos SQL.

Además, la aplicación pone a disposición de las demás un servicio RESTful, el cual utilizan para consultar información de la carta, como platos, categorías y fotografías. La aplicación implementa Spring Security, permitiendo la diferenciación de roles entre usuarios comunes y administradores, además de obviamente usuarios anónimos (que no han iniciado sesión). Actualmente, los usuarios pueden ver el detalle de platos y categorías, mientras que los administradores tienen permisos completos para la gestión de la carta (crear, modificar y eliminar).

Aplicación móvil de pedidos (cliente)

La aplicación móvil sigue un diseño modular, el cual organiza el código por paquetes de componentes, como los adapters, los fragments, activities, etc. La navegación entre pantallas se realiza mediante la sustitución de fragments, lo que nos permite una navegación más fluida y eficiente frente a la creación de múltiples activities.

El sistema de pedidos utiliza Firestore de Firebase como base de datos NoSQL, en la que guardamos los pedidos asociados a cada número de mesa. Utilizamos NoSQL por posibles cambios en la estructura de pedidos a lo largo del tiempo.

La autenticación de administrador se realiza mediante el sistema de Firebase, permitiendo únicamente cambiar el número de mesa desde un modal protegido. El resto de las funcionalidades están disponibles sin necesidad de iniciar sesión.

Aplicación móvil de gestión de pedidos (administración)

Esta aplicación consume los pedidos almacenados en Firestore de Firebase y modifica sus estados. Actualmente permite marcar pedidos como pagados y consultar el detalle de estos. Además, podemos marcar cada comanda individualmente como entregada, en caso de tener todas las comandas entregadas, el pedido se considera como entregado.

Dispone de un componente que permitirá consumir información adicional del servicio RESTful para futuros cambios, como mostrar la fotografía de los platos o visualizar la carta.

Principios generales de arquitectura

Independencia de aplicaciones: Aunque las aplicaciones estén interconectadas entre sí, son completamente independientes, lo que en un futuro nos facilitará el mantenimiento y la posible ampliación o cambio del sistema. Por ejemplo, se podrían añadir nuevas funcionalidades o incluso cambiar el sistema completo del servicio RESTful y se podrían seguir utilizando perfectamente las demás.

Modularidad: Las tres aplicaciones están diseñadas de manera modular, pudiendo añadir nuevas funcionalidades sin reestructurar el código.

Seguridad: La gestión de usuarios y roles se centraliza en cada aplicación según sus necesidades. La web utiliza Spring Security con gestión de roles, mientras que las aplicaciones móviles se apoyan en Firebase Authentication para los accesos de administrador.

5.2. Descripción de las aplicaciones

5.2.1. Aplicación web en Spring Boot (gestión de carta y servicio RESTFUL)

Arquitectura y estructura de paquetes

Como hemos especificado anteriormente, la aplicación web sigue el patrón MVC, mediante Spring Boot, utilizando el motor de plantillas Thymeleaf para renderizar las vistas. La organización en paquetes facilita la escalabilidad y el mantenimiento, algunos de los paquetes más importantes de la aplicación son:

- Auth y auth.handler: Gestión de autenticación y manejo de seguridad mediante Spring Security.
- Controllers: Controladores que gestionan la navegación entre pantallas y hacen las llamadas a las vistas de la aplicación.
- Restcontrollers: Controladores del servicio RESTful, utilizado por otras aplicaciones para consumir la carta completa.
- Models: Definición de entidades, acceso a la BBDD SQL y mapeos al dominio de la aplicación mediante JPA/Hibernate.
- Db: Configuración de la BBDD SQL.
- Util.paginator: Utilidades para paginación en la web.
- Static/images y templates/layout: Recursos estáticos y plantillas Thymeleaf.

Esta organización modular nos permite añadir nuevas funcionalidades en el futuro, como separar bebidas de platos o crear roles adicionales sin necesidad de reestructurar todo el proyecto.

Flujo de datos y servicios

La carta completa junto a los usuarios se almacena en la base de datos SQL. La carta es accesible tanto para la visualización de la web como para el servicio RESTful.

Las aplicaciones móviles no dependen directamente de la web, aunque sí consumen el servicio RESTful para obtener la carta, garantizando independencia entre apps.

El diseño asegura que cada aplicación pueda evolucionar por separado y añadir nuevas funcionalidades sin afectar a las demás.

Tecnologías utilizadas

Backend: Spring Boot, Spring Security, JDBC/Hibernate para acceso a la base de datos.

Frontend: Thymeleaf, HTML5, CSS, JavaScript.

Base de datos: SQL (PostgreSQL en local, MariaDB en IONOS VPS)

Servicios RESTful: JSON para intercambio de información con otras aplicaciones.

Seguridad: Autenticación y autorización basada en roles (usuario común y administrador).

Entorno de desarrollo local: Docker para base de datos PostgreSQL.

Despliegue a producción: VPS Linux S de IONOS, con control completo sobre el servidor y los servicios.

Flujo de trabajo

- Los usuarios anónimos pueden consultar la carta, en concreto el listado de platos y categorías.
- Los usuarios registrados (usuarios comunes) pueden acceder al detalle de categorías y platos, preparándose para futuras funcionalidades con permisos ampliados.
- Los administradores pueden realizar TODAS las acciones, además de las mencionadas anteriormente pueden crear, modificar y eliminar platos y categorías, así como acceder a todos los servicios RESTful (implementación token a futuro).

5.2.2. Aplicación para clientes (realización de pedidos)

La aplicación de pedidos para clientes está diseñada para que los usuarios puedan navegar por la aplicación consultando la carta del restaurante y realizar pedidos desde una tablet o dispositivo móvil.

Su arquitectura es modular y nos centramos en la utilización de fragments para la navegación entre pantallas, evitando cambiar de activity para mejorar el rendimiento y la experiencia de usuario.

Arquitectura y estructura

La aplicación está compuesta por paquetes organizados por funcionalidad y tipo de componente:

- Adapters: Contiene adaptadores para listas de elementos.
- Dialogs: Diálogos y modales, como el de añadir al carrito o cambiar el número de mesa.
- Fragments: Fragmentos de interfaz para cada pantalla: listado de categorías, platos, detalle del plato, etc.
- Interfaces: Interfaces para callbacks y comunicación entre componentes, ya que en Java no se pueden pasar funciones por parámetro.
- Managers: Clases de gestión, como control de carrito y manejo de sesiones.
- Models: Clases de datos que representan categorías, platos, pedidos y comandas.

- Rest: Contiene RestClient.java, que se encarga de la comunicación con el servicio RESTful proporcionado por la primera aplicación mencionada.

La activity principal (MainActivity) se encarga de gestionar la navegación entre fragments y el menú de hamburguesa.

Flujo de la aplicación

Al iniciar la aplicación, lo primero que ve el usuario es el listado de categorías, mostrando, por cada una, una fotografía y su nombre.

Al seleccionar una categoría, se despliega el listado de platos correspondientes.

Si seleccionamos un plato, accedemos a su detalle, donde se muestra la descripción del plato, fotografía, precio y un botón para añadir al carrito.

Al seleccionar dicho botón de añadir al carrito, se nos abre un modal preguntando por la cantidad mediante botones de incremento/decremento y un comentario (por ejemplo, poco hecho). Al confirmarlo, el plato se guarda en el carrito, accesible desde un icono en la parte superior derecha de la pantalla.

Al acceder al carrito podemos ver el listado de platos añadidos y seleccionar el botón de "Realizar pedido", el cual abre un modal de confirmación. En caso de confirmar, se hace el pedido y se vacía el carrito.

El menú lateral (hamburguesa) se encuentra en la parte superior izquierda, el cual contiene la opción de iniciar sesión. Solo los usuarios administradores pueden iniciar sesión, habilitando las opciones cambiar número de mesa y cerrar sesión. La opción de cambiar número de mesa abre un modal que permite seleccionar y guardar el número en el dispositivo, mientras que cerrar sesión revierte la interfaz al estado original.

Tecnologías utilizadas

- **Android nativo** (Java)
- **Fragments** para la navegación entre pantallas
- **Retrofit** para consumir la API RESTful de la aplicación web
- **Picasso** para cargar imágenes en Base64 recibidas desde la API

- **Firebase Firestore** para almacenamiento de pedidos y autenticación de administradores
- **Modales y diálogos** personalizados para interacción del usuario.

Independencia y escalabilidad

Como hemos mencionado anteriormente, cada aplicación funciona de manera independiente, por lo que nos permite que la aplicación evolucione y se añadan nuevas funcionalidades sin afectar prácticamente a las demás aplicaciones. Esto garantiza la modularidad, escalabilidad y facilidad de mantenimiento.

5.2.3. Aplicación de gestión de pedidos (pedidos actuales e histórico)

La aplicación de gestión de pedidos es la herramienta que más van a usar el personal del restaurante, es la que nos permite controlar en tiempo real los pedidos realizados por los clientes desde sus mesas. Esta aplicación ha sido desarrollada en Android utilizando Java como lenguaje, siguiendo una arquitectura modular.

Flujo de navegación

Al iniciar la aplicación, lo primero que el usuario ve es la pantalla de inicio de sesión, sin iniciar sesión como administrador no podremos continuar.

Al iniciar sesión nos redigirá automáticamente a la pantalla de pedidos actuales, sin necesidad de pasar por una pantalla intermedia. El menú lateral ofrece dos opciones sobre las cuales podemos navegar libremente:

- Pedidos actuales
- Pedidos históricos

Los pedidos actuales son los que se encuentran como no pagados, los del histórico ya han sido pagados.

Funcionalidades principales

En la sección de Pedidos actuales, el personal puede visualizar en un listado todos los pedidos activos (no pagados). Desde esta pantalla, es posible marcar un pedido como pagado, lo que lo traslada al histórico. Además, cada pedido puede abrirse para acceder a su detalle, donde se muestran todas sus comandas, con sus respectivos platos y comentarios.

Dentro del detalle, se permite realizar dos tipos de acciones:

- Marcar comanda como entregada, si todas las comandas están marcadas como entregadas el pedido también está en estado entregado.
- Marcar pedido como pagado, en caso de que se haya completado el proceso (también se puede desde el listado).

En la sección histórico de pedidos, se muestran los pedidos ya completados y pagados. Un pedido puede volver a trasladarse a los pedidos actuales en caso de que se modifique su estado de pago, garantizando flexibilidad.

Integración con servicios

La aplicación se comunica con Firebase Firestore, que actúa como base de datos NoSQL en la nube, lo que nos proporciona sincronización en tiempo real entre los diferentes dispositivos. Además, la app ya está preparada para integrar su funcionamiento con el servicio RESTful mediante el uso de Retrofit.

Estructura y componentes principales

- Adapters: Contienen los adaptadores para renderizar listados de pedidos y comandas
- Enums: Definen parámetros internos como el modo de consulta de pedidos (actual o histórico)
- Fragments: Incluyen dos fragmentos principales, uno para la gestión de pedidos y otro para el detalle de un pedido.
- Rest: Módulo encargado de la comunicación con servicios externos mediante Retrofit.
- Models: Definen las estructuras de datos para representar pedidos, comandas y estados internos de la aplicación.

Esta organización nos asegura la escalabilidad del sistema y un mantenimiento sencillo, permitiendo la incorporación de futuras funcionalidades de forma sencilla.

5.3. Modelado de datos y base de datos

5.3.1. Modelo relacional (carta y usuarios web)

En la aplicación web, la base de datos relacional se utiliza para almacenar tanto la carta del restaurante como la información de los usuarios que inician sesión en dicha página.

Las tablas principales son:

Categoria

- id (INT, PK, autoincremental)
- nombre (VARCHAR(255), único, no nulo)
- descripcion (VARCHAR(255), no nulo)
- foto (VARCHAR(255))

Plato

- id (INT, PK, autoincremental)
- nombre (VARCHAR(255), único, no nulo)
- descripcion (VARCHAR(255), no nulo)
- precio (DOUBLE, no nulo, mínimo 0)
- foto (VARCHAR(255))
- id_cat (INT, FK → categoria.id, no nulo)

Users

- id (INT, PK, autoincremental)
- username (VARCHAR(45), único, no nulo)
- password (VARCHAR(60), hasheado, no nulo)
- enabled (TINYINT(1), indica si el usuario está activo)

Authorities (gestiona los roles de los usuarios)

- id (INT, PK, autoincremental)
- user_id (INT, FK → users.id)
- authority (VARCHAR(45), define el rol, ej. ROLE_ADMIN o ROLE_USER)

Relaciones principales

La relación entre categoría y plato es 1:N, ya que una categoría puede tener múltiples platos asociados, mientras que cada plato solamente puede pertenecer a una categoría.

Por otro lado, entre users y authorities a nivel implementación es una relación 1:N, dado que un usuario puede tener varios registros en la tabla authorities. Sin embargo, conceptualmente se trata de una relación N:M, ya que un mismo rol puede estar asignado a múltiples usuarios.

Otros aspectos técnicos

Los usuarios se gestionan mediante Spring Security, que utiliza esta estructura de tablas para controlar autenticación y autorización.

Las contraseñas se almacenan utilizando algoritmos de hashing, para evitar robos de contraseñas.

La base de datos se ejecuta en PostgreSQL durante el desarrollo en local (a través de contenedores Docker) y en MariaDB en producción en el servidor VPS de IONOS.

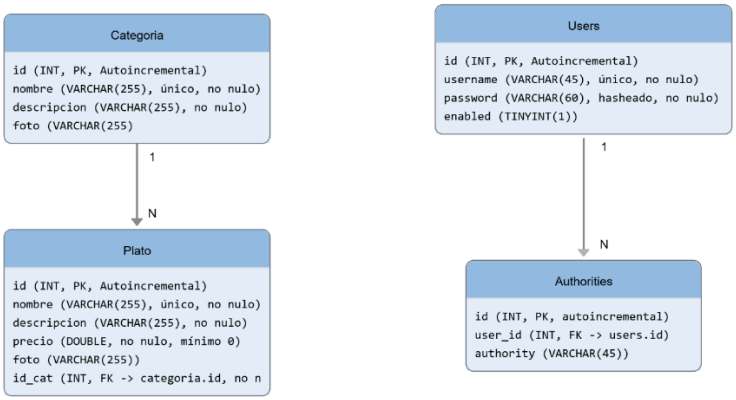

Ilustración 27. Diagrama BBDD aplicación Web Spring Boot

5.3.1. Modelo NoSQL (pedidos en Firebase Firestore)

Los pedidos se modelan en Firestore siguiendo una estructura jerárquica, donde cada documento representa un pedido y tiene diferentes comandas dentro. Esta organización nos facilita la actualización en tiempo real y la consulta de información desde las aplicaciones móviles.

La estructura general es la siguiente:

Pedido

- id (String, generado por Firestore)
- numMesa (int)
- pagado (booleano)
- fecha (Date, fecha de creación del pedido)
- comandas (List<Comanda> lista de comandas)

Comanda (subcolección dentro de cada pedido)

- horaPedida (Date)
- entregado (booleano)
- lineasComanda (List<LineaComanda> lista de líneas de comanda)

LineaComanda (elementos dentro de una Comanda)

- plato (Plato)
- cantidad (int)
- comentarioLinea (String)
- totalLinea (Double)

Este modelo permite que un pedido contenga múltiples comandas y que, cada comanda, incluya varias líneas con sus platos correspondientes. Además, el estado del pedido se actualiza dinámicamente.

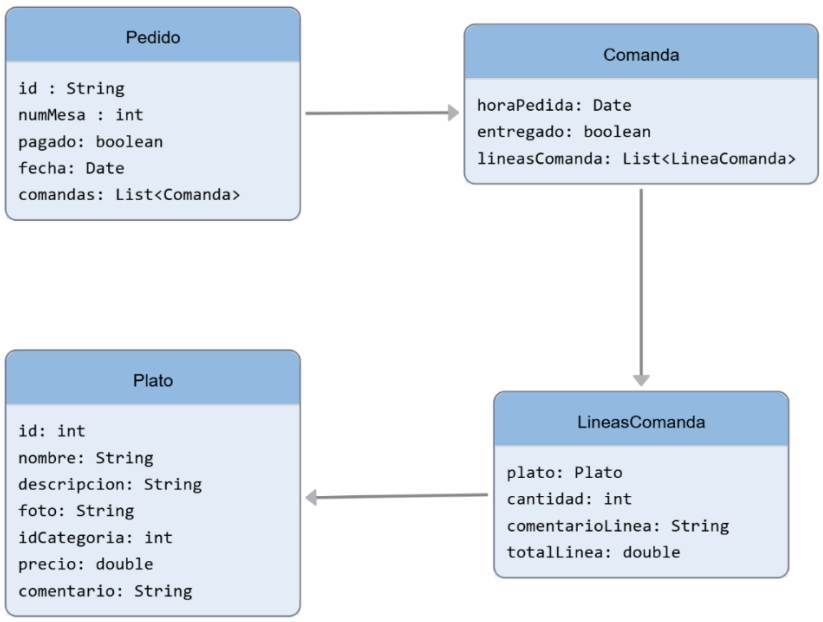

Ilustración 28. Modelo de datos de pedidos en Firebase

5.4 Diseño de interfaces gráficas

En este apartado describiremos la interfaz de usuario de las tres aplicaciones desarrolladas.

5.4.1. Aplicación web

La aplicación web está diseñada siguiendo una arquitectura MVC y utiliza Thymeleaf para el renderizado para el renderizado dinámico de las páginas, combinando HTML, CSS y Bootstrap para la interfaz gráfica. La interfaz se organiza en las siguientes páginas principales:

- **Página principal:** muestra accesos directos a los listados, tanto de categorías como de platos, permitiendo al usuario navegar.
- **Listado de categorías:** al acceder al listado, podremos observar un listado paginado de categorías, en el que tenemos botones para eliminarlas, modificarlas o añadir una nueva.
- **Detalle de categoría:** al seleccionar una categoría se puede ver su detalle completo, incluyendo nombre, descripción, imagen, etc.
- **Listado de platos:** al acceder al listado, podremos observar un listado paginado de platos, en el que tenemos botones para eliminarlos, modificarlos o añadir uno nuevo.

- **Detalle de plato:** al seleccionar un plato se puede ver su detalle completo, incluyendo nombre, descripción, imagen, precio, etc.

El diseño se centra en la usabilidad, manteniendo menús claros, botones de acción destacados y un flujo de navegación intuitivo y sencillo. Los usuarios administradores disponen de funciones adicionales como crear, modificar o eliminar categorías o platos.

Esta es la idea original de la pantalla principal, que contiene accesos directos a ambos listados y al inicio de sesión.

Ilustración 29. Wireframe página principal aplicación web

La idea original del listado de categorías es el listado con todas sus opciones paginado.

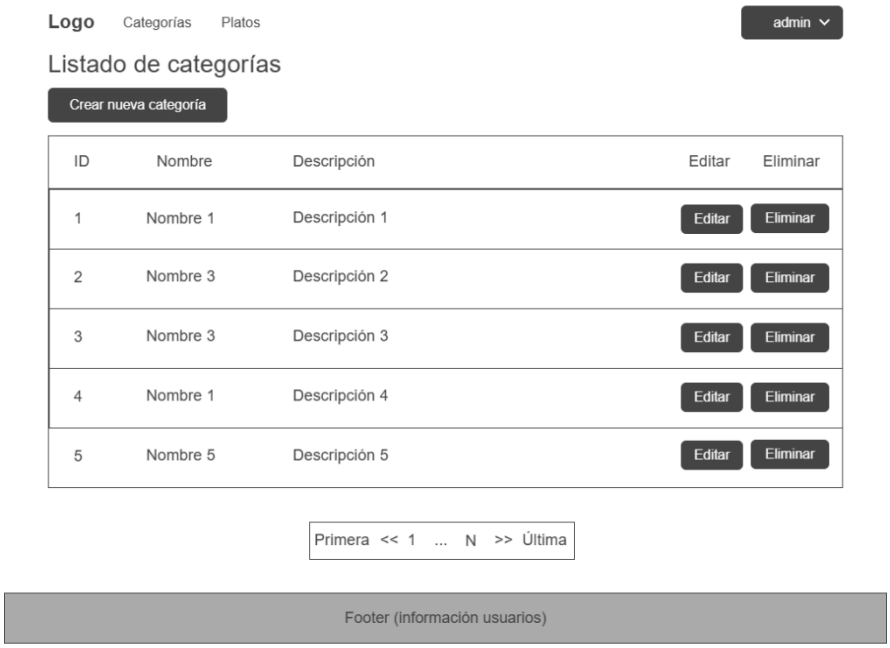

Ilustración 30. Wireframe listado de categorías aplicación web

Igual que la página anterior, tenemos el listado de platos, el cual contiene prácticamente la misma información y estilo.

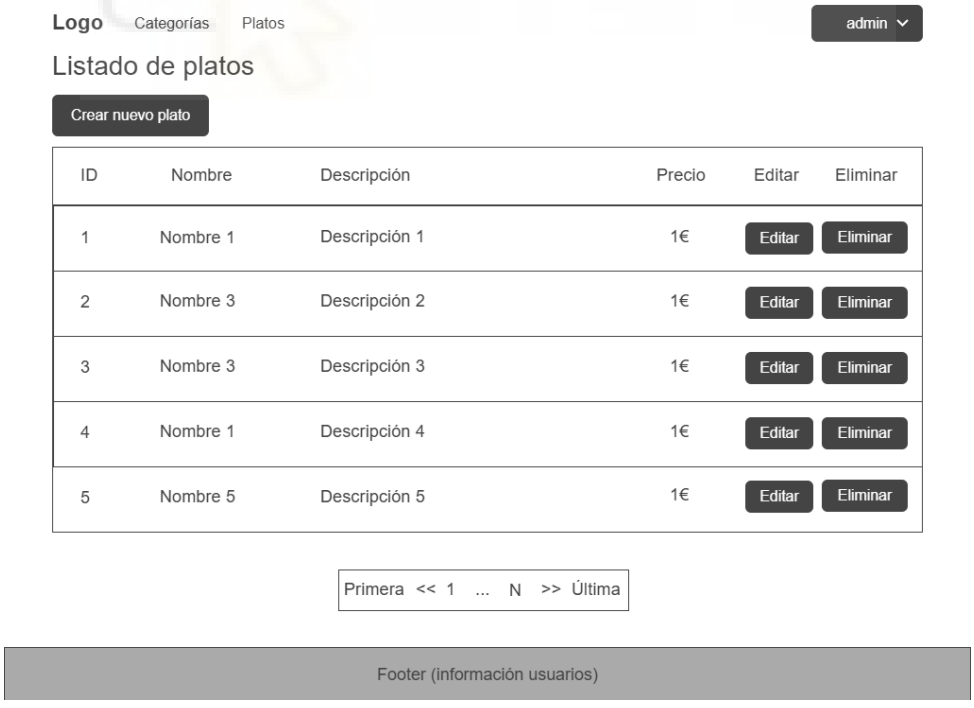

Ilustración 31. Listado de platos aplicación web

Al seleccionar un plato, podemos acceder a su detalle, el cual tendrá este estilo:

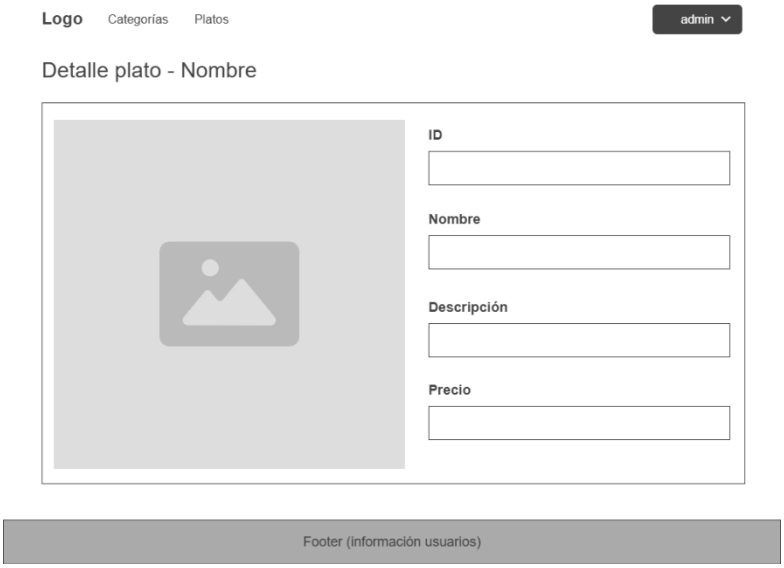

Ilustración 32. Wireframe detalle de plato aplicación web

Ocurre el mismo caso para las categorías, el diseño será el mismo, pero con toda la información acerca de la categoría.

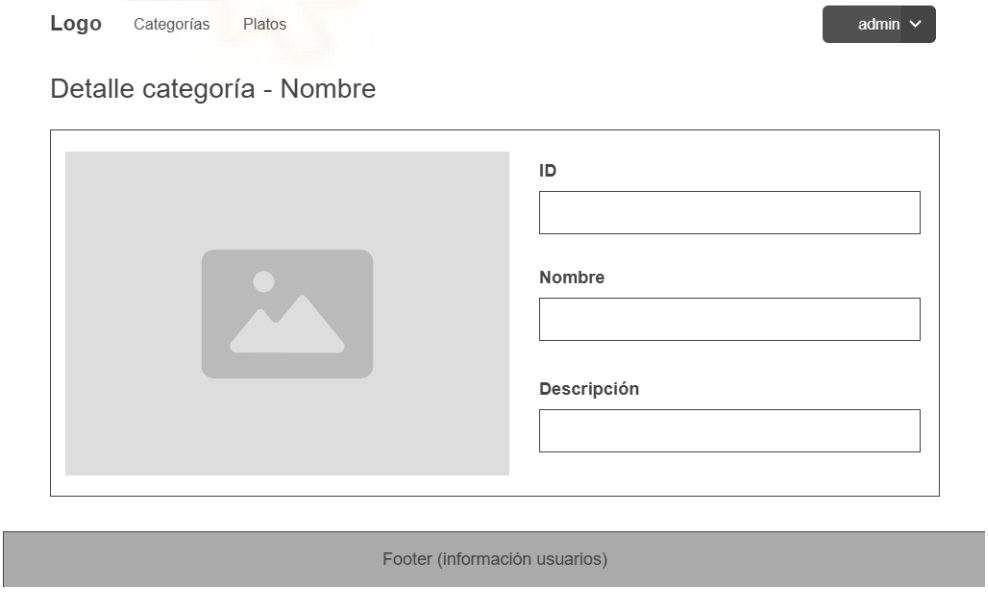

Ilustración 33. Wireframe detalle categoría aplicación web

Para cuando creamos o modificamos un plato nuevo se carga la misma vista, pero rellenando la información con la del plato correspondiente o dejándolo vacío.

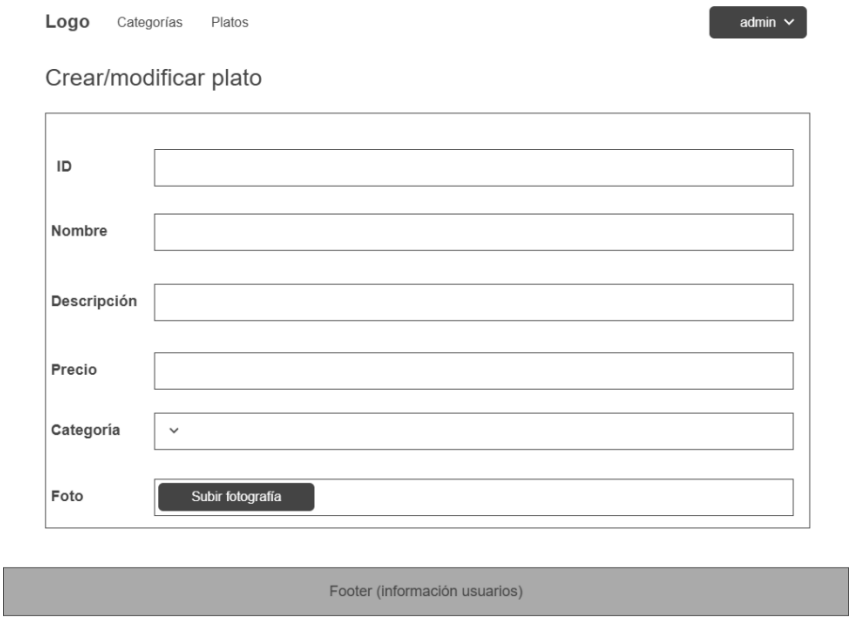

Ilustración 34. Wireframe detalle plato aplicación web

El mismo caso ocurre para el detalle de las categorías, utilizamos la misma plantilla para crear y para modificar.

Ilustración 35. Wireframe nueva categoría aplicación web

Por último, tenemos la página de inicio de sesión, la cual nos redirigirá a la página de inicio si el inicio es correcto.

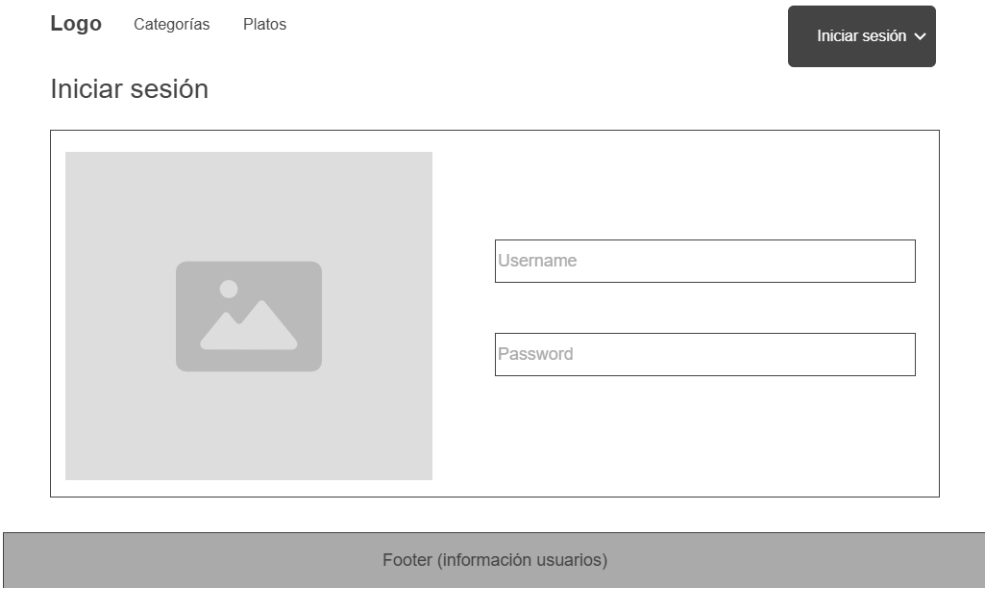

Ilustración 36. Wireframe inicio de sesión aplicación web

5.4.2. Aplicación de pedidos (cliente)

La aplicación móvil para clientes está desarrollada para Android utilizando el lenguaje de programación Java, con una arquitectura modular que separa los elementos por tipo en paquetes: adapters, fragments, views, managers, models e interfaces. La navegación entre pantallas se realiza mediante reemplazo de fragments dentro de la MainActivity, lo que permite un cambio rápido de pantallas sin necesidad de volver a crear actividades completas.

La interfaz se organiza en las siguientes páginas principales:

- **Pantalla principal:** Al acceder a la aplicación, se muestra directamente el listado de categorías, cada una con su imagen y nombre.
- **Listado de platos:** Al seleccionar una categoría, se muestra el listado de platos asociados a dicha categorías con su foto y nombre.
- **Detalle de plato:** Al seleccionar un plato, se accede a su detalle completo, incluyendo nombre, descripción, precio e imagen. Además, nos muestra un botón para añadirlo al carrito.

- **Modal añadir al carrito:** Al pulsar el botón para añadir al carrito, se abre un modal donde el usuario puede indicar la cantidad mediante botones de incremente y decremento y un campo de texto para añadir un comentario.
- **Carrito:** Accesible desde el icono situado arriba a la derecha en la toolbar, permite al usuario acceder al carrito con todos los platos, sus cantidades y comentarios. Además, nos permite realizar el pedido, que vacía el carrito y nos redirige a la pantalla principal.
- **Menú lateral:** el menú hamburguesa, situado en la esquina superior izquierda, permite al usuario iniciar sesión como administrador a través de un modal para habilitar funciones como cambiar el número de mesa y cerrar sesión.

En cuanto a las pantallas de esta aplicación móvil, vamos a listar las más importantes, entre ellas el listado de categorías, el listado de platos, el detalle de los platos y el carrito. Existen varias versiones según el dispositivo, en este caso vamos a hacer wireframes de la versión de tablet. Empezando con la pantalla de inicio de la aplicación, el listado de categorías:

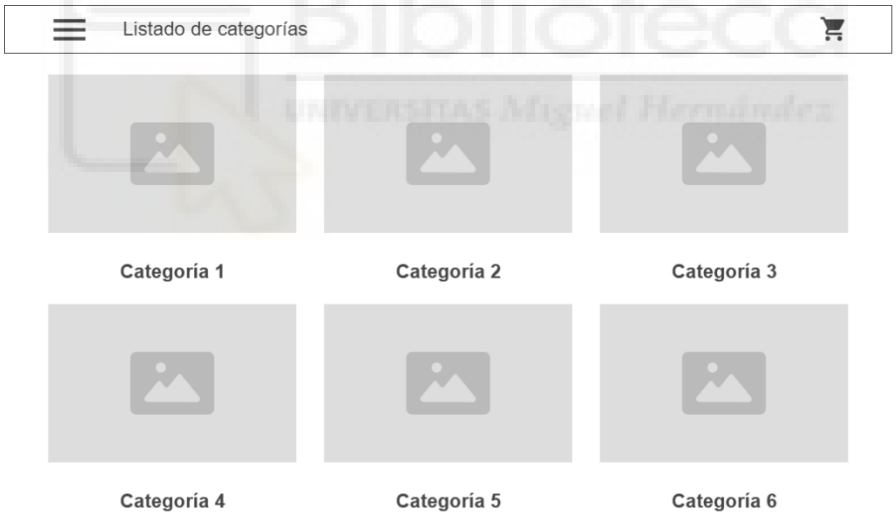

Ilustración 37. Wireframe listado de categorías aplicación móvil clientes

Al seleccionar una de las categorías, nos abrirá un listado de platos, en el que veremos un estilo muy parecido, pero con el precio al lado del nombre:

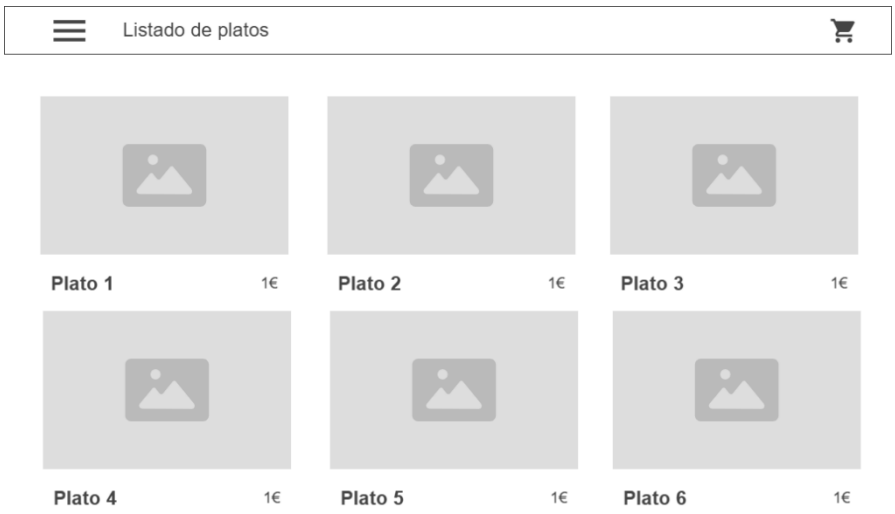

Ilustración 38. Wireframe listado de platos aplicación móvil clientes

Al seleccionar un plato, podemos acceder a su detalle, el cual nos muestra toda la información necesaria, junto a un botón de añadir al carrito.

Ilustración 39. Wireframe detalle de plato aplicación móvil clients

Al acceder al carrito, vemos un listado de productos con la cantidad y su comentario.

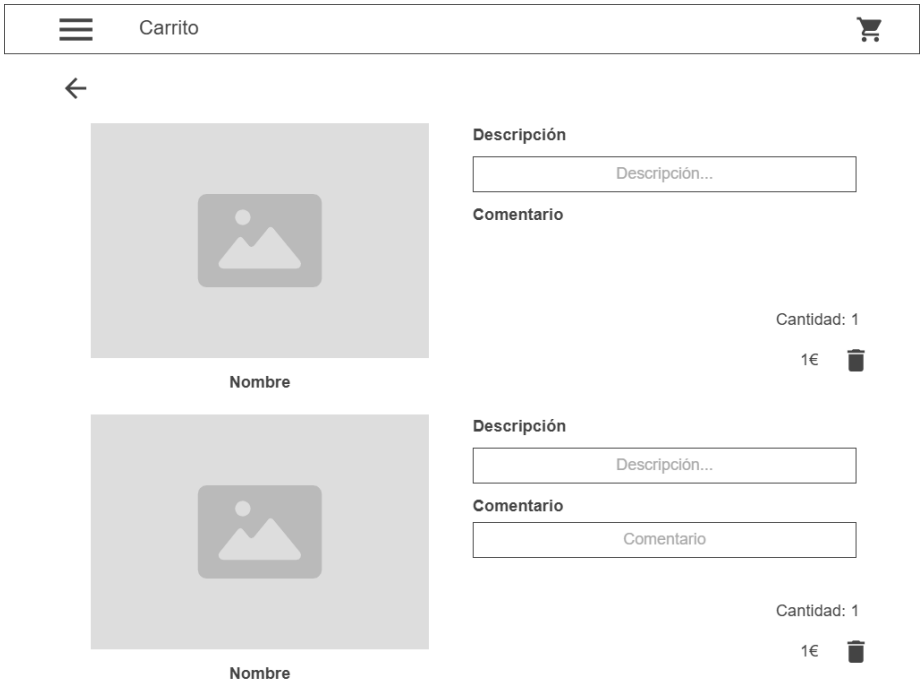

Ilustración 40. Wireframe carrito aplicación pedidos clientes

5.4.3. Aplicación de gestión de pedidos (interna)

La aplicación móvil de gestión de pedidos está desarrollada para Android utilizando Java, con una arquitectura modular que organiza los elementos por tipo en paquetes. La navegación entre pantallas dentro del flujo normal después del inicio de sesión se hace mediante reemplazo de fragments. La aplicación cuenta con dos activities principales: la LoginActivity y la MainActivity, donde se gestionan los fragments de listado de pedidos y detalle de pedidos.

La interfaz se organiza en las siguientes pantallas principales:

- **Pantalla de login:** Permite al usuario autenticarse mediante Firebase Authentication.
- **Listado de pedidos activos:** Tras iniciar sesión, se muestra directamente el listado de pedidos actuales. Cada pedido indica si está pagado y si se encuentra entregado (todas sus comandas se han entregado). Además, se puede marcar como pagado o no pagado desde el propio listado.
- **Histórico de pedidos:** Desde el menú lateral, se puede cambiar al histórico de pedidos, el cual nos muestra los pedidos que han sido pagados.

- **Detalle de pedido:** Al acceder al detalle de un pedido, podemos ver el listado de comandas que lo componen. Cada comanda muestra los platos incluidos, sus cantidades y comentarios. Es posible marcar cada comanda individualmente como entregada, además de marcar el pedido como pagado.

El diseño modular permite futuras ampliaciones y mejoras, como adición de filtros, estadísticas o nuevas funcionalidades. Los adaptadores se encargan de renderizar los pedidos y sus comandas en los fragments correspondientes, mientras que los enums permiten gestionar el modo de visualización (actuales o histórico).

En esta aplicación realmente solo tenemos 3 pantallas, una de ellas para iniciar sesión, otra para el listado de pedidos (se utiliza la misma para pedidos del histórico y actuales) y otra para el detalle de un pedido.

Al acceder a la aplicación veremos la pantalla de inicio de sesión, que tendrá este diseño:

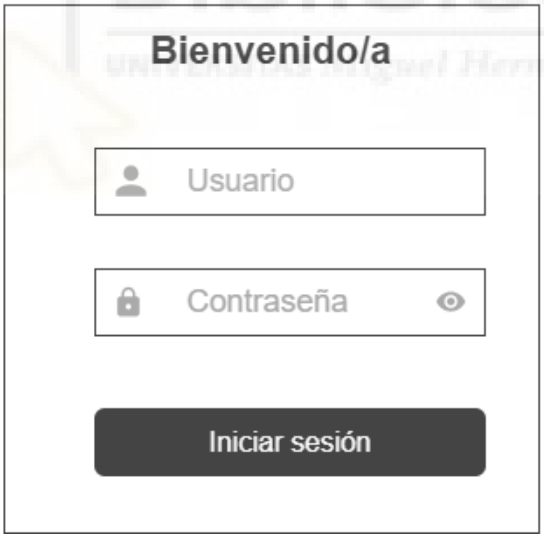

Ilustración 41. Wireframe página de inicio de sesión aplicación gestión de pedidos

Al iniciar sesión nos mostrará la pantalla del listado de pedidos actuales, que es la siguiente:

Ilustración 42. Wireframe página de listado de pedidos aplicación de gestión de pedidos

Al acceder a un pedido, veremos su listado de comandas junto al total del pedido, de esta forma:

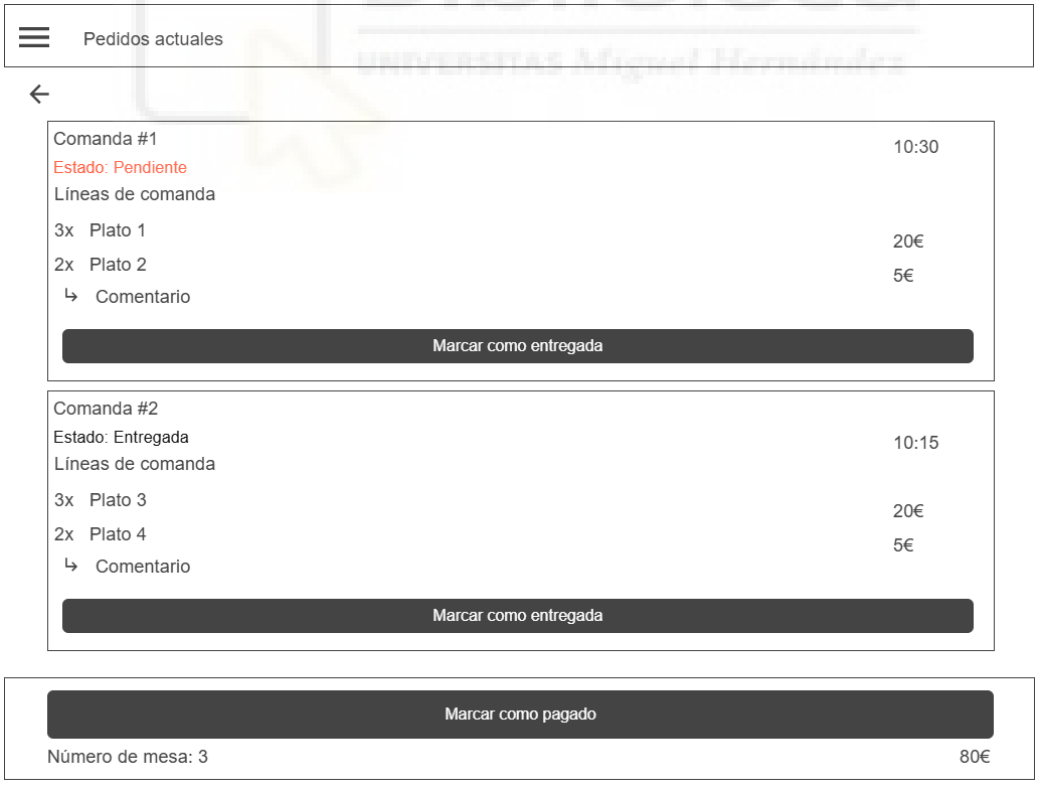

Ilustración 43: Wireframe detalle pedido aplicación gestión de pedidos

5.5. Seguridad y gestión de usuarios

En este caso, la seguridad y la gestión de usuarios se gestiona de forma diferente en la web con respecto a las aplicaciones móviles, para garantizar completa independencia, ya que la persona que puede acceder a modificar la carta no tiene por qué ser la misma que pueda gestionar los pedidos.

Aplicación web

La aplicación web utiliza Spring Security para la gestión de usuario y control de acceso. Los usuarios se almacenan en la base de datos relacional y se asignan roles mediante la tabla authorities.

- **Usuarios:** Se registran con un nombre de usuario único y una contraseña hasheada. La columna enabled indica si el usuario está activo.
- **Roles:** Cada usuario puede tener múltiples roles y un rol puede ser compartido por varios usuarios. Esto nos permite controlar el acceso a diferentes funcionalidades, como modificar categorías y platos o eliminarlos.
- **Control de acceso:** Dependiendo del rol, se limita el acceso a ciertas páginas y acciones. Actualmente, los usuarios registrados tienen acceso a los detalles de categorías y platos, mientras que los administradores pueden crear, modificar o eliminar elementos de la carta.

Aplicación móvil para clientes

En la aplicación móvil para clientes, no es necesario iniciar sesión para realizar pedidos, solamente para cambiar el número de mesa. Para este login se utiliza Firebase Authentication, que permite gestionar usuarios de manera segura sin necesidad de mantener una base de datos propia para la aplicación.

Aplicación móvil de gestión de pedidos

La aplicación de gestión de pedidos también utiliza este mismo sistema de Firebase Authentication para el inicio de sesión, pero éste sí es necesario para poder usar la aplicación. Esto garantiza que el personal autorizado pueda acceder al listado de pedidos, modificar estados de pago y entrega y consultar pedidos históricos.

Consideraciones generales de seguridad

Todas las aplicaciones funcionan de manera independiente, lo que significa que un fallo o acceso indebido no afectaría a las demás.

Se emplean conexiones seguras a los servicios backend (REST o Firebase) y se validan los datos en ambas direcciones (cliente y servidor).

Los roles y permisos están diseñados para permitir futuras ampliaciones sin necesidad de reestructurar completamente el sistema de seguridad.

CAPÍTULO 6: DESARROLLO E IMPLEMENTACIÓN

6.1. Tecnologías y herramientas utilizadas

El desarrollo del sistema de gestión de pedidos para restaurantes ha requerido el uso de varias tecnologías, seleccionadas en base a la compatibilidad y facilidad de integración. A continuación, se detallan las principales empleadas durante el proceso de desarrollo y despliegue de cada uno de los componentes que forman la solución.

La organización temporal y técnica del proyecto se ha planificado siguiendo las recomendaciones de McConnel [42] y Dawson [43], centradas en la correcta estimación de esfuerzos, la definición de hitos y el seguimiento del proceso a lo largo del ciclo de ingeniería del software.

6.1.1 Entornos de desarrollo

Para la implementación del sistema se han utilizado dos IDE (entornos de desarrollo integrados) según para cada tipo de tecnología.

- **IntelliJ IDEA** se utilizó para el desarrollo de la aplicación web basada en Spring Boot. Este entorno ofrece soporte nativo para proyectos Gradle, integración con Git y herramientas avanzadas de depuración.
- **Android Studio** fue empleado en el desarrollo de las dos aplicaciones móviles, tanto la destinada a clientes como la de gestión de pedidos.
  Su integración con Firebase, soporte para emuladores y su facilidad de creación de vistas y gestión de dependencias lo convierten en la solución perfecta para este tipo de soluciones.

El control de versiones se gestionó mediante Git, utilizando la plataforma GitHub para almacenarlo, lo que me permitió mantener un historial de cambios claro y estructurado. Para facilitar resolución de posibles errores y recuperación de diferentes versiones.

6.1.2. Tecnologías de backend

El servidor que aloja la aplicación web para gestionar la carta se ha desarrollado utilizando Spring Boot, un framework que simplifica enormemente la creación de aplicaciones Java basadas en Spring, proporcionando un entorno de configuración mínima y alta escalabilidad.

Entre las tecnologías más relevantes se encuentran:

- **Spring web:** para la creación de controladores RESTful que permiten la comunicación entre el servidor y las aplicaciones móviles.
- **Spring Data JPA:** para la gestión de la persistencia de datos mediante el patrón Repository.
- **Hibernate ORM:** como proveedor JPA responsable del mapeo objetos-relacional, encargándose de traducir las entidades Java a las tablas correspondientes en la BBDD.
- **Spring Security:** encargado de la autenticación y autorización de usuarios, implementando un sistema de control de acceso basado en roles.
- **Thymeleaf:** utilizado como motor de plantillas para las vistas de la aplicación web, permitiendo la generación dinámica de contenido HTML.

Durante la fase de desarrollo en local, se utilizó PostgreSQL como sistema de gestión de base de datos, ejecutado mediante un contenedor de Docker, lo que permitió replicar un entorno de trabajo limpio y portátil.

El esquema de la base de datos se generó automáticamente a partir de las clases Entity, aprovechando las capacidades de Hibernate para la creación del modelo relacional.

Para las pruebas en local, la base de datos se inicializó con datos de ejemplo usando una opción de configuración de Spring, que nos permite ejecutar cada vez lanzamos el proyecto un conjunto de sentencias SQL almacenadas en un fichero, las cuales contienen las inserciones necesarias para llenar las tablas principales.

En el entorno de producción, desplegado en un servidor VPS de IONOS , se utilizó MariaDB, por su alto rendimiento y su compatibilidad con el ecosistema Linux de IONOS.

El proyecto se gestionó utilizando Gradle como gestor de dependencias, lo que nos garantiza una configuración modular.

6.1.3. Tecnologías de frontent web

La interfaz web de administración fue desarrollada utilizando HTML5, CSS3 y Bootstrap, lo que permite construir una interfaz responsive, clara, moderna y adaptable tanto a dispositivos de escritorio como a pantallas más reducidas.

Bootstrap proporciona componentes reutilizables y una estructura de diseño coherente, reduciendo enormemente el tiempo de desarrollo de las vistas.

El uso de Thymeleaf, integrado con Spring Boot, nos permite desarrollar plantillas con los datos obtenidos de la BBDD sin necesidad de utilizar herramientas como JS.

6.1.4. Tecnologías de desarrollo móvil

Ambas aplicaciones móviles fueron desarrolladas en Java utilizando Android Studio y el sistema de compilación Gradle, con una arquitectura modular basada en paquetes y tipos de elementos.

Las principales tecnologías y librerías utilizadas son:

- **Firebase Firestore:** se utiliza como base de datos NoSQL en la nube para almacenamiento de pedidos, comandas y líneas de comanda. La misma tecnología nos provee herramientas para la actualización en tiempo real de la información mostrada en las aplicaciones.
- **Firebase Authentication:** proporciona el sistema de autenticación para los administradores.
- **Retrofit:** permite la comunicación entre las aplicaciones móviles y el servicio REST del backend, facilitando el consume de categorías o platos.
- **Picasso:** se utiliza para la carga y renderizado de imágenes recibidas en el formato Base64 desde el servicio REST (endpoint específico para imágenes).
- **RecyclerView:** Empleado para mostrar listas dinámicas de categorías, platos y pedidos.
- **Material Design Components y ViewBinding:** mejoraron la estructura visual.

Ambas aplicaciones comparten un conjunto de librerías comunes y una estructura basada en Fragments para la navegación, facilitando el mantenimiento y la reutilización de componentes.

6.1.5. Infraestructura y despliegue

El servidor de producción se desplegó en un VPS Linux S de IONOS, configurado manualmente mediante conexión SSH.

El despliegue se realizó ejecutando directamente el archivo JAR de la aplicación Spring Boot, acompañado de la configuración de la BBDD y variables de entorno necesarias.

La elección de un despliegue manual permitió un mayor control sobre los procesos y la gestión de recursos del servidor, asegurando un entorno estable para el funcionamiento de los servicios REST y la BBDD.

Para el entorno local, Docker ha facilitado la replicación del entorno de producción, reduciendo la posibilidad de errores derivados.

6.1.6. Herramientas de control y colaboración

Además del control de versiones mediante GitHub, se utilizaron diversas herramientas de desarrollo, entre ellas:

- **Gradle Build:** para la gestión de dependencias y automatización de compilaciones.
- **Firebase Console:** para la administración de la BBDD y autenticación de usuarios.
- **Postman:** como la herramienta de pruebas para verificar el correcto funcionamiento de los endpoints REST del backend.
- **Android Emulator y dispositivos físicos:** para las pruebas de las aplicaciones móviles.

6.2. Desarrollo de la aplicación web

La aplicación web desarrollada tiene un doble propósito: servir como interfaz de gestión de las categorías y platos del restaurante y ofrecer un servicio REST que permite la comunicación con las aplicaciones móviles cliente. Para su desarrollo he usado IntelliJ IDEA como IDE principal y he estructurado el proyecto siguiendo una arquitectura MVC, con una separación en capas, permitiendo separar la lógica de negocio, el acceso a datos, la presentación y los servicios web.

6.2.1. Tecnologías empleadas

El proyecto está implementado con Spring Boot, utilizando un servidor Tomcat embebido que el propio framework nos ofrece. Spring Boot me ha permitido simplificar la configuración del proyecto y facilitar a puesta en marcha de este.

Como base de datos he utilizado PostgreSQL para las pruebas en local, con la ayuda un contenedor Docker, mientras que para el despliegue en el servidor VPS se ha utilizado MariaDB. La conexión a la base de datos se gestiona desde el fichero application.properties, donde defino los parámetros

de conexión y las propiedades de Hibernate para la creación automática del esquema de tablas en base a nuestras entidades.

El proyecto utiliza Hibernate como capa de persistencia, lo que me permite mapear las entidades Java a tablas de la BBDD mediante JPA. Gracias a la propiedad *spring.jpa.hibernate.ddl-auto=create-drop*, las tablas se generan automáticamente al arrancar la aplicación a partir de las entidades y se ejecutan los insert almacenados en import.sql.

Para el control de versiones he utilizado Git, almacenando en Github. Para el control de dependencias he utilizado Gradle por su facilidad de uso y configuración.

En la parte de presentación he utilizado Thymeleaf como motor de plantillas junto con el framework Bootstrap, lo que me ha permitido generar páginas dinámicas con un diseño responsive.

Además, el proyecto admite subida de imágenes asociadas a los productos. Estas imágenes se almacenan físicamente en el directorio /uploads, mientras que en la BBDD se guarda únicamente el nombre del archivo con un prefijo aleatorio generado al momento para que no coincidan los nombres jamás.

Es importante matizar que el servidor no almacena todas las fotografías sin más, antes de cambiarle la fotografía a un producto, borra de la carpeta uploads la fotografía anterior, para así asegurarnos de que el servidor no se llena de fotografías innecesarias. En caso de crear un producto nuevo o que el producto a modificar no tuviese fotografía en ese momento no realizamos ninguna acción de borrado.

6.2.2. Estructura del proyecto

La estructura de este proyecto sigue una organización modular, típica de las aplicaciones de este estilo, con los siguientes paquetes principales:

- **controllers:** contiene los controladores web tradicionales que gestionan las peticiones HTTP y renderizan las vistas mediante Thymeleaf.
- **restcontrollers:** define los controladores REST que contiene los endpoints que utilizan las demás aplicaciones, que nos permiten consultar la carta.

- **models.dao:** incluye las interfaces de acceso a datos (ICategoriaDao, IPlatoDao), que extienden de CrudRepository y permiten la interacción con la BBDD.
- **models.entity:** contiene las entidades Categoria y Plato, que representan las tablas del sistema y definen sus relaciones y restricciones mediante anotaciones JPA.
- **models.service:** incluye la lógica de negocio, incluyendo las interfaces y sus implementaciones para la gestión de categorías, platos y fotografías.
- **util:** agrupa clases auxiliares como Paginator o Log, que encapsula el sistema de registro mediante la clase Logger de Java.

A nivel de vistas, dentro de la carpeta resources/templates, se encuentran los distintos archivos HTML (listarPlatos.html, listarCategorias.html, formPlatos.html, …) junto al layout principal que define la estructura común de las páginas como el footer o el encabezado.

6.2.3. Seguridad y gestión de usuarios

Para el sistema de seguridad se ha implementado mediante Spring Security, configurado en la clase llamada SpringSecurityConfig.

El proyecto define dos roles principales:

- ROLE_ADMIN: con permisos para crear, modificar y eliminar platos o categorías.
- ROLE_USER: con permisos de visualización.

Además, se contempla el acceso anónimo a determinadas rutas públicas, como la visualización de listados.

La configuración incluye autenticación basada en BBDD, donde los usuarios y sus roles se almacenan en las tablas users y authorities. El cifrado de contraseñas se utiliza mediante el algoritmo BCrypt.

6.2.4. Servicio REST

El proyecto expone una API REST propia, accesible bajo las rutas /categoría y /plato. Cada controlador REST ofrece operaciones CRUD mediante los verbos estándar:

- GET: para obtener todos los registros o filtrado por ID o por categoría en caso de platos

- POST: para insertar nuevos elementos
- PUT: para actualizar registros existentes.
- DELETE: para eliminar elementos.

Por ejemplo, el endpoint /categoria/all devuelve todas las categorías existentes, mientras que categoría/{id} devuelve la categoría con dicho ID. Estos endpoints son consultados desde las aplicaciones móviles, utilizando comunicación HTTP.

6.2.5. Despliegue y configuración del entorno

Durante el desarrollo, la aplicación se ejecutó localmente en un contenedor Docker con PostgreSQL. En el entorno de producción, el despliegue se realizó manualmente sobre un servidor VPS, empleando MariaDB como sistema gestor.

Al tratarse de una aplicación Spring Boot, el proceso de despliegue es bastante sencillo, basta con compilar le proyecto y ejecutar el archivo jar generado.

El resultado es una aplicación web completa, modular y escalable, que integra gestión de contenidos, autenticación de usuarios y una API REST para las aplicaciones móviles o cualquier otra que quiera consumir dicha información.

6.3. Desarrollo de la aplicación cliente

La aplicación móvil cliente se ha desarrollado con el objetivo de proporcionar una solución factible tanto al cliente como al hostelero. Una herramienta sencilla e intuitiva para consultar la carta del restaurante, seleccionar los platos deseados y enviar pedidos al sistema central.

Su implementación se realizó en Android Studio, utilizando Java como lenguaje de programación y Gradle como gestor de dependencias. La arquitectura se basa en una estructura modular que favorece la mantenibilidad y la separación de responsabilidades.

6.3.1. Arquitectura general

La aplicación se ha organizado en paquetes para así facilitar la escalabilidad y la claridad del código. Entre ellos se encuentran los siguientes.

- **adapters:** contienen las clases utilizadas para enlazar los datos con las vistas RecyclerView, tanto para categorías, como platos y el carrito.
- **fragments:** agrupa los fragmentos principales de la interfaz, como el listado de categorías, el listado de platos y el carrito de compra.
- **dialogs:** incluye los cuadros de diálogo que facilitan la interacción con el usuario, como dialogNumMesa y DialogLogin.
- **models:** incluye las clases de modelo como Categoria, Plato, Pedido, Comanda o LineaComanda.
- **rest e interfaces:** implementan la comunicación con el servicio REST del backend, utilizando Retrofit para la serialización y deserialización de objetos.
- **managers:** contiene clases auxiliares para la gestión de instancias compartidas para el carrito de compra, implementado mediante el patrón Singleton.
- **res:** incluye los recursos gráficos, layouts y valores XML (colores, estilos, menús, etc.) usados en la interfaz.

El flujo principal de la aplicación comienza mostrando el listado de categorías, obtenidas desde el servicio REST. Una vez seleccionada una categoría, se muestran los platos asociados y el usuario puede acceder al detalle de cada uno para consultar toda su información. Desde esa vista es posible añadir unidades del plato al carrito, junto con un comentario opcional.

Podremos acceder al carrito en todo momento desde la barra de herramientas a la derecha. Incorpora un contador dinámico que indica el número total de artículos añadidos. Finalmente, a confirmar el pedido, los datos se envían a Firebase Firestore, generando una colección de pedidos con sus respectivas comandas y líneas de comanda, además de vaciar el carrito.

6.3.2. Comunicación con el backend

La comunicación entre la aplicación descrita y el backend desarrollado en Spring se realiza mediante la librería Retrofit, que nos facilita la interacción con servicios RESTful.

Las peticiones GET se utilizan para obtener la información de la carta, que se carga dinámicamente al iniciar la aplicación o al navegar por ella.

La respuesta del backend se recibe en formato JSON y se mapea automáticamente a las clases modelo definidas. Esta integración asegura una sincronización en tiempo real de la carta, garantizando la integridad de la información, aunque el administrador modifique la carta.

6.3.3. Gestión del carrito y patrón Singletone

El carrito de compra es uno de los elementos más importantes de la aplicación. Para evitar inconsistencias y duplicidades, se ha implementado la clase CarroManager guardada en el paquete cf.victorlopez.pedidosrestaurante.managers.

Esta clase implementa el patrón de diseño Singleton, asegurando que siempre exista una sola instancia del carrito durante toda la ejecución de la aplicación.

El patrón permite mantener el estado del carrito incluso aunque naveguemos entre fragmentos, garantizando que el estado del carrito siempre es el mismo y no se pierden sus productos. Además, se ha añadido un método rebuild para reconstruir la instancia en caso de que se reinicie la información o esté corrupta.

El icono del carrito de la Toolbar incorpora un contador dinámico, que se actualiza en tiempo real cada vez que añadimos o eliminamos platos del carrito. Esto mejora la experiencia de usuario y proporciona feedback inmediato sobre el estado del pedido.

6.3.4. Persistencia de pedidos en Firebase Firestore

Una vez confirmamos el pedido, la aplicación sube la información a la BBDD de Firebase, donde cada pedido es un documento diferente. Dentro del mismo pedido existe un listado de comandas y cada comanda una lista de líneas de ocmanda.

Este modelo permite una gestión más flexible y escalable que la implementación inicial, en la cual los pedidos se representaban simplemente como listas de platos y un total.

Durante el desarrollo se detectó que el primer enfoque dificultaba el control de entregas y un posible segundo pedido de la misma mesa para pedir bebida o postres.

Gracias a esta modificación, diferenciamos cada comanda por hora de pedido y permitimos un control más óptimo.

Por último, me gustaría mencionar la forma de finalizar un pedido. En caso de que exista un pedido con ese número de mesa con la flag de pagado a falso, se añade la comanda a dicho pedido. En caso de que no exista ningún pedido con ese número de mesa o exista y está ya pagado, creamos un pedido nuevo.

6.3.5. Renderizado de imágenes con Picasso

Para optimizar la carga de imágenes en las vistas utilizamos la librería Picasso.

Inicialmente, las imágenes en formato Base64 se renderizaban manualmente, lo que provocaba retardos y un renderizado en ocasiones erróneo.

La incorporación de esta librería me permitió gestionar automáticamente la decodificación, el almacenamiento en caché y la actualización de imágenes, mejorando el rendimiento y la experiencia del usuario.

6.3.6. Diseño visual y experiencia de usuario

El diseño de la aplicación se ha basado en Material Design, utilizando como tema principal Theme.MaterialComponents.DayNight.NoActionBar. Se definió una paleta de color personalizada en el fichero colors.xml, en la que definimos los colors principales de la app, entre ellos el color rojo oscuro (#8c1713), que actúa como color primario. Además, se han creado estilos para los botones, adaptando colores de fondo y texto a tonos corporativos.

Esta coherencia visual con el uso de una tipografía a la par, nos ofrece una interfaz limpia, clara y fácil de utilizar.

6.4. Desarrollo de la aplicación de gestión de pedidos

La aplicación móvil de gestión de pedidos está destinada al personal del restaurante, se ha desarrollado con el objetivo de proporcionar una herramienta práctica y eficiente para controlar los pedidos de los clientes. Al igual que la otra aplicación, se ha implementado en Android Studio usando Java y Gradle como gestor de dependencias., siguiendo una arquitectura modular para facilitar la mantenibilidad y escalabilidad.

6.4.1. Diseño visual y experiencia de usuario

La estructura de la aplicación es muy similar a la de la aplicación cliente, organizada en paquetes que agrupan funcionalidades y responsabilidades:

- **adapters:** clases que vinculan los datos de los pedidos con los componentes RecyclerView.
- **Fragments:** contienen los fragmentos principales de la interfaz, como el listado de pedidos, el histórico y el detalle de cada pedido.
- **dialogs:** incluyen cuadros de diálogo para interacción gestionada por la aplicación.
- **models:** clases de modelo que representan la información gestionada, como Pedido, Comanda o LineaComanda.
- **rest e interfaces:** implementan la comunicación con Firebase Firestore para la gestión de pedidos y permiten recibir actualizaciones automáticas mediante listeners.
- **res:** recursos gráficos, layouts y valores XML utilizados en la interfaz.

6.4.2. Flujo principal

Al iniciar la aplicación, el usuario se encuentra con una pantalla de login que utiliza el mismo sistema de autenticación que la aplicación cliente. Una vez autenticado, se muestran los pedidos actuales. Podemos acceder al histórico de pedidos a través de una opción del menú de hamburguesa.

Cada pedido incluye su detalle completo, mostrando las comandas y líneas de comandas asociadas. La aplicación gestiona automáticamente el estado de los pedidos: un pedido se considera entregado únicamente cuando todos sus comandas se encuentran en dicho estado. Además, cuando un pedido se marca como pagado, se traslada automáticamente al histórico.

Aunque la aplicación actualmente no renderiza imágenes de los platos desde el backend, se ha dejado preparada la implementación para poder mostrar las fotos o hacer consultas al servicio RESTful en un futuro.

6.4.3. Actualización en tiempo real

Para garantizar que la información de los pedidos se mantiene actualizada en todos los dispositivos, la aplicación utiliza listeners de Firebase Firestore. Esto nos permite que cualquier cambio en los

pedidos, como nuevas comandas o modificaciones de estado se reflejen automáticamente en la interfaz sin necesidad de recargar manualmente la aplicación.

6.5. Integración entre las tres aplicaciones

La solución diseñada consta de tres aplicaciones interconectada: la aplicación web para la gestión de la carta, la aplicación de clientes para realizar pedidos y la aplicación para gestionarlos. Se integran mediante el uso de BBDD y servicios de comunicación. La web se conecta a la BBDD MySQL para la gestión centralizada de categorías y platos, mientras que las aplicaciones móviles utilizan Firebase Firestore para registrar los pedidos en tiempo real.

Gracias a esta configuración, los pedidos realizados desde la aplicación cliente se reflejan automáticamente en la aplicación de gestión, y su estado se sincroniza correctamente: cuando todas las comandas de un pedido están entregadas, el pedido se marca como entregado y al marcarlo como pagado, se traslada al histórico. De esta forma, la integración asegura coherencia en la información.

6.6. Problemas encontrados y soluciones adoptadas

6.6.1. Aplicación web

Eliminación de imágenes antiguas al modificar un plato

Al actualizar la imagen de un plato, la imagen anterior seguía almacenada en el servidor, cosa que al final derivaría en una saturación del servidor.

Solución: Se desarrolló un mecanismo que borra automáticamente la imagen antigua al reemplazarla con una nueva, evitando dicha acumulación innecesaria.

Restauración automática de la BBDD al arrancar la aplicación

Durante el desarrollo y las pruebas, restaurar la base de datos de forma manual es tedioso y propenso a errores.

Solución: Se creó un fichero insert.sql que, junto a una configuración del fichero de properties, crea todas las tablas y hace las inserciones cada vez que arrancamos la aplicación, facilitando así

las pruebas. También existe una opción de configuración para que solo lo cree una vez, en caso de no estar creada.

6.6.2. Aplicación pedidos cliente

Duplicación de pedidos al repetir un pedido anterior

Durante las pruebas se detectó que al hacer un pedido cuando el anterior aún no había sido pagado se creaba otro diferente. Además, también podía estar el problema de añadirlo y que se dupliquen líneas.

Solución: Para solucionar este problema, se refactorizó la clase pedido para que funcionase como una lista de comandas, cada línea de comanda tiene un plato y la cantidad de este. Esto nos permite gestionar correctamente pedidos completos y evitar duplicidad.

Gestión del carrito de la compra

Al iniciar la aplicación, podía crearse más de una instancia del carrito al navegar entre diferentes pantallas, provocando inconsistencias en los platos.

Solución: Se implementó el patrón Singleton mediante la clase CarroManager, garantizando una única instancia compartida del carrito en toda la aplicación. Además, se incorporó un contador dinámico en el icono del carrito que representa el total de platos.

Renderizado de imágenes de los platos

El backend entregaba imágenes en formato base64, y el método inicial resultaba ineficiente y ocurrían errores.

Solución: Se añadió la librería Picasso, que permite cargar y mostrar imágenes de manera asíncrona, optimizando el rendimiento y evitando bloqueos.

Gestión del número de mesa

Guardar y actualizar el número de mesa podía generar inconsistencias si el usuario cerraba sesión o cambiaba de dispositivo.

Solución: Se almacenó el número de mesa en SharedPreferences, garantizando persistencia entre sesiones, el cual se puede modificar iniciando sesión.

6.6.3. Aplicación gestión de pedidos

Sincronización de pedidos en tiempo real

Los cambios realizados desde la aplicación cliente no se reflejaban automáticamente, causando inconsistencias en la visualización de los pedidos.

Solución: Se implementaron listeners de Firebase, que detectan modificaciones en los documentos de pedidos y actualizan la interfaz en tiempo real.

CAPÍTULO 7: VALIDACIÓN DEL SISTEMA

7.1. Estrategia de pruebas manuales

Para las pruebas realizadas durante el desarrollo del sistema me he basado en una estrategia de validación manual. Me he centrado en comprobar la integración y funcionamiento en conjunto de las distintas aplicaciones del proyecto. Dado que el sistema está compuesto por una aplicación web (para la gestión de la carta), una aplicación móvil cliente (para realizar pedidos) y otra para la gestión de estos en tiempo real. Se ha considerado que las pruebas más adecuadas deben reproducir los flujos reales de uso que se producían en un entorno de trabajo norma.

Las pruebas manuales me han permitido verificar el comportamiento del sistema de forma global, comprobando no solamente el comportamiento de cada aplicación por separado, sino también la comunicación entre ellas. En especial lo relativo al almacenamiento y actualización de datos en Firebase y la gestión de archivos.

Se ha elegido no implementar test unitarios o de interfaz, ya que el objetivo principal es garantizar la estabilidad del sistema completo, más que validar los componentes de forma aislada. El testing automatizado se consideraría una tarea complementaria a la usada, útil en proyecto de mayor escala o en futuras implementaciones del sistema.

Durante esta fase de pruebas, se ha seguido una metodología de comprobación iterativa, con la que después de añadir nuevas funcionalidades o hacer una corrección, se procedía a hacer un conjunto de pruebas relacionadas con ese sistema. Además, con el sistema ya montado, se realizaron pruebas completas de principio a fin, simulando el flujo de trabajo de todos los actores del sistema.

Entre ellas incluyen la validación de operaciones CRUD, la verificación de restricciones de acceso por roles, la gestión de sesiones y autenticación, la actualización de imágenes del servidor y la sincronización de pedidos en tiempo real.

7.2. Escenarios de prueba

A continuación, se describen los escenarios de prueba manual seguidos para validar el funcionamiento y la integración de las tres aplicaciones del sistema. Las pruebas se centran en reproducir el flujo real de la aplicación, comprobando tanto las funcionalidades principales como la comunicación con la BBDD y los servicios.

7.2.1. Aplicación web

El objetivo de las pruebas de esta aplicación fue garantizar el correcto funcionamiento del sistema de gestión de platos y categorías, el manejo de imágenes en la carpeta uploads y la seguridad según los roles definidos.

Escenario 1: Autenticación

- Se intenta acceder a rutas protegidas sin iniciar sesión, comprobando que el sistema deniega el acceso.
- Se realizan intentos fallidos de inicio de sesión para verificar la validación.
- Finalmente, se inició sesión con credenciales válidas, tanto de usuario administrador como de usuario común, confirmando sus diferencias según su rol.

Escenario 2: Gestión de categorías

- Se navega por las categorías y a sus detalles para comprobar la visualización.
- Se crea una nueva categoría con todos los campos válidos.
- Posteriormente, se edita la categoría introduciendo valores erróneos para comprobar la validación del formulario.
- Al verificar los distintos mensajes de error, corregimos los datos modificando todo lo posible, verificando que los cambios se guardan correctamente y que la fotografía antigua se elimina de la carpeta uploads (para evitar acumulación de archivos "huérfanos").
- Para finalizar, eliminamos la categoría, comprobando que tanto la tupla en la BBDD como la foto de uploads han desaparecido correctamente.

Escenario 3: Gestión de platos

- Se crea un nuevo plato asignado a una categoría existente.
- Se accede a su detalle y se valida que toda la información se muestra bien.
- Se realizan modificaciones erróneas para comprobar la validación.
- Posteriormente se hacen modificaciones correctas y se guarda correctamente.
- Se comprueba que la imagen antigua se ha eliminado de uploads.

- Finalmente, se elimina el plato y se verifica que tanto la tupla como la imagen se han eliminado.

Escenario 4: Cierre de sesión y control de acceso

- Se cierra la sesión y se comprueba que el usuario es redirigido correctamente a la página principal y pierde acceso a las funciones administrativas.

Escenario 5: Verificación del servicio RESTful

El objetivo fue comprobar el correcto funcionamiento y formato del servicio REST implementado.

El servicio actúa como capa de comunicación entre la BBDD MySQL y las aplicaciones móviles.

Para ello, se realizan pruebas manuales utilizando la app Postman, interactuando con el servidor Tomcat integrado (alojado actualmente en el puerto 8090), verificando las respuestas a las operaciones CRUD pertinentes.

7.2.2. Aplicación pedidos (cliente)

Las pruebas en esta aplicación se enfocan principalmente en el flujo de pedidos, la gestión del carrito y el correcto almacenamiento en Firebase.

Escenario 1: Acceso y autenticación

- Al iniciar la aplicación, se comprueba la carga correcta de categorías en la pantalla principal.
- Se intenta acceder a las opciones administrativas sin iniciar sesión, no existen en el menú.
- Tras iniciar sesión, se habilitan funciones del menú administrativas.
- Se modifica el número de mesa y se valida que el valor se almacena correctamente en las SharedPreferences.
- Se cierra sesión y se comprueba que las funciones administrativas han desaparecido.

Escenario 2: Gestión del carrito

- Se accede al carrito vacío, comprobando que aparece la pantalla informativa de que está vacío.
- Se intenta realizar un pedido con el carrito vacío, validamos que el sistema lo impide y avisa al usuario.
- Se añadieron platos al carrito desde diferentes categorías, con cantidades y comentarios diferentes.
- Posteriormente, se eliminan varios productos del carro para comprobar que el total se recalcula correctamente.
- Para finalizar, se comprueba que el pedido se ha registrado correctamente en Firebase.

7.2.3. Aplicación gestión de pedidos (interna)

Escenario 1: Inicio de sesión y visualización de pedidos

- Se inicia sesión con credenciales incorrectas para comprobar la validación de la autenticación
- Después de iniciar sesión correctamente, se verifica la carga de pedidos actuales (no pagados).
- Se accede al detalle del pedido y se comprueba la correcta visualización de la información y de las comandas.

Escenario 2: Actualización de estados

- Se marcan las comandas de un pedido como entregadas y se comprueban que el estado del pedido cambia a “Entregado” en el listado.
- Desde otra aplicación cliente, se realiza un nuevo pedido verificando que aparece automáticamente en el listado ordenado por fecha.
- Se añade una nueva comanda al primer pedido y se comprueba que el pedido vuelve a aparecer el primero de la lista y como “No entregado”.

Escenario 3: Gestión del histórico

- Se marca un pedido como pagado desde el listado principal, comprobando que desaparece de este y pasa al histórico.

- Se repite la acción desde el detalle del pedido, verificando el mismo resultado.
- Se accede al histórico y se modifica un pedido pagado a “No pagado”, comprobando que vuelve al anterior listado.
- Finalmente, se hace un nuevo pedido con el mismo número de mesa que uno previamente cerrado, confirmando que se crea un pedido nuevo.

7.3. Resultados de la validación

Las pruebas manuales sobre las tres aplicaciones han confirmado el correcto funcionamiento del sistema y de todos sus componentes.

Aplicación web

Durante las pruebas se ha comprobado que la gestión de categorías y platos funciona perfectamente, incluyendo la validación de datos, la actualización de imágenes y la eliminación de imágenes antiguas de la carpeta uploads. Se detectaron algunos errores menores en la validación de formularios y en la actualización de las vistas tras ciertas operaciones, los cuales fueron corregidos.

Aplicación cliente de pedidos

Las pruebas han demostrado que la navegación, autenticación y la gestión del carrito funcionan a la perfección. Se detectaron pequeños fallos en la persistencia del número de mesa y en la actualización del contador de artículos al borrar, ambos resueltos.

Aplicación de gestión de pedidos

Las pruebas de integración con Firebase confirmaron la correcta sincronización en tiempo real de los pedidos y sus respectivas comandas. Al principio, hubo problemas gestionando el estado y el refresco de la interfaz, se solucionaron con la implementación de listeners.

CAPÍTULO 8: CONCLUSIONES Y TRABAJO FUTURO

8.1. Conclusiones principales

El desarrollo de este sistema ha cumplido con todos los objetivos planteados al inicio, el cual nos ofrece una solución compuesta por una aplicación web y dos aplicaciones móviles que trabajan conjuntamente sobre un mismo conjunto de datos.

La integración de estas tres aplicaciones ha demostrado ser funcional, permitiendo una comunicación fluida entre el cliente, el personal del restaurante junto a los administradores.

El resultado obtenido responde a la necesidad de optimizar el proceso de realización y gestión de los pedidos, reduciendo el contacto directo entre el cliente y el hostelero, agilizando y automatizando la gestión.

En términos generales, se puede decir que el proyecto cumple plenamente los objetivos planteados, ofreciendo un sistema estable, funcional y adaptable a diferentes tipos de negocios.

8.2. Aportaciones del proyecto

El sistema desarrollado nos aporta una visión unificada del proceso de gestión de pedidos en el restaurante, combinando tecnologías Android, web y servicios en la nube.

Entre sus principales funcionalidades destacan:

- Automatización del flujo de pedidos, desde la modificación de la carta hasta la selección de platos por parte del cliente hasta la gestión interna por parte del personal.
- Sincronización en tiempo real entre las aplicaciones mediante Firebase, garantizando la actualización inmediata de la actualización de los pedidos.
- Reducción del tiempo de atención y de los errores humanos, minimizando la intervención de personal de atención al cliente en la gestión de pedidos.
- Escalabilidad del sistema, que permite gestionar múltiples mesas y usuarios de manera concurrente sin pérdida de rendimiento.

El proyecto no solamente facilita esta gestión, sino que sienta bases de automatización para diferentes procesos, como análisis de carta, control de inventario, etc.

8.3. Limitaciones del sistema

Aunque hemos obtenido muy buenos resultados, hemos observado que aún existen ciertas limitaciones, entre ellas:

- No se ha implementado testing automatizado, por lo que las pruebas se deben realizar automáticamente. Aunque, en muchos proyectos profesionales, se designan personas especializadas en pruebas integradas que, a parte de los test unitarios, también hacen este tipo de pruebas.
- No es posible editar pedidos o comandas una vez han sido generadas, para evitar inconsistencias en los registros.
- El sistema no implementa control de roles en las aplicaciones de gestión de pedidos y del cliente, ya que de momento no separamos responsabilidades. Cuando se implementen diferentes funcionalidades como estadísticas sí se deberían implementar para que no todos los trabajadores puedan verlas.
- La gestión de imágenes y archivos aún podría optimizarse para mejorar el rendimiento, sobre todo al consultarlas.
- No se ha implementado un sistema de notificaciones push para los nuevos pedidos que llegan.
- El diseño actual del servicio RESTful no contempla mecanismos de autenticación robustos ni cifrado de datos en tránsito, aunque se ha planificado su futuro desarrollo.

Estas limitaciones no afectan al rendimiento general del sistema, pero son mejoras para tener en cuenta en futuras versiones.

8.4. Posibles mejoras y líneas de trabajo futuro

8.4.1. Aplicación web

Exportación de la carta en formato PDF

Una futura ampliación es permitir la generación automática de un documento PDF con la carta del restaurante, manteniendo imágenes y estructura, para que tanto el cliente como el administrador la puedan consultar o descargar.

Mejora de la seguridad del servicio RESTful

Implementar autenticación mediante tokens JWT y cifrado para protegerlas comunicaciones y garantizar acceso seguro a los servicios.

Sistema de valoraciones y opiniones

Incorporar un módulo que permita a los clientes valorar platos y dejar comentarios, añadiendo retroalimentación al sistema para futuras visitas y para los propios hosteleros.

Sistema de reservas en restaurante

Añadir la posibilidad de reservar mesas desde la aplicación cliente, sincronizadas con la BBDD y visibles desde el panel de administración.

Soporte multiidioma

Implementar un sistema de traducción en varios idiomas básicos como inglés, español, alemán, etc.

Panel de administración del usuario

Crear una opción del menú de administración para acceder a un panel que contenga la información del usuario y pueda modificarla.

Gestión de inventario

Desarrollar un módulo que permita llevar control de stock de ingredientes o productos, vinculado a los platos disponibles.

8.4.2. Aplicación pedidos (cliente)

Asistente IA de recomendación

Sería muy interesante añadir un modelo de IA que, según toda la información de la carta y el prompt que haga el usuario te recomiende platos. Además, también puede basarse en estadísticas de platos más pedidos, mejor valorados, etc.

Implementación de códigos QR por mesa

Un posible cambio que, según el tipo de negocio, puede ser interesante es permitir el acceso directo a la aplicación de pedidos mediante escaneo de QR en cada mesa, facilitando el uso por parte de los comensales y reduciendo el costo en hardware para el hostelero.

Botón de asistencia al camarero

Añadir una opción para solicitar atención desde la aplicación cliente, notificando al personal del restaurante en tiempo real.

8.4.3. Aplicación gestión de pedidos (interna)

Edición avanzada de comandas

Es muy importante añadir la posibilidad de modificar platos dentro de una comanda antes de su cierre o entrega, manteniendo un log o similar de dichos cambios.

Módulo de estadísticas e informes

Es interesante incorporar un panel analítico con estadísticas como los platos más vendidos, volumen de ingresos (diarios, semanales, mensuales, trimestrales, etc.), horas con más pedidos, etc.

Gestión de facturación anual

Implementar un sistema de facturación automatizada con exportación a formatos estándar (PDF o CSV) y cálculo de IVA, útil para contabilidad y Hacienda.

CAPÍTULO 9: BIBLIOGRAFÍA

[1] Secretaría de Estado de Turismo, <<Medidas para la reducción del contagio por el coronavirus SARS-CoV-2 en servicios de restauración>>, 2020 (Consultado: 14 de marzo 2025)

https://www.mintur.gob.es/ca-es/gabineteprensa/notasprensa/2020/paginas/turismo-elabora-una-gu%C3%ADa-de-buenas-pr%C3%A1cticas-para-el-sector-tur%C3%ADstico-y-sus-trabajadores-frente-al-covid-19.aspx

[2] Alicia Maeso <<¿Cómo afectó la pandemia de la COVID-19 en la hostelería en las distintas C.C.A.A?>>, NJN Jurídico (Consultado: 27 de marzo de 2025)

https://nbnjuridico.es/la-covid-19-en-la-hosteleria-en-las-distintas-ccaa/

[3] El País, <<Soluciones digitales para impulsar la actividad de bares y restaurantes>>, El País, 28 de marzo de 2025. (Consultado: 27 de marzo de 2025)

https://elpais.com/tecnologia/branded/especial-innovacion/2025-03-28/soluciones-digitales-para-impulsar-la-actividad-de-bares-y-restaurantes.html

[4] S. F. T. X. Y. Z., «Restaurantes 2.0: la digitalización como salvavidas en la era pospandemia», *Expansión Empleo*, 11 de abril de 2022. (Consultado: 11 de abril de 2025)

https://www.expansion.com/expansion-empleo/emprendedores/2022/04/11/62541924e5fdeac5798b45e2.html

[5] HRS Hospitality & Retail Systems, <<Transformación digital y eficiencia de los restaurantes>>, HRS Hospitality & Retail Systems News and Insights, 24 de noviembre de 2023. (Consultado: 12 de abril de 2025)

https://hrsinternational.com/es/news-and-insights/transformaci%C3%B3n-digital-y-eficiencia-de-restaurantes

[6] J. Nielsen, <<Usability 101: Introduction to Usability>>, Nielsen Normal Group, 3 de enero de 2012. (Consultado: 2 de mayo de 2025)

https://www.nngroup.com/articles/usability-101-introduction-to-usability/

[7] CoverManager <<Usability 101: Introduction to Usability>>, CoverManager Blog, 11 de mayo de 2023. (Consultado: 10 de mayo de 2025)

https://www.covermanager.com/es/como-mejorar-la-eficiencia-operativa-de-un-restaurante-con-software-de-gestion/

[8] I. Sommerville, Software Engineering, 10.ª ed. Pearson, 2016 (Consultado: 30 de mayo de 2025)

https://www.google.com/url?sa=E&q=https%3A%2F%2Fwww.pearson.com%2Fus%2Fhigher-education%2Fprogram%2FSommerville-Software-Engineering-10th-Edition%2FPGM319321.html

[9] M. Fowler, <<Patterns of Enterprise Application Architecture>>, Martin Fowler, 2002 (Consultado: 10 de Junio de 2025)

https://martinfowler.com/books/eaa.html

[10] Google Clod, <<Cloud Firestore (Firebase) Documentation>>, 2023 (Consultado: 12 de junio de 2025)

https://www.google.com/url?sa=E&q=https%3A%2F%2Fcloud.google.com%2Ffirestore%2Fdocs

[11] Minsait, << Minsait ofrece a hoteles y restaurantes nuevas soluciones para centralizar la gestión, mejorar el servicio al cliente e incrementar su competitividad>>, Noviembre 2018. (Consultado: 15 de junio de 2025)

https://www.minsait.com/es/actualidad/media-room/minsait-ofrece-hoteles-y-restaurantes-nuevas-soluciones-para-centralizar-la

[12] N. G. S. G. y T. G., «Impacto de la tecnología de autoservicio en la eficiencia operativa y la experiencia del cliente en restaurantes de servicio rápido», Journal of Foodservice Business Research, vol. 26, no. 4, pp. 315-330, 2023. (Consultado: 15 de junio de 2025)

[13] M. R. Pérez y L. M. Santos, «La digitalización de la carta: QR y otras herramientas de comunicación en la restauración post-COVID», *Revista de Hostelería y Turismo*, vol. 8, no. 1, pp. 60-75, 2022.

[14] K. Chen y S. Li, «El papel de las plataformas de mensajería social en la estrategia de marketing y ventas de pequeños restaurantes», International Journal of Contemporary Hospitality Management, vol. 35, no. 2, pp. 785-802, 2023.

[15] R. Sánchez y S. Díaz, Tecnología y Gestión en Restauración: Del TPV tradicional a la Inteligencia Artificial, Editorial Paraninfo, 2023.

[16] L. M. Santos y M. R. Pérez, «La digitalización de la carta: QR y otras herramientas de comunicación en la restauración post-COVID», Revista de Hostelería y Turismo, vol. 8, no. 1, pp. 60-75, 2022.

[17] La transformación digital del sector hostelero, en el foco del XVI Congreso Horeca de Aecoc. (Consultado:10 de Agosto de 2025)

https://www.hosteleriadigital.es/2018/05/31/la-transformacion-digital-del-sector-hostelero-en-el-foco-del-xvi-congreso-horeca-de-aecoc/

[18] C. A. García y B. J. Soto, «Análisis de la viabilidad de aplicaciones móviles de pedido propio para pequeños restaurantes frente a plataformas de terceros», Revista Española de Marketing Sectorial, vol. 10, no. 2, pp. 45-60, 2023.

[19] Deloitte, Tecnología en la Hostelería: Tendencias clave y oportunidades de inversión en España, 2023.

[20] WhatsApp< <<Casos de éxito: Cómo pequeños negocios usan WhatsApp Business para crecer>>, WhatsApp Business (Consultado: 11 de agosto de 2025)

https://business.whatsapp.com/resources/success-stories

[21] G. S. G. y T. G. N., «Análisis comparativo de sistemas de gestión de pedidos en restauración: del TPV tradicional a la mensajería instantánea», Revista de Gestión de Restaurantes e Innovación, vol. 12, no. 3, pp. 112-128, 2024.

[22] C. A. García y B. J. Soto, «Análisis de la viabilidad de aplicaciones móviles de pedido propio para pequeños restaurantes frente a plataformas de terceros», Revista Española de Marketing Sectorial, vol. 10, no. 2, pp. 45-60, 2023.

[23] L. M. Santos y M. R. Pérez, «La digitalización de la carta: QR y otras herramientas de comunicación en la restauración post-COVID», Revista de Hostelería y Turismo, vol. 8, no. 1, pp. 60-75, 2022.

[24] G. S. G. y T. G. N., «Análisis comparativo de sistemas de gestión de pedidos en restauración: del TPV tradicional a la mensajería instantánea», Revista de Gestión de Restaurantes e Innovación, vol. 12, no. 3, pp. 112-128, 2024.

[25] C. Courage y K. Baxter, <<Understanding your users: Practical guide to user requirements, methods, tools and techniques>>, San Francisco, CA: Morgan Kaufmann Publishers, 2004.

[26] J. Rubin y D. Chisnell, <<Handbook of usability testing>>, Indianapolis, IN: Wiley Publishing, 2008.

[27] S. Krug <<No me hagas pensar: una aproximación a la usabilidad en la Web>>, Madrid: Prentice Hall, 2001.

[28] S. Krug, <<Rocket surgery made easy: The do-it-yourself guide to finding and fixing usability problems>>, Berkeley, CA: New Riders, 2010.

[29] J. Nielsen, trad. R. Vázquez Llorente, <<Usabilidad de las páginas de inicio: análisis de 50 sitios Web>>, Madrid: Pearson Educación, 2002.

[30] J. Nielsen trad. S. Fraguas, <<Usabilidad: diseño de sitios Web>>, Madrid: Prentice Hall, 2002.

[31] M. Fowler y K. Scott, UML gota a gota, Naucalpan de Juárez (México): Addison Wesley Longman de México, cop. 1999

[32] Stevens, Perdita. Pooley, Rob J. "Using UML software engineering with objects and components "(abre en nueva ventana). Harlow, England [etc.] Addison-Wesley 2006.

[33] Douglass, Bruce Powel. "Real time UML advances in the UML for real-time systems"(abre en nueva ventana). Boston [etc.] Addison-Wesley cop. 2004.

[34] C. W. Dawson y G. Martín Quetglás, <<El proyecto fin de carrera en ingeniería informática: una guía para el estudiante>>, Madrid: Prentice Hall Educación, D.L. 2002.

[35] I. Sommerville y J.A. Domínguez Torres, <<Ingeniería de software>>, México [etc.]: Addison-Wesley Iberoamericana, cop. 2002.

[36] R.S. Pressman y D. Ince, <<Ingeniería del software: un enfoque práctico>>, Madrid [etc.]: McGraw-Hill, cop. 2005.

[37] I. Sommerville y J.A Domínguez Torres, Ingeniería de Software, México [etc.]: Addison-Wesley Iberoamericana, cop. 2002.

[38] R.S. Pressman y D.Ince, Ingeniería del software: un enfoque práctico, Madrid [etc.]: Mc Graw-Hill, cop. 2005.

[39] W.S. Humphrey, Introduction to the Personal Software Process (PSP), Addison Wesley Longman, Inc., 1997.

[40] M. Fowler y K. Scott, UML gota a gota, Naucalpan de Juárez (México): Addison Wesley Longman de México, cop. 1999.

[41] M.G. Piattini Velthuis, Análisis y diseño de aplicaciones informáticas de gestión: una perspectiva de ingeniería del software, Paracuellos de Jarama (Madrid): Ra-Ma, 2003.

[42] S. McConnel, Desarrollo y gestión de proyectos informáticos, Madrid [etc.]: McGraw-Hill, D.L.2000.

[43] C.W. Dawson y G.Martín Quetglás, El proyecto fin de carrera de ingeniería informática: una guía para el estudiante, Madrid: Prentice Hall Educación, D.L 2002.

ANEXOS

Anexo I. Aplicación web (Spring Boot - Gestión de la carta y servicio RESTful)

Manual de usuario

Al acceder a la aplicación, lo primero que podemos observar es la pantalla principal con un acceso a los diferentes listados.

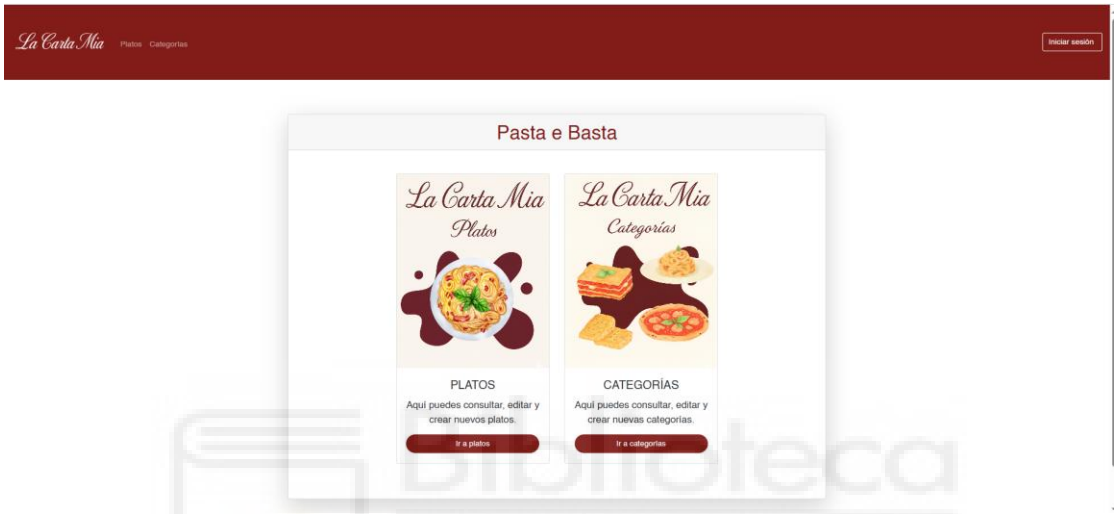

Ilustración 44. Página principal - Aplicación web

Al seleccionar cualquiera de las opciones nos mostrará un listado de los elementos correspondientes paginados. Como podemos observar, si accedemos como usuario anónimo, no tenemos acceso a crear, modificar, eliminar ni acceder al detalle de los productos.

Ilustración 45. Listado de platos (usuario anónimo) – Aplicación web

Al seleccionar la opción del menú, accedemos al panel de inicio de sesión.

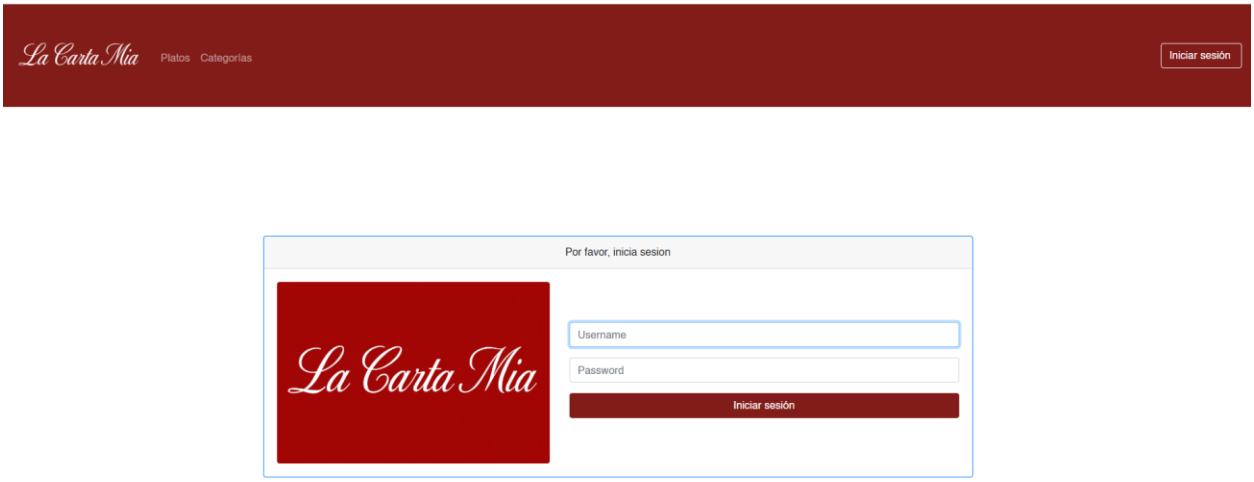

Ilustración 46. Página de inicio de sesión - Aplicación web

Si las credenciales de inicio de sesión son incorrectas, la aplicación nos avisa en la parte superior.

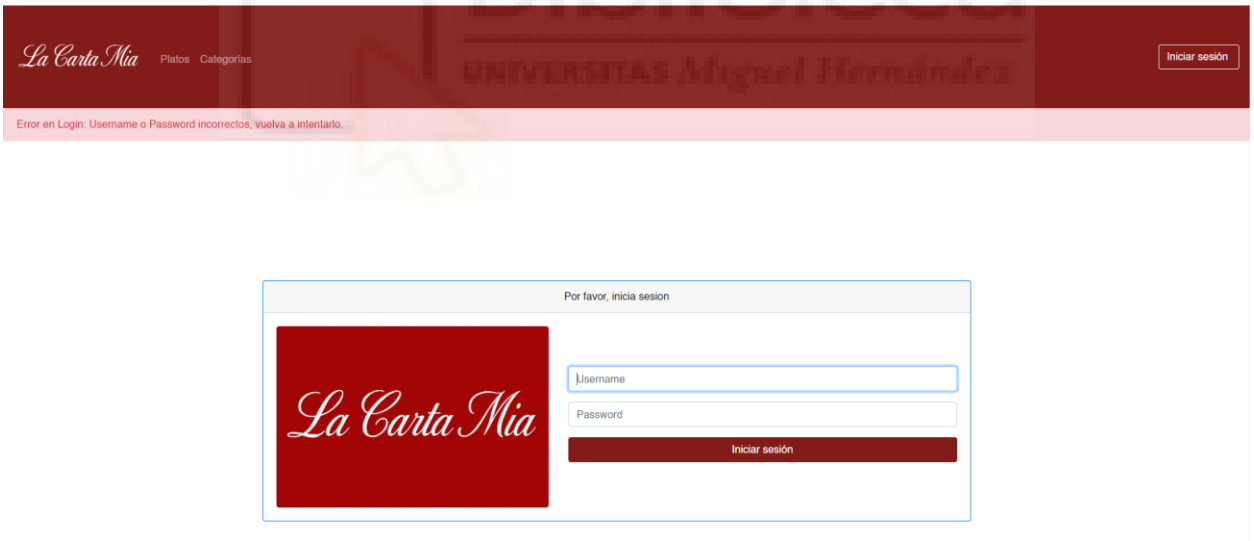

Ilustración 47. Inicio de sesión fallido - Aplicación web

Al iniciar sesión correctamente tenemos dos posibilidades, acceder mediante usuario común o usuario administrador, el cual puede desarrollar todas las tareas de la aplicación.

En ambos casos, si el inicio de sesión es correcto, nos redirige a la página de inicio y nos muestra un mensaje de inicio correcto.

Ilustración 48. Inicio de sesión correcto - Aplicación web

Al iniciar sesión como usuario común y acceder al listado, podemos ver que nos ha aparecido el campo ID, el cual si pulsamos nos deja acceder a su detalle.

Ilustración 49. Lista de platos como usuario común - Aplicación web

Al acceder al detalle podremos ver toda su información, pero no modificarla, ya que este usuario no tiene permisos para ello.

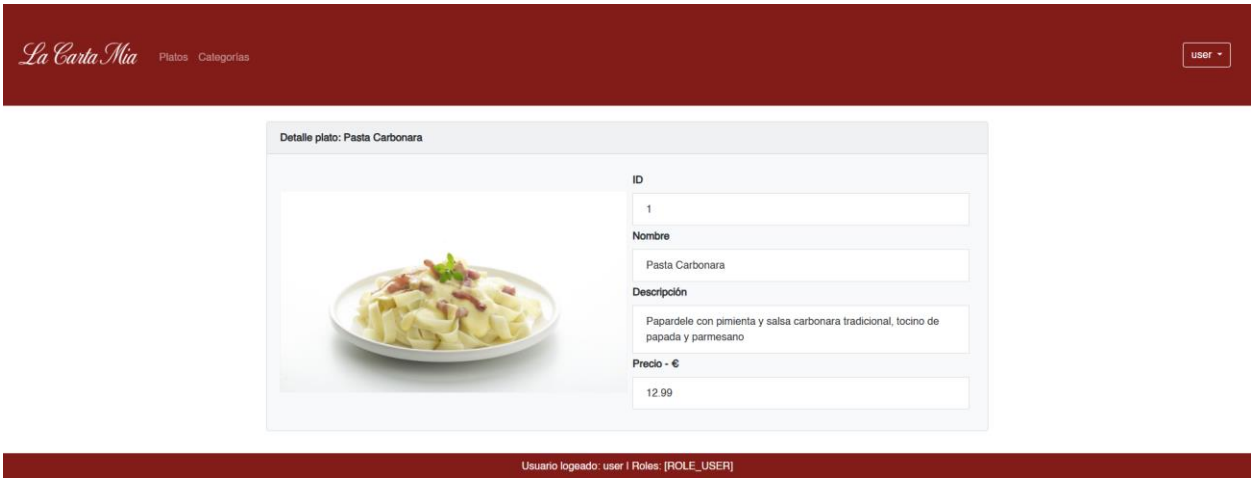

Ilustración 50. Detalle plato - Aplicación web

Ocurre exactamente lo mismo con las categorías, podemos acceder a su detalle y vemos su información pertinente.

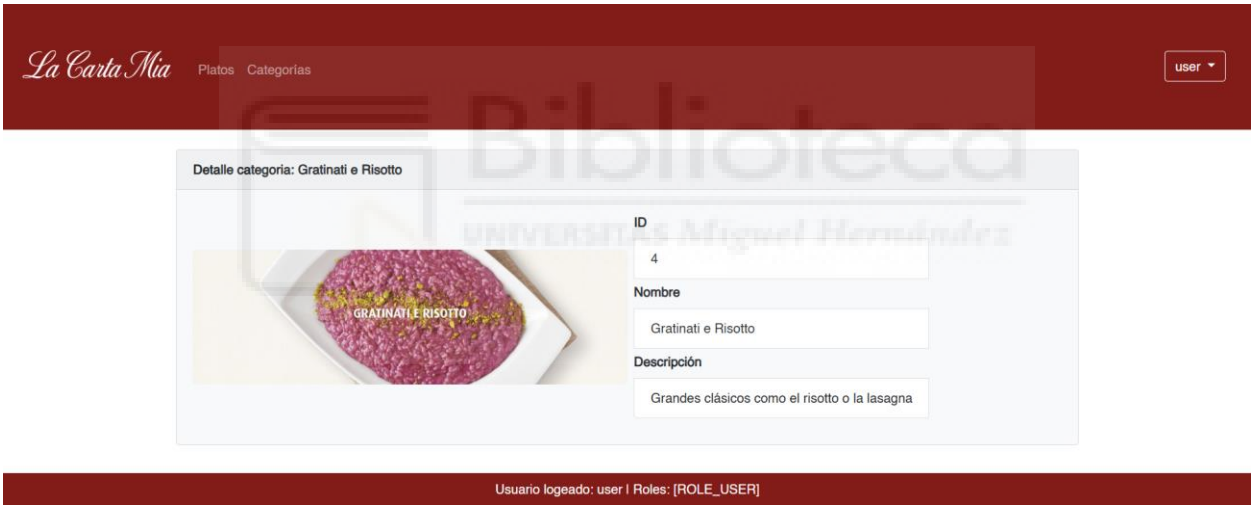

Ilustración 51. Detalle de una categoría - Aplicación web

Para poder seguir probando el resto de las funcionalidades, necesitamos un usuario con más permisos, para ello primero cerramos sesión abriendo el desplegable de arriba a la derecha del menú.

Ilustración 52. Desplegable para cerrar sesión - Aplicación web

Esta opción, cierra la sesión actual y nos redirige a la página de inicio de sesión, avisando que la sesión se ha cerrado correctamente. A continuación, iniciaremos sesión como administradores.

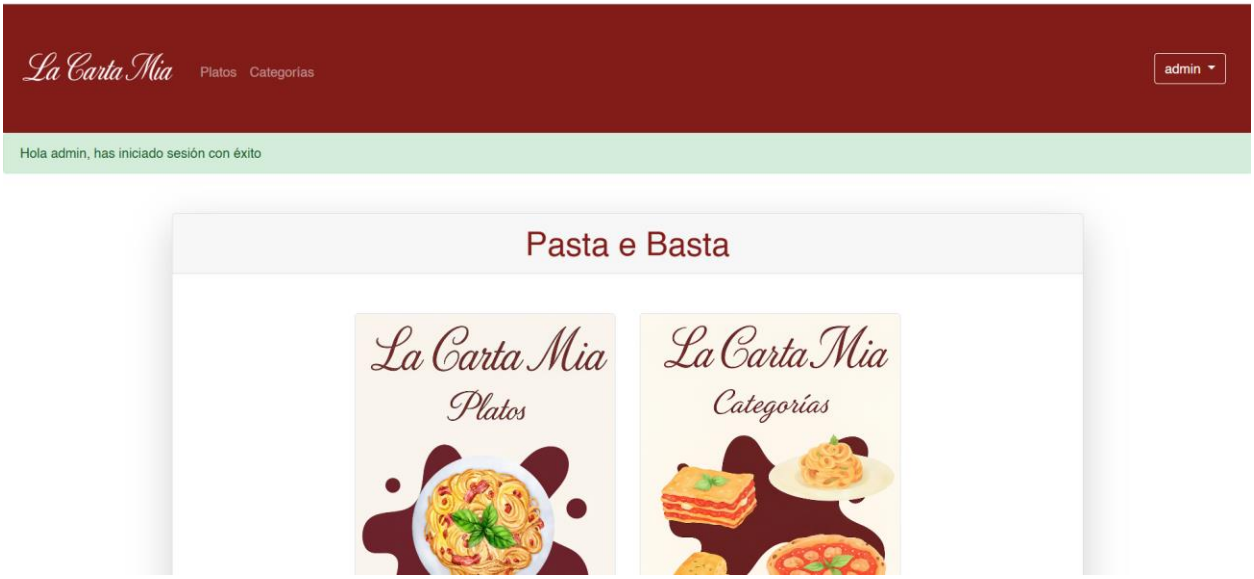

Ilustración 53. Inicio de sesión como administrador - Aplicación web

Ahora, tenemos lo sus permisos más elevados de la aplicación, los cuáles nos permiten crear, modificar y eliminar tanto platos como categorías.

Primero, accedemos al listado de platos para ver todas sus funciones.

Listado de platos

Crear Plato

ID	Nombre	Descripción	Precio	Editar	Eliminar
1	Pasta Carbonara	Papardele con pimienta y salsa carbonara tradicional, tocino de papada y parmesano	12.99€	Editar	Eliminar
2	Pasta boloñesa	Spaghetti con una maravillosa salsa hecha con carne picada, zanahoria, salsa de tomate, orégano, además de un toque de leche y vino blanco	11.99€	Editar	Eliminar
3	4 formaggi	Pizza con tomate, mozzarella, grana padano DOP, gorgonzola DOP y emmental.	9.99€	Editar	Eliminar
4	Pepperoni picante	Pizza con tomate, mozzarella y salami picante de Calabria	10.99€	Editar	Eliminar
5	Crepe	Relleno de helado de vainilla biscotto	7.89€	Editar	Eliminar
6	Coppa al gusto	Sapori: Vainilla biscotto, Nocciola (avellana), Leche merengada, Chocolate bombón, Fior di latte, Fresa y frutos rojos, Coco	6.78€	Editar	Eliminar

Primera « 1 2 3 4 5 6 » Última

Ilustración 54. Listado de platos como usuario administrador - Aplicación web

Si seleccionamos la opción de “Crear Plato”, accedemos al formulario para crear uno nuevo, pudiendo rellenar toda su información, incluida la fotografía, el precio, etc.

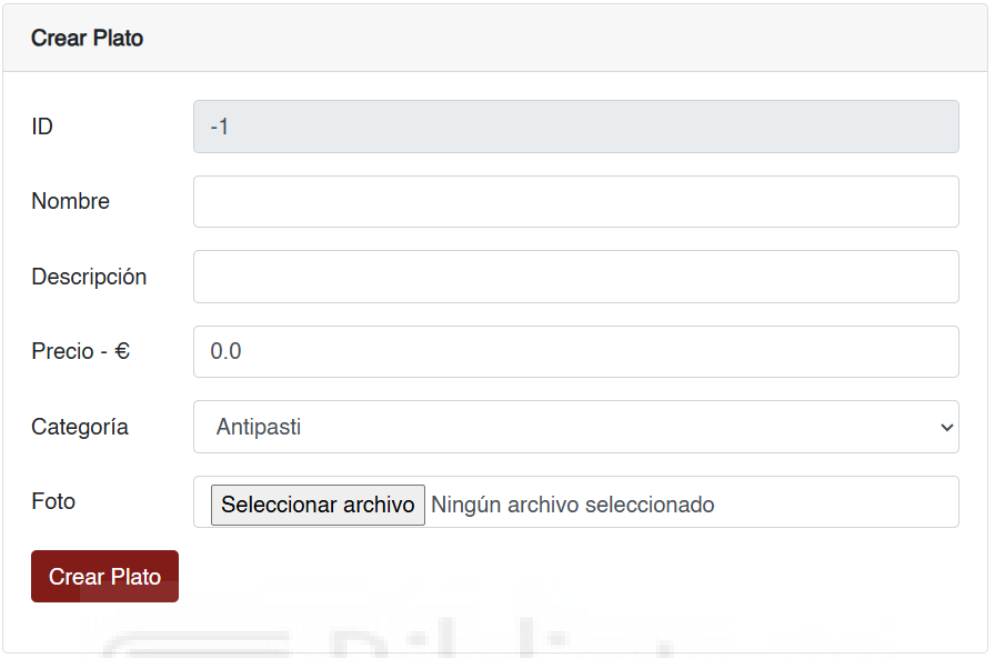

Ilustración 55. Formulario creación de un plato

El formulario comprueba si los tipos de datos son los correctos, por ejemplo, en el precio.

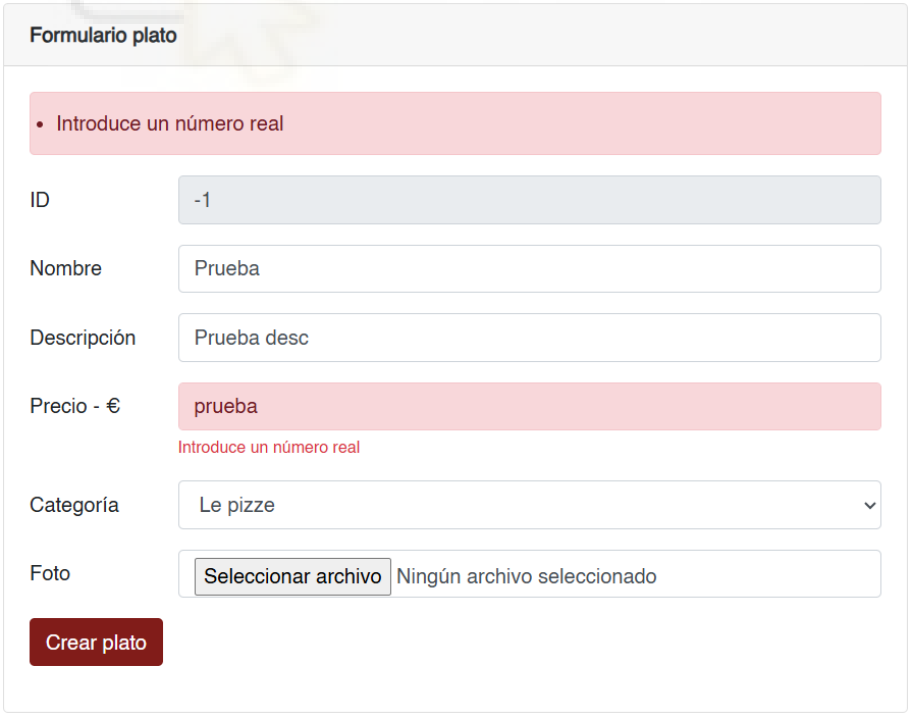

Ilustración 56. Creación nuevo plato (información errónea) - Aplicación web

Una vez todos los campos son correctos, lo podremos observar en el listado, justo el último elemento.

Ilustración 57. Comprobación existencia nuevo plato - Aplicación web

Al acceder a su detalle veríamos toda su información, en este caso accederemos a editar, para ver su funcionalidad.

Ilustración 58. Formulario editar plato - Aplicación web

Modificamos sus valores y aceptamos, posteriormente ya nos aparecerá la información actualizada.

Ilustración 59. Listado de platos después de modificar - Aplicación web

Para verificar toda la información y que todo se ha guardado correctamente, podemos acceder al detalle del plato.

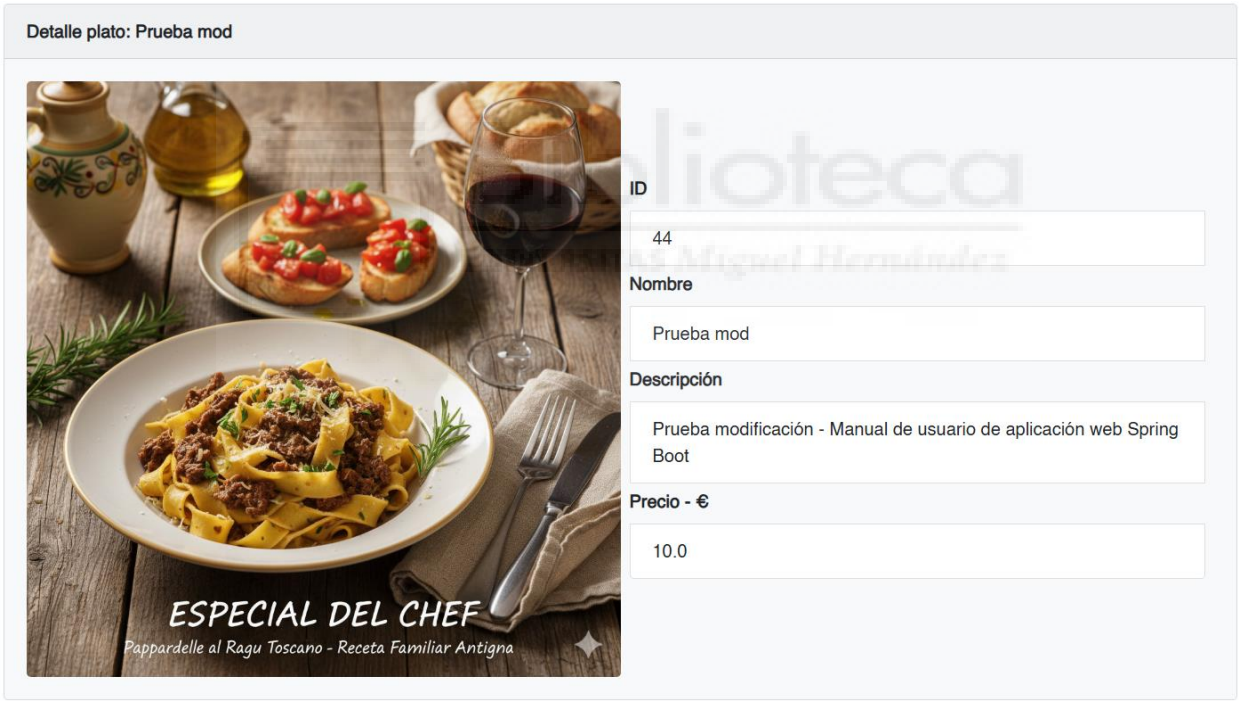

Ilustración 60. Detalle plato de prueba - Aplicación web

Por último, probaremos la eliminación del plato de prueba con la opción del listado. Dicha opción, nos abre una pestaña emergente pidiendo conformidad sobre la acción.

Una vez confirmamos la acción, la aplicación elimina el plato, toda su información asociada y nos avisa de ello.

Ilustración 61. Resultado eliminación de plato - Aplicación web

Para las categorías, seguimos los mismos pasos, primero accedemos a su listado.

Al seleccionar la opción para crear una categoría, vemos el formulario en blanco para crearla. En este caso, solamente tenemos nombre, descripción y fotografía.

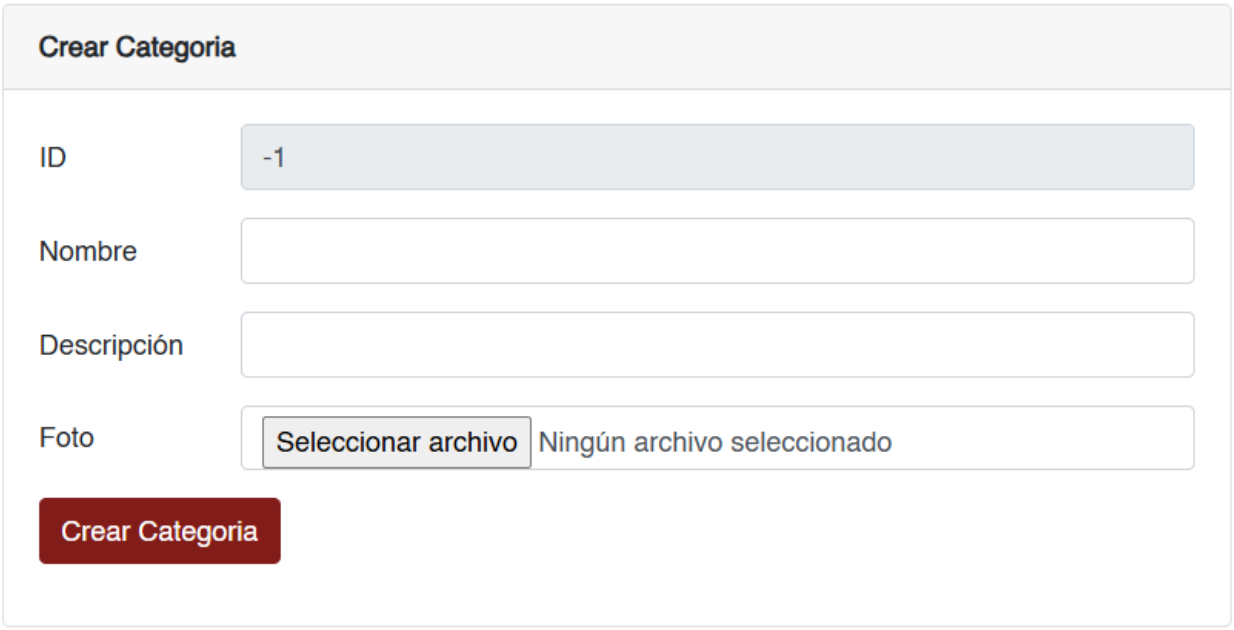

Ilustración 62. Creación de categoría - Aplicación web

Al crearla, la podemos visualizar la última del listado.

Ilustración 63. Listado con categoría de prueba creada - Aplicación web

Al igual que en el caso de los platos, podemos acceder para poder editarlas y cambiarles ciertos valores.

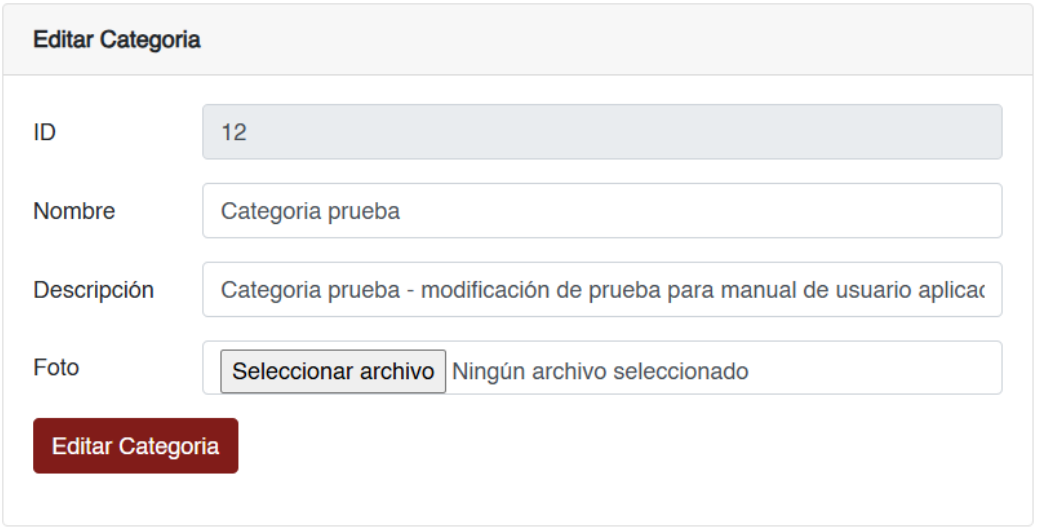

Ilustración 64. Modificación categoría de prueba - Aplicación web

Una vez la editamos, podemos visualizar los cambios en el listado.

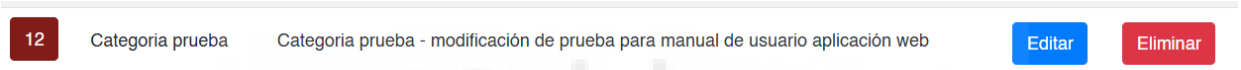

Ilustración 65. Comprobación en listado de modificación - Aplicación web

También podemos visualizar dichos cambios entrando a su detalle para ver que todo se ha creado y modificado correctamente.

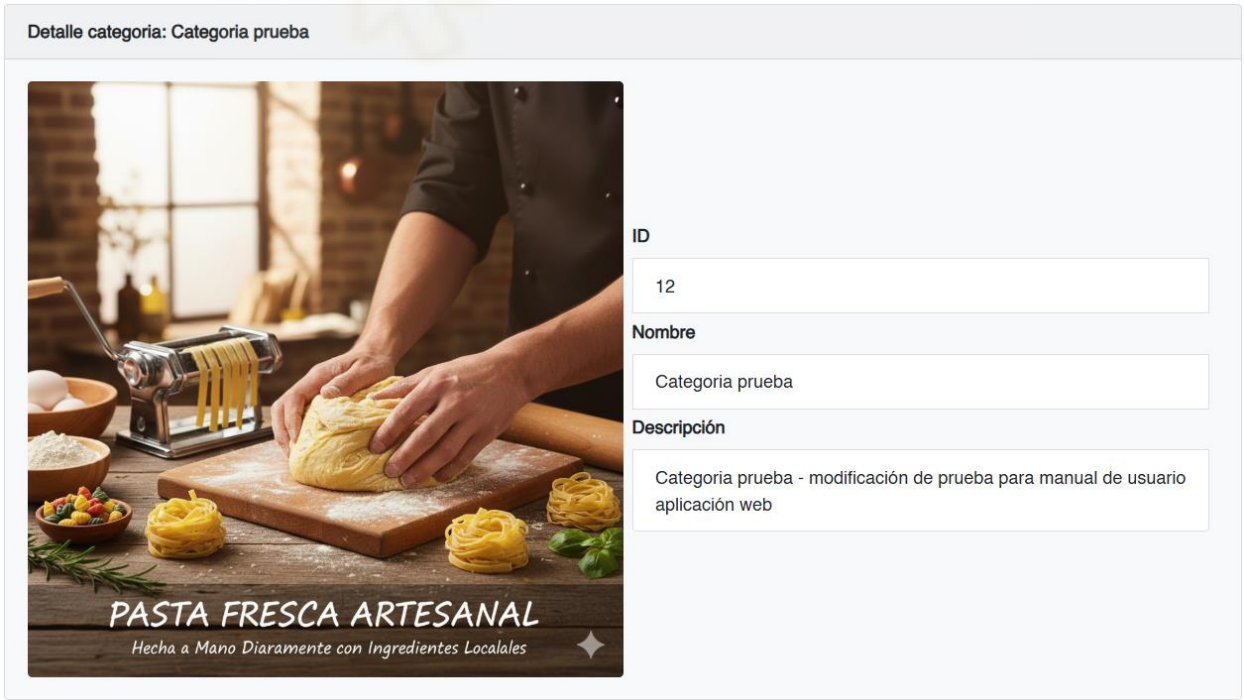

Ilustración 66. Detalle de categoría después de ser modificada

Por último, con respecto a la interfaz gráfica, nos queda la eliminación de dicha categoría, para ello seleccionamos la opción de eliminar y aceptamos el cuadro emergente.

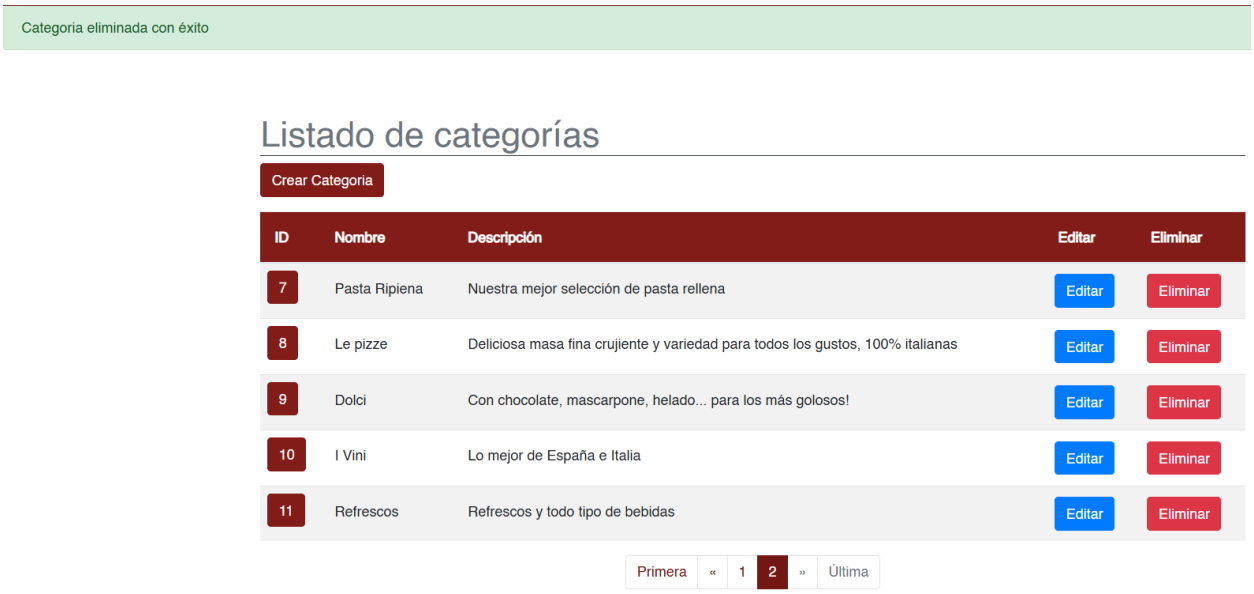

Ilustración 67. Comprobación eliminación de categoría - Aplicación web

Anexo II. Aplicación cliente (visualización de carta y realización de pedidos)

Manual de usuario

Esta aplicación está pensada para poder utilizarse tanto en dispositivo móvil como en Tablet. Por ello, existen vistas para ambos tipos de dispositivos. En este manual, mostraremos la versión de dispositivo móvil, aunque la vista sea ligeramente diferente el funcionamiento es el mismo.

Al acceder a la aplicación, lo primero que vemos es el listado de categorías con su nombre y fotografía.

Si seleccionamos la categoría, veremos todos los platos pertenecientes a dicha categoría, con su nombre, precio y fotografía.

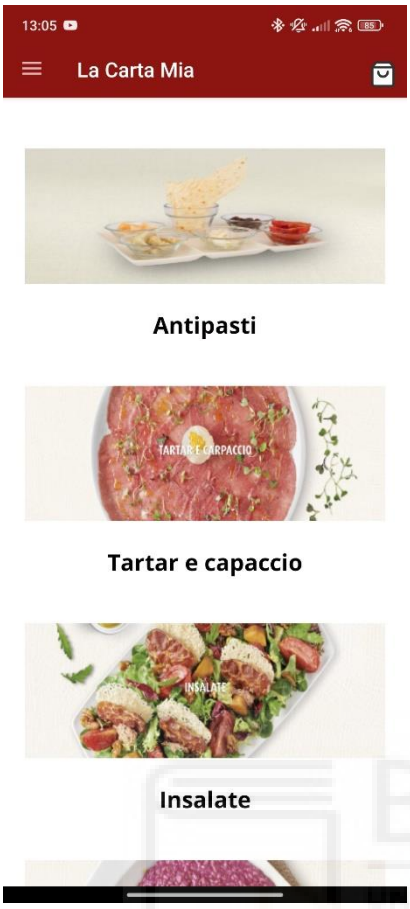

Ilustración 68. Listado de categorías - Aplicación cliente

Ilustración 69. Listado de platos de una categoría - Aplicación cliente

Al seleccionar un plato, podemos acceder a su detalle, el cual nos permite ver su descripción, nombre y precio, acompañado de un botón para añadir al carrito, el cual nos abre un modal que nos pide la cantidad de ese plato junto a un comentario (por ejemplo, sin queso, con la salsa a parte, el punto de la carne, etc.).

Ilustración 71. Detalle de cliente - Aplicación cliente

Ilustración 70. Modal añadir plato - Aplicación cliente

Cabe recalcar que, en caso de querer el mismo plato con diferentes comentarios se deben añadir en varias tandas, ya que si el comentario es diferente se considera un "plato diferente" en lo que respecta a la comanda.

Al añadir productos a la cesta, va cambiando el número que hay encima del carrito con la cantidad total. Si seleccionamos dicho elemento, podemos acceder al carrito con el listado de productos.

Ilustración 73. Comprobación número en el carrito – Aplicación cliente

Ilustración 72. Carrito lleno – Aplicación cliente

A continuación, podemos eliminar elementos del carrito, para ello pulsamos el icono de la papelera y aceptamos el diálogo que nos aparece.

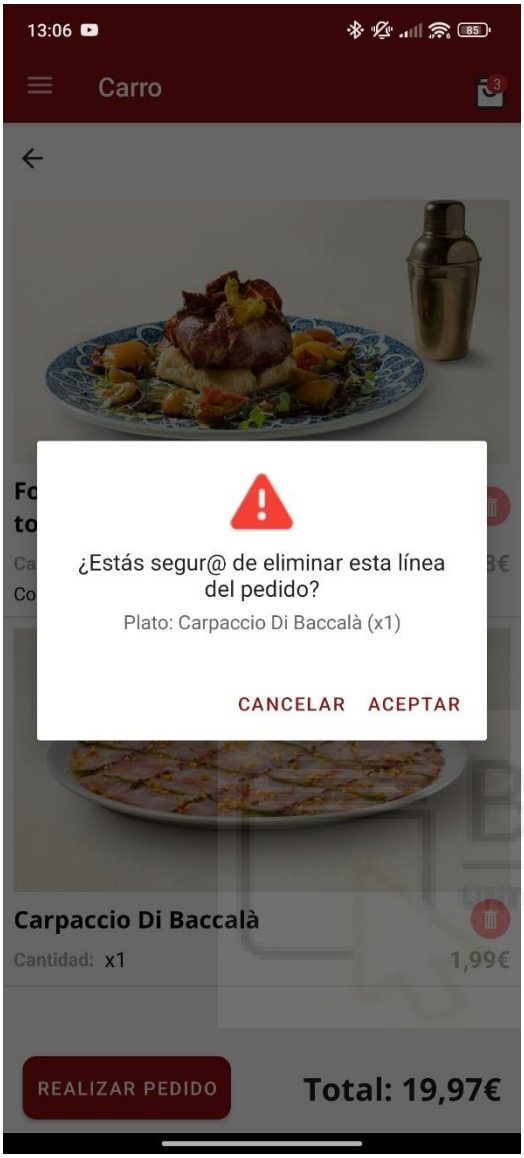

Ilustración 75. Modal borrado de plato - Aplicación cliente

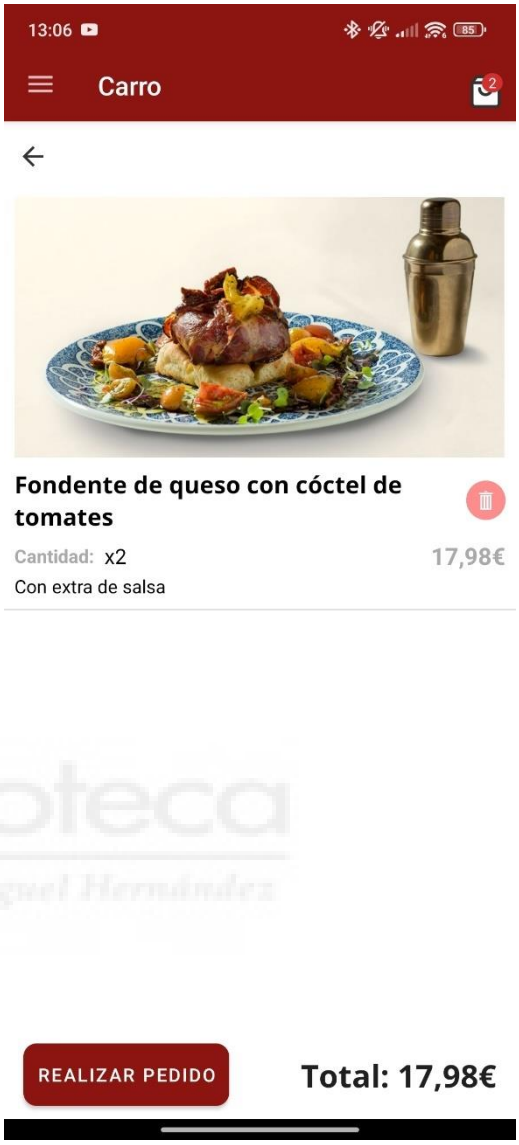

Ilustración 74. Carrito después del borrado

Como podemos observar, el número del carrito se va actualizando cada vez que se hace una actualización en el carrito, para que el cliente pueda seguir su pedido con más facilidad.

Ahora, en lo que respecta al cliente, solo nos queda realizar el pedido, el cual abre un modal pidiendo conformidad para realizar el pedido y lo envia. En caso de ser la primera comanda del pedido crea uno nuevo. En caso contrario añade una comanda al pedido existente. Además, se vacía el carrito y vuelve a la pantalla principal.

En cuanto a la parte de administración solamente cambia una cosa, el cambio de mesa. Para ello, abrimos el menú, el cual nos permite seleccionar una opción que nos abre un modal para iniciar sesión.

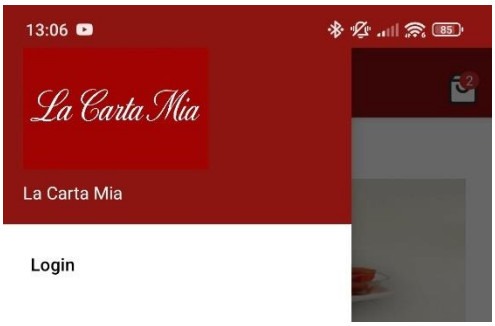

Ilustración 77. Menú para iniciar sesión

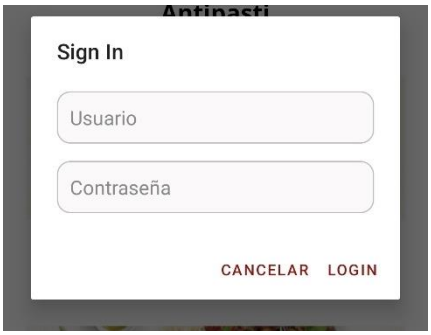

Ilustración 76. Modal inicio de sesión - Aplicación cliente

Una vez iniciada la sesión correctamente, podemos ver que el menú ha cambiado y tenemos la opción para cambiar el número de mesa, el cual nos abre un modal para cambiar el número. Además, también tenemos una opción para cerrar sesión.

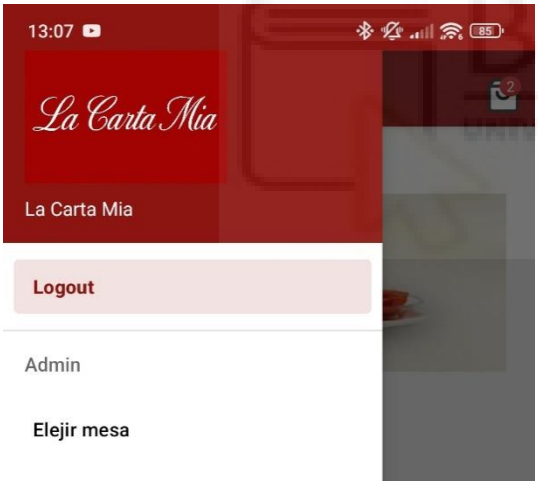

Ilustración 79. Menú de administrador - Aplicación cliente

Ilustración 78. Modal cambio número de mesa - Aplicación cliente

Anexo III. Aplicación de gestión de pedidos

Manual de usuario

Esta aplicación ahora mismo es la más simple de todas, ya que en la actualidad solamente gestiona los pedidos. Al acceder a la aplicación, lo primero que podemos observar es la pantalla de inicio de sesión. Si no iniciamos sesión, no podemos realizar ninguna acción.

Al igual que la otra aplicación, tiene diseño para Tablet y para dispositivo móvil, ya que el personal de sala probablemente tenga un dispositivo más pequeño y en la cocina o barra uno con la pantalla más grande. En este manual, mostraremos en base a la versión móvil.

Ilustración 80. Pantalla de inicio de sesión - Aplicación gestión de pedidos

Al iniciar sesión correctamente, no redirige automáticamente a la pantalla de pedidos actuales ordenador por fecha de la última comanda. Actualmente, en la prueba tenemos un pedido.

Además, en el propio listado tenemos el botón para marcar como pagado un pedido, el cual nos abre un modal de confirmación, ya que ese pedido pasará al histórico.

Además, podemos ver otra propiedad, la entrega, que tiene como posibles valores pendiente y entregado. Un pedido es marcada como entregado cuando todas sus comandas han sido entregadas, por lo que si una comanda nueva aparece ese estado cambiará.

Ilustración 81. Listado de pedidos actuales - Aplicación gestión de pedidos

Ilustración 82. Modal confirmación confirmar pago - Aplicación gestión de pedidos

Al pulsar sobre el pedido, accedemos a su detalle, el cual nos muestra todas las comandas, pudiendo marcarlas como entregadas y todos sus platos con sus respectivos comentarios.

Ilustración 83. Detalle de pedido - Aplicación gestión de pedidos

Para terminar, cuando un pedido se marca como pagado pasa al histórico y viceversa.

Para cambiar de opción, lo podemos hacer a través del menú lateral. Este menú nos ofrece dicha navegación entre histórico y actuales, además de la posibilidad de cerrar sesión.

Ilustración 85. Histórico de pedidos - Aplicación gestión de pedidos

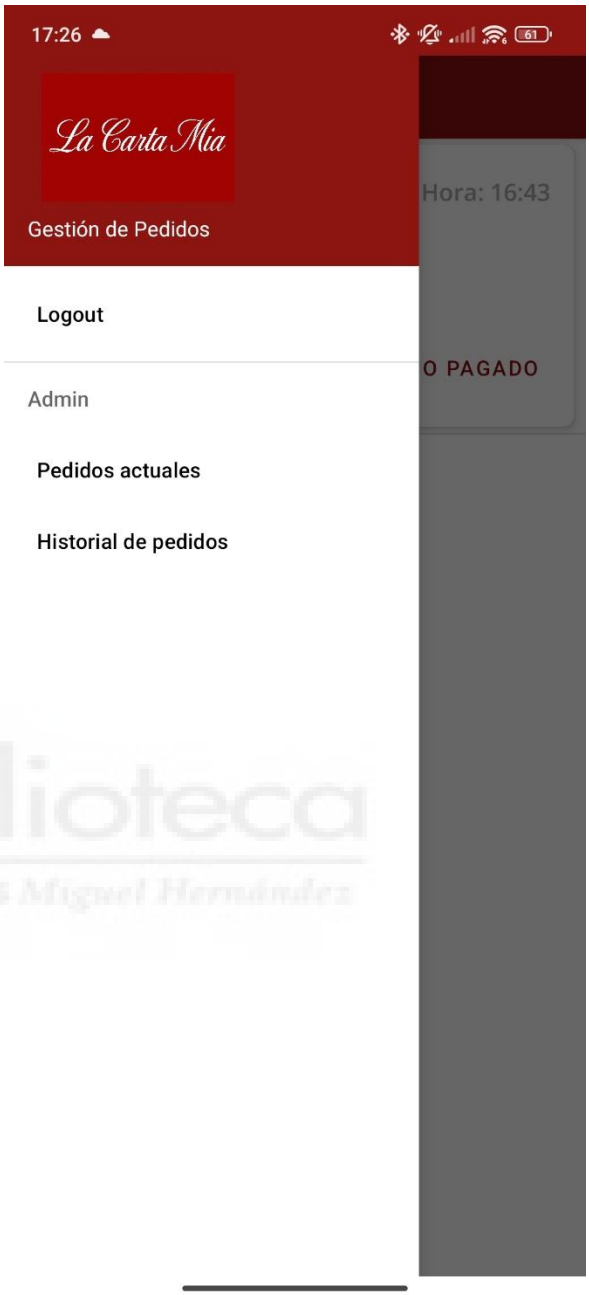

Ilustración 84. Menú lateral - Aplicación gestión de pedidos