Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA (SG)
Grado en Ingeniería Informática de Servicios y Aplicaciones

PadelBooking: Aplicación Web para la gestión de un club de pádel

---

Alumno: Jorge Calvo Blanco

Tutores: Fernando Díaz Gómez

José Ignacio Farrán Martín

“Per aspera ad astra”

Agradecimientos

Gracias a toda mi familia por la infinita paciencia y apoyo que han demostrado a lo largo de mis años de estudio.

Gracias a mis amigos, que han estado ahí cuando les he necesitado, demostrando un apoyo inquebrantable.

Gracias a mis profesores por la formación recibida. Mención especial a mis tutores, Fernando Díaz Gómez y José Ignacio Farrán Martín, por guiarme a lo largo de este proyecto.

Muchas gracias.

RESUMEN

El objetivo del presente Trabajo Final de Grado consiste en el desarrollo de un sistema integral para la gestión de un club de pádel. Los principales puntos a desarrollar son un sistema de reserva de pistas y una tienda online de productos para la práctica del pádel que sean rápidos y efectivos.

De esta manera se intentará mejorar la experiencia de usuario de un cliente de un club de pádel.

Palabras clave:

Desarrollo web, Python, Django, JavaScript, Angular, pádel, reserva de pistas deportivas

ABSTRACT

The main objective of this Degree Final Project is to develop a complete system capable of managing a paddle club. The main functionalities to develop are a booking system for paddle courts, and an online paddle store. Both systems must be quick and effective.

In this way, the user experience in a paddle club will be improved.

Keywords:

Web development, Python, Django, JavaScript. Angular, paddle, court booking system

Índice de Contenido

1. INTRODUCCIÓN 13
1.1. Motivación 14
1.2. Objetivos y Alcance 14
1.3. Entorno tecnológico 16
1.4. Estructura de la documentación 18

2. ESTADO DEL ARTE 19

3. ANÁLISIS 22
3.1. Actores del sistema 23
3.2. Reglas de negocio 25
3.3. Requisitos de Usuario 25
3.3.1. Casos de Uso 26
3.3.2. Especificación de Casos de Uso 30
3.4. Requisitos de Información 41
3.4.1. Diagrama Entidad-Relación 41
3.4.2. Entidades y relaciones 42
3.5. Requisitos No Funcionales 48
3.6. Requisitos Funcionales 49

4. PLANIFICACIÓN Y PRESUPUESTO 55
4.1. Metodología 56
4.2. Planificación Temporal 57
4.2.1. Estimación del Esfuerzo 57
4.2.2. Presupuesto 63
4.3. Coste real 65

5. DISEÑO 69
5.1. Arquitectura Lógica 70
5.2. Arquitectura Física 72
5.3. Diseño de la Base de Datos 72
5.3.1. Modelo relacional 72

5.3.2. Diccionario de datos ... 73
5.4. Diagramas de secuencia ... 76
5.5. Diagramas de interfaz de usuario ... 79

6. IMPLEMENTACIÓN ... 82
6.1. Herramientas y tecnologías utilizadas ... 83
6.1.1. Herramientas de desarrollo ... 83
6.1.2. Herramientas de soporte ... 84
6.1.3. Tecnologías Utilizadas ... 84
6.2. Django ... 85
6.3. Implementación de la base de datos ... 90
6.3.1. Consideraciones de implementación ... 90
6.3.2. Rendimiento de la base de datos ... 91
6.4. Estructura del proyecto ... 93
6.4.1. API ... 93
6.4.2. Front-end ... 101
6.5. PayPal ... 106
6.6. Implementación ... 106
6.6.1. Proyecto Django ... 107
6.6.2. Proyecto Angular ... 109
6.7. Heroku ... 112

7. PRUEBAS ... 114
7.1. Pruebas de Caja Negra ... 115

8. MANUALES ... 126
8.1. Manual de despliegue ... 127
8.2. Manual de Usuario ... 134
8.2.1. Perfil: Usuario No Autenticado ... 134
8.2.2. Perfil: Usuario Cliente ... 140
8.2.3. Perfil: Usuario Administrador ... 152

9. CONCLUSIONES Y FUTURAS MEJORAS ... 162
9.1. Conclusiones ... 163

9.2. Futuras mejoras 163

REFERENCIAS 164

Bibliografía 165

Webgrafía 166

Índice de Figura

Figura 1: Árbol de Características 15
Figura 2: Entorno Tecnológico 17
Figura 3: Club Pádel Peñalara 20
Figura 4: Club PadelZone 21
Figura 5: Club PadelPoint 21
Figura 6: Jerarquía de Actores 24
Figura 7: Diagrama Casos de Uso Usuario Anónimo 26
Figura 8: Diagrama Casos de Uso Usuario Registrado 27
Figura 9: Diagrama Casos de Uso Cliente 28
Figura 10: Diagrama Casos de Uso Administrador 29
Figura 11: Diagrama Entidad-Relación 41
Figura 12: Metodología Desarrollo en Cascada 56
Figura 13: Diagrama de Gantt inicial 62
Figura 14: Diagrama de Gantt Real 66
Figura 15: Arquitectura Lógica 71
Figura 16: Arquitectura física 72
Figura 17: Modelo Relacional 73
Figura 18: Diagrama de Secuencia CU-04 77
Figura 19: Diagrama de Secuencia CU-11 78
Figura 20: Diagrama de Secuencia CU-18 78
Figura 21: Diagrama de Interfaz de Usuario 01 79
Figura 22: Diagrama de Interfaz de Usuario 02 80
Figura 23: Diagrama de Interfaz de Usuario 03 81
Figura 24: Patrón Modelo-Vista-Template 86
Figura 25: Login Panel de Administración 87
Figura 26: Página principal Panel de Administración 88
Figura 27: Ejemplo modelo Panel de Administración 89
Figura 28: Ejemplo registro Panel de Administración 90
Figura 29: Iniciar pgbench 91
Figura 30: Benchmark base de datos 92
Figura 31: Estructura API 93
Figura 32: Carpeta de imágenes de la aplicación 94
Figura 33: Ejemplo app Django 94
Figura 34: Migraciones app 95
Figura 35: Ejemplo admin.py 95
Figura 36: Ejemplo apps.py 96
Figura 37: Ejemplo models.py 96
Figura 38: Ejemplo serializers.py 97
Figura 39: Ejemplo urls.py 97
Figura 40: Ejemplo views.py 98
Figura 41: App principal 98
Figura 42: Ejemplo asgi.py 99
Figura 43: Ejemplo settings.py 99
Figura 44: Ejemplo urls.py principal 100

Figura 45: Ejemplo wsgi.py 100
Figura 46: Entorno Virtual 100
Figura 47: Estructura de directorios Front-End 101
Figura 48: Ejemplo módulo Angular 102
Figura 49: Ejemplo componente Angular 102
Figura 50: Ejemplo servicio 103
Figura 51: Ejemplo componente typescript 103
Figura 52: Ejemplo componente html 104
Figura 53: Ejemplo componente css 105
Figura 54: Directorio modelos Angular 105
Figura 55: Creación entorno virtual 107
Figura 56: Instalación Django 107
Figura 57: Creación proyecto Django 108
Figura 58: Proyecto Django creado 108
Figura 59: Proyecto Django corriendo 108
Figura 60: Proyecto Django página principal 109
Figura 61: VSCode Terminal 110
Figura 62: Instalación Angular CLI 110
Figura 63: Creación app Angular 110
Figura 64: Routing app Angular 111
Figura 65: Hojas de estilos app Angular 111
Figura 66: Directorio de la app 111
Figura 67: Ejecutar app 112
Figura 68: App Angular página principal 112
Figura 69: Directorio instalación PostgreSQL 127
Figura 70: Componentes PostgreSQL 128
Figura 71: Fin instalación PostgreSQL 128
Figura 72: Directorio instalación PyCharm 129
Figura 73: PyCharm opciones instalación 130
Figura 74: Creación base de datos 130
Figura 75: Creación base de datos 2 131
Figura 76: pgAdmin Query Tool 131
Figura 77: Archivo backup 132
Figura 78: Settings Base de datos 132
Figura 79: Instalación librerías 133
Figura 80: Levantar proyecto 133
Figura 81: Pantalla principal de la aplicación 134
Figura 82: Pantalla de login de la aplicación 135
Figura 83: Login exitoso 135
Figura 84: Login erróneo 136
Figura 85: Formulario de registro de usuario 136
Figura 86: Formulario de registro incorrecto 137
Figura 87: Datos de registro correctos 137
Figura 88: Pantalla principal de la tienda 138
Figura 89: Pantalla de detalles de un producto 139
Figura 90: Necesaria una cuenta de cliente para añadir un producto 139
Figura 91: Pantalla principal de la aplicación 2 140

Figura 92: Sesión cerrada 141
Figura 93: Añadir producto a carrito 142
Figura 94: Carrito de productos 143
Figura 95: Eliminar producto de carrito 143
Figura 96: Ventana de pago PayPal 144
Figura 97: Pedido realizado con éxito 145
Figura 98: Seleccionar fecha y pista 145
Figura 99: Seleccionar sesión para la reserva 146
Figura 100: Reserva realizada 146
Figura 101: Perfil de usuario 147
Figura 102: Modificar perfil de usuario 147
Figura 103: Reservas activas 148
Figura 104: Cancelar reserva 148
Figura 105: Reserva cancelada 149
Figura 106: Pedidos realizados 149
Figura 107: Cancelar pedido 150
Figura 108: Pedido cancelado 150
Figura 109: Detalles de pedido 151
Figura 110: Chat 151
Figura 111: Enviar mensaje 152
Figura 112: Pantalla principal de Administrador 153
Figura 113: Modelos de la aplicación 154
Figura 114: Listado de usuarios 154
Figura 115: Detalles de usuario 155
Figura 116: Eliminar usuario 155
Figura 117: Listado de pistas 156
Figura 118: Detalles de pista 156
Figura 119: Eliminar pista 157
Figura 120: Dar de alta una pista 157
Figura 121: Listado de pedidos 158
Figura 122: Detalles de pedido 158
Figura 123: Estado de pedido modificado 159
Figura 124: Listado de productos 159
Figura 125: Detalles de producto 160
Figura 126: Eliminar producto 160
Figura 127: Añadir nuevo producto 161

1. INTRODUCCIÓN

1.1. Motivación

La idea de desarrollar esta aplicación surge a raíz de la falta de clubes de pádel que tengan todas sus herramientas de gestión informatizadas. Lo usual es que en los clubes únicamente se pueda reservar pistas a través de llamada telefónica o acudiendo en persona al club para realizar la reserva. Además, muchos de los clubes que tienen tienda de material para la práctica del pádel únicamente realizan ventas físicamente.

Ambas características presentan inconvenientes, ya que por un lado el sistema de reservas descrito conlleva una serie de molestias evitables para el cliente, y por otro, el club estaría perdiendo posibles ventas de material al únicamente realizar ventas físicamente.

Por ello, para desarrollar este proyecto se han tenido en cuenta ambas características, para así proporcionar una aplicación que resuelva estas deficiencias.

Además, como se podrá observar en la sección 2, Estado del Arte, de este documento, en el área de Segovia no existe ninguna herramienta que cubra las funcionalidades que proponemos en nuestra herramienta.

1.2. Objetivos y Alcance

Los principales objetivos que queremos conseguir con el desarrollo de esta aplicación son los siguientes:

ID	Objetivo
OB-01	Implementar un sistema eficiente de reservas de pistas de pádel.
OB-02	Implementar una tienda online dedicada a la venta de equipamiento para la práctica del pádel.
OB-03	Implementar un sistema de chat grupal que permita la comunicación entre usuarios para agilizar la planificación de partidos.

Para ello, la aplicación contará con cuatro módulos diferenciados, como se detalla en el siguiente árbol de características:

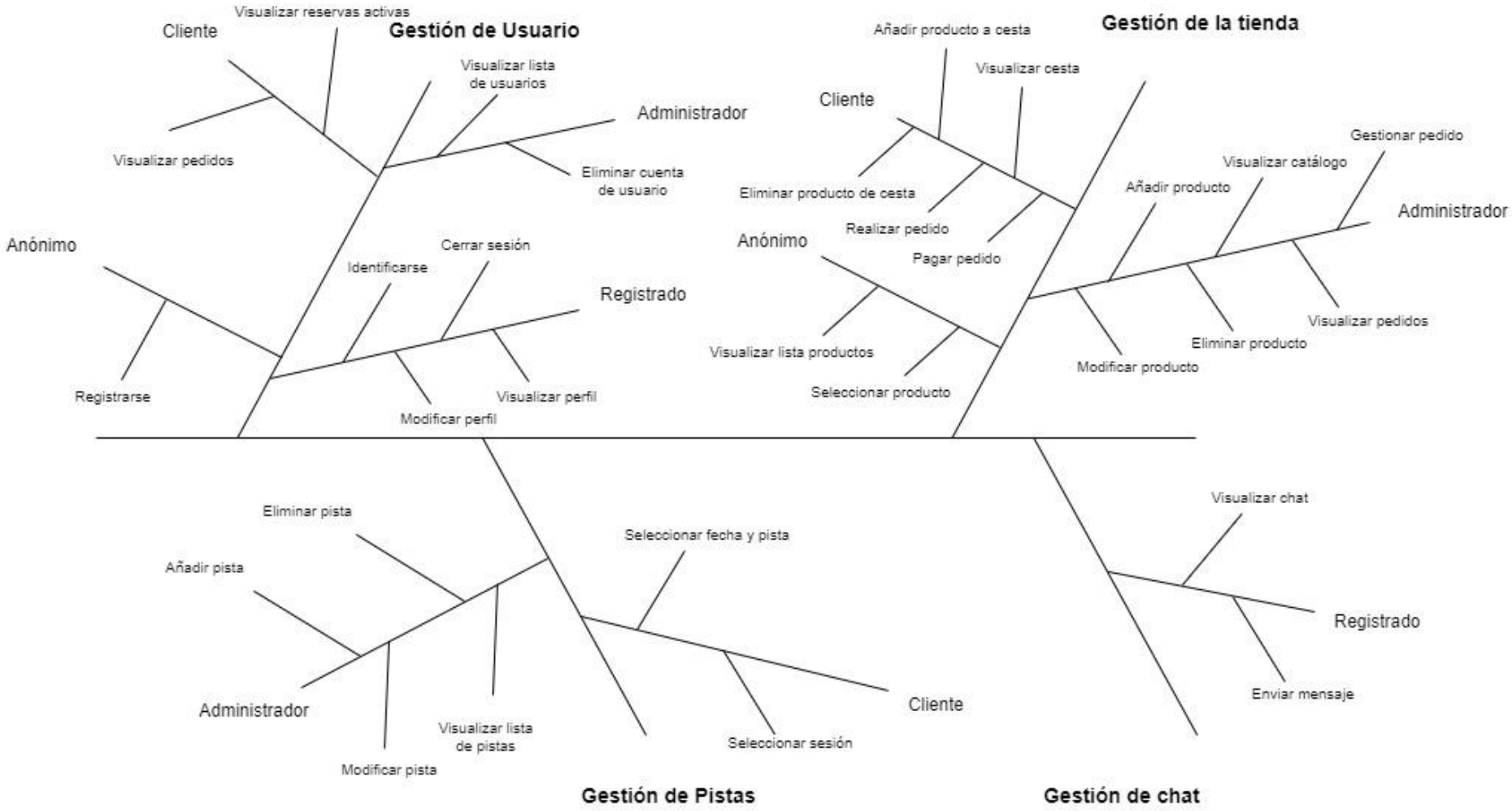

FIGURA 1: ÁRBOL DE CARACTERÍSTICAS

Como muestra el árbol de características, las principales características del sistema son:

- **Gestión de usuarios**: El sistema permitirá a los usuarios identificarse y, en caso de que no estén registrados en el sistema, les permitirá registrarse. Además, permitirá a los usuarios realizar todo tipo de gestiones en su perfil de usuario.
- **Gestión de la tienda**: El sistema permitirá a los usuarios registrados visualizar cualquier producto de la tienda, pero para realizar un pedido deberán estar autenticados en el sistema como clientes. Por otra parte, el administrador podrá realizar gestiones derivadas de la tienda y sus productos.
- **Gestión de pistas**: El sistema permitirá a los clientes realizar reservas de pistas, y a los administrador realizar las gestiones derivadas de las pistas.
- **Gestión de chat**: El sistema permitirá a los usuarios registrados acceder a un chat grupal con el resto de usuarios.

1.3. Entorno tecnológico

Esta aplicación web va a consistir en una API REST realizada con Django, framework para desarrollo web del lenguaje Python. Dentro de Django emplearemos un framework especializado en el desarrollo de API REST llamado Django REST Framework.

Esta API REST será consumida por un front-end desarrollado en Angular 9, framework de JavaScript, el cual incluye otras tecnologías como HTML5, Ajax, CSS3 o Webpack.

Para la Base de Datos usaremos el sistema gestor de bases de datos PostgreSQL.

En el siguiente diagrama se explica de forma más clara la situación de cada tecnología:

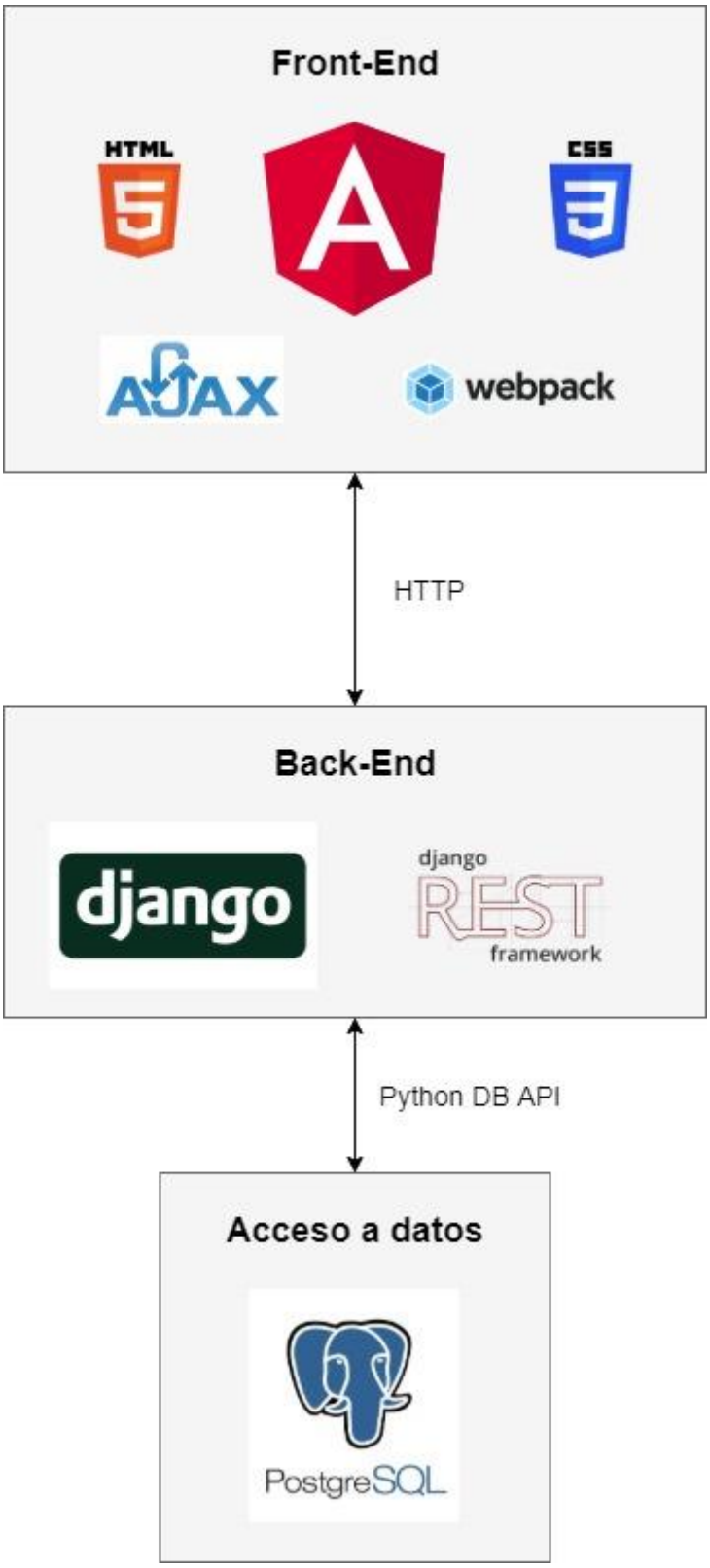

FIGURA 2: ENTORNO TECNOLÓGICO

1.4. Estructura de la documentación

- **Capítulo 1, Introducción**: Capítulo que contiene una breve introducción a la motivación del proyecto, sus objetivos y el entorno en el que se desarrolla.

- **Capítulo 2, Estado del Arte**: Capítulo en el que se analizan brevemente algunas aplicaciones similares a nuestro proyecto.

- **Capítulo 3, Análisis**: Capítulo que contiene el proceso de análisis de la aplicación. Se obtienen y detallan los requisitos necesarios para el correcto funcionamiento de la aplicación.

- **Capítulo 4, Planificación y presupuesto**: Capítulo en el que se detalla una estimación del coste en tiempo y recursos del proyecto y, posteriormente, se realiza un análisis del coste real del proyecto.

- **Capítulo 5, Diseño**: Capítulo en el que se detallan las distintas arquitecturas del proyecto, complementadas con sus respectivos diagramas explicativos. Además se especificará el diseño de la base de datos.

- **Capítulo 6, Implementación**: Capítulo en el que se describen detalles sobre la implementación y desarrollo del proyecto relacionados principalmente con las tecnologías empleadas para el desarrollo.

- **Capítulo 7, Pruebas**: Capítulo que contiene las distintas pruebas realizadas para asegurar que la aplicación cumple todos los requisitos y funciona correctamente.

- **Capítulo 8, Manuales**: Capítulo que contiene las explicaciones necesarias para desplegar la aplicación y usarla.

- **Capítulo 9, Conclusiones y futuras mejoras**: Capítulo que contiene unas breves conclusiones sobre el desarrollo del proyecto. Además se incluyen posibles mejoras para la aplicación.

- **Referencias**: Capítulo en el que se listan la bibliografía y webgrafía empleada durante el desarrollo del proyecto.

2. ESTADO DEL ARTE

Actualmente existe una amplia variedad de aplicaciones web para la gestión de un club de pádel, aunque en el área de Segovia no hemos encontrado ninguna aplicación que cubra todas las funcionalidades que planteamos nosotros con nuestra herramienta. Por lo tanto nos encontramos ante una buena oportunidad de negocio.

A continuación vamos a analizar las herramientas encontradas en el área de Segovia que más se asemejen en cuanto a funcionalidades.

Pádel Peñalara

Esta aplicación, en uso por el Club de Pádel Peñalara, comparte similitudes con nuestra aplicación, ya que permite que el usuario se registre en el sistema y realice reservas de pistas. Sin embargo, no cuenta con tienda online, ni con un chat entre usuarios.

Por otro lado, la aplicación ofrece información sobre las clases y torneos que ofertan, pero como no permite la matriculación online en ninguno de los dos aspectos, no los vamos a tener en cuenta.

FIGURA 3: CLUB PÁDEL PEÑALARA

Club PadelZone

Esta aplicación, utilizada por el club PadelZone, cuenta con las mismas funcionalidades que la anterior, es decir, registro de usuarios y reserva de pistas. No cuenta con tienda online, ni chat entre usuarios.

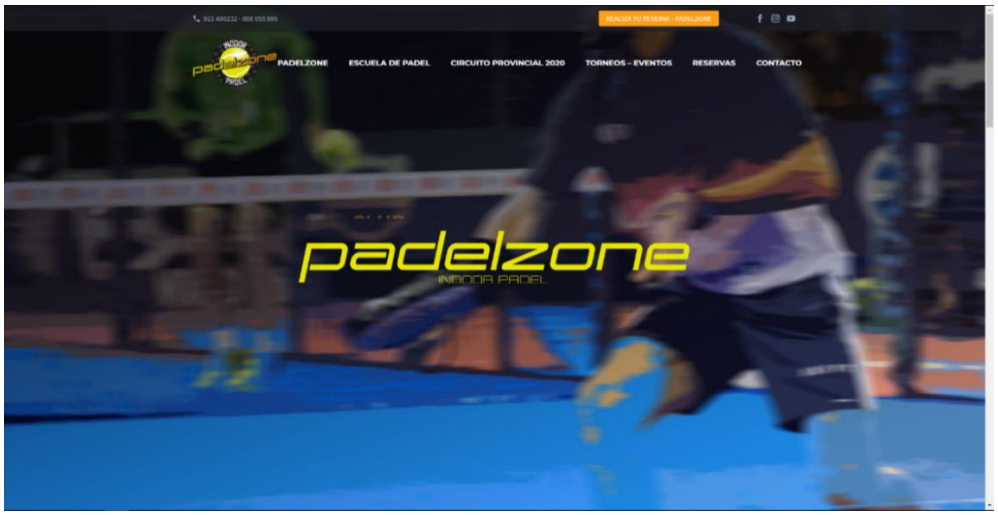

FIGURA 4: CLUB PADELZONE

Club PADELPOINT

Por último, vamos a analizar las funcionalidades de esta aplicación que, pese a no estar operativa en el área de Segovia, cubre todas las funcionalidades implementadas en nuestra aplicación.

Cuenta con registro de usuarios, reserva de pistas, tienda online y, aunque no tiene un chat entre usuarios propiamente dicho, cuenta con un sistema para que los usuarios planifiquen partidos entre ellos.

FIGURA 5: CLUB PADELPOINT

3. ANÁLISIS

En esta sección del documento especificaremos los distintos tipos de actores, requisitos o reglas obtenidos del análisis para desarrollar la aplicación.

3.1. Actores del sistema

Empezamos el análisis especificando los distintos actores que van a interactuar con el sistema, así como su jerarquía.

ID	Nombre	Descripción
AC-01	Usuario anónimo	Este actor representa al potencial usuario del sistema, al cual accede sin estar registrado. Dentro del sistema tiene la posibilidad de registrarse y de acceder a la tienda.
AC-02	Usuario registrado	Este actor representa al usuario que ya está registrado en la aplicación. Las funciones atribuidas a este actor son las comunes a los actores que le suceden en la jerarquía de actores que se muestra a continuación, como identificarse en el sistema o cerrar la sesión.
AC-03	Administrador	Este actor representa al superusuario de la aplicación, el cual tiene privilegios especiales. Se encarga de la gestión de los productos de la tienda y de la gestión de las pistas.
AC-04	Cliente	Este actor representa al usuario estándar que realiza compras en la tienda o realiza reservas de pistas.
AC-05	Plataforma de pago	Este actor representa al sistema de pago externo que se empleará para que el Cliente pueda realizar los pagos en la aplicación.

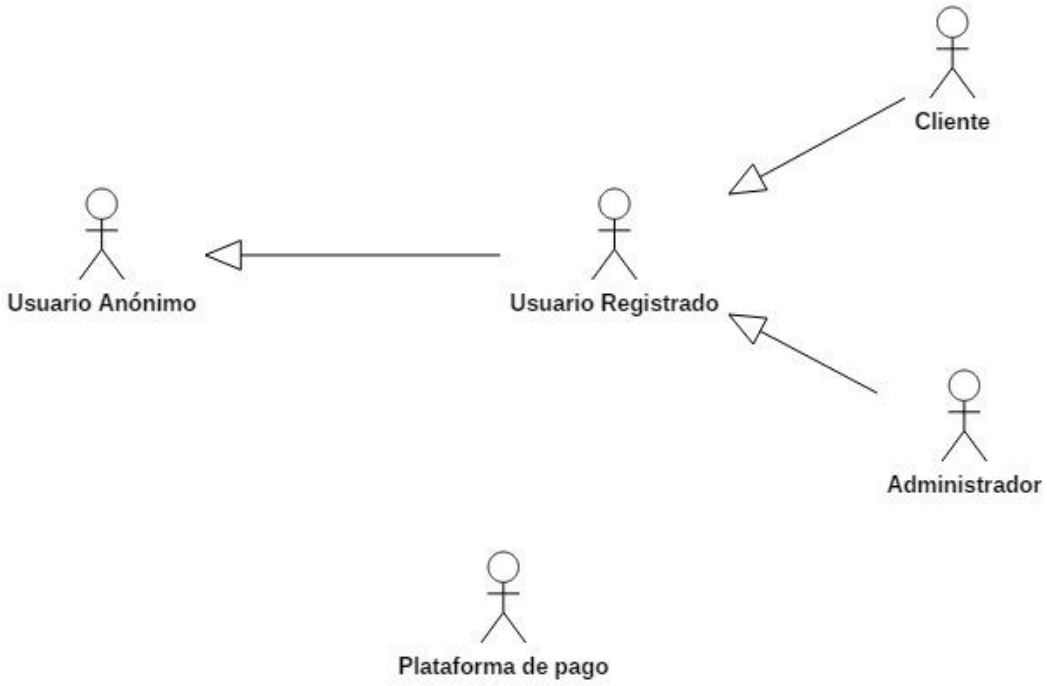

FIGURA 6: JERARQUÍA DE ACTORES

3.2. Reglas de negocio

Las reglas de negocio son un conjunto de políticas de empresa, normas o restricciones impuestas por el contexto o la empresa para la que se va a desarrollar la aplicación. Las reglas de negocio que debemos tener en cuenta para desarrollar nuestro sistema son las siguientes:

ID	Descripción
RN-01	Un usuario anónimo únicamente puede acceder a las funcionalidades RU-01, RU-02 y RU-03.
RN-02	Un pedido sólo podrá pasar a estado "Confirmado" cuando un administrador valide el pago de este.
RN-03	Un administrador no podrá acceder a ciertas funcionalidades, propias del cliente, tales como las referidas a reserva de pistas o realización de pedidos en la tienda.
RN-04	Esta plataforma está sujeta a la Ley Orgánica 3/2018 de Protección de Datos Personales y garantía de los derechos digitales.
RN-05	Tanto los clientes como el administrador se autenticarán en la aplicación de la misma manera.
RN-06	Únicamente el administrador podrá acceder a la gestión de pistas y a la gestión de la tienda.
RN-07	Para poder realizar un pedido o realizar una reserva de pista, un usuario debe estar autenticado como cliente.
RN-08	Un Administrador podrá eliminar la cuenta de un Cliente si considera que este infringe las normas de la plataforma.

3.3. Requisitos de Usuario

En este apartado trataremos los requisitos de usuario de la aplicación, así como la especificación de los casos de uso y sus diagramas. Pese a que todos los actores forman parte del sistema global que es la aplicación, para hacer más sencilla la comprensión presentaremos los diagramas de casos de uso separados por actor.

3.3.1. Casos de Uso

Actor Usuario Anónimo

ID	Nombre	Actor
RU-01	Registrar usuario	Usuario Anónimo
RU-02	Visualizar lista productos	Usuario Anónimo
RU-03	Seleccionar producto	Usuario Anónimo

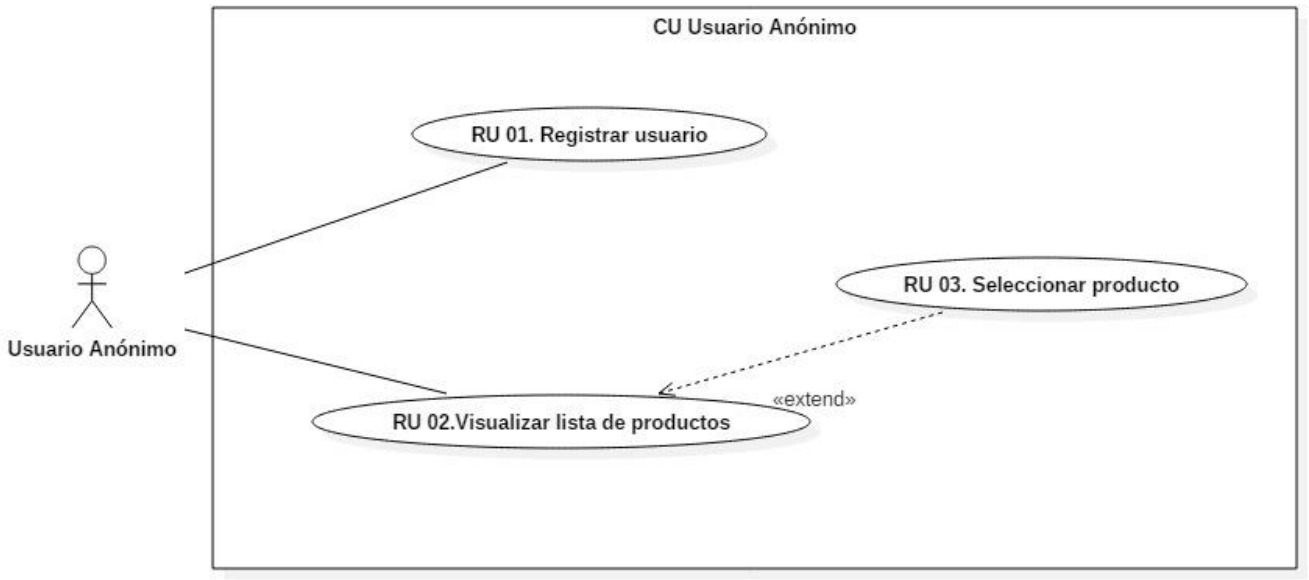

FIGURA 7: DIAGRAMA CASOS DE USO USUARIO ANÓNIMO

Actor Usuario Registrado

ID	Nombre	Actor
RU-04	Identificar usuario	Usuario Registrado
RU-05	Cerrar sesión	Usuario Registrado
RU-06	Visualizar perfil	Usuario Registrado
RU-07	Modificar perfil	Usuario Registrado
RU-08	Visualizar chat	Usuario Registrado
RU-09	Enviar mensaje	Usuario Registrado

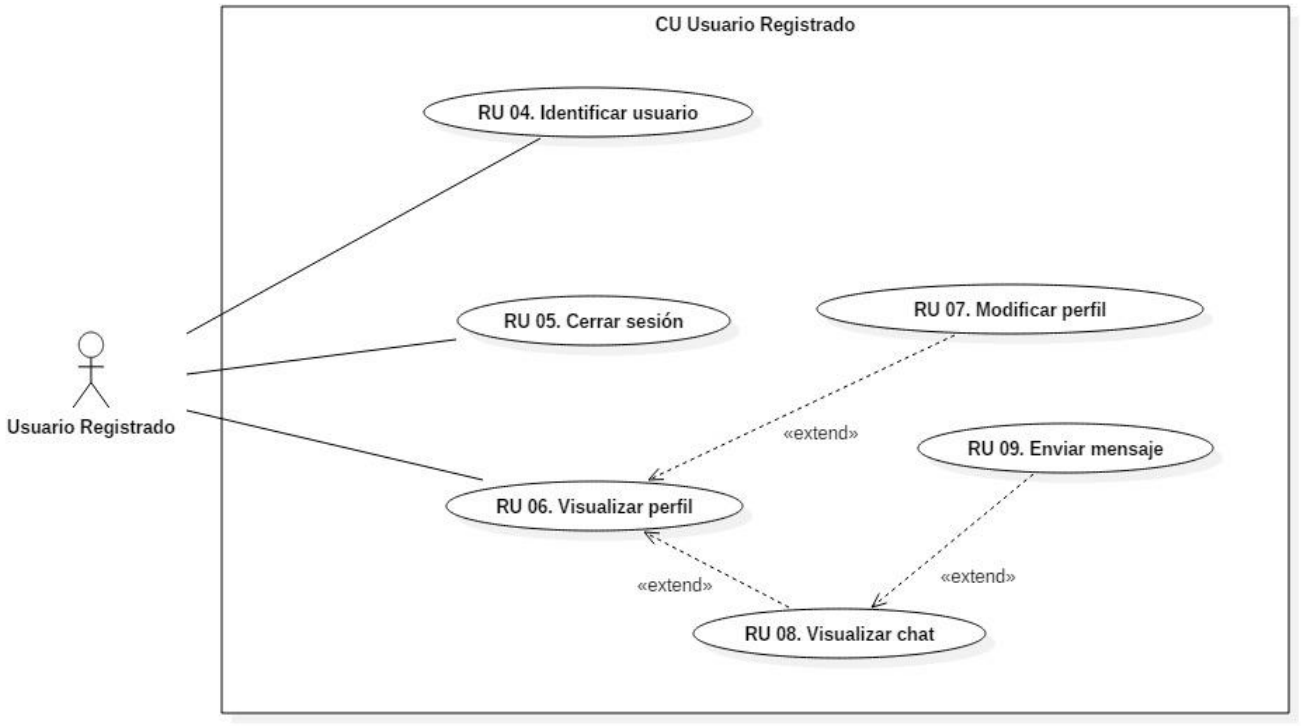

FIGURA 8: DIAGRAMA CASOS DE USO USUARIO REGISTRADO

Actor Cliente

ID	Nombre	Actor
RU-10	Visualizar reservas activas	Cliente
RU-11	Visualizar pedidos	Cliente
RU-12	Visualizar cesta	Cliente
RU-13	Eliminar producto de cesta	Cliente
RU-14	Añadir producto a cesta	Cliente
RU-15	Realizar pedido	Cliente
RU-16	Pagar pedido	Cliente
RU-17	Reservar	Cliente
RU-18	Cancelar reserva	Cliente

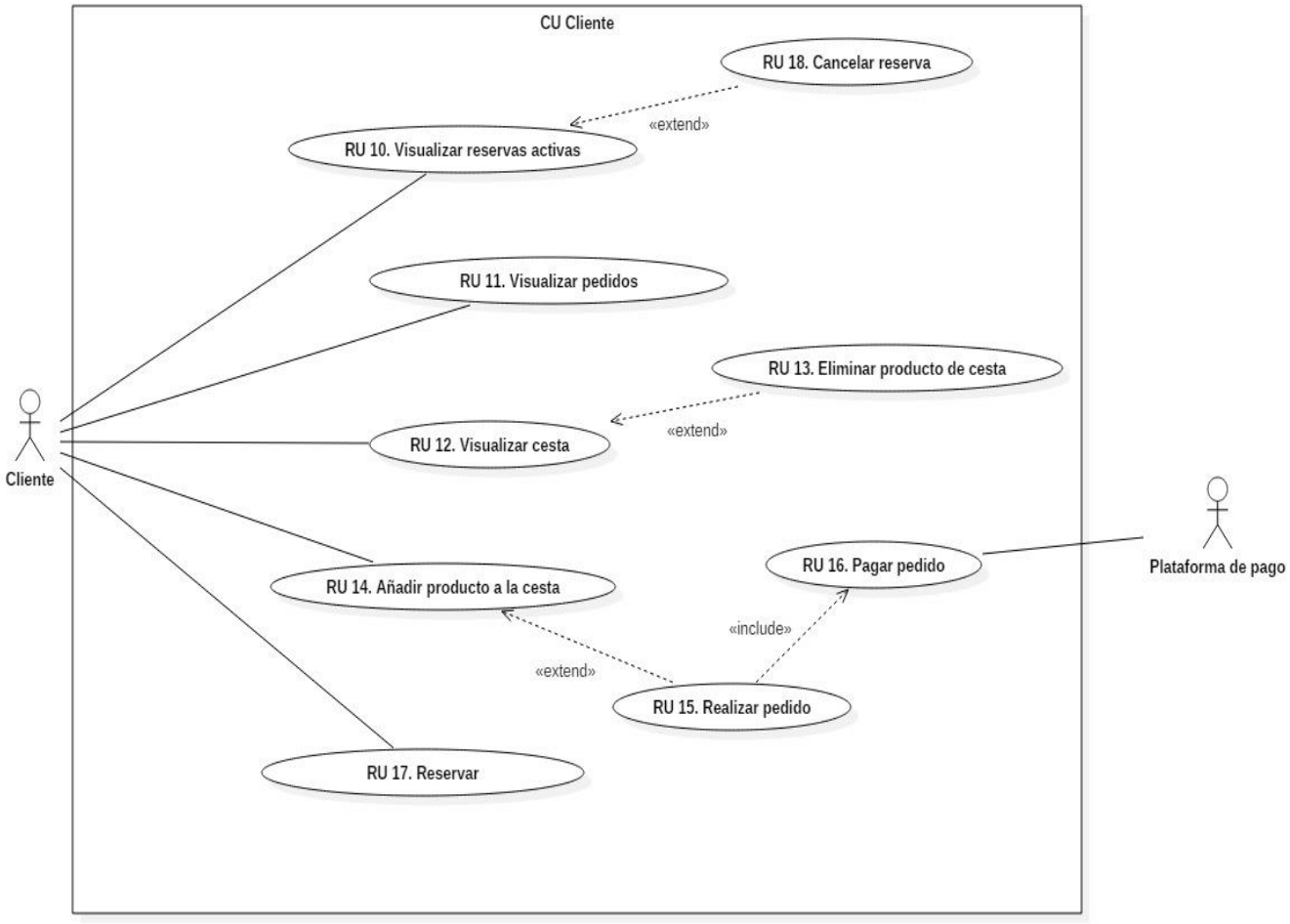

FIGURA 9: DIAGRAMA CASOS DE USO CLIENTE

Actor Administrador

ID	Nombre	Usuario
RU-19	Visualizar catálogo	Administrador
RU-20	Añadir producto	Administrador
RU-21	Modificar producto	Administrador
RU-22	Eliminar producto	Administrador
RU-23	Visualizar pedidos	Administrador
RU-24	Gestionar pedido	Administrador
RU-25	Visualizar lista de pistas	Administrador
RU-26	Añadir pista	Administrador
RU-27	Eliminar pista	Administrador
RU-28	Modificar pista	Administrador
RU-29	Visualizar lista de usuarios	Administrador
RU-30	Eliminar cuenta de usuario	Administrador

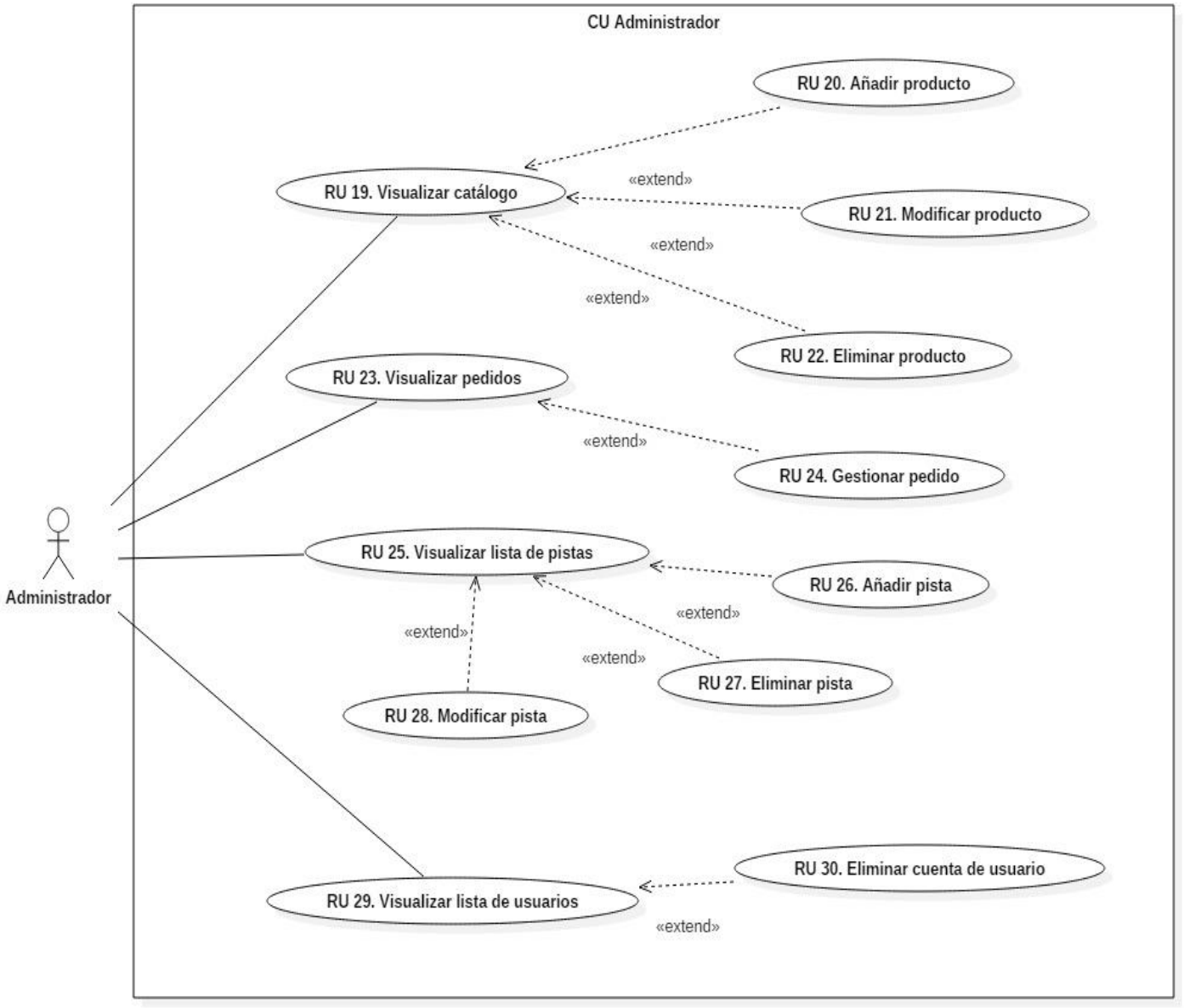

FIGURA 10: DIAGRAMA CASOS DE USO ADMINISTRADOR

3.3.2. Especificación de Casos de Uso

Actor AC-01: Usuario Anónimo

RU-01	**Registrar usuario**		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Anónimo darse de alta en la aplicación.		
Precondición			
Flujo normal	1. El usuario anónimo selecciona la opción de registrarse en el sistema. 2. El sistema muestra los campos que debe rellenar el usuario anónimo. 3. El usuario anónimo rellena los campos. 4. El sistema valida los campos. 5. El flujo finaliza correctamente.		
Postcondición	El Usuario Anónimo pasa a ser un Usuario Registrado.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que los datos introducidos son erróneos. 2. Retorna al punto 2. 4b. Email repetido. 1. El sistema notifica que el email introducido ya está en uso. 2. Retorna al punto 2.		

RU-02	**Visualizar lista de productos**		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Anónimo visualizar el contenido de la tienda.		
Precondición			
Flujo normal	1. El usuario anónimo solicita ver los productos de la tienda. 2. El sistema muestra al usuario los productos de la tienda. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario anónimo visualiza la lista de productos de la tienda.		
Flujo alternativo			

RU-03	Seleccionar producto		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Anónimo seleccionar un producto de la tienda.		
Precondición	El usuario debe haber visualizado la lista de productos de la tienda.		
Flujo normal	1. El usuario anónimo selecciona un producto para visualizar su información. 2. El sistema muestra al usuario los detalles del producto seleccionado. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario anónimo visualiza los detalles de un producto.		
Flujo alternativo			

Actor AC-02: Usuario Registrado

RU-04	Identificar usuario		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado identificarse en la aplicación.		
Precondición	Estar registrado en el sistema.		
Flujo normal	1. El usuario registrado solicita identificarse en el sistema. 2. El sistema solicita los datos necesarios para iniciar sesión. 3. El usuario rellena los campos con la información requerida. 4. El sistema valida los datos. 5. El usuario se autentica en el sistema correctamente. 6. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado se ha identificado en el sistema.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que los datos introducidos son erróneos. 2. Retorna al punto 2.		

RU-05	Cerrar sesión		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado salir del sistema.		
Precondición	Estar identificado en el sistema.		
Flujo normal	1. El usuario solicita al sistema cerrar sesión. 2. El sistema cierra su sesión. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado cierra su sesión.		
Flujo alternativo			

RU-06	Visualizar perfil		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado visualizar la información de su perfil.		
Precondición	Estar identificado en el sistema.		
Flujo normal	1. El usuario registrado solicita al sistema visualizar su perfil. 2. El sistema proporciona al usuario la información de su perfil. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado visualiza la información de su perfil.		
Flujo alternativo			

RU-07	Modificar perfil		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado modificar la información de su perfil de usuario.		
Precondición	El Usuario Registrado debe haber visualizado la información de su perfil.		
Flujo normal	1. El usuario registrado solicita al sistema modificar su perfil de usuario. 2. El sistema muestra los datos que el usuario puede modificar de su perfil. 3. El usuario registrado cambia la información que quiera de su perfil. 4. El sistema valida los cambios introducidos y modifica el perfil del usuario. 5. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha cambiado la información de su perfil.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que los datos introducidos son erróneos. 2. Retorna al punto 2.		

RU-08	Visualizar chat		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado visualizar el chat entre usuarios.		
Precondición	El Usuario Registrado debe haber visualizado la información de su perfil.		
Flujo normal	1. El usuario registrado solicita visualizar el chat entre usuarios. 2. El sistema muestra al usuario registrado el chat entre usuarios. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha visualizado el chat entre usuarios.		
Flujo alternativo			

RU-09	Enviar mensaje		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Usuario Registrado enviar un mensaje por el chat entre usuarios.		
Precondición	El Usuario Registrado debe haber visualizado la información de su perfil.		
Flujo normal	1. El usuario registrado solicita enviar un mensaje por el chat entre usuarios. 2. El sistema da la opción al usuario de escribir el mensaje. 3. El usuario registrado escribe el mensaje. 4. El usuario confirma el mensaje. 5. El sistema envía el mensaje. 6. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha enviado un mensaje por el chat entre usuarios.		
Flujo alternativo			

Actor AC-03: Cliente

RU-10	Visualizar reservas activas		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente visualizar las reservas que tiene activas.		
Precondición	El Cliente debe haber visualizado la información de su perfil.		
Flujo normal	1. El cliente solicita visualizar sus reservas activas. 2. El sistema muestra al cliente la información. 3. El flujo finaliza correctamente.		
Postcondición	El Cliente ha visualizado sus reservas activas.		
Flujo alternativo			

RU-11	Visualizar pedidos		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente visualizar sus pedidos.		
Precondición	El Cliente debe haber visualizado la información de su perfil.		
Flujo normal	1. El cliente solicita visualizar sus pedidos. 2. El sistema muestra al cliente la información. 3. El flujo finaliza correctamente.		
Postcondición	El Cliente ha visualizado sus pedidos.		
Flujo alternativo			

RU-12	Visualizar cesta		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente visualizar su cesta de productos.		
Precondición	Haber visualizado la lista de productos.		
Flujo normal	1. El cliente solicita visualizar el contenido de su cesta de productos. 2. El sistema muestra la información al cliente 3. El flujo finaliza correctamente.		
Postcondición	El Cliente ha visualizado su cesta de productos.		
Flujo alternativo			

RU-13	Eliminar producto de cesta		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente eliminar un producto de su cesta de productos.		
Precondición	El Cliente debe haber añadido algún producto a su cesta y haberla visualizado.		
Flujo normal	1. El cliente solicita eliminar un producto específico de su cesta. 2. El sistema elimina dicho producto de la cesta del cliente. 3. El flujo finaliza correctamente.		
Postcondición	El Cliente ha eliminado un producto de su cesta.		
Flujo alternativo			

RU-14	Añadir producto a la cesta		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente añadir un producto de la tienda a su cesta.		
Precondición	El Cliente debe haber visualizado los detalles de un producto de la tienda.		
Flujo normal	1. El cliente solicita añadir un producto específico a su cesta. 2. El sistema añade dicho producto a la cesta del cliente. 3. El flujo finaliza correctamente.		
Postcondición	El Cliente ha añadido un producto a su cesta.		
Flujo alternativo			

RU-15	Realizar pedido		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente realizar un pedido.		
Precondición	El Cliente debe haber añadido algún producto a su cesta.		
Flujo normal	1. El usuario registrado solicita al sistema realizar un pedido. 2. El sistema lleva al cliente a la funcionalidad de pagar pedido. 3. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha cambiado la información de su perfil.		
Flujo alternativo			

RU-16	Pagar pedido		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente realizar el pago de un pedido a través de la Plataforma de Pago.		
Precondición	El Cliente debe estar en proceso de realizar un pedido.		
Flujo normal	1. El cliente solicita al sistema realizar el pago de un pedido. 2. El sistema redirige al cliente a la plataforma de pago. 3. El cliente introduce los datos necesarios para realizar el pago. 4. La plataforma de pago valida los datos introducidos. 5. El flujo finaliza correctamente.		
Postcondición	El Cliente ha realizado el pago de un pedido correctamente.		
Flujo alternativo	4a. Datos erróneos. 3. La plataforma de pago notifica que los datos introducidos son erróneos. 4. Retorna al punto 3.		

RU-17	Reservar		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente seleccionar la hora y pista para realizar una reserva.		
Precondición	El Cliente debe estar autenticado en el sistema.		
Flujo normal	1. El cliente solicita al sistema realizar una reserva de pista. 2. El sistema pide al cliente que seleccione la pista y la fecha de su reserva. 3. El cliente selecciona la pista y la fecha de su reserva. 4. El sistema muestra, las horas libres de esa pista en esa fecha. 5. El cliente selecciona la hora de su reserva. 6. El flujo finaliza correctamente.		
Postcondición	El Cliente ha seleccionado pista, fecha y hora para su reserva.		
Flujo alternativo	4a. No hay disponibilidad. 1. El sistema notifica al cliente que no hay disponibilidad para la pista y fecha elegidas. 2. Retorna al punto 2.		

RU-18	Cancelar reserva		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Cliente cancelar una reserva activa.		
Precondición	El Cliente debe haber visualizado sus reservas activas.		
Flujo normal	1. El cliente solicita al sistema cancelar una reserva activa. 2. El sistema pide al cliente que confirme la acción. 3. El sistema valida los cambios producidos. 4. El flujo finaliza correctamente.		
Postcondición	El Cliente ha cancelado una reserva con éxito.		
Flujo alternativo			

Actor AC-04: Administrador

RU-19	Visualizar catálogo		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador visualizar el catálogo actual de la tienda.		
Precondición	El Administrador debe haberse autenticado en el sistema.		
Flujo normal	1. El administrador solicita al sistema visualizar el catálogo. 2. El sistema muestra el catálogo al administrador. 3. El flujo finaliza correctamente.		
Postcondición	El Administrador ha visualizado correctamente el catálogo de la tienda.		
Flujo alternativo			

RU-20	Añadir producto		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador añadir un producto al catálogo de la tienda.		
Precondición	El Administrador debe haber visualizado el catálogo de la tienda.		
Flujo normal	1. El administrador solicita al sistema añadir un nuevo producto al catálogo de la tienda. 2. El sistema muestra al administrador los campos que debe rellenar. 3. El administrador introduce la información del nuevo producto en los campos. 4. El sistema valida los cambios. 5. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha cambiado la información de su perfil.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que los datos introducidos son erróneos. 2. Retorna al punto 2.		

RU-21	Modificar producto		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador modificar las características de un producto del catálogo.		
Precondición	El Administrador debe haber visualizado el catálogo de la tienda.		
Flujo normal	1. El administrador solicita al sistema modificar los datos de un producto del catálogo. 2. El sistema muestra los datos que el usuario puede modificar del producto. 3. El administrador cambia la información que desee del producto. 4. El sistema valida los cambios introducidos y modifica la información del producto. 5. El flujo finaliza correctamente.		
Postcondición	El Usuario Registrado ha cambiado la información de su perfil.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que los datos introducidos son erróneos. 2. Retorna al punto 2.		

RU-22	Eliminar producto		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador eliminar un producto del catálogo de la tienda.		
Precondición	El Administrador debe haber visualizado el catálogo de la tienda.		
Flujo normal	1. El administrador solicita al sistema eliminar un producto determinado del catálogo. 2. El sistema pide al administrador que confirme la acción. 3. El administrador confirma la acción. 4. El sistema valida los cambios producidos. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha eliminado un producto del catálogo.		
Flujo alternativo			

RU-23	Visualizar pedidos		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador visualizar los pedidos que se han realizado.		
Precondición	El Administrador debe estar autenticado en el sistema.		
Flujo normal	1. El administrador solicita al sistema visualizar los pedidos realizados. 2. El sistema muestra al administrador los pedidos que se han realizado. 3. El flujo finaliza correctamente.		
Postcondición	El Administrador ha visualizado los pedidos realizados.		
Flujo alternativo			

RU-24	Gestionar pedido		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador gestionar un pedido realizado por un Cliente.		
Precondición	El Administrador debe haber visualizado los pedidos realizados.		
Flujo normal	1. El administrador solicita al sistema visualizar los datos de un pedido específico. 2. El sistema muestra los datos del pedido al administrador. 3. El administrador cambia el estado del pedido. 4. El sistema valida los cambios introducidos y modifica la información del pedido. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha gestionado un pedido.		
Flujo alternativo			

RU-25	Visualizar lista de pistas		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador visualizar la lista de pistas que hay en el sistema.		
Precondición	El Administrador debe estar autenticado en el sistema.		
Flujo normal	1. El administrador solicita al sistema visualizar las pistas que hay en el sistema. 2. El sistema muestra la lista de pistas al administrador. 3. El flujo finaliza correctamente.		
Postcondición	El Administrador ha visualizado las pistas que hay en el sistema.		
Flujo alternativo			

RU-26	Añadir pista		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador añadir una Pista al sistema.		
Precondición	El Administrador debe haber visualizado las pistas que hay en el sistema.		
Flujo normal	1. El administrador solicita al sistema dar de alta una nueva pista. 2. El sistema muestra los datos que el administrador debe rellenar. 3. El administrador introduce la información necesaria. 4. El sistema valida los cambios introducidos y da de alta la nueva pista. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha dado de alta una nueva pista en el sistema.		
Flujo alternativo	4a. Nombre duplicado. 3. El sistema notifica que el nombre de la pista ya está en uso. 4. Retorna al punto 2.		

RU-27	Eliminar pista		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador eliminar una Pista del sistema.		
Precondición	El Administrador debe haber visualizado la lista de pistas.		
Flujo normal	1. El administrador solicita al sistema dar de baja una pista del sistema. 2. El sistema pide al administrador que confirme la acción. 3. El administrador confirma la acción. 4. El sistema valida los cambios y elimina la pista del sistema. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha dado de baja una pista del sistema.		
Flujo alternativo			

RU-28	Modificar pista		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador modificar los datos de una Pista.		
Precondición	El Administrador debe haber visualizado la lista de pistas.		
Flujo normal	1. El administrador solicita al sistema modificar los datos de la pista seleccionada. 2. El sistema muestra los campos modificables de dicha pista al administrador. 3. El administrador modifica los datos que considere. 4. El sistema valida los cambios. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha modificado los datos de una Pista.		
Flujo alternativo	4a. Datos erróneos. 1. El sistema notifica que algunos de los cambios introducidos por el administrador son erróneos. 2. Retorna al punto 2.		

RU-29	Visualizar lista de usuarios		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador visualizar la lista de usuarios del sistema.		
Precondición	El Administrador debe estar autenticado en el sistema.		
Flujo normal	1. El administrador solicita al sistema visualizar la lista de usuarios. 2. El sistema muestra la lista de usuarios al administrador. 3. El flujo finaliza correctamente.		
Postcondición	El Administrador ha visualizado la lista de usuarios del sistema.		
Flujo alternativo			

RU-30	Eliminar cuenta de usuario		
Versión	1.0	**Autor**	Jorge Calvo
Descripción	El sistema permite a un Administrador eliminar la cuenta de un Usuario Registrado.		
Precondición	El Administrador debe haber visualizado la lista de usuarios del sistema.		
Flujo normal	1. El administrador solicita al sistema eliminar un usuario. 2. El sistema pide al administrador confirmar la acción. 3. El administrador confirma la acción. 4. El sistema valida los cambios. 5. El flujo finaliza correctamente.		
Postcondición	El Administrador ha eliminado un Usuario Registrado.		
Flujo alternativo			

3.4. Requisitos de Información

En este apartado detallaremos el análisis de los datos utilizados en la aplicación. Esto se llevará a cabo mediante un diagrama Entidad-Relación y la especificación de los requisitos de información ahí expuestos.

3.4.1. Diagrama Entidad-Relación

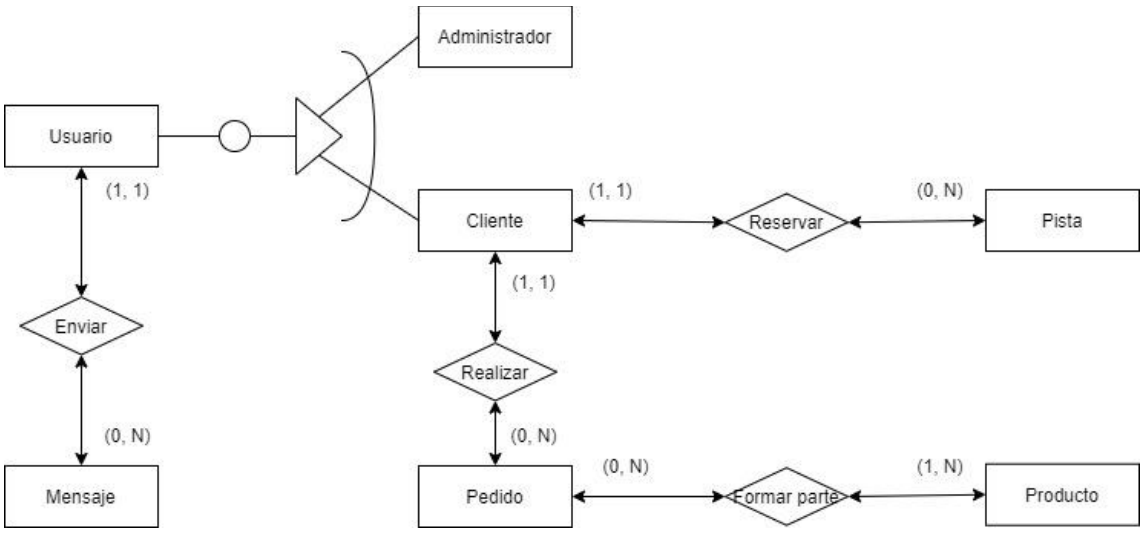

FIGURA 11: DIAGRAMA ENTIDAD-RELACIÓN

3.4.2. Entidades y relaciones

RI-E01	Usuario					
Definición	Esta entidad define al usuario base de la aplicación, del cual heredan las entidades Administrador y Cliente.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-E01.1	ID	Contiene el identificador único del usuario.	Integer	Si	No	Clave primaria
RI-E01.2	Nombre	Contiene el nombre del usuario.	Varchar	No	No	
RI-E01.3	Apellidos	Contiene los apellidos del usuario.	Varchar	No	No	
RI-E01.4	Password	Contiene la contraseña del usuario.	Varchar	No	No	
RI-E01.5	Email	Contiene el email del usuario.	Varchar	Si	No	
RI-E01.6	Fecha de nacimiento	Contiene la fecha de nacimiento del usuario.	Date	No	No	
RI-E01.7	Dirección	Contiene la dirección del usuario.	Varchar	No	Sí	
RI-E01.8	Municipio	Contiene el municipio de residencia del usuario.	Varchar	No	Sí	
RI-E01.9	Provincia	Contiene la provincia de residencia del usuario.	Enum	No	Sí	
RI-E01.10	Teléfono	Contiene el teléfono del usuario.	Integer	No	No	
RI-E01.11	Dni	Contiene el dni del usuario.	Integer	No	Sí	

RI-E02	Administrador					
Definición	Esta entidad define al superusuario de la aplicación, la cual es una entidad hija de Usuario, por lo que hereda sus atributos.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas

RI-E03	Cliente					
Definición	Esta entidad define al usuario estándar de la aplicación, la cual es una entidad hija de Usuario, así que hereda sus atributos.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas

RI-E04	Pista					
Definición	Esta entidad define la pista, la cual podrá ser reservada por el Cliente.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-E04.1	ID	Contiene el identificador único de la pista.	Integer	Si	No	Clave primaria
RI-E04.2	Nombre	Contiene el nombre de la pista.	Varchar	Si	No	
RI-E04.3	Disponibilidad	Indica la disponibilidad de la pista.	Boolean	No	No	

RI-E05	Producto					
Definición	Esta entidad define al producto.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-E05.1	ID	Contiene el identificador único del producto.	Varchar	Si	No	Clave primaria
RI-E05.2	Nombre	Contiene el nombre del producto.	Varchar	Sí	No	
RI-E05.3	Descripción	Contiene la descripción del producto.	Varchar	No	No	
RI-E05.4	Precio	Contiene el precio del producto.	Float	No	No	
RI-E05.5	Imagen	Contiene la imagen del producto.	Byte	No	No	
RI-E05.6	Stock	Contiene el stock del producto.	Integer	No	No	
RI-E05.7	Categoría	Contiene la categoría a la que pertenece el producto.	Enum	No	No	

RI-E06	Pedido					
Definición	Esta entidad define al pedido realizado por un cliente.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-E06.1	ID	Contiene el id del pedido.	Integer	Si	No	Clave primaria
RI-E06.2	Id_Cliente	Contiene el identificador del cliente.	Integer	No	No	Clave foránea
RI-E06.3	Fecha	Contiene la fecha de realización del pedido.	Date	No	No	
RI-E06.4	Coste	Contiene el coste total del pedido.	Float	Si	No	
RI-E06.5	Estado	Contiene el estado actual del pedido.	Enum	No	No	

RI-E07	Mensaje					
Definición	Esta entidad define el mensaje enviado por un usuario.					
Consideraciones	•					
Atributos						
ID	Nombre	Descripción	Dominio	UNIQUE	Null	Notas
RI-E07.1	ID	Contiene el id del mensaje.	Integer	Si	No	Clave primaria
RI-E07.2	Remitente	Contiene el identificador del usuario.	Integer	No	No	Clave foránea
RI-E07.3	Contenido	Contiene el contenido del mensaje.	Varchar	No	No	
RI-E07.4	Fecha_hora	Contiene la fecha y la hora en la que se envió el mensaje.	Date	No	No	

RI-R01	Reservar					
Definición	Esta relación establece los detalles de una reserva de pista.					
Consideraciones	•					
Entidades						
ID	Nombre	Cardinalidad	Notas			
RI-E03	Cliente	1, 1				
RI-E04	Pista	0, N				
Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-R01.1	ID	Contiene el identificador único de la reserva.	Integer	Si	No	Clave Primaria
RI-R01.2	Id_cliente	Contiene el identificador del Cliente que ha realizado la reserva.	Integer	No	No	Clave foránea
RI-R01.3	Id_pista	Contiene el identificador de la pista reservada.	Integer	No	No	Clave foránea
RI-R01.4	Fecha	Contiene la fecha de la reserva.	Date	No	No	
RI-R01.5	Sesión	Contiene la sesión de la reserva.	Enum	No	No	

RI-R02	Realizar		
Definición	Esta relación establece los detalles de la realización de un pedido por un cliente.		
Consideraciones	•		
Entidades			
ID	Nombre	Cardinalidad	Notas
RI-E03	Cliente	1, 1	
RI-E06	Pedido	0, N	

RI-R03	Formar parte
Definición	Esta relación establece los detalles al añadir un producto a un pedido.
Consideraciones	•

Entidades			
ID	Nombre	Cardinalidad	Notas
RI-E05	Producto	1, N	
RI-E06	Pedido	0, N	

Atributos						
ID	Nombre	Descripción	Dominio	Unique	Null	Notas
RI-R01.1	ID	Contiene el identificador único del producto del pedido.	Integer	Si	No	Clave Primaria
RI-R01.2	Id_pedido	Contiene el identificador del pedido.	Integer	No	No	Clave foránea
RI-R01.3	Id_producto	Contiene el identificador del producto.	Integer	No	No	Clave foránea
RI-R01.4	Cantidad	Contiene el número de unidades de un producto que hay en el pedido.	Integer	No	No	
RI-R01.5	Precio_actual	Contiene el precio del producto en el momento de la venta.	Float	No	No	

RI-R04	Gestionar
Definición	Esta relación establece los detalles de un cambio en una pista.
Consideraciones	•

Entidades			
ID	Nombre	Cardinalidad	Notas
RI-E02	Administrador	1, N	
RI-E04	Pista	0, N	

RI-R05	Enviar		
Definición	Esta relación establece los detalles de un envío de mensaje.		
Consideraciones	•		
Entidades			
ID	Nombre	Cardinalidad	Notas
RI-E01	Usuario	1, 1	
RI-E07	Mensaje	0, N	

3.5. Requisitos No Funcionales

Los requisitos no funcionales son aquellos que definen ciertas restricciones del sistema que tienen que ser cumplidas, relativas a la seguridad, robustez, etc.

Seguridad	
ID	**Descripción**
AC-01	Los datos ingresados por teclado serán validados correctamente por la aplicación.
AC-02	Las contraseñas de los usuarios se almacenarán utilizando el algoritmo de hashing SHA256.
AC-03	La contraseña de un usuario deberá tener una longitud mínima de 8 caracteres.
AC-04	Un usuario no puede visualizar la información personal de otro usuario.
AC-05	Los usuarios anónimos únicamente podrán acceder a las partes públicas de la aplicación.
AC-06	Una vez el usuario cierre la sesión, sus datos de sesión deberán ser eliminados.

Usabilidad	
ID	**Descripción**
AC-07	La aplicación deberá ser intuitiva y fácil de usar.
AC-08	La aplicación deberá estar disponible en castellano.

Robustez	
ID	**Descripción**
AC-09	El sistema estará preparado para hacer frente a situaciones anómalas.
AC-10	El sistema mostrará mensajes de aviso si detecta fallos al validar datos en la parte de cliente.
AC-11	El sistema mostrará mensajes de aviso si detecta fallos en la parte de servidor.

Disponibilidad	
ID	**Descripción**
AC-12	La aplicación deberá estar disponible 24x7.

3.6. Requisitos Funcionales

Los requisitos funcionales delimitan todas las acciones que el sistema debe llevar a cabo.

- **RU-01: Registrar usuario**
  - **RF-01:** El sistema habilita la función de registrarse en el sistema.
  - **RF-02**: El sistema solicita al usuario anónimo los datos de registro.
  - **RF-03**: El sistema valida los cambios.
  - **RF-04**: El sistema notifica al usuario anónimo si hay algún dato incorrecto.
  - **RF-05**: El sistema notifica al usuario anónimo si el email introducido ya está en uso.
  - **RF-06**: El sistema llevará acabo el registro si los datos son correctos.

- **RU-02: Visualizar lista de productos**
  - **RF-07**: El sistema habilita la opción de visualizar la lista de productos de la tienda.
  - **RF-08**: El sistema muestra al usuario dicha lista.

- **RU-03: Seleccionar producto**
  - **RF-09**: El sistema comprueba si el usuario ha seleccionado un producto de la lista.
  - **RF-10**: El sistema muestra al usuario la información de dicho producto.

- **RU-04: Identificar usuario**
  - **RF-11**: El sistema habilita la opción de identificarse en el sistema.
  - **RF-12**: El sistema solicita los datos necesarios para iniciar sesión.
  - **RF-13**: El sistema valida los datos.
  - **RF-14**: El sistema inicia sesión si los datos introducidos son correctos.
  - **RF-15**: El sistema notifica al usuario si hay algún dato incorrecto.

- **RU-05: Cerrar sesión**
  - **RF-16**: El sistema habilita la opción de cerrar sesión.
  - **RF-17**: El sistema cierra la sesión del usuario.

- **RU-06: Visualizar perfil**
  - **RF-18**: El sistema permite al usuario registrado solicitar visualizar su perfil de usuario.
  - **RF-19**: El sistema muestra al usuario registrado los datos de su perfil.

- **RU-07: Modificar perfil**
  - **RF-20**: El sistema comprueba que se haya visualizado el perfil del usuario registrado.
  - **RF-21**: El sistema permite al usuario registrado solicitar modificar su perfil.
  - **RF-22**: El sistema habilita los campos a modificar.
  - **RF-23**: El sistema valida y guarda los cambios realizados por el usuario registrado.
  - **RF-24**: El sistema no almacena los datos modificados si existe algún error en ellos.
  - **RF-25**: El sistema no almacena los datos modificados si existe algún campo en blanco.

- **RU-08: Visualizar chat**
  - **RF-26**: El sistema habilita la opción de visualizar el chat entre usuarios.
  - **RF-27**: El sistema muestra al usuario registrado el chat entre usuarios.

- **RU-09: Enviar mensaje**
  - **RF-28**: El sistema da la opción al usuario registrado de escribir un mensaje en el chat entre usuarios.
  - **RF-29**: El sistema envía el mensaje escrito por el usuario registrado.

- **RU-10: Visualizar reservas activas**
  - **RF-30**: El sistema da la opción al cliente de visualizar sus reservas activas.
  - **RF-31**: El sistema muestra al cliente sus reservas activas.

- **RU-11: Visualizar pedidos en curso**
  - **RF-32**: El sistema da la opción al cliente de visualizar sus pedidos activos.
  - **RF-33**: El sistema muestra al cliente sus pedidos activos.

- **RU-12: Visualizar cesta**
  - **RF-34**: El sistema permite al cliente visualizar el contenido de su cesta de productos.
  - **RF-35**: El sistema muestra al cliente el contenido de su cesta de productos.

- **RU-13: Eliminar producto de cesta**
  - **RF-36**: El sistema habilita la opción para que el cliente elimine un producto de su cesta.
  - **RF-37**: El sistema elimina de la cesta el producto seleccionado por el cliente.

- **RU-14: Añadir producto a la cesta**
  - **RF-38**: El sistema habilita la opción para que el cliente añada un producto a su cesta.
  - **RF-39**: El sistema añade a la cesta el producto seleccionado por el cliente.

- **RU-15: Realizar pedido**
  - **RF-40**: El sistema habilita la opción para que el cliente realice un pedido.
  - **RF-41**: El sistema lleva al cliente a la funcionalidad de pagar pedido.

- **RU-16: Pagar pedido**
  - **RF-42**: El sistema habilita la opción para que un cliente pague el pedido en curso.
  - **RF-43**: El sistema redirige al cliente a la plataforma de pago.

- **RU-17: Reservar**
  - **RF-44**: El sistema habilita la opción para que el cliente reserve una pista.
  - **RF-45**: El sistema solicita al cliente que seleccione la pista y la fecha de su reserva.
  - **RF-46:** El sistema muestra al cliente las horas libres de esa pista en esa fecha.
  - **RF-47:** El sistema notifica al cliente que no hay disponibilidad para la fecha y la pista elegidas.
  - **RF-48:** El sistema crea la reserva con la pista, fecha y hora seleccionadas por el usuario.

- **RU-18: Cancelar reserva**
  - **RF-49**: El sistema permite al cliente solicitar cancelar una reserva activa.
  - **RF-50**: El sistema solicita al cliente que confirme la acción.
  - **RF-51**: El sistema valida los cambios producidos.

- **RU-19: Visualizar catálogo**
  - **RF-52**: El sistema permite al administrador solicitar visualizar el catálogo de productos.
  - **RF-53**: El sistema muestra el catálogo al administrador.

- **RU-20: Añadir producto**
  - **RF-54**: El sistema habilita la opción de añadir un nuevo producto al catálogo de la tienda.
  - **RF-55**: El sistema muestra al administrador los campos que tiene que rellenar.
  - **RF-56**: El sistema valida los cambios introducidos.
  - **RF-57**: El sistema notifica al administrador si los datos introducidos son erróneos.

- **RU-21: Modificar producto**
  - **RF-58**: El sistema habilita la opción de modificar los datos de un producto.
  - **RF-59**: El sistema muestra al administrador los datos que puede modificar del producto.
  - **RF-60**: El sistema valida los cambios introducidos.
  - **RF-61**: El sistema notifica al administrador si los datos introducidos son erróneos.

- **RU-22: Eliminar producto**
    - **RF-62**: El sistema habilita la opción de eliminar un producto del catálogo de la tienda.
    - **RF-63**: El sistema solicita al administrador que confirme la acción.
    - **RF-64**: El sistema valida los cambios.

- **RU-23: Visualizar pedidos**
    - **RF-65**: El sistema habilita la opción de visualizar los pedidos realizados por todos los clientes.
    - **RF-66**: El sistema muestra al administrador los pedidos realizados.

- **RU-24: Gestionar pedido**
    - **RF-67**: El sistema habilita la opción de visualizar un pedido.
    - **RF-68**: El sistema muestra los datos del pedido al administrador.
    - **RF-69**: El sistema valida los cambios introducidos y modifica la información del pedido.

- **RU-25: Visualizar lista de pistas**
    - **RF-70**: El sistema habilita la opción de visualizar las pistas que hay en el sistema.
    - **RF-71**: El sistema muestra al administrador la lista de pistas que hay en el sistema.

- **RU-26: Añadir pista**
    - **RF-72**: El sistema permite al administrador solicitar dar de alta una nueva pista.
    - **RF-73**: El sistema muestra los campos que el administrador debe rellenar.
    - **RF-74**: El sistema valida los cambios introducidos y da de alta la nueva pista.
    - **RF-75**: El sistema notifica al administrador que el nombre de la pista ya está en uso.

- **RU-27: Eliminar pista**
    - **RF-76**: El sistema permite al administrador solicitar dar de baja una pista del sistema.
    - **RF-77**: El sistema solicita al administrador que confirme la acción.
    - **RF-78**: El sistema valida los cambios y elimina la pista del sistema.

- **RU-28: Modificar pista**
    - **RF-79**: El sistema permite al administrador solicitar modificar los datos de una pista.
    - **RF-80**: El sistema muestra al administrador los campos modificables de la pista.
    - **RF-81**: El sistema valida los cambios introducidos.
    - **RF-82**: El sistema notifica al administrador que hay datos erróneos en las modificaciones.

- **RU-29: Visualizar lista de usuarios**
    - **RF-83**: El sistema permite al administrador solicitar visualizar la lista de usuarios de la aplicación.
    - **RF-84**: El sistema muestra la lista de usuarios al administrador.

- **RU-30: Eliminar cuenta de usuario**
    - **RF-85**: El sistema permite al administrador solicitar eliminar una cuenta de usuario.
    - **RF-86**: El sistema solicita al administrador confirmar la acción.
    - **RF-87**: El sistema valida los cambios.

4. PLANIFICACIÓN Y PRESUPUESTO

4.1. Metodología

En este apartado se procederá a exponer la metodología de desarrollo que se ha seguido durante la realización de este proyecto.

Se ha elegido la metodología de Desarrollo en Cascada, el cual se basa en la división del desarrollo del proyecto en distintas etapas, cada una de las cuales se lleva a cabo según acaba la anterior.

El principal motivo que hemos tenido en cuenta para la elección de esta metodología frente a otras es que en nuestro caso los requisitos están muy claros, por lo que, en principio, no serán susceptibles a cambios.

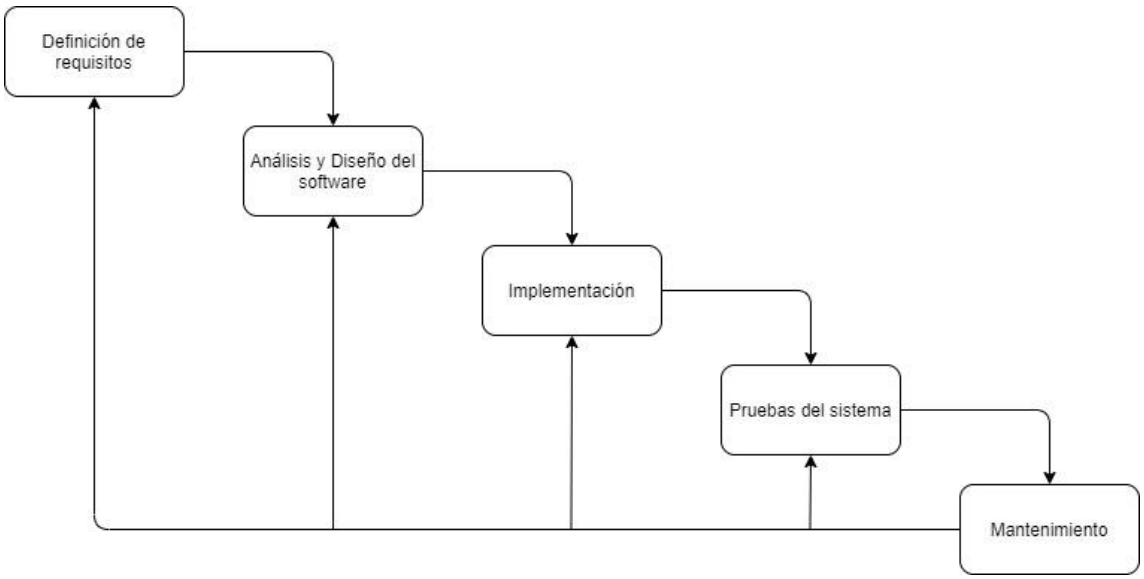

FIGURA 12: METODOLOGÍA DESARROLLO EN CASCADA

Las distintas etapas que conforman este modelo son las siguientes:

- **Definición de requisitos**: En esta primera etapa se especifican los requisitos que deberá cumplir nuestra aplicación, los cuales deberán cubrir todas las necesidades del cliente.
- **Análisis y Diseño del Software**: En esta etapa se definirá la estructura que presentará la aplicación.
- **Implementación**: En esta etapa se lleva a cabo el desarrollo de la aplicación, siguiendo siempre las pautas marcadas por las dos primeras etapas.
- **Pruebas del sistema**: Etapa en la que se comprueba el buen funcionamiento del software y se subsanan posibles errores que se hayan encontrado.
- **Mantenimiento**: Etapa final del proyecto, en la que hay que asegurarse el buen funcionamiento del software después de la entrega al cliente.

4.2. Planificación Temporal

En este apartado de la sección vamos a detallar los aspectos referentes a la planificación del proyecto. Para ello, llevaremos a cabo una medición del esfuerzo necesario para realizar el proyecto y aplicaremos los datos resultantes a un Diagrama de Gantt para definir la duración del proyecto.

4.2.1. Estimación del Esfuerzo

Para llevar a cabo la estimación del esfuerzo necesario se va a emplear el Método de Puntos de Función, más específicamente el Método de Albrecht. Este método se basa en la asignación de complejidades (alta, media, baja) a los distintos parámetros que sirven para evaluar dicha funcionalidad. Dichos parámetros son los siguientes:

- **Entradas de usuario**: Datos que el usuario aporta al sistema.
- **Salidas de usuario**: Datos que el sistema aporta al usuario.
- **Número de consultas externas**: Entradas que requieren de una respuesta por parte del sistema.
- **Número de ficheros lógicos internos** (**FLI**): Ficheros o bases de datos internos al sistema.
- **Número de ficheros externos**: (**FE**): Ficheros o bases de datos externos al sistema.

Explicado esto, se dispondrá a dividir las distintas funcionalidades teniendo en cuenta el grupo al que pertenecen.

ENTRADAS DE USUARIO	
DESCRIPCIÓN	COMPLEJIDAD
Datos de registro	Media
Datos de inicio de sesión	Media
Datos de modificación de perfil de usuario	Media
Mensajes escritos en el chat	Baja
Datos de reserva de pista	Alta
Selección de producto en la tienda	Media
Información de pago	Alta
Datos de registro de producto	Baja
Datos de modificación de producto	Baja
Datos de registro de pista	Baja
Datos de modificación de pista	Baja
Datos de gestión de pedido	Baja

SALIDAS DE USUARIO	
DESCRIPCIÓN	COMPLEJIDAD
Datos de perfil	Media
Listado de reservas activas	Baja
Listado de pedidos realizados	Baja
Información de pedido	Baja
Mensajes del chat	Baja
Listado de fechas y pistas disponibles	Baja
Listado de sesiones disponibles	Media
Listado de productos de la tienda	Baja
Información de producto de la tienda	Baja
Listado de productos añadidos al carrito	Media
Listado de usuarios	Baja
Información de usuario	Baja
Listado de pistas	Baja
Información de pista	Baja
Listado de pedidos	Baja
Información de pedido	Baja
Listado de productos	Baja
Información de producto	Baja

CONSULTAS EXTERNAS	
DESCRIPCIÓN	COMPLEJIDAD
Pago de pedido (PayPal)	Alta

FICHEROS LÓGICOS INTERNOS	
DESCRIPCIÓN	COMPLEJIDAD
Base de Datos (PostgreSQL)	Alta

Para realizar la equivalencia de las distintas complejidades de cada grupo se utilizarán los valores estándar provistos por el *International Function Point Users Group*:

Tipo/Complejidad	Baja	Media	Alta
Entradas de usuario	3 PF	4 PF	6 PF
Salidas de usuario	4 PF	5 PF	7 PF
Consultas externas	3 PF	4 PF	6 PF
Ficheros lógicos internos	7 PF	10 PF	15 PF
Ficheros externos	5 PF	7 PF	10 PF

Ahora obtendremos los puntos de función totales teniendo en cuenta los fatores de complejidad de cada tipo.

Parámetro	Complejidad	N.º total X Complejidad	Total por parámetro	Total
Entradas de usuario ($PFNA_E$)	Baja Media Alta	6 x 3 4 x 4 2 x 6	18 16 12	46
Salidas de usuario ($PFNA_S$)	Baja Media Alta	15 x 4 3 x 5 0 x 7	60 15 0	75
Consultas externas ($PFNA_Q$)	Baja Media Alta	0 x 3 0 x 4 1 x 6	0 0 6	6
Ficheros internos ($PFNA_{FLI}$)	Baja Media Alta	0 x 5 0 x 10 1 x 15	0 0 15	15
Ficheros externos ($PFNA_{FE}$)	Baja Media Alta	0 x 5 0 x 7 0 x 10	0 0 0	0
Total de puntos de función no ajustados				**142 PFNA**

Una vez hemos obtenido los Puntos de Función No Ajustados (PFNA) procederemos a ajustarlos. Para ello, se calculará el Factor de Ajuste (FA), el cual está basado en 14 factores que medirán la complejidad del sistema. Cada uno de estos factores se medirá en una escala de cero a cinco, siendo el cero el valor más bajo y cinco el valor más alto.

FACTORES	COMPLEJIDAD
1. Respaldo y recuperación	3
2. Comunicación de datos	4
3. Procesamiento distribuido	2
4. Rendimiento	4
5. Eficiencia con el usuario final	3
6. Entrada de datos en línea	4
7. Tasa de transacciones	3
8. Actualización en línea	4
9. Facilidad de instalación	2
10. Complejidad de la lógica de procesos internos	3
11. Código diseñado para la reutilización	2
12. Facilidad de operación	3
13. Instalaciones múltiples	2
14. Facilidad de cambios	2
TOTAL	**41**

Habiendo obtenido la tabla de complejidades, podemos obtener el valor del Facto de Ajuste (FA) mediante la siguiente expresión:

$$\mathbf{FA = (0{,}01 * \sum FC) + 0{,}65}$$

$$\mathbf{FA = (0{,}01 * 41) + 0{,}65} \text{ , donde } \sum FC = 41$$

$$\mathbf{FA = 1{,}06}$$

Se ha obtenido un Factor de Ajuste FA = 1,06, el cual nos permitirá obtener los Puntos de Función Ajustados (PFA). Para ello emplearemos la estimación de puntos de función no ajustados obtenida anteriormente, PFNA = 142, y los ajustaremos de acuerdo a la siguiente fórmula:

$$\mathbf{PFA = PFNA * FA}$$

$$\mathbf{PFA = 142 * (1{,}06)}$$

$$\mathbf{PFA = 150{,}52}$$

En último lugar, realizaremos el cálculo de la estimación del tiempo del proyecto, asumiendo que cada punto de función ajustado equivale a cinco horas de trabajo.

$$\mathbf{Tiempo\ estimado = PFA * N^{o}\ Horas/PFA}$$

$$\mathbf{Tiempo\ estimado = 150{,}52 * 5\ horas}$$

$$\mathbf{Tiempo\ estimado = 752{,}6\ horas}$$

El tiempo estimado de duración del proyecto será de 752,6 horas.

Nuestra previsión inicial es empezar el proyecto el día 15 de febrero de 2020, teniendo una jornada de trabajo de 4 horas diarias de lunes a viernes. A continuación, aplicaremos estos datos en el Diagrama de Gantt antes mencionado.

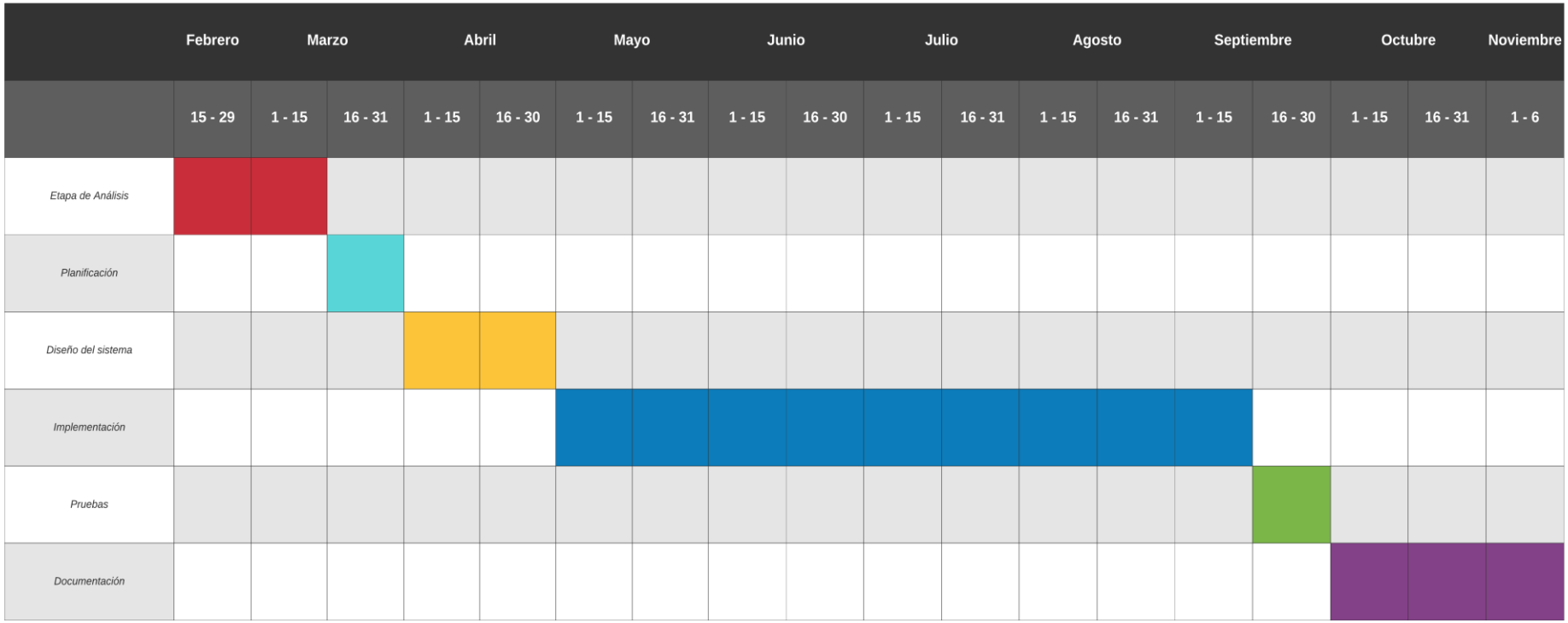

FIGURA 13: DIAGRAMA DE GANTT INICIAL

Atendiendo al Diagrama de Gantt presentado, se estima que el proyecto acabe el día 6 de noviembre de 2020.

4.2.2. Presupuesto

En este apartado detallaremos los costes del proyecto, desglosados por costes de recursos humanos, costes de software y costes de hardware, a partir de los datos generados en los apartados anteriores.

Para estimar los recursos humanos necesarios, se deberá saber el coste de los empleados que realizarán el trabajo, diferenciando entre empleados dependiendo de las funciones de las que se ocupen. En nuestro caso, contaremos con dos categorías de empleado, cuyas remuneraciones estimadas son las siguientes:

- Desarrollador: 10 € por hora.
- Analista: 14 € por hora.

Con estos datos, estimaremos el coste que supondrá realizar el proyecto.

Coste de Recursos Humanos			
Categoría	Horas	Coste/Hora	Total
Desarrollador	436	10,00€	4360,00€
Analista	315	14,00€	4410,00€
Total			**8770,00€**

A continuación detallaremos los costes relativos al software empleado para el desarrollo del proyecto. En algunos casos los programas usados serán de licencia libre, otros serán pruebas gratuitas, y en otros casos serán de pago. Sin embargo, se ha intentado optar por software que no genere gasto alguno para el proyecto. Además, en el software de pago habrá que tener en cuenta la duración de la licencia para saber el coste real que ha tenido cada programa para el proyecto.

Un apunte, la empresa propietaria del software "PyCharm Professional" ofrece una licencia gratuita de su producto a estudiantes y profesores universitarios, la cual hemos usado. Por lo tanto, el uso de esta herramienta no ha supuesto coste alguno.

Todo esto se detalla en la siguiente tabla:

Coste de Software			
Software	Extensión	Coste licencia	Coste real
Windows 10	-	10,00€	10,00€
Microsoft Office 2019	-	11,00€	11,00€
PyCharm Professional	-	-	-
Visual Studio Code	-	-	-
pgAdmin 4	-	-	-
Postman	-	-	-
StarUML	-	-	-
FoxitReader	-	-	-
Google Chrome	-	-	-
Diagrams	-	-	-
NinjaMock	-	-	-
Bloc de notas	-	-	-
Lucid.app	-	-	-
Total			**21,00€**

A continuación mostraremos los costes de los componentes hardware necesarios para realizar el proyecto. Se deberá tener en cuenta la duración del uso de los componentes durante el proyecto, para así saber el coste real que han tenido para el proyecto.

Coste de Hardware				
Software	Porcentaje de uso	Vida útil	Coste	Coste real
Ordenador	15%	5 años	800,00€	120,00€
Monitor	12,5%	6 años	120,00€	15,00€
ADSL	50%	10 meses	50,00€/Mes	250,00€
Total				**385,00€**

En consecuencia, el coste total estimado del proyecto es el siguiente:

Coste	
Recursos Humanos	8770,00€
Software	21,00€
Hardware	385,00€
Total	**9179,00€**

4.3. Coste real

La planificación inicial del proyecto no se ha podido cumplir por diversas razones, lo que ha supuesto un incremento en la duración del proyecto y las horas trabajadas, resultando en 901 horas trabajadas. Esto ha tenido impacto sobre el presupuesto del proyecto.

La distribución de tiempo final se presenta en el siguiente diagrama de Gantt:

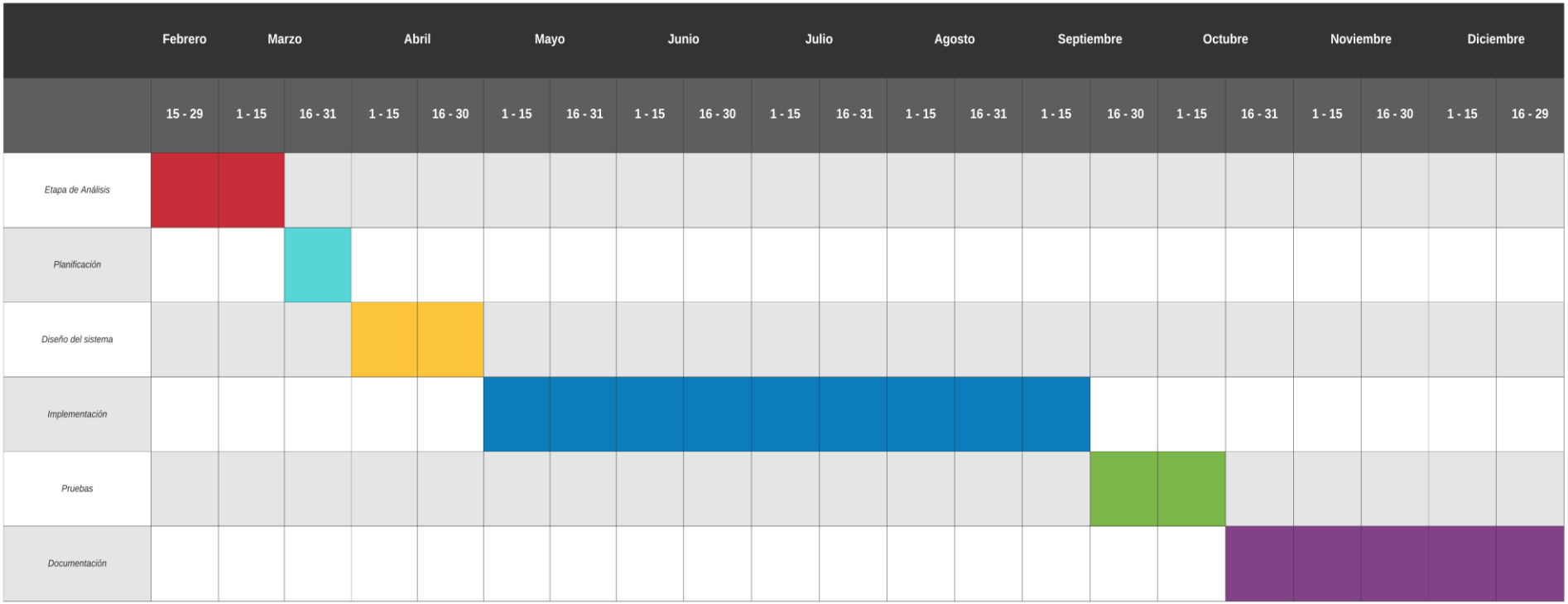

FIGURA 14: DIAGRAMA DE GANTT REAL

En consecuencia el presupuesto final del proyecto ha sido el siguiente:

Coste de Recursos Humanos			
Categoría	Horas	Coste/Hora	Total
Desarrollador	480	10,00€	4800,00€
Analista	421	14,00€	5894,00€
Total			**10694,00€**

Coste de Software			
Software	Extensión	Coste licencia	Coste real
Windows 10	-	10,00€	10,00€
Microsoft Office 2019	-	11,00€	11,00€
PyCharm Professional	-	-	-
Visual Studio Code	-	-	-
pgAdmin 4	-	-	-
Postman	-	-	-
StarUML	-	-	-
FoxitReader	-	-	-
Google Chrome	-	-	-
Diagrams	-	-	-
NinjaMock	-	-	-
Bloc de notas	-	-	-
Lucid.app	-	-	-
Total			**21,00€**

Coste de Hardware				
Software	Porcentaje de uso	Vida útil	Coste	Coste real
Ordenador	18%	5 años	800,00€	144,00€
Monitor	15%	6 años	120,00€	18,00€
ADSL	100%	11 meses	50,00€/Mes	275,00€
Total				**437,00€**

En consecuencia, el coste total del proyecto ha sido el siguiente:

Coste	
Recursos Humanos	10694,00€
Software	21,00€
Hardware	437,00€
Total	**10419,00€**

5. DISEÑO

En esta sección del documento se procederá a exponer los aspectos más importantes del diseño de la aplicación, tanto las diferentes arquitecturas empleadas como el paso al modelo relacional del modelo de datos documentado en la sección de Análisis.

5.1. Arquitectura Lógica

En este apartado se describirá la arquitectura lógica de la aplicación.

Como se ha expuesto anteriormente, la aplicación consta de dos subsistemas muy diferenciados, la API-REST desarrollada en Django y el Front-End que la consume, desarrollado en Angular 9. Por ello, habrá que diferenciar entre ambos subsistemas a la hora de exponer sus arquitecturas.

En lo que respecta a la API-REST, hay que tener en cuenta varias consideraciones. La arquitectura que usa Django por defecto está basada en el patrón Modelo-Vista-Plantilla (MVT, Model-View-Template), el cual deriva del patrón Modelo-Vista-Controlador (MVC, Model-View-Controller). Sin embargo, al considerar Django REST Framework para desarrollar la API-REST, esto ya no es así. Éste framework incluye directamente *routers* y *serializers* y prescinde de las plantillas, ya que éste framework está diseñado específicamente para entornos en los que componentes como Angular se encargan directamente de consumir las funcionalidades de la API-REST, haciendo innecesaria la utilización de las plantillas de la arquitectura original.

Teniendo esto en cuenta, el flujo de datos de nuestra aplicación sería el siguiente:

1. Llega una petición HTTP desde Angular, y el componente *router* se encarga de redireccionar dicha petición a la vista adecuada.
2. Dicha vista, dependiendo de su funcionalidad, usa el serializador que tiene asociado para validar y convertir los datos en tipos nativos de Python.
3. En el modelo, dependiendo del tipo de petición, se crean nuevos registros en la BD o se recuperan datos de ella, ambas operaciones usando Psycopg2, adaptador empleado por Django para bases de datos PostgreSQL. Los datos vuelven a pasar por el serializador para convertirlos de vuelta y ser servidos por la vista.

La arquitectura en el Front-End también es compleja, ya que Angular no emplea exactamente el patrón MVC, si no que más bien se orienta a componentes. Cada uno de dichos componentes consta de cuatro partes: Un archivo typescript, el cual engloba toda la lógica de negocio, un archivo HTML, el cual contiene la parte visual del componente, un archivo CSS, el cual contiene los estilos usados en el HTML, y un servicio, el cual se inyecta en la lógica de negocio y permite realizar las peticiones HTTP a la API-REST. Fuera del componente, tenemos los modelos, los cuales contienen las características de

los objetos que se van a tratar, y las directivas, las cuales son nativas de Angular y permiten manipular el DOM.

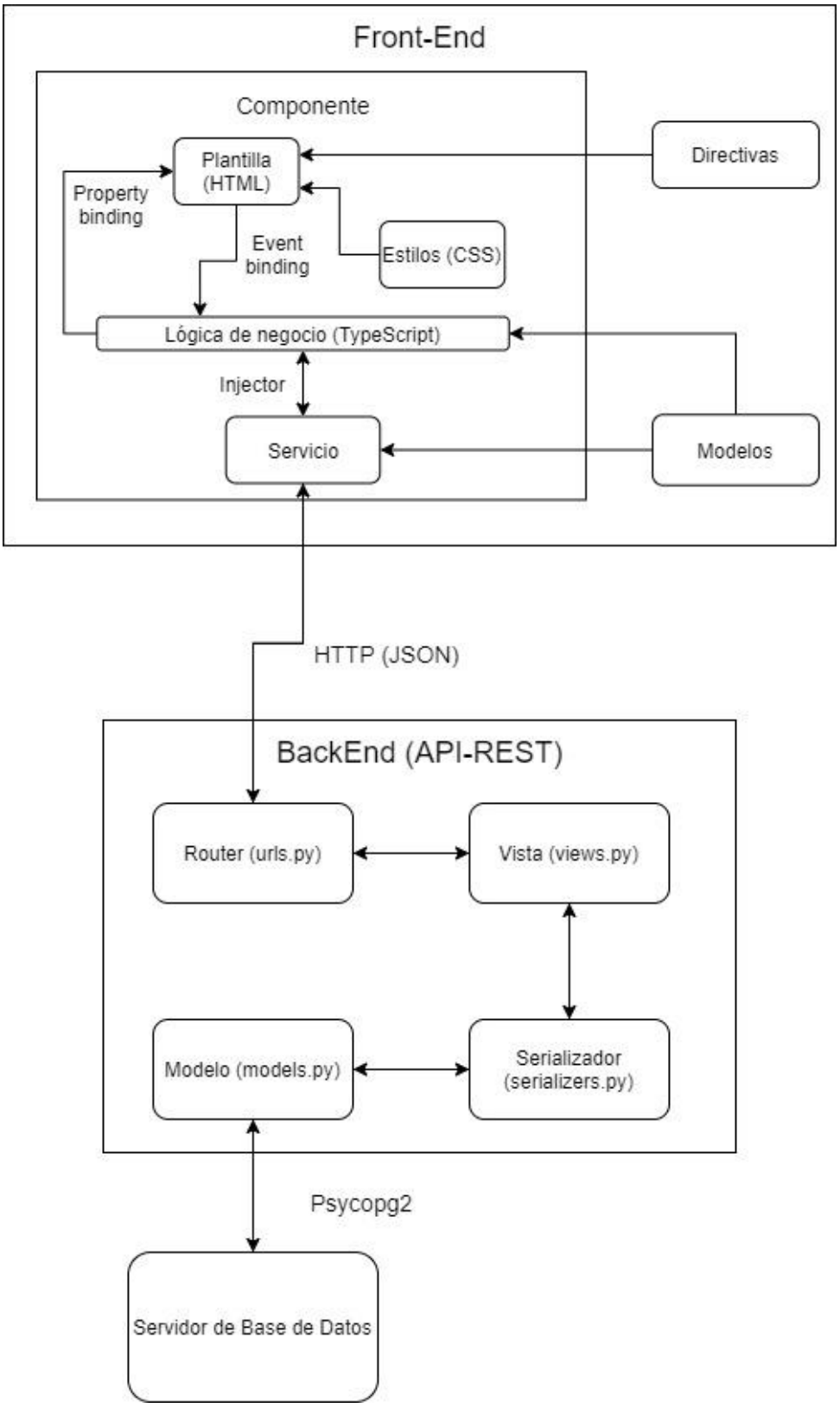

FIGURA 15: ARQUITECTURA LÓGICA

5.2. Arquitectura Física

La arquitectura física representa los componentes hardware requeridos para desplegar satisfactoriamente la plataforma.

En este caso hemos elegido emplear una arquitectura Cliente-Servidor, en la que el lado cliente solicitará los recursos necesarios para su funcionamiento a la API del lado servidor.

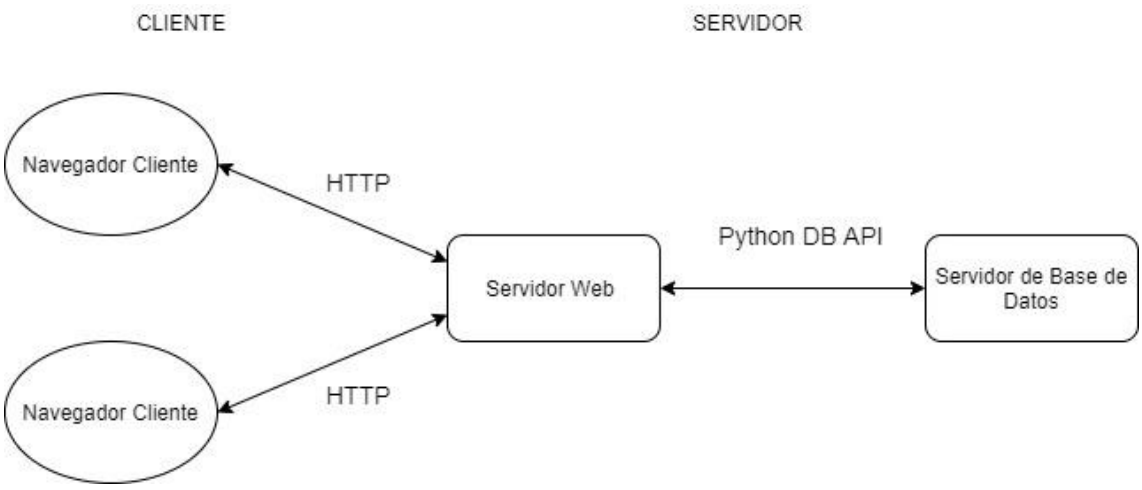

FIGURA 16: ARQUITECTURA FÍSICA

5.3. Diseño de la Base de Datos

En esta sección se realizará el diseño de la base de datos, el cual mostraremos a través del Modelo Relacional y el Diccionario de Datos.

5.3.1. Modelo relacional

Mediante el Modelo Relacional especificamos las distintas relaciones de las tablas de la Base de Datos, las cuales están basadas en los requisitos de información representados en el Modelo Entidad-Relación realizado en el apartado de Análisis. En nuestro caso, la base de datos se implementará a través del Gestor de Base de Datos PostgreSQL.

El Modelo se muestra a continuación.

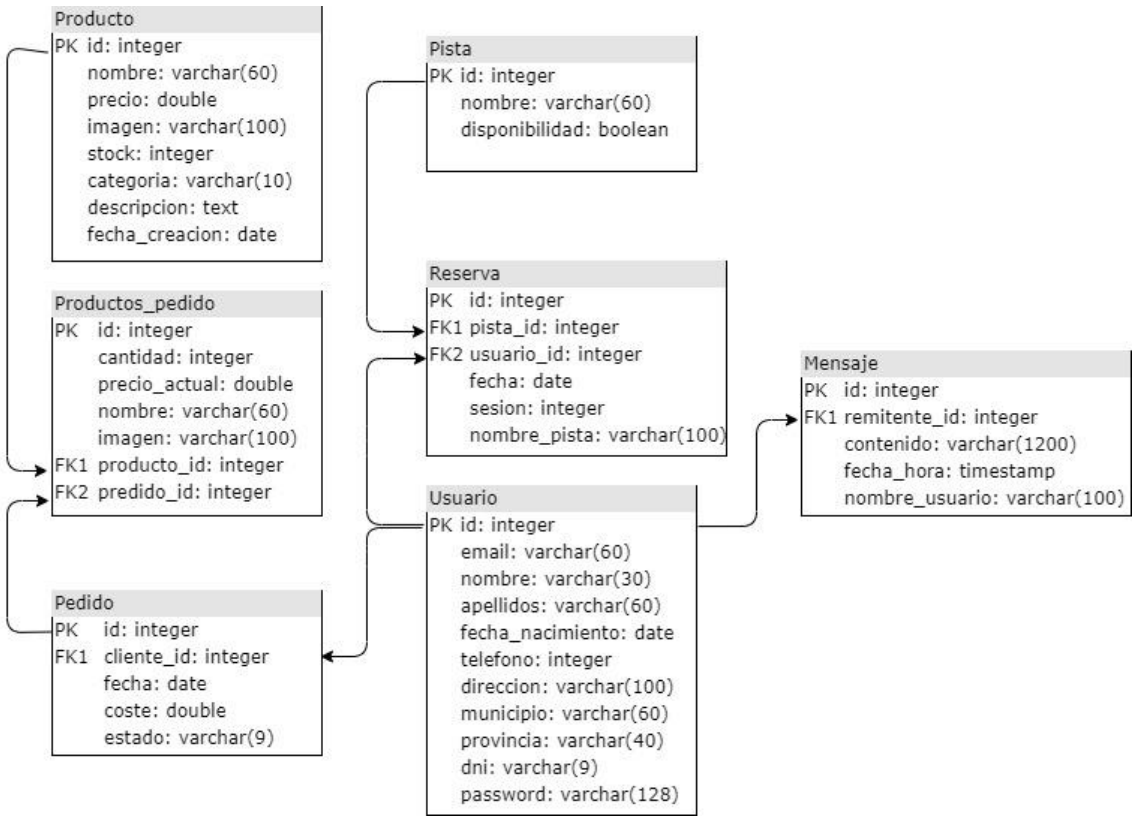

FIGURA 17: MODELO RELACIONAL

5.3.2. Diccionario de datos

En el Diccionario de Datos se detallan las restricciones derivadas del Modelo Relacional.

Respecto al nombre de las tablas, se debe hacer un apunte, y es que al manejar la BD a través de Django, éste es el encargado de generar los nombres de las tablas. Para hacer esto usa el nombre de la app en la que se ubica ese modelo y el propio nombre del modelo. Por ejemplo, el nombre de la tabla de usuarios será el nombre de la app en la que se encuentra (autenticación) y el propio nombre del modelo (usuario). El resultado será autenticación_usuario.

Tabla autenticación_usuario			
Definición	Esta tabla alberga la información del requisito RI-E01.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	PK
email	Varchar(60)	Unique Not Null	
nombre	Varchar(30)	Not Null	
apellidos	Varchar(60)	Not Null	
fecha_nacimiento	Date	Not Null	
telefono	Integer	Not Null	
direccion	Varchar(100)		
municipio	Varchar(60)		
provincia	Varchar(40)		
dni	Varchar(9)		
password	Varchar(128)		

Tabla chat_mensaje			
Definición	Esta tabla alberga la información del requisito RI-E07.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
remitente_id	Integer	Not Null	Foreign Key
contenido	Varchar(1200)	Not Null	
fecha_hora	Timestamp	Not Null	
nombre_usuario	Varchar(100)	Not Null	

Tabla pista_pista			
Definición	Esta tabla alberga la información del requisito RI-E04.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
nombre	Varchar(60)	Not Null Unique	
disponibilidad	Boolean	Not Null	

Tabla pista_reserva			
Definición	Esta tabla alberga la información del requisito RI-R01.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
pista_id	Integer	Not Null	Foreign Key
usuario_id	Integer	Not Null	Foreign Key
fecha	Date	Not Null	
Sesión	Integer	Not Null	

Tabla tienda_producto			
Definición	Esta tabla alberga la información del requisito RI-E05.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
nombre	Varchar(60)	Not Null Unique	
precio	Double	Not Null	
imagen	Varchar(100)	Not Null	
stock	Integer	Not Null	
categoria	Varchar(10)	Not Null	
descripción	Text	Not Null	
fecha_creacion	Timestamp	Not Null	

Tabla tienda_pedido			
Definición	Esta tabla alberga la información del requisito RI-E06.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
cliente_id	Integer	Not Null	Foreign Key
fecha	Date	Not Null	
coste	Double	Not Null	
estado	Varchar(9)	Not Null	

Tabla tienda_productos_pedido			
Definición	Esta tabla alberga la información del requisito RI-E01.		
Campo	**Tipo de dato**	**Restricción**	**Clave**
id	Integer	Unique Not Null Autoincremental	Primary Key
cantidad	Integer	Not Null	
precio_actual	Double	Not Null	
nombre	Varchar(60)	Not Null	
imagen	Varchar(100)	Not Null	
producto_id	Integer	Not Null	Foreign Key
pedido_id	Integer	Not Null	Foreign Key

5.4. Diagramas de secuencia

Un diagrama de secuencia es utilizado para representar las distintas interacciones entre los actores de un sistema. A continuación se mostrarán distintos diagramas relativos a Casos de Uso documentados en el apartado de Análisis. Debido a la alta similaridad entre los Casos de Uso tan sólo se expondrán unos pocos diagramas.

Diagrama de Secuencia CU-04: Identificar Usuario

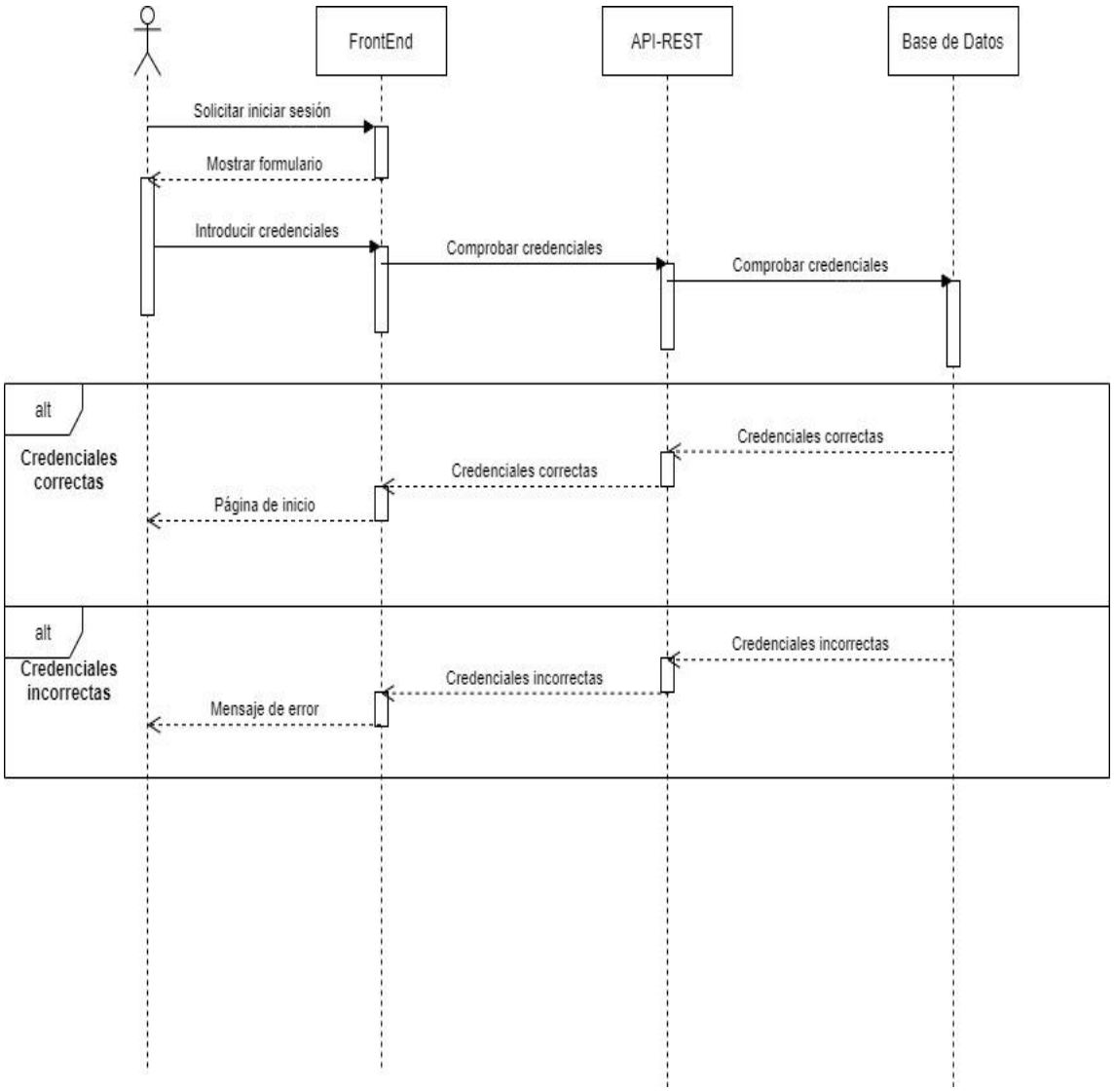

FIGURA 18: DIAGRAMA DE SECUENCIA CU-04

Diagrama de Secuencia CU-11: Visualizar reservas

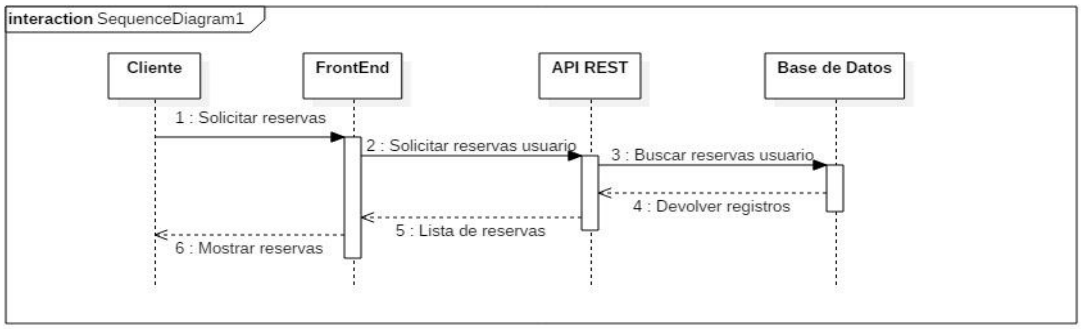

FIGURA 19: DIAGRAMA DE SECUENCIA CU-11

Diagrama de Secuencia CU-18: Reservar

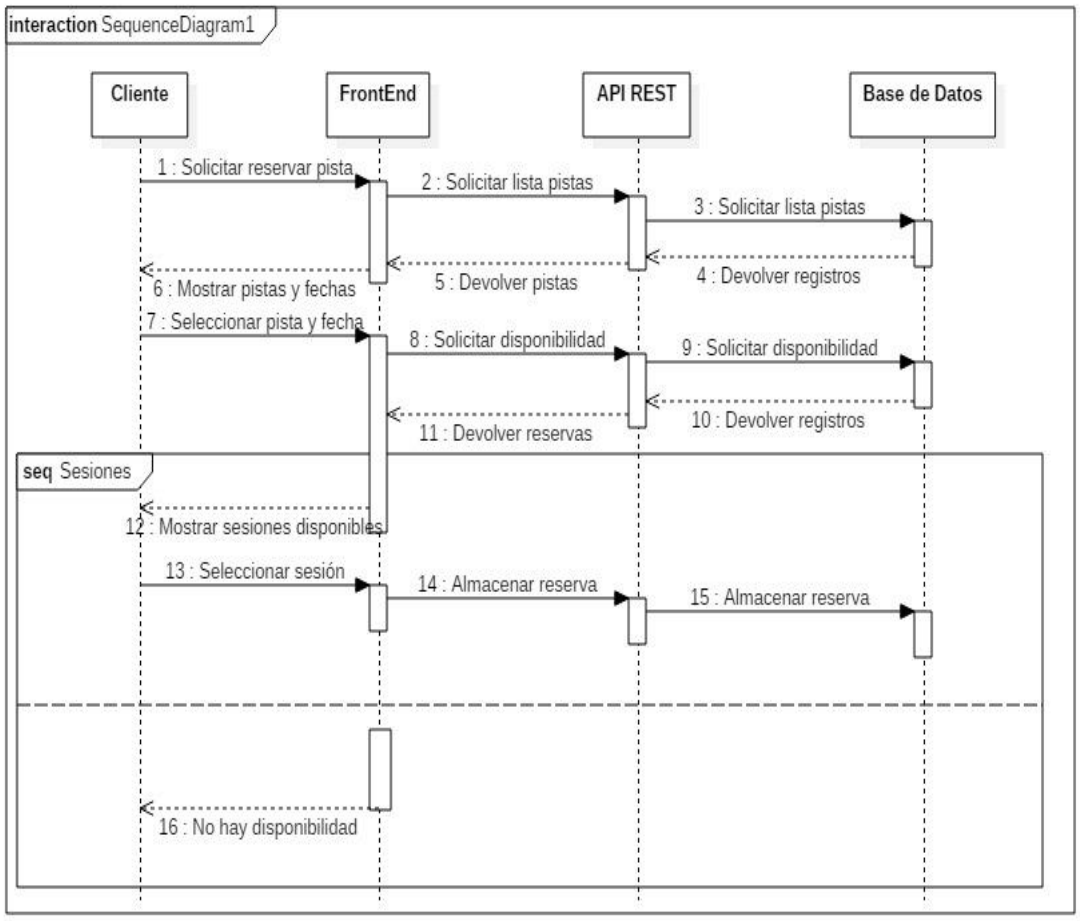

FIGURA 20: DIAGRAMA DE SECUENCIA CU-18

5.5. Diagramas de interfaz de usuario

A continuación se expondrán varios ejemplos de los bocetos de las distintas interfaces con las que interactuará el usuario mientras esté usando la aplicación. Estos bocetos nos permitirán dotar a las páginas de un estilo definido y común a todas ellas, y, además, el desarrollo de la aplicación deberá seguir estos bocetos.

Diagrama de Interfaz de Usuario 01: Pantalla de Registro

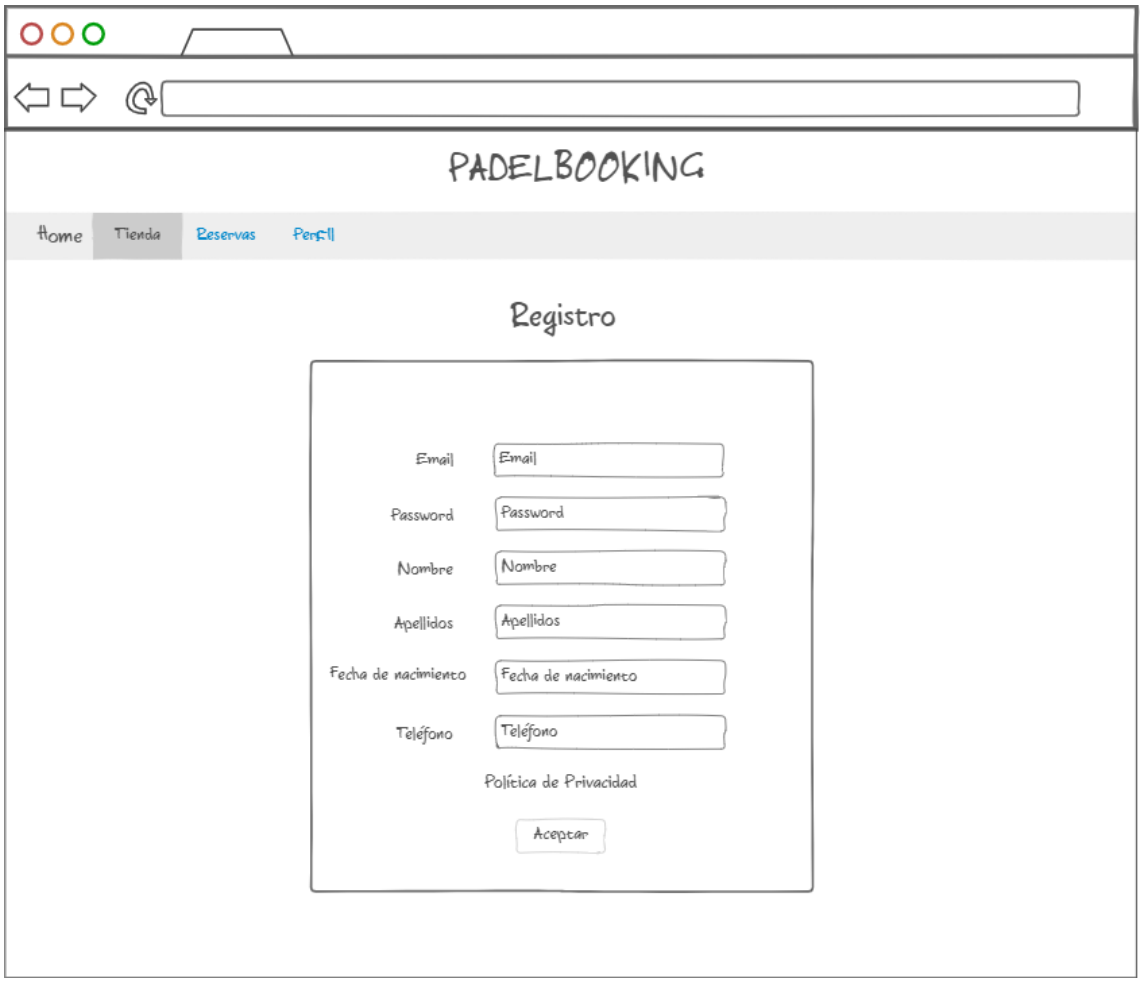

FIGURA 21: DIAGRAMA DE INTERFAZ DE USUARIO 01

Diagrama de Interfaz de Usuario 02: Pantalla principal de Tienda

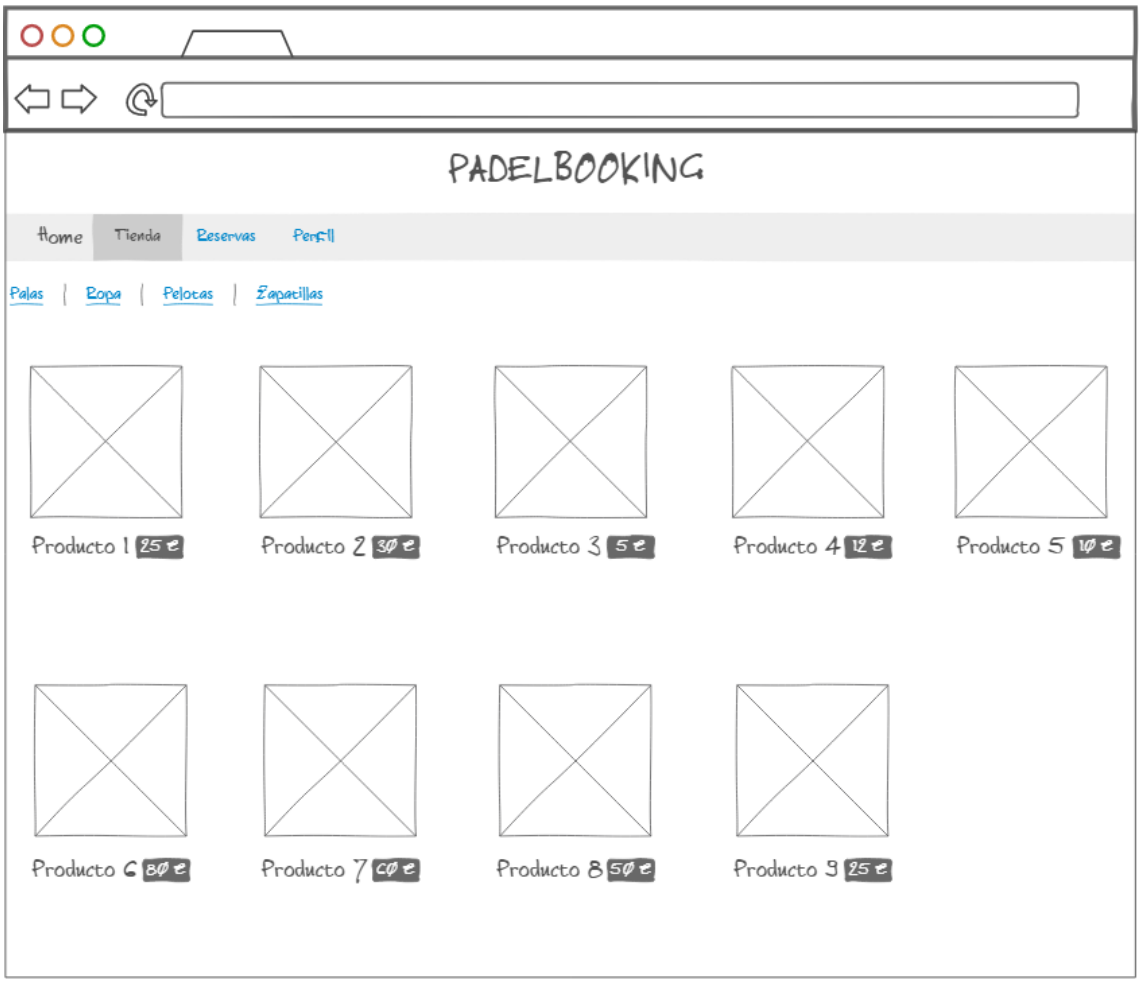

FIGURA 22: DIAGRAMA DE INTERFAZ DE USUARIO 02

Diagrama de Interfaz de Usuario 03: Pantalla completa de Reservas

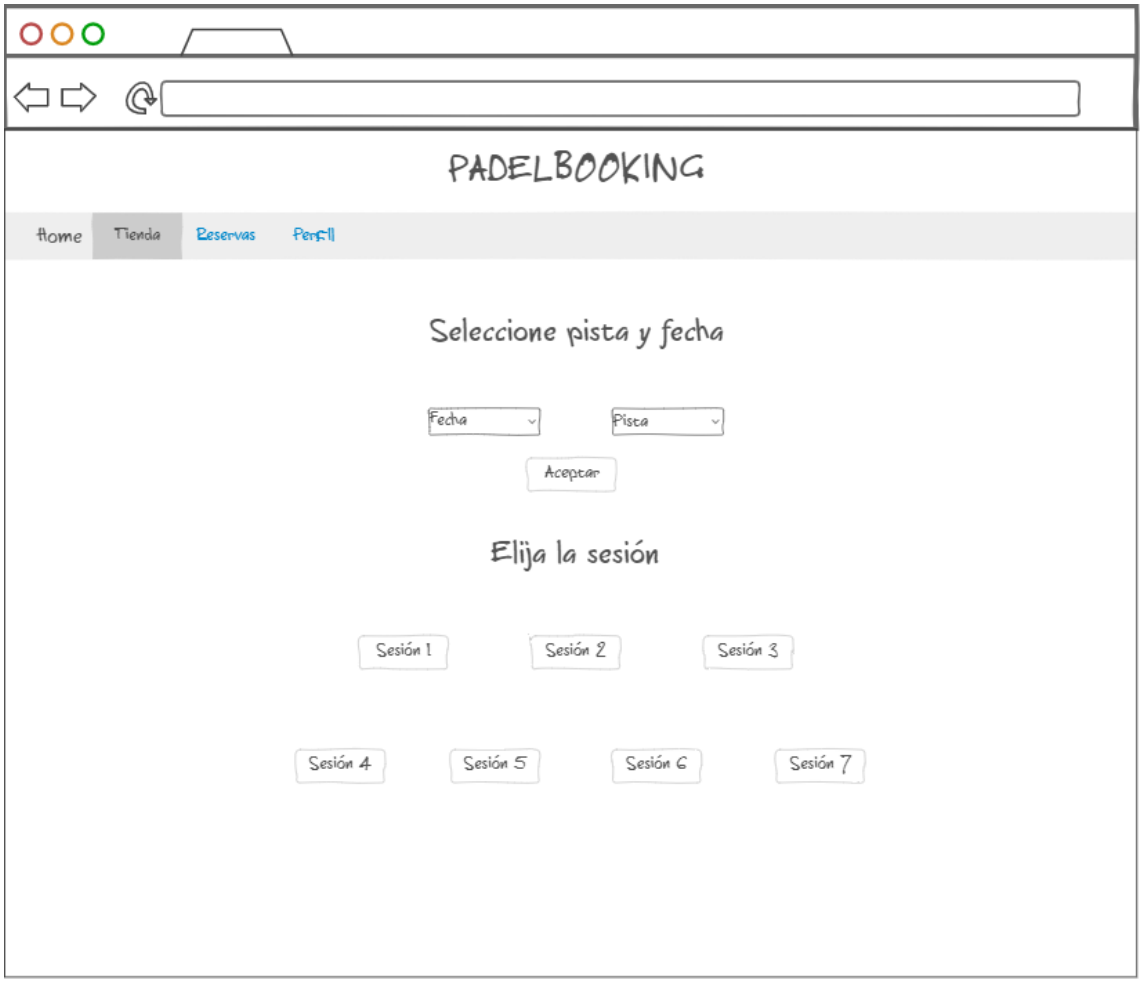

FIGURA 23: DIAGRAMA DE INTERFAZ DE USUARIO 03

6. IMPLEMENTACIÓN

En esta sección del documento detallaremos aspectos relacionados con la implementación del código de la aplicación, el cual ha sido desarrollado de los datos obtenidos de los apartados de análisis y diseño.

6.1. Herramientas y tecnologías utilizadas

En este apartado se proporcionará una breve explicación de cada una de las herramientas y tecnologías utilizadas en el desarrollo del proyecto.

6.1.1. Herramientas de desarrollo

- **PyCharm Community Edition**: PyCharm es un entorno de desarrollo de uso profesional para Python. Dispone de dos niveles de licencia, la versión Community, gratuita pero algo limitada, la cual es la que se ha empleado en el desarrollo de este proyecto, y la versión Professional, con más funcionalidades incluidas.
- **Visual Studio Code**: Editor de código fuente para desarrollo web en sistemas Windows. Incluye soporte para el software de control de versiones Git.
- **pgAdmin 4**: Herramienta especialmente diseñada para gestionar bases de datos relacionales PostgreSQL.
- **Postman**: Herramienta que permite realizar peticiones sobre una API de manera directa y sencilla, lo que permite comprobar el buen funcionamiento de la API antes de crear el front-end que la consuma.
- **PayPal**: Herramienta de pago online que soporta pagos entre usuarios. Más adelante en esta misma sección se profundizará en su uso y características.
- **Google Chrome**: Navegador que hemos utilizado para comprobar el buen funcionamiento de la aplicación.
- **Git**: Sistema de control de versiones creado por Linus Torvalds, que permite gestionar el código de una manera mucho más simple. Es el sistema de control de versiones más usado en el mundo ya que destaca en su eficiencia y confiabilidad a la hora del mantenimiento de versiones de aplicaciones. Emplea un Algoritmo Delta, el cual obtiene los bytes que han sido modificados desde la última versión del archivo.
- **GitHub**: Red social basada en el sistema de control de versiones de Git que permite gestionar proyectos software y controlar las versiones del código en la nube. Permite la creación de repositorios que, al enlazarse con los repositorios locales de Git, almacenan nuestro código en dicha nube. Estos repositorios pueden públicos o privados.

- **Heroku**: Heroku es una "Plataforma como Servicio" (PaaS: *Platform as a Service*) más empleadas en la actualidad. Permite desplegar aplicaciones y soporta una gran variedad de lenguajes de programación.

6.1.2. Herramientas de soporte

- **StarUML**: Aplicación que permite realizar diagramas UML. Tiene una versión de prueba, la cual se ha empleado en el desarrollo de este proyecto para los Diagramas de Casos de Uso.
- **FoxitReader**: Herramienta para la lectura de documentos pdf.
- **Diagrams**: Herramienta online de Google que permite realizar todo tipo de diagramas. Se ha empleado en el Diagrama de Entidad-Relación y en los Diagramas de Secuencia.
- **NinjaMock**: Aplicación para la creación de Diagramas de Interfaz de Usuario.
- **Lucid.app**: Herramienta online utilizada para crear los Diagramas de Gantt del proyecto. Tiene una versión gratuita, que es la que se ha empleado.

6.1.3. Tecnologías Utilizadas

- **Django**: Framework para la creación de aplicaciones web para el lenguaje Python, gratuito y de código abierto. Más adelante en esta misma sección profundizaremos más sobre este framework.
- **Django REST Framework** (**DRF**): Herramienta para el framework Django que permite la creación de API REST.
- **TypeScript**: Lenguaje de programación de código abierto basado en JavaScript que introduce una serie de cambios que hacen el código menos propenso a errores y más sencillo de mantener.
- **Angular 9**: Framework desarrollado por Google para facilitar la creación de aplicaciones web SPA (Single Page Application) que hace uso del lenguaje TypeScript.
- **PostgreSQL**: Gestor de Bases de Datos relacionales SQL de código abierto.

6.2. Django

Como ya hemos explicado, Django es un framework de alto nivel para desarrollo de aplicaciones web para el lenguaje Python. Las ventajas más importantes por las que nos hemos decantado por Django han sido las siguientes:

- Panel de Administración para la gestión de la Base de Datos, del cual hablaremos más adelante.
- Potente sistema de autenticación de usuarios.
- Su modularidad. Un proyecto de Django se divide en varias apps (módulos), las cuales pueden ser independientes unas de otras.
- Amplia comunidad de usuarios.
- Implementa medidas de seguridad por defecto, por ejemplo contra Inyección SQL o contra Cross Site Request Forgery
- ORM potente y accesible.
- Sencillez de programación.

Hay que recordar que, tal y como se expuso en el apartado de Arquitectura Lógica en la sección de Diseño, Django emplea una arquitectura MVT (Modelo-Vista-Template). Este tipo de arquitectura es una redefinición del modelo MVC (Modelo-Vista-Controlador) y sus componentes son los siguientes:

- **Modelo**: Se encarga de interactuar con la información de la aplicación, almacenada en una base de datos.
- **Vista**: La vista es la capa destinada a la lógica de negocio, recibe las peticiones HTTP y, tras recuperar los datos correspondientes del modelo, devuelve una respuesta HTTP.
- **Template**: Fichero de texto que renderizará la estructura HTML y representará los datos recogidos del modelo.

La estructura del patrón MVT se visualiza mejor en la siguiente ilustración:

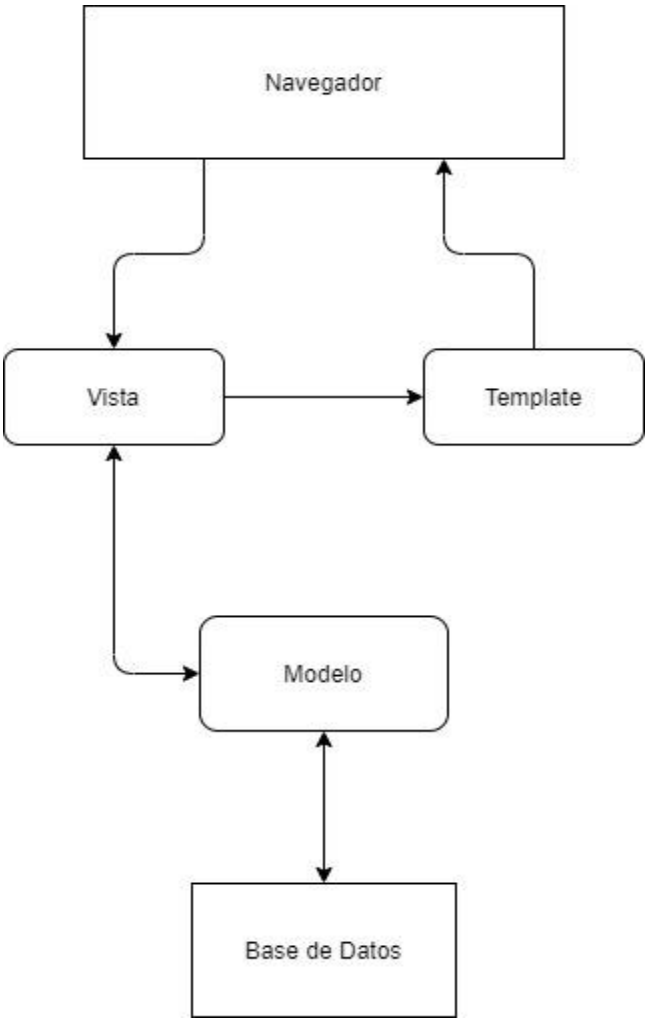

FIGURA 24: PATRÓN MODELO-VISTA-TEMPLATE

El modelo funciona de la siguiente forma:

1. Se recibe la petición del navegador y se pasa a la vista.
2. La vista encarga al modelo que corresponda la recuperación de los datos de la BD.
3. Dicho modelo interactúa con la BD para recuperar los datos.
4. El modelo devuelve los datos a la vista.
5. La vista inyecta los datos recuperados en la plantilla (Template).
6. La plantilla renderiza el resultado en el navegador.

Sin embargo, para la realización de este proyecto se ha usado una herramienta para Django llamada Django Rest Framework (DRF), la cual cambia aspectos de Django para permitir el desarrollo de API REST. El funcionamiento general de esta herramienta y como afecta a nuestro proyecto viene detallado en la sección de Diseño, en el apartado de Arquitectura Lógica.

Django Admin Panel

El Panel de Administración de Django es una de las funcionalidades más potentes de Django, que se inicia automáticamente al crear un proyecto. Dentro de esta área podremos usar los modelos definidos en nuestro proyecto para consultar, crear, eliminar y modificar registros, lo que puede ahorrar mucho tiempo de desarrollo. Además, se puede personalizar en cierta medida, eligiendo así que modelos queremos mostrar en el panel y como queremos que se vean.

Este panel también se usa en el entorno de producción, cuando la aplicación ya está operativa, pero se recomienda que únicamente tengan acceso a ella personal de la empresa u organización (administrador), y únicamente para el manejo de datos internos. El motivo principal para esto es que la interfaz que ofrece este panel expone datos que no deben ser visibles para el resto de usuarios.

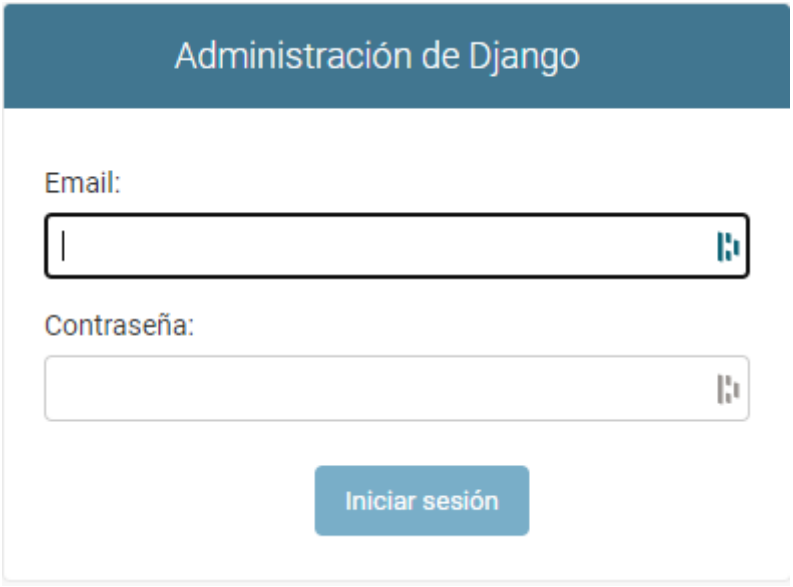

FIGURA 25: LOGIN PANEL DE ADMINISTRACIÓN

En lo que concierne a nuestra aplicación, hemos seguido la recomendación de que al Panel de Administración únicamente tenga acceso el administrador de la aplicación. En él, gestionará aspectos internas de la aplicación, como las pistas en las que se pueden realizar reservas o el catálogo de productos de la tienda, entre otras cosas.

La distribución del Panel de Administración está basada en las apps creadas en Django. Por ejemplo, en nuestro caso, tenemos los módulos de Autenticación, Chat, Pista y Tienda. Dentro de cada uno de estos módulos, están los distintos modelos que contienen.

FIGURA 26: PÁGINA PRINCIPAL PANEL DE ADMINISTRACIÓN

Para interactuar con los modelos es necesario entrar en uno de ellos, y aparecerá un listado con los registros de ese modelo. Tendremos las opciones de seleccionar un registro para visualizar toda su información y modificar sus campos, crear un nuevo registro o eliminar uno existente.

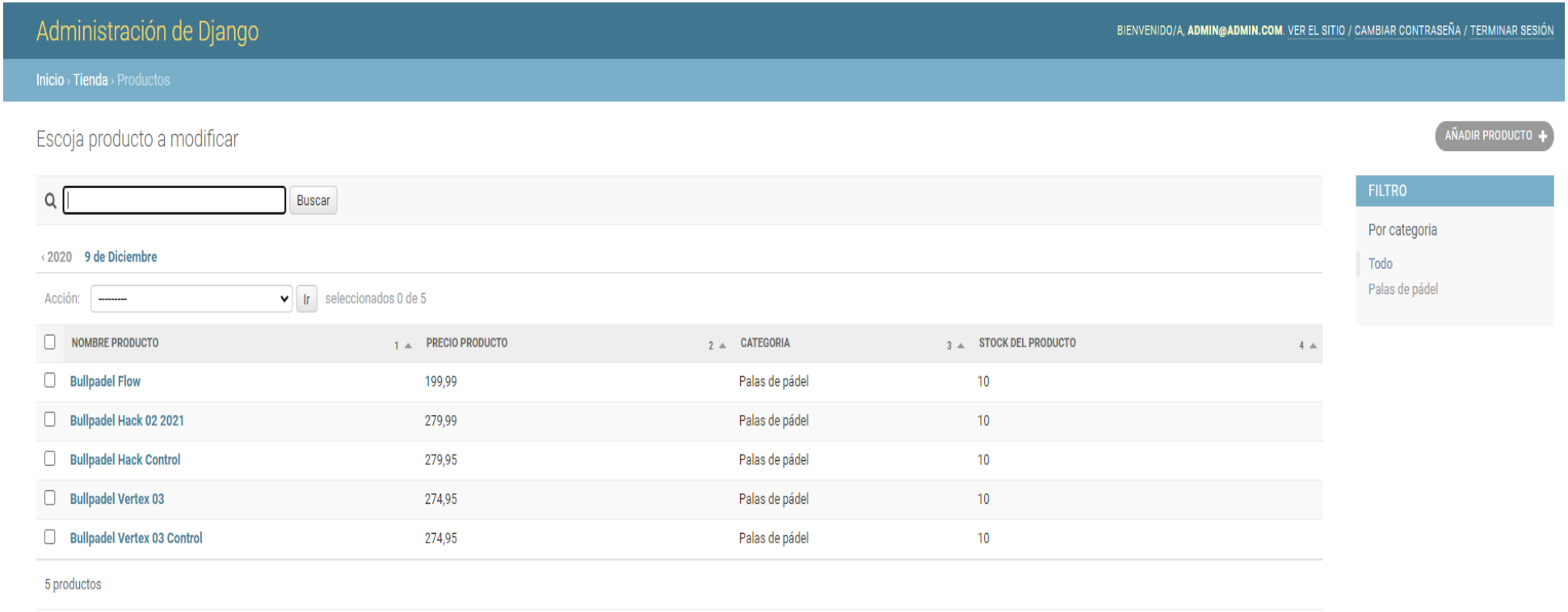

FIGURA 27: EJEMPLO MODELO PANEL DE ADMINISTRACIÓN

Al seleccionar la opción de crear un nuevo registro, se tendrán que rellenar todos los campos con la información del nuevo registro.

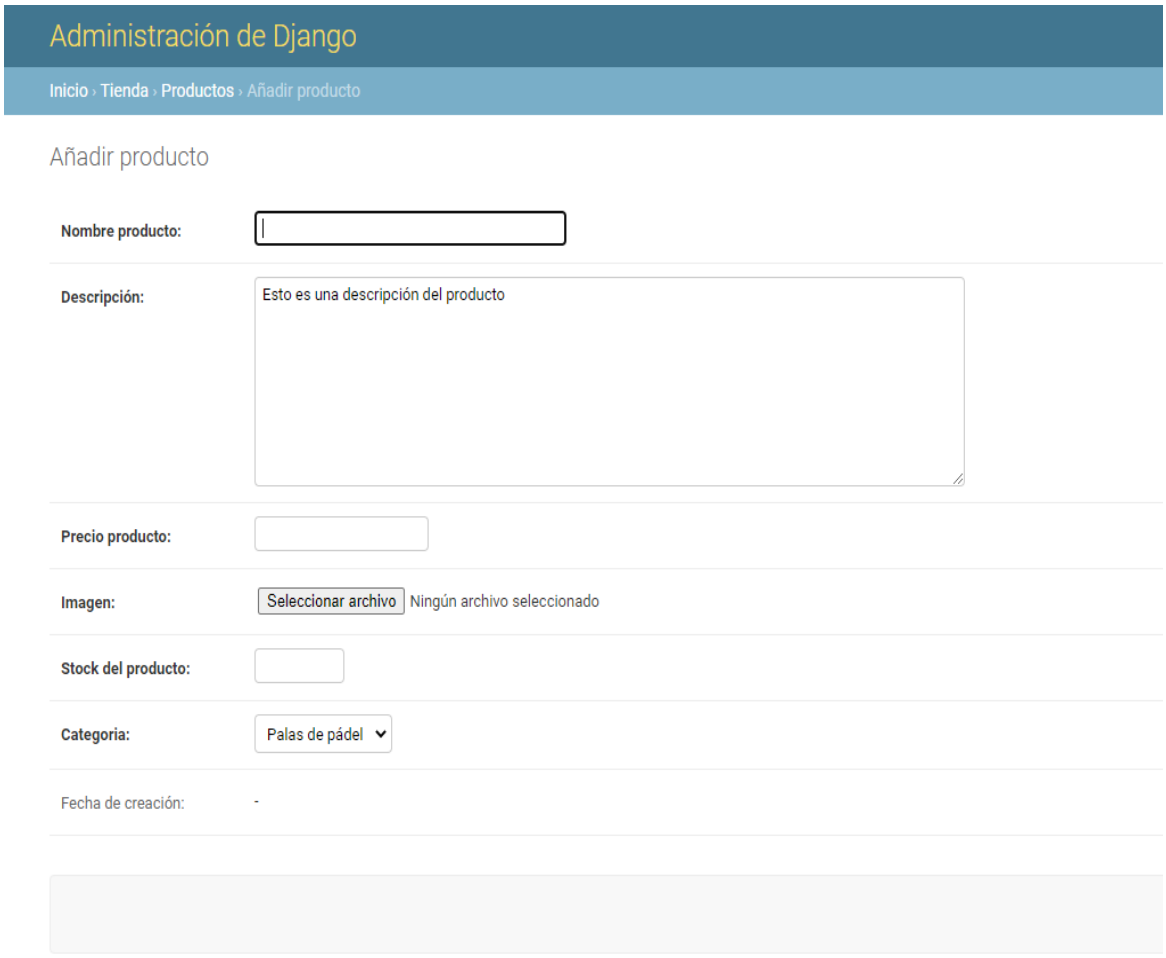

FIGURA 28: EJEMPLO REGISTRO PANEL DE ADMINISTRACIÓN

6.3. Implementación de la base de datos

En este apartado vamos a detallar algunos aspectos sobre la implementación de la base de datos.

6.3.1. Consideraciones de implementación

Por cuestiones de desarrollo se han tenido que incluir una serie de campos en diversas tablas que, pese a no estar plasmados en los requisitos de información del apartado de Análisis, son necesarios a la hora de implementar la aplicación.

- El campo nombre_usuario se ha incluido en la tabla Mensaje.
- Los campos nombre e imagen del producto se han incluido en la tabla Productos_pedido.
- El campo nombre_pista se ha incluido en la tabla Reserva.

Además, por cuestiones propias de Django, la relación Reservar se ha tenido que implementar como una relación N:M para que se pueda generar una tabla a partir de la propia relación. Django no permite que se generen tablas de relaciones 1:N.

6.3.2. Rendimiento de la base de datos

Para evaluar el rendimiento de la base de datos hemos empleado la herramienta pgbench, la cual está incluida en la instalación de PostgreSQL (la cual trataremos en la sección de Manuales). Esta herramienta permite evaluar el rendimiento de la base de datos simulando tráfico de usuarios realizando transacciones en unas tablas *mockup*, creadas especialmente para este *benchmarking*. Además, como no se espera un tráfico muy alto, al estar destinada la aplicación a un club de pádel de una ciudad pequeña, el pool de conexiones estará limitado a menos de cien conexiones simultáneas.

En primer lugar iniciaremos la herramienta con el siguiente comando, el cual creará las tablas mockup y creará quinientas mil tuplas, para simular una base de datos que se halle en uso.

```
C:\Program Files\PostgreSQL\12\bin>pgbench -U postgres -p 5432 -i -s 5  padelbooking
Password:
dropping old tables...
creating tables...
generating data...
100000 of 500000 tuples (20%) done (elapsed 0.02 s, remaining 0.08 s)
200000 of 500000 tuples (40%) done (elapsed 0.04 s, remaining 0.06 s)
300000 of 500000 tuples (60%) done (elapsed 0.14 s, remaining 0.09 s)
400000 of 500000 tuples (80%) done (elapsed 0.16 s, remaining 0.04 s)
500000 of 500000 tuples (100%) done (elapsed 0.52 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done.
```

FIGURA 29: INICIAR PGBENCH

Para realizar el *benchmarking*, pgbench seleccionará aleatoriamente de una lista predefinida una serie de scripts que incluyen transacciones en las que se realizan consultas de todo tipo.

El benchmarking lo llevaremos acabo de la siguiente manera:

```
C:\Program Files\PostgreSQL\12\bin>pgbench -U postgres -p 5432 -c 50 -j 4 -t 100 padelbooking
Password:
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 5
query mode: simple
number of clients: 50
number of threads: 4
number of transactions per client: 100
number of transactions actually processed: 5000/5000
latency average = 21.380 ms
tps = 2338.623627 (including connections establishing)
tps = 2402.309323 (excluding connections establishing)
```

FIGURA 30: BENCHMARK BASE DE DATOS

Aquí probamos el rendimiento de la base de datos con cincuenta conexiones simultáneas realizando cien transacciones cada uno. Como vemos, tanto la latencia media como las transacciones procesadas por segundo muestran unos tiempos aceptables. Sin embargo, estos tiempos se ven fuertemente limitados por el hardware en el que esté la base de datos (en este caso un ordenador de sobremesa). Si se deseara alcanzar mejores tiempos de acceso debería aumentarse la capacidad del hardware.

6.4. Estructura del proyecto

En este apartado detallaremos la estructura del proyecto, por una parte la API y por otra el front-end.

6.4.1. API

El árbol de directorios de la API es el siguiente:

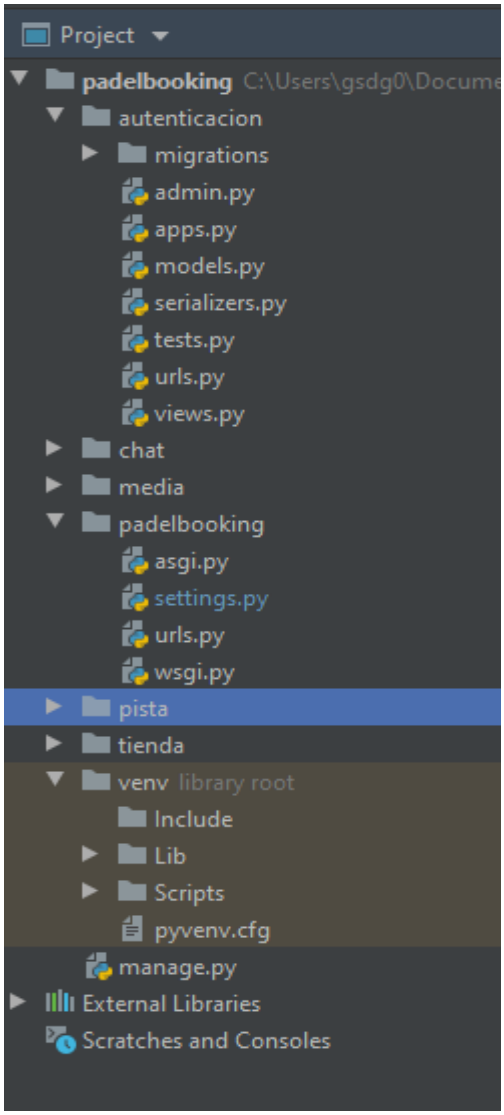

FIGURA 31: ESTRUCTURA API

Como ya hemos explicado en esta sección, la API está organizada por apps, las cuales son :

- **Autenticación**: app que se encarga de todo el sistema de autenticación y registro de los usuarios.
- **Chat**: app que se encarga del chat entre usuarios.
- **Pista**: app que se encarga del sistema de pistas y las reservas de los usuarios.
- **Tienda**: app que se encarga de la tienda de la aplicación.

Por convenio, las imágenes de la aplicación, en nuestro caso las imágenes de los productos que el administrador registra en el Panel de Administración se almacenan y se sirven desde una carpeta en el directorio /media.

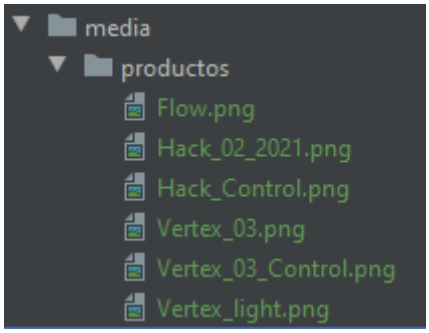

FIGURA 32: CARPETA DE IMÁGENES DE LA APLICACIÓN

Cada app cuenta con unos ficheros y directorios predeterminados, es decir, cada una de las cuatro apps desarrolladas tendrán un despliegue análogo. Dichos ficheros y una muestra de su contenido se muestran a continuación:

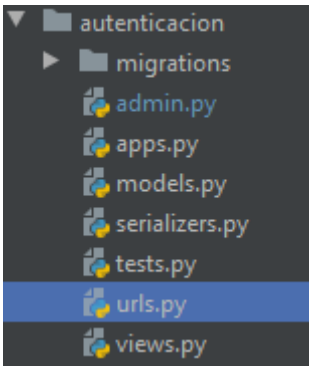

FIGURA 33: EJEMPLO APP DJANGO

- `/migrations`: Las migraciones en Django son la manera que tiene Django de propagar los cambios que se hacen a los modelos hasta la Base de Datos. Además actúan como un sistema de control de versiones para la Base de Datos, ya que en el directorio /migrations de cada app se almacenan todas las migraciones que se han ejecutado en esa app. Gracias a esto se puede volver atrás en los esquemas de los modelos.

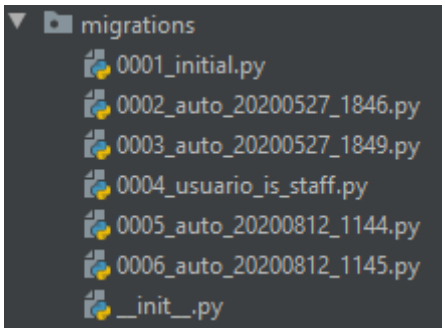

FIGURA 34: MIGRACIONES APP

- `admin.py`: Fichero en el que se registran los modelos que se van a visualizar en el Panel de Administración.

```python
class PistaAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'disponibilidad')
    ordering = ('nombre', 'disponibilidad')
    search_fields = ('nombre', 'disponibilidad')
    list_filter = ('disponibilidad',)


class ReservaAdmin(admin.ModelAdmin):
    readonly_fields = ('fecha', 'sesion')
    list_display = ('usuario', 'pista', 'fecha', 'sesion')
    ordering = ('pista', 'fecha', 'usuario')
    search_fields = ('pista__id', 'fecha', 'usuario__email')
    date_hierarchy = 'fecha'

    def has_add_permission(self, request, obj=None):
        return False


admin.site.register(Pista, PistaAdmin)
admin.site.register(Reserva, ReservaAdmin)
```

FIGURA 35: EJEMPLO ADMIN.PY

- `apps.py`: Fichero que almacena información extra de la app.

```python
from django.apps import AppConfig


class PistaConfig(AppConfig):
    name = 'pista'
```

FIGURA 36: EJEMPLO APPS.PY

- `models.py`: Archivo que contiene los modelos registrados en la app.

```python
class Pista(models.Model):
    nombre = models.CharField(verbose_name='nombre pista', max_length=60, unique=True)
    disponibilidad = models.BooleanField(verbose_name='disponibilidad de la pista', default=True)
    reservas = models.ManyToManyField(Usuario, through='Reserva')

    def __str__(self):
        return self.nombre


class Reserva(models.Model):
    usuario = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    pista = models.ForeignKey(Pista, on_delete=models.CASCADE)
    fecha = models.DateField(verbose_name="Fecha de reserva", default=now)
    sesion = models.IntegerField(verbose_name="Sesión", default=0)

    nombre_pista = models.CharField(verbose_name='Nombre de pista', max_length=100, default='null')

    class Meta:
        index_together = [['fecha'], ['pista']]

    def __str__(self):
        return 'Reserva de ' + str(self.usuario) + ' en ' + str(self.pista) + ' el ' + str(self.fecha)
```

FIGURA 37: EJEMPLO MODELS.PY

- `serializers.py`: Archivo que contiene los distintos serializadores de la app, cada uno de ellos conectado a un modelo.

```python
class PistaSerializer(serializers.ModelSerializer):
    class Meta:
        model = Pista
        fields = ['id', 'nombre', 'disponibilidad']


class ReservaSerializer(serializers.ModelSerializer):
    class Meta:
        model = Reserva
        fields = ['id', 'usuario', 'pista', 'fecha', 'sesion', 'nombre_pista']
```

FIGURA 38: EJEMPLO SERIALIZERS.PY

- `urls.py`: Fichero en el que se definen las direcciones de la app, cada una de ellas enlazada a una vista.

```python
router = routers.DefaultRouter()
router.register('pistas', PistaViewSet)
router.register('reservas', ReservaViewSet)
router.register('reservas_usuario', ReservasUsuarioViewSet)
router.register('reservar', ReservasDiaPistaViewSet)

urlpatterns = [
    path('', include(router.urls)),

]
```

FIGURA 39: EJEMPLO URLS.PY

- `views.py`: Fichero en el que se definen las vistas de la app.

```python
class PistaViewSet(viewsets.ModelViewSet):
    serializer_class = PistaSerializer
    queryset = Pista.objects.filter(disponibilidad=True)
    authentication_classes = (TokenAuthentication,)
    permission_classes = (AllowAny, )


class ReservaViewSet(viewsets.ModelViewSet):
    serializer_class = ReservaSerializer
    queryset = Reserva.objects.all()
    authentication_classes = (TokenAuthentication, )
    permission_classes = (IsAuthenticated, )


class ReservasUsuarioViewSet(viewsets.ModelViewSet):

    serializer_class = ReservaSerializer
    queryset = Reserva.objects.all()
    authentication_classes = (TokenAuthentication,)
    permission_classes = (IsAuthenticated,)

    def get_queryset(self):
        usuario = self.request.user
        return Reserva.objects.filter(usuario=usuario.id)
```

FIGURA 40: EJEMPLO VIEWS.PY

Existe una App core del proyecto, cuyo nombre es el mismo nombre del proyecto, y que contiene información general necesaria para el funcionamiento de la aplicación.

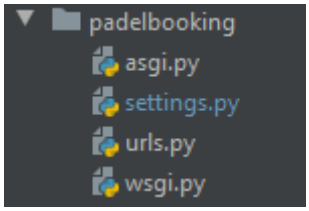

FIGURA 41: APP PRINCIPAL

- `asgi.py`: Fichero de configuración necesario si se quiere desplegar el proyecto en un servidor ASGI (*Asynchronous Server Gateway Interface*).

```python
import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'padelbooking.settings')

application = get_asgi_application()
```

FIGURA 42: EJEMPLO ASGI.PY

- `settings.py`: Archivo que contiene los parámetros generales del proyecto, tales como las aplicaciones instaladas, el middleware utilizado o la información de la conexión a la Base de Datos.

```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'autenticacion',
    'pista',
    'chat',
    'tienda',
    'rest_framework',
    'rest_framework.authtoken',
    'corsheaders',
    'django_filters',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'corsheaders.middleware.CorsMiddleware'
]
```

FIGURA 43: EJEMPLO SETTINGS.PY

- `urls.py`: Fichero en el que se definen las direcciones generales del proyecto, las cuales estarán enlazadas a las distintas apps.

```python
urlpatterns = [
    path('admin/', admin.site.urls),
    path('auth/', include('autenticacion.urls')),
    path('pista/', include('pista.urls')),
    path('chat/', include('chat.urls')),
    path('tienda/', include('tienda.urls'))
]

if settings.DEBUG:
    from django.conf.urls.static import static
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

FIGURA 44: EJEMPLO URLS.PY PRINCIPAL

- `wsgi.py` (*Web Server Gateway Interface*): Fichero de configuración necesario para desplegar el proyecto en un servidor WSGI.

```python
import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'padelbooking.settings')

application = get_wsgi_application()
```

FIGURA 45: EJEMPLO WSGI.PY

Por último tenemos el directorio del entorno virtual sobre el que está operando el proyecto. Un entorno virtual de Python encapsula la configuración, los paquetes, etc, de Python y de Django para un único proyecto para que no afecte al resto de proyectos del sistema.

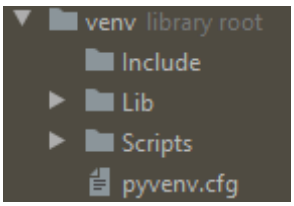

FIGURA 46: ENTORNO VIRTUAL

6.4.2. Front-end

El árbol de directorios del front-end es el siguiente:

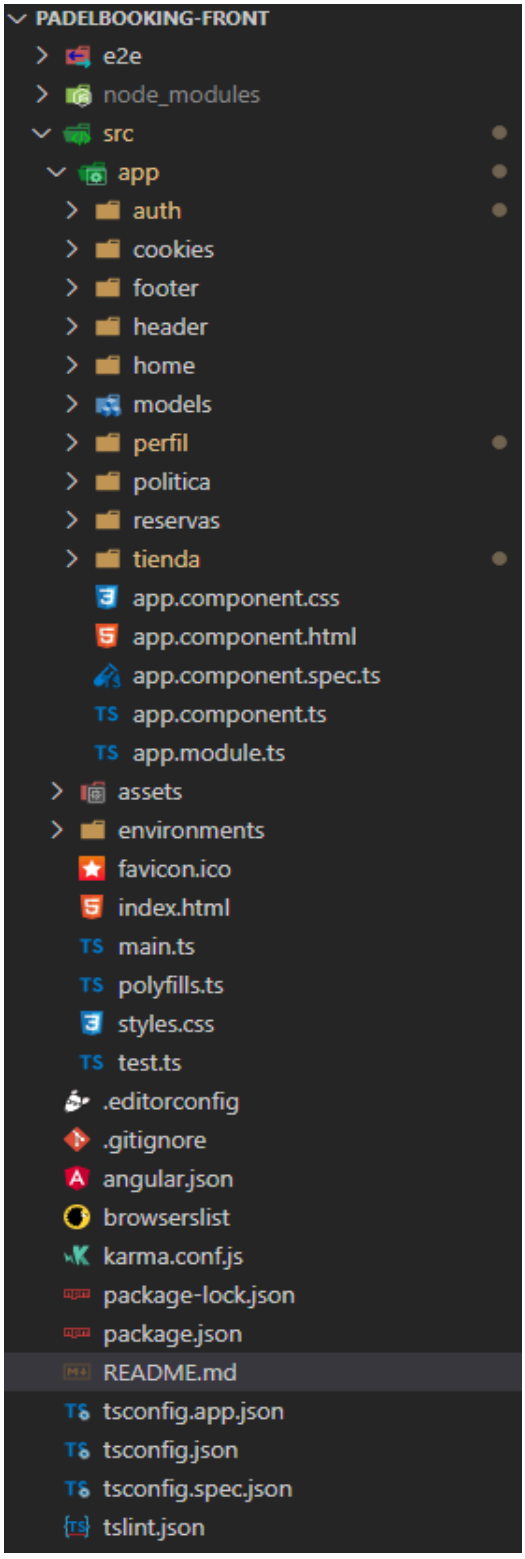

FIGURA 47: ESTRUCTURA DE DIRECTORIOS FRONT-END

Este árbol de directorios ha sido organizado por módulos, los cuales contienen a su vez los componentes que conforman dicho módulo. Cada módulo cuenta con un fichero TypeScript en el que se lleva a cabo el Routing del módulo (las distintas páginas del módulo, cada una de ellas asociada a un componente) importaciones necesarias, etc.

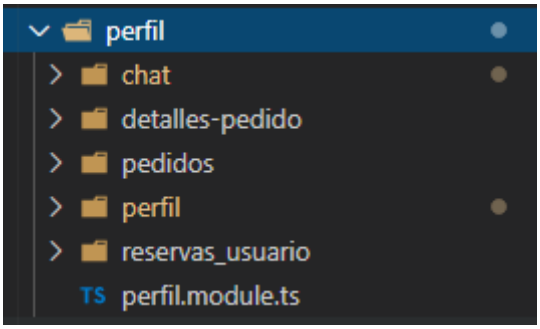

FIGURA 48: EJEMPLO MÓDULO ANGULAR

Como ya hemos dicho, cada componente es una página de la aplicación, y está formado por los siguientes ficheros y directorios:

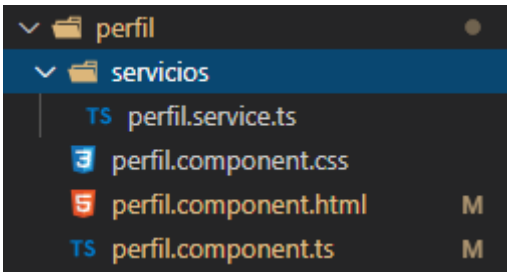

FIGURA 49: EJEMPLO COMPONENTE ANGULAR

- /servicios: Carpeta en el que está el servicio que usa el componente. El servicio es un fichero TypeScript que se encarga de traer los datos necesarios desde la API y posteriormente ser inyectado en el componente para usar dichos datos.

```
constructor(private httpClient: HttpClient, private cookieService: CookieService) { }

getProducto(id: number){
  let params = new HttpParams().set("id", id.toString());
  return this.httpClient.get<Producto[]>(`${this.baseUrl}`, {headers: this.headers, params: params});
}

getUser(): Observable<Usuario>{
  return this.httpClient.get<Usuario>(this.userUrl, {headers: this.getAuthHeaders()});
}

getAuthHeaders(){
  const token = this.cookieService.get('login-token');

  return new HttpHeaders({
    'Content-Type': 'application/json',
    Authorization : `Token ${token}`
  });
}
```

FIGURA 50: EJEMPLO SERVICIO

- `component.ts`: Fichero TypeScript que se encarga de la lógica de negocio de la aplicación, manipulando los datos recogidos de la API, pasar nuevos datos a la API, etc.

```
export class DetallesComponent implements OnInit {

  producto: Producto;
  usuario: Usuario;
  pedido: Pedido;
  loginToken;
  cantidad = new FormControl('');

  constructor(private router: Router, private snackBar: MatSnackBar, private _route: ActivatedRoute,
              private detallesService: DetallesService, private carritoService: CarritoService,
              private cookieService: CookieService) { }

  ngOnInit(): void {

    this.loginToken = this.cookieService.get('login-token');
    let id = +this._route.snapshot.paramMap.get('id');
    this.devolverUsuario();

    this.detallesService.getProducto(id).subscribe(
      (data: Producto[]) => {
        this.producto = data[0];
      },
      error => {
        console.log(error);
      }
    );
  }
```

FIGURA 51: EJEMPLO COMPONENTE TYPESCRIPT

- `component.html`: Fichero html que hará la función de plantilla.

```
<body>
    <nav>
        <div *ngIf="loginToken"><p>Hola {{usuario?.nombre}}</p></div>
        <button id="salir" *ngIf="loginToken" (click)="logout()" class="material-icons">exit_to_app</button>
    </nav>
    <div id="principal">
        <div id="producto" *ngIf="producto">
            <h2>{{producto.nombre}}</h2>
            <div id="imagen">
                <img src="{{producto.imagen}}" alt="foto">
            </div>

            <div class="detalles">
                <div class="precio-cantidad">
                    <h3>Precio: {{producto.precio}}€</h3>
                    <div class="cantidad">
                        <h3>Cantidad:</h3>
                        <input type="number" [formControl]="cantidad">
                    </div>
                </div>
                <div>
                    <button (click)="addProducto()">Añadir al carrito</button>
                </div>

            </div>
            <div id="descripcion">
                <p>{{producto.descripcion}}</p>
            </div>
        </div>
    </div>
</body>
```

FIGURA 52: EJEMPLO COMPONENTE HTML

- `component.css`: Hoja de estilos del componente, que transforma el aspecto del fichero html para renderizarlo en el navegador.

```css
body {
    margin-top: 140px;
    height: 85%;
    overflow-y: scroll;
}

body::-webkit-scrollbar {
    display: none;
}

nav {
    display: grid;
    grid-template-columns: auto auto;
    justify-content: flex-end;
    grid-gap: 10px;
}

nav div {
    display: flex;
    align-content: center;
}
```

FIGURA 53: EJEMPLO COMPONENTE CSS

Además de los componentes, en el árbol de directorios también tenemos el directorio /models, el cual contiene todos los modelos que vamos a usar en los componentes.

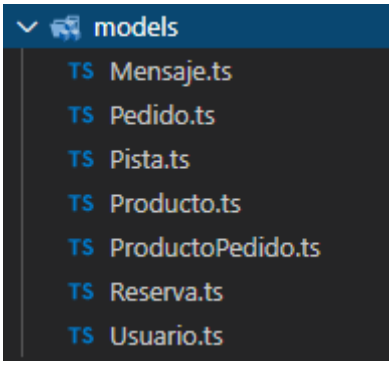

FIGURA 54: DIRECTORIO MODELOS ANGULAR

Otros ficheros y directorios importantes del proyecto son los siguientes:

- `package.json`: Fichero de configuración del gestor de paquetes npm, define los scripts del proyecto y todas las dependencias.
- `angular.json`: Fichero de configuración del proyecto que define su estructura.
- `/assets`: Directorio que almacena recursos usados en el proyecto, principalmente imágenes.
- `index.html`: Fichero html de entrada al proyecto.
- `styles.css`: Hoja de estilos global del proyecto.

6.5. PayPal

Al contar el proyecto con una sección de tienda online, hay que definir un método de pago. En este caso nos hemos decidido por PayPal, una conocida plataforma externa que permite realizar y recibir pagos online de forma rápida y segura.

Para simular los pagos en la aplicación se ha recurrido a cuentas "mockup", mediante las cuales se pueden realizar pagos de manera ficticia, sin dinero real.

La integración de la plataforma en la aplicación es la siguiente:

1. Se envía una petición a PayPal con los parámetros del pago (cantidad, divisa y cuenta PayPal del vendedor).
2. Si todo es correcto, aparecerá una ventana emergente en la aplicación, solicitando al cliente que introduzca las credenciales de su cuenta de PayPal.
3. Una vez el cliente ha introducido sus datos, en pantalla saldrán los datos del pago. El cliente podrá revisarlos y ejecutar el pago.
4. Al completar el pago la aplicación volverá a tomar el control, completando así la compra.

6.6. Implementación

En esta sección de la implementación se procederá a describir la implementación y la instalación de la infraestructura necesaria para crear de dos proyectos básicos realizados en Django y en Angular, para posteriormente enlazar con el despliegue de nuestra aplicación.

6.6.1. Proyecto Django

Para esta explicación supondremos que Python y un IDE (*Integrated Development Environment*) adecuado para trabajar en Python, en nuestro caso PyCharm.

Lo primero que tenemos que hacer es instalar un entorno virtual para nuestro proyecto, el cual encapsulará nuestra aplicación. Para hacer esto abriremos la consola de nuestro equipo y navegaremos hasta el directorio en el cual queremos alojar nuestra aplicación.

Una vez allí, crearemos nuestro entorno virtual de la siguiente manera y lo activaremos de la siguiente manera:

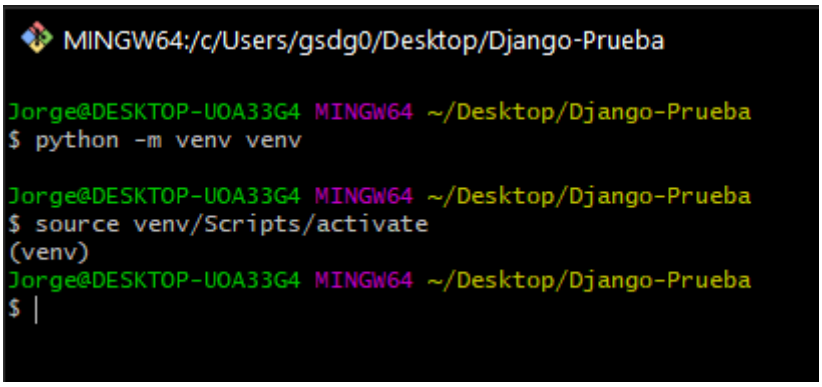

FIGURA 55: CREACIÓN ENTORNO VIRTUAL

Posteriormente instalaremos Django en el entorno virtual:

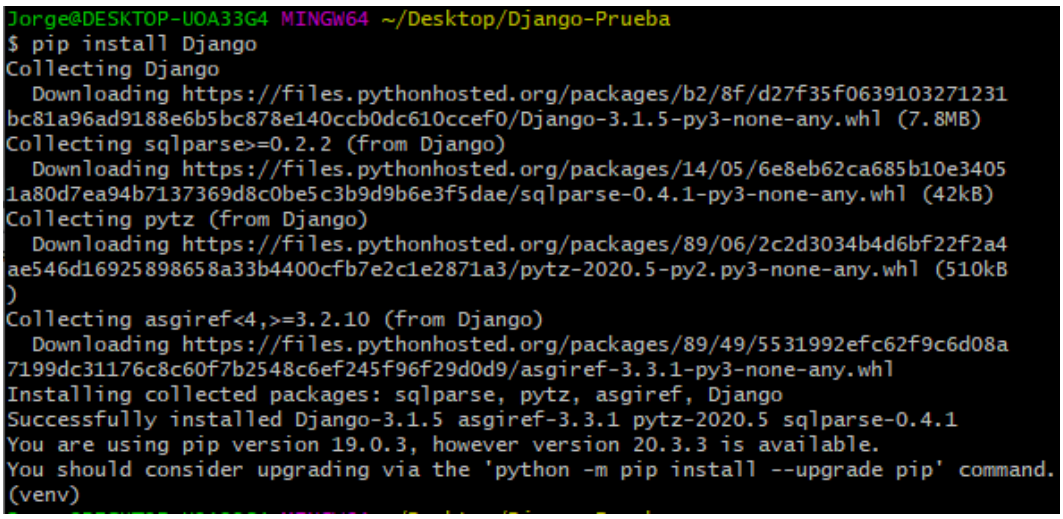

FIGURA 56: INSTALACIÓN DJANGO

Ahora que hemos instalado Django en el entorno, abriremos el directorio en el IDE y creamos el proyecto:

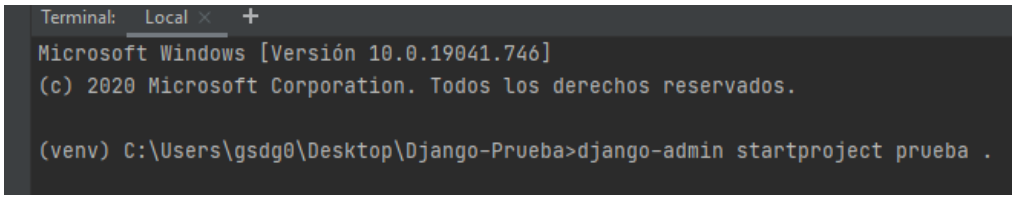

FIGURA 57: CREACIÓN PROYECTO DJANGO

Como resultado de esto, se creará el proyecto Django en el directorio actual:

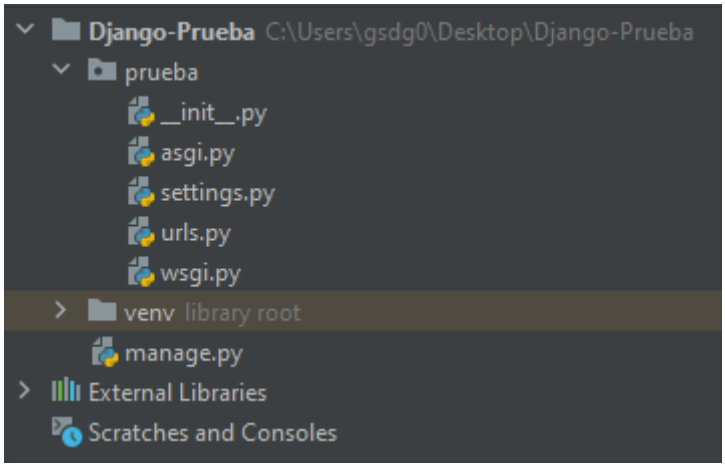

FIGURA 58: PROYECTO DJANGO CREADO

Por último, y para comprobar que todo ha salido bien, usaremos el siguiente comando para correr la aplicación en el servidor local:

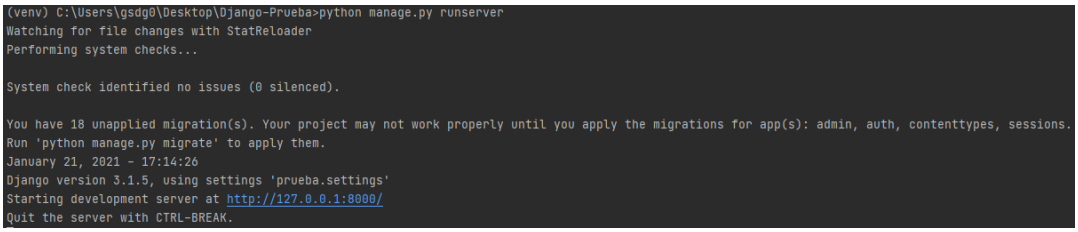

FIGURA 59: PROYECTO DJANGO CORRIENDO

El resultado es el siguiente:

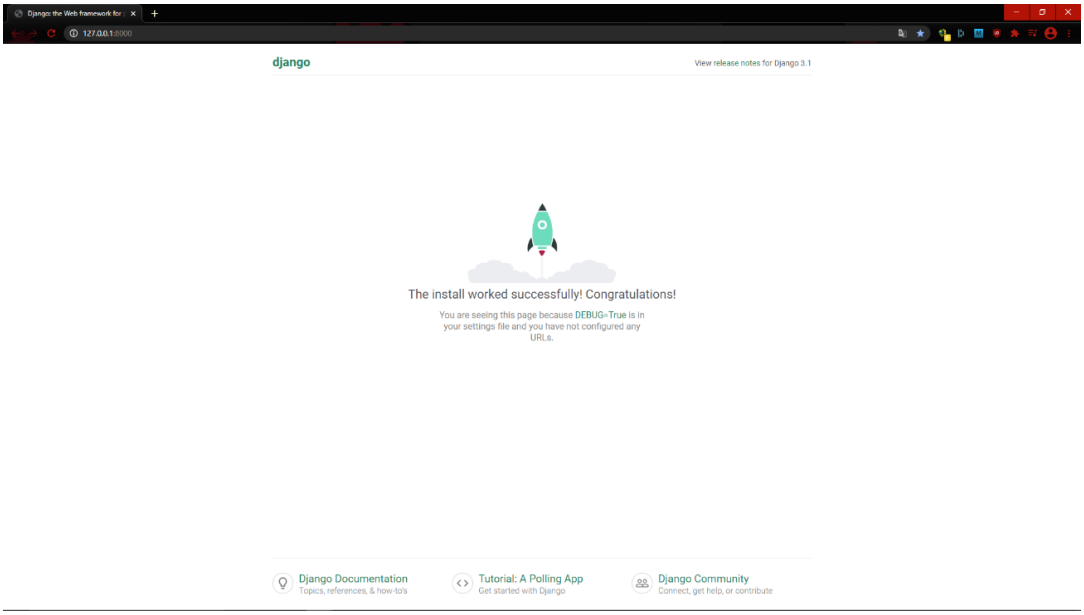

FIGURA 60: PROYECTO DJANGO PÁGINA PRINCIPAL

Nuestra app Django ya está lista para empezar con el desarrollo.

6.6.2. Proyecto Angular

Para esta aplicación supondremos que un IDE o editor de texto adecuado para desarrollo web está instalado en el equipo, en nuestro caso usaremos el editor Visual Studio Code.

Para crear una aplicación básica Angular lo primero que tenemos que hacer es instalar Node.js en nuestro equipo, y junto a él, el gestor de paquetes NPM. La principal razón por la que necesitamos Node.js para crear un entorno de desarrollo Angular es que este framework usa TypeScript en lugar de JavaScript, y los navegadores no soportan TypeScript por defecto. Mediante Node.js y NPM compilaremos los archivos TypeScript a JavaScript para que los navegadores los soporten.

Para instalar Node.js iremos a su dirección web (https://nodejs.org/) y descargaremos la versión recomendada para nuestro equipo. Posteriormente la instalaremos.

Una vez Node.js y el gestor de paquetes NPM estén instalados en nuestro equipo, el siguiente paso es abrir nuestro editor de código y, con él, abrir el directorio en el que queremos que esté alojada nuestra aplicación.

Una vez abierto el directorio, iniciaremos la terminal del programa:

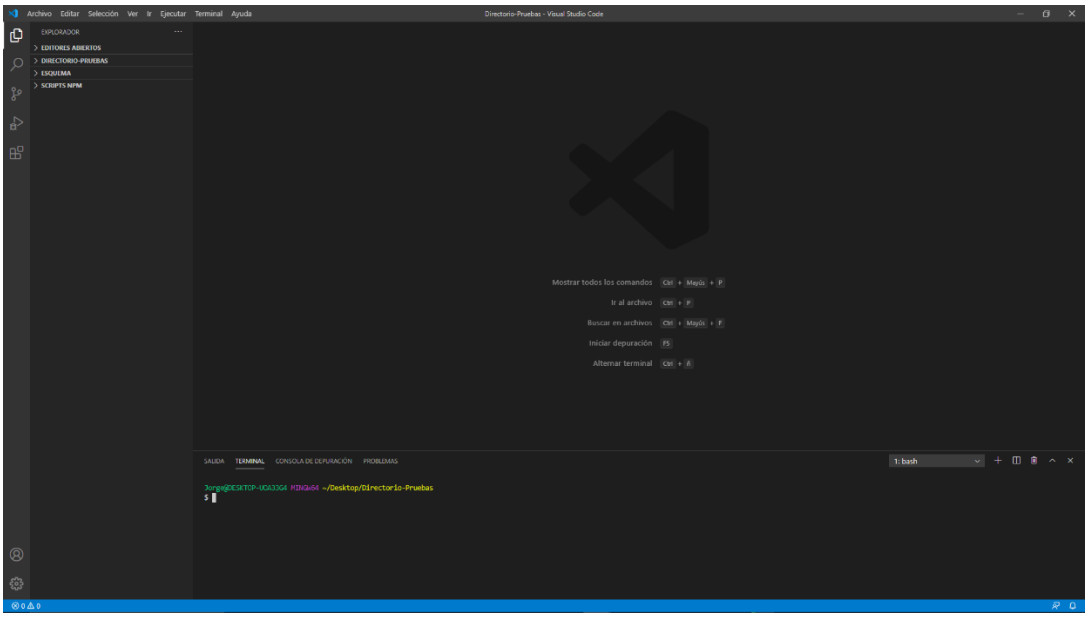

FIGURA 61: VSCODE TERMINAL

Hecho esto, instalaremos Angular CLI (*Command Line Interface*) mediante el siguiente comando:

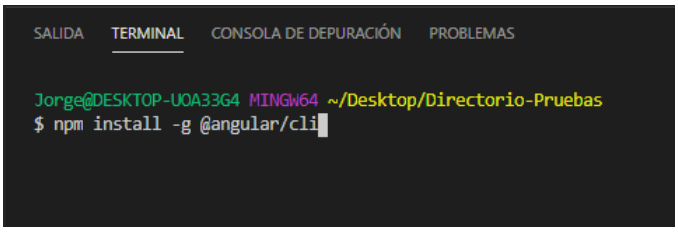

FIGURA 62: INSTALACIÓN ANGULAR CLI

Una vez instalado, podremos crear nuestra aplicación Angular usando el siguiente comando:

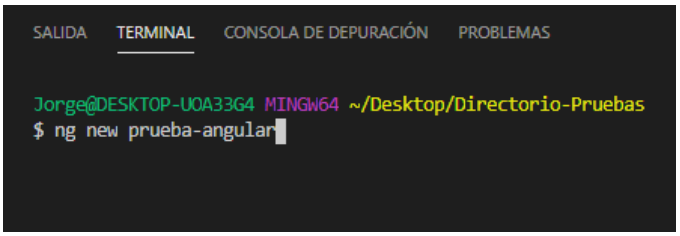

FIGURA 63: CREACIÓN APP ANGULAR

Durante la creación se nos presentarán dos cuestiones que, aunque realmente para la aplicación que nos atañe no importan, tenemos que especificar. La primera es si queremos que se cree el *routing* de la aplicación automáticamente, y en nuestro caso indicaremos que sí.

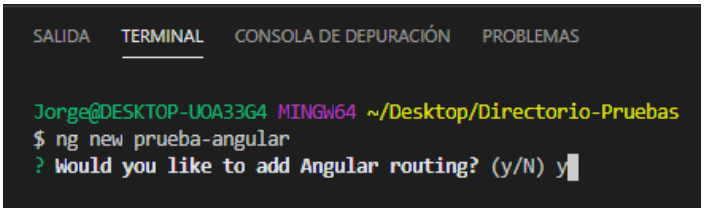

FIGURA 64: ROUTING APP ANGULAR

La segunda cuestión es el tipo de hoja de estilos que queremos emplear para la aplicación, en nuestro caso seleccionaremos la opción preseleccionada, CSS.

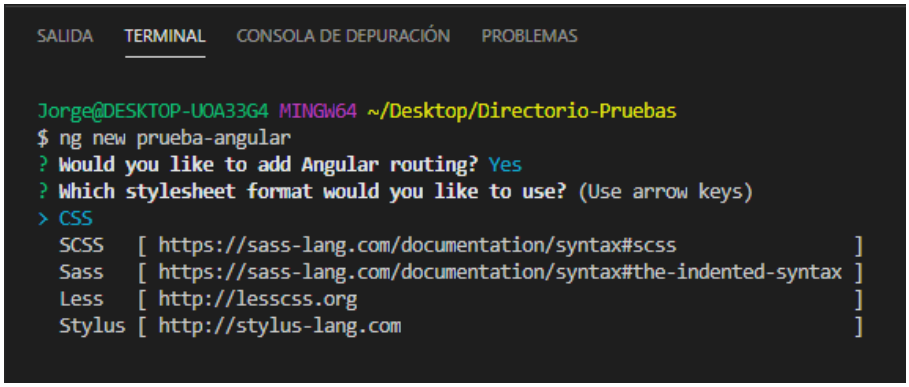

FIGURA 65: HOJAS DE ESTILOS APP ANGULAR

Una vez hagamos esto, la aplicación se habrá creado correctamente.

El siguiente paso sería movernos al directorio raíz de la aplicación:

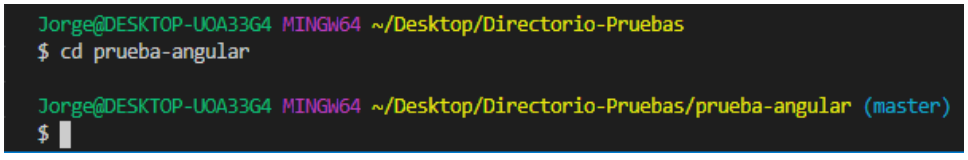

FIGURA 66: DIRECTORIO DE LA APP

Una vez aquí, ejecutaremos el siguiente comando, el cual lanzará nuestra aplicación y la mostrará automáticamente en el navegador:

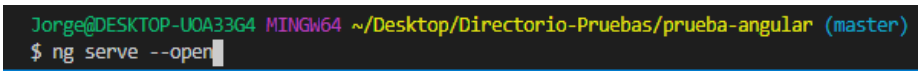

FIGURA 67: EJECUTAR APP

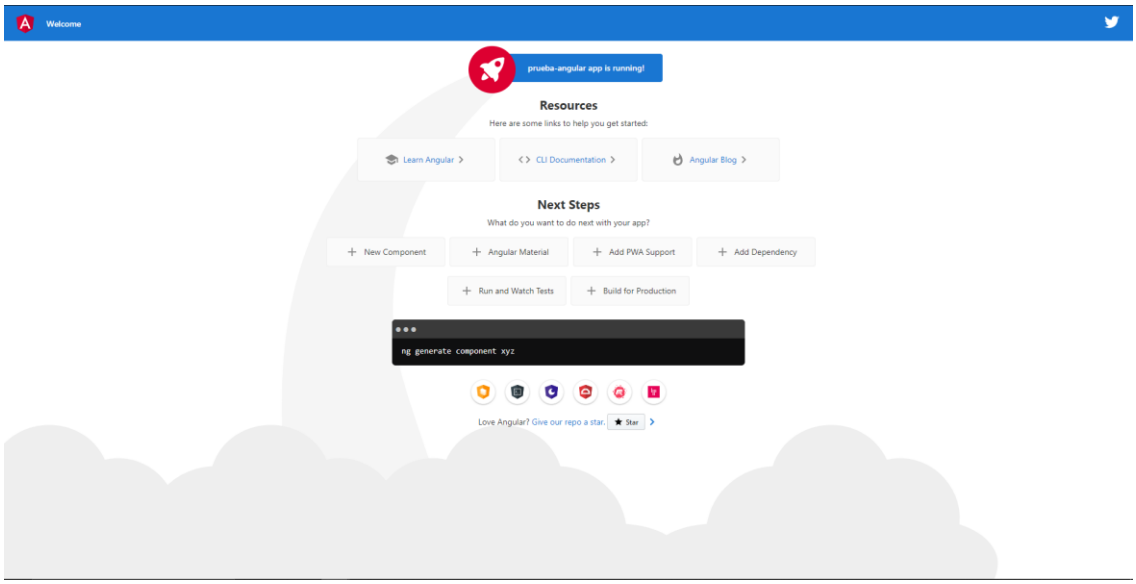

FIGURA 68: APP ANGULAR PÁGINA PRINCIPAL

Nuestra app Angular ya está lista para comenzar el desarrollo.

6.7. Heroku

Durante el despliegue de nuestra aplicación usaremos la herramienta Heroku para desplegar el front-end. Como ya hemos explicado en el apartado de Herramientas de Desarrollo, Heroku permite desplegar aplicaciones en la nube mediante el uso de contenedores (*dynos*). Las características de Heroku por las cuales nos hemos decidido por su uso son las siguientes:

- Heroku soporta múltiples lenguajes de programación.
- Cuenta con un *tier* gratuito, mediante el cual se pueden acceder a características básicas de la plataforma, tales como el uso de add-ons.
- Cuenta con una integración completa con GitHub, de modo que los cambios que se hacen en un repositorio se pueden desplegar en una nueva versión de la aplicación.
- Cuenta con una función de auto despliegue, la cual detecta los cambios realizados en el repositorio de la aplicación y automáticamente despliega la nueva versión.
- Heroku tiene un alto nivel de seguridad, ya que los *dynos* están aislados entre sí y las conexiones a los distintos servicios se hacen a través de HTTPS.
- Usar Heroku permite que una aplicación sea escalable tanto horizontalmente (aumentar el número de *dynos*) como verticalmente (aumentar los recursos de un *dyno*).

7. PRUEBAS

En esta sección se especificarán las pruebas llevadas a cabo para comprobar el correcto comportamiento de la aplicación. Para ello vamos a realizar pruebas de caja negra.

7.1. Pruebas de Caja Negra

Las pruebas de caja negra se llevan a cabo para comprobar el buen funcionamiento de las funcionalidades de la aplicación. Este tipo de pruebas se llevan a cabo de manera que no sea necesario tener conocimiento del funcionamiento y estructura del código.

A continuación mostraremos las pruebas de caja negra realizadas.

Pruebas de usuarios no administradores

PCN-01: Registrar usuario	
Objetivo	Comprobar que es posible registrarse en la aplicación.
Precondiciones	Ninguna
Datos de entrada	• Email: usuario@prueba.com • Contraseña: usuarioprueba • Nombre: usuario • Apellidos: prueba • Fecha de nacimiento: 13/06/1996 • Teléfono: 444666555
Acción esperada	Al hacer clic sobre el botón “Registrar” se dará de alta al usuario en el sistema
Secuencia	1. Pulsar el botón de login en el home. 2. Seleccionar la opción de registro de usuario. 3. Introducir los datos del nuevo usuario. 4. Pulsar el botón “Registrar”.
Resultado	Satisfactorio

PCN-02: Visualizar lista de productos	
Objetivo	Comprobar que es posible visualizar el listado de productos de la tienda.
Precondiciones	Ninguna
Datos de entrada	Ninguno
Acción esperada	Al hacer clic sobre el botón “Tienda” se navegará a la tienda online y se mostrará el listado de productos de dicha tienda.
Secuencia	1. Pulsar el botón de “Tienda” en el home.
Resultado	Satisfactorio

PCN-03: Seleccionar producto	
Objetivo	Comprobar que es posible seleccionar un producto del listado para ver sus características.
Precondiciones	Ninguna
Datos de entrada	Ninguno
Acción esperada	Al hacer clic sobre un producto se accederá a toda su información.
Secuencia	1. Hacer clic en un producto del listado.
Resultado	Satisfactorio

PCN-04: Iniciar sesión	
Objetivo	Comprobar que es posible iniciar sesión en la aplicación.
Precondiciones	Estar registrado en la aplicación.
Datos de entrada	• Email: usuario@prueba.com • Contraseña: usuarioprueba
Acción esperada	Al hacer clic sobre el botón "Login" se iniciará sesión en el sistema.
Secuencia	1. Pulsar el botón de login en el home. 2. Introducir los datos del usuario registrado. 3. Pulsar el botón "Login".
Resultado	Satisfactorio

PCN-05: Cerrar sesión	
Objetivo	Comprobar que es posible cerrar sesión en la aplicación.
Precondiciones	Estar autenticado en la aplicación.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón de cerrar sesión se cerrará la sesión del usuario.
Secuencia	1. Pulsar el botón de cerrar sesión.
Resultado	Satisfactorio

PCN-06: Visualizar perfil	
Objetivo	Comprobar que el usuario puede visualizar los datos de su perfil.
Precondiciones	Estar autenticado en el sistema.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón "Perfil" se mostrarán los datos del perfil del usuario.
Secuencia	1. Pulsar el botón "Perfil" en el home.
Resultado	Satisfactorio

PCN-07: Modificar perfil	
Objetivo	Comprobar que el usuario puede modificar los datos de su perfil.
Precondiciones	Estar autenticado en el sistema.
Datos de entrada	• Email: usuario@prueba.com • Nombre: usuario • Apellidos: prueba • Fecha de nacimiento: 13/06/1996 • Teléfono: 444666555 • Dirección: Plaza de la Universidad, 2 • Municipio: Segovia • Provincia: Segovia • Dni: 70565842D
Acción esperada	Al introducir los nuevos datos y hacer clic en "Guardar cambios" se modificarán los datos del perfil del usuario.
Secuencia	1. Pulsar el botón "Perfil" en el home. 2. Modificar los campos deseados. 3. Pulsar el botón "Guardar cambios".
Resultado	Satisfactorio

PCN-08: Visualizar chat	
Objetivo	Comprobar que es posible visualizar el chat entre usuarios.
Precondiciones	Estar autenticado en la aplicación.
Datos de entrada	Ninguno
Acción esperada	Al hacer clic sobre el botón "Chat" se accederá al chat y se podrán visualizar los mensajes.
Secuencia	1. Pulsar el botón "Chat" en el perfil de usuario.
Resultado	Satisfactorio

PCN-09: Enviar mensaje	
Objetivo	Comprobar que es posible enviar mensajes en el chat entre usuarios.
Precondiciones	Estar autenticado en el sistema.
Datos de entrada	• Mensaje: Hola buenos días.
Acción esperada	Al hacer introducir el mensaje deseado y pulsar el botón "Enviar", se enviará el mensaje y aparecerá en el chat.
Secuencia	1. Introducir el mensaje deseado. 2. Pulsar el botón "Enviar".
Resultado	Satisfactorio

PCN-10: Visualizar reservas activas	
Objetivo	Comprobar que es posible visualizar las reservas activas del usuario.
Precondiciones	El usuario debe estar autenticado en el sistema.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón "Sus reservas" en el perfil de usuario se mostrarán todas las reservas activas del usuario.
Secuencia	1. Pulsar el botón "Sus reservas" en el perfil de usuario.
Resultado	Satisfactorio

PCN-11: Visualizar pedidos	
Objetivo	Comprobar que es posible visualizar los pedidos activos del usuario.
Precondiciones	El usuario debe estar autenticado en el sistema.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón "Sus pedidos" se mostrarán los pedidos realizados por el usuario.
Secuencia	1. Pulsar el botón "Sus pedidos" en el perfil de usuario.
Resultado	Satisfactorio

PCN-12: Visualizar cesta	
Objetivo	Comprobar que es posible visualizar la cesta del usuario y los productos que haya añadido a ella.
Precondiciones	El usuario debe estar autenticado en el sistema.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón de la cesta se mostrará su contenido.
Secuencia	1. Pulsar el botón de la cesta en la tienda.
Resultado	Satisfactorio

PCN-13: Eliminar producto de cesta	
Objetivo	Comprobar que es posible eliminar un artículo de la cesta del usuario.
Precondiciones	El usuario debe estar autenticado en el sistema y haber añadido un producto a su cesta.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón de eliminar elemento en la cesta, este debe desaparecer de la cesta.
Secuencia	1. Pulsar el botón de eliminar producto en la cesta.
Resultado	Satisfactorio

PCN-14: Añadir producto a cesta	
Objetivo	Comprobar que es posible añadir un producto de la tienda a la cesta del usuario.
Precondiciones	El usuario debe estar autenticado en la aplicación y haber visualizado un producto de la tienda.
Datos de entrada	Ninguno.
Acción esperada	Al seleccionar una cantidad del producto y hacer clic en el botón "Añadir al carrito", el producto debe añadirse a la cesta del usuario.
Secuencia	1. Seleccionar la cantidad deseada del producto. 2. Pulsar el botón "Añadir al carrito".
Resultado	Satisfactorio

PCN-15: Realizar pedido	
Objetivo	Comprobar que es posible realizar un pedido en la aplicación.
Precondiciones	El usuario debe estar autenticado en la aplicación.
Datos de entrada	• Lista de productos añadidos a la cesta.
Acción esperada	Al hacer clic sobre el botón de PayPal se redigirá al usuario a la ventana de PayPal, donde podrá realizar el pago del pedido.
Secuencia	1. Pulsar el botón de PayPal en la cesta. 2. Introducir los datos de la cuenta de PayPal del usuario. 3. El usuario realiza el pago del pedido.
Resultado	Satisfactorio

PCN-16: Reservar	
Objetivo	Comprobar que es posible realizar la reserva de una pista en la aplicación.
Precondiciones	El cliente debe estar autenticado en el sistema.
Datos de entrada	• Fecha: 22/12/2020 • Pista: Pista 1 • Sesión: 11:00
Acción esperada	Al introducir una fecha, una pista y elegir una sesión, se realizará una reserva de pista con esos parámetros.
Secuencia	1. Pulsar el botón "Reservar pista" en el home. 2. Introducir la fecha y la pista deseadas. 3. Pulsar el botón "Aceptar". 4. Seleccionar una sesión de entre las disponibles.
Resultado	Satisfactorio

PCN-17: Cancelar reserva	
Objetivo	Comprobar que el usuario puede cancelar una reserva realizada.
Precondiciones	El usuario debe estar autenticado en el sistema y haber realizado una reserva de una pista.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre el botón de cancelar reserva en una reserva determinada se preguntará al usuario si desea cancelar dicha reserva. Si la respuesta es afirmativa, la reserva se eliminará del sistema.
Secuencia	1. Pulsar el botón de cancelar reserva en las reservas del usuario. 2. Confirmar la cancelación de la reserva.
Resultado	Satisfactorio

Pruebas usuario administrador

PCN-18: Visualizar catálogo	
Objetivo	Comprobar que es posible visualizar el catálogo de productos de la aplicación.
Precondiciones	Haber iniciado sesión como administrador y haber accedido al Panel de Administración.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre la sección "Productos" en el Panel de Administración aparecerá un listado con todos los productos presentes en la aplicación.
Secuencia	1. Seleccionar la sección de "Productos" en el Panel de Administración.
Resultado	Satisfactorio
PCN-19: Añadir producto	
Objetivo	Comprobar que es posible añadir productos al catálogo de la tienda.
Precondiciones	Haber iniciado sesión como administrador, y, dentro del Panel de Administración, haber accedido al listado de productos.
Datos de entrada	• Nombre: Bullpadel Flow • Descripción: Descubre las altas prestaciones de la nueva pala Bullpadel Flow 21 para mujer destinada a las jugadoras exigentes de nivel profesional que buscan una pala resistente y de altas prestaciones para llevar a la pista. Fabricada con la goma soft que ayuda a reducir las vibraciones. Además, está compuesta con materiales de carbono que proporciona mucha rigidez y durabilidad. • Precio: 199,99. • Imagen: Imagen seleccionada. • Stock: 10. • Categoría: Palas de pádel.
Acción esperada	Al seleccionar la opción de añadir producto, introducir los datos del nuevo producto y pulsar el botón "Grabar", se añadirá el nuevo producto al sistema.
Secuencia	1. Seleccionar la opción de añadir un nuevo producto. 2. Introducir los datos del nuevo producto. 3. Pulsar el botón "Grabar".
Resultado	Satisfactorio

PCN-20: Modificar producto	
Objetivo	Comprobar que el administrador puede modificar los datos de un producto de la tienda.
Precondiciones	Haber iniciado sesión como administrador, y, dentro de la sección de Productos, haber seleccionado uno.
Datos de entrada	• Nombre: Bullpadel Flow • Descripción: Descubre las altas prestaciones de la nueva pala Bullpadel Flow 21 para mujer destinada a las jugadoras exigentes de nivel profesional que buscan una pala resistente y de altas prestaciones para llevar a la pista. Fabricada con la goma soft que ayuda a reducir las vibraciones. Además, está compuesta con materiales de carbono que proporciona mucha rigidez y durabilidad. • Precio: 249,95. • Imagen: Imagen seleccionada. • Stock: 10. Categoría: Palas de pádel.
Acción esperada	Al seleccionar un producto, haber modificado alguno de sus datos, y haber pulsado el botón “Grabar”, se deben haber modificado los datos del producto.
Secuencia	1. Introducir los nuevos datos del producto. 2. Pulsar el botón “Grabar”.
Resultado	Satisfactorio

PCN-21: Eliminar producto	
Objetivo	Comprobar que es posible eliminar un producto del catálogo de la tienda.
Precondiciones	Haber iniciado sesión como administrador, y, dentro de la sección de Productos, haber seleccionado uno.
Datos de entrada	Ninguno.
Acción esperada	Al haber seleccionado un producto y haber seleccionado la opción de eliminar, se debe haber eliminado el producto del catálogo.
Secuencia	1. Seleccionar el producto a eliminar. 2. Pulsar el botón “Eliminar”.
Resultado	Satisfactorio

PCN-22: Visualizar pedidos	
Objetivo	Comprobar que es posible visualizar los pedidos que los usuarios han realizado.
Precondiciones	Haber iniciado sesión como administrador y haber accedido al Panel de Administración.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre la sección "Pedidos" en el Panel de Administración aparecerá un listado con todos los pedidos realizados por los usuarios.
Secuencia	1. Seleccionar la sección de "Pedidos" en el Panel de Administración.
Resultado	Satisfactorio

PCN-23: Gestionar pedido	
Objetivo	Comprobar que es posible gestionar el estado de un pedido.
Precondiciones	Un usuario debe haber realizado algún pedido. Además, el administrador debe haber seleccionado un pedido dentro del listado.
Datos de entrada	• Estado del pedido: Recogido.
Acción esperada	Cuando el administrador cambie el estado del pedido, esto se deberá ver reflejado en la aplicación.
Secuencia	1. Seleccionar el pedido a gestionar. 2. Cambiar el estado del pedido. 3. Pulsar el botón "Grabar".
Resultado	Satisfactorio

PCN-24: Visualizar lista de pistas	
Objetivo	Comprobar que es posible visualizar el listado de las pistas de la aplicación.
Precondiciones	Haber iniciado sesión como administrador y haber accedido al Panel de Administración.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre la sección "Pistas" en el Panel de Administración aparecerá un listado con todas las pistas de la aplicación.
Secuencia	1. Seleccionar la sección de "Pistas" en el Panel de Administración.
Resultado	Satisfactorio

PCN-25: Añadir pista	
Objetivo	Comprobar que es posible registrarse en la aplicación.
Precondiciones	Comprobar que es posible añadir pistas a la aplicación.
Datos de entrada	• Nombre: Pista 1 • Disponibilidad de la pista: True
Acción esperada	Al seleccionar la opción de añadir pista, introducir los datos de la nueva pista y pulsar el botón "Grabar", se añadirá la nueva pista al sistema.
Secuencia	1. Seleccionar la opción de añadir una nueva pista. 2. Introducir los datos de la nueva pista. 3. Pulsar el botón "Grabar".
Resultado	Satisfactorio

PCN-26: Eliminar pista	
Objetivo	Comprobar que es posible eliminar una pista del sistema.
Precondiciones	Haber iniciado sesión como administrador, y, dentro de la sección de Pistas, haber seleccionado una.
Datos de entrada	Ninguno.
Acción esperada	Al haber seleccionado una pista y haber seleccionado la opción de eliminar, se debe haber eliminado la pista del catálogo.
Secuencia	1. Seleccionar la pista a eliminar. 2. Pulsar el botón "Eliminar".
Resultado	Satisfactorio

PCN-27: Modificar pista	
Objetivo	Comprobar que el administrador puede modificar los datos de una pista.
Precondiciones	Haber iniciado sesión como administrador, y, dentro de la sección de Pistas, haber seleccionado una.
Datos de entrada	• Nombre: Pista 1 • Disponibilidad de la pista: False
Acción esperada	Al seleccionar una pista, haber modificado alguno de sus datos, y haber pulsado el botón "Grabar", se deben haber modificado los datos de dicha pista.
Secuencia	1. Introducir los nuevos datos de la pista. 2. Pulsar el botón "Grabar".
Resultado	Satisfactorio

PCN-28: Visualizar lista de usuarios	
Objetivo	Comprobar que es posible visualizar la lista de usuarios de la aplicación.
Precondiciones	Haber iniciado sesión como administrador y haber accedido al Panel de Administración.
Datos de entrada	Ninguno.
Acción esperada	Al hacer clic sobre la sección “Usuarios” en el Panel de Administración aparecerá un listado con todos los usuarios registrados en la aplicación.
Secuencia	1. Seleccionar la sección de “Usuarios” en el Panel de Administración.
Resultado	Satisfactorio

PCN-29: Eliminar cuenta de usuario	
Objetivo	Comprobar que es posible eliminar una cuenta de usuario de la aplicación.
Precondiciones	Haber iniciado sesión como administrador, y, dentro de la sección de Usuarios, haber seleccionado uno.
Datos de entrada	Ninguno.
Acción esperada	Al haber seleccionado una cuenta de usuario y haber seleccionado la opción de eliminar, se debe haber eliminado dicha cuenta del sistema.
Secuencia	1. Seleccionar la cuenta de usuario a eliminar. 2. Pulsar el botón “Eliminar”.
Resultado	Satisfactorio

8. MANUALES

A la hora de explicar el modo de uso de la aplicación, no vamos a dividir la explicación por los actores especificados en la sección de Análisis de este documento. En su lugar vamos a diferenciar entre Usuario No Autenticado, Cliente y Administrador. De esta manera simplificaremos las explicaciones.

8.1. Manual de despliegue

En este apartado vamos a detallar el proceso de instalación del software necesario para desplegar la aplicación.

Lo primero será descargar el sistema gestor de base de datos PostgreSQL, con el cual también se instalará el entorno pgAdmin, necesario para conectarnos a la base de datos. Para descargar PostgreSQL nos dirigiremos a la página oficial https://www.postgresql.org/ y descargaremos el instalador correspondiente a nuestro sistema operativo.

Una vez el instalador de PostgreSQL esté descargado, procederemos con su instalación. Los pasos relevantes de la instalación son los siguientes:

En esta ventana el instalador nos pedirá que especifiquemos el directorio en el cual se instalará PostgreSQL. Recomendamos dejar el directorio por defecto.

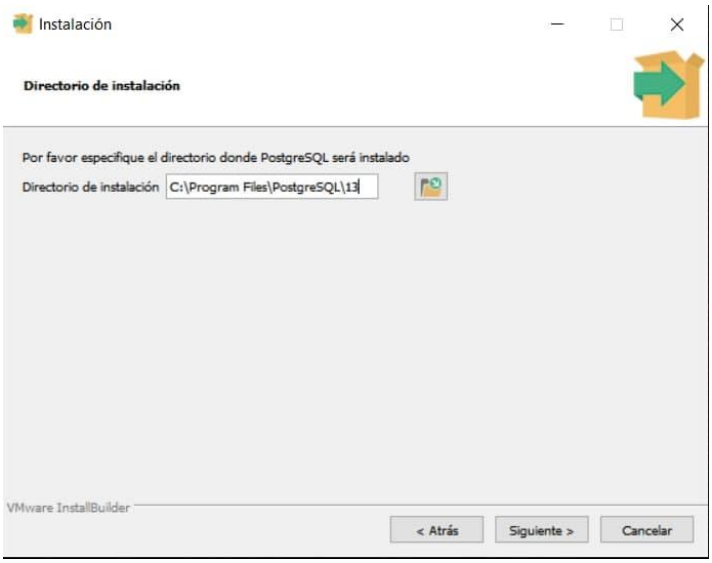

FIGURA 69: DIRECTORIO INSTALACIÓN POSTGRESQL

En esta ventana el instalador nos pedirá que especifiquemos que componentes queremos instalar en nuestro equipo, en nuestro caso marcaremos todas las casillas.

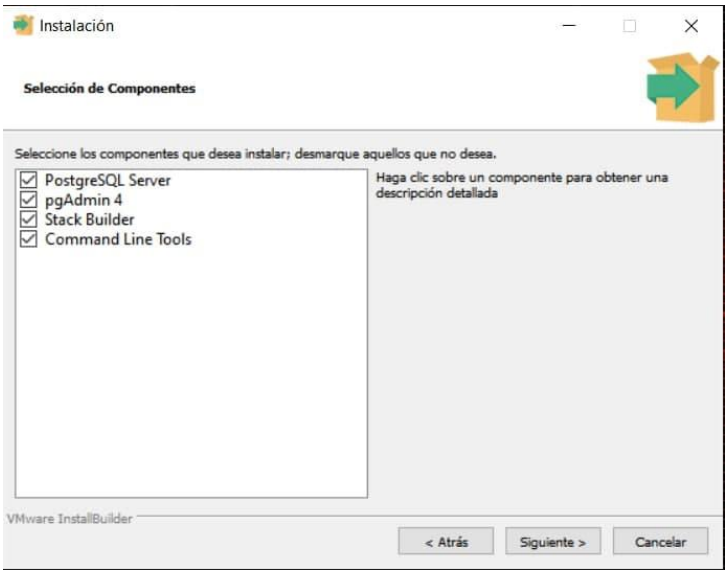

FIGURA 70: COMPONENTES POSTGRESQL

La última ventana del instalador, una vez ya terminado el proceso de instalación, nos preguntará si queremos ejecutar la herramienta Stack Builder. Seleccionaremos la opción de no ejecutar y saldremos del instalador.

FIGURA 71: FIN INSTALACIÓN POSTGRESQL

El siguiente paso es instalar PyCharm Community Edition, necesario para desplegar el backend de nuestra aplicación. Para descargar PyCharm nos dirigiremos a la página oficial https://www.jetbrains.com/pycharm/ y descargaremos el instalador de PyCharm Community Edition.

Una vez el instalador de PyCharm esté descargado, procederemos con su instalación. Los pasos relevantes de la instalación son los siguientes:

En esta ventana el instalador nos pedirá que especifiquemos el directorio en el cual se instalará PyCharm. Recomendamos dejar el directorio por defecto.

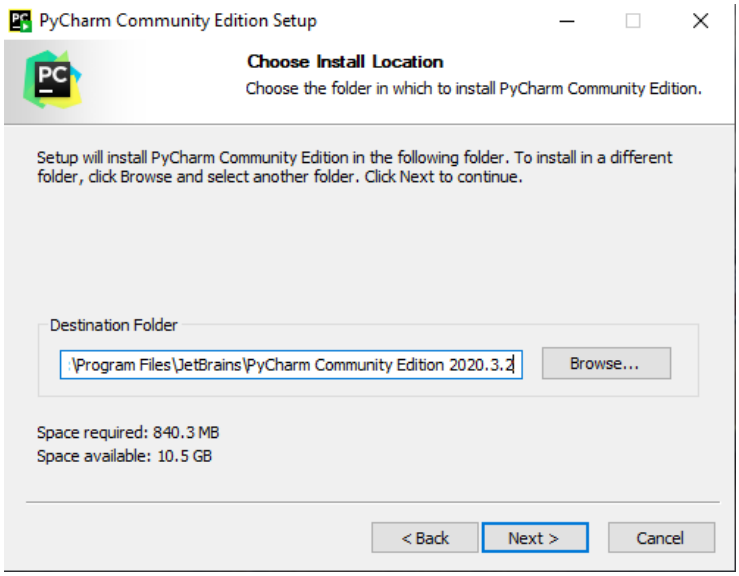

FIGURA 72: DIRECTORIO INSTALACIÓN PYCHARM

En esta ventana el instalador nos pedirá que seleccionemos las opciones de instalación que necesitemos. En nuestro caso no seleccionamos ninguna.

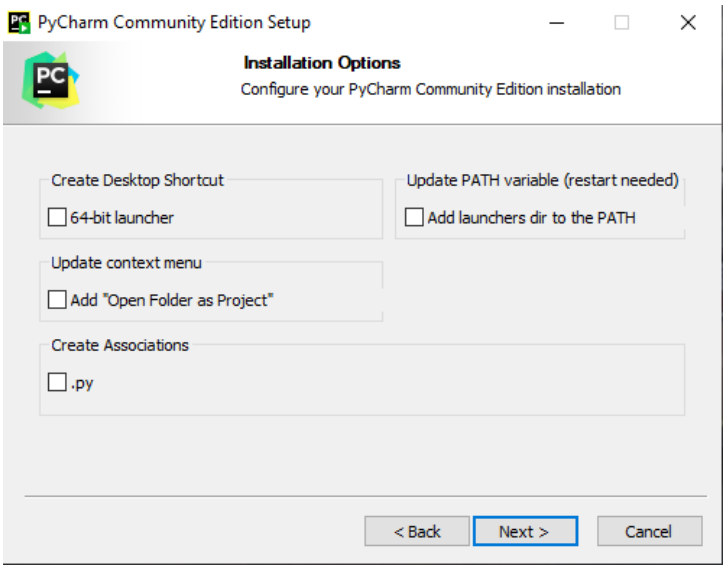

FIGURA 73: PYCHARM OPCIONES INSTALACIÓN

Una vez instalados PostgreSQL y PyCharm, el siguiente paso será crear la base de datos en pgAdmin, para posteriormente hacer el volcado de los datos en dicha base de datos.

El proceso es el siguiente:

En primer lugar abriremos pgAdmin e ingresaremos con nuestro usuario y contraseña. Una vez hayamos ingresado, crearemos la base de datos haciendo clic derecho en el desplegable de las bases de datos del sistema y seleccionando la opción “Create Database”.

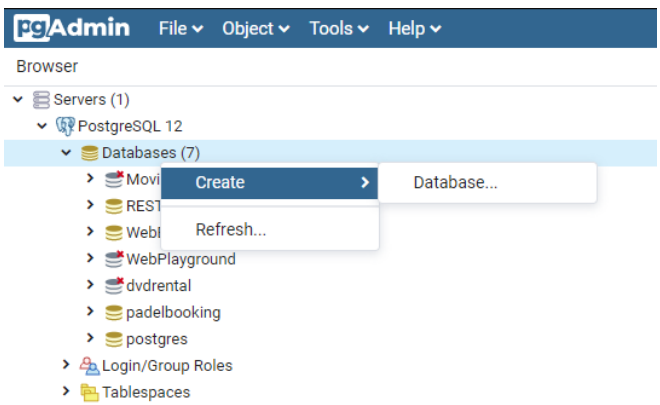

FIGURA 74: CREACIÓN BASE DE DATOS

Al seleccionar esta opción, el programa nos pedirá que introduzcamos los datos necesarios para crear la base de datos. El único dato que tenemos que ingresar es el nombre de la base de datos, el cual será “padelbooking”. Posteriormente pulsaremos el botón “Save” y se habrá creado la base de datos.

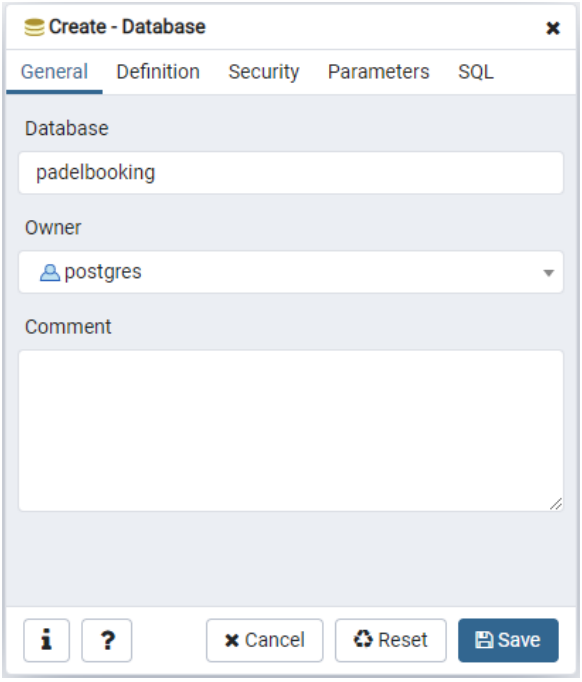

FIGURA 75: CREACIÓN BASE DE DATOS 2

Ahora seleccionaremos la base de datos recién creada y seleccionaremos la opción “Query Tool” en el desplegable “Tools” de la parte superior de la pantalla. Al hacer esto, aparecerá dicha Query Tool en la parte derecha de la pantalla.

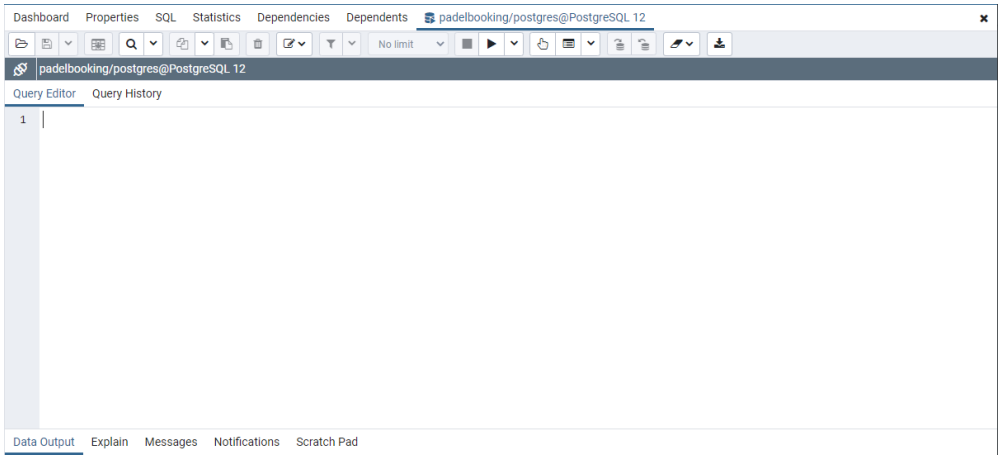

FIGURA 76: PGADMIN QUERY TOOL

Pulsaremos el botón en el extremo izquierdo de la barra de opciones, “Open File”, el cual nos permitirá seleccionar el archivo con los datos de la base de datos que queremos importar.

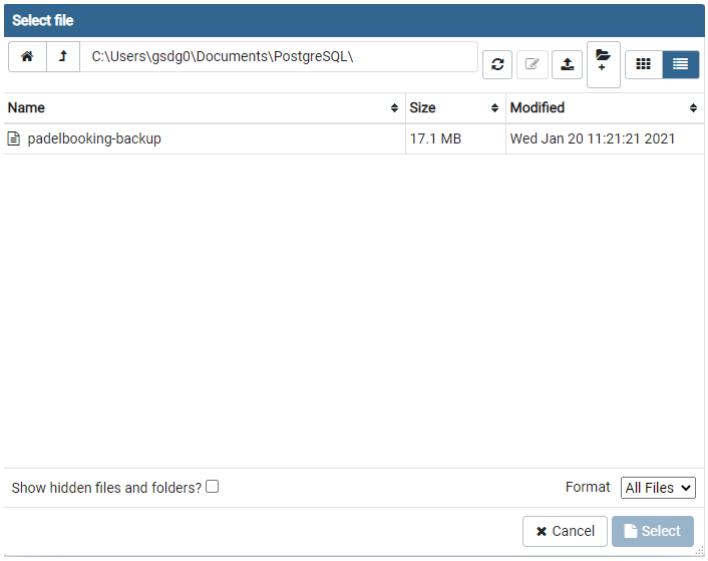

FIGURA 77: ARCHIVO BACKUP

Una vez hayamos seleccionado el archivo, ejecutaremos la query con el botón F5, y se habrá realizado el volcado de la base de datos.

El siguiente paso será configurar la conexión con la base de datos. Para ello abriremos con PyCharm la carpeta “padelbooking”, la cual contiene el backend del proyecto y, en el fichero settings.py, buscaremos la siguiente sección:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'padelbooking',
        'USER': 'postgres',
        'PASSWORD': 'root',
        'HOST': '',
        'PORT': '5432'
    }
}
```

FIGURA 78: SETTINGS BASE DE DATOS

Deberemos cambiar los datos de “USER”, “PASSWORD” y “PORT” con los nuestros datos de usuario de pgAdmin y el puerto en el que escucha PostgreSQL (por defecto el puerto 5432).

Una vez hecho esto, el siguiente paso será instalar las dependencias necesarias para ejecutar el backend. Para ello instalaremos en la consola las siguientes librerías.

```
pip install Django
pip install djangorestframework
pip install django-cors-headers
pip install django-filter
pip install psycopg2
pip install django-enumchoicefield
pip install Pillow
```

FIGURA 79: INSTALACIÓN LIBRERÍAS

Una vez instaladas dichas dependencias, sólo quedará ejecutar el siguiente comando en la consola:

```
(venv) C:\Users\gsdg0\Documents\Universidad\TFG\padelbooking>python manage.py runserver
```

FIGURA 80: LEVANTAR PROYECTO

Ahora podremos acceder a la aplicación en el navegador, en la dirección https://padelbooking.herokuapp.com/ .

8.2. Manual de Usuario

8.2.1. Perfil: Usuario No Autenticado

En esta parte del manual detallaremos los pasos a seguir para un Usuario No Autenticado que acceda a nuestra aplicación. Al acceder, lo que nos encontraremos será la página principal de la aplicación:

FIGURA 81: PANTALLA PRINCIPAL DE LA APLICACIÓN

En esta página el Usuario No Autenticado puede llevar a cabo varias acciones. Puede seleccionar el botón a la derecha de la pantalla, el cual le llevará a la página de registro y de inicio de sesión, o bien puede acceder a la tienda. Si el Usuario No Autenticado pulsa cualquiera de los otros botones del header, como el de perfil de usuario, será redirigido automáticamente a la pantalla de inicio de sesión, ya que estas áreas de la aplicación están restringidas a usuarios registrados.

Además, el Usuario No Autenticado puede visualizar las distintas políticas del sitio web, acceder a los perfiles del club en redes sociales, etc.

Como hemos mencionado, una de las opciones de este usuario es autenticarse en el sistema si ya tiene una cuenta, o, en caso contrario, registrarse en la aplicación.

Dicha página es la siguiente:

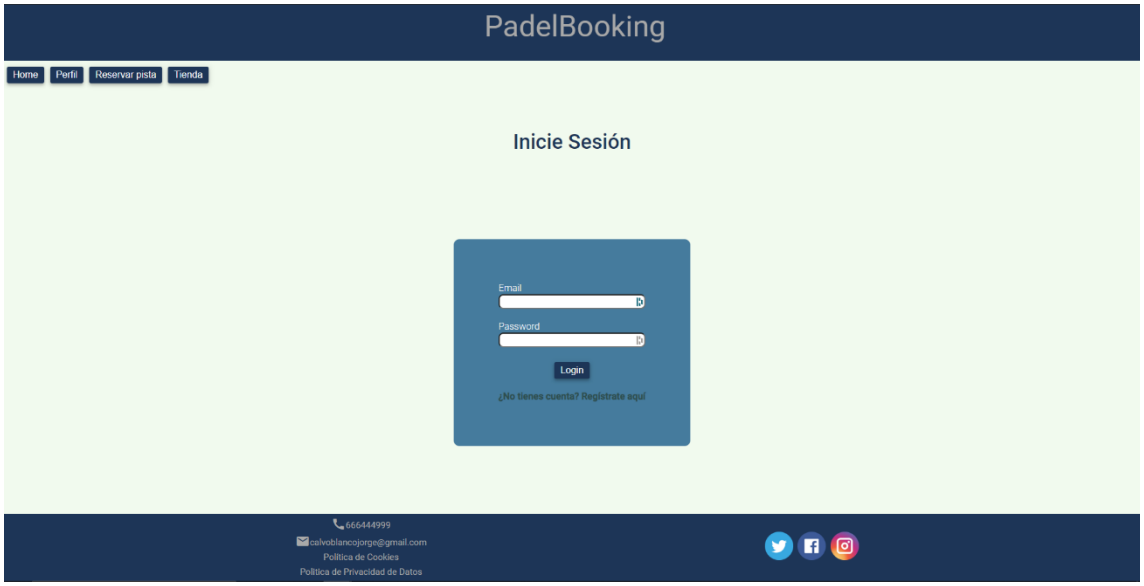

FIGURA 82: PANTALLA DE LOGIN DE LA APLICACIÓN

Si el Usuario No Autenticado ya tiene una cuenta en la página, deberá introducir sus datos de sesión, es decir, su email y su contraseña y posteriormente hacer clic en el botón de login. Si los datos de inicio de sesión son correctos, habrá iniciado sesión, se mostrará un mensaje al usuario y se le redigirá a la página de home.

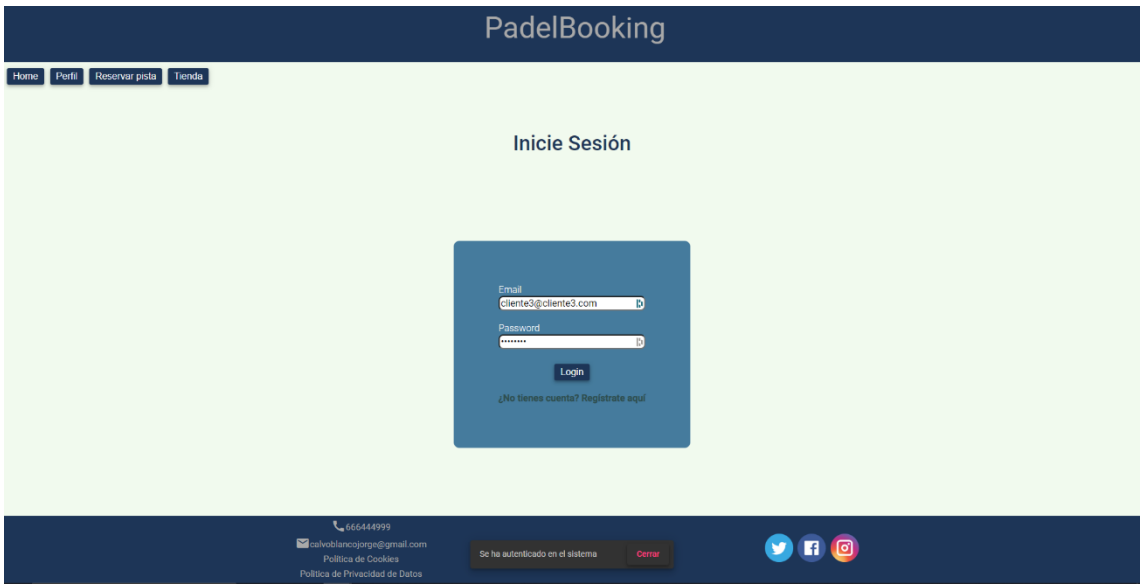

FIGURA 83: LOGIN EXITOSO

Si por el contrario hay un error en los campos de inicio de sesión, no se iniciará sesión y se notificará al usuario que hay campos erróneos en el formulario.

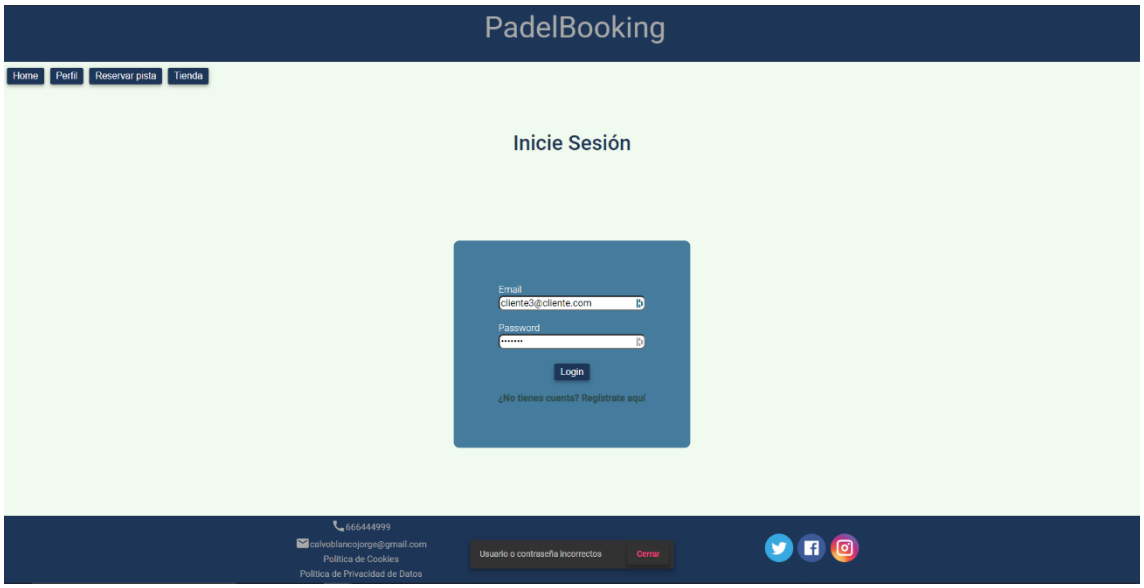

FIGURA 84: LOGIN ERRÓNEO

Por otro lado, si el usuario no tiene una cuenta en la aplicación, puede hacer clic en el mensaje que aparece debajo del botón de login para acceder a la página de registro.

FIGURA 85: FORMULARIO DE REGISTRO DE USUARIO

En esta página el usuario deberá introducir todos los datos necesarios para crear una cuenta en el sistema y pulsar el botón de registro. Si por algún motivo hubiera un error en el registro, se mostraría una alerta en la parte inferior de la pantalla para que el usuario revise los datos introducidos y se asegure que son correctos.

FIGURA 86: FORMULARIO DE REGISTRO INCORRECTO

En caso contrario, el registro se habrá realizado con éxito.

FIGURA 87: DATOS DE REGISTRO CORRECTOS

La otra opción del Usuario No Autenticado en la página de home era acceder a la tienda. Al acceder se mostrarán todos los productos disponibles, pudiendo filtrar por categoría de producto (palas, zapatillas…).

FIGURA 88: PANTALLA PRINCIPAL DE LA TIENDA

El botón del carrito en la esquina superior derecha de la pantalla sirve para visualizar los productos añadidos al carrito, pero, dado que el Usuario No Autenticado no puede añadir productos al carrito, si pulsa este botón se le redigirá a la pantalla de login.

Dentro de la tienda, el Usuario No Autenticado podrá seleccionar un producto para visualizar sus detalles.

FIGURA 89: PANTALLA DE DETALLES DE UN PRODUCTO

Sin embargo, como ya hemos mencionado antes, el Usuario No Autenticado no puede añadir productos al carrito, por lo que si intenta añadir el producto aparecerá un mensaje informándole que no es posible.

FIGURA 90: NECESARIA UNA CUENTA DE CLIENTE PARA AÑADIR UN PRODUCTO

8.2.2. Perfil: Usuario Cliente

En esta sección del manual detallaremos las opciones que tiene un Cliente dentro de nuestra aplicación.

En primer lugar, el Cliente se encuentra en la página de home, ya que al iniciar sesión es automáticamente redirigido a dicha página.

FIGURA 91: PANTALLA PRINCIPAL DE LA APLICACIÓN 2

En esta página se presentan varias opciones. Cerrar sesión, acceder a la tienda, reservar una pista y acceder al perfil de usuario.

Cerrar sesión: Si el Cliente pulsa el botón en la esquina superior derecha de la pantalla, cerrará su sesión en la aplicación y permanecerá en la página de home. Se le mostrará un mensaje indicando que su sesión ha sido cerrada.

FIGURA 92: SESIÓN CERRADA

Acceder a la tienda: Esta opción es la misma que se le presentaba al Usuario No Autenticado. Sin embargo, dentro de la tienda el Cliente puede añadir productos al carrito, visualizarlo, y realizar un pedido.

De esta manera, si el Cliente accede a los detalles de un producto, podrá seleccionar el número de unidades que quiere añadir al carrito. Posteriormente, al añadir el producto correctamente, aparecerá un mensaje indicando al Cliente que la operación ha sido exitosa.

FIGURA 93: AÑADIR PRODUCTO A CARRITO

Si el Cliente no introduce el número de unidades deseado u bien la cantidad introducida no es válida, el producto no se añadirá y aparecerá un mensaje informando al Cliente.

Para visualizar su carrito y los productos que ha añadido, el Cliente deberá acceder a él desde la pantalla de la tienda.

En la pantalla del carrito el Cliente podrá visualizar los productos que contiene su carrito, así como el número de unidades de cada uno.

FIGURA 94: CARRITO DE PRODUCTOS

Si lo desea, puede eliminar productos del carrito pulsando el botón asociado a cada producto del carrito. Al eliminarlo, se mostrará un mensaje informando al usuario de la eliminación del producto.

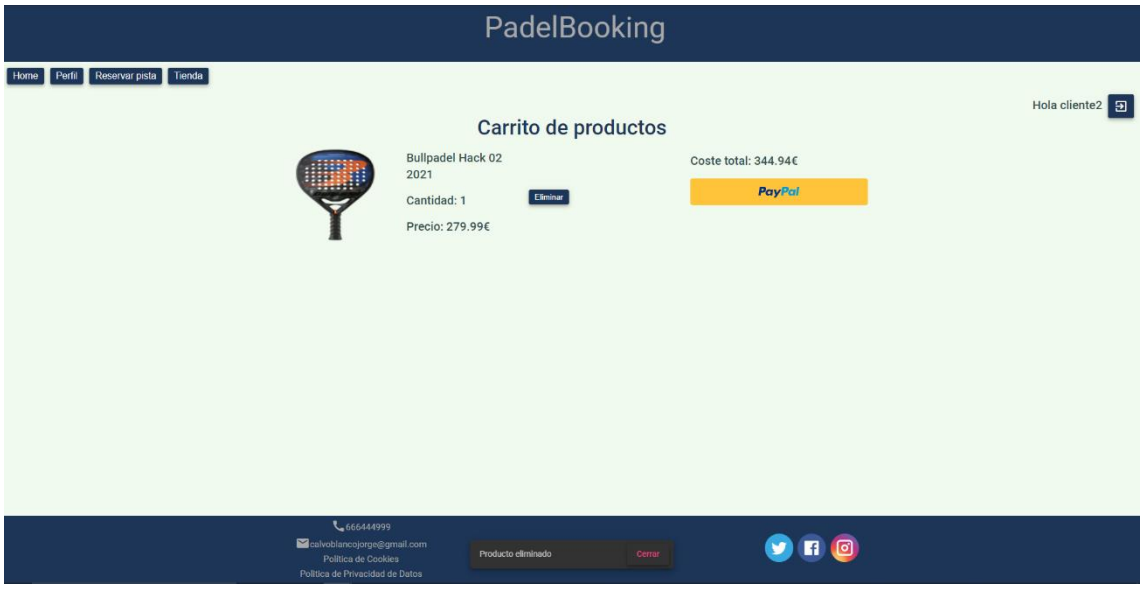

FIGURA 95: ELIMINAR PRODUCTO DE CARRITO

Si el Cliente desea realizar el pedido, deberá pulsar el botón de PayPal para continuar con el pago del pedido. Al hacer esto, se abrirá una ventana emergente de PayPal en la que se pedirá al Cliente sus credenciales para realizar el pago.

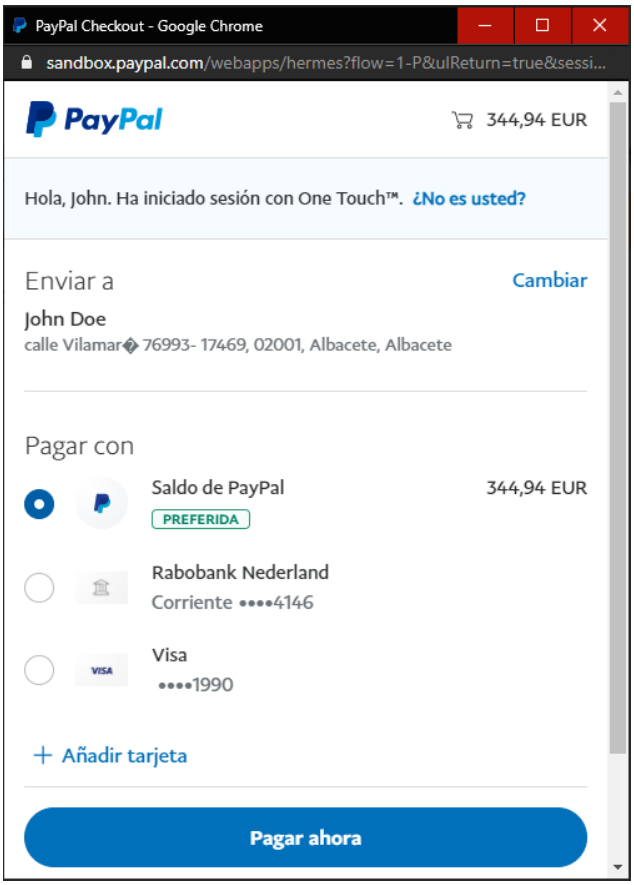

FIGURA 96: VENTANA DE PAGO PAYPAL

Si el Cliente las introduce correctamente y realiza el pago, se le redigirá a la aplicación con el pedido pagado, informándole del pago realizado a través de un mensaje emergente.

FIGURA 97: PEDIDO REALIZADO CON ÉXITO

Reservar una pista: Seleccionar esta opción lleva al Cliente a la página de reserva de pista.

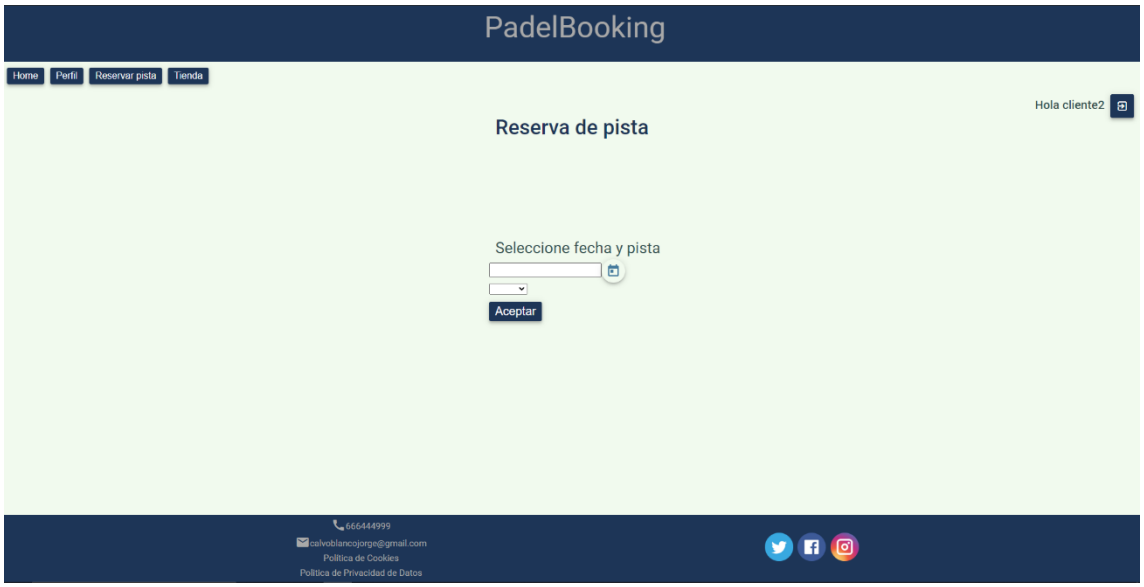

FIGURA 98: SELECCIONAR FECHA Y PISTA

En primer lugar, el Cliente tiene que seleccionar una fecha y una pista para realizar la reserva. Al elegirlas, se mostrará en pantalla las sesiones disponibles para esa fecha y esa pista.

FIGURA 99: SELECCIONAR SESIÓN PARA LA RESERVA

Si el Cliente selecciona una sesión, se realizará la reserva y aparecerá un mensaje informando al Cliente.

FIGURA 100: RESERVA REALIZADA

Acceder al perfil de usuario: Seleccionar esta opción lleva al Cliente a la página de su perfil de usuario. En esta página se le presentan varias opciones. Cambiar su información de perfil, acceder a sus reservas activas, acceder a sus pedidos y acceder al chat.

FIGURA 101: PERFIL DE USUARIO

Cambiar información de perfil: Si el Cliente desea añadir o modificar información de su perfil de usuario, deberá cambiar dicha información en los campos del perfil y posteriormente pulsar el botón de Guardar Cambios. Si los datos son correctos se mostrará un mensaje informando al Cliente de que los cambios han sido validados.

FIGURA 102: MODIFICAR PERFIL DE USUARIO

Acceder a reservas activas: Seleccionando esta opción el Cliente accederá a la página de sus reservas activas, en la que podrá visualizar las reservas activas que tiene.

FIGURA 103: RESERVAS ACTIVAS

En esta página también podrá cancelar una reserva pulsando el botón de eliminar en la columna derecha de la tabla. Al hacer esto, aparecerá una ventana emergente pidiendo al Cliente que confirme la acción.

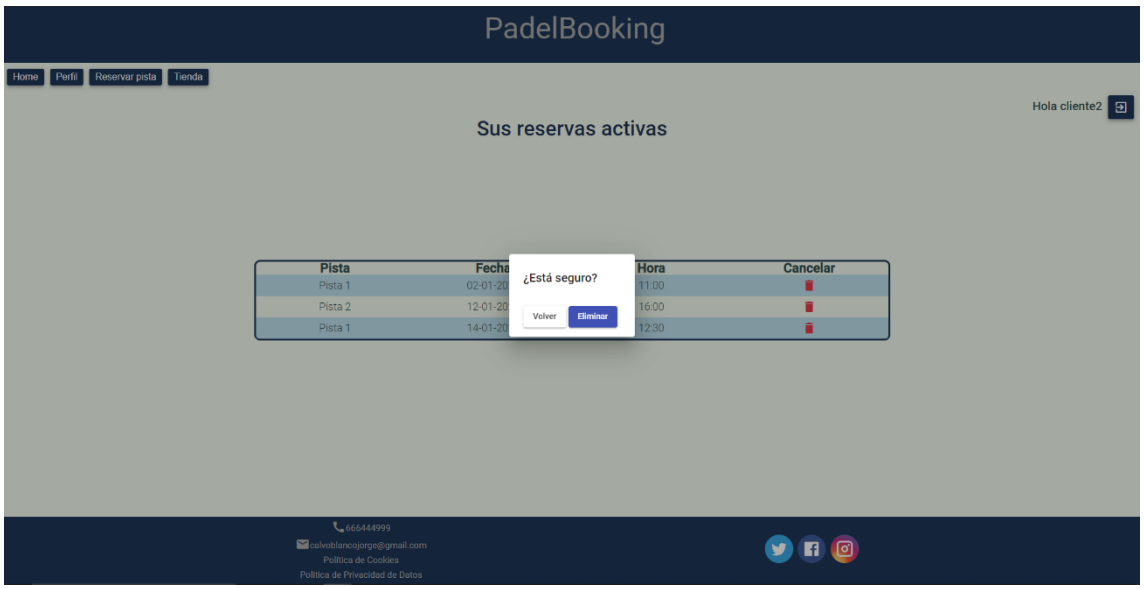

FIGURA 104: CANCELAR RESERVA

Si el Cliente confirma la acción, la reserva se eliminará y se mostrará un mensaje informando al Cliente.

FIGURA 105: RESERVA CANCELADA

Acceder a pedidos realizados: Seleccionando esta opción el Cliente será dirigido a su página de pedidos realizados, en la cual podrá visualizar todos los pedidos que ha realizado en la aplicación.

FIGURA 106: PEDIDOS REALIZADOS

En esta página se presentan dos opciones, cancelar un pedido realizado (siempre que no se haya entregado ya) y visualizar los detalles de un pedido.

El procedimiento para cancelar un pedido es el mismo que para cancelar una reserva.

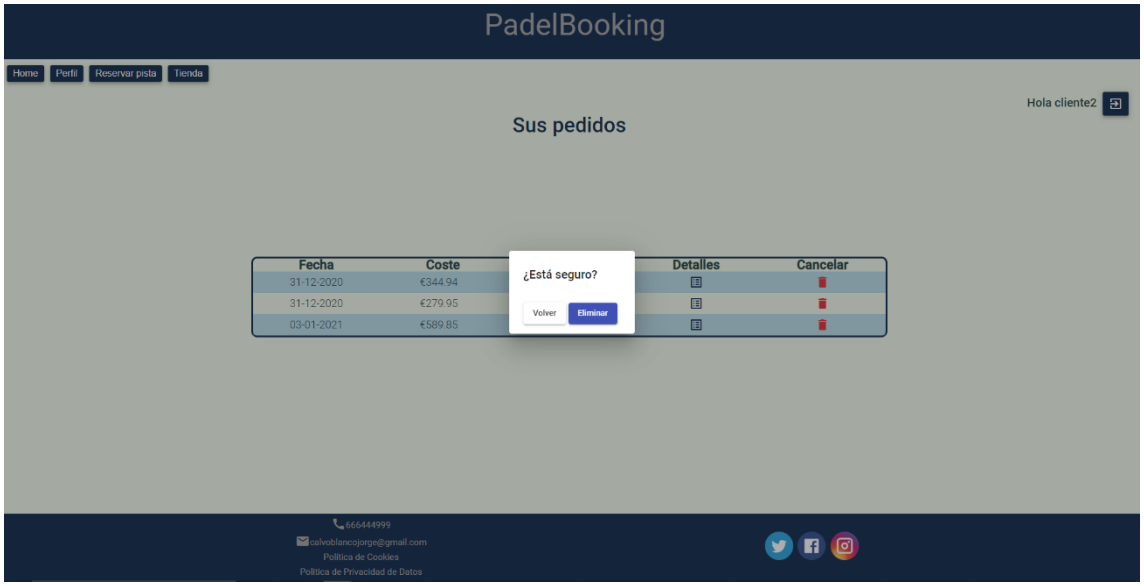

FIGURA 107: CANCELAR PEDIDO

FIGURA 108: PEDIDO CANCELADO

Para visualizar los detalles de un pedido el Cliente debe pulsar el botón correspondiente de la columna de detalles y será redirigido a la página de detalles del pedido.

FIGURA 109: DETALLES DE PEDIDO

Chat: El Cliente puede acceder a la página del chat pulsando en el botón correspondiente. En esta pantalla podrá visualizar los mensajes enviados por otros usuarios, además de enviar mensajes él mismo.

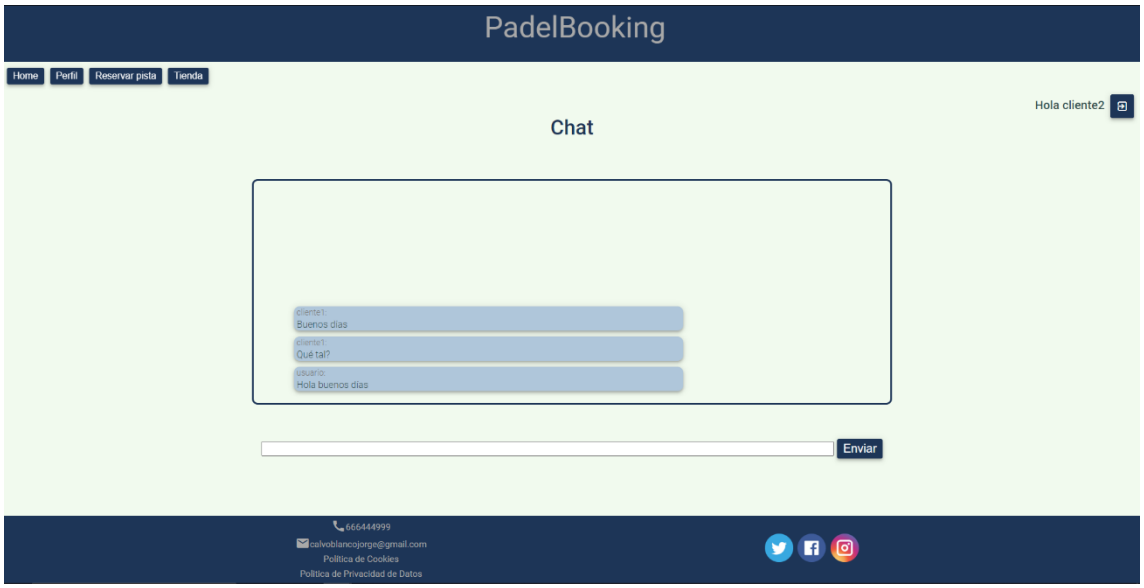

FIGURA 110: CHAT

Para enviar un mensaje el Cliente lo escribirá en el cuadro de texto y posteriormente pulsará enviar. Los mensajes propios aparecerán por la derecha del chat para diferenciarlos del resto de mensajes.

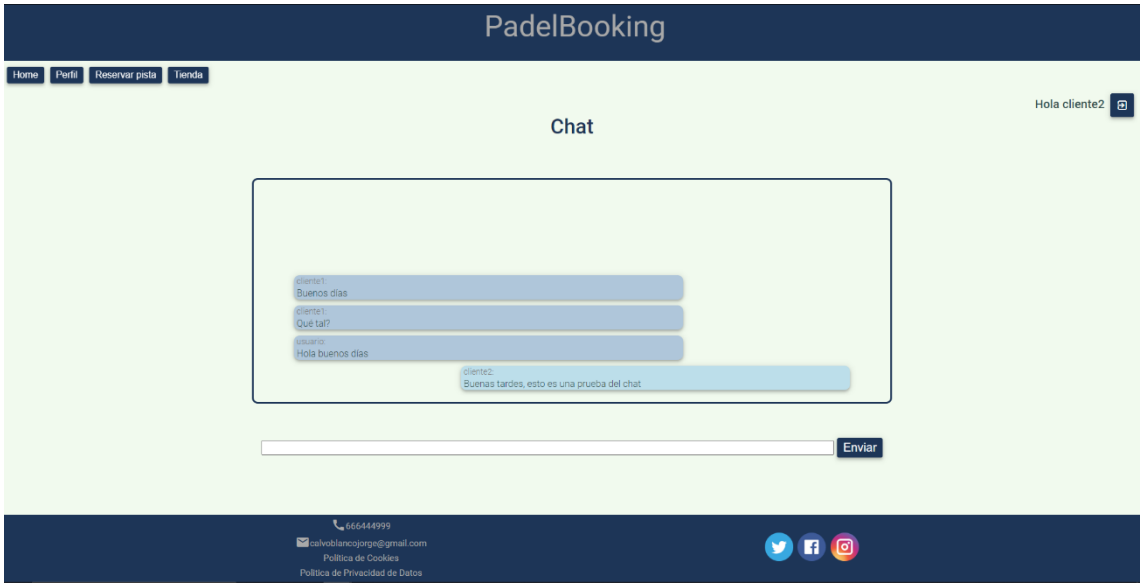

FIGURA 111: ENVIAR MENSAJE

8.2.3. Perfil: Usuario Administrador

En esta sección del manual detallaremos las opciones que tiene el Administrador dentro de nuestra aplicación.

En primer lugar, el Administrador se encuentra en la página de home, ya que al iniciar sesión es automáticamente redirigido a dicha página. Esta página es igual que la del Cliente salvo por ciertas diferencias:

- El Administrador no puede acceder a la parte de reserva de pistas ya que no es una funcionalidad de este tipo de usuario.
- Si el Administrador pulsa el botón junto a su nombre en la esquina superior derecha de la pantalla se le redigirá al Panel de Administración, en el cual están la mayoría de funcionalidades relativas al Administrador.
- El Administrador puede acceder a la tienda y visualizar los productos, pero no puede realizar un pedido.

Dicho esto, el Administrador puede acceder a la tienda (mismo proceso que el Cliente), acceder a su perfil y acceder al Panel de Administración.

FIGURA 112: PANTALLA PRINCIPAL DE ADMINISTRADOR

Acceder a su perfil: El Administrador accederá a su perfil de la misma manera que el Cliente, sin embargo las opciones que se le presentan no son las mismas, ya que únicamente tendrá opción a acceder al chat. Las funcionalidades en el chat serán las mismas que las que tiene el Cliente.

Acceder al Panel de Administración: Al pulsar el botón junto al nombre de usuario, el Administrador accederá al Panel de Administración, en el cual podrá gestionar las tablas de la aplicación.

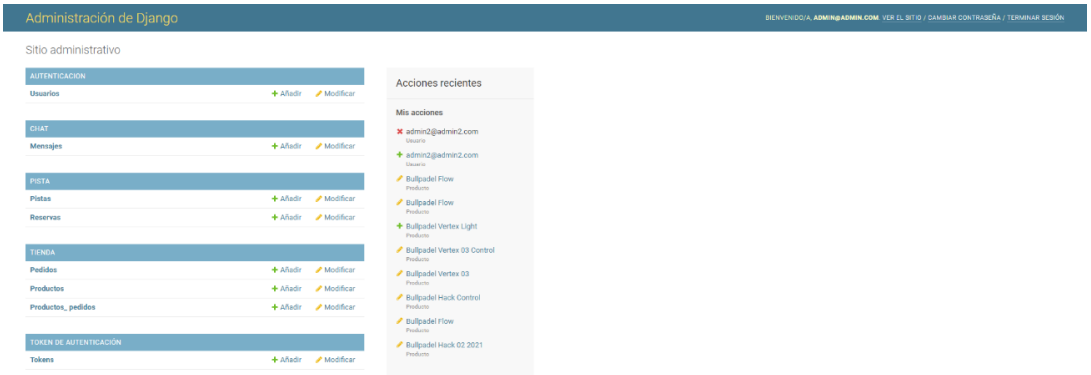

FIGURA 113: MODELOS DE LA APLICACIÓN

Aquí, tal y como se explica en el apartado de Análisis, el Administrador podrá realizar gestiones en los modelos de la aplicación:

- Usuarios: Si el Administrador selecciona este modelo, podrá eliminar una cuenta de usuario si considera que ha incumplido las políticas de la plataforma.

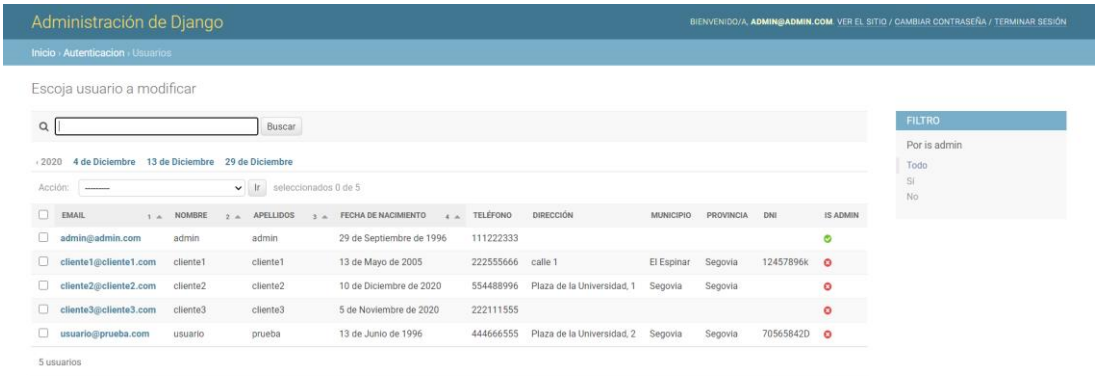

FIGURA 114: LISTADO DE USUARIOS

Para ello, deberá seleccionar el usuario en el listado y pulsar el botón Eliminar.

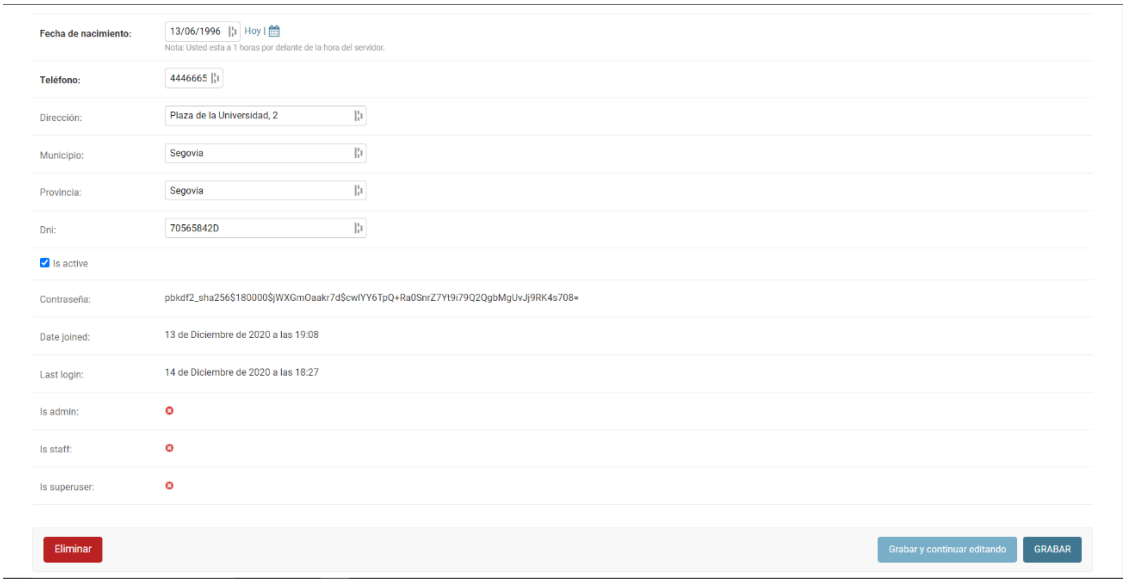

FIGURA 115: DETALLES DE USUARIO

Esta acción requerirá una confirmación, ya que al hacerlo se eliminarán todos los registros derivados de ese usuario, como los pedidos o reservas que pudiera haber realizado.

FIGURA 116: ELIMINAR USUARIO

- Pistas: Si el Administrador selecciona este modelo, podrá gestionar las pistas de la aplicación.

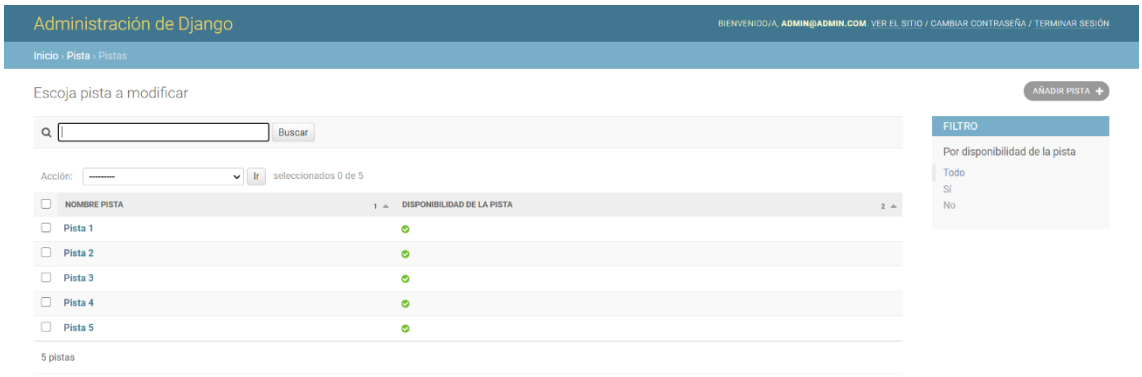

FIGURA 117: LISTADO DE PISTAS

Si el Administrador selecciona una pista del listado, podrá visualizar sus datos. Hecho esto, podrá modificar los datos de la pista introduciendo los nuevos datos y pulsando el botón Grabar.

FIGURA 118: DETALLES DE PISTA

También podrá eliminar la pista del sistema pulsando el botón Eliminar. El procedimiento seguido es el mismo que al eliminar un usuario.

FIGURA 119: ELIMINAR PISTA

Desde la página del listado de pistas también puede añadir una nueva pista al sistema introduciendo los datos de la nueva pista y pulsando el botón Grabar.

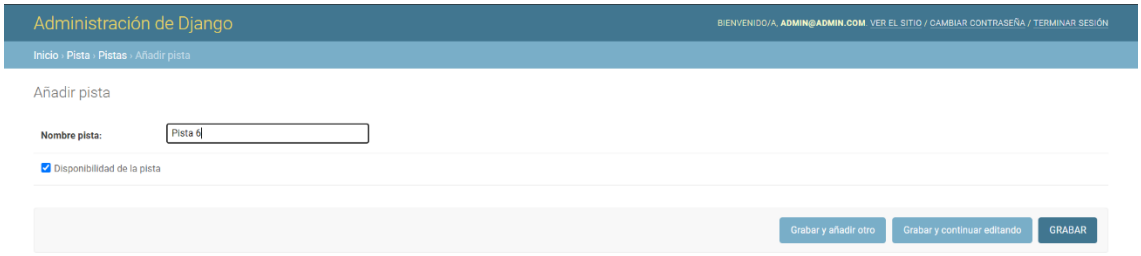

FIGURA 120: DAR DE ALTA UNA PISTA

- Pedidos: Si el Administrador selecciona este modelo podrá gestionar los pedidos de la aplicación. En primer lugar, aparecerán todos los pedidos realizados por los Clientes.

FIGURA 121: LISTADO DE PEDIDOS

Para cambiar el estado de un pedido, en primer lugar el Administrador debe seleccionar el pedido deseado del listado.

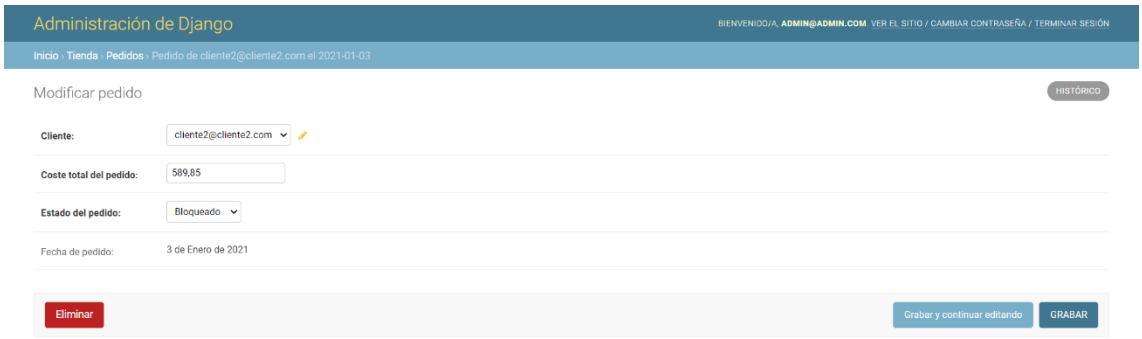

FIGURA 122: DETALLES DE PEDIDO

Una vez hecho esto, el Administrador cambiará el estado del pedido y pulsará el botón Grabar.

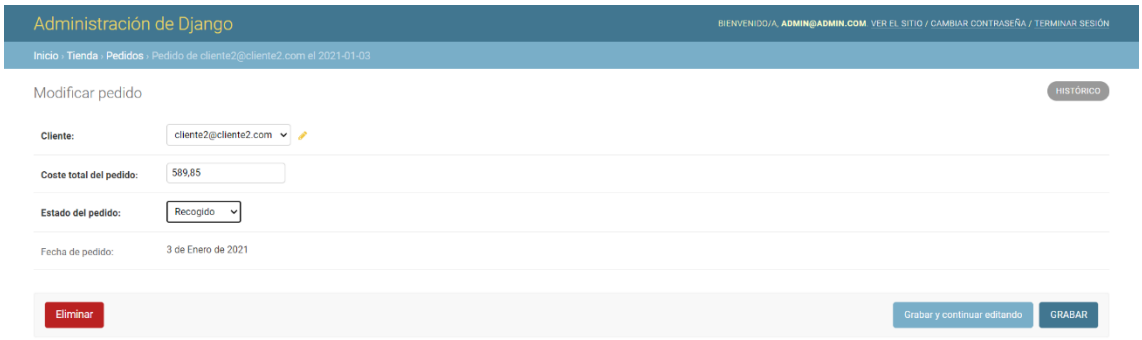

FIGURA 123: ESTADO DE PEDIDO MODIFICADO

- Productos: Si el Administrador selecciona este modelo podrá gestionar los productos de la tienda. En primer lugar, se le mostrarán todos los productos de la tienda.

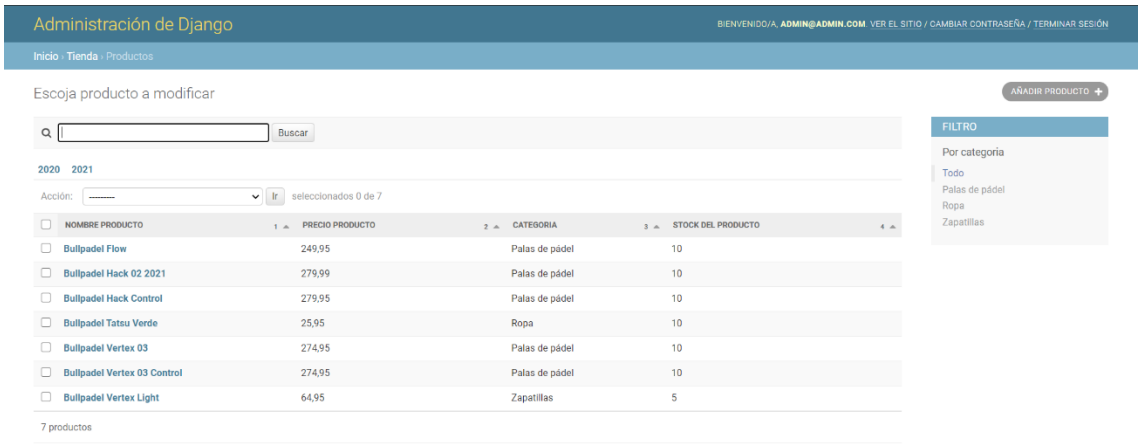

FIGURA 124: LISTADO DE PRODUCTOS

Si el Administrador selecciona un producto del listado, podrá visualizar sus datos. Hecho esto, podrá modificar los datos del producto introduciendo los nuevos datos y pulsando el botón Grabar.

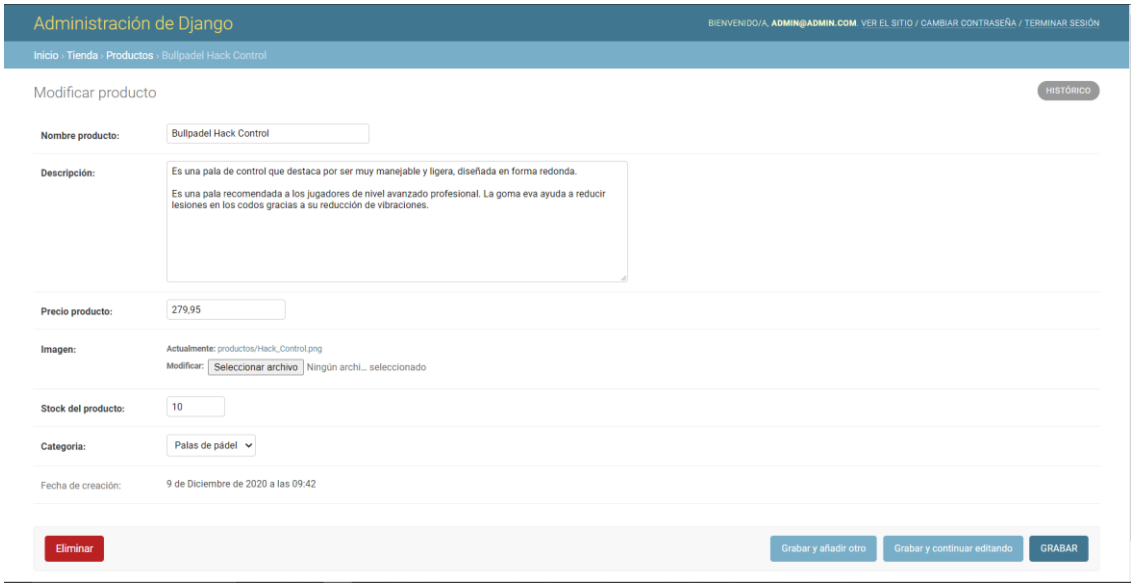

FIGURA 125: DETALLES DE PRODUCTO

También podrá eliminar el producto del sistema pulsando el botón Eliminar. El procedimiento seguido es el mismo que al eliminar una pista.

FIGURA 126: ELIMINAR PRODUCTO

Desde la página del listado de productos también puede añadir un nuevo producto al sistema introduciendo los datos del nuevo producto y pulsando el botón Grabar.

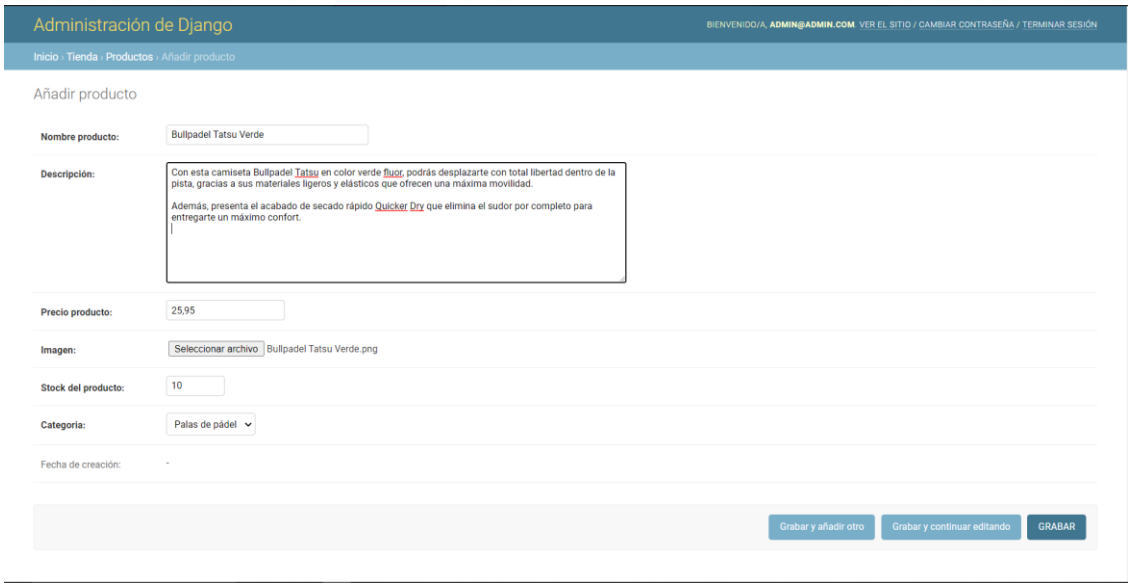

FIGURA 127: AÑADIR NUEVO PRODUCTO

9. CONCLUSIONES Y FUTURAS MEJORAS

9.1. Conclusiones

Este proyecto tenía dos objetivos iniciales.

El primero era desarrollar una aplicación web desde cero empleando tecnologías desconocidas para mí, y de esta manera ampliar mis conocimientos como desarrollador. Las tecnologías elegidas han sido Django y Angular, frameworks de los lenguajes Python y JavaScript, respectivamente. El principal motivo para la elección de estas tecnologías ha sido el de profundizar y ganar experiencia con lenguajes en los que ya tenía conocimientos básicos adquiridos durante el grado.

El segundo objetivo era el desarrollo de una aplicación que ayudase a digitalizar los procesos cotidianos de un club de pádel, como son las reservas de pistas, la planificación de partidos por parte de los usuarios (la cual suele hacerse por medios externos al club) y la venta de material deportivo.

Considero ambos objetivos cumplidos ya que, por un lado he adquirido conocimientos muy valiosos sobre Django y Angular que me servirán para el futuro y, por otro, considero que la aplicación cubre las necesidades normales de un club de pádel.

Sin embargo, considero que la aplicación tiene margen de mejora, por tanto en la siguiente sección se expondrán diversas funcionalidades que se han tenido que desestimar durante el desarrollo de este proyecto debido principalmente a la falta de tiempo.

9.2. Futuras mejoras

En este apartado se detallarán determinadas funcionalidades que se ha considerado añadir a la aplicación durante el desarrollo de este proyecto pero que por diversas razones han tenido que ser desestimadas.

- Sistema de matriculación y gestión de clases de pádel.
- Sistema de gestión de torneos entre los Clientes.
- Realización de una aplicación móvil o, en su defecto, modificar la aplicación para que sea responsive y se pueda visualizar de forma óptima en dispositivos móviles.
- Integración de un sistema de envío de pedidos a domicilio.
- Despliegue del Back-End de la aplicación en un host en la nube.

REFERENCIAS

Bibliografía

- Antonio Melé, Django 3 By Example, Tercera edición, Packt Publishing, 2020
- Nate Murray, Felipe Coury, Ari Lerner, Carlos Taborda. Ng-Book – The Complete Book on Angular, Revisión 74, Leanpub Publishing, 2019
- Carlos Azaustre, Aprendiendo JavaScript, Primera edición, Versión 1.2, 2016
- Mark Lutz, Learning Python, Quinta edición, Ed. O'Reilly, 2013

Webgrafía

- Formación en Django. Disponible en:
  https://www.udemy.com/course/curso-django-2-practico-desarrollo-web-python-3/
  Fecha de último acceso: 11 de marzo de 2020

- Formación en Django REST Framework. Disponible en:
  https://www.udemy.com/course/full-stack-development-web-app-mobile-app-back-end-api/
  Fecha de último acceso: 20 de marzo de 2020

- Formación en Angular. Disponible en:
  https://www.udemy.com/course/curso-de-angular-4-desde-cero-hasta-profesional/
  Fecha de último acceso: 2 de abril de 2020

- API de Django. Disponible en:
  https://docs.djangoproject.com/en/3.0/ref/
  Fecha de último acceso: 17 de noviembre de 2020

- API de Django REST Framework. Disponible en:
  https://www.django-rest-framework.org/
  Fecha de último acceso: 22 de septiembre de 2020

- API de Angular. Disponible en:
  https://angular.io/docs
  Fecha de último acceso: 9 de octubre de 2020

- API de PayPal. Disponible en:
  https://developer.paypal.com/home/
  Fecha de último acceso: 13 de noviembre de 2020

- Integración de PostgreSQL en Django. Disponible en:
  https://docs.djangoproject.com/en/3.0/ref/settings/#databases
  Fecha de último acceso: 5 de mayo de 2020

- Arquitectura de Django REST Framework, parte 1. Disponible en:
  https://content.breatheco.de/es/lesson/django-rest-framework
  Fecha de último acceso: 11 de agosto de 2020

- Arquitectura de Django REST Framework, parte 2. Disponible en:
  http://cam2149.blogspot.com/2012/06/arquitectura-del-django-framework.html
  Fecha de último acceso: 11 de agosto de 2020

- Patrón Modelo-Vista-Template, parte 1. Disponible en: https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv
  Fecha de último acceso: 9 de noviembre de 2020

- Patrón Modelo-Vista-Template, parte 2. Disponible en: https://docs.hektorprofe.net/django/web-personal/patron-mvt-modelo-vista-template/
  Fecha de último acceso: 9 de noviembre de 2020

- Django vs Flask. Disponible en: https://www.ilimit.com/blog/flask-vs-django/#:~:text=Ventajas%20de%20Django&text=Manejo%20de%20versiones%20que%20permite,un%20desarrollo%20%C3%A1gil%20y%20reutilizable.
  Fecha de último acceso: 19 de diciembre

- Personalizar el modelo de usuario para autenticación en Django. Disponible en: https://docs.djangoproject.com/en/3.0/topics/auth/customizing/#a-full-example
  Fecha de último acceso: 7 de mayo de 2020

- Personalizar el modelo de usuario para autenticación en Django, Parte 2. Disponible en: https://www.youtube.com/watch?v=eCeRC7E8Z7Y
  Fecha de último acceso: 7 de mayo de 2020

- Migrar un modelo personalizado de usuario en Django. Disponible en: https://django-authtools.readthedocs.io/en/latest/how-to/migrate-to-a-custom-user-model.html
  Fecha de último acceso: 11 de mayo de 2020

- Seguridad en Django. Disponible en: https://docs.djangoproject.com/en/3.1/topics/security/
  Fecha de último acceso: 23 de diciembre de 2020

- Recuperar registros relativos a un único usuario en Django REST Framework. Disponible en: https://stackoverflow.com/questions/48073471/django-rest-framework-get-data-based-on-current-userid-token
  Fecha de último acceso: 16 de junio de 2020

- Usar Datepicker en Angular 9. Disponible en: https://appdividend.com/2018/09/16/angular-datepicker-example-tutorial/
  Fecha de último acceso: 8 de julio de 2020

- Usar correctamente Angular Select. Disponible en: https://www.arquitecturajava.com/angular-select-y-todas-sus-opciones/
  Fecha de último acceso: 6 de julio de 2020

- Comunicación entre componentes en Angular. Disponible en: https://codingpotions.com/angular-comunicacion-componentes
  Fecha de último acceso: 25 de junio de 2020

- Uso de ChangeDetectorRef en Angular. Disponible en: https://angular.io/api/core/ChangeDetectorRef
  Fecha de último acceso: 6 de julio de 2020

- Añadir tiempo a una fecha de tipo string. Disponible en: https://stackoverflow.com/questions/17446466/add-15-minutes-to-string-in-javascript#:~:text=First%20convert%20your%20string%20to,e%20%3D%20new%20Date(d
  Fecha de último acceso: 8 de julio de 2020

- Snackbar en Angular Material. Disponible en: https://www.youtube.com/watch?v=jR3foHtmOvY&list=PLC3y8-rFHvwilEuCqFGTL5Gt5U6deIrsU&index=26&t=0s
  Fecha de último acceso: 3 de septiembre de 2020

- Subir y desplegar imágenes en Django. Disponible en: https://www.youtube.com/watch?v=8f9BWDzCIow
  Fecha de último acceso: 19 de agosto de 2020

- Aplicar filtros a *ngFor en Angular. Disponible en: https://www.it-swarm-es.tech/es/angular/como-aplicar-filtros-ngfor/1056308145/
  Fecha de último acceso: 9 de julio de 2020

- Usar Angular Material Dialog para el despliegue de ventanas emergentes. Disponible en: https://blog.angular-university.io/angular-material-dialog/
  Fecha de último acceso: 23 de noviembre de 2020

- Uso de asincronía para Angular Material Dialog. Disponible en: https://stackoverflow.com/questions/53283838/angular-6-material-await-until-mat-dialog-is-closed
  Fecha de último acceso: 23 de noviembre de 2020

- Integración de PayPal en Angular, parte 1. Disponible en: https://itelisoft.com/integracion-de-paypal-e-angular/
  Fecha de último acceso: 24 de septiembre de 2020

- Integración de PayPal en Angular, parte 2. Disponible en: https://www.youtube.com/watch?v=AtZGoueL4Vs
  Fecha de último acceso: 24 de septiembre de 2020

- Creación de paletas de colores. Disponible en:
  https://coolors.co/
  Fecha de último acceso: 18 de agosto de 2020

- Modificación de los campos en el Panel de Administración de Django. Disponible en:
  https://stackoverflow.com/questions/7341066/can-i-make-an-admin-field-not-required-in-django-without-creating-a-form
  Fecha de último acceso: 13 de mayo de 2020

- PayPal, deshabilitar métodos de pago. Disponible en:
  https://developer.paypal.com/docs/business/javascript-sdk/javascript-sdk-configuration/#disable-funding
  Fecha de último acceso: 2 de octubre de 2020

- Logo diseñado con DesignEvo. Disponible en:
  http://designevo.com/es/
  Fecha de último acceso: 22 de diciembre de 2020

- Iconos de la aplicación. Disponible en:
  https://icon-icons.com/
  Fecha de último acceso: 22 de diciembre de 2020

- Que es Heroku. Disponible en:
  https://videlcloud.wordpress.com/2018/12/22/que-es-heroku-para-que-sirve-ventajas-y-desventajas/
  Fecha de último acceso: 14 de enero de 2021

- Desplegar app Angular en Heroku. Disponible en:
  https://medium.com/@roliver_javier/como-desplegar-tu-aplicacion-de-angular-en-heroku-7b9941b6d39
  Fecha de último acceso: 11 de enero de 2021

- Documentación de Heroku. Disponible en:
  https://devcenter.heroku.com/categories/reference
  Fecha de último acceso: 13 de enero de 2021

- Documentación de pgbench. Disponible en:
  https://www.postgresql.org/docs/current/pgbench.html
  Fecha de último acceso: 19 de enero de 2021

- Guía de uso de pgbench. Disponible en:
  https://www.dba-ninja.com/2019/11/how-to-use-pgbench-for-postgresql-benchmark-testing.html
  Fecha de último acceso: 19 de enero de 2021