eman ta zabal zazu

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA INDUSTRIAL DE BILBAO

GRADO EN INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2014 / 2015

EASY ORDER

MEMORIA TRABAJO FIN DE GRADO

DATOS DE LA ALUMNA O DEL ALUMNO	DATOS DEL DIRECTOR/A
NOMBRE: TAMARA	NOMBRE: MIKEL
APELLIDOS: PÉREZ GARCÍA	APELLIDOS: VILLAMAÑE GIRONÉS
	DEPARTAMENTO: LSI
FDO.:	FDO.:
FECHA: 12/02/2015	FECHA: 12/02/2015

Anexo II

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Easy Order

Trabajo Fin de Grado

Grado en Ingeniería Informática de gestión y sistemas de información

Autor
Tamara Pérez García

Director
Mikel Villamañe Gironés

Resumen

Este proyecto ha consistido en la realización de dos aplicaciones, una web y otra móvil. Ambas aplicaciones se han realizado haciendo uso de diversas tecnologías de reciente aparición en el ámbito de la tecnología y el desarrollo, tales como: HTML5, CSS3, API de Geolocalización, entre otras.

Por un lado, la aplicación web se ha realizado bajo los requisitos solicitados por un cliente real, *Mercadotecnia, Ideas y Tecnología S.A.* de México. Mientras que la aplicación móvil, se ha realizado con la intención de crear una necesidad en el cliente que aún no existía.

Las aplicaciones consisten en la realización de búsquedas de restaurantes que repartan en uno o varios códigos postales de México, y realizar un pedido a cualquiera de estos siendo pagado a través de las aplicaciones utilizando pasarelas de pago ofrecidas por MIT conjuntamente con el banco Santander. Además la aplicación web también ofrece un apartado personalizado para restaurantes y administradores de la aplicación que permitirán personalizar diversos aspectos de la aplicación.

Índice general

1. Introducción **12**
- 1.1. Descripción . . . 12
- 1.2. Planteamiento del problema . . . 12
- 1.3. Justificación . . . 13
- 1.4. Glosario de términos . . . 14

2. Planteamiento inicial **15**
- 2.1. Objetivos . . . 15
- 2.2. Arquitectura . . . 15
- 2.3. Comunicación con el cliente . . . 18
- 2.4. Alcance . . . 19
  - 2.4.1. Ciclo de vida . . . 19
  - 2.4.2. EDT . . . 20
  - 2.4.3. Paquetes de trabajo . . . 22
- 2.5. Planificación temporal . . . 37
- 2.6. Herramientas . . . 38
  - 2.6.1. Herramientas de hardware . . . 38
  - 2.6.2. Herramientas de software . . . 39
- 2.7. Gestión de riesgos . . . 40
- 2.8. Evaluación económica . . . 45
  - 2.8.1. Ingresos . . . 45
  - 2.8.2. Gastos . . . 45
  - 2.8.3. Conclusiones de la viabilidad económica . . . 47

3. Antecedentes **48**
- 3.1. Situación actual y alternativas . . . 48
- 3.2. Tecnologías a utilizar . . . 51
  - 3.2.1. Desarrollo web . . . 51
  - 3.2.2. Android . . . 54

4. Captura de requisitos **60**
- 4.1. Web . . . 60
  - 4.1.1. Modelo de casos de uso . . . 61

4.1.2. Jerarquía de actores . . . 63
4.2. Android . . . 64
4.2.1. Modelo de casos de uso . . . 64
4.2.2. Jerarquía de actores . . . 65
4.3. Modelo de dominio . . . 65
4.3.1. Entidades . . . 67

5. Análisis y diseño **70**
5.1. Base de datos . . . 70
5.1.1. Diagrama ER . . . 71
5.1.2. Triggers . . . 73
5.2. Diagrama de clases . . . 73
5.2.1. Aplicación web . . . 73
5.2.2. Aplicación Android . . . 76
5.3. Diseño . . . 77

6. Desarrollo **79**
6.1. Aplicación web . . . 79
6.1.1. Estructura del proyecto . . . 84
6.1.2. Comunicaciones . . . 84
6.1.3. Geolocalización . . . 91
6.1.4. Otras funcionalidades . . . 92
6.2. Aplicación Android . . . 93
6.2.1. Estructuración proyecto . . . 94
6.2.2. Comunicación servidor . . . 95

7. Verificación y evaluación **99**

8. Conclusiones y trabajo futuro **131**
8.1. División del trabajo aplicación web . . . 131
8.2. Tiempo real invertido . . . 131
8.3. Cliente . . . 133
8.3.1. Gestión de dudas . . . 133
8.3.2. Reuniones . . . 133
8.4. Mejoras/Ampliaciones . . . 134
8.5. Conclusión final . . . 135

Bibliografía **137**

Índice de figuras

2.1. Modelo de comunicación a través de AJAX 17
2.2. Arquitectura 18
2.3. Jerarquía MIT para el proyecto Easy Order 18
2.4. Ciclo de vida del software 20
2.5. EDT 21
2.6. Funcionalidad EDT 21
2.7. Diagrama de Gantt 38

3.1. Captura de *Sin Delantal* 49
3.2. Captura de *GrubHub* 49
3.3. Captura de *Just Eat* 50
3.4. Captura de *La Nevera Roja* 51
3.5. Estructura HTML5 52
3.6. Estructura detallada HTML5 53
3.7. Arquitectura Android 55
3.8. Ciclo de vida Android 58

4.1. Diagrama CU Web 61
4.2. Jerarquía de actores Web 63
4.3. Diagrama CU Android 64
4.4. Jerarquía de actores Android 65
4.5. Modelo de dominio 66
4.6. Entidades modelo de dominio 67

5.1. Diagrama ER 71
5.2. Entidades diagrama ER 72
5.3. Trigger Ticket 73
5.4. Diagrama de clases Web 74
5.5. Diagrama Javascript 76
5.6. Diagrama de clases Android 77

6.1. Estructura documento HTML5 80
6.2. Ejemplo pattern HTML5 80
6.3. Easy Order sin plantilla 81

6.4. Easy Order con plantilla . . . 82
6.5. Ejemplo JS plantilla . . . 82
6.6. Ejemplo estructura proyecto . . . 84
6.7. . . . 85
6.8. Ejemplo de envío de datos mediante AJAX desde JavaScript en el proyecto 85
6.9. Ejemplo de recogida de datos en el servidor con PHP . . . 85
6.10. Ejemplo formato JSON . . . 86
6.11. Ejemplo llamada SOAP . . . 87
6.12. Construcción de la URL . . . 89
6.13. Ejemplo conexión BBDD . . . 90
6.14. Ejemplo de una consulta . . . 90
6.15. Ejemplo de una consulta almacenada en JSON . . . 90
6.16. Cómo geolocalizarse desde la aplicación web . . . 91
6.17. Ejemplo de uso de la geolocalización . . . 92
6.18. Permisos necesarios . . . 94
6.19. Ejemplo estructura proyecto . . . 95
6.20. Ejemplo petición asíncrona . . . 96
6.21. Ejemplo extends petición asíncrona . . . 96
6.22. Ejemplo parámetros petición asíncrona . . . 97
6.23. Cómo geolocalizarse desde la aplicación movil . . . 97
6.24. Ejemplo de uso de la geolocalización . . . 98

8.1. Tabla planificación estimada VS real . . . 132
2. CU Identificarse . . . 138
3. Interfaces CU Identificarse . . . 139
4. CU Registrar cliente . . . 140
5. Interfaces CU Registrar cliente . . . 141
6. Interfaces CU Registrar cliente . . . 142
7. CU Registrar restaurante . . . 143
8. Interfaces CU Registrar restaurante . . . 144
9. Interfaces CU Registrar restaurante . . . 145
10. CU Búsqueda restaurantes . . . 146
11. Interfaces CU Búsqueda restaurantes . . . 147
12. Extend CU Realizar pedido . . . 148
13. Interfaces Extend CU Realizar pedido . . . 149
14. Extend CU Registrar ficticio . . . 150
15. Interfaces Extend CU Realizar pedido . . . 151
16. CU Gestionar pedidos . . . 152
17. Interfaces CU Gestionar pedidos . . . 153
18. Extend CU Pagar . . . 154
19. Interfaces Extend CU Pagar . . . 155
20. Extend CU Webpay . . . 156
21. Interfaces Extend CU Webpay . . . 157

22. Interfaces Extend CU Webpay . . . 158
23. Extend CU Points2 . . . 159
24. Interfaces Extend CU Points2 . . . 160
25. Extend CU InvisibleCard . . . 161
26. Extend CU Calificar . . . 162
27. Interfaces Extend CU Calificar . . . 163
28. Interfaces Extend CU Calificar . . . 164
29. CU Modificar menú . . . 165
30. Interfaces CU Modificar menú . . . 166
31. Extend CU Dar de alta categoría . . . 167
32. Interfaces CU Dar de alta categoría . . . 168
33. Extend CU Dar de alta plato . . . 169
34. Interfaces CU Dar de alta plato . . . 170
35. CU Modificar datos . . . 171
36. Interfaces CU Modificar datos . . . 171
37. Extend CU Modificar información . . . 172
38. Interfaces CU Modificar información . . . 173
39. Extend CU Modificar reparto . . . 174
40. Formulario añadir código postal . . . 175
41. Interfaces CU Modificar información . . . 175
42. CU Gestionar pedidos . . . 176
43. Interfaces CU Gestionar pedidos . . . 177
44. CU Desconectarse . . . 178
45. Interfaces CU Desconectarse . . . 179
46. CU Gestionar restaurantes . . . 180
47. Interfaces CU Gestionar restaurantes . . . 181
48. CU Extraer reportes . . . 182
49. Interfaces CU Extraer reportes . . . 183
50. CU Identificarse . . . 184
51. Interfaces CU Identificarse . . . 185
52. CU Desconectarse . . . 186
53. Interfaces CU Desconectarse . . . 187
54. CU Gestionar pedidos . . . 188
55. Interfaces CU Gestionar pedidos . . . 189
56. CU Pagar . . . 190
57. Interfaces CU Pagar . . . 191
58. Extend CU Points2 . . . 192
59. Interfaces Extend CU Points2 . . . 193
60. Extend CU Calificar . . . 194
61. Interfaces Extend CU Calificar . . . 195
62. CU Búsqueda restaurantes . . . 196
63. Interfaces CU Búsqueda restaurantes . . . 197
64. Interfaces CU Búsqueda restaurantes . . . 197

65. Extend CU Realizar pedido 198
66. Interfaces Extend CU Realizar pedido 199
67. Diagrama de secuencia identificarse 201
68. Diagrama de secuencia registrar cliente 203
69. Diagrama de secuencia registrar restaurante 205
70. Diagrama de secuencia búsqueda restaurantes 208
71. Diagrama de secuencia desconectarse 210
72. Diagrama de secuencia identificarse 211
73. Diagrama de secuencia desconectarse 212
74. Inicio sesión PhpMyAdmin 234
75. Pasos creación base de datos PhpMyAdmin 234
76. Pasos creación base de datos PhpMyAdmin 235
77. Pasos creación base de datos PhpMyAdmin 235

Índice de cuadros

2.1. Suma horas/tarea tabla 1 . . . 23
2.2. Suma horas/tarea tabla 2 . . . 24
2.3. Suma horas/tarea tabla 3 . . . 25
2.4. Paquete de trabajo x.1 . . . 26
2.5. Paquete de trabajo x.2.1 . . . 26
2.6. Paquete de trabajo x.2.2 . . . 27
2.7. Paquete de trabajo x.1 . . . 27
2.8. Paquete de trabajo 1.1 . . . 28
2.9. Paquete de trabajo 1.2 . . . 28
2.10. Paquete de trabajo 1.3.1 . . . 28
2.11. Paquete de trabajo 1.3.2 . . . 29
2.12. Paquete de trabajo 1.3.3 . . . 29
2.13. Paquete de trabajo 1.4.1.1 . . . 29
2.14. Paquete de trabajo 1.4.2.1 . . . 30
2.15. Paquete de trabajo 1.4.2.2 . . . 30
2.16. Paquete de trabajo 1.4.2.3 . . . 30
2.17. Paquete de trabajo 1.5.1 . . . 31
2.18. Paquete de trabajo 1.5.2 . . . 31
2.19. Paquete de trabajo 1.5.3 . . . 31
2.20. Paquete de trabajo 1.5.4 . . . 32
2.21. Paquete de trabajo 1.5.5 . . . 32
2.22. Paquete de trabajo 1.5.6 . . . 32
2.23. Paquete de trabajo 1.5.7.1 . . . 33
2.24. Paquete de trabajo 2.1 . . . 33
2.25. Paquete de trabajo 2.2 . . . 33
2.26. Paquete de trabajo 2.3.1 . . . 34
2.27. Paquete de trabajo 2.4.1 . . . 34
2.28. Paquete de trabajo 2.4.2 . . . 34
2.29. Paquete de trabajo 2.4.3 . . . 35
2.30. Paquete de trabajo 3.1 . . . 35
2.31. Paquete de trabajo 3.2 . . . 35
2.32. Paquete de trabajo 3.3.1 . . . 36
2.33. Paquete de trabajo 3.3.2 . . . 36

2.34. Paquete de trabajo 3.3.3 . . . 36
2.35. Paquete de trabajo 3.3.4.1 . . . 37
2.36. Paquete de trabajo 3.3.4.2 . . . 37
2.37. Fechas inicio/fin tareas . . . 38
2.38. Caída del servidor . . . 41
2.39. Pérdida de datos . . . 41
2.40. El cliente rechaza los requerimientos ofrecidos . . . 42
2.41. El cliente no paga la aplicación . . . 42
2.42. Cambios en los requisitos funcionales de la aplicación . . . 43
2.43. Comunicación ineficiente con el cliente . . . 43
2.44. Bajas por enfermedad . . . 44
2.45. Desastre natural en el lugar donde se aloja el servidor de integración . . . 44
2.46. Tabla de viabilidad económica . . . 47

7.1. Pruebas escenario I - parte 1 . . . 100
7.2. Pruebas escenario I - parte 2 . . . 101
7.3. Pruebas escenario I - parte 3 . . . 102
7.4. Pruebas escenario I - parte 4 . . . 103
7.5. Pruebas escenario I - parte 5 . . . 104
7.6. Pruebas escenario I - parte 6 . . . 105
7.7. Pruebas escenario I - parte 7 . . . 106
7.8. Pruebas escenario I - parte 8 . . . 107
7.9. Pruebas escenario II - parte 1 . . . 108
7.10. Pruebas escenario II - parte 2 . . . 109
7.11. Pruebas escenario II - parte 3 . . . 110
7.12. Pruebas escenario II - parte 4 . . . 111
7.13. Pruebas escenario III - parte 1 . . . 112
7.14. Pruebas escenario III - parte 2 . . . 113
7.15. Pruebas escenario III - parte 3 . . . 114
7.16. Pruebas escenario IV . . . 115
7.17. Pruebas escenario V - parte 1 . . . 116
7.18. Pruebas escenario V - parte 2 . . . 117
7.19. Pruebas escenario V - parte 3 . . . 118
7.20. Pruebas escenario V - parte 4 . . . 119
7.21. Pruebas escenario V - parte 5 . . . 120
7.22. Pruebas escenario V - parte 6 . . . 121
7.23. Pruebas escenario VI - parte 1 . . . 122
7.24. Pruebas escenario VI - parte 2 . . . 123
7.25. Pruebas escenario VI - parte 3 . . . 124
7.26. Pruebas escenario VII - parte 1 . . . 125
7.27. Pruebas escenario VII - parte 2 . . . 126
7.28. Pruebas escenario VII - parte 3 . . . 127
7.29. Pruebas escenario VII - parte 4 . . . 128

7.30. Pruebas escenario VIII 129
7.31. Pruebas escenario IX 130

Capítulo 1

Introducción

En este capítulo se detallará en qué va a consistir el proyecto a desarrollar y cómo surge la idea. También se describirá brevemente cómo se abordará el problema y qué tecnologías se utilizarán.

1.1. Descripción

El proyecto *Easy Order* consistirá en la realización de una aplicación web y otra móvil a un cliente real que permita realizar búsquedas de restaurantes en México y realizar un pedido a cualquiera de estos pagando a través de unas pasarelas de pago (invocadas a través de servicios web) ofrecidas por la empresa *Mercadotecnia Ideas y Tecnología S.A. de México* (*MIT*, de ahora en adelante).

En la aplicación web, podrán registrarse clientes y restaurantes, y cualquier funcionalidad de la aplicación web se verá adaptada a smartphones, tablets y PCs.

En la aplicación móvil, sin embargo, podrán acceder únicamente los clientes ya registrados anteriormente en la página web. La idea principal es que la aplicación web recoja todas las funcionalidades solicitadas por *MIT*, mientras que la aplicación móvil impulse las compras por parte de los clientes. La aplicación móvil no pretende sustituir el uso de la aplicación web, sino más bien complementarla.

1.2. Planteamiento del problema

MIT es una empresa ubicada en México desde el año 2000 y, según aseguran ellos mismos, está posicionada como la primera en desarrollar y operar soluciones de pago online para diversos sectores de México. Su página web se puede consultar en la siguiente URL: `www.mitec.com.mx`. Ahí, además, podemos observar que cuenta con una amplia cartera de clientes, incluyendo, por ejemplo, entre los más reseñables a empresas tales como: diferentes aseguradoras (*Mapfre, Ace Seguros, Zurich, Santander, Qualitas...*),

Brother International, Cablevision, diferentes gobiernos y municipios, cadenas hoteleras, colegios y universidades de México, *Hertz, Editorial Planeta, The Phone House,* etcétera.

MIT ha desarrollado diversos servicios web que brindan algunos métodos de pago online (entre otros productos) y que actualmente gestiona con el banco Santander. Son los siguientes:

- **Webpay**: Integración del pago con tarjeta de crédito (tarjetas American Express, VISA y Mastercard), con la posibilidad de convertir las divisas a pesos mexicanos.
- **Points2**: Pago con certificados otorgados a un determinado cliente (una especie de cheque electrónico).
- **Invisible Card**: Pago con el móvil a través de una aplicación, fotografiando un código QR mostrado en la aplicación web.

MIT desea impulsar sus métodos de pago creando una página web en la que, realizando pedidos de comida a diferentes restaurantes, los clientes puedan pagar con ellos. Así, se conseguiría promover la venta de comida por parte de los restaurantes de México que deseasen unirse a la plataforma y, la razón más importante para *MIT*: que se obtuviese una pequeña comisión por cada venta realizada por el uso de sus medios de pago (beneficio económico) y además conseguir una mayor aceptación de sus métodos de pago en la sociedad mexicana (publicidad).

1.3. Justificación

El proyecto solicitado por *MIT* (tal como se ha indicado en la sección anterior), se denominaría **Easy Order**. Las competencias del desarrollo del mismo se obtuvieron a través de la plataforma Freelancer en la siguiente URL: `www.freelancer.com.es`. *MIT* solicita el desarrollo de una aplicación web que pueda usarse desde cualquier dispositivo (móviles, tablets, PCs...) utilizando tecnologías web novedosas en la parte del cliente (HTML5) y siendo obligatorio el uso de PHP en el lado del servidor. También solicita la configuración del servidor que se utilizará de cara a los usuarios reales. Sin embargo, con el objetivo de obtener un mayor beneficio económico y de mejorar las relaciones con *MIT*, se desarrollará también una aplicación móvil que incluirá una parte de la funcionalidad solicitada en la aplicación web. Posteriormente, esta aplicación móvil se le ofrecerá a *MIT* por el precio que se indicará en la sección 2.8.

La aplicación web se desarrollará conjuntamente por dos alumnos de la EUITI (Ekaitz Portillo y yo), sin embargo, la aplicación móvil se desarrollará únicamente por mí. El desarrollo de la aplicación web, con objetivo de evitar interferencias para la realización de este Trabajo de Fin de Grado, se ha planificado y aplicado una división clara, detallada en la sección 2.5.

1.4. Glosario de términos

- TFG: Trabajo Fin de Grado.
- SGBD: Sistema de Gestión de Bases de Datos (por ejemplo, *MySQL*).
- BBDD: Bases de datos.
- AJAX: Acrónimo de Asynchronous JavaScript And XML.
- PHP: Lenguaje de programación utilizado en el lado del servidor.
- MySQL: Sistema de Gestión de Base de Datos utilizado en el proyecto.
- API: Interfaz de Programación de Aplicaciones.
- SMTP: Protocolo para la Transferencia Simple de Mail.
- Platillo: Denominación en México de los platos que ofrece un restaurante.
- Categoría: Nombre de la agrupación de los platos de un restaurante.
- Menú: Conjunto de categorías y platos ofrecidos por un restaurante.
- Tipos de comida: Clasificación del tipo de comida que ofrece un restaurante (comida china, comida japonesa...).
- Colonia: Denominación en México de un vecindario sin autonomía jurisdiccional.
- Calle: Espacio urbano.
- Estado: Denominación en México de las divisiones territoriales del País.
- Municipio: Denominación en México de las divisiones territoriales del Estado.
- Centro de pagos: Sistema de gestión que permite recibir y enviar pagos ofrecido por MIT.

Capítulo 2

Planteamiento inicial

En el capítulo actual se detallará la gestión del proyecto incluyendo los objetivos, el alcance, la arquitectura, las herramientas a utilizar, la planificación temporal y económica y la gestión de dudas, reuniones y riesgos.

2.1. Objetivos

Los objetivos propuestos para el presente proyecto, se pueden resumir en el logro satisfactorio del desarrollo completo y con un funcionamiento correcto de la aplicación web y móvil en el plazo solicitado, consiguiendo además que la aplicación web sea intuitiva y sencilla para los usuarios finales. También es importante que, además de intuitiva, sea también eficiente.

Adicionalmente, y en beneficio de las personas que realizan el proyecto, considero que hay que conseguir mantener una buena relación con *MIT* con el objetivo de que, tras la finalización del proyecto, se obtenga un contrato de mantenimiento de las aplicaciones desarrolladas. Además, esto permitirá obtener una buena referencia de cara a obtener futuros clientes. Evidentemente, también se desea obtener un beneficio económico por la realización del proyecto.

Además, se desea finalizar el Trabajo de Fin de Grado y obtener una buena valoración con el mismo. Personalmente, pretendo obtener la experiencia y los conocimientos necesarios para poder trabajar como desarrolladora web, o poder desarrollar proyectos del ámbito web más rápidamente. Paralelamente quiero aumentar mi portfolio de aplicaciones, para que, en caso de ser necesario o si surge la oportunidad en el futuro, poder desempeñar la labor profesional en este ámbito por cuenta propia.

MIT desea que la aplicación sea finalizada en plazo y de manera correcta.

2.2. Arquitectura

Para definir la arquitectura del proyecto es necesario definir previamente los entornos utilizados, para poder comprender correctamente la arquitectura de la utilizada en el proyecto. En este caso, diferenciaremos los entornos de *MIT* y los que se utilizarán para

el desarrollo de la aplicación.

MIT, independientemente de cómo se estructure el desarrollo de la aplicación, dispone de 3 entornos para poder utilizar sus productos:

- **Desarrollo:** Es el entorno con el que se comunicará la aplicación web/móvil desde un entorno de desarrollo local para poder testear el funcionamiento de las diversas funcionalidades que harán uso de los servicios web de *MIT*.
- **Aseguramiento de calidad (QA):** Es el entorno que se utilizará para simular las condiciones de producción.
- **Producción:** Es el entorno que se utilizará cuando la aplicación esté ya en producción, es decir, funcionando para usuarios reales. En este entorno se encuentra el servidor que aloja la aplicación web y el SGBD.

MIT también ha solicitado que los desarrolladores estructuremos el desarrollo del proyecto en 2 entornos, muy similares a los suyos:

- **Desarrollo:** En este entorno se desarrollará la aplicación y se harán pruebas unitarias y otras pruebas que se consideren oportunas en local. Este entorno sólo se comunicará con el entorno de desarrollo de *MIT* y únicamente en caso de necesidad (por ejemplo, testear los métodos de pago).
- **Integración**: En este entorno se probarán funcionalmente las aplicaciones en un servidor ajeno a *MIT* que se encontrará en idénticas condiciones al servidor de producción de *MIT*. En este caso, se comunicará con el entorno de Aseguramiento de Calidad de *MIT*.

Por otra parte, la arquitectura a utilizar en la aplicación es una **Arquitectura Orientada a Servicios (SOA)** basada en la conocida arquitectura cliente-servidor siendo la parte del servidor propiedad de MIT. La Arquitectura Orientada a Servicios se basa en implementar en la parte del servidor diversos servicios relativamente sencillos que implementen funcionalidades invocadas desde diferentes clientes (por ejemplo: navegadores web, dispositivos móviles, otras aplicaciones web en PHP, Java, Python...). De esta forma, abstraemos la tecnología utilizada en un entorno concreto, ya que se utilizará un formato común de intercambio de datos (JSON).
El flujo de información se puede observar gráficamente en la figura 2.2. El proceso a seguir para la comunicación cliente-servidor es el siguiente:

1. La aplicación ejecutada en el cliente (en un navegador web), lanzará una petición AJAX mediante JavaScript al servidor de producción.

   Es decir, a través de llamadas asíncronas desde JavaScript se tratará la información que gestiona la aplicación. Para el intercambio de datos entre cliente y servidor se

utilizará el formato JSON. El modelo de comunicación a través de AJAX se puede observar gráficamente en la figura 2.1.

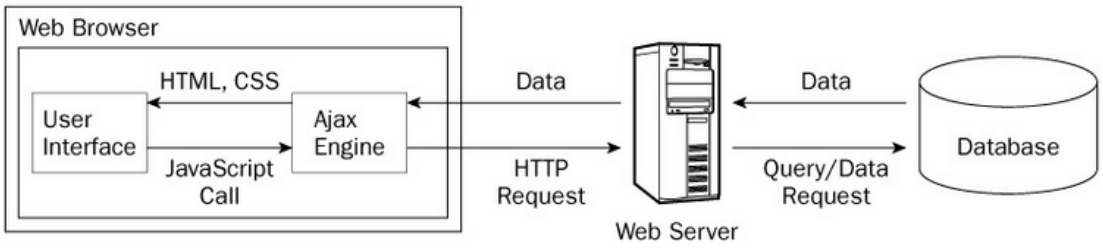


Imagen extraída de `http://www.yaldex.com/ajax-tutorial-4/BBL0013.html`

Figura 2.1: Modelo de comunicación a través de AJAX

2. El código PHP del servidor de producción accederá a la BBDD, si fuese necesario, para modificar/añadir/eliminar/recuperar datos.

3. El servidor de producción, en el caso de tratarse de un pago o del alta de sucursales, se comunicará con los servicios web ofrecidos por *MIT*.

4. Éstos, a su vez, se comunicarán con el Centro de Pagos. Se desconoce como se efectuará este flujo puesto que no se necesitará interactuar directamente con el Centro de Pagos desde el servidor de producción.

5. Si fuese necesario el envío de algún correo, se comunicará con el servidor SMTP de correo.

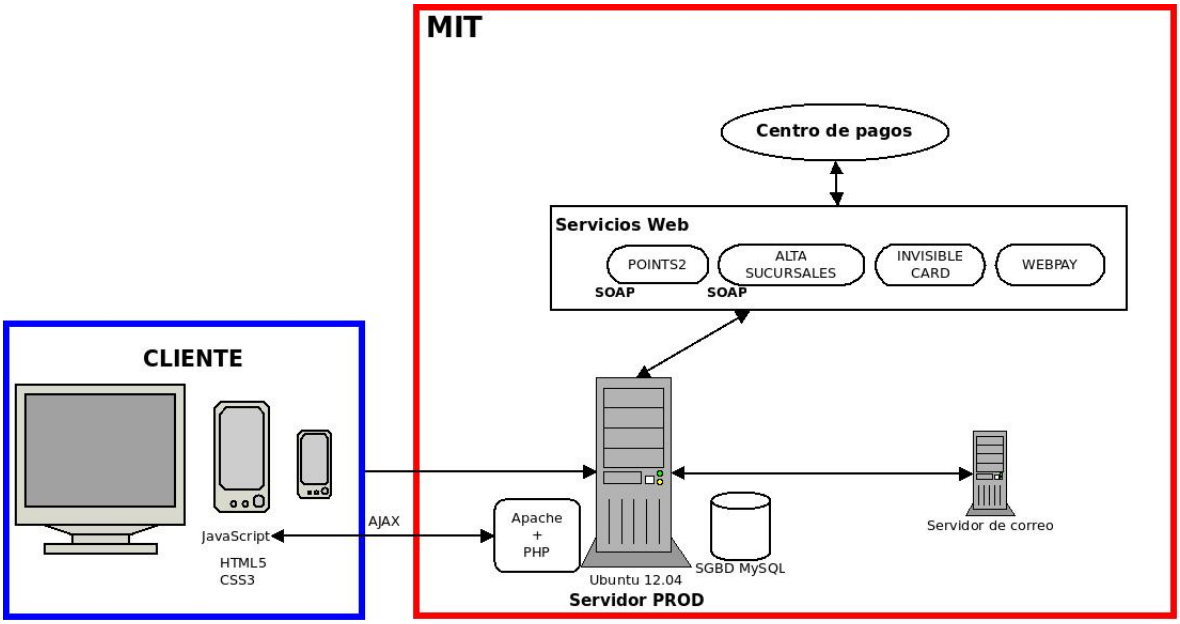


Figura 2.2: Arquitectura

2.3. Comunicación con el cliente

MIT nos ha comunicado qué jerarquía van a seguir dentro de la empresa para el desarrollo de este proyecto. Es importante conocerla, ya que a la hora de gestionar las dudas o de solicitar reuniones hay que saber a quién dirigirse.

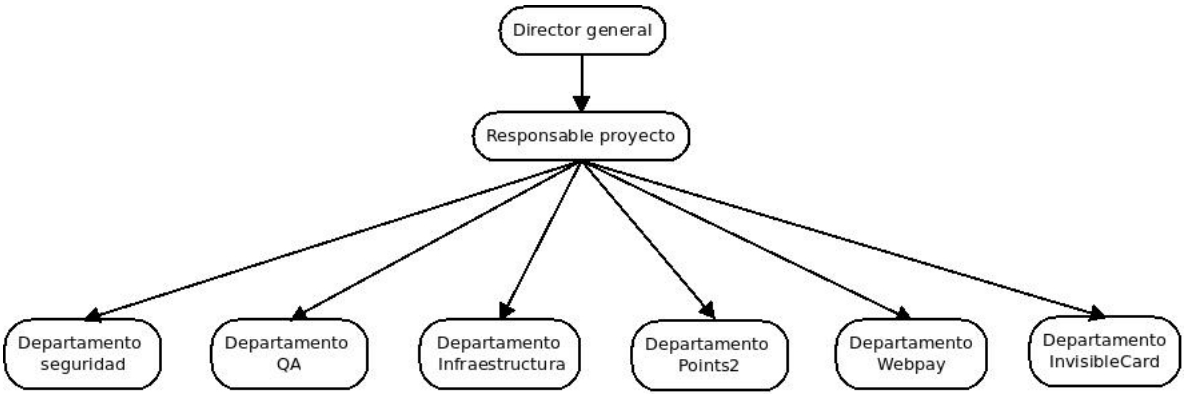


Figura 2.3: Jerarquía MIT para el proyecto Easy Order

Como se puede observar en la figura 2.3 *MIT* sigue una organización jerárquica tradicional (estructura con forma de pirámide). En la cual, el Director General es la persona con mayor responsabilidad.

Del mismo modo, *MIT* solicita gestionar las dudas de una forma concreta: a través de e-mail y por Skype. La definición de los requisitos de la aplicación la desarrolló el Director General, sin embargo, el Responsable del Proyecto es con quién gestionaremos todas las dudas. Posteriormente, él será quien se encargará de dirigírselas a quién correspondiese.

Para gestionar las dudas, habrá diferentes tipos reuniones:

- **Seguimiento**: Aquellas en las que estará presente el Responsable del Proyecto y en las que se verificará el estado global del proyecto. La intención es hacer una reunión de este tipo cada 15 días.
- **Funcionales**: Aquellas en las que se tratará algún punto concreto de alguna funcionalidad solicitada inicialmente. Bien para modificarla, eliminarla o añadir alguna funcionalidad nueva.
- **Técnicas**: Aquellas en las que se quiera hacer alguna subida a producción[1] relativamente grande, o cuando se quiera aplicar alguna configuración al servidor de producción. Estas reuniones se realizarán principalmente con el departamento de infraestructura debido a que eran ellos los que debían autorizar las subidas a producción y las configuraciones en el servidor. En estas reuniones también se utilizará *Team Viewer* con control remoto, ya que no se dispone de acceso al servidor de producción fuera de la red interna de *MIT*.
- **Pruebas:** Aquellas en las que se testearán las diversas funcionalidades con el Responsable del Proyecto, o bien, con el responsable del Departamento de QA, o bien con algún otro departamento (principalmente, con los departamentos involucrados en los métodos de pago).

2.4. Alcance

En esta sección se hablará sobre el ciclo de vida que ha seguido el proyecto, el EDT y los paquetes de trabajo.

2.4.1. Ciclo de vida

Inicialmente se le propuso a MIT un ciclo de vida ágil, como puede ser Scrum[2]. Sin embargo, MIT se opuso y finalmente se ha decidido hacer un ciclo de vida tradicional y lineal: Captura de requisitos, análisis, diseño, desarrollo y pruebas. Finalizando una fase completa, se pasaba a la siguiente.

[1] Denominamos **subida a producción** cuando se sube al servidor de producción de *MIT* el código actualizado y con previa validación del cliente en nuestro entorno de integración

[2] Es una metodología de desarrollo ágil que define el proceso de desarrollo de un proyecto de forma no secuencial.

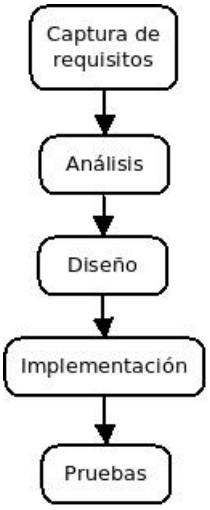


Figura 2.4: Ciclo de vida del software

2.4.2. EDT

En la figura 2.5 se puede observar la Estructura de Descomposición del Trabajo. En él se refleja el orden en el que se realizarán las tareas del proyecto. Se divide, principalmente, en tres fases:

- Los **requerimientos iniciales** constan de la primera parte que será entregada al cliente. Esta entrega será una aplicación web con un flujo completo para el usuario, es decir, habrá un administrador que pueda dar de alta/baja restaurantes, habrá restaurantes que puedan subir su menú y gestionar pedidos, y habrá clientes que puedan realizar pedidos.

- Los **requerimientos posteriores** constan de la segunda y última entrega al cliente. Esta entrega constará de la adaptación a dispositivos móviles y de extras en la aplicación, como lo son calificar pedidos, compartir por redes sociales o extraer reportes. Estos requerimientos han sido solicitados a la vez que los requerimientos iniciales, pero no hay que tenerlos finalizados hasta que se finalicen y funcionen los iniciales.

- Los **requerimientos ofrecidos** no serán ninguna entrega, ya que se desarrollará con intención de ofrecérsela al cliente a posteriori (una vez entregadas y validadas las tareas anteriores).

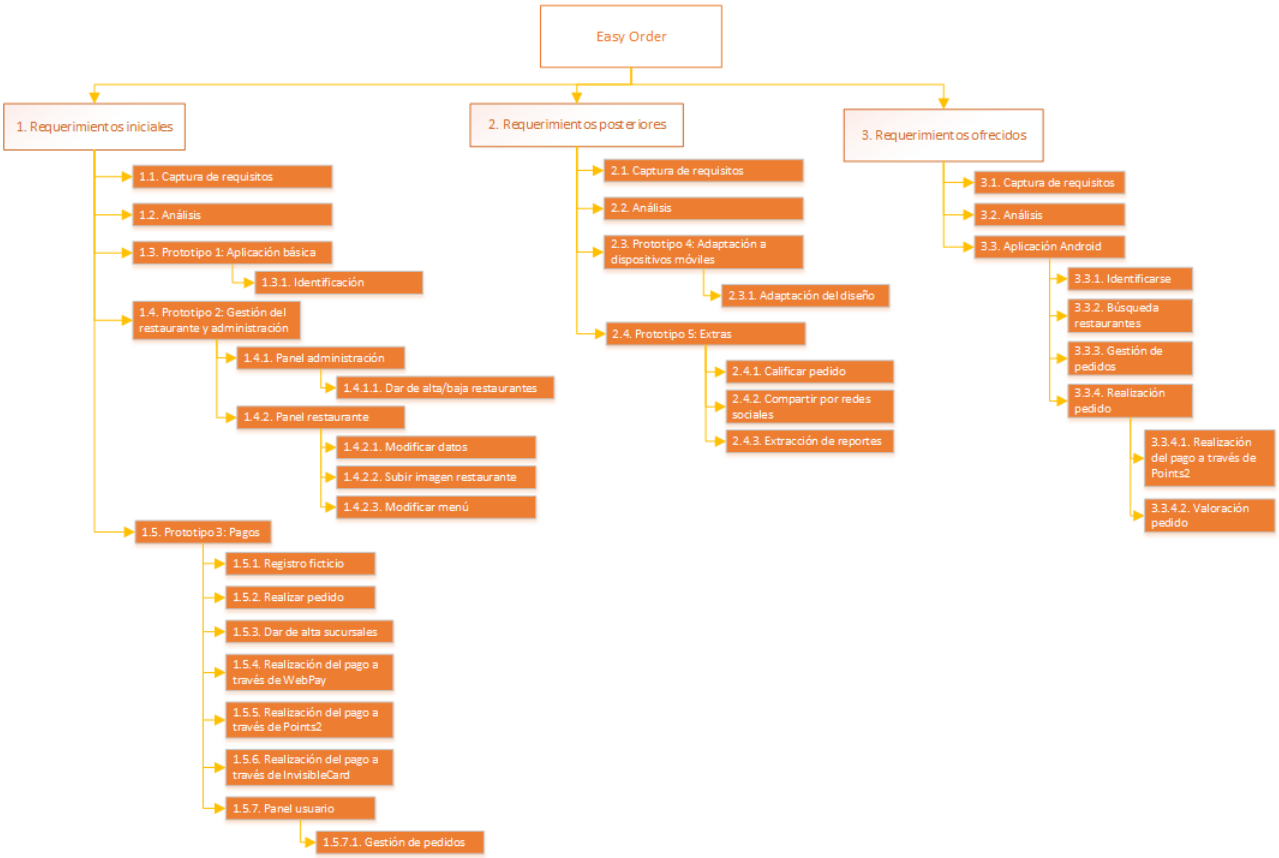


Figura 2.5: EDT

Además, cada una de las funcionalidades indicadas en el EDT (detalladas en el anexo 8.5) se subdividen de la siguiente forma:

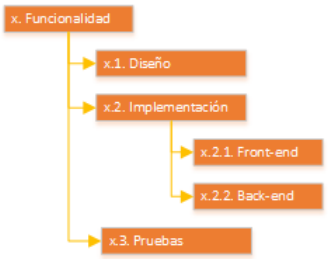


Figura 2.6: Funcionalidad EDT

El desglose de tareas detallado se puede ver en el Anexo 8.5.

2.4.3. Paquetes de trabajo

En los paquetes de trabajo referentes a las funcionalidades se han obviado las entradas, salidas y recursos necesarios debido a que vienen detalladas en los paquetes de trabajo comunes (2.4.3.1).

Tarea	Horas Tamara	Horas Ekaitz
1.1. Captura de requisitos	17'5h	17'5h
1.2. Análisis	7h	7h
1.3.1.1. Diseño	4h	4h
1.3.1.2.1. Front-end	10h	-
1.3.1.2.2. Back-end	3h	3h
1.3.1.3. Pruebas	-	2h
1.3.2.1. Diseño	7h	7h
1.3.2.2.1. Front-end	8h	-
1.3.2.2.2. Back-end	-	6h
1.3.2.3. Pruebas	2h	2h
1.3.3.1. Diseño	4h	4h
1.3.3.2.1. Front-end	12h	-
1.3.3.2.1. Back-end	-	9h
1.3.3.3. Pruebas	1h	1h
1.4.1.1.1. Diseño	4h	-
1.4.1.1.2.1. Front-end	3h	-
1.4.1.1.2.2. Back-end	2h	-
1.4.1.1.3. Pruebas	-	1h
1.4.2.1.1. Diseño	8h	-
1.4.2.1.2.1. Front-end	12h	-
1.4.2.1.2.2. Back-end	-	10h
1.4.2.1.3. Pruebas	4h	4h
1.4.2.2.1. Diseño	-	3h
1.4.2.2.2.1. Front-end	3h	-
1.4.2.2.2.2. Back-end	-	6h
1.4.2.2.3. Pruebas	-	2h
1.4.2.3.1. Diseño	8h	-
1.4.2.3.2.1. Front-end	10h	-
1.4.2.3.2.2. Back-end	-	8h
1.4.2.3.3. Pruebas	1'5h	1'5h
1.5.1.1. Diseño	8h	-
1.5.1.2.1. Front-end	10h	-
1.5.1.2.2. Back-end	7'5h	7'5h
1.5.1.3. Pruebas	4h	4h
1.5.2.1. Diseño	10h	10h
1.5.2.2. Front-end	7h	9h
1.5.2.2.2. Back-end	-	-
1.5.2.3. Pruebas	4h	4h

Cuadro 2.1: Suma horas/tarea tabla 1

Tarea	Horas Tamara	Horas Ekaitz
1.5.3.1. Diseño	-	3h
1.5.3.2.1. Back-end	-	14h
1.5.3.3. Pruebas	-	10h
1.5.4.1. Diseño	6'5h	6'5h
1.5.4.2.1. Front-end	4h	-
1.5.4.2.2. Back-end	7h	7h
1.5.4.3. Pruebas	5h	5h
1.5.5.1. Diseño	6h	6h
1.5.5.2.1. Front-end	4h	-
1.5.5.2.2. Back-end	7h	7h
1.5.5.3. Pruebas	3h	3h
1.5.6.1. Diseño	3h	-
1.5.6.2.1. Front-end	1h	-
1.5.6.2.2. Back-end	7'5h	7'5h
1.5.6.3. Pruebas	2h	2h
1.5.7.1.1. Diseño	8h	-
1.5.7.1.2.1. Front-end	12h	-
1.5.7.1.2.2. Back-end	4h	4h
1.5.7.1.3. Pruebas	1'5h	1'5h
2.1. Captura de requisitos	7'5h	7'5h
2.2. Análisis	4h	4h
2.3.1.1.1. Front-end	42h	-
2.3.1.2. Pruebas	18h	18h
2.4.1.1. Diseño	6h	-
2.4.1.2.1. Front-end	8h	-
2.4.1.2.2. Back-end	-	4h
2.4.1.3. Pruebas	2h	2h
2.4.2.1. Diseño	1h	-
2.4.2.2.1. Front-end	2h	-
2.4.2.3. Pruebas	1h	-
2.4.3.1. Diseño	-	4h
2.4.3.2.1. Front-end	1h	-
2.4.3.2.2. Back-end	-	12h
2.4.3.3. Pruebas	-	3h

Cuadro 2.2: Suma horas/tarea tabla 2

Tarea	Horas Tamara	Horas Ekaitz
3.1. Captura de requisitos	12h	-
3.2. Análisis	1h	-
3.3.1.1. Diseño	5h	-
3.3.1.2.1. Front-end	5h	-
3.3.1.2.2. Back-end	3h	-
3.3.1.3. Pruebas	2h	-
3.3.2.1. Diseño	5h	-
3.3.2.2.1. Front-end	16h	-
3.3.2.2.2. Back-end	10h	-
3.3.2.3. Pruebas	5h	-
3.3.3.1. Diseño	10h	-
3.3.3.2.1. Front-end	20h	-
3.3.3.2.2. Back-end	32h	-
3.3.3.3. Pruebas	10h	-
3.3.4.1.1. Diseño	5h	-
3.3.4.1.2.1. Front-end	4h	-
3.3.4.1.2.2. Back-end	12h	-
3.3.4.1.3. Pruebas	10h	-
3.3.4.2.1. Diseño	2h	-
3.3.4.2.2.1. Front-end	4h	-
3.3.4.2.2.2. Back-end	3h	-
3.3.4.2.3. Pruebas	1h	-
TOTAL HORAS	**573 horas**	**264 horas**

Cuadro 2.3: Suma horas/tarea tabla 3

El total de horas planificadas que trabajará Ekaitz Portillo en este proyecto ascienden a **264 horas**, mientras que las horas planificadas para mi serán **573 horas**. Siendo el total del proyecto **867 horas**.

Se puede ver claramente que se superan ampliamente las horas estimadas para un trabajo de fin de grado de 12 créditos ECTS (1 crédito ECTS = 30h).

A continuación se expondrán los paquetes de trabajo que son comunes a todas las funcionalidades y posteriormente, los paquetes de trabajo de las tareas indicadas en el EDT. Con respecto a los paquetes comunes, en el Anexo 8.5 se detallará la duración y responsable de cada una de estas tareas.

2.4.3.1. Paquetes de trabajo comunes a las funcionalidades

Hay ciertos paquetes que son comunes a todas las funcionalidades, por lo que no se describirán independientemente sino genéricamente. A continuación, se detallará cada

una de ellas.

Diseño

Paquete de trabajo	x.1 Diseño
Descripción	Creación de los diagramas de secuencia de la funcionalidad asociada.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Diagrama de casos de uso correspondiente a la funcionalidad y diagrama de base de datos.
Salidas/Entregables	Diagramas de secuencia.
Recursos necesarios	Dia
Precedencias	El análisis de la funcionalidad asociada.

Cuadro 2.4: Paquete de trabajo x.1

Front-end

Paquete de trabajo	x.2.1 Front-end
Descripción	Implementación del código de la parte alojada en el cliente.
Responsable	Tamara Pérez
Entradas	Diagramas de secuencia de la funcionalidad asociada.
Salidas/Entregables	≻ Aplicación web: Código Javascript, HTML y CSS. ≻ Aplicación Android: Código XML.
Recursos necesarios	≻ Aplicación web: IDE Eclipse, navegador web y Cliente SSH. ≻ Aplicación Android: IDE Eclipse, SDK, ADT, dispositivo Android y Cliente SSH.

Cuadro 2.5: Paquete de trabajo x.2.1

Back-end

Paquete de trabajo	x.2.2 Back-end
Descripción	Implementación del código de la parte alojada en el servidor.
Responsable	Ekaitz Portillo
Entradas	Diagramas de secuencia de la funcionalidad asociada.
Salidas/Entregables	≻ Aplicación web: Código PHP. ≻ Aplicación Android: Código Java y PHP.
Recursos necesarios	≻ Aplicación web: IDE Eclipse, PDT, navegador web y Cliente SSH. ≻ Aplicación Android: IDE Eclipse, dispositivo Android, PDT, JDK y Cliente SSH.

Cuadro 2.6: Paquete de trabajo x.2.2

Pruebas

Paquete de trabajo	x.3 Pruebas
Descripción	Realización de pruebas funcionales (es decir, no unitarias) de la funcionalidad asociada.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Funcionalidad.
Salidas/Entregables	Documento que resuma los resultados a estas pruebas.
Recursos necesarios	≻ Aplicación web: Microsoft Excel y navegador web. ≻ Aplicación Android: Microsoft Excel y dispositivo Android.
Precedencias	La implementación de la funcionalidad asociada.

Cuadro 2.7: Paquete de trabajo x.1

2.4.3.2. Requerimientos iniciales

A continuación se detallarán los paquetes de trabajo de los requerimientos iniciales.

Captura de requisitos

Paquete de trabajo	1.1 Captura de requisitos
Duración (horas/persona)	17.5 horas
Descripción	Reunirse con el cliente para obtener la captura de requisitos inicial.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	-
Salidas/Entregables	Diagrama de casos de uso extendido y modelo de dominio.
Recursos necesarios	Skype, Microsoft Excel, Microsoft Word, Dia.
Precedencias	-

Cuadro 2.8: Paquete de trabajo 1.1

Análisis

Paquete de trabajo	1.2 Análisis
Duración (horas/persona)	7 horas
Descripción	Transformación del modelo de dominio en una base de datos.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Modelo de dominio.
Salidas/Entregables	Diagrama de bases de datos.
Recursos necesarios	Dia, Cliente MySQL.
Precedencias	1.1

Cuadro 2.9: Paquete de trabajo 1.2

Identificación

Paquete de trabajo	1.3.1 Identificación
Duración (horas/persona)	13 horas
Descripción	Desarrollo de la funcionalidad que permite identificarse a un cliente, a un restaurante o a un administrador en la aplicación.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.10: Paquete de trabajo 1.3.1

Registro

Paquete de trabajo	1.3.2 Registro
Duración (horas/persona)	16 horas
Descripción	Desarrollo de la funcionalidad que permite a cualquier usuario registrarse en el sistema como cliente o restaurante.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.11: Paquete de trabajo 1.3.2

Buscar restaurantes

Paquete de trabajo	1.3.3 Buscar restaurantes
Duración (horas/persona)	15.5 horas
Descripción	Desarrollo de la funcionalidad que permite realizar búsquedas de restaurantes en función de un código postal.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.12: Paquete de trabajo 1.3.3

Dar de alta/baja restaurantes

Paquete de trabajo	1.4.1.1 Dar de alta/baja restaurantes
Duración (horas/persona)	10 horas
Descripción	Desarrollo de la funcionalidad que permite a un administrador dar de alta y baja restaurantes.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.13: Paquete de trabajo 1.4.1.1

Modificar datos

Paquete de trabajo	1.4.2.1 Modificar datos
Duración (horas/persona)	19 horas
Descripción	Desarrollo de la funcionalidad que permite a un restaurante modificar sus datos.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.14: Paquete de trabajo 1.4.2.1

Subir imagen restaurante

Paquete de trabajo	1.4.2.2 Subir imagen restaurante
Duración (horas/persona)	7 horas
Descripción	Desarrollo de la funcionalidad que permite a un restaurante subir una imagen.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.15: Paquete de trabajo 1.4.2.2

Modificar menú

Paquete de trabajo	1.4.2.3 Modificar menú
Duración (horas/persona)	14.5 horas
Descripción	Desarrollo de la funcionalidad que permite a un restaurante subir una imagen.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.16: Paquete de trabajo 1.4.2.3

Registro ficticio

Paquete de trabajo	1.5.1 Registro ficticio
Duración (horas/persona)	20.5 horas
Descripción	Desarrollo de la funcionalidad que permite registrarse a un usuario de forma temporal.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.17: Paquete de trabajo 1.5.1

Realizar pedido

Paquete de trabajo	1.5.2 Realizar pedido
Duración (horas/persona)	22 horas
Descripción	Desarrollo de la funcionalidad que permite realizar un pedido.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	1.3.2; 1.3.3; 1.5.1

Cuadro 2.18: Paquete de trabajo 1.5.2

Dar de alta sucursales

Paquete de trabajo	1.5.3 Dar de alta sucursales
Duración (horas/persona)	13.5 horas
Descripción	Desarrollo de la funcionalidad que permite dar de alta sucursales en MIT.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	1.3.2

Cuadro 2.19: Paquete de trabajo 1.5.3

Realización del pago a través de WebPay

Paquete de trabajo	1.5.4 Realización del pago a través de WebPay
Duración (horas/persona)	20.5 horas
Descripción	Desarrollo de la funcionalidad que permite registrarse a un usuario pagar a traves de WebPay.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	1.5.2

Cuadro 2.20: Paquete de trabajo 1.5.4

Realización del pago a través de Points2

Paquete de trabajo	1.5.5 Realización del pago a través de Points2
Duración (horas/persona)	18 horas
Descripción	Desarrollo de la funcionalidad que permite registrarse a un usuario pagar a traves de Points2.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	1.5.2

Cuadro 2.21: Paquete de trabajo 1.5.5

Realización del pago a través de InvisibleCard

Paquete de trabajo	1.5.6 Realización del pago a través de InvisibleCard
Duración (horas/persona)	11.5 horas
Descripción	Desarrollo de la funcionalidad que permite registrarse a un usuario pagar a través de InvisibleCard.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	1.5.2

Cuadro 2.22: Paquete de trabajo 1.5.6

Gestión de pedidos

Paquete de trabajo	1.5.7.1 Gestión de pedidos
Duración (horas/persona)	30.5 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente gestionar sus pedidos.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.23: Paquete de trabajo 1.5.7.1

2.4.3.3. Requerimientos posteriores

A continuación se detallarán los paquetes de trabajo de los requerimientos posteriores.

Captura de requisitos

Paquete de trabajo	2.1 Captura de requisitos
Duración (horas/persona)	7.5 horas
Descripción	Reunirse con el cliente para obtener una segunda captura de requisitos con los nuevos requerimientos.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Diagrama de casos de uso extendido inicial y modelo de dominio inicial.
Salidas/Entregables	Diagrama de casos de uso extendido y modelo de dominio.
Recursos necesarios	Skype, Microsoft Excel, Microsoft Word, Dia.
Precedencias	1

Cuadro 2.24: Paquete de trabajo 2.1

Análisis

Paquete de trabajo	2.2 Análisis
Duración (horas/persona)	4 horas
Descripción	Transformación del modelo de dominio en una base de datos.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Modelo de dominio.
Salidas/Entregables	Diagrama de bases de datos.
Recursos necesarios	Dia, Cliente MySQL.
Precedencias	2.1

Cuadro 2.25: Paquete de trabajo 2.2

Adaptación del diseño

Paquete de trabajo	2.3.1 Adaptación del diseño
Duración (horas/persona)	39 horas
Descripción	Adaptación del diseño de la aplicación orientándola hacia dispositivos móviles.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.26: Paquete de trabajo 2.3.1

Calificar pedido

Paquete de trabajo	2.4.1 Calificar pedido
Duración (horas/persona)	11 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente valorar un pedido.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.27: Paquete de trabajo 2.4.1

Compartir por redes sociales

Paquete de trabajo	2.4.2 Compartir por redes sociales
Duración (horas/persona)	2 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente compartir por redes sociales su satisfacción con el restaurante.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	2.4.1

Cuadro 2.28: Paquete de trabajo 2.4.2

Extracción de reportes

Paquete de trabajo	2.4.3 Extracción de reportes
Duración (horas/persona)	10 horas
Descripción	Desarrollo de la funcionalidad que permite al administrador obtener reportes de los restaurantes dados de alta en el sistema.
Responsable	Tamara Pérez y Ekaitz Portillo
Precedencias	-

Cuadro 2.29: Paquete de trabajo 2.4.3

2.4.3.4. Requerimientos ofrecidos

A continuación se detallarán los paquetes de trabajo de los requerimientos ofrecidos.

Captura de requisitos

Paquete de trabajo	3.1 Captura de requisitos
Duración (horas/persona)	6 horas
Descripción	Analizar lo que el cliente ha solicitado anteriormente para después ofrecérselo en otra aplicación.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Diagrama de casos de uso extendido y modelo de dominio.
Salidas/Entregables	Diagrama de casos de uso extendido y modelo de dominio.
Recursos necesarios	Microsoft Excel, Microsoft Word, Dia.
Precedencias	-

Cuadro 2.30: Paquete de trabajo 3.1

Análisis

Paquete de trabajo	3.2 Análisis
Duración (horas/persona)	0.5 horas
Descripción	Transformación del modelo de dominio en una base de datos.
Responsable	Tamara Pérez y Ekaitz Portillo
Entradas	Modelo de dominio.
Salidas/Entregables	Diagrama de bases de datos.
Recursos necesarios	Dia, Cliente MySQL.
Precedencias	3.1

Cuadro 2.31: Paquete de trabajo 3.2

Identificarse

Paquete de trabajo	3.3.1 Identificarse
Duración (horas/persona)	15 horas
Descripción	Desarrollo de la funcionalidad que permite identificarse a un cliente en la aplicación.
Responsable	Tamara Pérez
Precedencias	-

Cuadro 2.32: Paquete de trabajo 3.3.1

Búsqueda restaurantes

Paquete de trabajo	3.3.2 Búsqueda restaurantes
Duración (horas/persona)	36 horas
Descripción	Desarrollo de la funcionalidad que permite realizar búsquedas de restaurantes en función de un código postal.
Responsable	Tamara Pérez
Precedencias	3.3.1

Cuadro 2.33: Paquete de trabajo 3.3.2

Gestión de pedidos

Paquete de trabajo	3.3.3 Gestión de pedidos
Duración (horas/persona)	72 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente gestionar sus pedidos.
Responsable	Tamara Pérez
Precedencias	3.3.1

Cuadro 2.34: Paquete de trabajo 3.3.3

Realización del pago a través de Points2

Paquete de trabajo	3.3.4.1 Realización del pago a través de Points2
Duración (horas/persona)	31 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente pagar a través de Points2.
Responsable	Tamara Pérez
Precedencias	-

Cuadro 2.35: Paquete de trabajo 3.3.4.1

Valoración del pedido

Paquete de trabajo	3.3.4.2 Valoración del pedido
Duración (horas/persona)	10 horas
Descripción	Desarrollo de la funcionalidad que permite a un cliente valorar un pedido.
Responsable	Tamara Pérez
Precedencias	-

Cuadro 2.36: Paquete de trabajo 3.3.4.2

2.5. Planificación temporal

Para exponer el tiempo estimado de dedicación del proyecto se ha utilizado el diagrama de Gantt (figura 2.7).

En la parte superior del diagrama están representados los meses, y se puede observar también una subdivisión en semanas. A la izquierda, están representados los números de las tareas del EDT.

En este caso, se ha indicado en cada semana planificada el número de personas que estarán trabajando en esa tarea. Como puede observarse, a partir de la tarea 3.1 sólo habrá una persona asignada debido a que esa parte del proyecto únicamente la voy a realizar yo, no es conjunta (aplicación Android).

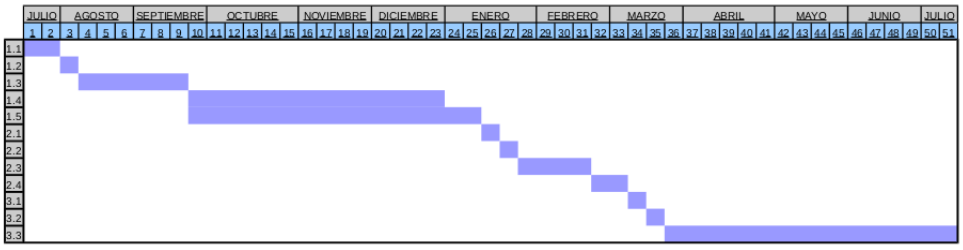


Figura 2.7: Diagrama de Gantt

A continuación, se pueden observar las fechas de inicio/fin planificadas de cada tarea en detalle:

Tarea	Fecha inicio	Fecha fin
1.1	22/07/2013	05/08/2013
1.2	05/08/2013	13/08/2013
1.3	13/08/2013	23/09/2013
1.4	25/09/2013	27/12/2013
1.5	24/09/2013	17/01/2014
2.1	21/01/2014	27/01/2014
2.2	27/01/2014	28/01/2014
2.3	29/01/2014	04/03/2014
2.4	04/03/2014	18/03/2014
3.1	18/03/2014	24/03/2014
3.2	24/03/2014	24/03/2014
3.3	25/03/2014	27/06/2014

Cuadro 2.37: Fechas inicio/fin tareas

2.6. Herramientas

En esta sección se detallarán las herramientas de hardware y software utilizadas.

2.6.1. Herramientas de hardware

- PC con diferentes navegadores para la realización de pruebas.
- Móvil y tablet con sistema operativo Android.
- Servidor cloud: Servidor para hacer uso de un entorno de integración. El cual está alojado en Amsterdam y será contratado a través de la empresa DigitalOcean (URL: `https://www.digitalocean.com/`) con las siguientes especificaciones (suficientes para tener un entorno de integración sin múltiples usuarios concurrentes):

- 512MB de RAM.
- 20GB de disco SSD.
- Ubuntu 14.04 de 32 bits como sistema operativo.

2.6.2. Herramientas de software

- IDE Eclipse: Entorno de desarrollo utilizado para la implementación del código de las aplicaciones.
- Android SDK *(Software Development Kit)*: Conjunto de herramientas necesarias para la implementación de aplicaciones en Android.
- JDK *(Java Development Kit)*: Conjunto de herramientas necesarias para la creación de aplicaciones en Java.
- PDT *(PHP Development Tools)*: Conjunto de herramientas necesarias para la implementación de aplicaciones web basadas en PHP.
- ADT *(Android Development Tools)*: Plugin de Eclipse que permite utilizar el Android SDK.
- Amateras: Plugin de Eclipse para dibujar diagramas de clases y diagramas de secuencia.
- Lyx: Herramienta para la generación de documentos con el lenguaje Latex.
- DIA: Aplicación de dibujo vectorial para la realización de diagramas
- Paquete Microsoft Office:
  - Microsoft Visio: Aplicación de dibujo vectorial para la realización de diagramas.
  - Microsoft Project: Aplicación de gestión de proyectos utilizado para la estimación temporal y económica. Utilizado únicamente con fines didácticos ya que es una herramienta que se utiliza en múltiples empresas. Herramienta que es proporcionada por Microsoft a través del MSDNAA[3]
- Libre Office Writer: Aplicación de procesamiento de textos.
- Thunderbird: Cliente de correo electrónico.
- Dropbox: Servicio de alojamiento de archivos en la nube.

[3]MSDNAA: Siglas de Microsoft Soft Developer's Network Academic Allianz (fuente: `https://msdn.microsoft.com/es-pe/default.aspx`). Un acuerdo entre universidades y Microsoft, en el cual la UPV/EHU es partícipe, por lo que los alumnos disponen de diversas aplicaciones de software de Microsoft.

- ≻ Subversion (SVN): Herramienta de control de versiones.
  - Cliente para Eclipse: Cliente de Subversion integrado en Eclipse.
  - Servidor Apache Subversion: Utilizado para el control de versiones.
- ≻ TeamViewer: Aplicación que permite el control remoto o visualización de otro equipo.
- ≻ Skype: Aplicación que permite la realización de llamadas y envío de texto.
- ≻ Protocolo de Transferencia de Archivos (FTP): Protocolo de red para la transferencia de archivos. El software utilizado para hacer uso de este protocolo es:
  - Cliente FileZilla: Aplicación que permite gestionar la transferencia de archivos gráficamente.
  - Servidor vsftpd: Utilizado para gestionar la transferencia de archivos en sistemas Unix.
- ≻ Putty: Cliente SSH. Aplicación que permite conectarse a otro equipo mediante SSH.
- ≻ MySQL Workbench: Herramienta gráfica de diseño de bases de datos que permite conectarse a un SGBD.
- ≻ MySQL Client: Herramienta gráfica que permite conectarse a un SGBD a través de la consola.

2.7. Gestión de riesgos

A la hora de realizar un proyecto es necesario identificar los riesgos, evaluarlos y plantear planes para evitar o reducir los riesgos. Además, es necesario realizar un seguimiento de los mismos a lo largo de la duración de todo el proyecto.
Los riesgos se han determinado mediante el método Delphi[2], que tiene en cuenta la opinión de un comité de expertos. En este proyecto, los expertos serán los responsables del proyecto, Ekaitz Portillo y yo.

Caída del servidor

Descripción	Caída del servidor de integración que se encuentra en Amsterdam, en el cual se aloja el servidor SVN, por lo tanto, implicaría la no disponibilidad del código actualizado.
Prevención	Realizar copias de seguridad eventualmente en Dropbox.
Plan de contingencia	Importar la última versión exportada del código y hacer un control de cambios manual.
Probabilidad	Baja
Impacto	Alto

Cuadro 2.38: Caída del servidor

Pérdida de datos

Descripción	Pérdida de datos por avería en el equipo de desarrollo.
Prevención	Subir el código a Subversion habitualmente a una rama de desarrollo. Hacer copia de seguridad de la documentación en Dropbox.
Plan de contingencia	Recuperar la última versión exportada de los datos y hacer un control de cambios manual.
Probabilidad	Media
Impacto	Alto

Cuadro 2.39: Pérdida de datos

El cliente rechaza los requerimientos ofrecidos

Descripción	El cliente rechaza los requerimientos ofrecidos de la aplicación móvil
Prevención	Intentar hacer una buena campaña comercial del producto a MIT y evitar dedicar demasiado tiempo al desarrollo inicial del producto por si el cliente lo rechazase
Plan de contingencia	No es posible desarrollar un plan de contingencia para este riesgo ya que no habría pérdidas materiales
Probabilidad	Alta
Impacto	Muy bajo

Cuadro 2.40: El cliente rechaza los requerimientos ofrecidos

El cliente no paga la aplicación

Descripción	El cliente no paga la aplicación
Prevención	Firma de un contrato
Plan de contingencia	Interponer una denuncia mediante vías legales
Probabilidad	Bajo
Impacto	Muy alto

Cuadro 2.41: El cliente no paga la aplicación

Cambios en los requisitos funcionales de la aplicación

Descripción	Cambios en los requisitos funcionales por parte del cliente.
Prevención	Documentar los requisitos con el máximo detalle posible e intentar obtener la información adecuada planteándole al cliente las preguntas necesarias para cada una de las funcionalidades. Confirmar con el cliente, tras realizar el análisis del requisito funcional, la documentación realizada.
Plan de contingencia	Facturar los cambios producidos y volver a planificar la planificación inicial para adecuarse a los nuevos requerimientos.
Probabilidad	Muy alta
Impacto	Medio - Alto

Cuadro 2.42: Cambios en los requisitos funcionales de la aplicación

Comunicación ineficiente con el cliente

Descripción	No recibir contestación a las dudas planteadas a MIT.
Prevención	Repartir las tareas de modo eficiente, intentando hacer otras mientras se espera a la contestación de la duda.
Plan de contingencia	Cambiar, si es posible, a otra tarea la cual no tenga a ésta como precedente; evitando retrasos en la entrega.
Probabilidad	Muy alta
Impacto	Medio

Cuadro 2.43: Comunicación ineficiente con el cliente

Bajas por enfermedad

Descripción	Nula disponibilidad de alguno de los componentes del proyecto por enfermedad.
Prevención	Dividir las tareas de tal forma que se pueda avanzar en el proyecto, intentando que las tareas críticas sean desarrolladas por cualquiera de los dos componentes del equipo, si se goza de tiempo de holgura entre tareas.
Plan de contingencia	En caso de que la baja se alargue, será necesario volver a realizar la repartición de tareas para evitar retrasos en el proyecto y volver a realizar una planificación de la estimación temporal.
Probabilidad	Alta
Impacto	Medio

Cuadro 2.44: Bajas por enfermedad

Desastre natural en el lugar donde se aloja el servidor de integración

Descripción	Un desastre natural destruye físicamente el servidor que, actualmente, se encuentra en Amsterdam.[4]
Prevención	Realizar copias de seguridad eventualmente en Dropbox.
Plan de contingencia	Si el código se encuentra actualizado en uno de los equipos de desarrollo o en Dropbox, se deberá subir nuevamente a Subversion. En caso de que no lo esté, se debería utilizar la última versión disponible y hacer un control de cambios manual. La recuperación del servidor de integración es competencia de Digital Ocean.
Probabilidad	Muy baja
Impacto	Muy alto

Cuadro 2.45: Desastre natural en el lugar donde se aloja el servidor de integración

2.8. Evaluación económica

En esta sección se va a detallar el coste de la realización de este proyecto. Es necesario mencionar que el cliente únicamente ha solicitado una aplicación web y que la aplicación móvil se ha realizado con intención de crear una necesidad en el cliente que aún no tiene y negociar un futuro contrato de mantenimiento.

Al tratarse del primer proyecto real realizado, se ha pactado desde el inicio un precio cerrado (no un precio por hora, ni nada similar), que son 45.000 pesos mexicanos por la aplicación web (en el momento en el que se cerró el trato, la cantidad en euros era de 2.700€, aproximadamente).

Sin embargo, **la estimación de los gastos va a ser hipotética** (lo que le costaría realmente a una empresa el desarrollo de este proyecto), y en ello es en lo que se basará la viabilidad económica. En este caso, realmente no se ha gastado lo que indica la sección de Gastos, debido a que no se ha utilizado oficina, y que los desarrolladores se llevarían el dinero integro del proyecto; no habría que contratar programadores para la realización del mismo.

2.8.1. Ingresos

Como se ha indicado, por la aplicación web se han facturado 2.700€. Posteriormente, por la aplicación móvil se solicitarán 2.000€ más. En total, los ingresos ascenderían a 4.700€. Finalmente, tras el desarrollo de las aplicaciones web y móvil se tratará de pactar un contrato de mantenimiento por un precio de 48€/hora.

2.8.2. Gastos

Los gastos en los que se ha incurrido son, principalmente, en Mano de Obra, Hardware y Software. A continuación, en los apartados posteriores se detallan cada uno de ellos.

2.8.2.1. Mano de obra

Supongamos que el coste por hora de un analista/programador es de 20€/hora. En total, el proyecto dura 837 horas, por tanto, el coste total de la mano de obra es de:

- 837x20 = **16.740,00€**.

También supondremos que los desarrolladores (al ser freelance), abonarán ellos mismos los impuestos y gastos en los que se incurre al estar dado de alta como autónomo/empresa en una actividad profesional.

2.8.2.2. Hardware

- **Equipos informáticos**: Se utilizarán dos equipos informáticos para el desarrollo del proyecto. Cada equipo tiene un coste de 670,00€. Un equipo informático se

amortiza en 5 años, por lo que:

- (670 x 1)/5 = 134,00€, correspondientes a un año. Son dos equipos, por lo que el coste total es de **268,00€.**

➢ **Dispositivos móviles:** Se utilizará un Samsung Galaxy S3 y una tablet Samsung Galaxy tab 4 de 7". El móvil cuesta de segunda mano 185,00€ y la tablet vale 141,00€. No tendremos en cuenta la amortización debido a que son de segunda mano. Por tanto:

- 185 + 141 = **326,00€**

➢ **Servidor de pruebas:** Se adquirirá un servidor cloud a través de la empresa Digital Ocean. Tiene un precio de 3,99€/mes, por tanto:

- 3,99 x 12 = **47,88€**

➢ **Dominio** ekatam.net: Se adquirirá un dominio para poder tener una cuenta de correo corporativa y además, que cuando MIT tenga que realizar pruebas, pueda hacerlo accediendo a través del dominio, sin tener que recordar la IP. El dominio tiene un coste de 23,91€ al año.

- **23,91€**

2.8.2.3. Software

A continuación se detallará una parte del software utilizado, ya que el resto que se ha utilizado es de código abierto por lo que no ha supuesto ningún gasto.

➢ **Microsoft Project Professional:** Como va a ser utilizada conjuntamente con Office 365, Microsoft proporciona una "extensión" del paquete Office 365 con Project Professional por un precio de 44,50€ por usuario y por mes. Supondremos que de los dos trabajadores, sólo habrá uno que lo utilice para planificar el proyecto, por lo tanto, el precio será de:

- 44,50 x 12 = **534,00€**

➢ **Microsoft Visio Professional:** También será utilizado conjuntamente con Office 365. Por lo tanto, el coste de la extensión que Microsoft proporciona es de:

- 10,00 x 12 = **120,00€**

2.8.2.4. Otros gastos

➢ **Luz**: El coste de luz con Iberdrola es de 0,12€/kWh.

- Un ordenador consume una media de 0,4kWh. La duración del proyecto es de 837h, por lo que: 837 x (0,4 x 0,12) = **40,18€** consumirá.
- Además, hay que sumar el coste de luz y otros equipos instalados en la oficina (máquinas de agua, cafetera, nevera, televisión, cargador de móvil...), lo que puede hacer que la factura de luz ascienda a unos 35€ al mes. Por tanto: 35 x 12 = **420,00€** de luz se consumirá al año en otros aparatos que consuman electricidad.

➢ **Oficina:** El coste de una oficina en Bilbao puede ser cercano a 500,00€ al mes, dependiendo de la zona. Por tanto:

- 500 x 12 = **6.000€**

2.8.3. Conclusiones de la viabilidad económica

Como se puede observar en el cuadro 2.46, el desarrollo del proyecto **no es viable económicamente**, ya que las pérdidas ascienden a **-19.990,99€**. Sin embargo, como se ha indicado anteriormente, este caso es totalmente hipotético ya que no se ha necesitado oficina, ni muchas licencias de software (puesto que se disponían ya de algunas), ni pagar programadores, es decir, el gasto real para el desarrollo del proyecto en este caso será solo el del hardware (y no completamente, ya que de ordenadores, por ejemplo, ya se disponía).

Ingresos totales	4.700€
Gastos totales	24.036,99€
TOTAL	**-19.336,99€**

Cuadro 2.46: Tabla de viabilidad económica

Capítulo 3

Antecedentes

En el capítulo actual se describirán las tecnologías a utilizar, así como las aplicaciones que actualmente hay funcionando similares a la que se ha desarrollado.

3.1. Situación actual y alternativas

La novedad de esta aplicación reside en que se utilizarán las tecnologías de pago solicitadas por MIT. Además, el objetivo de la aplicación es el fomento del uso de este tipo de aplicaciones en México, un país en el cual, hasta hace un año, no estaba tan normalizado el uso de este tipo de aplicaciones.

El estudio de mercado lo ha realizado MIT, con lo cual desconocemos en qué casos se han fijado para determinar la viabilidad del desarrollo de esta aplicación, sin embargo, las aplicaciones existentes que más similitud tienen con Easy Order y que se han estudiado son:

- **Sin delantal:** De todas las alternativas que indicamos, es de las pocas que opera en México, con lo cual, sería el competidor directo de Easy Order. Además, en algunas de las solicitudes realizadas por MIT, se ha indicado como ejemplo esta aplicación. La diferencia principal de Easy Order frente a Sin Delantal, es la posibilidad de pagar con los medios de pago ofrecidos por MIT. MIT asegura que sus medios de pago son muy utilizados en México, con lo cual, si esto fuera cierto, es posible que atraigan un buen volumen de público.
  Para la búsqueda de restaurantes en Sin Delantal es necesario indicar varios datos (la ciudad, la calle, el número y la colonia), en Easy Order es suficiente con el código postal. Además, ofrece ayuda en línea y pequeños tutoriales de cómo funciona la página, cosa que Easy Order no hace. Se le podría ofecer al cliente como requerimiento adicional.

Imagen extraída de `http://sindelantal.mx/`

Figura 3.1: Captura de *Sin Delantal*

- **Grubhub:** Se podría indicar que esta aplicación no es un competidor directo de Easy Order debido que sólo opera en EEUU. Sin embargo, debido a la cercanía con México quizá en un futuro próximo decidiese operar ahí también, con lo cual, sería un competidor potente de Easy Order.
  Para la búsqueda de restaurantes es necesario indicar la dirección completa (calle, ciudad y estado) y además indicar una palabra clave para filtrar los restaurantes. Tambien ofrece ayuda en línea y pequeños tutoriales de cómo realizar pedidos.

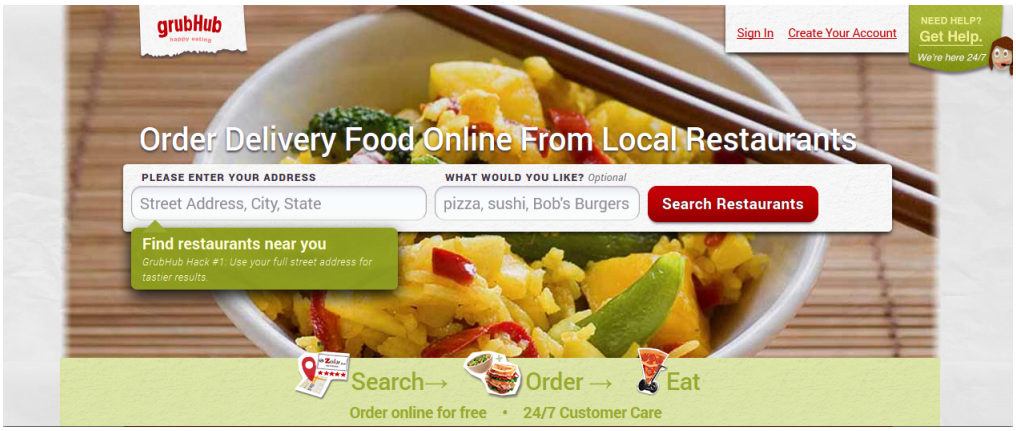


Imagen extraída de `https://www.grubhub.com/`

Figura 3.2: Captura de *GrubHub*

- **Just-eat:** Momentáneamente no opera en México, pero si lo hiciese, sería uno de los principales competidores debido a su extensa fama en diferentes países.
  Para la búqueda de restaurantes únicamente es necesario introducir un código postal y un tipo de comida. Además, tiene una entrada de menú exclusiva para ayudas donde se podrán consultar diferentes tutoriales o bien, consultar cualquier duda a un asistente en línea.

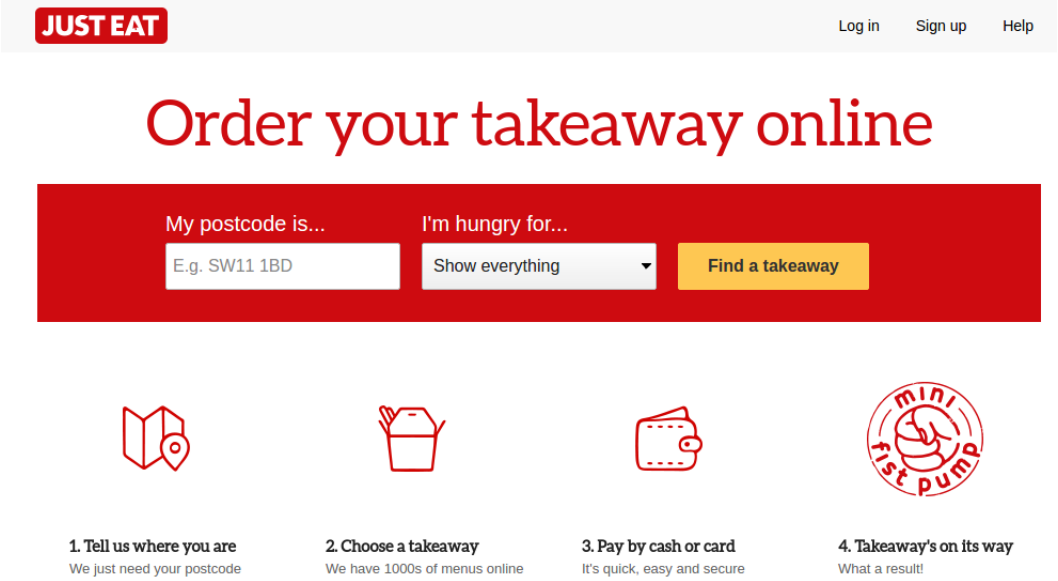


Imagen extraída de `http://www.just-eat.co.uk/`

Figura 3.3: Captura de *Just Eat*

- **La nevera roja:** Es una aplicación que sólo opera en España y hasta hace pocos meses, no era demasiado conocida. Sin embargo, recientemente ha desplegado una fuerte campaña publicitaria en TV, internet y otros medios, lo que implica que es posible que comience a tener más fuerza en España. No parece, de momento, muy probable su expansión a otros países.
  Para la búsqueda de restaurantes es necesario dar una ubicación más exacta que la solicitada en Easy Order, incluyendo la ciudad, la calle y el número. Además, al igual que en el resto de alternativas, ofrece ayuda en línea, con la diferencia de que tiene muy presente las redes sociales.

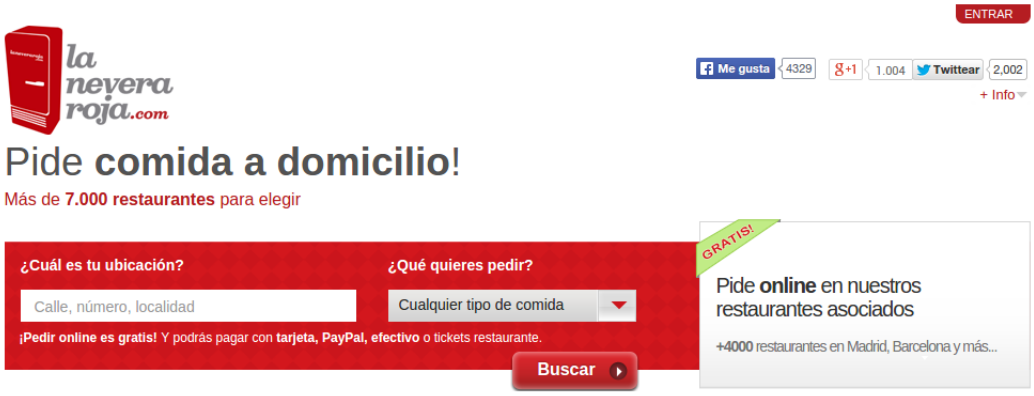


Imagen extraída de `http://www.laneveraroja.com/`

Figura 3.4: Captura de *La Nevera Roja*

- **Otros**: Hay otras aplicaciones que operan en México (Hellofood, Se Me Antoja, Pedidosya...), sin embargo, MIT no las ha tenido en cuenta a la hora de solicitar funcionalidades similares. En cualquier caso, como se ha indicado anteriormente, la viabilidad de la aplicación la ha estudiado MIT.

3.2. Tecnologías a utilizar

En este apartado se detallarán las diferentes tecnologías a utilizar, subdivididas en dos partes. El primero, hablará sobre las tecnologías utilizadas en la aplicación web mientras que el segundo, hablará sobre las tecnologías utilizadas en el desarrollo de la aplicación móvil.

3.2.1. Desarrollo web

Se conoce como desarrollo web a la creación de páginas web. En este caso, nos ayudaremos de tecnologías tales como HTML5, CSS3 y Javascript para el lado del cliente, y de PHP para el lado del servidor.

Además, se utilizará MySQL como SGBD. Uno de los motivos de está elección es su gratuidad, por lo que no repercute económicamente en el cliente. Además, es un SGBD que conocemos.

HTML5 es la versión 5 del estándar HTML *(HyperText Markup Language)*, el cual ha sido aprobado por el consorcio World Wide Web (W3C). El estándar HTML es un lenguaje de marcado para organizar y crear la estructura básica de páginas web.

La estructura de los documentos en HTML5, puede verse en la Figura 3.5. De igual modo, en la Figura 3.6 puede verse detallada la estructura.

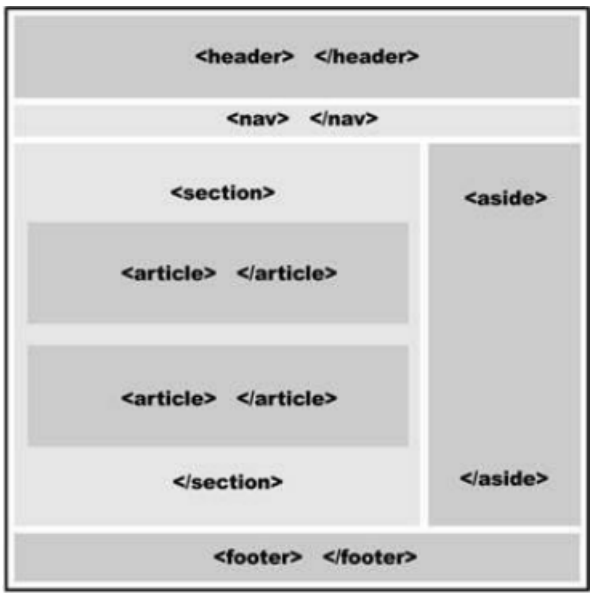


Figura 3.5: Estructura HTML5

- ≻ *<!DOCTYPE>*: Es necesario indicar el tipo de documento que estamos creando. Además, esta debe ser la primera línea del texto HTML.
- ≻ *<html>*: Después de declarar el tipo de documento, debemos comenzar a construir la estructura HTML. La cual englobará a las etiquetas posteriores. En esta etiquete es obligatorio especificar el idioma mediante el atributo lang.
  - *<head>*: Cabecera del documento.
    - *<meta>*: Define el juego de caracteres a utilizar para mostrar el documento.
    - *<title>*: Especifica el título del documento.
    - *<link>*: Permite incorporar estilos, código Javascript, imágenes o iconos desde archivos externos.
  - *<body>*: Cuerpo del documento.
    - *<header>*: Permite introducir información introductoria, como lo son títulos, subtítulos, logos... Además puede ser aplicado en diferentes secciones del documento.
    - *<nav>*: Permite introducir los elementos de la barra de navegación, indicando una sección de enlaces con propósitos de navegación.
    - *<section>*: Permite introducir secciones. Normalmente es utilizado para construir varios bloques de contenido.
      - ◇ *<article>*: Permite dividir las secciones en apartados.
    - *<aside>*: Permite introducir los elementos de la barra lateral. Estos elementos, suelen contener información relacionada con la principal pero que no es igual de relevante.
    - *<footer>*: Permite introducir los elementos pertenecientes al pie de página.

Figura 3.6: Estructura detallada HTML5

CSS es un lenguaje utilizado para aplicar estilos a un documento que proviene de las siglas "Cascading Style Sheets". La especificación de **CSS3** está presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento. Por ejemplo, mediante sus propiedades permite crear efectos visuales y dinámicos.

Oficialmente CSS nada tiene que ver con HTML5 ya que CSS no forma parte de la especificación. Este lenguaje es un complemento desarrollado para superar las limitaciones y reducir la complejidad de HTML.

En cambio, la especificación de HTML5 ha sido desarrollada considerando CSS a

cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web. Esta es la razón por la que cada vez que se habla de HTML5 también se hace referencia a CSS3 de algún modo, aunque oficialmente se trate de dos tecnologías completamente separadas.

Javascript es un lenguaje de programación interpretado alojado en el lado del cliente. Este lenguaje permite interactuar con los elementos del documento HTML y gestionar eventos. Además, tiene una librería gratuita muy amplia llamada jQuery, la cual es usada para el desarrollo de web dinámicas debido a que facilita diversas tareas que se efectúan con JavaScript puro.

AJAX (*Asynchronous JavaScript + XML*) es una tecnología utilizada para el intercambio de datos con el servidor de manera asíncrona.

JSON (*JavaScript Object Notation*) es una sintaxis que permite almacenar y transportar datos. Para formatear estos datos se ayuda de un array asociativo. Suele utilizarse cuando se quieren enviar datos entre una aplicación web y un servidor.

PHP es un lenguaje de programación gratuito del lado del servidor, y una potente herramienta para realizar páginas web dinámicas e interactivas.

3.2.2. Android

Android es un sistema operativo de código abierto basado en Linux para dispositivos móviles creado por Google. Actualmente, se utiliza también en televisiones, PCs...

La arquitectura de Android se divide en 4 capas, como se puede observar en la siguiente imagen:

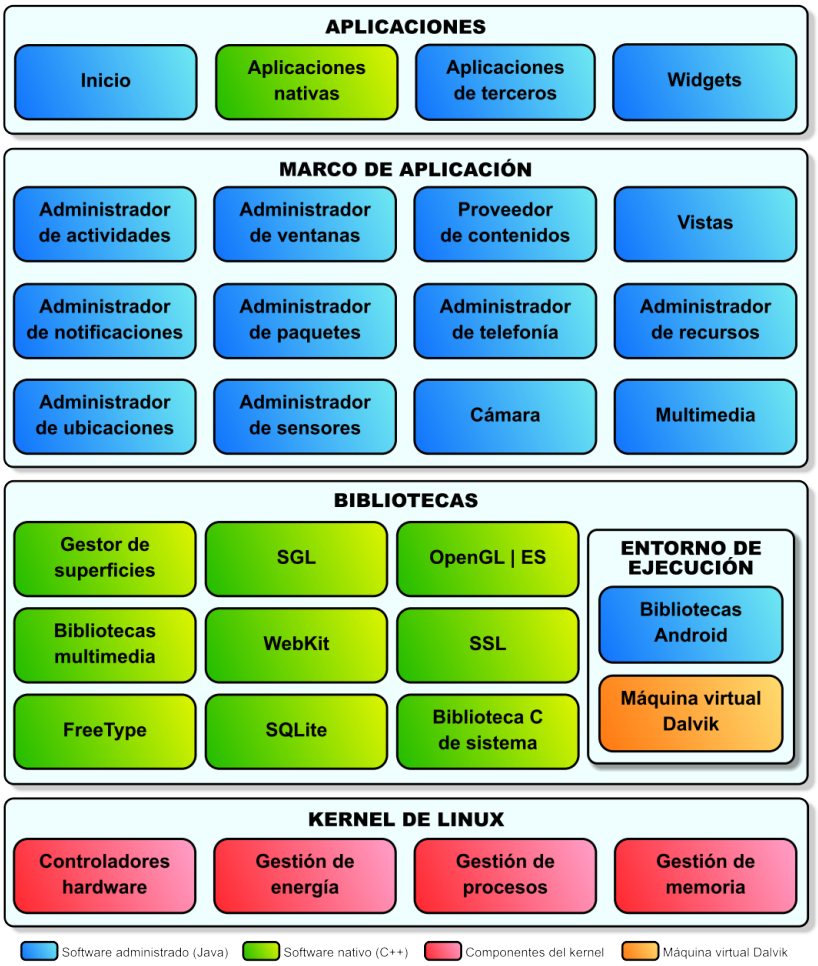


Imagen extraída de
http://columna80.wordpress.com/2011/02/17/arquitectura-de-android/

Figura 3.7: Arquitectura Android

Android está construido utilizando un **kernel de Linux**. Este kernel, o núcleo, se utiliza para gestionar la memoria, los procesos, la energía y controlar el hardware. Facilitando así una capa de abstracción a los elementos de hardware de tal forma que una aplicación en Android (el software) puede interactuar con los elementos de hardware sin necesidad de conocer ningún detalle sobre estos.

Por ejemplo, si una aplicación necesita el GPS, podrá utilizarlo sin necesidad de conocer el modelo ni la versión, ya que el propio kernel contiene todos los controladores (drivers) del hardware.

Las **bibliotecas** nativas, o librerías, escritas en C o C++, están compiladas para

la arquitectura de hardware y preinstaladas en el dispositivo móvil preparadas para ser utilizadas por los componentes del sistema. Las librerías que se incluyen son las siguientes:

- Gestor de superficies *(Surface Manager)*: Gestión del acceso a la representación gráfica en 2D y 3D.
- Bibliotecas multimedia: Permiten la reproducción multimedia, soportando codecs de reproducción y grabación de multitud de formatos de audio, vídeo e imágenes.
- FreeType: Fuentes bitmap y renderizado vectorial.
- SGL *(Scalable Graphics Library)*: motor de gráficos 2D.
- WebKit: Navegador optimizado.
- SQLite: Motor de base de datos.
- OpenGL | ES *(OpenGL for Embedded Systems)*: Motor de gráficos 2D.
- SSL *(Secure Sockets Layer)*: Proporciona seguridad.
- Biblioteca C del sistema: Proporciona la funcionalidad básica para la ejecución de las aplicaciones.

El entorno de ejecución, no se considera una capa en sí mismo ya que también está formado por bibliotecas. Entre estas se encuentran librerías habituales de Java y específicas de Android. El componente principal del entorno de ejecución, es la maquina virtual Dalvik[1].

El **marco de aplicación** contiene el conjunto de herramientas necesarias para el desarrollo de aplicaciones. Las partes más importantes son las siguientes:

- Administrador de actividades *(Activity manager)*: Controla el ciclo de vida de las aplicaciones.
- Proveedor de contenidos *(Content providers)*: Encapsula los datos necesarios que se comparten entre las aplicaciones. Por ejemplo, los contactos.
- Administrador de recursos *(Resource Manager)*: Controla los elementos que no son "código". Por ejemplo, los sonidos.
- Administrador de ubicaciones *(Location manager)*: Controla el acceso a la ubicación.
- Administrador de notificaciones *(Notification manager)*: Controla las notificaciones, alertas...

[1] Dadas las limitaciones de memoria y procesador de los dispositivos móviles, Google tomó la decisión de crear una nueva máquina virtual que sustituyese a la máquina virtual de Java estándar.

La capa de las **aplicaciones** contiene todas las aplicaciones que interactúan con el usuario. Es decir, las aplicaciones nativas (vienen por defecto en el dispositivo), las aplicaciones instaladas por el usuario, los widget añadidos por el usuario y la pantalla principal (launcher).

Estas aplicaciones tienen un formato .apk *(Application PacKage file)*. Las aplicaciones se pueden instalar directamente desde la tienda oficial de Android (Play Store) o bien, configurando el dispositivo para que permita instalar aplicaciones de terceros.

Android podría dividirse en los siguientes elementos básicos:

- **Activity**: Es el componente encargado de mostrar al usuario la interfaz gráfica que permite la interacción de los usuarios. Para ello, se le asigna una interfaz gráfica.
- **Vista (layout)**: Es el componente encargado de generar la interfaz gráfica. Está diseñada en XML.
- **Intent**: Es el mecanismo que sirve para invocar componentes. Estos componentes pueden ser aplicaciones externas, otras interfaces de usuario.
- **Adaptador**: Es el responsable de generar a partir de datos unas vistas específicas. Por ejemplo, si tenemos un menú con cuatro platos y una vista definida con el diseño que se quiere utilizar para mostrar el plato, el adaptador será el encargado de generar en otra vista los cuatro platos con el mismo formato.
- **Android Manifest**: Es uno de los archivos más importantes de cualquier aplicación Android. Se genera automáticamente al crear el proyecto, y en él se encuentra definida la configuración del proyecto en XML (Activitys, Intents, permisos de la aplicación...).

En Android, las actividades tienen un ciclo de vida, es decir, pasan por diferentes estados desde que se inician hasta que se destruyen. Este ciclo de vida puede observarse en la Figura 3.8.

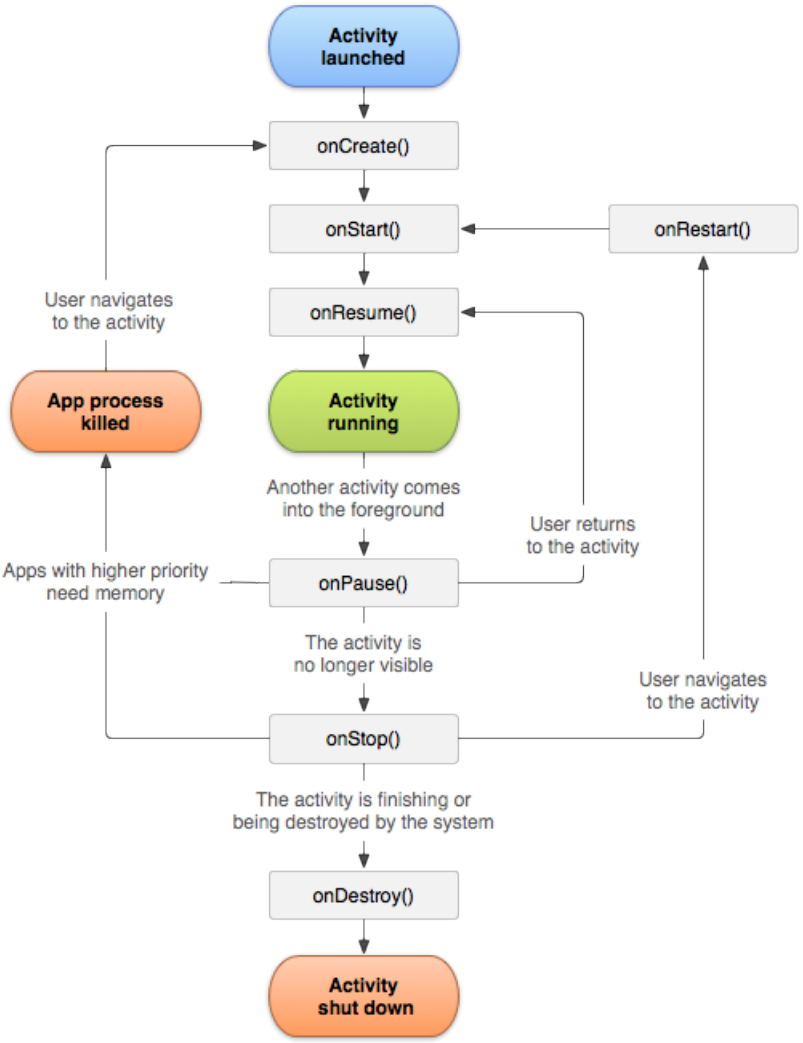


Imagen extraída de
http://developer.android.com/reference/android/app/Activity.html

Figura 3.8: Ciclo de vida Android

Cualquier actividad hereda varios métodos al extender de la clase Activity. Entre ellos están:

- ***onCreate****(Bundle savedInstanceState)*: Este método se ejecuta al crear la actividad. El parámetro de entrada puede ser null o la información del estado anterior guardada mediante el método *onSaveInstanceState*(). En el *onCreate*, normalmente se le asigna la vista mediante el método *setContentView(vista)*.
- ***onStart****()*: Este método indica si la actividad se muestra al usuario.

- ***onResume()***: Se llama a este método cuando la actividad puede empezar a interactuar con el usuario.
- ***onPause()***: Se ejecuta este método cuando la actividad pasa a un segundo plano. Si hay problemas de memoria, puede que no se llame a este método debido a que el sistema puede terminar el proceso directamente.
- ***onStop()***: Se llama a este método cuando la actividad ya no es visible para el usuario. Si hay problemas de memoria, puede que no se llame a este método debido a que el sistema puede terminar el proceso directamente.
- ***onRestart()***: Si se llama a este método indica que se va a volver a cargar la actividad. Su estado anterior es *stop*.
- ***onDestroy()***: Se llama a este método antes de destruirse la actividad.
- ***onSaveInstanceState****(Bundle)*: Android llamará a este método para permitir a una actividad guardar su estado.
- ***onRestoreInstanceState****(Bundle)*: Se llama a esta actividad cuando va a reinicializarse habiendo guardado su estado previamente mediante *onSaveInstanceState()*.

Capítulo 4

Captura de requisitos

En el capítulo actual se describirá que deben cumplir las aplicaciones desarrolladas. Para ello, se detallarán los modelos de casos de uso, la jerarquía de actores de ambas aplicaciones y el modelo de dominio con sus respectivas explicaciones.

4.1. Web

En este apartado se mostrará el modelo de casos de uso y la jerarquía de actores de la aplicación web.

4.1.1. Modelo de casos de uso

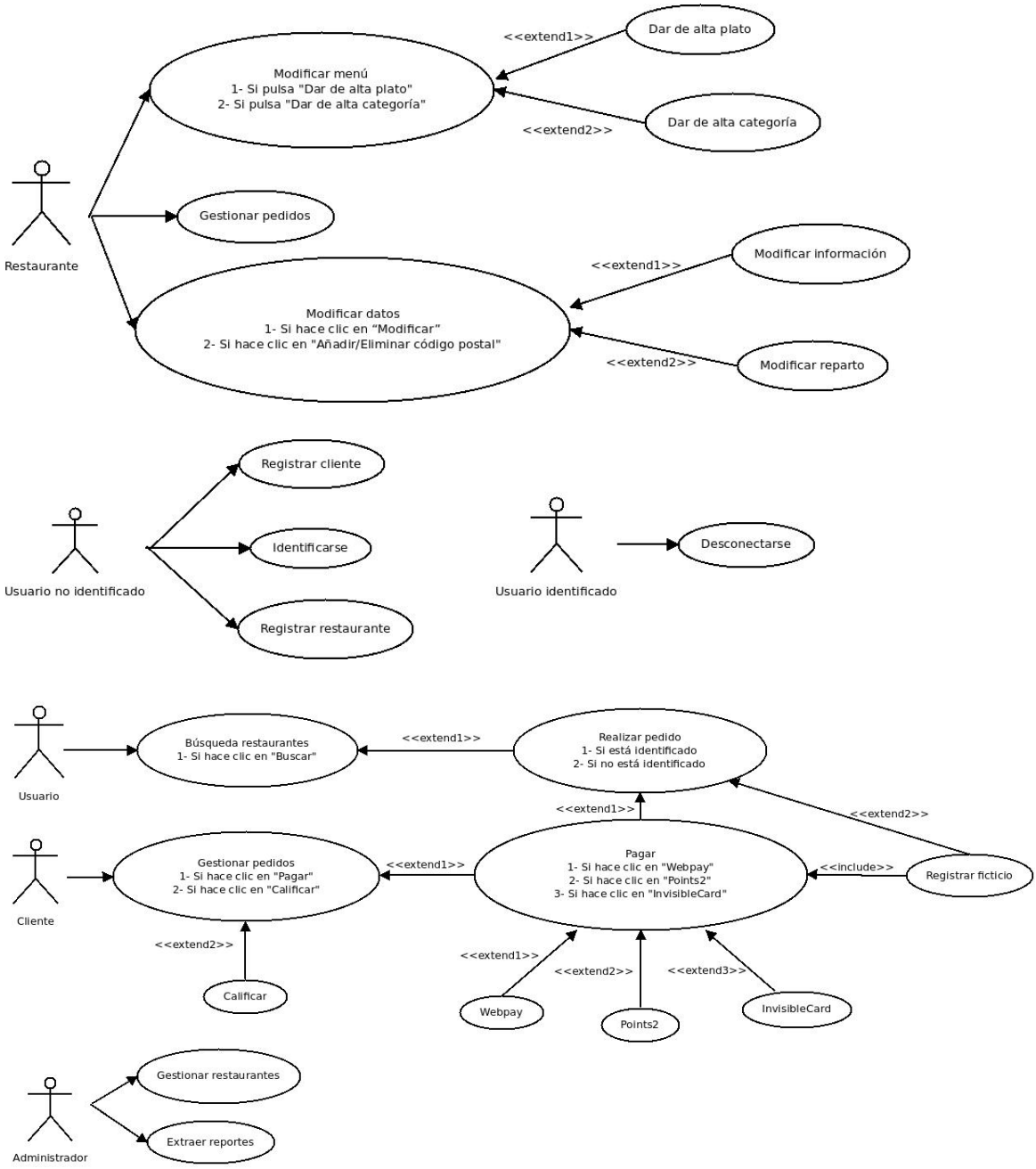


Figura 4.1: Diagrama CU Web

A continuación, se explicarán los casos de uso de cada actor:

Usuario no identificado

- ≻ Identificarse: Un usuario no identificado podrá identificarse en el sistema.
- ≻ Registrar cliente: Un usuario no identificado podrá registrarse como cliente para poder hacer un seguimiento de sus pedidos.
- ≻ Registrar restaurante: Un usuario no identificado podrá registrarse como restaurante para poder ofrecer sus productos a través de la aplicación.

Usuario

- ≻ Búsqueda restaurantes: Un usuario podrá buscar restaurantes para poder realizar un pedido.
  - Realizar pedido: Un usuario podrá realizar un pedido seleccionando platos de un restaurante.
    - Registro ficticio: Un usuario sin identificarse podrá dar sus datos para poder pagar el pedido.
  - Pagar: Un usuario podrá pagar el pedido realizado.
    - Webpay: Un usuario podrá pagar el pedido con *Webpay*.
    - Points2: Un usuario podrá pagar el pedido con *Points2*.
    - InvisibleCard: Un usuario podrá pagar el pedido con *InvisibleCard*.

Cliente

- ≻ Gestionar pedidos: Un cliente podrá gestionar pedidos realizados.
  - Calificar: Un cliente podrá calificar los pedidos realizados.
  - Pagar: Un usuario podrá pagar el pedido realizado.
    - Webpay: Un usuario podrá pagar el pedido con *Webpay*.
    - Points2: Un usuario podrá pagar el pedido con *Points2*.
    - InvisibleCard: Un usuario podrá pagar el pedido con *InvisibleCard*.

Restaurante

- ≻ Modificar menú: Un restaurante podrá modificar su menú pudiendo dar de alta/baja categorías y platos.
  - Dar de alta plato: Un restaurante podrá dar de alta nuevos platos en su menú bajo una categoría.
  - Dar de alta categoría: Un restaurante podrá dar de alta nuevas categorías en su menú.

- ≻ Modificar datos: Un restaurante podrá modificar su información pudiendo cambiar sus datos y gestionar los códigos postales en los que repartir.
  - Modificar información: Un restaurante podrá modificar su información.
  - Modificar reparto: Un restaurante podrá añadir/eliminar códigos postales de reparto.
- ≻ Gestionar pedidos: Un restaurante podrá cambiar el estado de los pedidos.

Usuario identificado

- ≻ Desconectarse: Un usuario identificado podrá desconectarse del sistema.

Administrador

- ≻ Gestionar restaurantes: El administrador podrá dar de alta/baja restaurantes.
- ≻ Extraer reportes: El administrador podrá extraer reportes sobre los restaurantes.

4.1.2. Jerarquía de actores

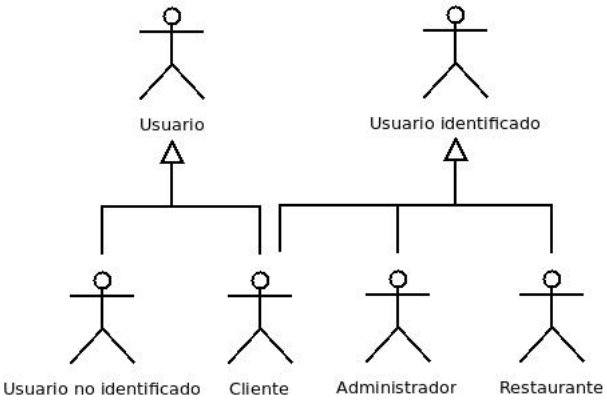


Figura 4.2: Jerarquía de actores Web

- ≻ Usuario: Es aquel que usa la aplicación.
- ≻ Usuario identificado: Es aquel que se ha identificado en el sistema.
- ≻ Usuario no identificado: Es aquel que usa la aplicación sin identificarse en el sistema.
- ≻ Cliente: Es aquel que se ha identificado en el sistema como cliente.
- ≻ Administrador: Es aquel que se ha identificado en el sistema como administrador.
- ≻ Restaurante: Es aquel que se ha identificado en el sistema como restaurante.

4.2. Android

En este apartado se mostrará el modelo de casos de uso y la jerarquía de actores de la aplicación Android.

4.2.1. Modelo de casos de uso

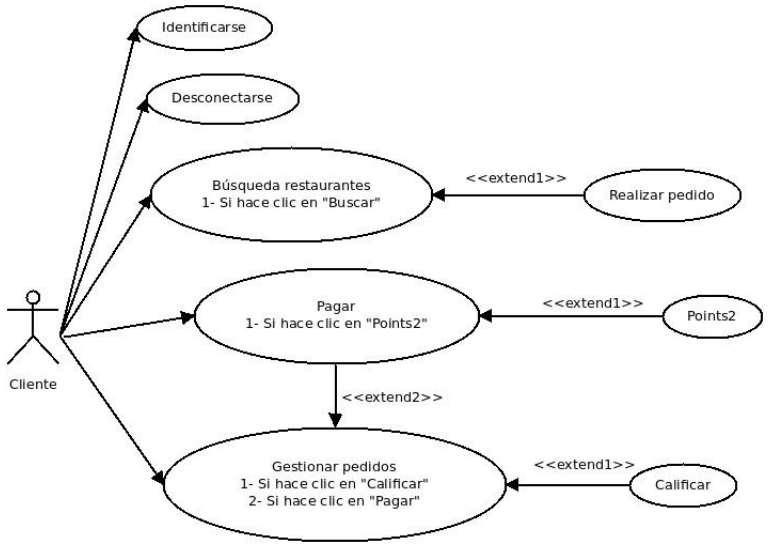


Figura 4.3: Diagrama CU Android

A continuación, se explicarán los casos de uso de cada actor:

Cliente

- Identificarse: Un cliente podrá identificarse en el sistema.
- Búsqueda restaurantes: Un cliente podrá buscar restaurantes para poder realizar un pedido.
  - Realizar pedido: Un cliente podrá realizar un pedido seleccionando platos de un restaurante.
- Pagar: Un cliente podrá pagar el pedido realizado.
  - Points2: Un cliente podrá pagar el pedido con *Points2*.
- Desconectarse: Un cliente podrá desconectarse del sistema.
- Gestionar pedidos: Un cliente podrá gestionar los pedidos realizados.

- Calificar: Un cliente podrá calificar los pedidos realizados.
- Pagar: Un cliente podrá pagar el pedido realizado.
    - Points2: Un cliente podrá pagar el pedido con *Points2*.

4.2.2. Jerarquía de actores

Figura 4.4: Jerarquía de actores Android

➢ Cliente: Es aquel que usa la aplicación.

4.3. Modelo de dominio

Tanto la aplicación móvil como la aplicación web usan el mismo modelo de dominio, ya que los datos que utiliza la aplicación móvil deben ser los mismos que los de aplicación web. Por lo que a continuación se mostrará el modelo de dominio de ambas aplicaciones:

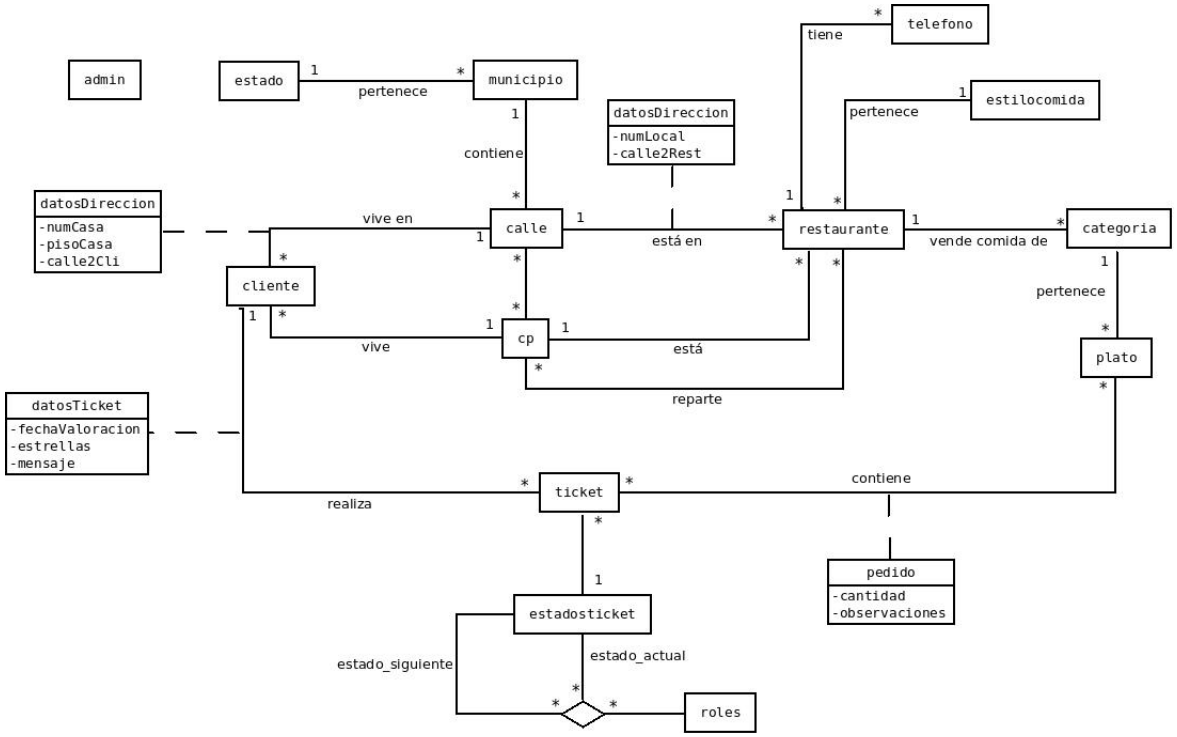


Figura 4.5: Modelo de dominio

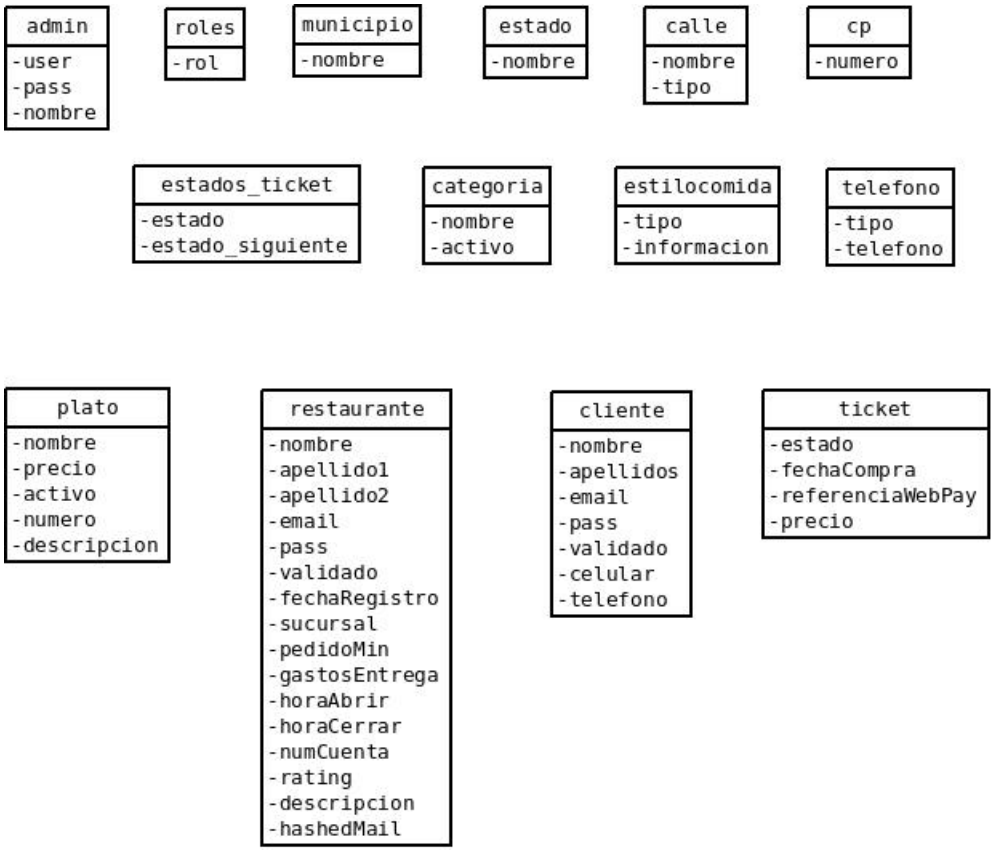


Figura 4.6: Entidades modelo de dominio

4.3.1. Entidades

admin Entidad compuesta por el nombre del administrador, y el usuario y contraseña de identificación.

estado Entidad compuesta por el nombre del Estado.

- ≻ Un Estado puede contener varios municipios.

municipio Entidad compuesta por el nombre del municipio.

- ≻ Un municipio pertenece a un Estado.
- ≻ Un municipio puede contener varias calles.

calle Entidad compuesta por el nombre y el tipo de la calle.

- ≻ Una calle pertenece a un municipio.

- ≻ En una calle pueden vivir varios clientes. Complementando esta información con otros datos de la dirección solicitados en el registro del cliente.
- ≻ En una calle pueden estar varios restaurantes. Complementando esta información con otros datos de la dirección solicitados en el registro del restaurante.
- ≻ Una calle puede contener varios códigos postales.

CLIENTE Entidad compuesta por los datos solicitados en el registro del cliente y la confirmación de si este está validado.

- ≻ Un cliente vive en una calle. Complementando esta información con otros datos de la dirección solicitados en el registro.
- ≻ Un cliente vive en un código postal.
- ≻ Un cliente puede tener muchos tickets. Complementando esta información con otros datos del ticket obtenidos a través de la aplicación.

CP Entidad compuesta por el número de código postal.

- ≻ Un código postal puede pertenecer a varias calles.
- ≻ En un código postal pueden vivir varios clientes.
- ≻ En un código postal pueden estar varios restaurantes.
- ≻ En un código postal pueden repartir varios restaurantes.

RESTAURANTE Entidad compuesta por los datos solicitados en el registro del restaurante y la confirmación de si este está validado.

- ≻ Un restaurante está en una calle. Complementando esta información con otros datos de la dirección solicitados en el registro.
- ≻ Un restaurante está en un código postal.
- ≻ Un restaurante reparte en varios códigos postales.
- ≻ Un restaurante puede tener varios teléfonos.
- ≻ Un restaurante ofrece un tipo de comida.
- ≻ Un restaurante vende comida de varias categorías.

TELÉFONO Entidad compuesta por el tipo y el número de teléfono.

- ≻ Un teléfono pertenece a un restaurante.

ESTILOCOMIDA Entidad compuesta por el tipo y la información del estilo de comida.

- ≻ Un estilo de comida puede pertenece a varios restaurantes.

CATEGORÍA Entidad compuesta por el nombre de la categoría y si está activa.

- ≻ Una categoría pertenece a un restaurante.
- ≻ Una categoría puede tener varios platos.

PLATO Entidad compuesta por el nombre, el precio, el número en la carta, la descripción del plato y si está activo.

- ≻ Un plato pertenece a una categoría.
- ≻ Un plato pertenece a varios tickets. Complementando esta información la cantidad de platos y comentarios sobre estos.

TICKET Entidad compuesta por el estado del ticket, el precio, la fecha de compra y una referencia interna para pagos con WebPay.

- ≻ Un ticket pertenece a un cliente. Complementando esta información con otros datos del ticket obtenidos a través de la aplicación.
- ≻ Un ticket contiene varios platos. Complementando esta información la cantidad de platos y comentarios sobre estos.

ESTADOSTICKET Entidad compuesta por el estado actual y siguiente.

- ≻ Un estado de ticket puede pertenecer a varios roles.
- ≻ Un estado de ticket actual puede optar a varios estados de ticket siguientes.

ROLES Entidad compuesta por el nombre del rol.

- ≻ Un rol pueden tenerlo varios estados de ticket.

Capítulo 5

Análisis y diseño

En el capítulo actual se describirá cómo se plantea el desarrollo del proyecto. Para ello, se mostrará el diagrama relacional de la base de datos y los diagramas de clases de ambas aplicaciones.

5.1. Base de datos

La base de datos es común a ambas aplicaciones. Por lo tanto, a continuación se mostrará el diagrama ER y los triggers utilizados en ambas aplicaciones.

5.1.1. Diagrama ER

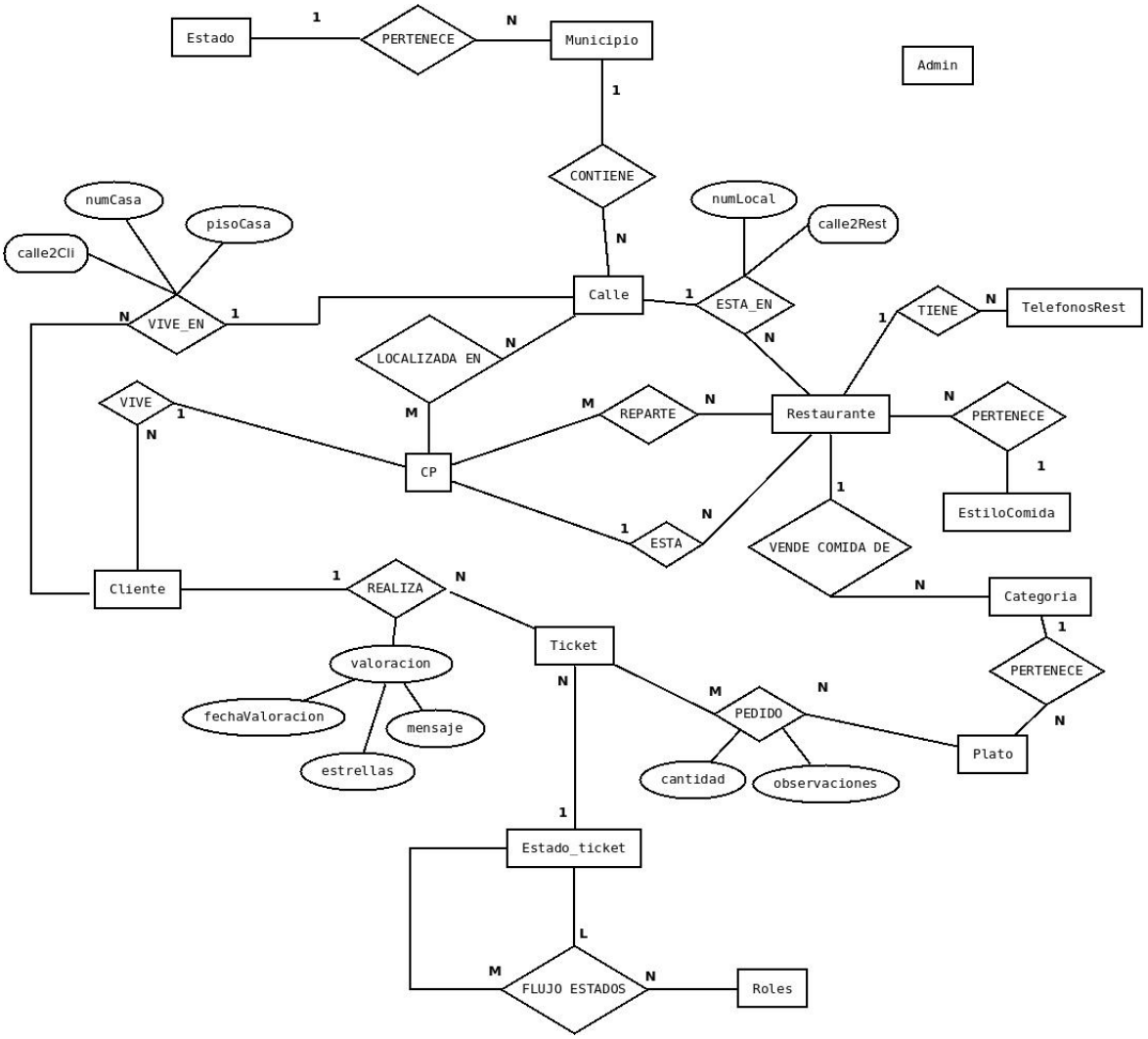


Figura 5.1: Diagrama ER

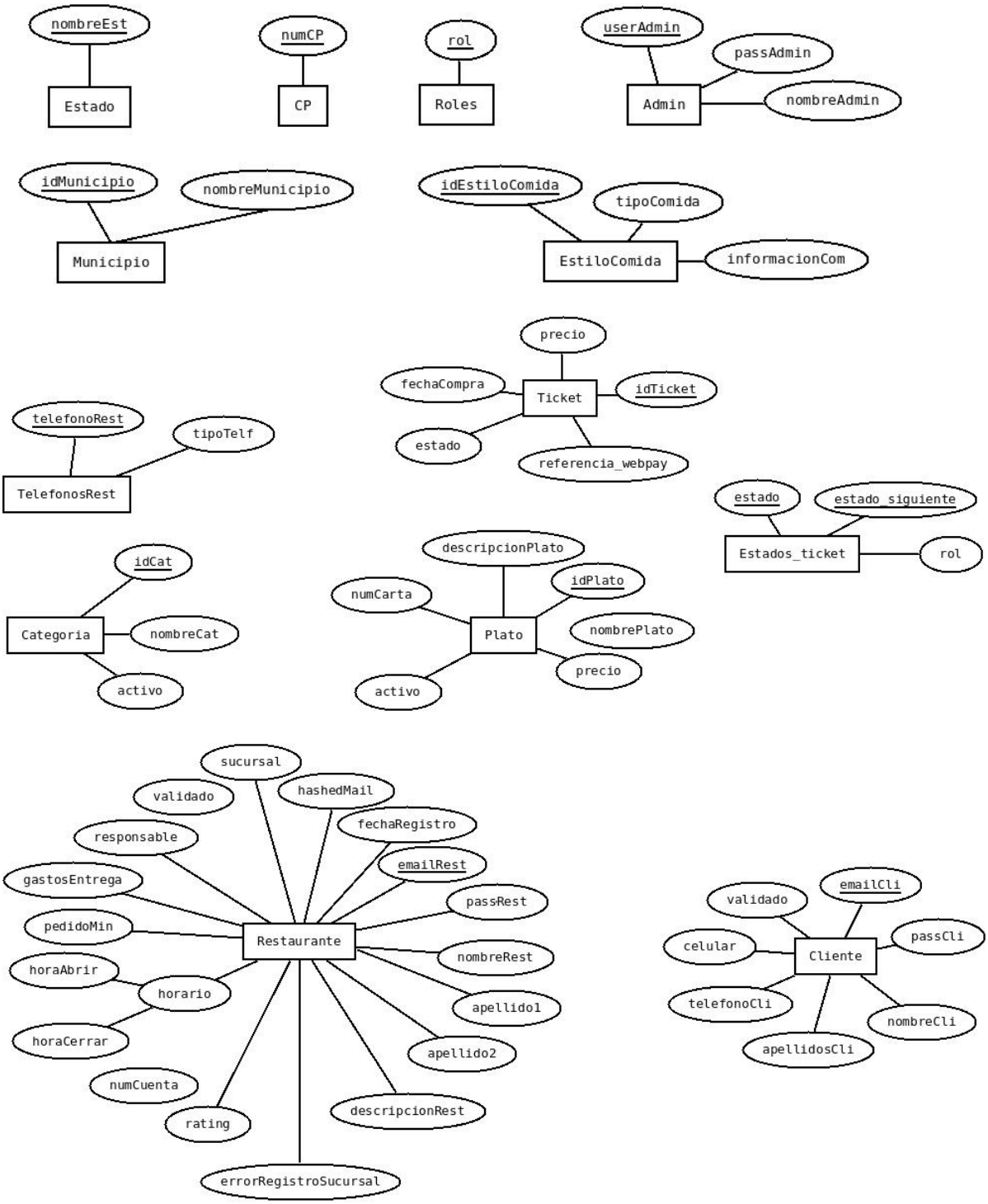


Figura 5.2: Entidades diagrama ER

5.1.2. Triggers

Se ha creado un trigger en la tabla *Ticket* que calcula la puntuación que tienen los restaurantes en función del número de estrellas con las que los clientes han valorado los pedidos, haciendo la media aritmética con las valoraciones que los clientes han hecho en todos los pedidos que tengan alguna valoración.

```
6   BEGIN
7
8   IF NEW.estrellas <> OLD.estrellas THEN
9
10  UPDATE restaurante SET rating =
11  (SELECT SUM(estrellas) /
12  (SELECT COUNT(emailRest)
13  FROM ticket natural join pedido natural join plato
14  WHERE emailRest = (SELECT distinct emailRest FROM ticket natural join pedido natural join plato
    where ticket.idTicket = OLD.idTicket limit 1) AND estrellas IS NOT NULL)
15  FROM ticket natural join pedido natural join plato WHERE emailRest =
16  (SELECT distinct emailRest FROM ticket natural join pedido natural join plato
17  where ticket.idTicket = OLD.idTicket limit 1))
18
19  WHERE emailRest = (SELECT DISTINCT emailRest FROM ticket NATURAL JOIN pedido NATURAL JOIN plato
    WHERE ticket.idTicket = OLD.idTicket);
20
21  END IF;
22
23  END
```

Figura 5.3: Trigger Ticket

5.2. Diagrama de clases

A continuación, se mostrarán los diagramas de clases de ambas aplicaciones.

5.2.1. Aplicación web

El diagrama de clases de la figura 5.4 muestra la estructura de clases desarrollada en PHP para el proyecto. Como se puede observar, las funcionalidades de la aplicación usan el *framework* para realizar tareas comunes, como lo pueden ser la conexión con Base de Datos, envío de correos electrónicos, etcétera.

El framework es independiente de las funcionalidades y podría ser utilizado en otros proyectos y aplicaciones web.

A continuación se explicará en que consiste cada paquete o carpeta del diagrama de clases de la figura 5.4:

- *Easyorder*: Contiene cada una de las funcionalidades asociadas a la aplicación. Realmente, la mayoría de los ficheros PHP no son clases, sino ficheros que interactúan con la vista mediante Ajax y con el framework para recuperar datos u otras tareas.

Además, también contiene un directorio el cual será el encargado de comunicar la aplicación móvil con el resto de funcionalidades.

- *Framework*: Contiene las clases de un framework propio, que se ha decidido denominar "ekatam".

  - *communication*: Contiene las clases necesarias para llevar a cabo tareas de comunicación. Por ejemplo, enviar correos electrónicos.
  - *database*: Contiene las clases necesarias para interactuar con la base de datos.
  - *files*: Contiene las clases necesarias para interactuar con ficheros.
  - *log*: Contiene las clases necesarias para imprimir trazas en los ficheros de log.
  - *security*: Contiene las clases necesarias para comprobar y encriptar/desencriptar datos.
  - *Librerías*: Contiene las librerías para la gestión de correos electrónicos, gestión de ficheros Excel y gestión de imágenes.

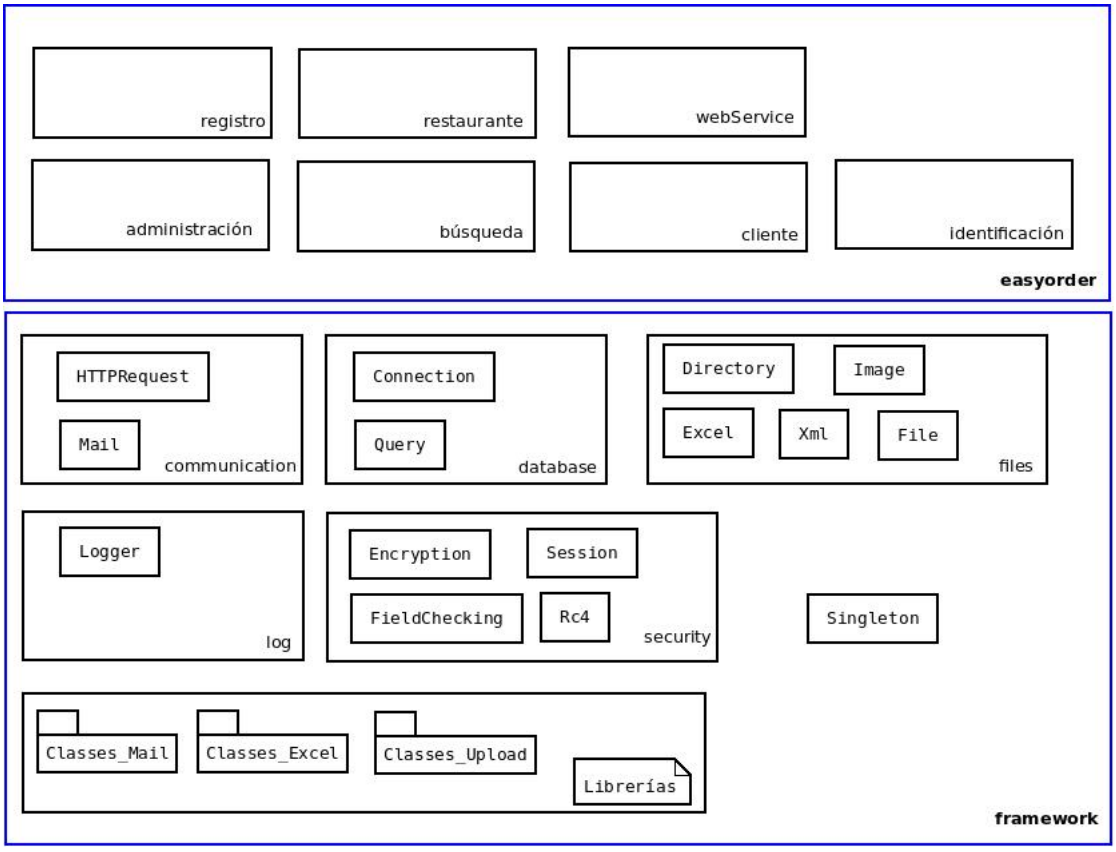


Figura 5.4: Diagrama de clases Web

Por otro lado, para Javascript no aplica un diagrama de clases puesto que no se han utilizado clases, sino ficheros .js con funciones que se comunicarán con el servidor mediante Ajax, o que interactuarán con la propia vista (el DOM) para modificar los componentes de la misma. En este caso, se mostrará en la figura 5.5 un diagrama con los paquetes de software de Javascript.

A continuación se explicará en que consiste cada carpeta o paquete del diagrama de la figura 5.4:

- ≻ *administración*: Carpeta que contiene los ficheros asociados a las funcionalidades que puede realizar el rol administrador.
- ≻ *búsqueda*: Carpeta que contiene los ficheros asociados a la funcionalidad de búsqueda de restaurantes.
- ≻ *registro*: Carpeta que contiene los ficheros asociados a las funcionalidades de registrar clientes y restaurantes.
- ≻ *cliente*: Carpeta que contiene los ficheros asociados a las funcionalidades que puede realizar el rol cliente.
    - *pagos*: Carpeta que contiene los ficheros asociados a los pagos.
- ≻ *headers*: Carpeta que contiene los ficheros que gestionan los diferentes tipos de cabeceras. Existen cuatro tipos de cabeceras: la cabecera del usuario no identificado y las cabeceras de los roles cliente, restaurante y administrador.
- ≻ *ekatam*: Carpeta que contiene ficheros con funciones comunes a toda la aplicación web.
- ≻ *restaurante*: Carpeta que contiene los ficheros asociados a las funcionalidades que puede realizar el rol restaurante.

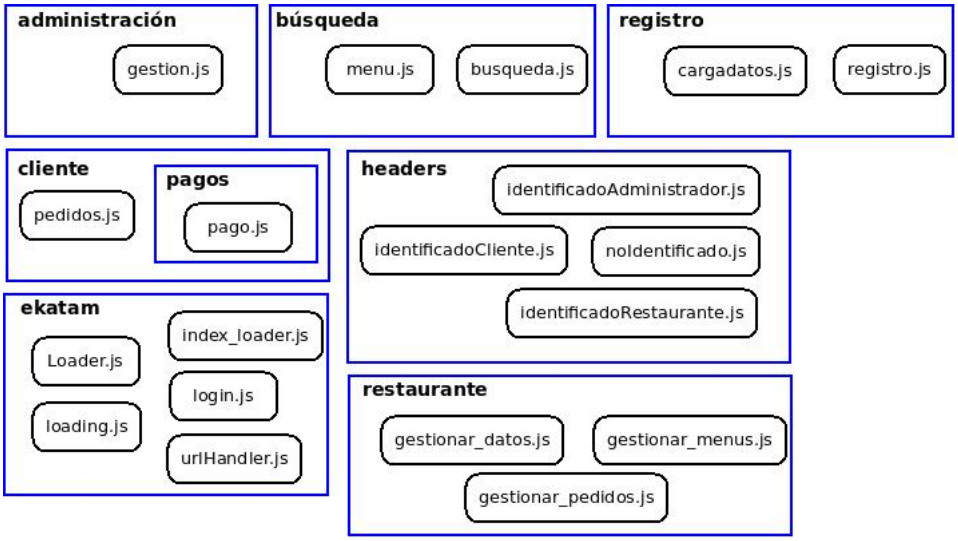


Figura 5.5: Diagrama Javascript

5.2.2. Aplicación Android

El diagrama de clases de la figura 5.6 muestra la estructura de clases desarrollada en Android para el proyecto. La estructura de la aplicación está dividida en paquetes:

- ≻ *mx.mit.easyorder.gestores*: Contiene las clases que actúan de controlador.
- ≻ *mx.mit.easyorder.modelos*: Contiene las clases asociadas a las diferentes funcionalidades.
- ≻ *mx.mit.easyorder.peticiones*: Contiene las clases que realizan las peticiones al servidor.
- ≻ *mx.mit.easyorder.activity*: Contiene las clases que controlan las vistas.
- ≻ *mx.mit.easyorder.adaptadores*: Contiene las clases que usaremos como adaptadores en la aplicación.
- ≻ *Clases Android*: Clases genéricas de Android de las cuales extienden algunas clases de la aplicación.

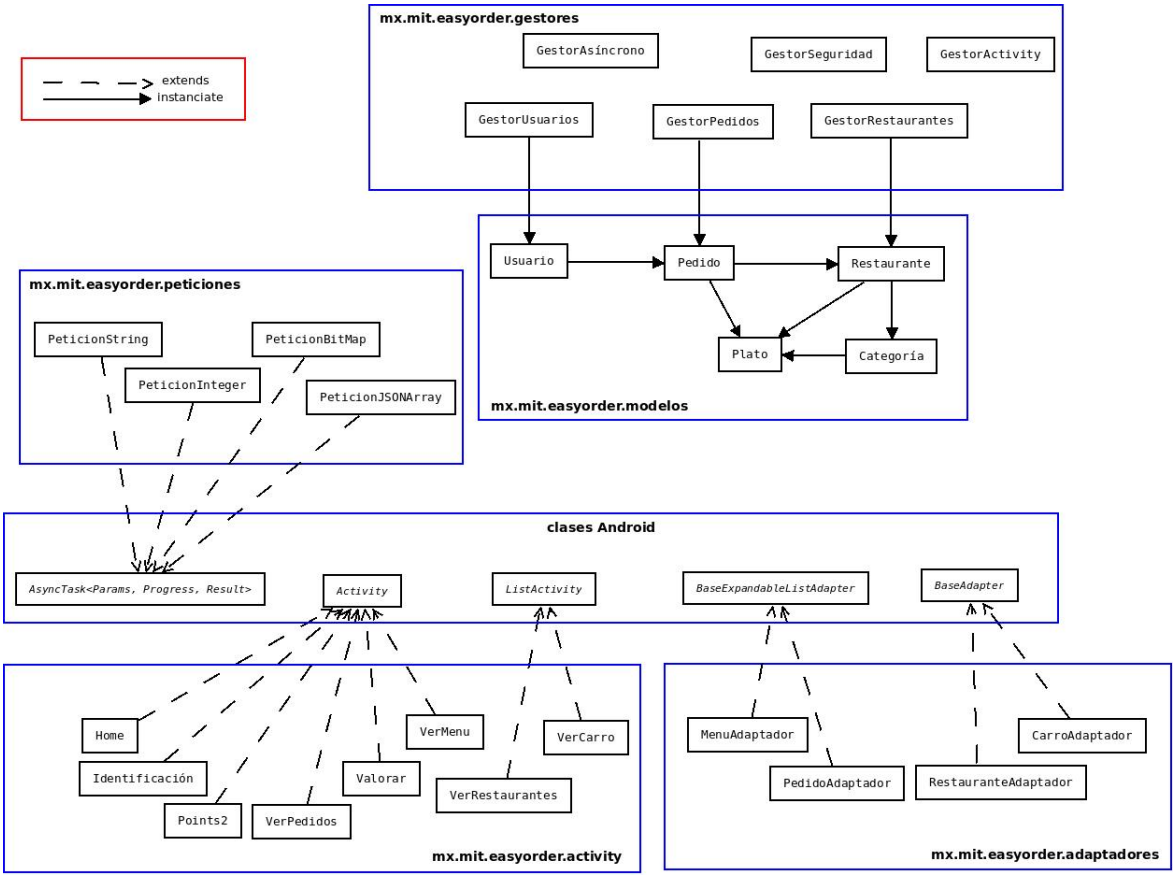


Figura 5.6: Diagrama de clases Android

5.3. Diseño

Se ha programado haciendo uso del paradigma de programación orientado a objetos.

En Easy Order se puede observar la programación orientada a objetos en el diagrama de clases de la aplicación web (ver figura 5.4), o bien, en el diagrama de la aplicación móvil (ver figura 5.6). También, se han utilizado los siguientes patrones de diseño:

- Modelo-Vista-Controlador (MVC).
- Patrón Singleton

Modelo-Vista-Controlador (MVC) El patrón Model-View-Controller (Modelo-Vista-Controlador), es patrón que se encarga de separar los datos de una aplicación, la interfaz de usuario, y la lógica de control. Por un lado, el controlador es el encargado de responder a lo que ocurre en la vista y comunicárselo al modelo. Siendo la vista la

interfaz de usuario y el modelo la lógica de la aplicación.

El modelo debe ser independiente facilitando así su reusabilidad. Mientras que el controlador no debe ver la vista, siendo ésta la que envía los datos al controlador.

En Easy Order se puede observar claramente el uso de este patrón, tal y como se indica en la Figura 2.1.

Patrón Singleton El patrón de diseño Singleton se utiliza para restringir la creación de varias instancias de una clase, proporcionando un acceso global a la instancia única generada.

Se puede observar claramente el uso del patrón Singleton en cualquier Gestor de la aplicación móvil (ver figura 5.6), o en la aplicación web, dentro del framework de PHP (ver figura 5.4).

Capítulo 6

Desarrollo

En el capítulo actual se explicarán detalladamente los conceptos de la implementación más relevantes de la aplicación web y la aplicación móvil.

6.1. Aplicación web

En el apartado correspondiente a la aplicación web se explicarán las comunicaciones que han sido necesarias, el motivo de adquirir una plantilla y la justificación de la estructuración de la aplicación web. Además, se mostrará cómo se ha realizado la geolocalización del usuario en la aplicación.

La aplicación web se ha dividido en tres capas: la primera capa, es la ofrecida por el navegador (la que ve el usuario), es decir, la **vista**. La siguiente capa, la capa correspondiente al **controlador**, es la que comunicará la interfaz del usuario y la lógica de control (el **modelo**). Esto es conocido como una arquitectura de *Modelo-Vista-Controlador,* tal como se detalla en el apartado 5.3. En este proyecto se ha decidido aplicar esta división porque se siguen siempre las mismas pautas para la creación de nuevas interfaces y además, facilita el mantenimiento del código por seguir siempre la misma organización.

Referente a la capa correspondiente a la vista, se ha desarrollado haciendo uso de tecnologías novedosas como son HTML5 para la estructuración del documento, conjuntamente con CSS3 para añadir estilos a la página web. Además, HTML5 junto con Javascript permite añadir dinamismo, ya que adapta el comportamiento de la web en función de las acciones del usuario. Se ha decidido usar HTML5 ya que es la última versión de HTML estable que, además, añade semántica en el contenido de los documentos web, aportando así nuevas etiquetas que añadan significado y facilitando el posicionamiento de la página web en buscadores. Es una de las principales diferencias con HTML4.01.

Un ejemplo de uso en el proyecto se puede observar en la figura 6.1 la cual utiliza algunas de las nuevas etiquetas, como son *header (*etiqueta que incluye información de la

cabecera del documento), *footer* (etiqueta que incluye información que cierra una sección del documento), etc.

Además no sólo hay nuevas etiquetas, también hay nuevos atributos que facilitan la interacción con el usuario final. En la figura 6.2, se puede observar un ejemplo de uso de los nuevos atributos, como lo son *pattern* y *type* (nuevos en HTML5) y *maxlength*. Estos atributos, se deben añadir en la definición de los campos de los formularios.

```
11 <!DOCTYPE html>
12 <html lang="es">
13 <head>
14 <title>EasyOrder</title>
15 <meta charset="utf-8">
16 <link rel="icon" href="/images/easyorder_logo_ico.png"
17     type="image/x-icon">
18 <link rel="shortcut icon" href="/images/easyorder_logo_ico.png"
19     type="image/x-icon" />
20
21 <link rel="stylesheet" href="/css/style.css">
22 <link rel="stylesheet" href="/css/superfish.css">
23
24 <!-- jQuery -->
25
26 <!-- Scripts -->
27
28
29 </head>
30 <body>
31     <header></header>
32     <section class="content"></section>
33     <footer> </footer>
34 </body>
35 </html>
```

Figura 6.1: Estructura documento HTML5

En el caso del atributo *pattern*, permite insertar reglas sobre el tipo de dato que se espera y su longitud. La ventaja de usar *pattern* es que siempre que el usuario introduce algún valor, se evaluará automáticamente sin necesidad de recurrir a ninguna función en Javascript y no permitirá enviar el formulario hasta que estén todos los campos acorde al *pattern* establecido. En el ejemplo de la figura 6.2, se le indica al campo que se puede introducir cualquier tipo de caracter, y que la longitud puede ser de 8 a 16 dígitos. Además, el atributo *maxlength* no permitirá al usuario introducir más de 16 caracteres.

```
<input type="password" pattern=".{8,16}" maxlength="16"
```

Figura 6.2: Ejemplo pattern HTML5

En el caso del atributo *type*, permite indicar que tipo de valor va esperar el campo

(fecha, e-mail, texto...). Esto es muy útil, por ejemplo, cuando se accede a la página desde un dispositivo móvil ya que el teclado se adaptará al tipo indicado en el campo cuando se intente escribir sobre éste.

Respecto al diseño gráfico de la aplicación, tras comprobar que esto era especialmente complicado de desarrollar debido a los nulos conocimientos que dispongo sobre esta temática, se ha adquirido una plantilla que estructura y estila el documento.

Inicialmente se desarrolló la aplicación web con un diseño hecho desde cero haciendo uso de CSS3. Pero como se puede observar en la figura 6.3, el diseño es mucho más pesado y menos adaptable a dispositivos móviles que el actual (figura 6.4). Además, el tiempo de dedicación es mucho mayor ya que la plantilla adquirida proporciona diferentes estilos que permiten aprovechar el espacio de la pantalla en cualquier dispositivo, según sus dimensiones.

Figura 6.3: Easy Order sin plantilla

Figura 6.4: Easy Order con plantilla

El diseño de esta plantilla está orientada al ámbito de la hostelería e incluye numerosas librerías en Javascript, además de alguna propia de quien diseñó la plantilla. Un ejemplo de uso de estas librerías se puede observar en el formulario del cliente: cuando el usuario no introduce ningún valor en un campo obligatorio, o bien lo introduce mal, se inserta de manera automática el error en un lateral del propio campo, tal y como se puede observar en la figura 6.5.

Figura 6.5: Ejemplo JS plantilla

Respecto a la lógica de la aplicación, se ha tratado de implementar con una Arquitectura Orientada a Servicios (SOA). Esta arquitectura, se basa en que la parte del cliente haga peticiones al servidor con objetivos muy concretos (login con parámetros "user" y "password", cálculo del precio de n productos, etcétera), y es por ello que la mayor parte de la lógica esté alojada en el cliente; hoy en día los navegadores tienen la potencia suficiente para ejecutar un peso elevado de las operaciones de la aplicación. Por ejemplo, en el caso de Easy Order, el código PHP se encargará en muchos casos de verificar la sesión del usuario, las cadenas de datos enviadas a través del navegador, etcétera. Esto hace que sean necesarias menos peticiones al servidor y que, de esta forma, evitemos tener que disponer de servidores muy potentes para la ejecución de la aplicación y de tener que distribuir la carga entre servidores.

Dejando de lado la vista, ahora se detallarán las capas correspondientes al controlador y al modelo, es decir, la parte correspondiente al servidor. Existen numerosos lenguajes para el desarrollo en el lado del servidor, como pueden ser: PHP, Java, Python, Node.js, etcétera. Por tanto, el primer problema que surge es decidir qué lenguaje será el candidato para desarrollar la aplicación. Sin embargo, la duda se planteó entre dos: Java y PHP. El resto no se valoraron puesto que es probable que la curva de aprendizaje en ellos fuese elevada y se disponía de poco tiempo.

Java, es uno de los lenguajes aprendidos en la universidad, por lo que desarrollar la aplicación en este lenguaje sería una de las soluciones más sencillas de aplicar.

La principal diferencia entre ambos lenguajes es el tipado. Es decir, las declaraciones de variables en Java deben ser explícitas, mientras que en PHP, no es necesario declarar el tipo las variables. Además, PHP es un lenguaje interpretado mientras que Java es un lenguaje compilado. Es decir, aplicar un cambio en el código PHP en un servidor de producción no es ningún problema: el cambio se aplica al instante de modificar el código. En Java no. Primero hay que compilar un fichero ejecutable en un servidor web y desplegarlo.

A pesar de todo esto, el cliente como requisito ha solicitado que se desarrollase en PHP. Para ello, existe la posibilidad de utilizar frameworks que te faciliten el trabajo ya que contienen actividades comunes en el desarrollo web, como puede ser el acceso a base de datos. Como pueden ser: CakePHP, Zend Framework, etcétera. Por lo tanto, uno de los planteamientos que surgen es decidir si es necesario el uso de frameworks.

CakePHP, es un framework para PHP que utiliza el patrón de diseño MVC. Es de código abierto.

Zend Framework, es un framework para PHP que utiliza el patrón de diseño MVC y una programación orientada a objetos. Es de código abierto y se distribuye bajo licencia BSD nueva.

Ambos frameworks son muy buenos, pero la curva de aprendizaje del Zend framework es mucho más elevada que la de CakePHP ya que está diseñado para proyectos grandes, mientras que CakePHP está diseñado para proyectos más pequeños.

A pesar de todo esto, se ha decidido implementar un framework propio como forma de adquirir conocimientos más elevados y así poder profundizar en el lenguaje, ya que hasta este proyecto apenas se había desarrollado ninguna aplicación en PHP. Además, se evita la "curva de aprendizaje" del framework y se reduce la carga que éste aporte a la aplicación, ya que sólo se implementará lo necesario para este proyecto.

6.1.1. Estructura del proyecto

Inicialmente se plantea seguir una estructura basada en el uso de clases que gestionasen cada modelo, pero debido a la magnitud de funcionalidades se decidió dividir la estructura en paquetes funcionales. Es decir, se tenían pocos ficheros con mucho código, por lo que era difícil tener una organización clara del proyecto.

Esta nueva estructura, como se puede observar en la sección 5.2.1, simplifica la longitud de los ficheros y permite obtener una estructura en la que el código está más repartido y organizado.

Un ejemplo de cómo está estructurado el proyecto se puede observar en la Figura 6.6.

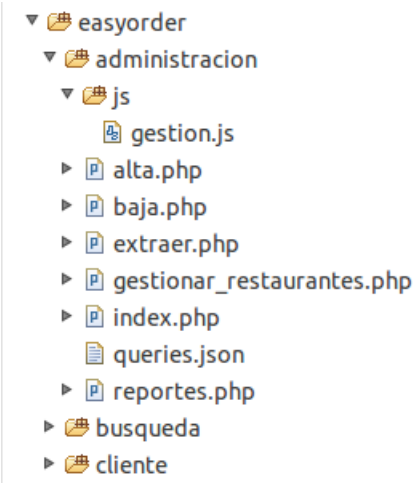


Figura 6.6: Ejemplo estructura proyecto

6.1.2. Comunicaciones

Las llamadas al servidor y al centro de pagos se realizan a través de servicios web. En esta sección se tratarán las diferentes comunicaciones que han sido necesarias en la aplicación web.

6.1.2.1. Comunicación cliente-servidor

Este apartado es muy importante en el desarrollo, ya que la mayor parte de la lógica de la aplicación está alojada en el cliente y es necesario comunicarse con el servidor para efectuar cualquier cambio en la base de datos.

La comunicación desde el lado del cliente se ha realizado a través de AJAX, siendo

este el encargado de enviar los datos al servidor por POST[1] y esperar la respuesta en el caso de ser necesaria (ver 6.7).

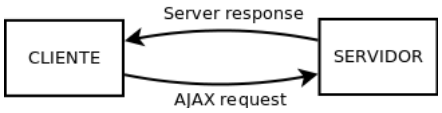


Figura 6.7

El servidor únicamente deberá validar los datos y comunicarse con el modelo, si fuese necesario. Y ahí, recuperará, modificará, insertará o eliminará datos de la base de datos.

En la figura 6.8 se puede observar un ejemplo de una función en Javascript que realiza una llamada asíncrona mediante AJAX. En este ejemplo, la función *login* es la encargada de llamar al servidor, y se utiliza cuando un usuario quiere identificarse en la aplicación. Para ello, será necesario especificar cómo se quieren enviar los datos (GET o POST), la dirección que tiene el servicio web y los datos que se quieren pasar.

```
10 function login(user, pwd){
11     var direccion = pathidentificacion + 'checkSession.php';
12     $.ajax({
13         url: direccion,
14         type:'POST',
15         data: {'user':user, 'pwd':pwd, 'm':'login'},
16         success: function (loggedUser) {
17
18         }
19     });
20 }
```

Figura 6.8: Ejemplo de envío de datos mediante AJAX desde JavaScript en el proyecto

En este caso, se ha decidido enviar los datos por POST, al fichero *checkSession.php*, y como datos: el usuario y contraseña que ha introducido el usuario. Además, se ha decidido utilizar una variable de nombre *m* para almacenar el tipo de consulta que se quiere hacer en el servidor y así poder identificarlo más fácilmente como se puede observar en la figura 6.9.

También, se ha definido el método *success* que se ejecutará cuando la llamada a través de AJAX obtenga una respuesta. En este método, se le indicará el valor que se va a recibir.

```
15 if($_POST['m'] =='login'){
16     login($_POST['user'], $_POST['pwd']);
17 }
```

Figura 6.9: Ejemplo de recogida de datos en el servidor con PHP

[1]Los datos son enviados en el cuerpo del mensaje HTTP.

Cuando el formato de los datos intercambiados entre el cliente y el servidor incluye listas o, principalmente, objetos se utiliza el formato JSON. Un ejemplo del formato JSON puede observarse en la figura 6.10.

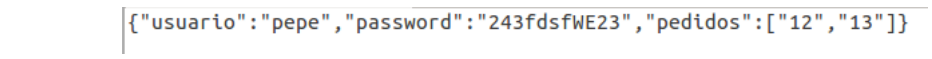


Figura 6.10: Ejemplo formato JSON

JSON es un formato de intercambio de datos que permite que los dos lenguajes (PHP y JS) "entiendan" los mensajes que se envían entre ellos cuando se está tratando con instancias de objetos o con listas. Para ello:

- En lado del cliente, Javascript, será necesario el uso de dos funciones:
  - *JSON.parse(textJSON)*: función encargada de convertir un texto en JSON en un objeto de Javascript.
  - *JSON.stringify(text)*: función encargada de convertir un objeto de Javascript en un texto en JSON.
- En el lado del servidor, PHP, será necesario el uso de dos funciones:
  - *json_ decode(textJSON)*: función encargada de convertir un texto en JSON en una variable de PHP.
  - *json_ encode(text)*: función encargada de convertir un objeto de PHP en un texto en JSON.

6.1.2.2. Comunicación aplicación-centro de pagos

Este apartado explicará cómo se ha comunicado el servidor con el entorno donde se encuentran alojados los servicios web de los pagos (centro de pagos).

Para ello, se debe tener en cuenta que el envío de datos de la transacción (nombre del restaurante, usuario que realiza la compra, importe del ticket...) viajarán desde el servidor en el que se aloja la aplicación al servidor que contiene el servicio web de manera encriptada.

También será necesario crear un servicio web propio que esté pendiente de las respuestas de los métodos de pago para notificar al usuario, y en caso de ser necesario, actualizar la base de datos. Estas respuestas, al igual que el envío de datos, vendrán encriptadas.

La encriptación y desencriptación de los datos de la comunicación entre la aplicación y el centro de pagos, siempre será a través de algoritmos proporcionados por MIT en Java[2],

[2]A excepción del método de encriptación RC4, que ha sido proporcionado por MIT en código PHP directamente

por lo que antes de efectuar ninguna operación con los datos, hay que pasar la cadena de texto a encriptar/desencriptar por un fichero ejecutable de Java que nos devuelva la cadena encriptada/desencriptada. Los algoritmos de encriptación utilizados son:

- RC4. Este algoritmo se utiliza para encriptar y desencriptar los pagos de Points2 y Webpay. Además, también se utiliza para la creación de sucursales.
- TripleDes. Este algoritmo se utiliza para encriptar y desencriptar los códigos QR de InvisibleCard.

Los servicios web de los métodos de pago Points2 y Webpay utilizan SOAP[3]. Además, el formato de los datos con los que tratan son XML. Por lo que para gestionar estos métodos de pago será necesario generar un XML con los datos de la transacción e encriptar/desencriptar según corresponda.

Un ejemplo de llamada a un web service que utiliza SOAP se puede observar en la figura 6.11.

```
313         $client = new \SoapClient(PAGOS_POINTS2_URL);
314         $params = array('xml' => $encriptedXML);
315         $respuesta = $client->__soapCall("redimirCobro", array($params));
```

Figura 6.11: Ejemplo llamada SOAP

El algoritmo que se debe seguir para implementar una llamada deberá ser el siguiente:

Algoritmo 6.1 Llamada a un web service que utiliza SOAP

1. Se define un cliente para conectarse con el servidor.
2. Se insertan los parámetros en un array asociativo poniendo como clave el nombre del parámetro que espera el servicio web.
3. Se hace una llamada a una función SOAP pasándole como parámetros el nombre del método al que queremos enviar los datos y los parámetros.

A continuación, se mostrarán los métodos de pago y sus peculiaridades.

Points2

Este método de pago únicamente valida certificados. Es decir, si un usuario tiene un certificado de Points2, puede pagar con el siempre que la compra sea igual o inferior al valor que contiene el certificado.

[3]Acrónico de *Simple Object Access Protocol*

Por lo tanto, únicamente será necesario crear un formulario en el que el usuario pueda introducir el certificado.

La invocación de este método de pago implica los siguientes pasos:

1. Creación de un XML con los campos necesarios para el Centro de Pagos de MIT. Son las siguientes etiquetas:

   ➢ <business></business>: Recoge los datos referentes al comercio que realiza el cobro

   - <id_company></id_company>: Identificador del comercio que realiza el cobro
   - <id_branch></id_branch>: Sucursal/oficina del comercio
   - <country></country>: País
   - <user></user>: Usuario proporcionado al comercio para realizar cobros
   - <pwd></pwd>: Contraseña proporcionada al comercio para realizar cobros

   ➢ <transaction></transaction>: Recoge los datos referentes a la transacción

   - <reference></reference>: Identificador del ticket/transacción
   - <source></source>: Un valor fijo indicado por MIT
   - <card></card>: Recoge los datos referentes al certificado
     - <printednumber></printednumber>: Número del certificado
     - <code></code>: Código que se utilizará para recoger la respuesta
   - <amount></amount>: Importe de la transacción
   - <currency></currency>: Moneda utilizada (MXN, USD, EUR)
   - <usrtransacction></usrtransacction>: Usuario que efectúa la transacción
   - <version></version>: Versión de la transacción

2. Se envían los datos del XML informados (a excepción del campo "amount" y "code") y encriptados en TripleDES con una semilla proporcionada por MIT.

3. Se obtiene un nuevo XML del servicio web y se desencripta con el algoritmo TripleDES que contiene la misma información que el enviado, pero con un código de respuesta.

   a) Si el código de respuesta es incorrecto (el número de certificado es inválido o se ha producido un error en la comunicación), se debe informar al usuario de que el certificado es incorrecto o que se ha producido un error.

   b) Si el código de respuesta es correcto (el número de certificado es válido), se enviará nuevamente al servicio web el XML encriptado introduciendo el valor "amount".

1) Se recibirá nuevamente el XML encriptado con un nuevo código:

   a' Si el código es correcto, se le informará al usuario que el pago se ha realizado correctamente.

   b' Si el código es incorrecto, se le informará al usuario que no dispone de cantidad suficiente en el certificado.

InvisibleCard

Este método de pago muestra un código QR[4] que el usuario deberá capturar con su teléfono móvil y pagarlo desde otra aplicación de MIT. En este caso, el servicio web únicamente espera una cadena con los datos de la transacción encriptados enviados mediante GET[5].

Para ello, usando la etiqueta *img* de HTML5, se mostrará el código QR al usuario especificando en el atributo *src* la URL que contiene la imagen con el código. La URL se genera a partir de la encriptación utilizando el algoritmo RC4 del importe, el identificador del comercio que realiza el cobro junto con la sucursal, la referencia del ticket/transacción y una fecha de expiración. Gráficamente:

URL = CADENA_FIJA + algoritmoRC4(companyId + amount + transactionID + expirationDate)

Figura 6.12: Construcción de la URL

Webpay

Este método de pago permite al usuario pagar con tarjeta de crédito y le proporciona un formulario con los datos del pedido y en el que deberá introducir los datos de su tarjeta. Para este formulario, se ha realizado un diseño en HTML que ha sido validado por los responsables de Webpay, y que está alojado en un servidor de MIT.

El funcionamiento de este método de pago se basa en la utilización de la etiqueta *iframe*[6] de HTML, gracias a la cual se mostrará el formulario especificando en el atributo *src* la URL que lo contiene. Tras rellenar los datos de este *iframe* y pulsar el botón de "Pagar", el sistema de MIT enviará una petición a nuestro servidor con el resultado de la transacción que leeremos y procesaremos. Según la respuesta, podrá darse el caso que:

- La entidad bancaria rechace la transacción
- Ocurra un error en el proceso de pago

[4] Acrónimo de *Quick Response Code*.

[5] Los datos son enviados en la URL del servicio web.

[6] Etiqueta de HTML que permite introducir otro documento HTML en la página HTML actual.

- ≻ El cliente no tenga dinero suficiente en la tarjeta de crédito
- ≻ El pago se efectúe correctamente

En función de la respuesta obtenida, se le mostrará al usuario final un mensaje u otro.

6.1.2.3. Comunicación aplicación-base de datos

La comunicación de la aplicación con la base de datos, se realiza a través del *framework*. Como se puede observar en la figura 6.13, se ha creado un proceso común para comunicarse con la base de datos.

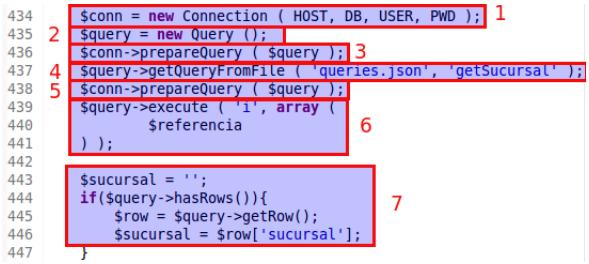


Figura 6.13: Ejemplo conexión BBDD

También, se ha decidido generar ficheros JSON que almacenen las consultas SQL con una pequeña explicación como se puede observar en la figura 6.15. Además también se puede ver que se utilizan sentencias preparadas (prepared statements) que impiden realizar ataques de SQL Injection (ya que el propio lenguaje lo filtra) y se le permiten pasar los parámetros obtenidos a la query, evitando tener que escribir cadenas largas y difíciles de depurar de código.

Otra forma de realizar las consultas SQL sería sin utilizar sentencias preparadas (prepared statements). Un ejemplo se puede observar en la figura 6.14. Esta forma es más costosa e impide realizar la misma consulta en diferentes puntos del código sin tener que volver a escribirla.

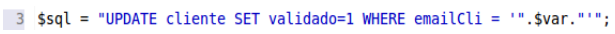


Figura 6.14: Ejemplo de una consulta

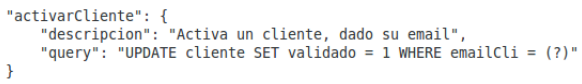


Figura 6.15: Ejemplo de una consulta almacenada en JSON

El algoritmo que deben seguir todos los métodos que deseen realizar una consulta a la base de datos es el siguiente:

Algoritmo 6.2 Comunicación aplicación con base de datos

1. Se crea una nueva conexión. Para ello, será necesario crear una instancia de la clase Connection.
2. Se crea una instancia de la clase Query.
3. Se añade a la conexión la instancia de la query creada.
4. Se recoge del fichero *queries.json* la consulta.
5. Se prepara la sentencia SQL para la consulta.
6. Se ejecuta la sentencia y se almacena en una variable de la clase Query el resultado. En el caso de que la sentencia necesite parámetros, se pasarán al método siendo el primer parámetro el formato de los datos y el segundo, un array con los parámetros.
7. Se comprueba si la consulta realizada ha recibido algún resultado, y en el caso de ser necesario, se podrá leer este resultado.

6.1.3. Geolocalización

Una de las funcionalidades ofrecidas en la aplicación web es la posibilidad de geolocalizarse con ayuda de la API de Geolocalización y de Google Maps. Es decir, permite al usuario mediante el clic de un botón, localizar el código postal de la dirección en la que se encuentra.

En la imagen de la figura 6.16 se puede observar sobre qué botón es necesario clicar para permitir la geolocalización en la aplicación web. Una vez clicado el botón, el navegador nos pedirá permiso para obtener nuestra ubicación y escribirá el código postal.

Figura 6.16: Cómo geolocalizarse desde la aplicación web

En la figura 6.17 se puede observar un ejemplo de como se obtiene el código postal a partir del clic de un usuario en un botón.

```
$('#locate').click(
    function() {
1       navigator.geolocation.getCurrentPosition(mostrarLocalizacion);
    });

function mostrarLocalizacion(posicion) {
2   var latitud = posicion.coords.latitude;
    var longitud = posicion.coords.longitude;
3   var latlng = new google.maps.LatLng(latitud, longitud);
    var geocoder = new google.maps.Geocoder();

    geocoder.geocode(
        {'latLng' : latlng},
        function(results, status) {
            if (status == google.maps.GeocoderStatus.OK) {
                if (results[0]) {
4                   for (var i = 0; i < results[0].address_components.length; i++) {
                        if (results[0].address_components[i].types[0] == 'postal_code') {
                            $("#cp_input").val(results[0].address_components[i].long_name);
                        }
                    }
                }
            }
```

Figura 6.17: Ejemplo de uso de la geolocalización

El algoritmo que se debe seguir es el siguiente:

Algoritmo 6.3 Geolocalización

1. Si el navegador permite la geolocalización, se llamará al método *getCurrentPosition()* de la API de Geolocalización. Siendo los parámetros de este método, el nombre del método al que se debe llamar en caso de éxito.

2. Se obtendrá la latitud y la longitud de la posición actual.

3. Con ayuda de la API de Google Maps, se obtendrá un punto geográfico, pasando como parámetros la latitud y longitud previamente calculadas. Y se creará una instancia del servicio Geocoder, encargado de obtener en función de un punto geográfico calculado la dirección.

4. Tras haber realizado una llamada al servicio Geocoder, se leerá de la respuesta obtenida el código postal. Y en este caso, también se escribirá por pantalla el código postal obtenido.

6.1.4. Otras funcionalidades

A continuación se explicará de forma breve en qué consisten algunas funcionalidades importantes que ofrece Easy Order.

Envío e-mails Una de las funcionalidades ofrecidas en la aplicación web es el envío de e-mails de forma interna con ayuda de la API de PHPMailer. Por ejemplo, por solicitud de MIT en los registros de los clientes se envía un e-mail al cliente para que valide su cuenta.

Reportes Una de las funcionalidades ofrecidas en la aplicación web es la posibilidad de imprimir reportes con ayuda de la API PHPExcel. Es decir, permite al usuario imprimir en formato excel los reportes de los restaurantes.

6.2. Aplicación Android

HTML5 utilizado también en dispositivos móviles es un claro competidor de las aplicaciones nativas.

La mayor ventaja que ofrece HTML5 frente a las aplicaciones nativas es que permite ser visualizada en la mayoría de plataformas o sistemas móviles mediante el navegador. El inconveniente más importante es que al ser una aplicación web la potencia, la fluidez y riqueza de recursos no será igual que si ésta fuera nativa de la plataforma.

El inconveniente de las aplicaciones nativas, es la necesidad de conocer el lenguaje que utilice la plataforma.

Existen varios sistemas operativos para el móvil: Android, iOS, Symbian, WindowsPhone, etcétera. Por lo tanto el primer problema que surge es decidir qué sistema operativo será el candidato para desarrollar la aplicación. Sin embargo, la duda se planteó entre dos: iOS y Android.

iOS, está diseñado en ObjetiveC y al ser un lenguaje exclusivo de Apple, la curva de aprendizaje es elevada. Además, es necesario disponer de una licencia específica de Apple para poder desarrollar aplicaciones y de un iPhone y un Mac.

Android, está diseñado para ser programado en Java y XML. También, permite desarrollar en cualquier PC y proporciona herramientas para el desarrollo gratuitas.

Finalmente, se ha decidido realizar una aplicación únicamente para Android, ya que la cota de uso de este sistema operativo es mucho mayor que el uso de iOS en los últimos años y porque es gratuíto.

Con respecto a la aplicación desarrollada, se irán detallando diversos puntos que pueden ser de interés.

En primer lugar, se explicarán los permisos necesarios para ejecutar la aplicación. Android requiere de la aceptación de diversos permisos a la hora de instalar una aplicación. En Easy Order ha sido necesario la adición de algunos, tal como se puede observar en la figura 6.18. Significan lo siguiente:

- ≻ *INTERNET*: Permitir a la aplicación conectarse a internet.
- ≻ *ACCESS_FINE_LOCATION*: Permitir a la aplicación calcular la posición exacta por medio del GPS.
- ≻ *ACCESS_COARSE_LOCATION*: Permitir a la aplicación calcular la posición aproximada a través de fuentes de red.

```
11      <uses-permission android:name="android.permission.INTERNET" />
12      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
13      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Figura 6.18: Permisos necesarios

Centrándonos en aspectos concretos de eficiencia y almacenamiento temporal de datos de la aplicación móvil, se podría destacar el uso de listas (concretamente de *ArrayLists*) y de tablas hash (*HashMap*).

Las listas nos proporcionarán iteradores que nos permitan recorrer los datos de manera sencilla, mientras que en las tablas hash se almacenan aquellos datos en los que será necesario realizar alguna búsqueda mediante ID o similar ya que son fácilmente identificables mediante una clave, obteniendo así su valor correspondiente (pares clave-valor).

Por ejemplo, se han utilizado tablas hash para buscar restaurantes. En este caso, la clave será el email del restaurante y de esta forma podremos obtener los datos de un restaurante (previamente almacenados en tablas hash) con un coste de ejecución cercano a O(1) (lineal).

6.2.1. Estructuración proyecto

A la hora de estructurar el proyecto en la aplicación móvil se ha utilizado una organización más genérica que la de la aplicación web, en la que a primera vista queda claro el patrón utilizado *(Model-View-Controller)*. Es decir, se ha intentado realizar una aplicación sencilla utilizando clases que gestionen los modelos.

Un ejemplo de cómo está estructurado el proyecto se puede observar en la Figura 6.19.

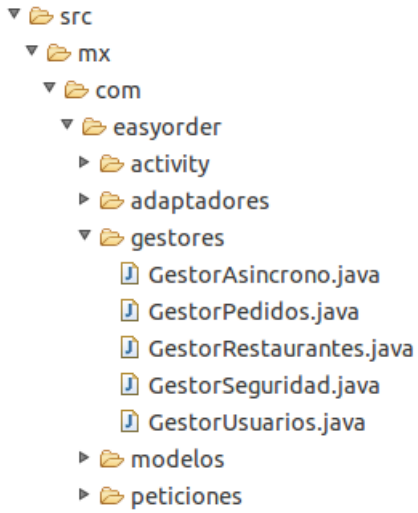


Figura 6.19: Ejemplo estructura proyecto

6.2.2. Comunicación servidor

La aplicación móvil se conecta con el servidor de la aplicación web siempre que es necesario realizar algún cambio o comprobación en la base de datos. Esto es posible ya que lo proporciona la Arquitectura Orientada a Servicios utilizada. Además, de esta forma conseguiremos evitar incoherencias por disponer de diversas fuentes de datos, puesto que sólo habrá una única con un punto de acceso común.

6.2.2.1. Comunicación desde la aplicación

Las comunicaciones de la aplicación con el servidor se han realizado a través de peticiones asíncronas (al igual que la comunicación en la web entre cliente-servidor), las cuales son las encargadas tanto de enviar como de tratar los datos recibidos.

En la figura 6.20 se puede observar un ejemplo de uso de una petición. En el cual, la clase en la que se ha realizado la petición extiende de la clase genérica *AsyncTask*[7], y siguiendo este mismo ejemplo, en la figura 6.21 se puede ver la definición de la clase.

[7]Para extender de la clase *AsyncTask* es necesario pasar tres parámetros. El formato de los datos que se van a enviar, del progreso y de los datos que se esperan recibir.

```
protected Integer doInBackground(ArrayList<NameValuePair>... listparametros) {
    int resultado = 0;
    ArrayList<NameValuePair> parametros = listparametros[0];

1   HttpParams httpParameters = new BasicHttpParams();
    HttpConnectionParams.setConnectionTimeout(httpParameters, 15000);
    HttpConnectionParams.setSoTimeout(httpParameters, 15000);

2   HttpClient httpclient = new DefaultHttpClient(httpParameters);
    HttpPost httppost = new HttpPost(getUrl());
    try {
3       httppost.setEntity(new UrlEncodedFormEntity(parametros));
        HttpResponse response = httpclient.execute(httppost);
        httpclient.execute(httppost);
        HttpEntity entity = response.getEntity();

        resultado = Integer.parseInt(EntityUtils.toString(entity));
    } catch (Exception e) {
        resultado = 0;
    }
    return resultado;
}
```

Figura 6.20: Ejemplo petición asíncrona

```
public class PeticionInteger extends AsyncTask<ArrayList<NameValuePair>, Void, Integer>{
```

Figura 6.21: Ejemplo extends petición asíncrona

Como se ha podido observar en el ejemplo anterior (figura 6.20), los pasos necesarios para realizar una petición asíncrona van en el método *doInBackground* (método de la clase *AsyncTask*). El algoritmo a seguir a la hora de realizar peticiones asíncronas es el siguiente:

Algoritmo 6.4 Peticiones asíncronas

1. Se crea una instancia de una clase que representa los parámetros del protocolo HTTP y se le asigna un tiempo máximo de respuesta.
2. Se define un cliente con los parámetros creados en el paso anterior y se crea una conexión *HttpPost* a la dirección en la que está el servicio web.
3. Se insertan los parámetros que se quieren enviar, se ejecuta la llamada y se recoge la respuesta en la variable *entity*.

Asimismo, también es necesario que los datos que se envíen a través de una petición asíncrona estén en un array de elementos con el formato nombre/valor. Para ello se utilizará la clase *BasicNameValuePair*[8].

En la figura 6.22, se puede observar un ejemplo en el que se crea una variable que va a contener los valores que deseamos enviar a la petición. En este caso, se quiere enviar el usuario y contraseña que ha introducido el usuario, y para seguir con la lógica de la

[8]Clase que encapsula valores en el formato Nombre/Valor.

aplicación web, una variable de nombre *m* que guarde el tipo de consulta que se quiere hacer en la aplicación.

```
ArrayList<NameValuePair> parametros = new ArrayList<NameValuePair>();
parametros.add(new BasicNameValuePair("m", "identificacion"));
parametros.add(new BasicNameValuePair("usuario", usuario));
parametros.add(new BasicNameValuePair("password", password));
```

Figura 6.22: Ejemplo parámetros petición asíncrona

6.2.2.2. Geolocalización

Una de las funcionalidades ofrecidas en la aplicación móvil es la posibilidad de geolocalizarse con ayuda de los servicios de localización del dispositivo móvil. Es decir, permite al usuario mediante el clic de un botón, localizar el código postal de la dirección en la que se encuentra.

En la imagen de la figura 6.23 se puede observar sobre qué botón es necesario clicar para permitir la geolocalización en la aplicación web. Una vez clicado el botón, el navegador nos pedirá permiso para obtener nuestra ubicación y escribirá el código postal.

Figura 6.23: Cómo geolocalizarse desde la aplicación movil

En la figura 6.24 se puede observar un ejemplo de como se obtiene el código postal.


```
96          LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
97
98          boolean isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
99          if (!isGPSEnabled) {
100             Toast.makeText(getApplicationContext(),
101                     "No está activada la geolocalización",
102                     Toast.LENGTH_LONG).show();
103         } else {
104             Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
105             double longitude = location.getLongitude();
106             double latitude = location.getLatitude();
107
108             Geocoder geocoder = new Geocoder(Home.this, Locale.getDefault());
109             try {
110                 List<Address> addresses = geocoder.getFromLocation(latitude, longitude, 1);
111                 if (addresses.isEmpty()) {
112                     Toast.makeText(getApplicationContext(),
113                             "Error en la geolocalización",
114                             Toast.LENGTH_LONG).show();
115                 }
116                 codigopostal.setText(addresses.get(0).getPostalCode());
117             } catch (IOException e) {
118                 Toast.makeText(getApplicationContext(),
119                         "Error en la geolocalización",
120                         Toast.LENGTH_LONG).show();
121             }
122         }
```

Figura 6.24: Ejemplo de uso de la geolocalización

El algoritmo que se debe seguir para poder obtener el código postal en una aplicación Android es el siguiente:

Algoritmo 6.5 Geolocalización

1. Se intenta acceder a los servicios de localización del dispositivo.
2. Se comprueba si los servicios de localización están activos.
3. Si el dispositivo móvil tiene activada la geolocalización, se obtiene la latitud y la longitud de la posición actual.
4. Se crea una instancia del servicio Geocoder, encargado de obtener en función de un punto geográfico calculado la dirección.
5. Se realiza una llamada al servicio Geocoder obteniendo una dirección, pasándole como parámetros la latitud y longitud previamente calculadas.
6. Se escribirá por pantalla el código postal obtenido de la respuesta del Geocoder.

Capítulo 7

Verificación y evaluación

Las pruebas realizadas han sido funcionales, es decir, se han realizado pruebas de aceptación tanto alfa (realizadas por el cliente sin ayuda de los desarrolladores) como beta (realizadas por el cliente con ayuda de los desarrolladores). No se han realizado pruebas unitarias por los ajustados plazos que MIT definía.

El plan de pruebas ha sido desarrollado por MIT y cada vez que se finalizaba un módulo de software, había que ejecutar las pruebas descritas. En el caso de la aplicación web, estas pruebas son realizadas en diferentes navegadores.

Uno de los problemas principales en la evaluación del sistema, ha sido que se solicitaban más requisitos en las pruebas que en la captura de requisitos.

Por ejemplo, al inicio del proyecto se nos facilitó una lista con todos los códigos postales y direcciones de México. Los códigos postales contenían 4 y 5 dígitos. A la hora de que el usuario final introdujese un código postal, a aquellos que contenían 4 dígitos no se les añadía ningún 0 a la izquierda del código postal; ni para tratarlo contra la BBDD, ni para mostrárselo al usuario. Sin embargo, tras haber desarrollado la funcionalidad, MIT en sus pruebas definió que sería necesario que de cara al usuario final, si éste introducía "4500", en la aplicación se tratase el código como si fuese "04500".

A continuación, se detallarán las pruebas divididas en escenarios (tal y como MIT lo ha proporcionado):

Escenario I - Registro restaurante

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_001	En la pantalla principal mostrar 4 botones: - Logo de Easy Order. - Registrarse. - Ingresar. - Búsqueda de restaurante por código postal.	Se deberán mostrar las opciones en la página principal.	Se muestran las opciones en la página principal.	Cumplidos los requisitos solicitados.
QA_002	El logo de Easy Order deberá redireccionar a la página principal.	Se redireccionará a la página principal al hacer clic en el logo.	Se redirecciona a la página principal al hacer clic en el logo.	Cumplidos los requisitos solicitados.
QA_003	Link a Twitter.	Se deberá direccionar a la página de Twitter de Easy Order.	-	Pendiente de proporcionar por parte de MIT la URL.
QA_004	Link a Facebook.	Se deberá direccionar a la página de Facebook de Easy Order.	-	Pendiente de proporcionar por parte de MIT la URL.
QA_005	Registrar restaurante	Se dará de alta un restaurante en el sistema junto a su sucursal.	Se da de alta un restaurante en el sistema, pero da un error al registrar la sucursal.	Corregido. **Solución**: Cambiar los parámetros enviados al servicio web de MIT para registrar la sucursal.
QA_006	Introducir nombre de restaurante con 40 caracteres y con espacios.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.1: Pruebas escenario I - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_007	Introducir nombre de restaurante con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_008	Introducir nombre de restaurante con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_009	Introducir nombre de restaurante con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_010	Introducir nombre de restaurante con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_011	Introducir nombre responsable con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_012	Introducir nombre responsable con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_013	Introducir nombre responsable con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_014	Dar de alta correo electrónico con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_015	Dar de alta correo electrónico con sólo números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.2: Pruebas escenario I - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_016	Dar de alta correo electrónico alfanumérico.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_017	Dar de alta correo electrónico con guiones.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_018	Dar de alta correo electrónico vacío.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_019	Introducir contraseña con mayúsculas y minúsculas.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_020	Introducir contraseña con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_021	Introducir contraseña con caracteres especiales.	Se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_022	Introducir contraseña vacía.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_023	Introducir contraseña de confirmación diferente.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_024	Introducir contraseña con minúsculas y contraseña de confirmación con mayúsculas.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_025	Introducir número de 8 números en el teléfono local.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.3: Pruebas escenario I - parte 3

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_026	Introducir número de 8 caracteres especiales en el teléfono local.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_027	Introducir número de 8 letras en el teléfono local.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_028	Introducir número de 10 números en el celular.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_029	Introducir número de 10 caracteres especiales en el celular.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_030	Introducir número de 10 letras en el celular.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_031	Seleccionar tipo de comida de un combo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_032	Introducir código postal con 5 dígitos.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_033	Introducir código postal con 4 dígitos.	Se añadirá un 0 al inicio del dato y se permitirá buscar los restaurantes que reparten en ese código postal.	Se muestra un mensaje de error y no se realiza la búsqueda de restaurantes.	Corregido. **Solución:** Se añade un 0 al inicio del datos siempre que sea un código postal de 4 dígitos.

Cuadro 7.4: Pruebas escenario I - parte 4

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_034	Introducir código postal con 5 ceros.	No se permitirá guardar el dato.	Se permite guardar el dato.	Corregido. **Solución:** Se comprueba si el código postal introducido contiene 5 ceros, y de ser así se le muestra un mensaje al usuario.
QA_035	Introducir código postal vacío.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_036	Visualizar las colonias asociadas al código postal.	Se mostrarán las colonias asociadas al código postal.	Se muestran las colonias asociadas al código postal.	Cumplidos los requisitos solicitados.
QA_037	Introducir calle con 20 caracteres.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_038	Introducir calle con sólo números.	Se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución:** Se cambia la definición del campo, para que permita al usuario introducir números.
QA_039	Introducir calle con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_040	Introducir calle con letras y números.	Se permitirá guardar el dato.	No se permite guardar el dato.	Corregido. **Solución:** Se cambia la definición del campo, para que permita al usuario introducir letras y números.

Cuadro 7.5: Pruebas escenario I - parte 5

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_041	Introducir calle con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_042	Introducir calle vacía.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_043	Introducir número con 20 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_044	Introducir número sólo con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_045	Introducir número sólo con letras.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_046	Introducir número con letras y números.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_047	Introducir número con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_048	Introducir número vacío.	No se permitirá registrar el restaurante.	No se permite registrar el restaurante.	Cumplidos los requisitos solicitados.
QA_049	Adjuntar una imagen con un mínimo de bytes.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_050	Adjuntar una imagen con 1MB de máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.6: Pruebas escenario I - parte 6

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_051	Adjuntar una imagen con un nombre superior a 40 caracteres.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_052	Introducir cuenta clabe de 18 dígitos.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_053	Introducir cuenta clabe de visa/mc.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_054	Introducir cuenta clabe de amex.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_055	Introducir cuenta clabe de mayor a 18 dígitos.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_056	Introducir cuenta clabe de menor a 18 dígitos.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_057	Registrar restaurante con un correo electrónico de sólo letras ya registrado.	No se permitirá registrar el restaurante mostrándole un mensaje.	Se permite registrar el restaurante.	Corregido. **Solución**: No se permite registrar el restaurante y se muestra un mensaje de información.

Cuadro 7.7: Pruebas escenario I - parte 7

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_058	Registrar restaurante con un correo electrónico alfanumérico ya registrado.	No se permitirá registrar el restaurante mostrándole un mensaje.	Se permite registrar el restaurante.	Corregido. **Solución**: No se permite registrar el restaurante y se muestra un mensaje de información.
QA_059	Registrar restaurante con un correo electrónico numérico ya registrado.	No se permitirá registrar el restaurante mostrándole un mensaje.	Se permite registrar el restaurante.	Corregido. **Solución**: No se permite registrar el restaurante y se muestra un mensaje de información.
QA_060	Registrar restaurante con un correo electrónico con guiones ya registrado.	No se permitirá registrar el restaurante mostrándole un mensaje.	Se permite registrar el restaurante.	Corregido. **Solución**: No se permite registrar el restaurante y se muestra un mensaje de información.
QA_061	Acceder a la aplicación con una contraseña incorrecta.	No se permitirá identificarse al restaurante.	No se permitirá identificarse al restaurante.	Cumplidos los requisitos solicitados.
QA_062	Acceder a la aplicación con una contraseña en mayúsculas, cuando la contraseña del registro fue en minúsculas.	No se permitirá identificarse al restaurante.	No se permitirá identificarse al restaurante.	Cumplidos los requisitos solicitados.

Cuadro 7.8: Pruebas escenario I - parte 8

Escenario II - Modificar información del restaurante (Rol restaurante)

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_063	En la pantalla principal mostrar 4 botones: - Logo de Easy Order. - Personalizar datos. - Personalizar menú. - Gestionar pedidos.	Se deberán mostrar las opciones en la página principal.	Se muestran las opciones en la página principal.	Cumplidos los requisitos solicitados.
QA_064	Permitir modificar la información del restaurante	Se permitirá modificar los datos del restaurante.	Se permite modificar los datos del restaurante.	Cumplidos los requisitos solicitados.
QA_065	Permitir modificar el tipo de comida.	Se permitirá modificar el tipo de comida.	No se permite modificar el tipo de comida.	Corregido. **Solución:** Se permite modificar el tipo de comida.
QA_066	Introducir nombre de restaurante con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_067	Introducir nombre de restaurante con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_068	Introducir nombre restaurante con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.9: Pruebas escenario II - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_069	Introducir nombre de restaurante vacío.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_070	Introducir contraseña con mayúsculas y minúsculas.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_071	Introducir contraseña con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_072	Introducir contraseña con caracteres especiales.	Se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_073	Introducir contraseña vacía.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_073	Introducir contraseña vacía.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_074	Introducir contraseña de confirmación diferente.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_075	Introducir contraseña con minúsculas y contraseña de confirmación con mayúsculas.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_076	Introducir número de 8 números en el teléfono local.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.10: Pruebas escenario II - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_077	Introducir número de 8 caracteres especiales en el teléfono local.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_078	Introducir número de 8 letras en el teléfono local.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_079	Introducir número de 10 números en el celular.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_080	Introducir número de 10 caracteres especiales en el celular.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_081	Introducir número de 10 letras en el celular.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_082	Introducir hora de apertura.	Se permitirá guardar el dato.	No se permite introducir hora de apertura.	Corregido. **Solución**: Se permite introducir el dato en el sistema.
QA_083	Introducir hora de cierre.	Se permitirá guardar el dato.	No se permite introducir hora de cierre.	Corregido. **Solución**: Se permite introducir el dato en el sistema.

Cuadro 7.11: Pruebas escenario II - parte 3

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_084	Adjuntar una imagen un mínimo de bytes.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_085	Adjuntar una imagen con 1MB de máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_086	Adjuntar una imagen con un nombre superior a 40 caracteres.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_087	Permitir añadir códigos postales de reparto.	Se permitirá añadir códigos postales.	No se permite añadir códigos postales de reparto.	Corregido. **Solución**: Se permiten añadir códigos postales de reparto.
QA_088	Permitir eliminar códigos postales de reparto.	Se permitirá eliminar códigos postales, siempre que quede un código postal mínimo.	No se permite eliminar códigos postales de reparto.	Corregido. **Solución:** Se permite eliminar códigos postales del sistema, con la condición de que quede mínimo un código postal.

Cuadro 7.12: Pruebas escenario II - parte 4

Escenario III - Personalizar menú (Rol restaurante)

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_089	Permitir introducir categorías.	Se permitirá añadir categorías.	Se permite añadir categorías.	Cumplidos los requisitos solicitados.
QA_090	Eliminar categorías.	Se dará de baja la categoría.	Se da de baja la categoría en el sistema.	Cumplidos los requisitos solicitados.
QA_091	Introducir categoría con más de una palabra.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_092	Introducir categoría vacía.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_093	Permitir introducir platos.	Se permitirá añadir platos.	Se permite añadir platos.	Cumplidos los requisitos solicitados.
QA_094	Introducir del plato: -El número. -El nombre. -El precio.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_095	Introducir del plato: -El número. -El precio.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.13: Pruebas escenario III - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_096	Introducir 0 como número de plato.	No se permitirá guardar el dato.	Se guarda el dato en el sistema.	Corregido. **Solución:** Se cambia la definición del campo, para que el número mínimo que pueda introducir el usuario sea un 1.
QA_097	Introducir letras y caracteres especiales como número de plato.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_098	Introducir letras como número de plato.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_099	Introducir caracteres especiales como número de plato.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_100	Introducir una letra como nombre del plato.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_101	Introducir letras como nombre del plato.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_102	Introducir caracteres especiales como nombre del plato.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_103	Introducir del plato: -El nombre. -El número.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.14: Pruebas escenario III - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_104	Introducir 0 como precio del plato.	No se permitirá guardar el dato.	Se guarda el dato en el sistema.	Corregido. **Solución**: No se permite guardar el dato en el sistema.
QA_105	Introducir letras como precio del plato.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_106	Introducir caracteres especiales como precio del plato.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_107	Eliminar categorías con platos.	Se mostrará un mensaje de advertencia y si lo acepta, se darán de baja tanto los platos como las categorías.	Se muestra un mensaje de advertencia y si lo acepta, se dan de baja tanto los platos como las categorías.	Cumplidos los requisitos solicitados.
QA_108	Eliminar plato.	Se mostrará un mensaje de advertencia y si lo acepta, se dará de baja el plato.	Se muestra un mensaje de advertencia y si lo acepta, se da de baja el plato.	Cumplidos los requisitos solicitados.

Cuadro 7.15: Pruebas escenario III - parte 3

Escenario IV - Gestión pedidos (Rol restaurante)

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_109	Mostrar los pedidos.	Se mostrarán los pedidos.	Se muestran los pedidos.	Cumplidos los requisitos solicitados.
QA_110	Cambiar el estado de un pedido.	Se cambiará el estado del pedido.	Se cambia el estado del pedido.	Cumplidos los requisitos solicitados.
QA_111	Mostrar los información del cliente en los pedidos.	Se mostrarán los pedidos junto a la información del cliente.	No se muestran correctamente los datos.	Corregido. **Solución:** Se muestran los pedidos junto a la información del cliente.

Cuadro 7.16: Pruebas escenario IV

Escenario V - Registro cliente

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_112	Registrar cliente.	Se dará de alta un cliente en el sistema.	Se da de alta un cliente en el sistema.	Cumplidos los requisitos solicitados.
QA_113	Introducir nombre con 40 caracteres y con espacios.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_114	Introducir nombre con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_115	Introducir nombre con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_116	Introducir nombre con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_117	Introducir nombre con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_119	Introducir apellido paterno con 40 caracteres y con espacios.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_120	Introducir apellido paterno con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_121	Introducir apellido paterno con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.17: Pruebas escenario V - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_122	Introducir apellido paterno con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_123	Introducir apellido paterno con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_124	Introducir apellido materno con 40 caracteres y con espacios.	Se permitirá guardar el dato.	-	No solicitado.
QA_125	Introducir apellido materno con 40 caracteres máximo.	Se permitirá guardar el dato.	-	No solicitado.
QA_126	Introducir apellido materno con 0 caracteres.	No se permitirá guardar el dato.	-	No solicitado.
QA_127	Introducir apellido materno con caracteres especiales.	No se permitirá guardar el dato.	-	No solicitado.
QA_128	Introducir apellido materno con más de 40 caracteres.	No se permitirá guardar el dato.	-	No solicitado.
QA_129	Dar de alta correo electrónico con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_130	Dar de alta correo electrónico con sólo números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_131	Dar de alta correo electrónico alfanumérico.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_132	Dar de alta correo electrónico con guiones.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.18: Pruebas escenario V - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_133	Dar de alta correo electrónico vacío.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_134	Introducir contraseña con mayúsculas y minúsculas.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_135	Introducir contraseña con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_136	Introducir contraseña con caracteres especiales.	Se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_137	Introducir contraseña vacía.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_138	Introducir contraseña de confirmación diferente.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_139	Introducir contraseña con minúsculas y contraseña de confirmación con mayúsculas.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_140	Introducir número de 8 números en el teléfono local.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_141	Introducir número de 8 caracteres especiales en el teléfono local.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.

Cuadro 7.19: Pruebas escenario V - parte 3

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_142	Introducir número de 8 letras en el teléfono local.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_143	Introducir número de 10 números en el celular.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_144	Introducir número de 10 caracteres especiales en el celular.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_145	Introducir número de 10 letras en el celular.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_146	Introducir código postal con 5 dígitos.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_147	Introducir código postal con 4 dígitos.	Se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución**: Se añadirá un 0 al inicio del dato y se permitirá guardar el dato.
QA_148	Introducir código postal con 5 ceros.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_149	Introducir código postal vacío.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_150	Visualizar las colonias asociadas al código postal.	Se mostrarán las colonias asociadas al código postal.	Se muestran las colonias asociadas al código postal.	Cumplidos los requisitos solicitados.
QA_151	Introducir calle con 20 caracteres.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.20: Pruebas escenario V - parte 4

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_152	Introducir calle con sólo números.	Se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución**: Se permitirá guardar el dato en el sistema.
QA_153	Introducir calle con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_154	Introducir calle con letras y números.	Se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución**: Se permitirá guardar el dato en el sistema.
QA_155	Introducir calle con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_156	Introducir calle vacía.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_157	Introducir número con 20 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_158	Introducir número sólo con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_159	Introducir número sólo con letras.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.21: Pruebas escenario V - parte 5

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_160	Introducir número con letras y números.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_161	Introducir número con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_162	Introducir número vacío.	No se permitirá registrar el cliente.	No se permite registrar el cliente.	Cumplidos los requisitos solicitados.
QA_163	Acceder a la aplicación con una contraseña incorrecta.	No se permitirá identificarse al cliente.	No se permitirá identificarse al cliente.	Cumplidos los requisitos solicitados.
QA_164	Acceder a la aplicación con una contraseña en mayúsculas, cuando la contraseña del registro fue en minúsculas.	No se permitirá identificarse al cliente.	No se permitirá identificarse al cliente.	Cumplidos los requisitos solicitados.

Cuadro 7.22: Pruebas escenario V - parte 6

Escenario VI - Compra cliente

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_165	Introducir código postal con 5 dígitos.	Se mostrarán los restaurantes que reparten en ese código postal.	Se muestran los restaurantes que reparten en ese código postal.	Cumplidos los requisitos solicitados.
QA_166	Introducir código postal con 4 dígitos.	Se añadirá un 0 al inicio del dato y se mostrarán los restaurantes que reparten en ese código postal.	Se muestra un mensaje de error y no se realiza la búsqueda.	Corregido. **Solución**: Se añadirá un 0 al inicio del dato y se permitirá buscar restaurantes con ese código postal.
QA_167	Introducir código postal con menos de 5 dígitos.	Se mostrará un mensaje de error y no se realizará la búsqueda.	Se muestra un mensaje de error y no se realiza la búsqueda.	Cumplidos los requisitos solicitados.
QA_168	Introducir código postal vacío.	Se mostrará un mensaje de error y no se realizará la búsqueda.	Se muestra un mensaje de error y no se realiza la búsqueda.	Cumplidos los requisitos solicitados.
QA_169	Introducir código postal con 5 letras.	Se mostrará un mensaje de error y no se realizará la búsqueda.	Se muestra un mensaje de error y no se realiza la búsqueda.	Cumplidos los requisitos solicitados.
QA_170	Seleccionar restaurante.	Se redireccionará al menú del restaurante.	Se redirecciona al menú del restaurante.	Cumplidos los requisitos solicitados.
QA_171	Mostrar menú del restaurante.	Se mostrarán los platos clasificados por categoría y con un botón que permita añadirlos al pedido.	Se muestran los platos clasificados por categoría y con un botón que permite añadirlos al pedido.	Cumplidos los requisitos solicitados.
QA_172	Añadir plato al pedido.	Se mostrará una caja de texto que permita introducir comentarios al plato.	Se muestra una caja de texto que permite introducir comentarios al plato.	Cumplidos los requisitos solicitados.

Cuadro 7.23: Pruebas escenario VI - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_173	Pagar pedido con 0 pesos.	No se permitirá pagar el pedido.	No se muestra el botón de pagar el pedido.	Cumplidos los requisitos solicitados.
QA_174	Pagar pedido sin alcanzar el pedido mínimo del restaurante.	No se permitirá pagar el pedido.	Se permite pagar el pedido.	Corregido. **Solución:** No se muestra el botón de pagar el pedido.
QA_175	Pagar pedido.	Se mostrará la suma del pedido y de los gastos de entrega y se permitirá pagar el pedido.	Se muestra la suma del pedido y de los gastos de entrega y se permite pagar el pedido.	Cumplidos los requisitos solicitados.
QA_176	Visualizar formas de pago.	Se mostrarán las diferentes formas de pago.	Se muestran las diferentes formas de pago.	Cumplidos los requisitos solicitados.
QA_177	Pagar con Webpay.	Se mostrará el formulario de Webpay y se pagará el pedido.	No se carga el formulario de WebPay.	Corregido. **Solución:** Se muestra correctamente el formulario de Webpay.
QA_177	Pagar con invisibleCard.	Se mostrará el código QR del pago.	No se muestra el código QR del pago.	Corregido. **Solución:** Se muestra correctamente el código QR del pago.
QA_177	Pagar con Points2.	Se mostrará un campo donde el usuario introducirá el certificado.	Se muestra un campo donde el usuario introducirá el certificado, pero da error al pagar.	Corregido. **Solución:** Se muestra un campo donde el usuario introducirá el certificado y permite pagar.

Cuadro 7.24: Pruebas escenario VI - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_177	Pagar con invisibleCard.	Se mostrará el código QR del pago.	No se muestra el código QR del pago.	Corregido. **Solución:** Se muestra correctamente el código QR del pago.
QA_177	Pagar con Points2.	Se mostrará un campo donde el usuario introducirá el certificado.	Se muestra un campo donde el usuario introducirá el certificado, pero da error al pagar.	Corregido. **Solución:** Se muestra un campo donde el usuario introducirá el certificado y permite pagar.
QA_178	Introducir certificado de Points2 de 16 dígitos.	Si el certificado es correcto, se pagará el pedido.	Si el certificado es correcto, se paga el pedido.	Cumplidos los requisitos solicitados.
QA_179	Introducir certificado de Points2 de menos de 16 dígitos.	Mostrará un mensaje de error.	Muestra un mensaje de error.	Cumplidos los requisitos solicitados.
QA_180	Introducir certificado de Points2 inválido o sin saldo suficiente.	Mostrará un mensaje de error.	Muestra un mensaje de error.	Cumplidos los requisitos solicitados.

Cuadro 7.25: Pruebas escenario VI - parte 3

Escenario VII - Registro ficticio

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_181	Registrar cliente ficticio.	Se dará de alta un cliente ficticio en el sistema.	No se da de alta ningún cliente ficticio.	Corregido. **Solución:** Se da de alta un cliente ficticio en el sistema.
QA_182	Introducir nombre con 40 caracteres y con espacios.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_183	Introducir nombre con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_184	Introducir nombre con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_185	Introducir nombre con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_186	Introducir nombre con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_187	Introducir apellido paterno con 40 caracteres y con espacios.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_188	Introducir apellido paterno con 40 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_189	Introducir apellido paterno con 0 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.26: Pruebas escenario VII - parte 1

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_190	Introducir apellido paterno con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_191	Introducir apellido paterno con más de 40 caracteres.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_192	Dar de alta correo electrónico con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_193	Dar de alta correo electrónico con sólo números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_194	Dar de alta correo electrónico alfanumérico.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_195	Dar de alta correo electrónico con guiones.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_196	Dar de alta correo electrónico vacío.	No se permitirá registrar el cliente ficticio.	No se permite registrar el cliente ficticio.	Cumplidos los requisitos solicitados.
QA_197	Introducir número de 8 números en el teléfono local.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_198	Introducir número de 8 caracteres especiales en el teléfono local.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_199	Introducir número de 8 letras en el teléfono local.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_200	Introducir número de 10 números en el celular.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.27: Pruebas escenario VII - parte 2

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_201	Introducir número de 10 caracteres especiales en el celular.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_202	Introducir número de 10 letras en el celular.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_203	Introducir código postal con 5 dígitos.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_204	Introducir código postal con 4 dígitos.	Se añadirá un 0 al inicio del dato y se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución**: Se añadirá un 0 al inicio del dato y se permitirá guardar el dato.
QA_205	Introducir código postal con 5 ceros.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_206	Introducir código postal vacío.	No se permitirá registrar el cliente ficticio.	No se permite registrar el cliente ficticio.	Cumplidos los requisitos solicitados.
QA_207	Visualizar las colonias asociadas al código postal.	Se mostrarán las colonias asociadas al código postal.	Se muestran las colonias asociadas al código postal.	Cumplidos los requisitos solicitados.
QA_208	Introducir calle con 20 caracteres.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_209	Introducir calle con sólo números.	Se permitirá guardar el dato.	No se guarda el dato en el sistema.	Corregido. **Solución:** Permite guardar el dato en el sistema.

Cuadro 7.28: Pruebas escenario VII - parte 3

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_210	Introducir calle con sólo letras.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_211	Introducir calle con letras y números.	Se permitirá guardar el dato.	No permite guardar el dato en el sistema.	Corregido. **Solución:** Permite guardar el dato en el sistema.
QA_212	Introducir calle con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_213	Introducir calle vacía.	No se permitirá registrar el cliente ficticio.	No se permite registrar el cliente ficticio.	Cumplidos los requisitos solicitados.
QA_214	Introducir número con 20 caracteres máximo.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_215	Introducir número sólo con números.	Se permitirá guardar el dato.	Se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_216	Introducir número sólo con letras.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_217	Introducir número con letras y números.	No se permitirá guardar el dato.	No se guarda el dato en el sistema.	Cumplidos los requisitos solicitados.
QA_218	Introducir número con caracteres especiales.	No se permitirá guardar el dato.	Se permite guardar con algún caracter especial.	Corregido. **Solución**: Se comprueba si el texto introducido contiene caracteres especiales.
QA_219	Introducir número vacío.	No se permitirá registrar el cliente ficticio.	No se permite registrar el cliente ficticio.	Cumplidos los requisitos solicitados.

Cuadro 7.29: Pruebas escenario VII - parte 4

Escenario VIII - Gestión restaurantes (Rol administrador)

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_220	Acceder a la aplicación con una contraseña incorrecta.	No se permitirá identificarse al administrador.	No se permitirá identificarse al administrador.	Cumplidos los requisitos solicitados.
QA_221	Acceder a la aplicación con una contraseña en mayúsculas, cuando la contraseña del registro fue en minúsculas.	No se permitirá identificarse al administrador.	No se permitirá identificarse al administrador.	Cumplidos los requisitos solicitados.
QA_222	En la pantalla principal mostrar 4 botones: - Logo de Easy Order. - Dar de baja restaurantes. - Dar de alta restaurantes. - Reportes.	Se deberán mostrar las opciones en la página principal.	Se muestran las opciones en la página principal.	Cumplidos los requisitos solicitados.
QA_223	Dar de baja restaurantes.	Se permitirá dar de baja restaurantes en el sistema.	Se permite dar de baja restaurantes en el sistema.	Cumplidos los requisitos solicitados.
QA_224	Dar de alta restaurantes.	Se permitirá dar de alta restaurantes en el sistema.	Se permite dar de alta restaurantes en el sistema.	Cumplidos los requisitos solicitados.

Cuadro 7.30: Pruebas escenario VIII

Escenario IX - Gestión reportes (Rol administrador)

Código de prueba	Descripción	Resultado Esperado	Resultado Obtenido	Observaciones
QA_224	Extraer reportes de los restaurantes.	Se permitirá extraer reportes de los restaurantes.	Se permite extraer reportes de los restaurantes.	Cumplidos los requisitos solicitados.

Cuadro 7.31: Pruebas escenario IX

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo se valorará el trabajo realizado y se obtendrán una serie de conclusiones.

8.1. División del trabajo aplicación web

Inicialmente, se decidió hacer una aplicación web haciendo uso de clases que gestionasen cada modelo y teniendo en el lado del servidor la mayor parte de la lógica de la aplicación. De esta forma, la división del trabajo no estaba clara y los ficheros eran demasiado extensos.

Cuando se empezó a dar la asignatura de *Aplicaciones Web Enriquecidas* nos dimos cuenta que debíamos habernos centrado en el uso de tecnologías más novedosas. Es decir, albergar la mayor parte de la lógica de la aplicación en el lado del cliente haciendo uso de HTML5, CSS3 y Javascript.

Entonces, se decidió rehacer la aplicación focalizando la mayor parte de la lógica en el lado del cliente. Para ello se decidió dividir la estructura en paquetes funcionales, simplificando la longitud de los ficheros y permitiendo obtener una estructura en la que el código está más repartido y organizado. De esta forma, fue más sencilla la repartición de tareas ya que yo me centré en el lado del cliente y Ekaitz Portillo en el lado del servidor.

Cabe destacar que el desarrollo del front-end de la aplicación no ha hecho que esté exenta del desarrollo de la parte web, ya que al finalizar tareas que tenía asignadas, también realizaba tareas de back-end (PHP).

8.2. Tiempo real invertido

La magnitud del proyecto ha sido más extensa de lo planteado en un principio. Esto ha venido dado tanto por la inexperiencia en proyectos con clientes reales como por los

cambios en los requisitos por parte del cliente.

Por otro lado, mientras se estaba realizando este proyecto también se ha estado yendo a clase y trabajando en otra empresa de prácticas. Por lo que ha sido muy difícil gestionar correctamente el tiempo ya que no estaba previsto la realización de prácticas.

El proyecto de la aplicación web aún sigue en marcha, por lo que como se puede observar, el tiempo real invertido es muy superior al tiempo planificado. De cualquier modo, hasta el momento, la tabla de la figura 8.1 se corresponde con las horas dedicadas a cada tarea a día de hoy.

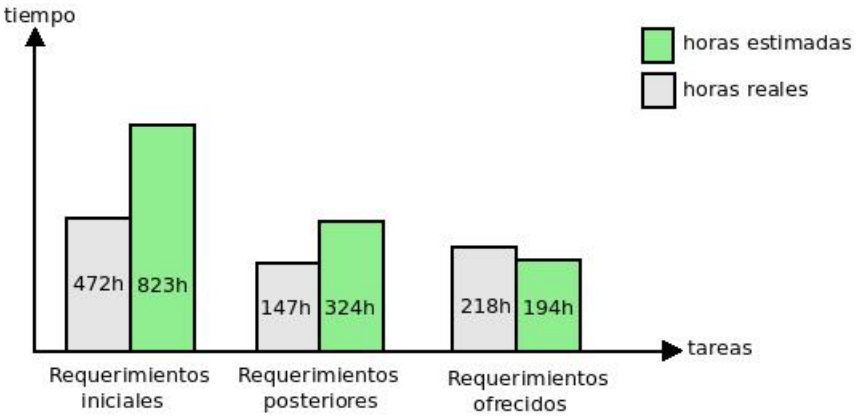


Figura 8.1: Tabla planificación estimada VS real

La aplicación móvil está finalizada, pero no se han cumplido los plazos estipulados ya que aplicación web se ha retrasado considerablemente. Por lo tanto, no se ha comenzado a hacer la aplicación móvil hasta agosto. Y como consecuencia de no haber dado por finalizada la aplicación web, aún no se le ha ofrecido al cliente.

En los requerimientos ofrecidos, se ha tardado menos de las horas estimadas ya que conocía la estructura, es decir, el modelo de datos y la estructura de PHP. Por lo tanto, me ha resultado más fácil de lo esperado hacer el web service ya que también he podido reutilizar parte de la funcionalidad de la aplicación web.

Cuando se planificaron las horas, no contaba con realizar una parte de la funcionalidad el lado del servidor. Lo que por un lado ha hecho que dedique más horas de las estimadas a los requerimientos iniciales y posteriores, y por otro, me ha facilitado el desarrollo de la aplicación móvil más rápidamente.

Los retrasos de la aplicación web no han supuesto ninguna penalización por parte del cliente. Entiendo que se han dado cuenta, al igual que yo, que la gestión de un proyecto

a una distancia de 1000km es cuanto menos dificultoso. Ha habido muchos factores que han afectado al retraso del proyecto, entre ellos:

- Poca comunicación, ya que podían pasar semanas sin que contestasen al e-mail.
- Cambios de opinión en cuanto a la captura de requisitos ya aceptada.
- Solicitud de cambios, que más tarde volverían a pedir deshacerlos.
- Poca documentación para los métodos de pago. Es decir, fue necesario solicitar por e-mail la documentación y al no estar clara, fue necesario realizar varias conferencias para resolver todas las dudas.

Por otra parte, se han cumplido algunos de los riesgos previstos, pero ha surgido otro riesgo no previsto: que se decidiese que la aplicación no se estaba desarrollando bien, y se tuviese que volver a rehacer.

Como se ha indicado anteriormente los riesgos que se ha cumplido han sido la realización de cambios en los requisitos funcionales de la aplicación y la comunicación ineficiente con el cliente, donde se han cumplido los planes de contingencia pero ha retrasado considerablemente el proyecto.

8.3. Cliente

Uno de los mayores problemas encontrados en este proyecto ha sido la comunicación con el cliente.

8.3.1. Gestión de dudas

Pese a que se nos solicitó seguir el procedimiento indicado en el apartado **??**, finalmente no ha sido así. Las dudas se han gestionado de la siguiente forma:

- **Dudas funcionales:** han sido dirigidas al Responsable del Proyecto.
- **Dudas técnicas**: han sido dirigidas al departamento que corresponda.

Se comenzó a seguir este procedimiento debido a que el Responsable del Proyecto, al trabajar también en otros proyectos u otras tareas asignadas por su empresa, la cantidad de tiempo invertida en resolver una duda era demasiado elevada (tiempos superiores a 1 semana), lo cual retrasaba el proyecto de forma considerable.

8.3.2. Reuniones

Pese a que se llegó al acuerdo de hacer las reuniones detalladas en el apartado **??**, finalmente las reuniones realizadas son las siguientes:

- **Reuniones de seguimiento**: Apenas se han hecho 4 reuniones de este tipo, debido a la baja disponibilidad por parte del Responsable del Proyecto.

- **Reuniones funcionales**: Cada vez que se quería modificar una funcionalidad, se nos notificaba por e-mail con lo cual, finalmente, apenas ha habido reuniones de este tipo.

    - **Reuniones técnicas**: Estas reuniones han sido más largas y frecuentes ya que ha sido necesario configurar el servidor de MIT (configuración del entorno para que la aplicación web funcione correctamente). Además, cada vez que era necesario realizar cualquier modificación en el entorno producción (tanto en el código, como en el servidor apache, como en la BBDD) era necesario ponerse en contacto con el departamento técnico y que éstos hiciesen el cambio. Esto era debido a que por la LOPD[1] de México, las personas ajenas a la empresa no pueden tener acceso a los datos de sus clientes.

      Además, MIT tiene un protocolo que evita riesgos de seguridad basado en que personas ajenas a la empresa no puedan acceder a sus servidores, es por ello por lo que se nos ha facilitado una dirección de un servidor FTP al que tenemos acceso donde poder depositar los ficheros que más tarde, los responsables pertinentes validarían y subirán a la ruta del servidor de producción que se les indique.

- **Reuniones de testeo**: Apenas se han hecho 2 reuniones de este tipo, debido a que aún no se ha dado por finalizado el proyecto por parte de MIT.

Cabe destacar que la hora en Ciudad de México es 7 horas inferior (GMT -6[2]) a la hora en Bilbao (GMT +1), ubicación en la que se ha desarrollado el proyecto. Esto implicaba que fuese dificultoso concertar reuniones.

8.4. Mejoras/Ampliaciones

Una de las mejoras posibles en la aplicación móvil es la ampliación de los usuarios a los que va dirigida. Es decir, actualmente a la aplicación sólo pueden acceder clientes ya registrados pero podría ampliarse para que también pudieran registrarse y así poder acceder a la aplicación.

Esta solución, no llevaría mucho tiempo. La forma más rápida de hacerla sería poniendo un link al registro de la página web, pero la opción correcta sería realizar un pequeño apartado que muestre un formulario donde se pueda registrar cualquier cliente. Además, se podría reutilizar parte del código ya desarrollado para la aplicación web.

Otra de las mejoras posibles, y que más necesaria veo tanto en la aplicación web como en la aplicación móvil, es un apartado que te permita modificar la contraseña, o bien,

[1] Acrónimo de *Ley de Protección de Datos*.
[2] GMT o *Greenwich Mean Time* (Tiempo medio de Greenwich) es un estándar de tiempo.

recordarla si la has olvidado.

Esta opción, tan natural de ver en cualquier aplicación, se habló con MIT para añadirla en un futuro, pero aún no han pedido su desarrollo. Además, es una opción que otorgaría mucho beneficio a ambas aplicaciones y no sería muy costosa de realizar.

Otra de las mejoras posibles, podría ser la realización de otra aplicación móvil en la que los restaurantes puedan gestionar los pedidos recibidos al instante desde cualquier dispositivo móvil.

Esta opción, puede resultar algo costosa ya que sería necesario realizar otra aplicación y hacer uso de servicios que permitan ejecutar la aplicación en segundo plano y así, aunque el restaurante no tenga abierta la aplicación, le lleguen notificaciones.

Por otro lado, una de las mejoras posibles en la aplicación web es la realización de un apartado que permita gestionar la base de datos. Creo que esto sería muy útil ya que en MIT cada vez que se necesita realizar cualquier gestión con la base de datos es necesario solicitar un permiso específico , por lo que podría haber un encargado del panel de gestión de Easy Order que pudiese interactuar con la base de datos desde una interfaz amigable y dentro de la propia aplicación. Esta mejora, en mi opinión sería muy beneficiosa pero sería muy costosa de desarrollar.

8.5. Conclusión final

Para poder realizar este proyecto, ha sido necesario investigar cómo crear páginas web y aplicaciones Android ya que este proyecto salió antes de darse las asignaturas en la universidad. Además, en el caso de la aplicación web, una vez casi terminada se decidió rehacer la aplicación para hacerla más actual, es decir, haciendo uso de tecnologías novedosas (HTML5, CSS3 y Javascript).

Se sabía de antemano que esto iba a retrasar el proyecto, pero también se sabía que se iba a mejorar y facilitar el mantenimiento de la aplicación en un futuro. Además, se decidió aprovechar para adquirir una plantilla y así mejorar el diseño de la aplicación y hacerla adaptable a cualquier dispositivo. Esto se debe a que la plantilla adquirida está orientada a aplicaciones web de comida y proporciona diferentes funciones en Javascript que facilitan el correcto funcionamiento de esta en cualquier dispositivo.

Por lo tanto, se realizó una reunión con MIT donde se expusieron los motivos de esta mejora, y se aceptó. Cabe añadir que esta mejora también nos iba a servir para poder promocionarnos en un futuro hacia otras empresas ya que se puede usar como port-folio.

De cualquier modo, a pesar de la carga de trabajo y que la remuneración no ha sido del todo satisfactoria, ha sido un proyecto muy enriquecedor que me ha proporcionado

una muy buena experiencia. Por lo que a día de hoy, a punto de acabar la universidad y trabajando en una empresa, me estoy planteando trabajar en mi tiempo libre en otros proyectos de este tipo.

Además, se espera que en los próximos meses se llegue a un acuerdo con MIT para la realización del mantenimiento de la aplicación web. Y así obtener una remuneración aceptable y recompensar el esfuerzo de este último año.

Por último, exponer la URL de la aplicación web: `https://www.easyorder.com.mx`. De la cual no se puede asegurar su disponibilidad debido a que está siendo administrada por MIT.

Bibliografía

[1] *Ed Burnette, Hello, Android: Introducing Google's Mobile Development Platform, 2009*

[2] *Harold A. Linstone y Murray Turoff, Delphi Method: Techniques and Applications, 1975*

[3] *Juan Diego Gauchat, El gran libro de HTML5, CSS3 y Javascript, 2012*

[4] *Mark Pilgrim, HTML5: Up and Running. Dive into the Future of Web Development, 2010*

[5] *David Flanagan, JavaScript: The Definitive Guide, 4th Edition, 2001*

[6] *Thomas Powell, Ajax: The Complete Reference, 2008*

[7] *Michael K. Glass, Yann Le Scouarnec, Elizabeth Naramore, Gary Mailer, Jeremy Stolz y Jason Gerner, Desarrollo Web con PHP, Apache y MySQL, 2004*

[8] *Ramez Elmasri y Shamkant B. Navathe, Fundamentos de Sistemas de Bases de Datos, 2007*

Anexo I - Casos de uso extendidos

Casos de uso extendidos Web

Identificarse

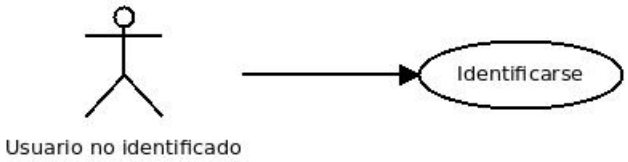


Figura 2: CU Identificarse

- **Nombre**: Identificarse.
- **Descripción**: Permite a un usuario no identificado identificarse en el sistema.
- **Actores**: Usuario no identificado.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El usuario no identificado pasa el ratón por la entrada de menú "*Ingresar*". Véase la figura 3a.
  2. El usuario no identificado introduce el nombre de usuario y contraseña y hace clic en "*Acceder*".

     [Si el usuario y contraseña son correctos]

     a) Se identifica el usuario en el sistema.

     [Si el usuario o contraseña no son correctos]

     a) Se muestra un mensaje de error. Véase la figura 3b.

- **Requisitos no funcionales:** Ninguno.

- ≻ **Post-condiciones**: Se almacenará el usuario identificado en las variables de sesión.
- ≻ **Interfaces**:

(a) Entrada de menú "*Ingresar*" (b) Mensaje de error "*Ingresar*"

Figura 3: Interfaces CU Identificarse

Registrar cliente

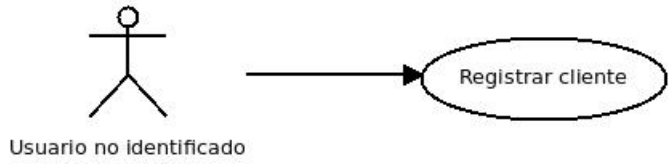


Figura 4: CU Registrar cliente

- **Nombre**: Registrar cliente.
- **Descripción**: Permite a un usuario no identificado registrarse como cliente para poder hacer un seguimiento de sus pedidos.
- **Actores**: Usuario no identificado.
- **Precondiciones**: Ninguno.
- **Flujo de eventos**:

  1. El usuario no identificado hace clic en sobre la entrada de menú "*Registrarse*". Véase la figura 5a.
  2. El usuario no identificado introducirá los datos con los que quiere registrarse. Véase la figura 5b.

     [Si el usuario no identificado hace clic en "Enviar"]

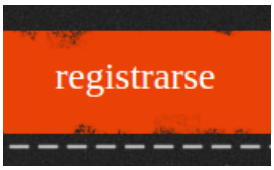
     - *[Si los datos son correctos]*
       - *[Si el e-mail no está registrado en el sistema]*
         a) Se añadirá el cliente al sistema.
         b) Se mostrará un mensaje con los datos del registro. Véase la figura 6a.
       - *[Si el e-mail está registrado en el sistema]*
         a) Se muestra un mensaje de error. Véase la figura 6a.
     - *[Si los datos no son correctos]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 6c.

     [Si el usuario no identificado hace clic en "Vaciar"]

     a) Se borrará el contenido de todos los campos. Véase la figura 5b.

- **Requisitos no funcionales:** Se enviará un e-mail al usuario para confirmar el registro.
- **Post-condiciones**: Se añadirá un nuevo cliente al sistema.

≻ **Interfaces**:

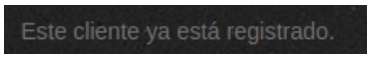


(a) Entrada de menú "*Registrarse*"

(b) Formulario de registro cliente

Figura 5: Interfaces CU Registrar cliente

(a) Mensaje registro correcto



(b) Error email ya registrado

(c) Error campo incorrecto

Figura 6: Interfaces CU Registrar cliente

Registrar restaurante

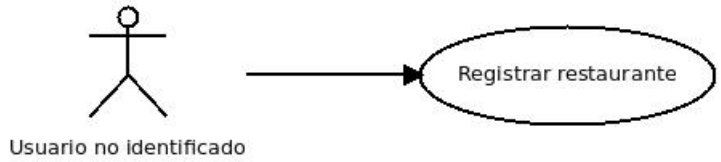


Figura 7: CU Registrar restaurante

- ≻ **Nombre**: Registrar restaurante.
- ≻ **Descripción**: Permite a un usuario no identificado registrarse como restaurante para poder ofrecer sus productos a través de la aplicación.
- ≻ **Actores**: Usuario no identificado.
- ≻ **Precondiciones**: Ninguno.
- ≻ **Flujo de eventos**:

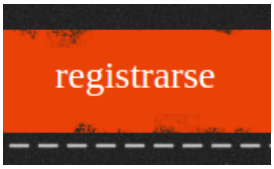
  1. El usuario no identificado hace clic sobre la entrada de menú "*Registrarse*". Véase la figura 8a.
  2. El usuario no identificado hace clic sobre "*¿Eres un restaurante? ¡Regístrate aquí!*". Véase la figura 8b.
  3. El usuario no identificado introducirá los datos con los que quiere registrarse. Véase la figura 8c.
     [Si el usuario no identificado hace clic en "Enviar"]
     - *[Si los datos son correctos]*
       - *[Si el e-mail no está registrado en el sistema]*
         a) Se añadirá un nuevo restaurante al sistema.
         b) Se mostrará un mensaje con los datos del registro. Véase la figura 9a.
       - *[Si el e-mail está registrado en el sistema]*
         a) Se mostrará un aviso de error. Véase la figura 9b.
     - *[Si los datos no son correctos]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 9c.

     [Si el usuario no identificado hace clic en "Vaciar"]
     a) Se borrará el contenido de todos los campos. Véase la figura 8a.

- ≻ **Requisitos no funcionales:** Se comunicará con un servicio web de MIT para obtener el código de sucursal.

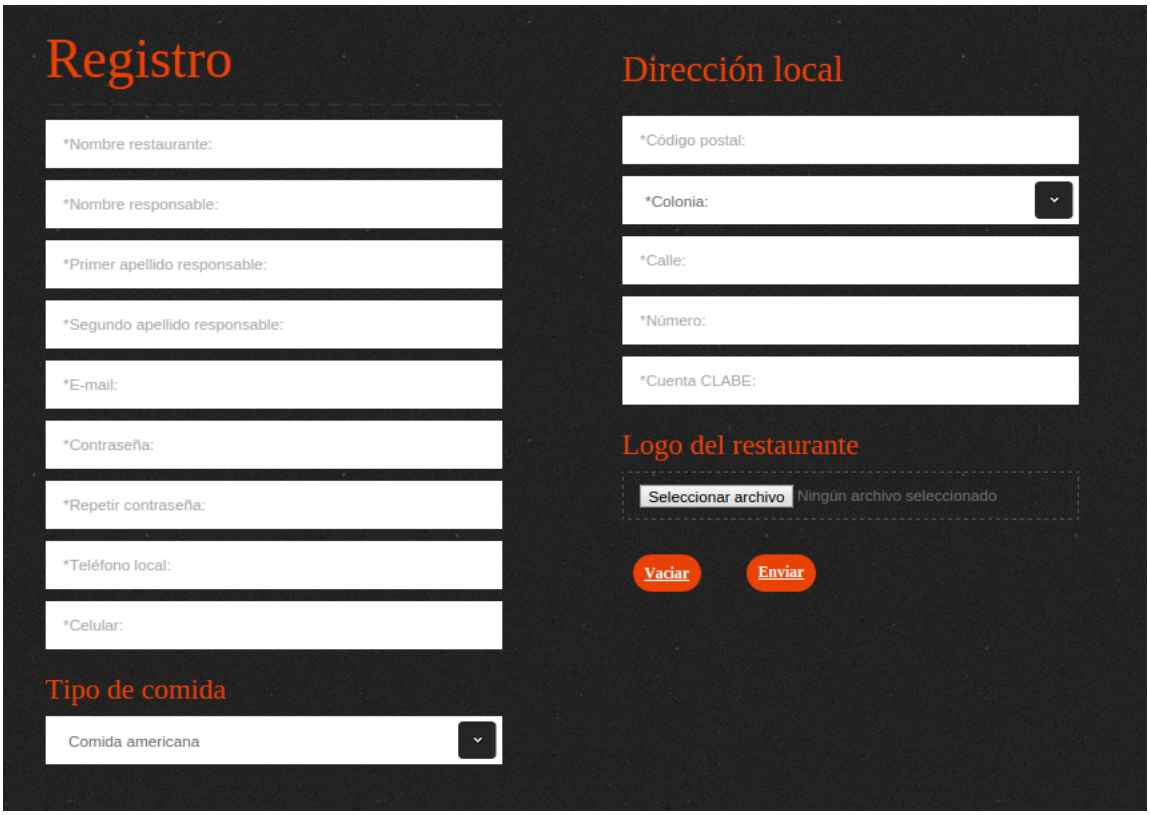
➢ **Post-condiciones**: Se añadirá un nuevo restaurante al sistema.

➢ **Interfaces**:



(a) Entrada de menú "*Registrarse*"

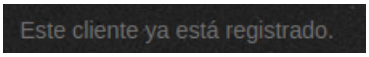
(b) Opción registro restaurante



(c) Formulario de registro restaurante

Figura 8: Interfaces CU Registrar restaurante

(a) Mensaje registro correcto



(b) Error email ya registrado

(c) Error campo incorrecto

Figura 9: Interfaces CU Registrar restaurante

Búsqueda restaurantes

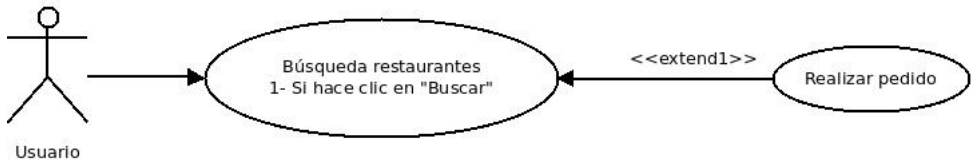


Figura 10: CU Búsqueda restaurantes

- **Nombre**: Búsqueda restaurantes.
- **Descripción**: Permite a un usuario buscar restaurantes para poder realizar un pedido.
- **Actores**: Usuario.
- **Precondiciones**: Ninguno.
- **Flujo de eventos**:

1. El usuario hace clic sobre la entrada de menú "*Home*". Véase la figura 11a.
   [Si hace clic en "Localízame"]
   a) Se cargará el código postal de la zona en la que se encuentre el usuario.
2. El usuario introducirá el código postal de la zona en la que quiere recibir el pedido. Véase la figura 11b.
   [Si hace clic en "Buscar"]
   - *[Si el código postal es correcto]*
     - *[Si reparte algún restaurante en esa zona]*
       a) Se muestran los restaurantes que reparten en esa zona. Véase la figura 11c.
       ◇ *[Si hace clic en "Ver menú"]*
       a) Ver Extend CU "*Realizar pedido*" (8.5).
     - *[Si no reparte ningún restaurante en esa zona]*
       a) Se mostrará un mensaje de información. Véase la figura 11d.
   - *[Si el código postal no es correcto]*

   a) Se mostrará un mensaje de error. Véase la figura 11e.

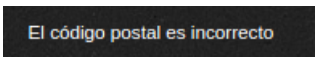
- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se registrará el pedido en el sistema.
- **Interfaces**:

(a) Entrada de menú "*Home*"

(b) Formulario código postal

(c) Listado restaurantes zona

(d) Mensaje no restaurantes en código postal



(e) Error código postal incorrecto

Figura 11: Interfaces CU Búsqueda restaurantes

Extend Realizar pedido

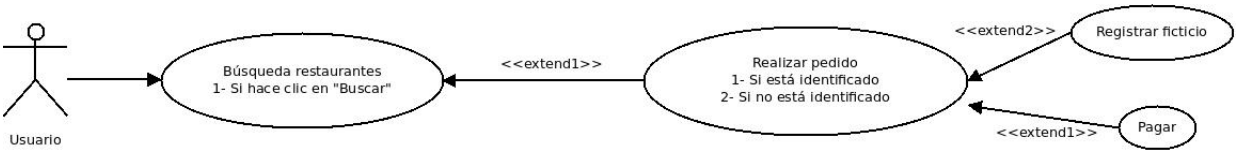


Figura 12: Extend CU Realizar pedido

- **Nombre**: Realizar pedido.
- **Descripción**: Permite a un usuario realizar un pedido seleccionando los platos de un restaurante.
- **Actores**: Usuario.
- **Precondiciones**: Ninguno.
- **Flujo de eventos**:

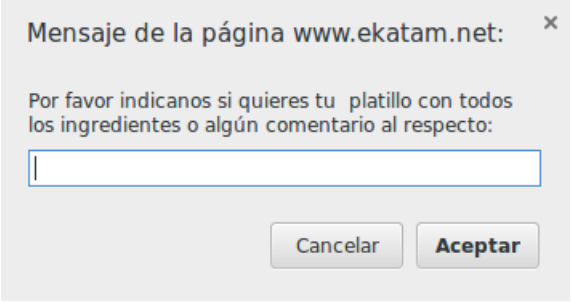
  [Si el restaurante tiene menús disponible]

  1. Se mostrarán los menús del restaurante. Véase la figura 13a.
  2. El usuario añade un comentario sobre el plato seleccionado. Véase la figura 13b.
  3. El usuario añade los platos del restaurante a su pedido. Véase la figura 13c.
     [Si hace clic en "Pagar"]
     - *[Si el usuario está identificado]*
       a) Ver Extend CU "*Pagar*" (8.5).
     - *[Si el usuario no está identificado]*
       a) Ver Extend CU "*Registrar ficticio*" (8.5).

  [Si el restaurante no tiene menús]

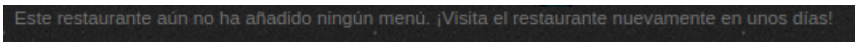
  1. Se mostrará un mensaje de información. Véase la figura 13d.
- **Requisitos no funcionales:** Se asignará un identificador de ticket único en todo el sistema.
- **Post-condiciones**: Se registrará el pedido en el sistema.
- **Interfaces**:

(a) Listado menús restaurante



(b) Comentario plato

(c) Pedido



(d) Mensaje restaurante sin menús

Figura 13: Interfaces Extend CU Realizar pedido

Extend Registrar ficticio

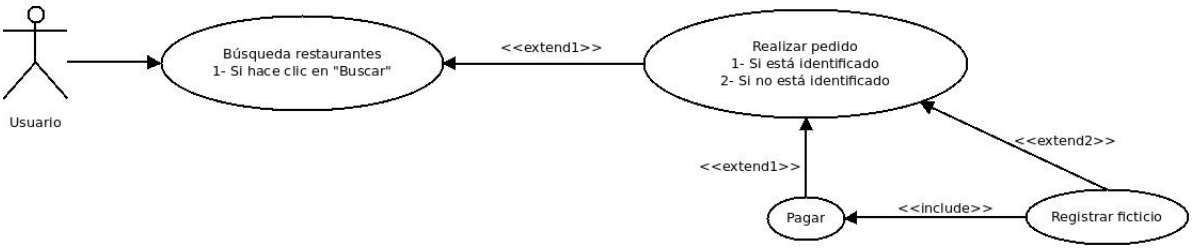


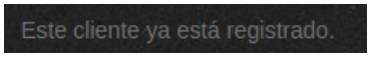
Figura 14: Extend CU Registrar ficticio

- **Nombre**: Registrar ficticio.
- **Descripción**: Permite a un usuario identificarse para poder pagar un pedido.
- **Actores**: Usuario.
- **Precondiciones**: Ninguno.
- **Flujo de eventos**:

  1. El usuario no identificado introducirá los datos necesarios para poder recibir el pedido. Véase la figura 15a.
     [Si el usuario no identificado hace clic en "Enviar"]
     - *[Si los datos son correctos]*
       - *[Si el e-mail está registrado en el sistema]*
         a) Se mostrará un mensaje de error. Véase la figura 15b.
       - *[Si el e-mail no está registrado en el sistema]*
         a) Se registrará un cliente ficticio en el sistema.
         b) Ver CU "*Pagar*" (8.5).
     - *[Si los datos no son correctos]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 15c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se registrará el cliente ficticio en el sistema.
- **Interfaces**:

(a) Formulario registro ficticio



(b) Error email ya registrado

(c) Error campo incorrecto

Figura 15: Interfaces Extend CU Realizar pedido

Gestionar pedidos

Figura 16: CU Gestionar pedidos

- ≻ **Nombre**: Gestionar pedidos.
- ≻ **Descripción**: Permite a un cliente gestionar los pedidos realizados.
- ≻ **Actores**: Cliente.
- ≻ **Precondiciones**: Ninguna.
- ≻ **Flujo de eventos**:

1. El cliente hace clic sobre la entrada de menú "*Pedidos*". Véase la figura 17a.
   [Si hay pedidos]
   a) Se muestran los pedidos. Véase la figura 17b.
      [Si hace clic en "*+ Info*"*]*
      1) Se mostrará la información del pedido.
      [Si hay pedidos sin pagar]
      - *[Si* hace clic en "Pagar"*]*
        1) Ver Extend CU "*Pagar*" (8.5).
      - *[Si hace clic en "Cancelar pedido"]*
        1) Se borrará el pedido del sistema.

      [Si hay pedidos pagados]
      - *[Si hace clic en "Calificar"]*
        1) Ver Extend CU "*Calificar*" (8.5).

   [Si no hay pedidos]
   a) Se mostrará un mensaje de información. Véase la figura .

- ≻ **Requisitos no funcionales:** Ninguno.
- ≻ **Post-condiciones**: Se modificará el pedido en el sistema.

➢ **Interfaces**:

(a) Entrada de menú "*Pedidos*"

(b) Listado pedidos cliente

Figura 17: Interfaces CU Gestionar pedidos

Extend Pagar

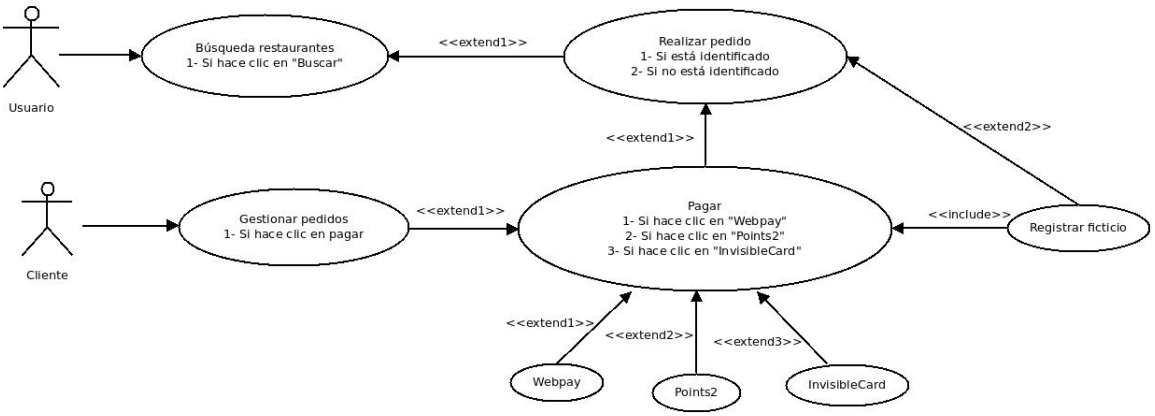


Figura 18: Extend CU Pagar

- **Nombre**: Pagar.
- **Descripción**: Permite a un usuario pagar el pedido realizado.
- **Actores**: Cliente y Usuario.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. Se mostrará el resumen del pedido. Véase la figura 19a.
     [Si el cliente/usuario hace clic en "Webpay"]
     a) Ver Extend CU *"Webpay"* (8.5).
     [Si el cliente/usuario hace clic en "InvisibleCard"]
     a) Ver Extend CU *"InvisibleCard"* (8.5).
     [Si el cliente/usuario hace clic en "Points2"]
     a) Ver Extend CU *"Points2"* (8.5).

- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta. Además, el sistema enviará un e-mail al cliente confirmando el pedido.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema.
- **Interfaces**:

(a) Resumen pedido

Figura 19: Interfaces Extend CU Pagar

Extend Webpay

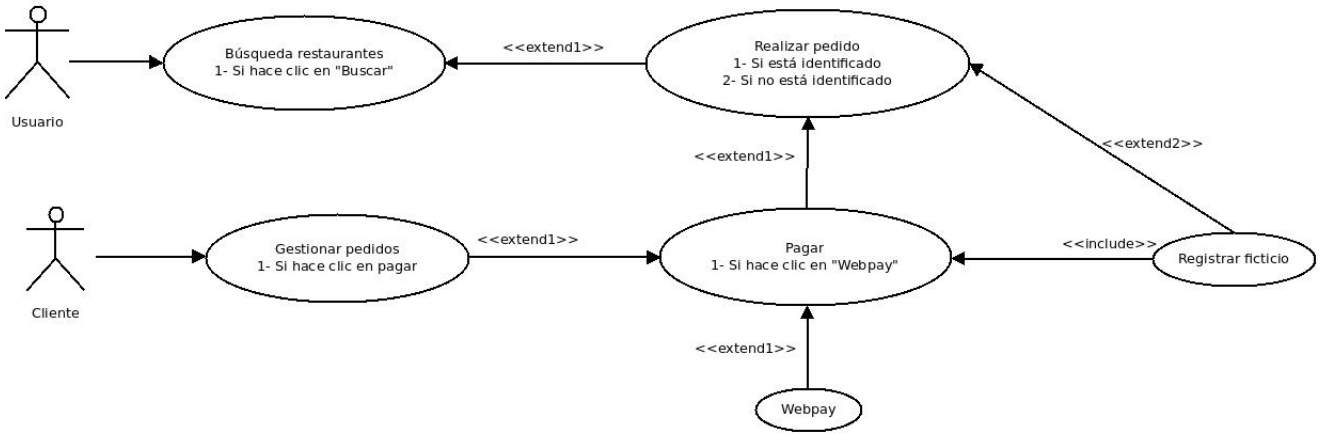


Figura 20: Extend CU Webpay

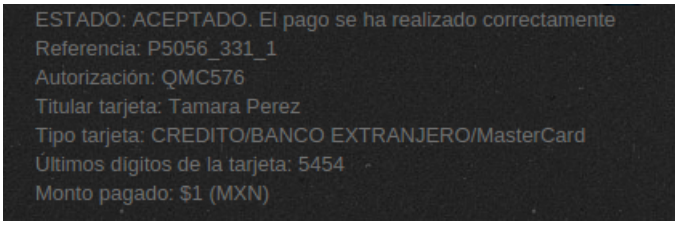
- **Nombre**: Webpay.
- **Descripción**: Permite a un usuario pagar el pedido realizado con Webpay.
- **Actores**: Cliente y Usuario.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El cliente/usuario introducirá los datos solicitados para el pago. Véase la figura 21a.
     [Si hace clic en "Pagar"]
     - *[Si los datos son correctos]*
       a) Se registrará el pedido como pagado en el sistema.
       b) Se mostrará un mensaje con los datos de la transacción. Véase la figura 22a.
     - *[Si los datos no son correctos]*
       a) Se mostrará un mensaje de error. Véase la figura 22b.

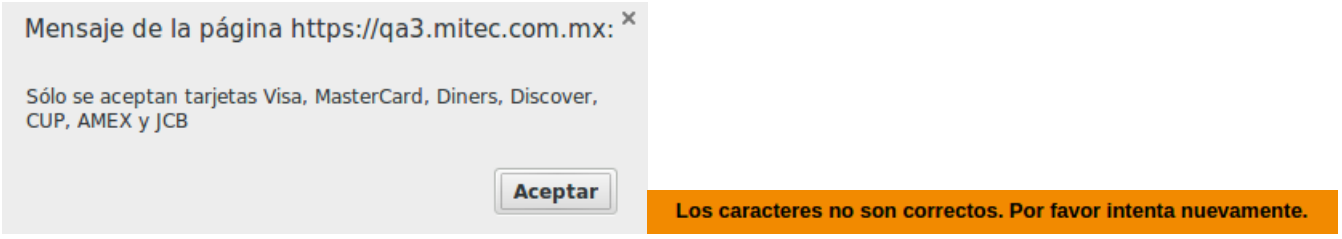
- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema. Además el sistema enviará un e-mail al cliente confirmando el pedido.
- **Interfaces**:

(a) Formulario Webpay

Figura 21: Interfaces Extend CU Webpay



(a) Mensaje transacción



(b) Error datos Webpay

Figura 22: Interfaces Extend CU Webpay

Extend Points2

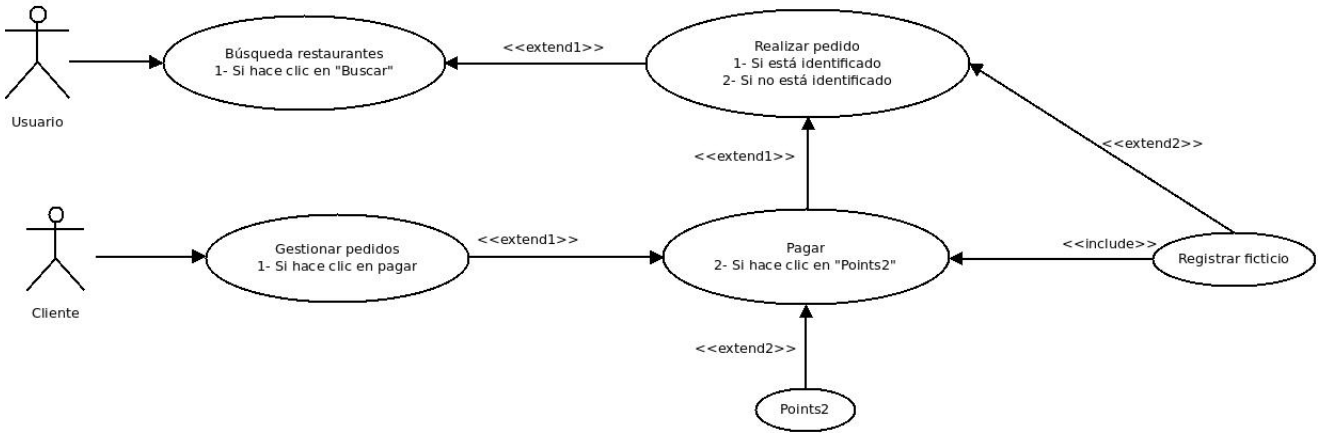


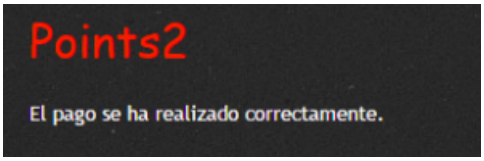
Figura 23: Extend CU Points2

- **Nombre**: Points2.
- **Descripción**: Permite a un usuario pagar el pedido realizado con Points2.
- **Actores**: Cliente y Usuario.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El cliente/usuario introduce un certificado. Véase la figura 24a.
     [Si hace clic en "Pagar"]
     - *[Si el certificado es correcto]*
       a) Se registrará el pedido como pagado en el sistema.
       b) Se mostrará un mensaje con los datos de la transacción. Véase la figura 24b.
     - *[Si el certificado no es correcto]*
       a) Se mostrará un mensaje de error. Véase la figura 24c.

- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema. Además el sistema enviará un e-mail al cliente confirmando el pedido.
- **Interfaces**:

(a) Formulario Points2



(b) Mensaje transacción

(c) Error certificado

Figura 24: Interfaces Extend CU Points2

Extend Invisiblecard

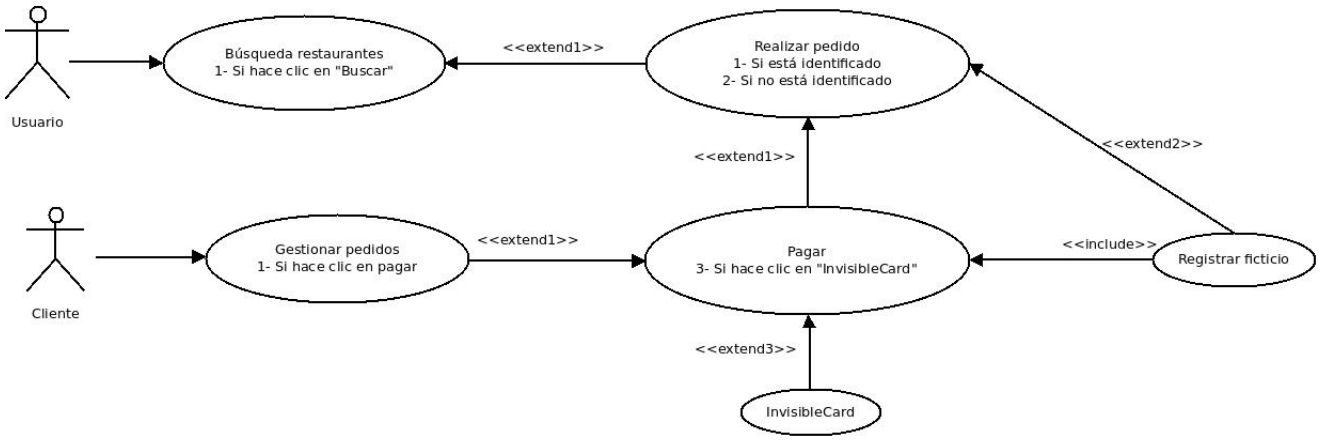


Figura 25: Extend CU InvisibleCard

- **Nombre**: Points2.
- **Descripción**: Permite a un usuario pagar el pedido realizado con InvisibleCard.
- **Actores**: Cliente y Usuario.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. Se mostrará un código QR.
  2. Cuando el cliente/usuario pague desde la aplicación móvil *"InvisibleCard"* se registrará el pedido como pagado en el sistema.

- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema. Además el sistema enviará un e-mail al cliente confirmando el pedido.
- **Interfaces**: Ninguna.

Extend Calificar

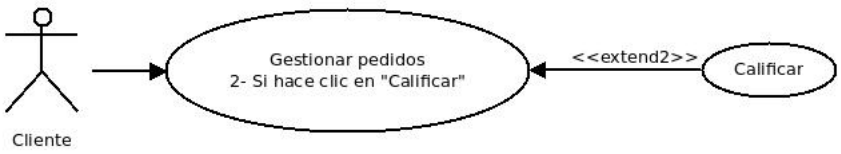


Figura 26: Extend CU Calificar

- **Nombre**: Calificar.
- **Descripción**: Permite a un cliente valorar los pedidos realizados.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

1. El cliente hace clic sobre la entrada de menú "*Pedidos*". Véase la figura 27a.
   [Si hay pedidos]
   a) Se muestran los pedidos. Véase la figura 27b.
   b) El cliente selecciona un pedido y hace clic en "*Calificar*".
   c) El cliente introduce la valoración. Véase la figura 28a.
      [Si el cliente *hace clic en "Validar"]*
      - *[Si los datos son correctos]*
        1) Se añadirá la valoración al sistema.
        2) Se mostrará un mensaje con la confirmación del envío de la valoración. Véase la figura 28b.
      - *[Si los datos no son correctos]*
        1) Se marcarán los campos erróneos en rojo. Véase la figura 28c.

      [Si el cliente *hace clic en "Cancelar edición"]*
      1) Volverá a la pantalla de edición inicial. Véase la figura 27b.

   [Si no hay pedidos]
   a) Se mostrará un mensaje de información. Véase la figura 28d.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se añadirá la calificación del pedido al sistema.
- **Interfaces**:

(a) Entrada de menú "*Pedidos*"

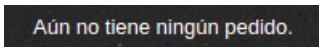
(b) Pedidos cliente

Figura 27: Interfaces Extend CU Calificar

(a) Formulario valoración

(b) Confirmación valoración

(c) Error campo incorrecto



(d) Cliente sin pedidos

Figura 28: Interfaces Extend CU Calificar

Modificar menú

Figura 29: CU Modificar menú

- **Nombre**: Modificar menú.
- **Descripción**: Permite a un restaurante modificar su menú pudiendo dar de alta/baja categorías y platos.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

1. El restaurante pasa el ratón por la entrada de menú "*Personalizar*" y hace clic en "*Menú*". Véase la figura 30a.

   [Si el restaurante hace clic en "Dar de alta categoría"]

   a) Ver Extend CU "*Dar de alta categoría*" (8.5).

   [Si hay categorías creadas]

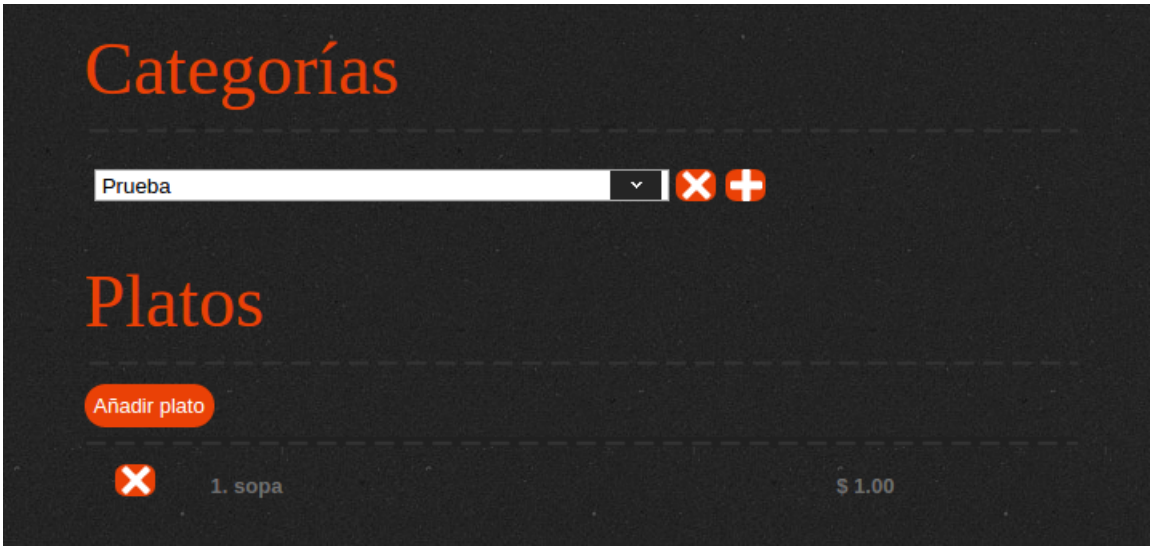
   a) Se muestran las categorías y platos. Véase la figura 30b.
   [Si el restaurante hace clic en "Dar de baja categoría"]
   1) La categoría seleccionada se dará de baja en el sistema.
   [Si el restaurante hace clic en "Dar de alta plato"]
   1) Ver Extend CU "*Dar de alta plato*" (8.5).
   [Si hay platos creados]
   - *[Si el restaurante hace clic en "Dar de baja plato"]*
     1) El plato seleccionado se dará de baja en el sistema.

   [Si no hay categorías creadas]

   a) Se muestra un mensaje de información. Véase la figura 30c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se dará de alta/baja el plato o categoría seleccionada.
- **Interfaces**:

(a) Entrada de menú "*Menú*" dentro de "*Personalizar*"



(b) Listado categorías y platos

(c) Restaurante sin categorías

Figura 30: Interfaces CU Modificar menú

Extend Dar de alta categoría

Figura 31: Extend CU Dar de alta categoría

- **Nombre**: Dar de alta categoría.
- **Descripción**: Permite a un restaurante dar de alta nuevas categorías en su menú.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El restaurante introducirá un nombre para la categoría. Véase la figura 32a.
     [Si el restaurante hace clic en "Validar"]
     - *[Si el dato es correcto]*
       a) Se añadirá la categoría al sistema.
     - *[Si el dato no es correcto]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 32b.

     [Si el restaurante hace clic en "Cancelar edición"]

     a) Volverá a la pantalla de edición inicial. Véase la figura 32c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se añadirá la categoría al sistema.
- **Interfaces**:

(a) Formulario nueva categoría

(b) Error campo incorrecto

(c) Listado categorías

Figura 32: Interfaces CU Dar de alta categoría

Extend Dar de alta plato

Figura 33: Extend CU Dar de alta plato

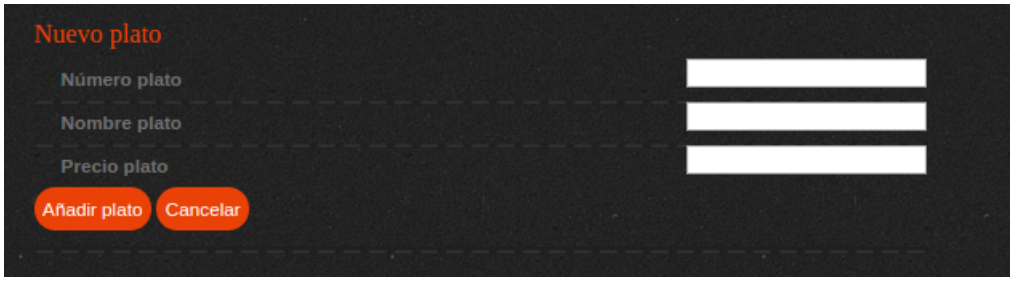
- **Nombre**: Dar de alta plato.
- **Descripción**: Permite a un restaurante dar de alta nuevos platos en su menú bajo una categoría.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El restaurante selecciona una categoría
  2. El restaurante introducirá los datos del plato. Véase la figura 34a.
     [Si el restaurante hace clic en "Validar"]
     - *[Si los datos son correctos]*
       a) Se añadirá el plato al sistema.
     - *[Si los datos no son correctos]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 34b.

     [Si el restaurante hace clic en "Cancelar edición"]

     a) Volverá a la pantalla de edición inicial. Véase la figura 34c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se añadirá el plato al sistema.
- **Interfaces**:



(a) Formulario nuevo plato

(b) Error campo incorrecto

(c) Listado platos

Figura 34: Interfaces CU Dar de alta plato

Modificar datos

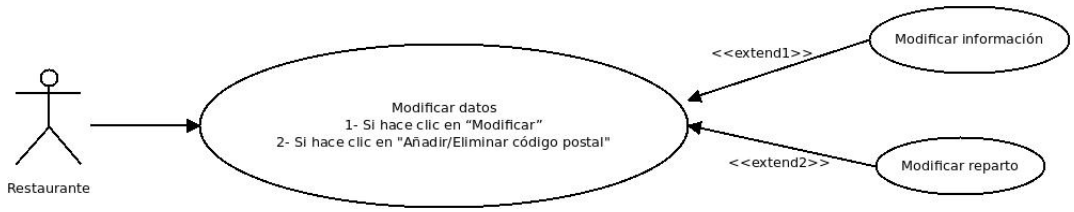


Figura 35: CU Modificar datos

- **Nombre**: Modificar datos.
- **Descripción**: Permite a un restaurante modificar su información pudiendo cambiar sus datos y gestionar los códigos postales en los que repartir.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El restaurante pasa el ratón por la entrada de menú "*Personalizar*" y hace clic en "*Datos*". Véase la figura 36a.
     [Si el restaurante hace clic en "Modificar" en algún campo]
     a) Ver Extend CU "*Modificar información*" (8.5).
     [Si el restaurante hace clic en "Añadir código postal"]
     a) Ver Extend CU "*Modificar reparto*" (8.5).
     [Si el restaurante hace clic en "Eliminar código postal"]
     a) Ver Extend CU "*Modificar reparto*" (8.5).

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se modificará/añadirá el dato que el restaurante desee.
- **Interfaces**:

(a) Entrada de menú "*Datos*" dentro de "*Personalizar*"

Figura 36: Interfaces CU Modificar datos

Extend Modificar información

Figura 37: Extend CU Modificar información

- **Nombre**: Modificar información.
- **Descripción**: Permite a un restaurante modificar su información.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El restaurante introducirá un nuevo valor para ese campo. Véase la figura 38a.
     [Si el restaurante hace clic en "Validar"]
     - *[Si el dato es correcto]*
       a) Se modificará el valor de ese campo en el sistema.
     - *[Si el dato no es correcto]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 38b.

     [Si el restaurante hace clic en "Cancelar edición"]

     a) Volverá a la pantalla de edición inicial. Véase la figura 38c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se modificará el dato que el restaurante desee.
- **Interfaces**:

(a) Formulario modificación campo

(b) Error campo incorrecto

(c) Información restaurante

Figura 38: Interfaces CU Modificar información

Extend Modificar reparto

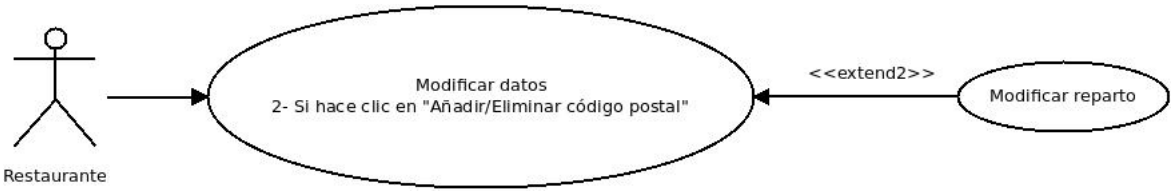


Figura 39: Extend CU Modificar reparto

- **Nombre**: Modificar reparto.
- **Descripción**: Permite a un restaurante añadir/eliminar códigos postales de reparto.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  [Si el restaurante hace clic en "Añadir código postal"]

  1. El restaurante introducirá un nuevo código postal. Véase la figura 40.
     [Si el restaurante hace clic en "Validar"]
     - *[Si el código postal es correcto]*
       a) Se añadirá el código postal de reparto al sistema.
     - *[Si el código postal no es correcto]*
       a) Se marcarán los campos erróneos en rojo. Véase la figura 41a.

     [Si el restaurante hace clic en "Cancelar edición"]

     a) Volverá a la pantalla de edición inicial. Véase la figura 41b.

  [Si hay códigos postales]

  - *[Si el restaurante hace clic en "Eliminar código postal"]*
    - *[Si hay más de un código postal en la lista]*
      1. Se eliminará el código postal del sistema.
    - *[Si no hay más códigos postales en la lista]*
      1. Se mostrará un mensaje de error. Véase la figura 41c.

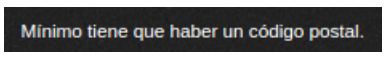
- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se añadirá/eliminará el código postal introducido por el restaurante en el sistema.

≻ **Interfaces**:

Figura 40: Formulario añadir código postal

(a) Error campo incorrecto

(b) Información restaurante



(c) Error mínimo códigos postales

Figura 41: Interfaces CU Modificar información

Gestionar pedidos

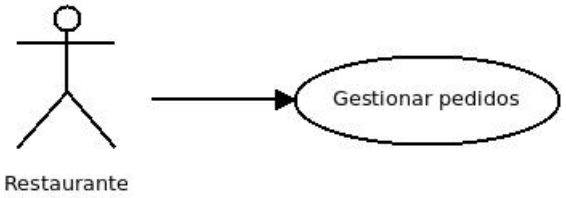


Figura 42: CU Gestionar pedidos

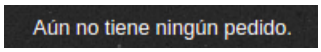
- **Nombre**: Gestionar pedidos.
- **Descripción**: Permite a un restaurante cambiar el estado de los pedidos.
- **Actores**: Restaurante.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

1. El restaurante hace clic en *"Pedidos"*. Véase la figura 43a.
   [Si hay pedidos]
   a) Se muestran los pedidos del restaurante. Véase la figura 43b.
   b) El restaurante selecciona un pedido y hace clic en "Modificar".
   c) El restaurante seleccionará otro estado.
      [Si el restaurante hace clic en "Validar"]
      1) Se modificará el estado del pedido en el sistema.
      [Si el restaurante hace clic en "Cancelar edición"]
      1) Volverá a la pantalla de edición inicial. Véase la figura 43b.
   [Si no hay pedidos]
   a) Se mostrará un mensaje de información. Véase la figura 43c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se actualizarán los estados de los pedidos seleccionados.
- **Interfaces**:

(a) Entrada de menú *"Pedidos"*

(b) Listado pedidos restaurante



(c) Error sin pedidos

Figura 43: Interfaces CU Gestionar pedidos

Desconectarse

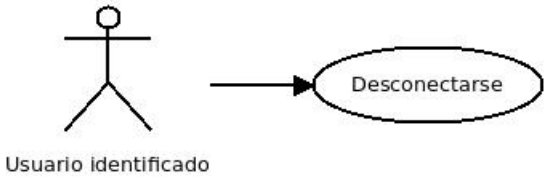


Figura 44: CU Desconectarse

- **Nombre**: Desconectarse
- **Descripción**: Permite a un usuario identificado desconectarse.
- **Actores**: Usuario identificado.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El usuario identificado hace clic en la entrada de menú *"Desconectarse"*. Véase la figura 45a.
  2. Volverá a la pantalla principal. Véase la figura 45b.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se eliminarán las variables de sesión del usuario identificado.
- **Interfaces**:

(a) Entrada de menú *"Desconectarse"*

(b) Página principal

Figura 45: Interfaces CU Desconectarse

Gestionar restaurantes

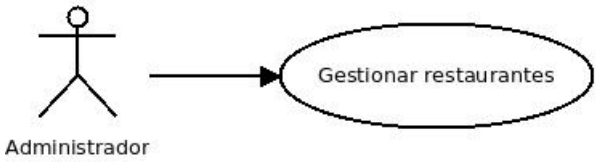


Figura 46: CU Gestionar restaurantes

- **Nombre**: Gestionar restaurantes.
- **Descripción**: Permite al administrador dar de alta/baja restaurantes.
- **Actores**: Administrador.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El administrador pasa el ratón por la entrada de menú "*Gestionar restaurantes*". Véase la figura 47a.
     [Si hay restaurantes pendientes]
     a) Se muestran los restaurantes del sistema. Véase la figura 47b.
     [Si hace clic en "Dar de alta"]
     1) Se dará de alta el restaurante en el sistema.
     [Si hace clic en "Dar de baja"]
     1) Se dará de baja el restaurante en el sistema.
     [Si no hay restaurantes pendientes]
     a) Se muestra un mensaje de información. Véase la figura 47c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se dará de alta/baja el restaurante en el sistema.
- **Interfaces**:

(a) Entrada de menú *"Gestionar restaurantes"*

(b) Listado restaurantes

(c) Mensaje sin restaurantes

Figura 47: Interfaces CU Gestionar restaurantes

Extraer reportes

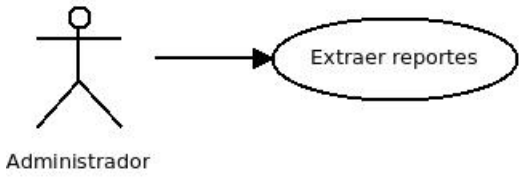


Figura 48: CU Extraer reportes

- **Nombre**: Extraer reportes.
- **Descripción**: Permite al administrador extraer reportes sobre los restaurantes.
- **Actores**: Administrador.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:
  1. El administrador hace clic en sobre la entrada de menú "*Reportes*". Véase la figura 49b.
  2. Se muestran los tipos de reporte. Véase la figura 49b.
  3. El administrador hace clic sobre el botón "*Imprimir*".
  4. Se descargará en el ordenador del usuario el reporte en formato "*.xls*".
- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Ninguna.
- **Interfaces**:

(a) Entrada de menú *"Reportes"*

(b) Listado pedidos restaurante

Figura 49: Interfaces CU Extraer reportes

Casos de uso extendidos Android

Identificarse

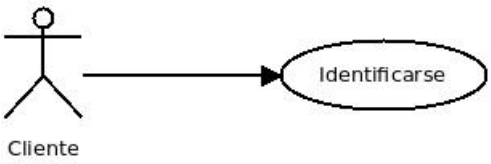
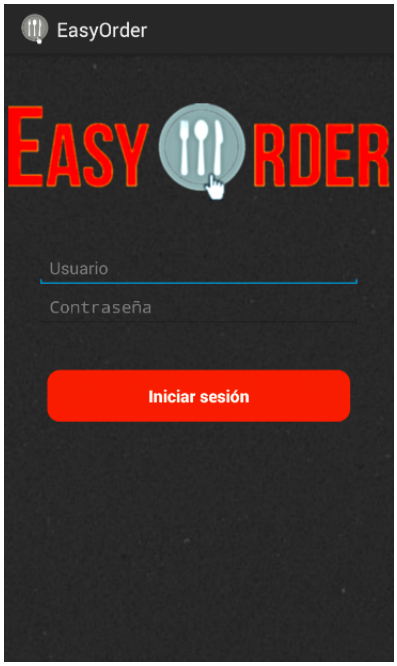


Figura 50: CU Identificarse

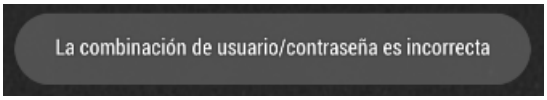
- **Nombre**: Identificarse.
- **Descripción**: Permite a un cliente identificarse en el sistema.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El cliente introduce el nombre de usuario y contraseña y hace clic en "*Iniciar sesión*". Véase la figura 51a.
     [Si el usuario y contraseña son correctos]
     a) Se identifica el cliente en el sistema.
     [Si el usuario o contraseña no son correctos]
     a) Se muestra un mensaje de error. Véase la figura 51b.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se almacenará el usuario identificado en la aplicación.
- **Interfaces**:



(a) Pantalla identificación



(b) Error identificación

Figura 51: Interfaces CU Identificarse

Desconectarse

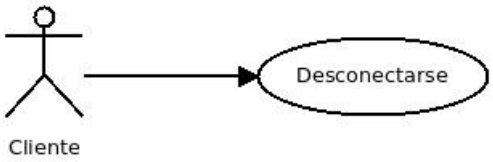
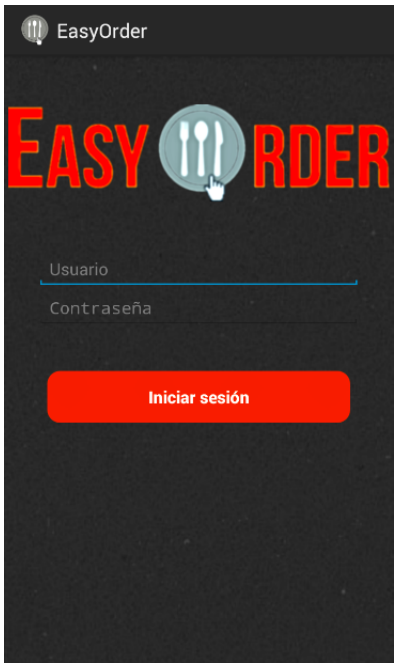


Figura 52: CU Desconectarse

- **Nombre**: Desconectarse
- **Descripción**: Permite a un cliente desconectarse.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:
  1. El cliente hace clic en *"Desconectarse"*. Véase la figura 53a.
  2. Volverá a la pantalla de inicio de sesión. Véase la figura 53b.
- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se eliminará el usuario identificado de la aplicación.
- **Interfaces**:

(a) Botón desconectarse



(b) Pantalla identificación

Figura 53: Interfaces CU Desconectarse

Gestionar pedidos

Figura 54: CU Gestionar pedidos

- ≻ **Nombre**: Gestionar pedidos.
- ≻ **Descripción**: Permite a un cliente gestionar los pedidos realizados.
- ≻ **Actores**: Cliente.
- ≻ **Precondiciones**: Ninguna.
- ≻ **Flujo de eventos**:

1. El cliente hace clic sobre la entrada de menú "*Pedidos*". Véase la figura 55a.
   [Si hay pedidos]
   a) Se muestran los pedidos. Véase la figura 55b.
   [Si hay pedidos sin pagar]
   - *[Si* hace clic en "Pagar"*]*
     1) Ver CU "*Pagar*" (8.5).
   - *[Si hace clic en "Cancelar pedido"]*
     1) Se mostrará un mensaje de información. Véase la figura 55c.
     2) Se borrará el pedido del sistema.

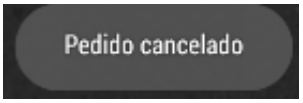
   [Si hay pedidos pagados]
   - *[Si hace clic en "Calificar"]*
     1) Ver Extend CU "*Calificar*" (8.5).

   [Si no hay pedidos]
   a) Se mostrará un mensaje de información. Véase la figura 55d.

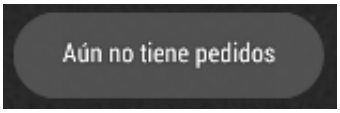
- ≻ **Requisitos no funcionales:** Ninguno.
- ≻ **Post-condiciones**: Se modificará el pedido en el sistema.
- ≻ **Interfaces**:

(a) Botón desconectarse

(b) Gestión pedidos



(c) Mensaje pedido cancelado



(d) Mensaje sin pedidos

Figura 55: Interfaces CU Gestionar pedidos

Pagar

Figura 56: CU Pagar

- **Nombre**: Pagar.
- **Descripción**: Permite a un cliente pagar el pedido realizado.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El cliente hace clic en *"Carro"*. Véase la figura 57a.
  2. Se mostrará el resumen del pedido. Véase la figura 57b.
     [Si el cliente hace clic en "Points2"]
     a) Ver Extend CU *"Points2"* (8.5).

- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta. Además, el sistema enviará un e-mail al cliente confirmando el pedido.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema.
- **Interfaces**:

(a) Botón carro

(b) Resumen pedido

Figura 57: Interfaces CU Pagar

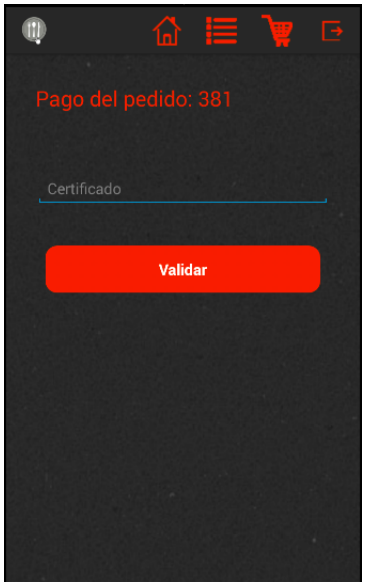
Extend Points2

Figura 58: Extend CU Points2

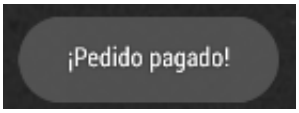
- **Nombre**: Points2.
- **Descripción**: Permite a un cliente pagar el pedido realizado con Points2.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

  1. El cliente introduce un certificado. Véase la figura 59a.
     [Si hace clic en "Validar"]
     - *[Si el certificado es correcto]*
       a) Se mostrará un mensaje de información. Véase la figura 59b.
       b) Se registrará el pedido como pagado en el sistema.
     - *[Si el certificado no es correcto]*
       a) Se mostrará un mensaje de error. Véase la figura 59c.

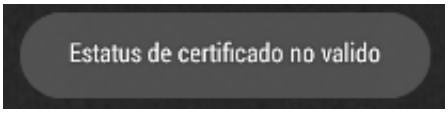
- **Requisitos no funcionales:** Se comunica con los servidores de MIT para comprobar si la transacción es correcta.
- **Post-condiciones**: Se registrará el pedido como pagado en el sistema. Además el sistema enviará un e-mail al cliente confirmando el pedido.
- **Interfaces**:



(a) Pantalla Points2



(b) Mensaje pedido pagado



(c) Mensaje error pedido

Figura 59: Interfaces Extend CU Points2

Extend Calificar

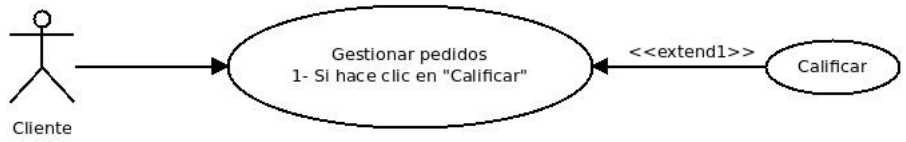
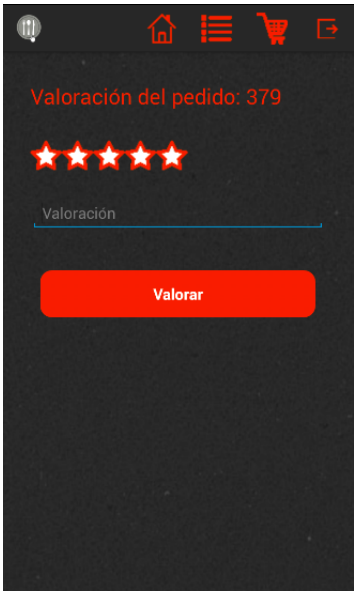


Figura 60: Extend CU Calificar

- **Nombre**: Calificar.
- **Descripción**: Permite a un cliente valorar los pedidos realizados.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

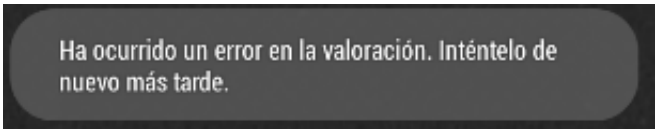
  1. El cliente introduce la valoración. Véase la figura 61a.
     [Si el cliente *hace clic en "Valorar"]*
     - *[Si los datos son correctos]*
       a) Se añadirá la valoración al sistema.
       b) Se mostrará un mensaje de información. Véase la figura 61b.
     - *[Si los datos no son correctos]*
       a) Se mostrará un mensaje de error. Véase la figura 61c.

- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se añadirá la calificación del pedido al sistema.
- **Interfaces**:



(a) Valoración

(b) Mensaje valoración correcta



(c) Mensaje valoración incorrecta

Figura 61: Interfaces Extend CU Calificar

Búsqueda restaurantes

Figura 62: CU Búsqueda restaurantes

- **Nombre**: Búsqueda restaurantes.
- **Descripción**: Permite a un cliente buscar restaurantes para poder realizar un pedido.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

1. El usuario hace clic sobre la entrada de menú "*Home*". Véase la figura 63a.
   [Si hace clic en "Localízame"]
   - *[Si está activada la localización]*
     a) Se cargará el código postal de la zona en la que se encuentre el usuario.
   - *[Si no está activada la localización]*
     a) Se mostrará un mensaje de error. Véase la figura 64a.
2. El usuario introducirá el código postal de la zona en la que quiere recibir el pedido. Véase la figura 63b.
   [Si hace clic en "Buscar"]
   - *[Si el código postal es correcto]*
     - *[Si reparte algún restaurante en esa zona]*
       a) Se muestran los restaurantes que reparten en esa zona. Véase la figura 63c.
       ◇ *[Si hace clic en "Ver menú"]*
       a) Ver Extend CU "*Realizar pedido*" (8.5).
     - *[Si no reparte ningún restaurante en esa zona]*
       a) Se mostrará un mensaje de información. Véase la figura 64b.
   - *[Si el código postal no es correcto]*

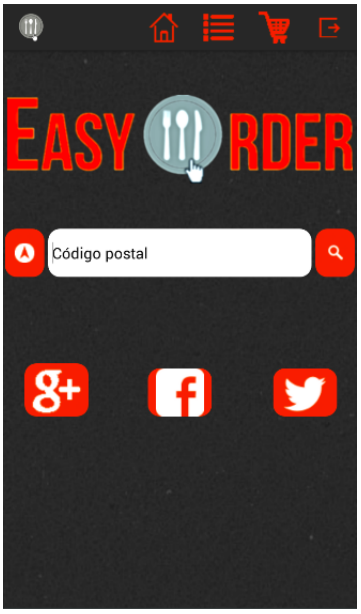
   a) Se mostrará un mensaje de error. Véase la figura 64c.

- **Requisitos no funcionales:** Ninguno.

➢ **Post-condiciones**: Se registrará el pedido en el sistema.

➢ **Interfaces**:

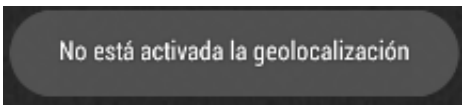
(a) Botón Home



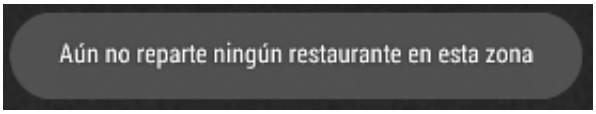
(b) Pantalla principal

(c) Pantalla búsqueda restaurantes

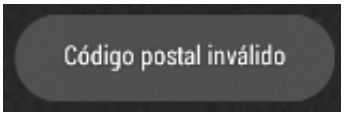
Figura 63: Interfaces CU Búsqueda restaurantes



(a) Mensaje error localización



(b) Mensaje información restaurantes



(c) Mensaje error código postal inválido

Figura 64: Interfaces CU Búsqueda restaurantes

Extend Realizar pedido

Figura 65: Extend CU Realizar pedido

- **Nombre**: Realizar pedido.
- **Descripción**: Permite a un cliente realizar un pedido seleccionando platos de un restaurante.
- **Actores**: Cliente.
- **Precondiciones**: Ninguna.
- **Flujo de eventos**:

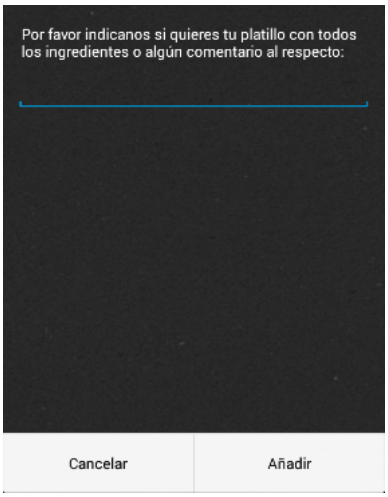
  [Si el restaurante tiene menús]

  1. Se mostrarán los menús del restaurante. Véase la figura 66a.
  2. El usuario añade un comentario al plato seleccionado. Véase la figura 66b.

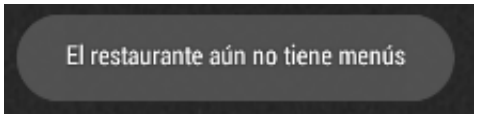
  [Si el restaurante no tiene menús]

  1. Se mostrará un mensaje de información. Véase la figura 66c.
- **Requisitos no funcionales:** Ninguno.
- **Post-condiciones**: Se almacenará el plato en la aplicación.
- **Interfaces**:

(a) Menús restaurante



(b) Añadir comentario plato



(c) Mensaje información sin menú

Figura 66: Interfaces Extend CU Realizar pedido

Anexo II - Diagramas de secuencia

Web

Identificarse

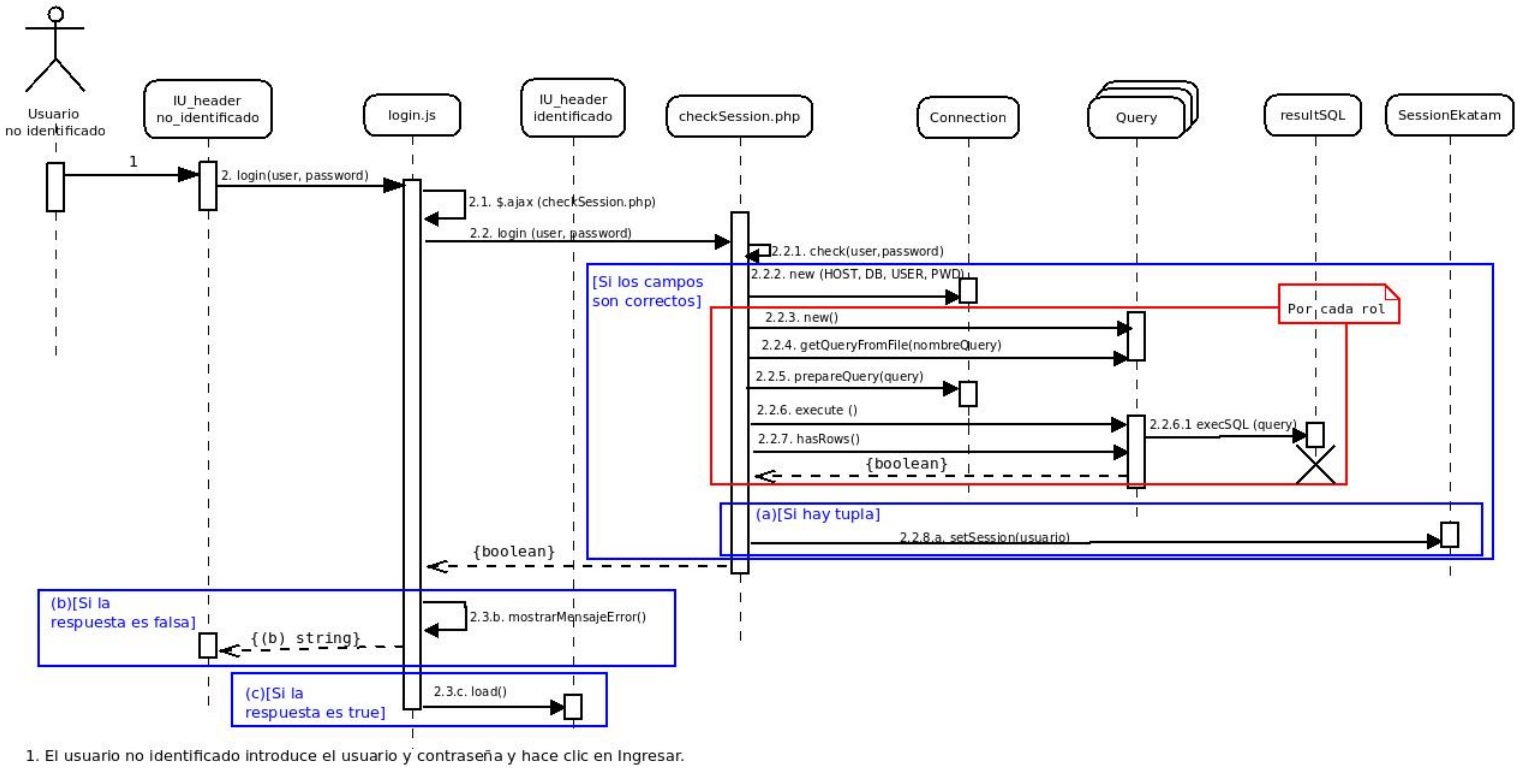


Figura 67: Diagrama de secuencia identificarse

En la aplicación web, hay tres posibles roles: restaurante, cliente y administrador. Por lo tanto, sus respectivas consultas son las siguientes:

- Query rol restaurante: *SELECT nombreRest FROM restaurante WHERE emailRest = (?) AND passRest = (?)*
- Query rol cliente: *SELECT nombreCli FROM cliente WHERE emailCli = (?) AND passCli = (?)*
- Query rol administrador: *SELECT userAdmin FROM administrador WHERE userAdmin = (?) AND passAdmin = (?)*

Registrar cliente

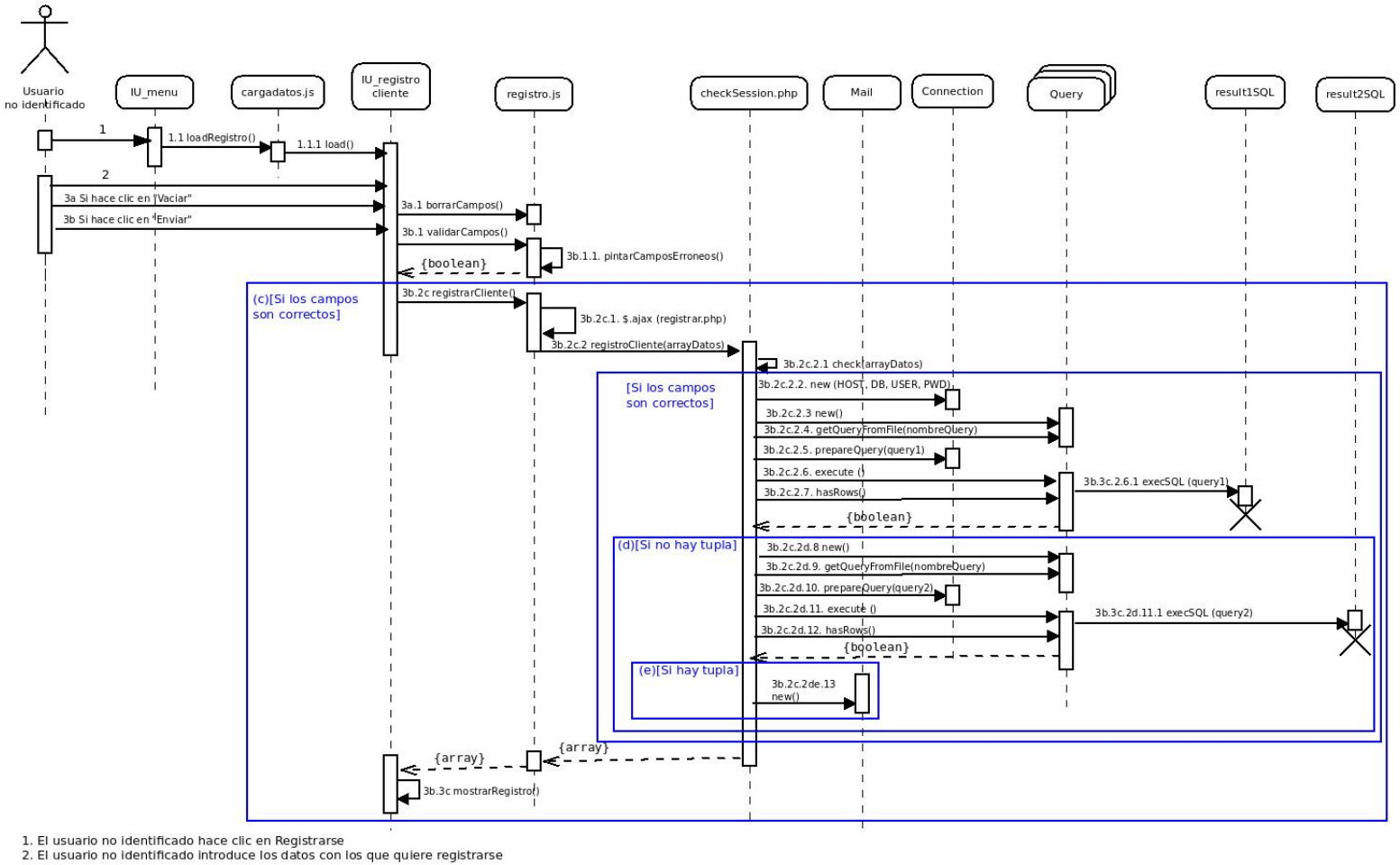
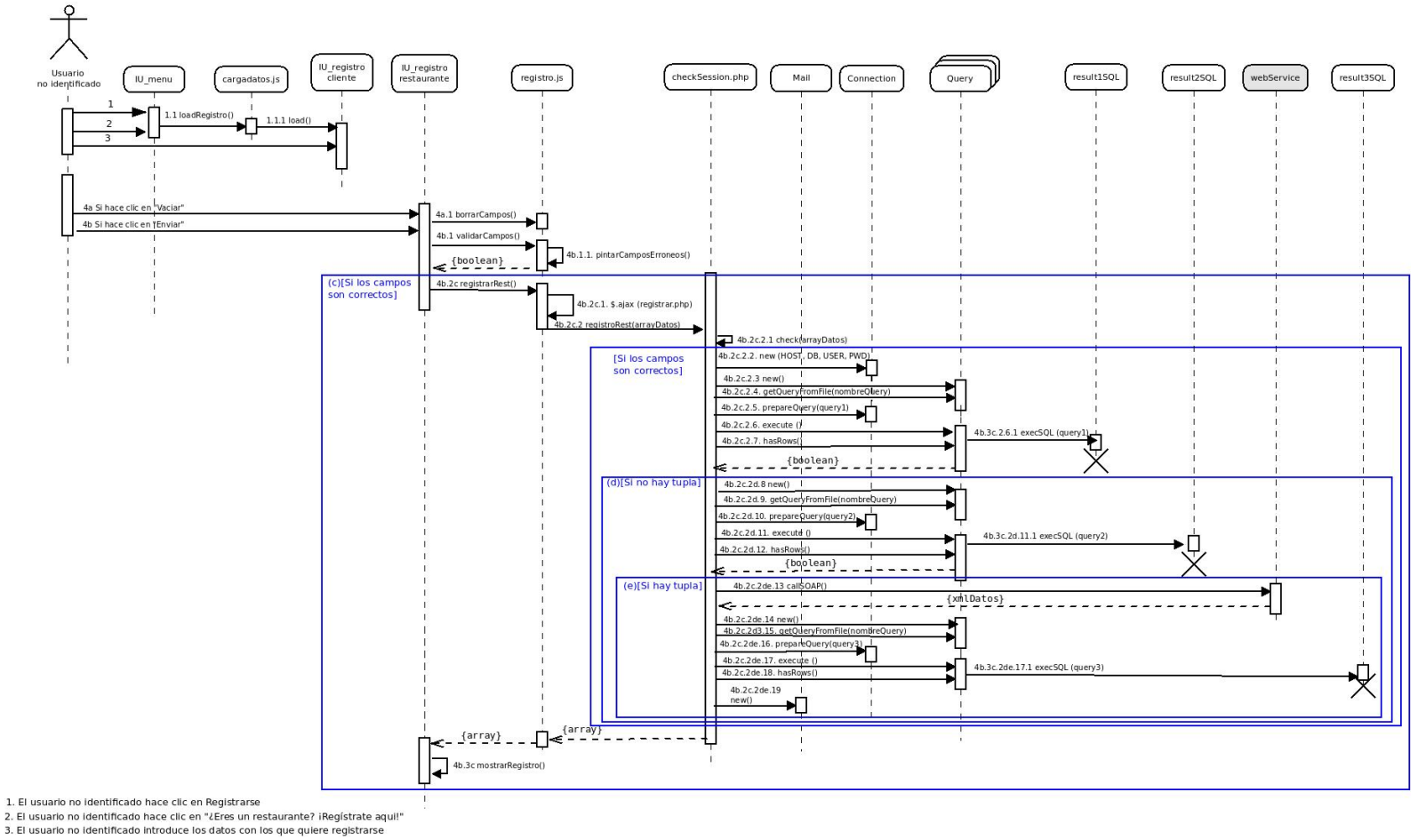


Figura 68: Diagrama de secuencia registrar cliente

- Query1: *SELECT emailCli AS email FROM cliente WHERE emailCli = (?) AND validado = 1 UNION SELECT emailRest AS email FROM restaurante WHERE emailRest = (?)*
- Query2: *INSERT INTO cliente(emailCli, passCli, nombreCli, apellidosCli, telefono-Cli, numCP, validado, idCalle, calle2Cli, numCasa, celular) VALUES ((?),(?),(?),(?),(?),(?),(?), (SELECT idCalle FROM calle A NATURAL JOIN localizacion B WHERE A.nombreCalle = (?) AND B.numCP = (?)),(?),(?),(?))*

Registrar restaurante



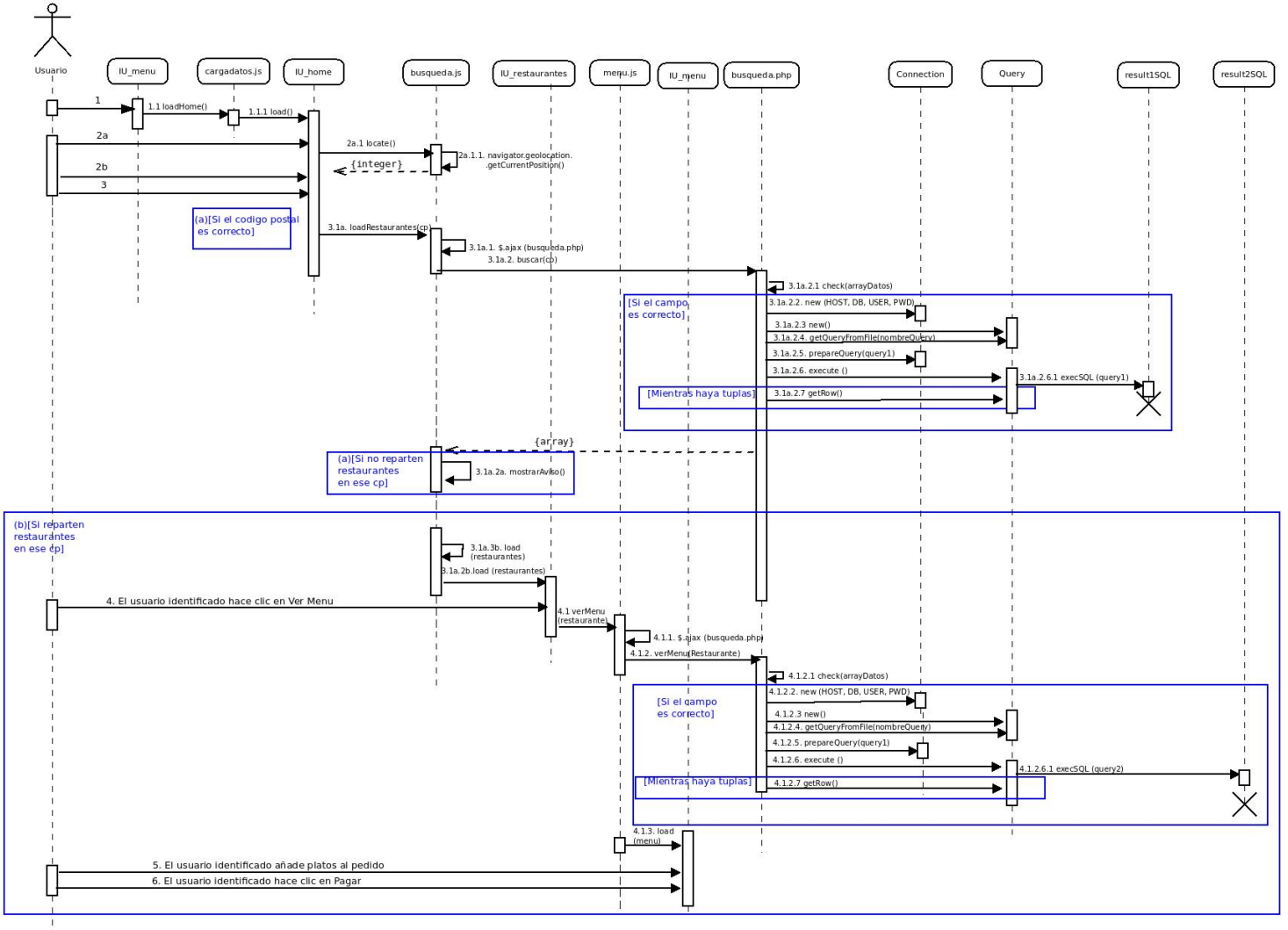
1. El usuario no identificado hace clic en Registrarse
2. El usuario no identificado hace clic en "¿Eres un restaurante? ¡Regístrate aquí!"
3. El usuario no identificado introduce los datos con los que quiere registrarse

Figura 69: Diagrama de secuencia registrar restaurante

- ≻ Query1: *SELECT emailCli AS email FROM cliente WHERE emailCli = (?) AND validado = 1 UNION SELECT emailRest AS email FROM restaurante WHERE emailRest = (?)*
- ≻ Query2: *INSERT INTO restaurante(emailRest, passRest, nombreRest, idCalle, numLocal, idEstiloComida, numCP, responsable, apellido1,apellido2, calle2Rest, numCuenta, hashedMail) VALUES ((?),(?),(?), (SELECT idCalle FROM calle A NATURAL JOIN localizacion B WHERE A.nombreCalle = (?) AND B.numCP = (?)),(?),(SELECT idEstiloComida FROM estilocomida WHERE tipoComida = (?)), (?),(?),(?),(?),(?),(?),(?))*
- ≻ Query3: SELECT extract(hour from horaCerrar) as cierre, hashedMail, nombreEstado as estado, nombreMunicipio as municipio, nombreCalle as colonia, numLocal as numExterior, numCP, responsable, apellido1, apellido2, calle2Rest as calle, telefonoRest FROM restaurante NATURAL JOIN calle NATURAL JOIN municipio NATURAL JOIN telefonosrest WHERE emailRest = (?) and tipoTelf='local'

** Clase webService propia de MIT.

Búsqueda restaurantes



1. El usuario identificado hace clic en Home
2a. Si el usuario identificado hace clic en Localizame
2b. Si el usuario identificado introduce un codigo postal
3. El usuario identificado hace clic en Buscar

Figura 70: Diagrama de secuencia búsqueda restaurantes

- Query1: *SELECT nombreRest, descripcionRest, hashedMail, gastosEntrega, pedidoMin, horaAbrir, horaCerrar, nombreCalle, numLocal, tipoComida, rating FROM reparte rep INNER JOIN restaurante rest ON rest.emailRest = rep.emailRest NATURAL JOIN calle INNER JOIN estilocomida e ON rest.idEstiloComida = e.idEstiloComida WHERE rep.numCP = (?) AND validado = 1*

- Query2: *SELECT nombreRest, descripcionRest, gastosEntrega, pedidoMin, horaAbrir, horaCerrar, nombreCalle, numLocal, tipoComida, rating, nombreCat as categoria, concat(p.numCarta,'. ', p.nombrePlato) as plato, p.precio, p.idPlato FROM restaurante rest NATURAL JOIN calle INNER JOIN estilocomida e ON rest.idEstiloComida = e.idEstiloComida natural join plato p inner join categoria c on p.idCat = c.idCat where rest.hashedMail = (?) and c.activo = 1 order by nombreCat, p.numCarta*

Desconectarse

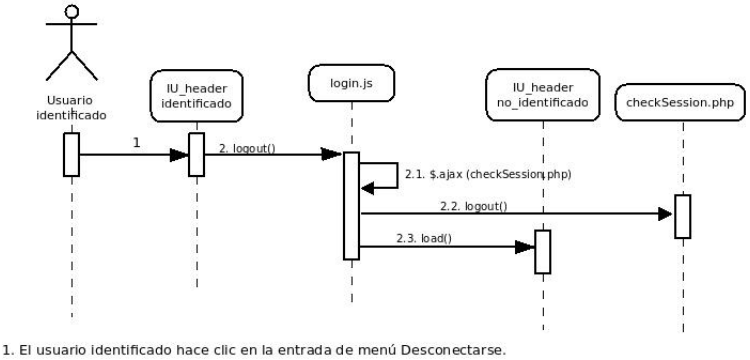


Figura 71: Diagrama de secuencia desconectarse

Android

Identificarse

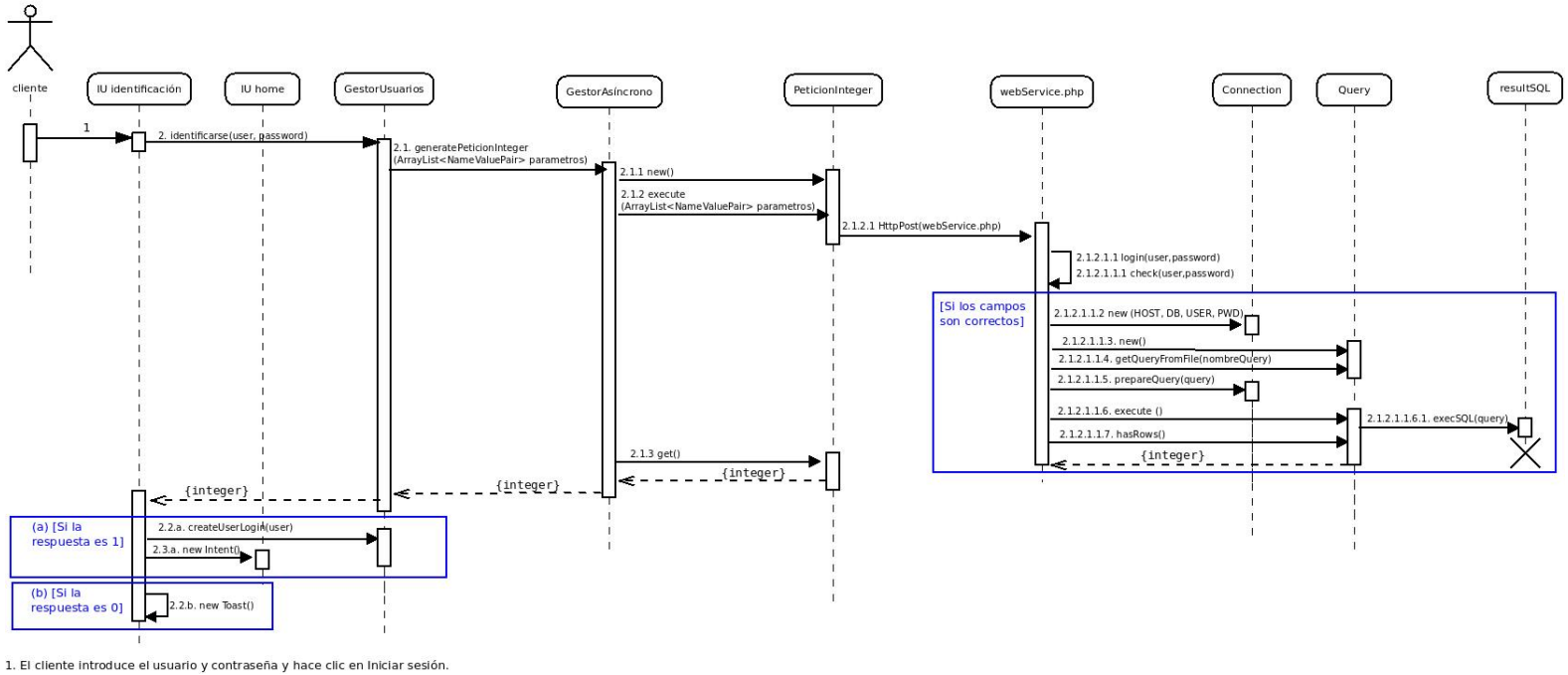


Figura 72: Diagrama de secuencia identificarse

La aplicación móvil está orientada únicamente a los clientes, por lo tanto, sólo hay una consulta:

- Query: *SELECT nombreCli FROM cliente WHERE emailCli = (?) AND passCli = (?)*

Desconectarse

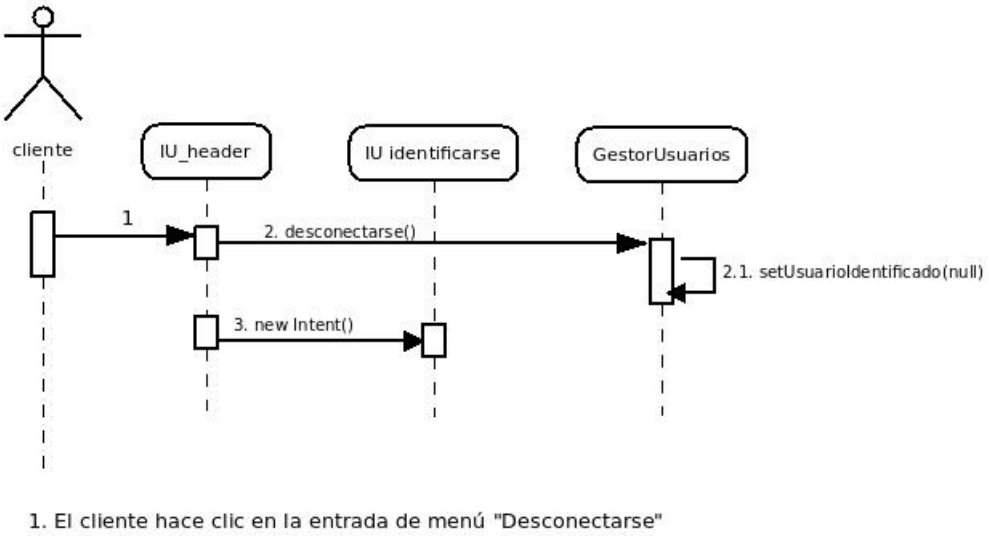


Figura 73: Diagrama de secuencia desconectarse

Anexo III - Paquetes de trabajo

TAREAS DESGLOSADAS

1. Requerimientos iniciales

- **1.1.** Captura de requisitos
- **1.2.** Análisis
- **1.3.** Prototipo 1: Aplicación básica
  - **1.3.1.** Identificación
    - **1.3.1.1.** Diseño
    - **1.3.1.2.** Implementación
      - **1.3.1.2.1.** Front-end
      - **1.3.1.2.2.** Back-end
    - **1.3.1.3.** Pruebas
  - **1.3.2.** Registro
    - **1.3.2.1.** Diseño
    - **1.3.2.2.** Implementación
      - **1.3.2.2.1.** Front-end
      - **1.3.2.2.2.** Back-end
    - **1.3.2.3.** Pruebas
  - **1.3.3.** Buscar restaurantes
    - **1.3.3.1.** Diseño
    - **1.3.3.2.** Implementación
      - **1.3.3.2.1.** Front-end
      - **1.3.3.2.2.** Back-end
    - **1.3.3.3.** Pruebas
- **1.4.** Prototipo 2: Gestión del restaurante y administración
  - **1.4.1.** Panel de administración
    - **1.4.1.1.** Dar de alta/baja restaurantes
      - **1.4.1.1.1.** Diseño

1.4.1.1.2. Implementación

1.4.1.1.2.1. Front-end

1.4.1.1.2.2. Back-end

1.4.1.1.3. Pruebas

1.4.2. Panel de administración

1.4.2.1. Modificar datos

1.4.2.1.1. Diseño

1.4.2.1.2. Implementación

1.4.2.1.2.1. Front-end

1.4.2.1.2.2. Back-end

1.4.2.1.3. Pruebas

1.4.2.2. Subir imagen restaurante

1.4.2.2.1. Diseño

1.4.2.2.2. Implementación

1.4.2.2.2.1. Front-end

1.4.2.2.2.2. Back-end

1.4.2.2.3. Pruebas

1.4.2.3. Modificar menú

1.4.2.3.1. Diseño

1.4.2.3.2. Implementación

1.4.2.3.2.1. Front-end

1.4.2.3.2.2. Back-end

1.4.2.3.3. Pruebas

1.5. Prototipo 3: Pagos

1.5.1. Registro ficticio

1.5.1.1. Diseño

1.5.1.2. Implementación

1.5.1.2.1. Front-end

1.5.1.2.2. Back-end

1.5.1.3. Pruebas

1.5.2. Realizar pedido

1.5.2.1. Diseño

1.5.2.2. Implementación

1.5.2.2.1. Front-end

1.5.2.2.2. Back-end

1.5.2.3. Pruebas

1.5.3. Dar de alta sucursales

1.5.3.1. Diseño

1.5.3.2. Implementación

1.5.3.2.1. Back-end

1.5.3.3. Pruebas

1.5.4. Realización del pago a través de WebPay

1.5.4.1. Diseño

1.5.4.2. Implementación

1.5.4.2.1. Front-end

1.5.4.2.2. Back-end

1.5.4.3. Pruebas

1.5.5. Realización del pago a través de Points2

1.5.5.1. Diseño

1.5.5.2. Implementación

1.5.5.2.1. Front-end

1.5.5.2.2. Back-end

1.5.5.3. Pruebas

1.5.6. Realización del pago a través de InvisibleCard

1.5.6.1. Diseño

1.5.6.2. Implementación

1.5.6.2.1. Front-end

1.5.6.2.2. Back-end

1.5.6.3. Pruebas

1.5.7. Panel de usuario

1.5.7.1. Gestión de pedidos

1.5.7.1.1. Diseño

1.5.7.1.2. Implementación

1.5.7.1.2.1. Front-end

1.5.7.1.2.2. Back-end

1.5.7.1.3. Pruebas

2. Requerimientos posteriores

2.1. Captura de requisitos

2.2. Análisis

2.3. Prototipo 4: Adaptación a dispositivos móviles

2.3.1. Adaptación del diseño

2.3.1.1. Implementación

2.3.1.1.1. Front-end

2.3.1.2. Pruebas

2.4. Prototipo 5: Extras

2.4.1. Calificar pedido

2.4.1.1. Diseño

2.4.1.2. Implementación

2.4.1.2.1. Front-end

2.4.1.2.2. Back-end

2.4.1.3. Pruebas

2.4.2. Compartir por redes sociales

2.4.2.1. Diseño

2.4.2.2. Implementación

2.4.2.2.1. Front-end

2.4.2.3. Pruebas

2.4.3. Extracción de reportes

2.4.3.1. Diseño

2.4.3.2. Implementación

2.4.3.2.1. Front-end

2.4.3.2.2. Back-end

2.4.3.3. Pruebas

3. Requerimientos ofrecidos

3.1. Captura de requisitos

3.2. Análisis

3.3. Aplicación Android

3.3.1. Identificarse

3.3.1.1. Diseño

3.3.1.2. Implementación

3.3.1.2.1. Front-end

3.3.1.2.2. Back-end

3.3.1.3. Pruebas

3.3.2. Búsqueda restaurantes

3.3.2.1. Diseño

3.3.2.2. Implementación

3.3.2.2.1. Front-end

3.3.2.2.2. Back-end

3.3.2.3. Pruebas

3.3.3. Gestión de pedidos

3.3.3.1. Diseño

3.3.3.2. Implementación

3.3.3.2.1. Front-end

3.3.3.2.2. Back-end

3.3.3.3. Pruebas

3.3.4. Realización pedido

3.3.4.1. Realización del pago a través de Points2

3.3.4.1.1. Diseño

3.3.4.1.2. Implementación

3.3.4.1.2.1. Front-end

3.3.4.1.2.2. Back-end

3.3.4.1.3. Pruebas

3.3.4.2. Valoración del pedido

3.3.4.2.1. Diseño

3.3.4.2.2. Implementación

3.3.4.2.2.1. Front-end

3.3.4.2.2.2. Back-end

3.3.4.2.3. Pruebas

Duración tareas comunes

Requisitos iniciales

Identificación (Paquete de trabajo 1.3.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	8 horas
Front-End	Tamara Pérez	10 horas
Back-End	Tamara Pérez; Ekaitz Portillo	6 horas
Pruebas	Ekaitz Portillo	2 horas

Registro (Paquete de trabajo 1.3.2)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	14 horas
Front-End	Tamara Pérez	8 horas
Back-End	Ekaitz Portillo	6 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	4 horas

Buscar restaurantes (Paquete de trabajo 1.3.3)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	8 horas
Front-End	Tamara Pérez	12 horas
Back-End	Ekaitz Portillo	9 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	2 horas

Dar alta/baja restaurantes (Paquete de trabajo 1.4.1.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	4 horas
Front-End	Tamara Pérez	3 horas
Back-End	Tamara Pérez	2 horas
Pruebas	Ekaitz Portillo	1 hora

Modificar datos (Paquete de trabajo 1.4.2.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	8 horas
Front-End	Tamara Pérez	12 horas
Back-End	Ekaitz Portillo	10 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	8 horas

Subir imagen restaurante (Paquete de trabajo 1.4.2.2)

Tarea	Responsable	Duración
Diseño	Ekaitz Portillo	3 horas
Front-End	Tamara Pérez	3 horas
Back-End	Ekaitz Portillo	6 horas
Pruebas	Ekaitz Portillo	2 horas

Modificar menú (Paquete de trabajo 1.4.2.3)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	8 horas
Front-End	Tamara Pérez	10 horas
Back-End	Ekaitz Portillo	8 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	3 horas

Registro ficticio (Paquete de trabajo 1.5.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	8 horas
Front-End	Tamara Pérez	10 horas
Back-End	Tamara Pérez; Ekaitz Portillo	15 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	8 horas

Realizar pedido (Paquete de trabajo 1.5.2)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	20 horas
Front-End	Tamara Pérez	7 horas
Back-End	Ekaitz Portillo	9 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	8 horas

Dar de alta sucursales (Paquete de trabajo 1.5.3)

Tarea	Responsable	Duración
Diseño	Ekaitz Portillo	3 horas
Front-End	-	-
Back-End	Ekaitz Portillo	14 horas
Pruebas	Ekaitz Portillo	10 horas

Realización del pago a través de Webpay (Paquete de trabajo 1.5.4)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	13 horas
Front-End	Tamara Pérez	4 horas
Back-End	Tamara Pérez; Ekaitz Portillo	14 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	10 horas

Realización del pago a través de Points2 (Paquete de trabajo 1.5.5)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	12 horas
Front-End	Tamara Pérez	4 horas
Back-End	Tamara Pérez; Ekaitz Portillo	14 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	6 horas

Realización del pago a través de InvisibleCard (Paquete de trabajo 1.5.6)

Tarea	Responsable	Duración
Diseño	Tamara Pérez; Ekaitz Portillo	3 horas
Front-End	Tamara Pérez	1 hora
Back-End	Ekaitz Portillo	15 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	4 horas

Gestión de pedidos (Paquete de trabajo 1.5.7.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	8 horas
Front-End	Tamara Pérez	12 horas
Back-End	Tamara Pérez; Ekaitz Portillo	8 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	3 horas

Requerimientos posteriores

Adaptación del diseño (Paquete de trabajo 2.3.1)

Tarea	Responsable	Duración
Diseño	-	-
Front-End	Tamara Pérez	42 horas
Back-End	-	-
Pruebas	Tamara Pérez; Ekaitz Portillo	36 horas

Calificar pedido (Paquete de trabajo 2.4.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	6 horas
Front-End	Tamara Pérez	8 horas
Back-End	Ekaitz Portillo	4 horas
Pruebas	Tamara Pérez; Ekaitz Portillo	4 horas

Compartir por redes sociales (Paquete de trabajo 2.4.2)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	1 hora
Front-End	Tamara Pérez	2 horas
Back-End	-	-
Pruebas	Tamara Pérez	1 hora

Extracción de reportes (Paquete de trabajo 2.4.3)

Tarea	Responsable	Duración
Diseño	Ekaitz Portillo	4 horas
Front-End	Tamara Pérez	1 hora
Back-End	Ekaitz Portillo	12 horas
Pruebas	Ekaitz Portillo	3 horas

Requerimientos ofrecidos

Identificarse (Paquete de trabajo 3.3.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	5 horas
Front-End	Tamara Pérez	5 horas
Back-End	Tamara Pérez	3 horas
Pruebas	Tamara Pérez	2 horas

Búsqueda restaurantes (Paquete de trabajo 3.3.2)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	5 horas
Front-End	Tamara Pérez	16 horas
Back-End	Tamara Pérez	10 horas
Pruebas	Tamara Pérez	5 horas

Gestión de pedidos (Paquete de trabajo 3.3.3)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	10 horas
Front-End	Tamara Pérez	20 horas
Back-End	Tamara Pérez	32 horas
Pruebas	Tamara Pérez	10 horas

Realización del pago a través de Points2 (Paquete de trabajo 3.3.4.1)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	5 horas
Front-End	Tamara Pérez	4 horas
Back-End	Tamara Pérez	12 horas
Pruebas	Tamara Pérez	10 horas

Valoración del pedido (Paquete de trabajo 3.3.4.2)

Tarea	Responsable	Duración
Diseño	Tamara Pérez	2 horas
Front-End	Tamara Pérez	4 horas
Back-End	Tamara Pérez	3 horas
Pruebas	Tamara Pérez	1 hora

Anexo IV - Actas reuniones

A continuación, se mostrarán algunas actas de reunión en las cuales se han omitido los nombres de los trabajadores de MIT por temas de confidencialidad. El motivo de no adjuntar todas las actas de reunión se debe a que la cantidad de reuniones que se han tenido ha sido muy elevada.

Además, a día de hoy aún se siguen realizando reuniones.

Acta reunión 1

Proyecto	Easy Order
Asunto	Primera toma de contacto
Lugar	Skype
Fecha	19/07/2013
Hora comienzo	21:00
Hora fin	22:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Director General MIT
- ≻ Ausencias
- ≻ Orden del día
  - Realizar una primera toma de contacto.
  - Explicación de la gestión de dudas.
  - Explicación de la jerarquía y organización de la empresa.
- ≻ Acuerdos alcanzados
  - Se indica como será la gestión de dudas.
  - Se indica como está organizada la empresa.
  - Se facilitan los datos de contacto de los responsables.
- ≻ Próxima reunión:
  - 22/07/2013 a las 21:00 en Skype.

Acta reunión 2

Proyecto	Easy Order
Asunto	Captura de requisitos
Lugar	Skype
Fecha	22/07/2013
Hora comienzo	21:00
Hora fin	24:00

- Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable proyecto Easy Order
- Ausencias
- Orden del día
  - Realizar una primera toma de contacto.
  - Explicación de la aplicación que quiere MIT.
- Acuerdos alcanzados
  - Se realiza la captura de requisitos.
- Próxima reunión:
  - 05/08/2013 a las 21:00 en Skype.

Acta reunión 3

Proyecto	Easy Order
Asunto	Revisión diseño aplicación
Lugar	Skype
Fecha	05/08/2013
Hora comienzo	21:00
Hora fin	23:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable proyecto Easy Order
- ≻ Ausencias
- ≻ Orden del día
  - Revisión del diseño de la aplicación.
- ≻ Acuerdos alcanzados
  - Modificación de los colores del diseño de la aplicación.
  - Enviarán el logo de Easy Order en el plazo de una semana para poder adaptar el diseño.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se envíe el logo.

Acta reunión 4

Proyecto	Easy Order
Asunto	Revisión diseño aplicación
Lugar	Skype
Fecha	16/09/2013
Hora comienzo	21:00
Hora fin	23:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable proyecto Easy Order
- ≻ Ausencias
- ≻ Orden del día
  - Revisión del diseño de la aplicación.
- ≻ Acuerdos alcanzados
  - Se acepta el diseño actual, pendiente de revisarlo en un futuro.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se finalicen los requerimientos iniciales.

Acta reunión 5

Proyecto	Easy Order
Asunto	Gestión de dudas métodos de pago
Lugar	Skype
Fecha	01/10/2013
Hora comienzo	21:00
Hora fin	22:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable proyecto Easy Order
- ≻ Ausencias
- ≻ Orden del día
  - Solicitar información sobre los métodos de pago.
- ≻ Acuerdos alcanzados
  - El responsable del proyecto contactará con los responsables de los métodos de pago y enviarán la documentación por e-mail.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se proporcione la información de los pagos.

Acta reunión 6

Proyecto	Easy Order
Asunto	Gestión de dudas de Points2
Lugar	Skype
Fecha	11/11/2013
Hora comienzo	21:00
Hora fin	23:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable del método de pago Points2
- ≻ Ausencias
- ≻ Orden del día
  - Resolver las dudas de la documentación proporcionada.
- ≻ Acuerdos alcanzados
  - Se han resuelto las dudas.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se finalicen los métodos de pago para realizar las pruebas oportunas.

Acta reunión 7

Proyecto	Easy Order
Asunto	Gestión de dudas de InvisibleCard
Lugar	Skype
Fecha	18/11/2013
Hora comienzo	21:00
Hora fin	23:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable del método de pago InvisibleCard
- ≻ Ausencias
- ≻ Orden del día
  - Resolver las dudas de la documentación proporcionada.
- ≻ Acuerdos alcanzados
  - Se han resuelto las dudas.
  - Se enviará por e-mail más información.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se finalicen los métodos de pago para realizar las pruebas oportunas.

Acta reunión 8

Proyecto	Easy Order
Asunto	Gestión de dudas de WebPay
Lugar	Skype
Fecha	20/11/2013
Hora comienzo	21:00
Hora fin	23:00

- ≻ Asistentes:
  - Ekaitz Portillo
  - Tamara Pérez
  - Responsable del método de pago WebPay
- ≻ Ausencias
- ≻ Orden del día
  - Resolver las dudas de la documentación proporcionada.
- ≻ Acuerdos alcanzados
  - Se han resuelto las dudas.
  - Se enviará por e-mail más información.
- ≻ Próxima reunión:
  - Se concertará por e-mail cuando se finalicen los métodos de pago para realizar las pruebas oportunas.

Anexo V - Manual configuración de acceso a base de datos

A continuación se detallará cómo configurar el sistema Easy Order en el ambiente productivo a partir de un fichero sql proporcionado.

Requerimientos

Los requerimientos necesarios para hacer funcionar Easy Order son:

PHP

- Es necesario tener la versión 5 de PHP o superior. Para comprobar la versión que está instalada en el servidor, es suficiente con:
  - Importar a la carpeta pública el fichero phpinfo.php (es suficiente con hacer copy-paste en el directorio que corresponda del servidor).
    - Acceder al navegador y abrir el fichero.
    - Ahí se cargará una página, en la cual se podrá ver en la parte superior la versión de PHP instalada.
    - Por ejemplo, en nuestro caso, el fichero se encuentra en el siguiente directorio: *http://localhost:8080/eo/phpinfo.php*

MySQL

- Es estrictamente necesario tener la versión 5 ó superior de MySQL instalada para poder tener una instancia de Bases de Datos y así, poder acceder a los datos. Para visualizar la versión que tenemos instalada es suficiente con seguir los siguientes pasos:
  - Acceder al directorio del servidor en el cual está instalado MySQL.
    - Si se utiliza Windows, bastará con:

1. Acceder a Símbolo del sistema.
2. Escribir Dir (ruta de MySQL).
3. Escribir mysql –u nombre_usuario –p
4. Introducir clave.
5. Escribir STATUS.

- Si se utiliza Unix, bastará con:
  1. Acceder al terminal.
  2. Escribir cd (ruta de MySQL).
  3. Escribir mysql –u nombre_usuario –p
  4. Introducir clave.
  5. Escribir STATUS.

PhpMyAdmin

- Si se dispone de PhpMyAdmin se podrá efectuar el proceso de importación/creación de Bases de Datos y usuarios de manera más sencilla. No es indispensable tener PhpMyAdmin instalado en el servidor para que el sistema EasyOrder funcione.

Configuración

A continuación se describirá el proceso de configuración del servidor para poder albergar los datos de la Base de Datos del sistema Easy Order.

Mediante PhpMyAdmin

Los pasos a seguir son los siguientes:

1. Acceder a la ruta en la que se encuentre PhpMyAdmin instalado en nuestro servidor, en nuestro caso: *http://localhost:8080/phpMyAdmin/*

Figura 74: Inicio sesión PhpMyAdmin

2. Acceder con el usuario root, introduciendo la clave que tengamos fijada.
3. Hacer clic en "Bases de Datos" (Paso 1 de la figura 75).
4. Escribir el nombre de la base de datos, que tiene que ser: easyorder (Paso 2 de la figura 75).
5. Hacer clic en crear (Paso 3 de la figura 75).

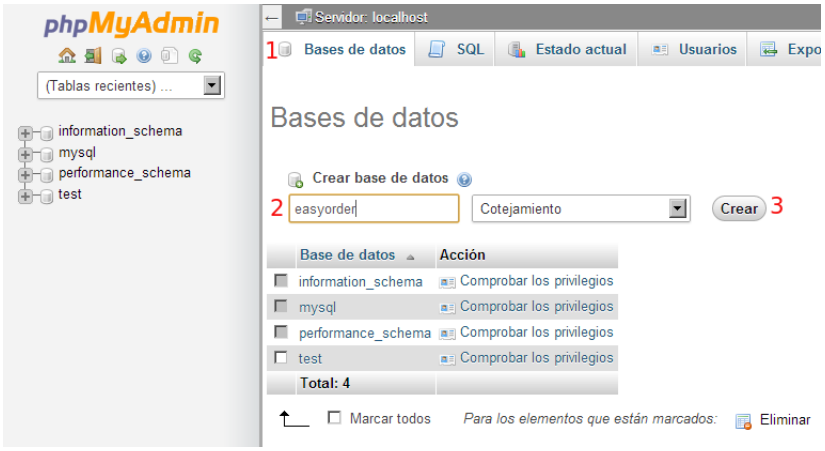


Figura 75: Pasos creación base de datos PhpMyAdmin

6. Hacer clic en la Base de Datos creada (Paso 4 de la figura 76).
7. Hacer clic en "Importar" (Paso 5 de la figura 76).

8. Hacer clic en “Seleccionar archivo” y seleccionamos el archivo “easyorder.sql” (Paso 6 de la figura 76).

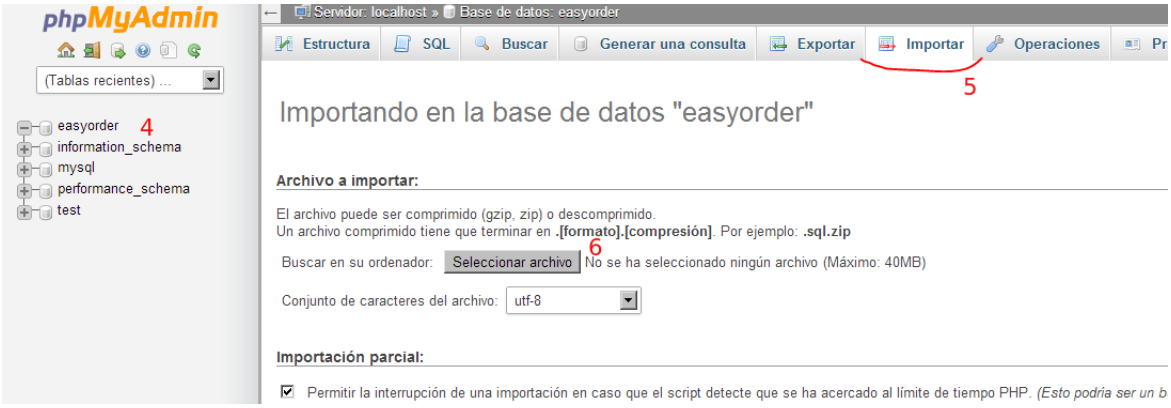


Figura 76: Pasos creación base de datos PhpMyAdmin

9. Hacer clic en “Continuar”.

10. Hacer clic en “Privilegios” (Paso 7 de la figura 77).

11. Hacer clic en “Agregar usuario” (Paso 8 de la figura 77).

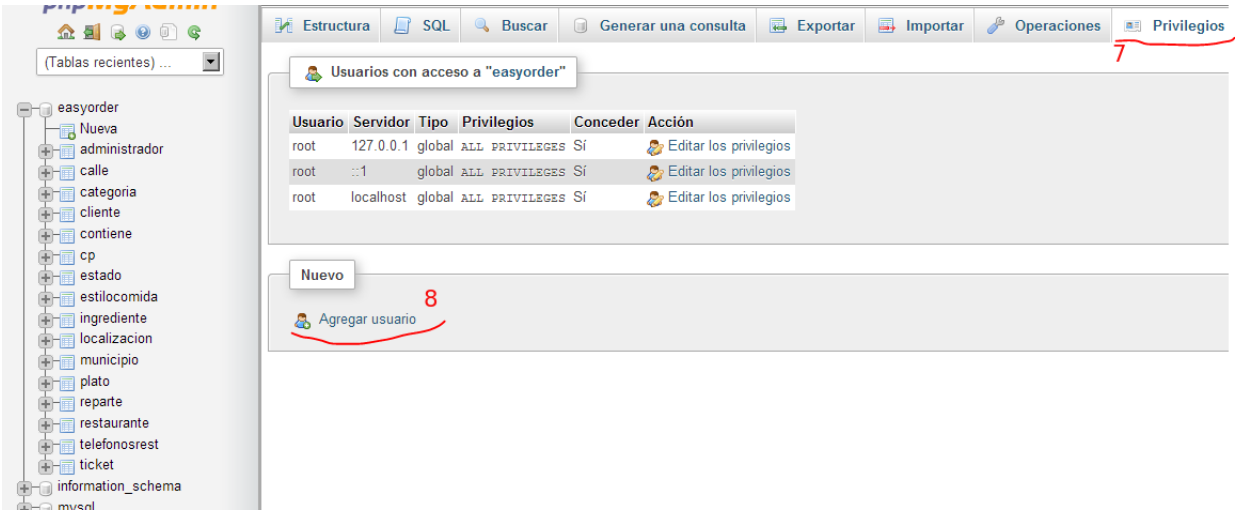


Figura 77: Pasos creación base de datos PhpMyAdmin

12. Introducir los datos marcados:

    a) Nombre de usuario: easyorderuser.
    b) Clave: *******.

c) Marcar todos los privilegios globales.