



TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE



Aplicación Web Java para la gestión de reservas en pisos u hoteles

Estudiante: Álvaro Pérez Pedreira

Dirección: José Losada Pérez

A Coruña, febrero de 2025.

Este trabajo está dedicado a mi familia, amigos y compañeros cuyo apoyo ha sido fundamental a lo largo de esta carrera.

Agradecimientos

Quiero agradecer en especial a mi tutor, José Losada Pérez, por toda la ayuda brindada y por su disponibilidad durante el transcurso de este trabajo. También quiero destacar y agradecer a mis padres y amigos que me han apoyado y ayudado a pulir el diseño de la aplicación.

Resumen

En la actualidad, el turismo vacacional está en constante crecimiento impulsado por una mayor accesibilidad a las opciones de transporte y alojamiento; gracias a este crecimiento se ha experimentado un incremento en la demanda de plataformas que centralicen la búsqueda y reserva de alojamientos.

Por ello, este trabajo que llevará por nombre DeepDive, tiene como objetivo el análisis, diseño e implementación de una aplicación web para gestionar ofertas de alojamiento en un mismo sitio web. La aplicación permitirá consultar las diferentes ofertas de alojamiento en cualquier ubicación geográfica y también permitirá realizar reservas, valorarlas y subir tu propio alojamiento.

El principal objetivo de DeepDive es proporcionar a los usuarios una herramienta que les permita de manera rápida, sencilla e intuitiva consultar, buscar y reservar alojamientos en cualquier ubicación geográfica. También permitirá a un usuario publicar su alojamiento y actualizarlo siempre que lo considere conveniente. DeepDive también proporcionará varias vistas específicas para consultar tanto tus reservas como tus alojamientos. Una vez se ha cumplido el periodo de una reserva, y con el fin de promover la calidad del servicio, los usuarios podrán escribir una valoración sobre su experiencia en dicho alojamiento. DeepDive busca que la aplicación sea inclusiva, integrando opciones de accesibilidad para que los usuarios puedan personalizar la interfaz a sus necesidades visuales. Además la aplicación está internacionalizada en varios idiomas.

La aplicación integra un servicio web real que permite acceder a una [API](#) con una amplia variedad de alojamientos, lo que permitirá a los usuarios acceder a una mayor variedad de resultados y opciones al buscar alojamientos, lo que facilita que las búsquedas sean más completas y precisas, optimizando el tiempo de los usuarios y facilitando la selección de su alojamiento ideal.

La aplicación se ha desarrollado utilizando Maven, Java y Spring para la arquitectura *backend* y React, Node, JavaScript y CSS para la arquitectura *frontend*.

Para el desarrollo del proyecto se ha empleado la metodología ágil Scrum, dividiendo el trabajo en tareas manejables que se ejecutarán de manera iterativa en distintos Sprints; una vez

finalizado el Sprint se validará que las funcionalidades implementadas cumplen con los requisitos establecidos en la planificación.

Abstract

Currently, vacation tourism is constantly growing, driven by greater accessibility to transportation and lodging options; as a result of this growth, there has been an increase in demand for platforms that centralize the search and booking of lodges.

Therefore, this work, named DeepDive, aims to analyze, design, and implement a web application to manage lodge offers on a single website. The application will allow users to search for different lodge offers in any geographic location, make bookings, rate them, and upload their own lodges.

The main objective of DeepDive is to provide users with a tool that allows them to quickly, easily, and intuitively search, browse, and book lodges in any geographic location. It will also allow a user to publish their lodge and update it whenever they deem appropriate. DeepDive will also provide several specific views to consult both your bookings and your lodges. Once a booking period has ended, and in order to promote service quality, users will be able to write a review about their experience at the lodge. DeepDive aims for the application to be inclusive, integrating accessibility options so that users can customize the interface to their visual needs. Additionally, the application is internationalized in multiple languages.

The application integrates a real web service that allows access to an [API](#) with a wide variety of lodges, enabling users to access a greater variety of results and options when searching for lodges, which makes searches more complete and accurate, optimizing users' time and facilitating the selection of their ideal lodge.

The application has been developed using Maven, Java, and Spring for the *backend* architecture, and React, Node, JavaScript, and CSS for the *frontend* architecture.

For the project's development, the agile Scrum methodology has been employed, dividing the work into manageable tasks that will be executed iteratively in different Sprints; once the Sprint is completed, it will be validated that the implemented features meet the requirements

set in the planning.

Palabras clave:

- Alojamiento
- Reserva
- React
- JavaScript
- CSS
- SpringBoot

Keywords:

- Lodge
- Booking
- React
- JavaScript
- CSS
- SpringBoot

Índice general

1	Introducción	1
1.1	Contexto y motivaciones	1
1.2	Descripción de objetivos	2
2	Estado del arte	4
2.1	Alternativas clásicas	4
2.2	Alternativas modernas	5
3	Tecnologías, lenguajes y herramientas	9
3.1	Tecnologías y lenguajes	9
3.1.1	Lado servidor (<i>Back-end</i>)	9
3.1.2	Lado cliente (<i>Front-end</i>)	12
3.2	Herramientas	14
4	Introducción al desarrollo	17
4.1	Metodologías	17
4.1.1	Metodologías tradicionales	17
4.1.2	Metodologías ágiles	18
4.2	Scrum	18
4.2.1	Roles	18
4.2.2	Sprint	20
4.2.3	Artefactos	21
4.2.4	Integración de Scrum en el proyecto	22
5	Viabilidad	23
5.1	Viabilidad técnica	23
5.2	Viabilidad económica	24
5.2.1	Recursos humanos	24

5.2.2	Recursos técnicos	25
5.2.3	Recursos materiales	25
5.2.4	Costes totales	25
5.3	Planificación de <i>Sprints</i> en el proyecto	26
5.3.1	Sprint 1 - Puesta en marcha (05/05/2024 - 18/05/2024)	26
5.3.2	Sprint 2 - Gestión de usuarios (19/05/2024 - 29/06/2024)	27
5.3.3	Sprint 3 - Gestión de alojamientos (30/06/2024 - 27/07/2024)	27
5.3.4	Sprint 4 - Gestión de listados y búsquedas (28/07/2024 - 24/08/2024)	28
5.3.5	Sprint 5 - Integración con un servicio web externo (25/08/2024 - 28/09/2024)	28
5.3.6	Sprint 6 - Gestión de reservas (29/09/2024 - 09/11/2024)	28
5.3.7	Sprint 7 - Gestión de comentarios (10/11/2024 - 23/11/2024)	29
5.3.8	Sprint 8 - Redacción de la memoria (24/11/2024 - 22/12/2024)	29
5.4	Comparativa de tiempo estimado y tiempo real	29
6	Ánáisis	32
6.1	Actores	32
6.2	Requisitos	34
6.2.1	Requisitos funcionales	34
6.2.2	Requisitos no funcionales	35
6.3	Historias de usuario	35
7	Diseño	46
7.1	Arquitectura general	46
7.2	Patrones y principios	47
7.2.1	Diseño por capas	47
7.2.2	Uso de DTOs	48
7.2.3	Repositorios	48
7.2.4	Inyección de dependencias	48
7.2.5	Principio de encapsulación	48
7.2.6	Principio de inversión de la dependencia	48
7.2.7	Principio de segregación de interfaces	48
7.2.8	Principio de responsabilidad única	49
7.2.9	Principio DRY	49
7.2.10	Principio KISS	49
7.2.11	Principio YAGNI	49
7.3	Arquitectura del subsistema servidor	49
7.3.1	Entidades persistentes	50
7.3.2	Capa de repositorios	53

7.3.3	Capa de servicios	56
7.3.4	Capa de controladores	59
7.4	Arquitectura del subsistema cliente	63
8	Implementación y pruebas	67
8.1	Subsistema servidor	67
8.1.1	Modelo de datos	67
8.1.2	Jerarquía de paquetes	69
8.2	Subsistema cliente	70
8.2.1	Jerarquía de paquetes	70
8.3	Compilación y puesta en funcionamiento	73
8.3.1	Servidor	73
8.3.2	Cliente	74
8.4	Pruebas	74
8.4.1	Pruebas de integración	74
8.4.2	Pruebas sobre los controladores	76
8.4.3	Pruebas de aceptación	76
9	Conclusiones y futuras líneas de trabajo	78
9.1	Conclusiones	78
9.2	Futuras líneas de trabajo	79
A	Material adicional	81
A.1	Gestión de usuarios	81
A.1.1	Autenticación y registro de usuarios	81
A.1.2	Selección de tema e idioma	83
A.1.3	Desplegable de usuario	83
A.1.4	Visualización y actualización del perfil	85
A.1.5	Cambio de contraseña	85
A.1.6	Cierre de sesión	86
A.1.7	Contactar con el servicio técnico	86
A.1.8	Detalles del perfil de un usuario	87
A.2	Gestión de alojamientos	87
A.2.1	Página principal de la aplicación	87
A.2.2	Búsqueda de alojamientos	89
A.2.3	Detalles de un alojamiento de DeepDive	91
A.2.4	Detalles de un alojamiento de Booking	92
A.2.5	Publicación de un alojamiento	93

A.2.6 Consulta los alojamientos de un usuario	95
A.2.7 Actualización de los detalles de un alojamiento	96
A.2.8 Abrir y cerrar un alojamiento	96
A.3 Gestión de reservas	96
A.3.1 Consultar la disponibilidad de un alojamiento	96
A.3.2 Realizar una reserva	97
A.3.3 Consultar tus reservas	99
A.3.4 Cancelar una reserva	100
A.4 Gestión de comentarios	100
A.4.1 Valorar una reserva	100
A.4.2 Ver los comentarios de un alojamiento	100
A.5 Módulo de administrador	101
A.5.1 Desplegable de usuario del administrador	101
A.5.2 Bloquear un usuario	102
A.5.3 Consultar usuarios bloqueados	104
A.5.4 Desbloquear a un usuario	104
A.5.5 Bloquear un alojamiento	105
A.5.6 Consultar alojamientos bloqueados	105
A.5.7 Desbloquear un alojamiento	106
A.5.8 Bloquear un comentario	106
Lista de acrónimos	107
Glosario	109
Bibliografía	110

Índice de figuras

2.1	Catálogos de viajes	5
2.2	Página principal de Booking	5
2.3	Detalles de un alojamiento de Booking	6
2.4	Página principal de Airbnb	7
2.5	Detalles de un alojamiento de Airbnb	7
4.1	Roles en Scrum	19
4.2	Flujo básico en un proyecto <i>Scrum</i>	22
7.1	Arquitectura general del sistema	47
7.2	Arquitectura general por capas del subsistema servidor	50
7.3	Diagrama de entidades	51
7.4	Diagrama Entidad/Relación	53
7.5	Ejemplo de herencia de <i>JpaRepository</i>	54
7.6	Lista de repositorios	55
7.7	Diagrama de servicios	57
7.8	Arquitectura general del subsistema cliente	63
7.9	API de Booking	65
8.1	Modelo de datos	68
8.2	Jerarquía de paquetes del subsistema servidor	69
8.3	Jerarquía de paquetes del subsistema cliente	71
8.4	Archivo index.jsx	73
8.5	Diagrama de pruebas de integración	75
8.6	Cobertura de las pruebas de integración	76
8.7	Cobertura de las pruebas de integración	76
A.1	Autenticación	82

A.2	Formulario de registro	82
A.3	Página principal de la aplicación	83
A.4	Selección de tema e idioma	83
A.5	Desplegable de usuario	84
A.6	Desplegable de usuario	84
A.7	Visualizar y actualizar perfil	85
A.8	Cambio de contraseña	86
A.9	Contactar con el servicio técnico	87
A.10	Detalles de un usuario	87
A.11	Página principal de la aplicación	88
A.12	Resultado de una búsqueda de alojamientos	88
A.13	Página principal de la aplicación	89
A.14	Barra de búsqueda	89
A.15	Ejemplo de búsqueda	90
A.16	Ejemplo de búsqueda	90
A.17	Ejemplo de búsqueda	91
A.18	Detalles de un alojamiento de DeepDive	91
A.19	Detalles de un alojamiento de DeepDive	92
A.20	Detalles de un alojamiento de Booking	93
A.21	Detalles de un alojamiento de Booking	93
A.22	Registra un alojamiento	94
A.23	Registra un alojamiento	94
A.24	Registra un alojamiento	95
A.25	Consulta los alojamientos de un usuario	95
A.26	Consulta la disponibilidad de un alojamiento	97
A.27	Reservar un alojamiento	98
A.28	Reservar un alojamiento	98
A.29	Método de pago	99
A.30	Método de pago	99
A.31	Consultar las reservas de un usuario	99
A.32	Valorar una reserva	100
A.33	Ver comentarios de un alojamiento	101
A.34	Ver comentarios de un alojamiento	101
A.35	Desplegable de usuario administrador	102
A.36	Bloquear a un usuario	103
A.37	Error de acceso de un usuario bloqueado	104
A.38	Lista de usuarios bloqueados	104

A.39 Bloquear un alojamiento	105
A.40 Lista de alojamientos bloqueados	105
A.41 Bloquear un comentario	106

Índice de cuadros

5.1	Costes estimados de recursos humanos	24
5.2	Costes reales de recursos humanos	24
5.3	Costes de recursos técnicos	25
5.4	Costes de recursos técnicos	25
5.5	Costes totales estimados	26
5.6	Costes totales reales	26
5.7	Comparación de tiempos	31
6.1	HU-<01>: Registrarse.	36
6.2	HU-<02>: Iniciar sesión.	37
6.3	HU-<03>: Cerrar sesión.	37
6.4	HU-<04>: Ver perfil.	37
6.5	HU-<05>: Editar perfil.	38
6.6	HU-<06>: Establecer un avatar.	38
6.7	HU-<07>: Bloquear a un usuario.	38
6.8	HU-<08>: Desbloquear a un usuario.	39
6.9	HU-<09>: Consultar los usuarios bloqueados.	39
6.10	HU-<10>: Registrar un alojamiento.	39
6.11	HU-<11>: Consultar los detalles de un alojamiento.	40
6.12	HU-<12>: Visualizar tus alojamientos.	40
6.13	HU-<13>: Editar los detalles de tu alojamiento.	40
6.14	HU-<14>: Cerrar tu alojamiento.	41
6.15	HU-<15>: Abrir tu alojamiento.	41
6.16	HU-<16>: Consultar la disponibilidad de un alojamiento.	41
6.17	HU-<17>: Realizar consultas sobre los alojamientos.	42
6.18	HU-<18>: Bloquear un alojamiento.	42
6.19	HU-<19>: Desbloquear un alojamiento.	42

6.20 HU-<20>: Consultar los alojamientos bloqueados.	43
6.21 HU-<21>: Realizar una reserva.	43
6.22 HU-<22>: Consultar tus reservas.	43
6.23 HU-<23>: Cancelar una reserva.	44
6.24 HU-<24>: Valorar una reserva.	44
6.25 HU-<25>: Bloquear una valoración.	44
6.26 HU-<26>: Seleccionar el tema.	45
6.27 HU-<27>: Seleccionar el idioma.	45
6.28 HU-<28>: Contactar con el servicio técnico.	45

Capítulo 1

Introducción

1.1 Contexto y motivaciones

En las últimas décadas, el turismo vacacional ha experimentado un crecimiento exponencial impulsado por la creciente accesibilidad a diversos medios de transporte y una amplia oferta de alojamientos [1]. Además, varios estudios [2] sugieren que el turismo puede tener beneficios significativos para la salud, tal como la reducción de estrés, la mejora del bienestar emocional y la renovación física. Actividades como el turismo de bienestar, que incluye retiros de yoga, meditación o spás, están diseñadas para ayudar a los viajeros a mejorar tanto su salud física como mental. Esto no solo ofrece una pausa mental, sino que puede contribuir a una mejor calidad de vida a largo plazo [3].

Con el paso del tiempo, y el auge de plataformas como Booking, Trivago, Airbnb, TripAdvisor... Es evidente que el turismo vacacional ha experimentado un crecimiento significativo. Gracias a la accesibilidad y facilidad de estas aplicaciones para encontrar y reservar alojamientos, más personas se han animado a viajar, lo que ha facilitado la expansión del turismo a destinos previamente menos conocidos o visitados, de esta idea salen nuevos términos como el turismo rural [4], que consiste en ofrecer experiencias auténticas en entornos rurales y naturales, permitiendo a los viajeros desconectar del ritmo acelerado de la ciudad. Todo esto ha abierto un abanico más amplio de posibilidades a la hora de viajar, permitiendo que cada persona pueda personalizar su viaje de acuerdo a sus necesidades e intereses, lo que les permitirá disfrutar de una experiencia enriquecedora.

Este proyecto consiste en diseñar una plataforma en línea enfocada en la gestión y reserva de alojamientos turísticos. La plataforma tiene como objetivo simplificar la búsqueda, reserva y gestión de propiedades, brindando a los usuarios una amplia variedad de opciones adaptadas a sus preferencias y necesidades. En este proyecto también se han intentado abordar algunos de los problemas de las grandes plataformas ya existentes, como la prevención del

uso de alojamientos fraudulentos. Para ello se proporcionará al perfil del administrador las herramientas necesarias para realizar un control exhaustivo de los alojamientos disponibles, permitiéndole suspender tanto a usuarios como alojamientos que incumplan la normativa vigente de la aplicación. Esto tiene el objetivo de garantizar que los usuarios, tanto anfitriones como huéspedes, tengan una experiencia óptima al utilizar la plataforma.

El origen de este proyecto surge de una experiencia personal cercana del alumno, en la que un amigo cercano sufrió una mala experiencia al hospedarse en un alojamiento fraudulento. El alojamiento presentaba notorias deficiencias en comparación con lo acordado en la plataforma, lo que generó la motivación para crear una plataforma que asegure la fiabilidad y calidad de los alojamientos disponibles.

1.2 Descripción de objetivos

El principal objetivo del proyecto es el análisis, diseño e implementación de una aplicación web que permitirá proporcionar a todas las personas que deseen viajar y disfrutar de una experiencia enriquecedora una herramienta que les permita consultar, buscar y reservar uno o varios alojamientos de forma rápida y sencilla, eliminando la necesidad de invertir mucho tiempo en estas tareas; lo que facilita que los usuarios puedan centrarse más en la experiencia del viaje que en las gestionan previas. Las principales funcionalidades de la aplicación son las siguientes:

- Gestión de usuarios: la aplicación permitirá el registro de usuarios, inicio de sesión, cierre de sesión y actualizar perfil. En la aplicación habrá los siguientes tipos de usuarios:
 - Usuario registrado: usuario que se ha registrado en la aplicación.
 - Usuario no registrado: usuario que necesita iniciar sesión o registrarse para entrar a la aplicación.
 - Usuario administrador: usuario encargado de la gestión de los diferentes elementos que forman parte de la aplicación (como usuarios, alojamientos o comentarios).
- Gestión de alojamientos: la aplicación permitirá a los usuarios registrados añadir un alojamiento, ver los detalles de un alojamiento, modificar tu alojamiento y cerrarlo (temporal o permanentemente).
- Búsqueda de alojamientos: la aplicación permitirá buscar alojamientos según diversos parámetros.
- Gestión de reservas: la aplicación permitirá reservar un alojamiento, consultar las reservas realizadas y cancelarlas si se hace con cierta antelación.

- Gestión de comentarios: los usuarios podrán comentar en alojamientos en los que ya se hayan hospedado para dejar su comentario y valoración.
- Integración de la aplicación con un servicio web real: la aplicación se integrará con una [API](#) que permite obtener alojamientos según diferentes parámetros de búsqueda, como ubicación, fechas de la estancia o número de adultos, niños o habitaciones.
- Funcionalidades relacionadas con requisitos no funcionales.
 - Internacionalización: la aplicación contará con un sistema de internacionalización en tres idiomas (castellano, inglés y francés) permitiendo que los usuarios elijan el idioma que mejor se adapte a sus necesidades. En un futuro, se podrían añadir más idiomas de una manera sencilla.
 - Personalización de los colores de la interfaz de usuario: la aplicación también cuenta con un sistema de selección de tema claro u oscuro, lo que permite a los usuarios personalizar la interfaz según sus preferencias y condiciones de iluminación, mejorando así su experiencia visual.

Capítulo 2

Estado del arte

En la actualidad, gestionar adecuadamente la búsqueda de un alojamiento es fundamental, ya que tiene un impacto directo en la satisfacción del viajero y en el éxito general del viaje, al garantizar que se cumplan sus expectativas y necesidades.

2.1 Alternativas clásicas

Tradicionalmente, los viajes se organizaban de manera mucho más manual y limitada en comparación a las soluciones digitales actuales, como se puede ver en la figura 2.1. Los viajeros solían acudir a agencias de viajes tradicionales, donde los agentes le proporcionaban información a través de folletos o catálogos impresos, éstos contenían detalles sobre destinos, actividades, precios y paquetes turísticos.

Para muchos, este era el único medio para acceder a información confiable sobre los posibles destinos y opciones de alojamiento. El proceso consistía en reuniones o consultas periódicas para recopilar dicha información. Las reservas se hacían mediante llamadas telefónicas al hotel o a través de la agencia, lo que añadía una capa de complejidad y lentitud en el proceso.

Este método es más lento y menos flexible que las opciones actuales ya que no permitía comparar fácilmente opciones ni personalizar el viaje según las preferencias personales del viajero. Además, la información disponible estaba limitada a lo que las agencias ofrecían y podría no ajustarse a las necesidades o gustos de los viajeros.



Figura 2.1: Catálogos de viajes

2.2 Alternativas modernas

Actualmente, existen alternativas en el ámbito de las plataformas para la búsqueda y reserva de alojamientos. Uno de los ejemplos más conocidos es Booking [5] 2.2. Esta plataforma permite a los usuarios buscar, comparar y reservar una amplia variedad de alojamientos, incluidos hoteles, apartamentos, casas vacacionales y mucho más como vuelos y alquileres de coche.

A screenshot of the main page of the Booking.com website. The top navigation bar includes links for 'Alquiler de apartamentos', 'Vuelos', 'Vuelo + Hotel', 'Alquiler de coches', 'Actividades', and 'Taxis aeropuerto'. The main header reads 'Encuentra tu próxima estancia' with the sub-instruction 'Busca ofertas en hoteles, casas y mucho más...'. Below this is a search bar with fields for '¿Adónde vas?', 'Fecha de entrada...', 'Fecha de salida...', '2 adultos - 0 niños - 1 habitación', and a 'Buscar' button. There is also a checkbox for 'Busco vuelos'. The page features sections for 'Ofertas' (Promotions, discounts and special offers for you) and 'Inspírate para tu próximo viaje' (Inspire yourself for your next trip), showing images of a plane, a couple in a boat, and snowy mountains.

Figura 2.2: Página principal de Booking

Sin embargo, a pesar de sus amplias funcionalidades, Booking ha sido objeto de críticas, especialmente por su interfaz, que muchos consideran poco intuitiva. Aunque la plataforma incluye una amplia variedad de opciones, este exceso puede resultar útil en algunos casos, pero a la vez puede hacer que la experiencia de navegación se sienta sobrecargada y menos eficiente, como se puede ver en la figura 2.3. Además, su enfoque en ofrecer una amplia gama de productos puede comprometer la calidad del control sobre los alojamientos disponibles, lo que permite que algunos puedan no cumplir con las expectativas de los usuarios, incluso existiendo casos de alojamientos fraudulentos que podrían pasar desapercibidos.

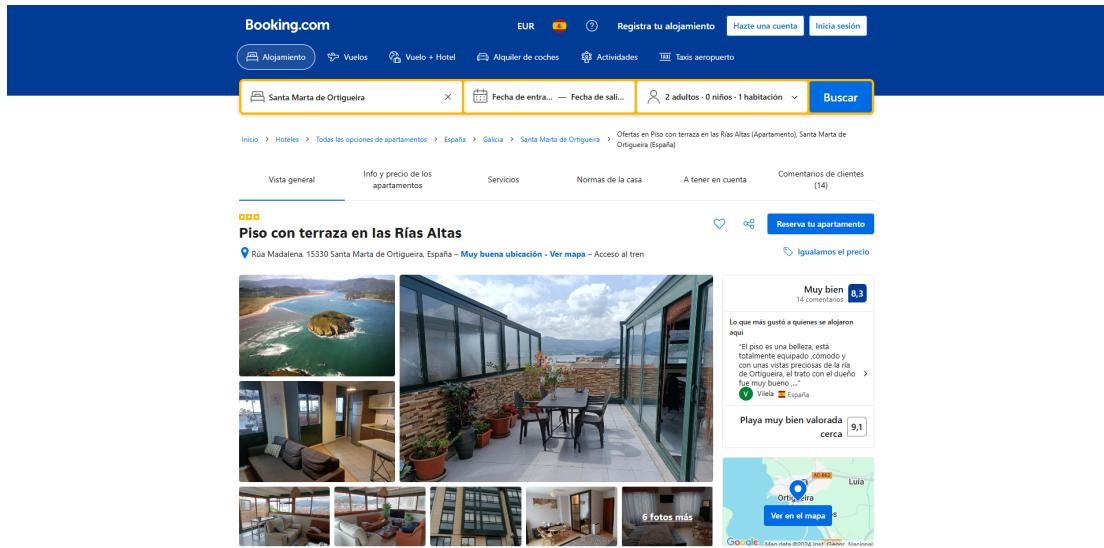


Figura 2.3: Detalles de un alojamiento de Booking

Además de Booking, también existen otras opciones como Airbnb [6], que ofrece una interfaz más acorde con las necesidades del usuario moderno, como se puede ver en la figura 2.4.

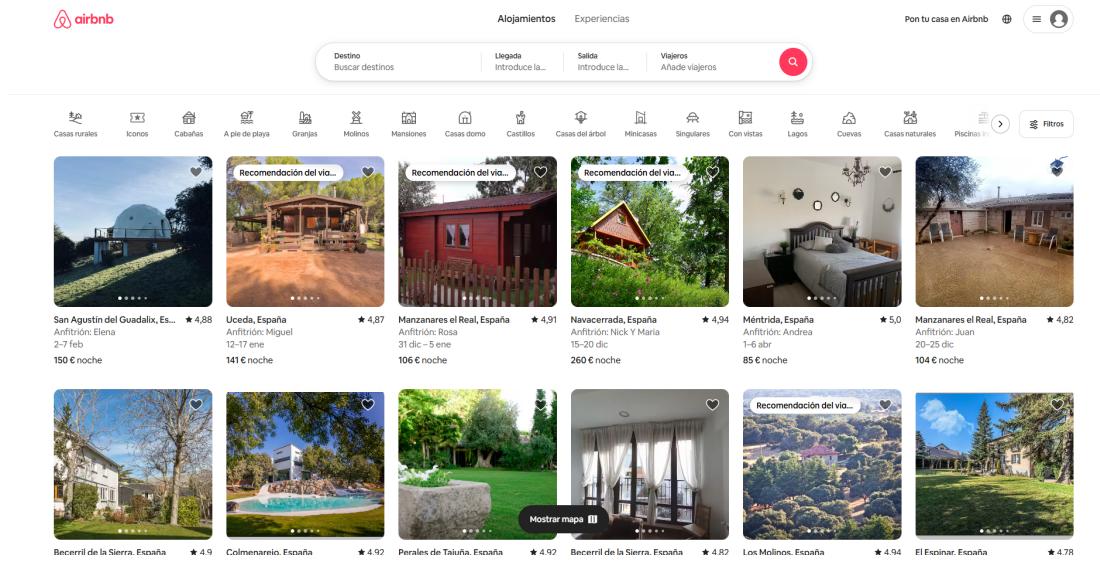


Figura 2.4: Página principal de Airbnb

También a diferencia de otras plataformas como Booking, Airbnb se centra exclusivamente en los alojamientos, lo que le permite ofrecer experiencias únicas y personalizadas 2.5. También permite una comunicación directa con los anfitriones antes y después de la reserva, lo que mejora la experiencia del usuario y proporciona un nivel adicional de seguridad. Sin embargo, al igual que otras plataformas, Airbnb ha enfrentado críticas por problemas relacionados con la veracidad de algunos anuncios y la calidad de los alojamientos, y en algunos casos, se han reportado situaciones de fraude.

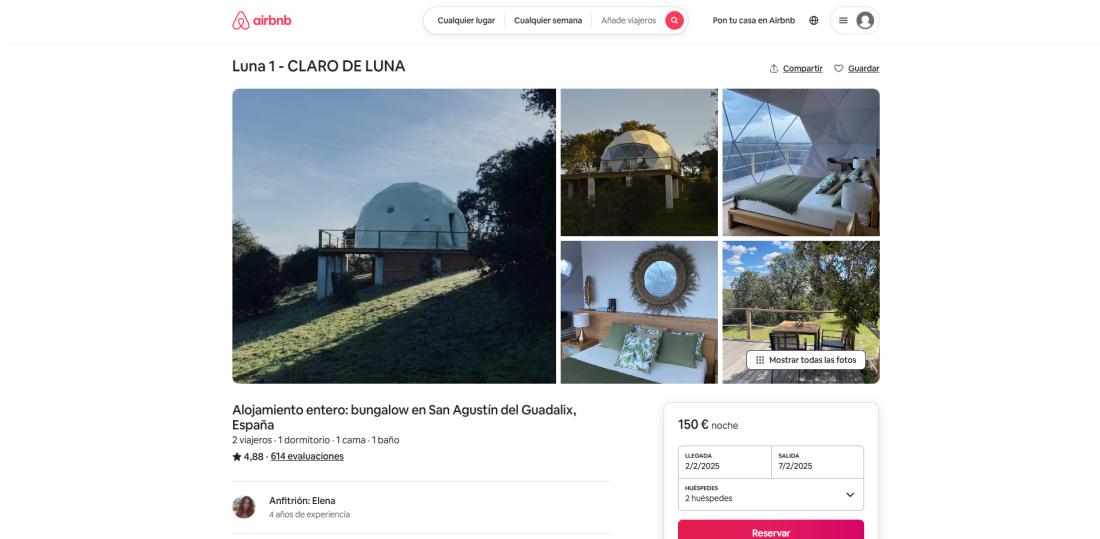


Figura 2.5: Detalles de un alojamiento de Airbnb

Analizando las opciones existentes, se puede concluir que es necesario implementar una solución que permita gestionar alojamientos y reservas de manera eficiente, garantizando al usuario una experiencia satisfactoria. La interfaz de DeepDive se desarrollará para ser sencilla e intuitiva, que no sobrecargue al usuario con demasiada información, permitiéndole realizar las gestiones de forma rápida y sencilla, de manera que no se convierta en una parte tediosa de la planificación de su viaje.

Capítulo 3

Tecnologías, lenguajes y herramientas

EN este capítulo se describirán los lenguajes, tecnologías y herramientas utilizados durante el desarrollo del proyecto en cada uno de los subsistemas.

3.1 Tecnologías y lenguajes

A continuación, se describirán las tecnologías y lenguajes empleados en cada etapa del desarrollo. La aplicación se dividirá en dos partes: lado cliente y lado servidor.

3.1.1 Lado servidor (*Back-end*)

Para la implementación del subsistema servidor, se empleó el lenguaje de programación Java junto con Springboot y el framework Hibernate.

Java

Java [7, 8] es un lenguaje de programación orientado a objetos de propósito general, diseñado para que los programas escritos en él puedan ejecutarse de manera independiente de la plataforma y el tipo de *hardware*. Esta característica permite que los programas escritos en Java puedan ejecutarse en cualquier dispositivo sin importar su sistema operativo. La filosofía principal de Java es el principio de "escribir una vez, ejecutar en cualquier lugar", lo que lo hace especialmente versátil en el desarrollo de aplicaciones multiplataforma. Una de las ventajas más importantes de Java es su recolector de basura automático, lo que permite liberar al desarrollador de la responsabilidad de gestionar la memoria, liberando recursos de objetos que ya no están en uso y reduciendo significativamente las fugas de memoria. La elección de Java como lenguaje para este proyecto se debe a la experiencia del autor en este lenguaje, que

es el encargado de implementar y diseñar todas las funcionalidades de la lógica de negocio. Además, Java dispone de una amplia variedad de **framework(s)** y materiales de apoyo, lo que facilita el desarrollo y la integración de nuevas tecnologías.

Springboot

SpringBoot [9] es un proyecto del ecosistema Spring Framework que simplifica el desarrollo de aplicaciones Java al automatizar gran parte de la configuración. Esta librería permite a los desarrolladores centrarse en la lógica de negocio, eliminando la necesidad de configurar manualmente las dependencias y servidores de aplicaciones. Con Spring Boot, se pueden usar servidores embebidos como Tomcat o Jetty, facilitando el despliegue de aplicaciones autónomas. Además, se puede integrar con herramientas de gestión de dependencias como Maven o Gradle para simplificar la incorporación de librerías de terceros. En este proyecto, Spring Boot se emplea para agilizar la configuración y el despliegue. Junto con él, se utiliza Spring Security [10], un **framework** que proporciona mecanismos de autenticación y control de acceso para proteger aplicaciones, especialmente en entornos web y servicios REST, lo que lo convierte en una herramienta clave para asegurar la aplicación.

MySQL

MySQL [11] es un sistema de gestión de bases de datos relacional (**SGBD**) ampliamente utilizado, especialmente en proyectos de tamaño pequeño a mediano, gracias a su destacada eficiencia en el manejo de cargas tanto ligeras como intensivas. Este gestor se caracteriza por su robustez, seguridad y escalabilidad, lo que lo convierte en una opción ideal para aplicaciones que requieren un manejo eficiente de grandes volúmenes de datos.

Debido a estas cualidades, MySQL ha sido seleccionado como el sistema encargado de gestionar la persistencia de los datos en la aplicación, proporcionando una plataforma confiable para el almacenamiento y acceso rápido a la información.

Hibernate

Hibernate [12] es un **framework ORM** para Java que permite mapear objetos a bases de datos. Su principal función es simplificar la interacción entre las aplicaciones Java y las bases de datos relacionales, evitando que los desarrolladores tengan que escribir consultas **SQL** manualmente para manejar la correspondencia entre los objetos Java y las tablas de la base de datos. Hibernate se encarga de realizar este mapeo de forma automática, permitiendo que los objetos Java se almacenen y recuperen fácilmente de la base de datos sin necesidad de escribir consultas **SQL**. Una de las principales ventajas de Hibernate es la alta cohesión con el ecosistema de Spring, un **framework** muy utilizado para el desarrollo de aplicaciones Java,

lo que asegura una integración fluida y eficiente entre ambos. Gracias a esta compatibilidad, Hibernate fue seleccionado para gestionar el acceso a los datos de este proyecto.

Maven

Maven [13] es una herramienta de gestión de la construcción de proyectos Java que facilita la configuración, administración de dependencias y la gestión del ciclo de vida de desarrollo. Su principal ventaja es que gestiona tareas recurrentes como la compilación, el empaquetado y la implementación, lo que mejora la productividad y eficiencia del equipo de desarrollo. Maven utiliza un archivo de configuración llamado *pom.xml*, donde se definen las dependencias, *plugins* y configuraciones necesarias para construir el proyecto de manera coherente y reproducible. En este proyecto, Maven se empleó para gestionar las dependencias del *Back-end*, tales como Spring Boot y Hibernate, asegurando que todas las librerías necesarias estén correctamente descargadas e integradas en el entorno de desarrollo.

JUnit

JUnit [14] es *framework* de pruebas unitarias para Java. Su principal objetivo es facilitar la creación y ejecución de pruebas automatizadas que validen el funcionamiento de los componentes del software. JUnit permite comprobar que las distintas partes de una aplicación, como métodos y clases, se comportan de acuerdo con lo esperado. Para ello, ofrece métodos de aserción que ayudan a verificar si los resultados de las pruebas coinciden con los valores esperados. La característica principal de JUnit es su capacidad para ejecutar las pruebas de manera automática y reportar los resultados, lo que permite detectar posibles errores en el código en una fase temprana del ciclo de desarrollo, esto es especialmente útil en los proyectos ya que asegura la calidad del software a medida que se van implementando las funcionalidades. En este proyecto, se utilizará JUnit para implementar las pruebas unitarias del *Back-end*, asegurando así que cada componente del sistema funcione correctamente y de acuerdo a los requisitos especificados.

JWT

JWT [15] (**JSON** Web Token) es un estándar abierto que define un formato compacto y seguro para transmitir información entre dos partes como un objeto **JSON**. Este formato es utilizado principalmente para la autenticación y autorización en aplicaciones web y servicios **API**. La información contenida en el JWT puede ser verificada y confiable porque está firmada digitalmente usando un algoritmo de clave secreta o un par de claves públicas/privadas. El principal uso de JWT es el intercambio de información segura, por ejemplo, entre un servidor de autenticación y un servidor de recursos. En este proyecto, JWT se emplea para implementar

los módulos de autenticación y autorización con el soporte de SpringSecurity, de modo que los usuarios puedan acceder sólo a los recursos para los que tienen permisos. Además, su uso facilita la gestión de sesiones, eliminando la necesidad de mantener información de sesión en el servidor, lo que mejora la escalabilidad y la seguridad de la aplicación.

3.1.2 Lado cliente (*Front-end*)

Para la implementación del subsistema cliente del proyecto, se empleó el lenguaje de programación JavaScript en conjunto con el framework React.

JavaScript

JavaScript [16, 17] es un lenguaje de programación de alto nivel, dinámico y de interpretación, ampliamente utilizado para el desarrollo web. Es uno de los lenguajes más populares debido a su compatibilidad con todos los navegadores web modernos y su capacidad para interactuar con el DOM de las páginas web, lo que permite crear interfaces dinámicas y responder a las interacciones del usuario en tiempo real. JavaScript se distingue por su naturaleza de tipado dinámico, lo que significa que las variables no requieren una declaración explícita de tipo. Aunque esto otorga flexibilidad y facilita el desarrollo rápido, también puede conducir a errores difíciles de detectar durante la ejecución, ya que el tipo de las variables se resuelve en tiempo de ejecución. Las principales ventajas de usar JavaScript son:

- Ampliamente soportado en todos los navegadores, lo que facilita el desarrollo de aplicaciones web interactivas y dinámicas.
- Flexibilidad para trabajar con diferentes paradigmas de programación, como programación imperativa, orientada a objetos y funcional.
- Comunidad activa y abundancia de recursos, lo que hace más fácil encontrar soluciones a problemas comunes.
- Facilidad de integración con otras tecnologías web como HTML y CSS, permitiendo la creación de aplicaciones web completas.

React

React [18, 19] es una biblioteca de JavaScript de código abierto desarrollada por Facebook para construir interfaces de usuario interactivas y dinámicas, especialmente en aplicaciones web de una sola página (SPA). React permite crear componentes reutilizables que gestionan su propio estado, lo que facilita la construcción de interfaces complejas de manera más eficiente. Una de sus principales características es el uso del *Virtual DOM*, que mejora el rendimiento

al actualizar solo las partes de la interfaz que han cambiado, en lugar de volver a renderizar la página completa. React fue elegido debido a su amplia adopción en la industria y a su comunidad activa, lo que garantiza una gran cantidad de recursos, tutoriales, información y documentación disponibles para los desarrolladores. Además, su enfoque basado en componentes facilita la creación de interfaces reutilizables y escalables. React es también compatible con una gran variedad de tecnologías y herramientas, lo que hace que sea una opción ideal para este proyecto.

HTML

HTML [20] (HyperText Markup Language) es un lenguaje de marcado estándar utilizado para estructurar y crear el contenido de las páginas web. Como estándar aceptado por todos los navegadores actuales, HTML permite definir la estructura de un documento, incluyendo encabezados, párrafos, listas, tablas, enlaces y otros elementos necesarios para la presentación de información en la web.

CSS

CSS [21] (Cascade Style Sheets) es un lenguaje de diseño utilizado para controlar la presentación de los documentos escritos en HTML. Permite separar la estructura del contenido de la apariencia visual, lo que facilita la gestión y mantenimiento de los estilos en las páginas web. CSS define aspectos como los colores, fuentes, márgenes, espaciados, alineaciones y otros elementos visuales, lo que permite crear interfaces atractivas y coherentes. En este proyecto, CSS se ha empleado para modificar la apariencia visual de las diferentes páginas y componentes de la aplicación, con la intención de que la interfaz de usuario sea clara, sencilla, accesible y visualmente agradable.

Tailwind

Tailwind [22] es un framework de diseño de código abierto que permite la creación de interfaces de usuario personalizadas de manera rápida y eficiente. A diferencia de otros framework(s) como Bootstrap, Tailwind adopta un enfoque utilitario, proporcionando clases pequeñas y específicas que se aplican directamente a los elementos HTML para definir sus estilos, mientras que Bootstrap ofrece clases más complejas y componentes listos para usar. Esto permite un control más granular sobre el diseño y una mayor flexibilidad para adaptarlo a las necesidades específicas de cada proyecto. En este proyecto, Tailwind CSS se utilizó específicamente para dar formato a algunos componentes de la interfaz de usuario, mejorando la experiencia visual y facilitando el desarrollo del *Front-end*.

HTTP

HTTP [23] (HyperText Transfer Protocol) es un protocolo de comunicación utilizado para la transferencia de información entre un cliente (como un navegador web) y un servidor. Este protocolo es la base de la web, permitiendo que los clientes realicen solicitudes y reciban respuestas de los servidores, facilitando el intercambio de datos como páginas web, imágenes, archivos y otros recursos. Las transacciones HTTP consisten en una solicitud enviada por el cliente y una respuesta proporcionada por el servidor, las cuales pueden incluir no solo los recursos solicitados, sino también metadatos sobre esos recursos, como encabezados y códigos de estado.

NextUI

NextUI [24] es una librería para React que proporciona un conjunto de componentes estilizados y accesibles, diseñados para facilitar la creación de aplicaciones web modernas y atractivas.

Esta librería se enfoca en la simplicidad, personalización y rendimiento, ofreciendo elementos como botones, modales, tarjetas, entre otros, todo con un diseño visual limpio y consistente. NextUI se integra fácilmente con proyectos React.

Embla Carousel y Swiper

Embla Carousel [25] y Swiper [26] son dos librerías de componentes de IU para React que facilitan la implementación de carruseles y deslizadores de imágenes en aplicaciones web. Ambas librerías permiten crear interfaces de usuario atractivas e interactivas, ofreciendo a los desarrolladores opciones flexibles para integrar elementos deslizables de forma eficiente.

3.2 Herramientas

A continuación se describirán las herramientas utilizadas en cada parte del desarrollo de este proyecto.

IntelliJ IDEA

IntelliJ IDEA [27] es un IDE desarrollado por Jetbrains [28]. IntelliJ IDEA es uno de los IDE(s) más utilizados por la comunidad debido a sus potentes capacidades y funcionalidades. Admite una amplia variedad de lenguajes de programación y framework(s), como Java y Spring. IntelliJ IDEA ofrece tanto una versión gratuita, *Community*, como una versión de

pago más avanzada, *Ultimate*. También ofrece un sistema de autocompletado inteligente, integración con control de versiones como Git, soporte para pruebas integradas como JUnit y un eficiente sistema de depuración. Por todos estos motivos, se ha elegido esta herramienta para el desarrollo del lado servidor (*Back-end*) en este proyecto.

Visual Studio Code

Visual Studio Code [29] es un editor de código fuente desarrollado por Microsoft [30], ligero y ampliamente utilizado en el mundo del desarrollo de software. Es una solución gratuita y de código abierto que está disponible para Windows, Linux, macOS y Web. Este editor ofrece una amplia gama de características que facilitan el trabajo de los desarrolladores, tales como:

- Soporte para depuración de código.
- Integración con herramientas de control de versiones como Git.
- Autocompletado de código y resaltado de sintaxis.
- Alta personalización de la interfaz de usuario.
- Terminal integrada directamente en el editor.
- Gran cantidad de extensiones y *plugins* disponibles.

Por todos estos motivos, se ha elegido esta herramienta para el desarrollo del lado cliente (*Front-end*) en este proyecto.

MySQL Command Line Client

MySQL Command Line Client [31] es una herramienta de línea de comandos que permite interactuar directamente con bases de datos MySQL. A través de esta interfaz, los usuarios pueden ejecutar consultas SQL, gestionar bases de datos y realizar diversas operaciones de administración y mantenimiento. Su principal ventaja es que ofrece un control total sobre las operaciones de la base de datos, permitiendo una ejecución rápida y eficiente de tareas complejas. En este proyecto, se utilizó MySQL Command Line Client para gestionar y manipular la base de datos de manera directa, facilitando la administración de datos en el sistema.

Node

Node.js [32] es un entorno de ejecución de JavaScript en el lado del servidor que permite ejecutar aplicaciones JavaScript fuera del navegador. Es ampliamente utilizado en el desarrollo de aplicaciones web y móviles debido a su eficiencia, escalabilidad y a la capacidad de manejar

grandes cantidades de operaciones simultáneas. Node.js ha sido utilizado únicamente para el desarrollo del *Front-end*, en combinación con React, para gestionar la ejecución del código JavaScript.

Postman

Postman [33] es una herramienta ampliamente utilizada para probar API(s). Permite a los usuarios realizar peticiones HTTP a una API y recibir sus respuestas, facilitando así el proceso de prueba y depuración de los servicios que esta expone. Postman ofrece la posibilidad de crear colecciones de peticiones, lo que agiliza las pruebas y evita que los usuarios tengan que escribirlas repetidamente. Esta herramienta fue elegida para este proyecto debido a su facilidad de uso y a la experiencia previa del autor con ella.

Git

Git [34] es un sistema de control de versiones ampliamente utilizado en proyectos de desarrollo de software colaborativos. Su principal objetivo es facilitar la colaboración entre equipos y permitir el trabajo paralelo de manera eficiente. En este proyecto se eligió esta herramienta principalmente para gestionar las versiones del código.

Overleaf

Overleaf [35] es una plataforma en línea que permite la colaboración y la creación de documentos escritos en LaTeX. Destaca por su capacidad para crear documentos bien estructurados y estilizados. En este proyecto, se utilizó para redactar la memoria.

Capítulo 4

Introducción al desarrollo

En este proyecto, se propone la creación de una aplicación web para la búsqueda y gestión de reservas y alojamientos. Para ello, antes de comenzar con el resto de fases, se necesita establecer un objetivo, así como unas pautas y reglas para cumplirlo de manera efectiva. Con este fin, existen las *metodologías*.

En el ámbito de desarrollo de *software*, una *metodología* puede definirse como un enfoque estructurado que define procesos, prácticas y principios para planificar, crear y mantener *software*. Ayuda a organizar equipos, gestionar recursos y asegurar la calidad del producto. Existen dos grandes grupos: metodologías ágiles, que enfatizan en la flexibilidad y la iteración, y metodologías tradicionales o en cascada, que se centran en un enfoque más lineal y secuencial para el desarrollo.

4.1 Metodologías

Dentro del enfoque de las metodologías de desarrollo software se pueden adoptar dos grupos diferenciados.

4.1.1 Metodologías tradicionales

- Se enfocan en planificar detalladamente todo el proceso desde el inicio.
- Requieren una documentación exhaustiva antes de proceder con la implementación.
- Siguen un enfoque secuencial donde una tarea no comienza hasta que la anterior ha terminado.
- Ponen mayor énfasis en la estabilidad y previsibilidad.
- Son difíciles de adaptar a cambios una vez iniciado el proyecto.

- El producto final se entrega únicamente al finalizar el proyecto.

4.1.2 Metodologías ágiles

- Se basan en ciclos de desarrollo iterativos.
- Se adaptan fácilmente a cambios en los requisitos iniciales.
- Ajustan la planificación según avanza el proyecto.
- Promueven la comunicación y colaboración constante entre las partes involucradas.
- Priorizan la retroalimentación rápida y continua.
- Ofrecen valor de forma frecuente entregando nuevas funcionalidades al cliente.
- Facilitan la detección y corrección de errores durante el desarrollo.

Tras una cuidadosa valoración de las características del proyecto, considerando los posibles cambios en los requisitos iniciales y la necesidad de un control periódico, se optó por una metodología ágil, concretamente *Scrum* adaptada a un proyecto con un equipo reducido, que también es una metodología ampliamente utilizada en entornos empresariales.

4.2 Scrum

Scrum [36, 37] es una metodología ágil basada en la entrega iterativa e incremental de valor al cliente. Destaca por su enfoque altamente efectivo y flexible, además de buscar potenciar la colaboración, la comunicación constante y la transparencia a lo largo de todo el proceso de desarrollo software.

4.2.1 Roles

Dentro de la metodología *Scrum* se distinguen tres roles principales: **Product Owner** o propietario del producto, **Scrum Master** o experto en *Scrum* y el **Equipo de Desarrollo** encargado de implementar las tareas. Cada uno de estos roles tiene responsabilidades específicas que contribuyen al éxito del proyecto. En la figura 4.1 se ilustra un flujo básico de trabajo entre estos roles, destacando su interacción constante para garantizar la eficiencia del proceso.

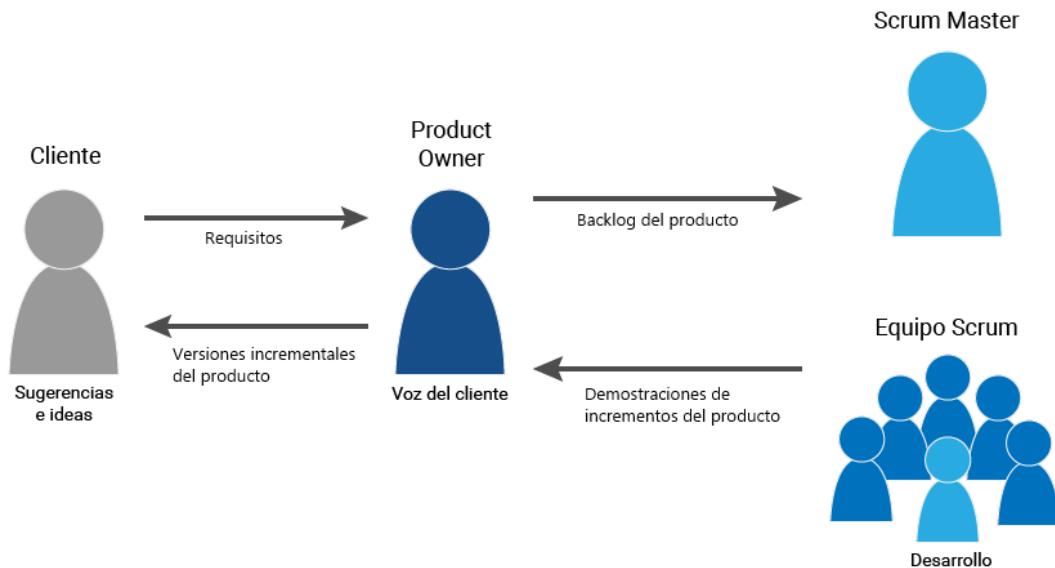


Figura 4.1: Roles en Scrum

Product Owner

El **Product Owner** es el rol dentro de un equipo ágil, especialmente en *Scrum*, responsable de maximizar el valor del producto y gestionar el *backlog* del producto. Este rol actúa como el principal punto de contacto entre los interesados y el equipo de desarrollo, asegurándose de que las necesidades del cliente y del negocio estén reflejadas de manera clara en los requisitos y prioridades del producto. En este proyecto, el rol de **Product Owner** es ejercido por el alumno. Algunas de sus tareas incluyen:

- Definir y priorizar las características del producto.
- Mantener y gestionar el **Product Backlog**.
- Asegurar que el equipo de desarrollo entienda los requisitos y expectativas.
- Tomar decisiones sobre qué se debe construir, en qué orden y cuándo.
- Tiene el control de la organización de los *Sprints*.
- Es el encargado de controlar que el producto se implementa de acuerdo con los requisitos del cliente.

Scrum Master

El **Scrum Master** actúa como coordinador y tutor del equipo *Scrum*, ayudando a entender y aplicar correctamente las prácticas y reglas el marco de trabajo *Scrum*. Su responsabilidad

también abarca la resolución de cualquier obstáculo que pueda afectar el progreso del equipo, aumentando así la productividad. Además, fomenta la comunicación y colaboración entre los distintos equipos y facilita los eventos de *Scrum*, como las reuniones diarias o las revisiones del *Sprint*. En resumen, el **Scrum Master** asegura que el equipo trabaje de manera eficiente y que el marco *Scrum* se siga adecuadamente. En este proyecto, el rol de **Scrum Master** es ejercido por el tutor.

Equipo de desarrollo

El **Equipo de Desarrollo** está formado por un grupo de personas encargadas de implementar y entregar el producto final de manera incremental en cada *Sprint*. Para lograrlo, planifican y se autoorganizan durante los *Sprints*, colaborando con el **Product Owner** para entender los requisitos y con el **Scrum Master** para mantener un flujo de trabajo efectivo y productivo. En resumen, el equipo trabaja de forma conjunta para asegurar que el producto evolucione y cumpla con los objetivos establecidos. En este proyecto, el rol del **equipo de desarrollo** es ejercido por el alumno.

4.2.2 Sprint

Un *Sprint* es el período de tiempo en el que se realiza el trabajo de desarrollo. Generalmente, tiene una duración fija y predefinida de entre una a cuatro semanas, siendo dos semanas lo más común. Aunque la duración del *Sprint* suele ser constante, puede ajustarse dependiendo del ritmo del equipo de desarrollo. Durante este tiempo, se implementan los requisitos y funcionalidades que el **Product Owner** ha planificado y definido previamente.

Cada *Sprint* suma nuevos elementos de trabajo respecto al anterior, lo que se conoce como **incremento**. Al finalizar cada *Sprint*, el equipo presenta una versión funcional del producto con los avances logrados, que puede ser entregada al cliente. Este enfoque permite revisar y ajustar el desarrollo de forma continua, lo que facilita la adaptación a cambios y la entrega constante de valor al cliente.

Un *Sprint* se puede dividir en las siguientes fases:

- **1. Planificación del Sprint (*Sprint Planning*):** en esta fase, se define la cantidad de trabajo que se realizará durante el *Sprint*, estableciendo las tareas y los objetivos a alcanzar.
- **2. Implementación del Sprint:** durante esta fase, se llevan a cabo las tareas asignadas, trabajando en la implementación de las funcionalidades definidas en la planificación.

- **3. Revisión del Sprint (Sprint Review):** al finalizar el *Sprint*, se revisa el trabajo realizado y se entrega el incremento desarrollado, mostrando el progreso alcanzado y asegurando que cumpla con los requisitos establecidos.
- **4. Retrospectiva del Sprint (Sprint Retrospective):** al concluir el *Sprint*, se analiza el desempeño del equipo y del producto, identificando las posibles mejoras en los procesos y en el rendimiento para optimizar el siguiente *Sprint*.

4.2.3 Artefactos

Los artefactos en *Scrum* son herramientas que proporcionan transparencia y visibilidad sobre el trabajo realizado a lo largo del proyecto, facilitando así el proceso de adaptación para todas las partes involucradas. Al aplicar la metodología *Scrum*, se generan los siguientes artefactos.

Backlog del Producto (Product Backlog)

El *Backlog del Producto* o *Product Backlog* consiste en una lista priorizada que incluye todos los elementos que se esperan incorporar en la solución final. Estos elementos pueden ser funcionalidades, características, correcciones...

Cada *item* del *Product Backlog* está representado mediante una *historia de usuario*. El **Product Owner** es el encargado de gestionar y mantener este artefacto ordenado y estructurado.

Backlog del Sprint (Sprint Backlog)

El *Backlog del Producto* o *Sprint Backlog* es una lista de elementos seleccionados del *Product Backlog* que el **Equipo de Desarrollo** se compromete a implementar en un *Sprint* específico. El *Sprint Backlog* es gestionado por el equipo de desarrollo.

Incremento

El incremento es el resultado final de cada *Sprint*, el cual puede ser entregado al cliente. Este incremento incluye las funcionalidades y mejoras completadas durante el *Sprint* y refleja el avance con respecto al *Sprint* anterior, añadiendo valor al producto.

En la figura 4.2 se ilustra un flujo básico de trabajo de un proyecto *Scrum*.

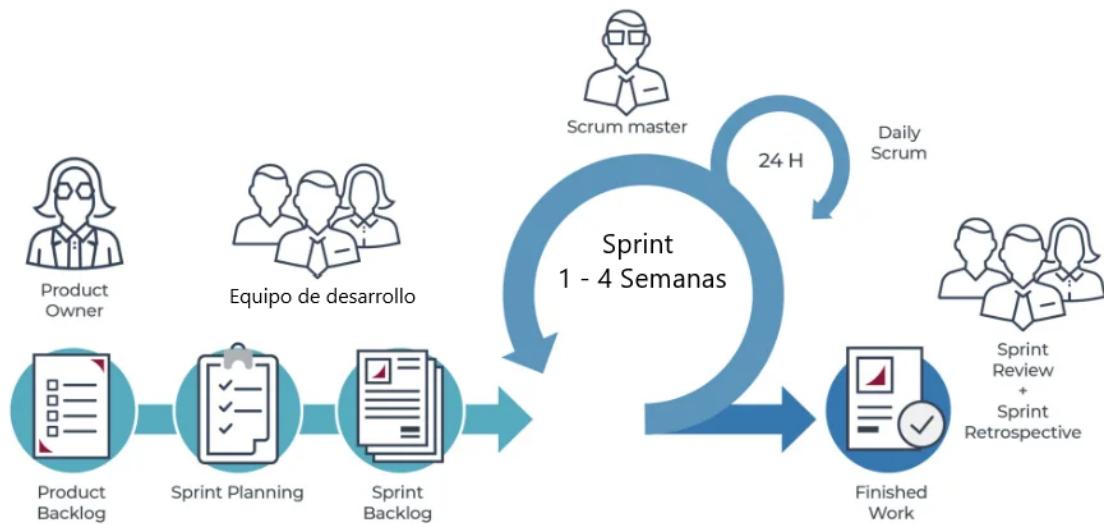


Figura 4.2: Flujo básico en un proyecto *Scrum*

4.2.4 Integración de Scrum en el proyecto

Para este proyecto, se abordó la metodología *Scrum* con una adaptación específica para las dimensiones del equipo, ya que está compuesto únicamente por el tutor y el autor del proyecto. Como resultado, se consideró necesario implementar una versión simplificada de la metodología *Scrum*.

Para optimizar la ejecución del proyecto y hacerlo de forma efectiva, en lugar de realizar reuniones diarias, se establecieron puntos de control y seguimiento regulares durante el transcurso de cada *Sprint*, adaptando de esta forma todo el proceso acorde al tamaño reducido del equipo.

El alumno ejercerá el rol de *equipo de desarrollo*, encargado de la implementación de las *historias de usuario* y el rol de *Product Owner*, encargado de la responsabilidad de gestionar y priorizar el *Product Backlog*. Por otro lado, el tutor representará el rol de *Scrum Master*, el cual facilitará el proceso *Scrum* y se encargará de que se cumpla el enfoque que busca esta metodología.

Capítulo 5

Viabilidad

PARA analizar el posible éxito de un proyecto de software, es necesario estimar las capacidades tanto técnicas como económicas del trabajo. Este proceso permite evaluar la factibilidad del proyecto en diferentes áreas, lo que facilitará la toma de decisiones durante el desarrollo e implementación. Se considerarán las siguientes competencias:

5.1 Viabilidad técnica

Las tecnologías elegidas para este proyecto fueron seleccionadas en función de los objetivos y requisitos del mismo.

En cuanto a la infraestructura, se verificó que el entorno de desarrollo del autor del proyecto fuera compatible con los elementos fundamentales para el desarrollo de un producto *software*, incluyendo servidores, bases de datos, recursos *software* y recursos *hardware*. En este caso, el entorno de desarrollo permitió la ejecución del servidor de bases de datos MySQL sin inconvenientes a lo largo de todo el proceso.

También se ha analizado la capacidad del propio autor para poder llevar a cabo este proyecto, quien, desde un primer momento, ofreció soluciones realistas acordes a su experiencia y formación. El autor también tiene conocimientos previos en algunas de las tecnologías utilizadas (como pueden ser los lenguajes Java, JavaScript, [SQL](#) o [CSS](#)), lo que le permitió anticipar y abordar los posibles obstáculos de manera efectiva.

Esta evaluación de la viabilidad técnica aseguró que las tecnologías elegidas son las adecuadas para el proyecto y que el autor cuenta con las habilidades necesarias para realizar su implementación.

5.2 Viabilidad económica

En esta sección se examinarán los aspectos económicos del proyecto, concretamente se analizarán los costes estimados y reales de los recursos técnicos, humanos y materiales.

5.2.1 Recursos humanos

El equipo *Scrum* está formado por un *Scrum Master* y el *equipo de desarrollo*. Para calcular los costes, se considerará que el *equipo de desarrollo* estará integrado por un programador *Junior* y que el *Scrum Master* llevará a cabo reuniones semanales de una hora con el equipo.

- Salario medio de un programador *Junior* en España: **1.750€/mes 10,10€/hora** [38].
- Salario medio de un *Scrum Master* en España: **3.550€/mes 20,47€/hora** [39].
- Porcentaje de impuestos a cargo de la empresa por trabajador en España: **30%** [40].

	Trabajo estimado	Coste medio	Coste estimado
<i>Programador Junior</i>	280 horas	10,10 €/hora	2.828 €
<i>Scrum Master</i>	28 horas	20,47 €/hora	573,16 €
<i>Total</i>			3.401,16 €
<i>Total + Impuestos</i>			4.421,51 €

Cuadro 5.1: Costes estimados de recursos humanos

	Trabajo real	Coste medio	Coste real
<i>Programador Junior</i>	330 horas	10,10 €/hora	3.333 €
<i>Scrum Master</i>	33 horas	20,47 €/hora	675,51 €
<i>Total</i>			4.008,51 €
<i>Total + Impuestos</i>			5.211,06 €

Cuadro 5.2: Costes reales de recursos humanos

Como se puede observar en la tabla 5.1, el coste estimado inicialmente para los recursos humanos era de 4.421,51 €. No obstante, debido a la dificultad de algunos *Sprints*, se requirió más tiempo del previsto, lo que incrementó los costes por las horas extras de trabajo. Como se detalla en la tabla 5.2), el coste total ascendió a 5.211,06 €, superando la estimación inicial en 789,55 €.

5.2.2 Recursos técnicos

En cuanto al coste de recursos técnicos, como pueden ser herramientas o tecnologías, se utilizó la versión *Ultimate* de IntelliJ IDEA, cuyo coste mensual es de 16,90 € [41].

	Coste mensual	Tiempo de uso	Coste
<i>IntelliJ IDEA Ultimate</i>	16,90 €	8 meses	135,20 €
<i>Total</i>			135,20 €

Cuadro 5.3: Costes de recursos técnicos

En la tabla 5.3 vemos que el coste total de uso de 8 meses de la versión *Ultimate* de IntelliJ IDEA asciende a un total de 135,20 €.

5.2.3 Recursos materiales

Para analizar el coste total de los recursos materiales, se considerará un único elemento, un ordenador portátil utilizado para realizar el trabajo del desarrollo e implementación. Como el proyecto ha tenido una duración de 8 meses, se tomarán en cuenta los siguientes datos.

- Valor de un portátil de gama media: **1000 €**.
- Vida útil promedio de un portátil de ese precio [42]: **4 años**.

En la siguiente tabla 5.4, se considera una depreciación del 20% sobre el valor de compra del portátil. Como se puede observar, el coste total asciende a la suma de 450 €.

	Coste inicial	Depreciación	Amortización	Coste
<i>Portátil</i>	1.000 €	200 €	250 €	450 €
<i>Total</i>				450 €

Cuadro 5.4: Costes de recursos técnicos

5.2.4 Costes totales

En esta sección, se realizará una comparativa entre los costes estimados y reales para el proyecto. En la tabla 5.5 se muestran los costes de cada recurso implicado siguiendo la planificación inicial, con un total de 5.006,71 €. Sin embargo, debido al aumento en el tiempo de desarrollo, el coste total del proyecto también ha incrementado. Como se puede observar en la tabla 5.6 el coste total asciende a 5.796,26 €, lo que representa un incremento de 789,55 €.

Este aumento en el coste no supuso ningún obstáculo significativo para alcanzar los objetivos del proyecto.

	Coste
Recursos humanos	4.421,51 €
Recursos técnicos	135,20 €
Recursos materiales	450 €
Total	5.006,71 €

Cuadro 5.5: Costes totales estimados

	Coste
Recursos humanos	5.211,06 €
Recursos técnicos	135,20 €
Recursos materiales	450 €
Total	5.796,26 €

Cuadro 5.6: Costes totales reales

5.3 Planificación de *Sprints* en el proyecto

La planificación de los *Sprints* para este proyecto se hizo por fases. Para cada *Sprint*, el equipo de desarrollo implementará un módulo específico, y tendrán una duración de entre 2 a 4 semanas, dependiendo de la complejidad de las funcionalidades que se quieran abordar.

A continuación, se muestra el listado de *Sprints* planificados junto con las tareas que se abordaron en cada uno de ellos, así como de las historias de usuario que se implementarán y los tiempos estimados y reales del trabajo por parte del alumno.

5.3.1 Sprint 1 - Puesta en marcha (05/05/2024 - 18/05/2024)

En este *Sprint* se realizará la fase de análisis de requisitos del producto y la configuración inicial del entorno de desarrollo. Este *Sprint* estará dividido por las siguientes fases:

- Análisis del sistema que se va a desarrollar.
- Creación y configuración inicial del entorno de desarrollo para la parte servidor (*Backend*) en Java y Springboot.

- Creación y configuración inicial del entorno de desarrollo para la parte cliente (*Front-end*) en React.
- Creación y configuración de la base de datos MySQL para la persistencia de datos del sistema.

Tiempo estimado de duración: **20 horas**.

Tiempo real de duración: **20 horas**.

5.3.2 Sprint 2 - Gestión de usuarios (19/05/2024 - 29/06/2024)

En este *Sprint* se implementará el módulo de gestión de usuarios, este *Sprint* estará dividido en:

- Implementación del módulo en *Back-end* y *Front-end*. Entre las funcionalidades de este módulo estarán el registro, inicio y cierre de sesión, edición del perfil, capacidad del administrador para bloquear y desbloquear usuarios, selección de tema (modo claro o modo oscuro) y selección de idioma.
- Realización de las pruebas del módulo de gestión de usuarios.

El retraso en el tiempo de este *Sprint* se debió principalmente a la falta de experiencia del alumno en la selección del tema y algunas complicaciones a la hora de la actualización del avatar, lo que requirió más tiempo del previsto de cara a su implementación.

Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **60 horas**.

5.3.3 Sprint 3 - Gestión de alojamientos (30/06/2024 - 27/07/2024)

En este *Sprint* se comenzará a implementar el módulo de gestión de alojamientos, este *Sprint* estará dividido en:

- Implementación del módulo en *Back-end* y *Front-end*. Entre las funcionalidades de este módulo estarán el registro de un alojamiento, editar los detalles de un alojamiento que has registrado, ver tus propios alojamientos y capacidad de cerrar y abrir tus alojamientos.
- Realización de las pruebas del módulo de gestión de alojamientos.

Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **40 horas**.

5.3.4 Sprint 4 - Gestión de listados y búsquedas (28/07/2024 - 24/08/2024)

En este *Sprint* se continuara implementando el módulo de gestión de alojamientos, este *Sprint* estará dividido en:

- Implementación del módulo en *Back-end* y *Front-end*. Entre las funcionalidades de este módulo estarán la búsqueda de alojamientos con diferentes parámetros, capacidad del administrador para bloquear y desbloquear alojamientos.
- Realización de las pruebas de esta parte del módulo de gestión de alojamientos.

Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **40 horas**.

5.3.5 Sprint 5 - Integración con un servicio web externo (25/08/2024 - 28/09/2024)

En este *Sprint* se continuara implementando el módulo de gestión de alojamientos, este *Sprint* estará dividido en:

- Integración de la [API](#) en *Front-end*. Entre las funcionalidades de este módulo estarán la integración de la [API](#) para que se puedan realizar búsquedas contra ella, poder consultar los detalles de un alojamiento de **DeepDive** y poder consultar los detalles de un alojamiento servido por la [API](#).
- Realización de las pruebas de esta parte del módulo de gestión de alojamientos.

El retraso en el tiempo de este *Sprint* se debió principalmente a la falta de experiencia del alumno en la selección e integración de una [API](#) en el proyecto, ya que algunas opciones permitían pocas peticiones mensuales de manera gratuita.

Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **50 horas**.

5.3.6 Sprint 6 - Gestión de reservas (29/09/2024 - 09/11/2024)

En este *Sprint* se implementará el módulo de gestión de reservas, este *Sprint* estará dividido en:

- Implementación del módulo en *Back-end* y *Front-end*. Entre las funcionalidades de este módulo estarán realización de reservas para los dos tipos de alojamiento (los registrados en **DeepDive** y los servidos por la [API](#)), visualizar tus propias reservas y cancelar la reserva si hay suficiente antelación.
- Realización de las pruebas del módulo de gestión de reservas.

El retraso en el tiempo de este *Sprint* se debió a la dificultad de integrar los dos tipos de reserva (una para cada tipo de alojamiento) en la aplicación, lo que requirió más tiempo del previsto de cara a su implementación.

Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **60 horas**.

5.3.7 Sprint 7 - Gestión de comentarios (10/11/2024 - 23/11/2024)

En este *Sprint* se implementará el módulo de la gestión de comentarios, este *Sprint* estará dividido en:

- Integración del módulo en *Back-end* y *Front-end*. Entre las funcionalidades de este módulo se incluirán la posibilidad de realizar comentarios para evaluar los alojamientos en los que el usuario se haya hospedado, visualizar los comentarios de los dos tipos de alojamiento (los registrados en **DeepDive** y los servidos por la [API](#)) y la capacidad del administrador de bloquear comentarios.
- Realización de las pruebas del módulo de gestión de comentarios.

Tiempo estimado de duración: **20 horas**.

Tiempo real de duración: **20 horas**.

5.3.8 Sprint 8 - Redacción de la memoria (24/11/2024 - 22/12/2024)

En este *Sprint* se redactará la memoria. Tiempo estimado de duración: **40 horas**.

Tiempo real de duración: **40 horas**.

5.4 Comparativa de tiempo estimado y tiempo real

En esta sección se realizará una comparativa entre los tiempos estimados para cada *Sprint* y el tiempo real que se empleó en completarlos. De este modo, se destacarán las áreas en las que las previsiones iniciales se mantuvieron, así como aquellas que experimentaron alguna variación.

Como se puede observar en la tabla 5.7, el proyecto requirió 50 horas adicionales de trabajo en comparación con la planificación inicial. Teniendo en cuenta que el trabajo realizado fue de 10 horas a la semana (2 horas diarias), esto resultó en un retraso de 5 semanas en la entrega del producto final.

La flexibilidad que aporta la metodología *Scrum*, junto con la coordinación y comunicación constante con el *Product Owner* y *Scrum Master*, permitió gestionar de manera eficaz la

variación temporal observada con respecto a la planificación inicial, garantizando así que el proyecto se completará exitosamente y de acuerdo con los objetivos propuestos.

	Duración estimada	Duración real	Desviación
<i>Sprint 1</i>	20 horas	20 horas	0 horas
<i>Sprint 2</i>	40 horas	60 horas	20 horas
<i>Sprint 3</i>	40 horas	40 horas	0 horas
<i>Sprint 4</i>	40 horas	40 horas	0 horas
<i>Sprint 5</i>	40 horas	50 horas	10 horas
<i>Sprint 6</i>	40 horas	60 horas	20 horas
<i>Sprint 7</i>	20 horas	20 horas	0 horas
<i>Sprint 8</i>	40 horas	40 horas	0 horas
<i>Total</i>	280 horas	330 horas	50 horas

Cuadro 5.7: Comparación de tiempos

Capítulo 6

Análisis

La fase de análisis de un proyecto de desarrollo *software* es un punto clave, ya que de ella depende en gran medida el éxito o fracaso del proyecto. En este capítulo, se estudiarán los requisitos que el producto debe cumplir para ser considerado completo y correcto.

6.1 Actores

Un aspecto clave en el desarrollo de software es identificar los posibles actores que interactuarán con el producto, en este caso, los usuarios finales de la aplicación. Después de un periodo de análisis, se planteó la siguiente identificación de actores:

- **Usuario no registrado:** se trata de cualquier usuario que no se haya registrado o no se haya autenticado en la aplicación. Este usuario solo tendrá acceso a:
 - Funcionalidades de usuario limitadas:
 - * Registro de un nuevo usuario.
 - * Inicio de sesión con un usuario que ha sido registrado previamente.
 - * Selección de idioma.
 - * Selección de tema.
- **Usuario registrado:** es un usuario que tiene la sesión iniciada en la aplicación. Este usuario tendrá acceso a:
 - Funcionalidades de usuario:
 - * Cierre de sesión.
 - * Actualizar los detalles de su perfil.
 - * Establecer un nuevo avatar.
 - * Consultar las políticas de privacidad, consultar información sobre **DeepDive** y contactar con el servicio técnico.

- * Selección de idioma.
- * Selección de tema.
- Funcionalidades de alojamiento:
 - * Registro de un alojamiento.
 - * Editar los detalles de un alojamiento que previamente ha registrado.
 - * Consultar sus propios alojamientos.
 - * Cerrar un alojamiento.
 - * Abrir un alojamiento que previamente haya cerrado.
 - * Ver los detalles de un alojamiento ya sea suyo o de otro usuario.
 - * Consultar la disponibilidad del alojamiento en un periodo determinado.
 - * Realizar búsquedas sobre los alojamientos.
- Funcionalidades de reservas:
 - * Realizar una reserva.
 - * Consultar sus reservas.
 - * Cancelar una reserva si se realiza con suficiente antelación.
 - * Valorar una reserva que ya ha concluido.
- **Usuario administrador:** es un usuario que tiene privilegios de administrador. Este usuario tiene acceso a todas las características del usuario registrado y, además, dispone de:
 - Funcionalidades de usuario:
 - * Bloquear a un usuario.
 - * Desbloquear a un usuario que previamente había bloqueado.
 - * Consultar los usuarios que han sido bloqueados.
 - * Selección de idioma.
 - * Selección de tema.
 - Funcionalidades de alojamiento:
 - * Bloquear un alojamiento.
 - * Desbloquear un alojamiento.
 - * Consultar los alojamientos que han sido bloqueados.
 - Funcionalidades de reservas:
 - * Bloquear una valoración.

6.2 Requisitos

En esta sección se describirán los requisitos que el producto final debe cumplir para alcanzar una solución satisfactoria. También, en esta sección se especificarán las funcionalidades y características necesarias del producto final, diferenciando entre dos tipos de requisitos: *requisitos funcionales* y *requisitos no funcionales*.

6.2.1 Requisitos funcionales

Los *requisitos funcionales* definen las características y funcionalidades específicas que el producto *software* debe ofrecer. En este proyecto, se han identificado los siguientes:

- Gestión de usuarios:
 - Registro de un usuario.
 - Inicio de sesión de un usuario que previamente se ha registrado.
 - Cierre de sesión de un usuario que tuviera la sesión iniciada.
 - Visualización de los datos del perfil del usuario.
 - Actualización de los datos del perfil del usuario.
 - Actualización del avatar del usuario.
 - Bloquear a un usuario.
 - Desbloquear a un usuario.
 - Consultar los usuarios que han sido bloqueados.
 - Consultar información sobre **DeepDive** y la política de privacidad.
 - Contactar con el servicio técnico de **DeepDive**.
- Gestión de alojamientos:
 - Registrar un alojamiento.
 - Consultar tus alojamientos.
 - Editar los detalles de tus alojamientos.
 - Ver los detalles de un alojamiento.
 - Consultar la disponibilidad de un alojamiento en un periodo determinado.
 - Realizar consultas sobre los alojamientos.
 - Cerrar un alojamiento que previamente ha sido registrado por el usuario.
 - Abrir un alojamiento que previamente ha sido cerrado.

- Bloquear un alojamiento.
 - Desbloquear un alojamiento.
 - Consultar los alojamientos que han sido bloqueados.
- Gestión de reservas:
 - Realizar una reserva.
 - Consultar tus reservas.
 - Cancelar una reserva si se realiza con la debida antelación.
 - Valorar una reserva si ya ha concluido.
 - Bloquear una valoración.

6.2.2 Requisitos no funcionales

Los *requisitos no funcionales* no están directamente relacionados con las funcionalidades del producto, pero son esenciales para garantizar su éxito. En este proyecto, se han identificado los siguientes:

- **Rendimiento:** se intentó optimizar la aplicación minimizando las llamadas al servidor mediante el almacenamiento de información en el cliente siempre que sea posible.
- **Escalabilidad:** el objetivo es desarrollar una aplicación que permita añadir nuevas funcionalidades de forma sencilla, sin afectar a las ya existentes.
- **Seguridad:** el objetivo es implementar una aplicación que proteja los recursos mediante roles.
- **Usabilidad:** el objetivo de este trabajo es desarrollar una aplicación con una interfaz de usuario sencilla e intuitiva.
- **Accesibilidad:** el objetivo de este trabajo es desarrollar una aplicación que integre funcionalidades diseñadas para mejorar la experiencia del usuario final, como la selección de tema (claro u oscuro) y la posibilidad de elegir el idioma de la aplicación.

6.3 Historias de usuario

Las historias de usuario [43] son una técnica ágil ampliamente empleada para capturar las necesidades y expectativas de los usuarios finales respecto al producto *software*.

Las historias de usuario son una representación escrita de un requisito que debe ser cumplido, generalmente en una o dos frases, utilizando un lenguaje comprensible para el usuario.

Están diseñadas para gestionar de forma rápida los requisitos de los usuarios, evitando la necesidad de generar grandes volúmenes de documentación.

En resumen, las historias de usuario se enfocan en el "quien", "que" y "por qué", omitiendo detalles técnicos para que el equipo de desarrollo pueda comprender rápidamente las necesidades del usuario. Una historia de usuario sigue la siguiente estructura:

- Título descriptivo.
- Descripción breve.
- Rol del usuario.
- Acción que el usuario desea realizar.
- Objetivo esperado por el usuario.

A continuación, se listarán las historias de usuario del proyecto.

HU-<01>	Registrarse
<i>Título</i>	Registrarse en la aplicación.
<i>Descripción</i>	Como usuario no registrado, quiero poder registrarme en la aplicación para poder acceder a sus funcionalidades.
<i>Actor</i>	Usuario no registrado.
<i>Acción</i>	Como usuario no registrado, quiero poder proporcionar mi correo electrónico, mi nombre de usuario, mi contraseña, mi fecha de nacimiento y mi género para poder registrarme en la aplicación.
<i>Objetivo</i>	Acceder a las funcionalidades que ofrece la aplicación.

Cuadro 6.1: HU-<01>: Registrarse.

HU-<02>	Iniciar sesión
<i>Título</i>	Iniciar sesión en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder iniciar sesión en la aplicación con mis credenciales para poder acceder a sus funcionalidades.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero proporcionar mi correo electrónico y mi contraseña para poder iniciar sesión en la aplicación.
<i>Objetivo</i>	Acceder a las funcionalidades que ofrece la aplicación.

Cuadro 6.2: HU-<02>: Iniciar sesión.

HU-<03>	Cerrar sesión
<i>Título</i>	Cerrar sesión en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder cerrar la sesión para dejar de usar la aplicación.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero cerrar la sesión y desconectarme de los servicios de la aplicación.
<i>Objetivo</i>	Cerrar la sesión del usuario en la aplicación.

Cuadro 6.3: HU-<03>: Cerrar sesión.

HU-<04>	Ver perfil
<i>Título</i>	Ver el perfil del usuario en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder ver los datos de mi perfil en la aplicación para poder gestionarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero consultar los datos de mi perfil, como el nombre, apellidos, dirección, etc...
<i>Objetivo</i>	Consultar los datos del perfil del usuario.

Cuadro 6.4: HU-<04>: Ver perfil.

HU-<05>	Editar perfil
<i>Título</i>	Editar el perfil del usuario en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder editar los datos de mi perfil en la aplicación para poder actualizarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero actualizar los datos de mi perfil, como el nombre, apellidos, dirección, etc...
<i>Objetivo</i>	Actualizar los datos del perfil del usuario.

Cuadro 6.5: HU-<05>: Editar perfil.

HU-<06>	Establecer un avatar
<i>Título</i>	Establecer el avatar del usuario en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder establecer un nuevo avatar en la aplicación para poder actualizarlo.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero establecer un nuevo avatar en la aplicación.
<i>Objetivo</i>	Establecer un nuevo avatar del usuario.

Cuadro 6.6: HU-<06>: Establecer un avatar.

HU-<07>	Bloquear a un usuario
<i>Título</i>	Bloquear a un usuario en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder bloquear a un usuario en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero bloquear a un usuario en la aplicación.
<i>Objetivo</i>	Bloquear a un usuario.

Cuadro 6.7: HU-<07>: Bloquear a un usuario.

HU-<08>	Desbloquear a un usuario
<i>Título</i>	Desbloquear a un usuario en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder desbloquear a un usuario en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero desbloquear a un usuario en la aplicación.
<i>Objetivo</i>	Desbloquear a un usuario.

Cuadro 6.8: HU-<08>: Desbloquear a un usuario.

HU-<09>	Consultar los usuarios bloqueados
<i>Título</i>	Consultar los usuarios que han sido bloqueados en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder consultar los usuarios que han sido bloqueados en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero consultar los usuarios que han sido bloqueados en la aplicación.
<i>Objetivo</i>	Consultar los usuarios bloqueados.

Cuadro 6.9: HU-<09>: Consultar los usuarios bloqueados.

HU-<10>	Registrar un alojamiento
<i>Título</i>	Registrar un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder registrar un alojamiento en la aplicación para otros usuarios puedan hacer uso de él.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero registrar un alojamiento en la aplicación proporcionando diversos datos como nombre, dirección, ciudad, país, facilidades, imágenes, etc...
<i>Objetivo</i>	Registrar un alojamiento.

Cuadro 6.10: HU-<10>: Registrar un alojamiento.

HU-<11>	Consultar los detalles de un alojamiento
<i>Título</i>	Consultar los detalles de un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder ver los detalles de un alojamiento que ha sido registrado en la aplicación para poder consultarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero consultar los detalles de un alojamiento que ha sido registrado en la aplicación.
<i>Objetivo</i>	Consultar los detalles de un alojamiento.

Cuadro 6.11: HU-<11>: Consultar los detalles de un alojamiento.

HU-<12>	Visualizar tus alojamientos
<i>Título</i>	Visualizar tus alojamientos en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder ver los alojamientos que he registrado en la aplicación para poder gestionarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero ver los alojamientos que he registrado en la aplicación.
<i>Objetivo</i>	Visualizar tus alojamientos.

Cuadro 6.12: HU-<12>: Visualizar tus alojamientos.

HU-<13>	Editar los detalles de tu alojamiento
<i>Título</i>	Editar los detalles de tu alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder editar los detalles de los alojamientos que he registrado previamente en la aplicación para poder gestionarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero editar los detalles de un alojamiento que he registrado previamente en la aplicación.
<i>Objetivo</i>	Editar los detalles de tu alojamiento.

Cuadro 6.13: HU-<13>: Editar los detalles de tu alojamiento.

HU-<14>	Cerrar tu alojamiento
<i>Título</i>	Cerrar tu alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder cerrar un alojamiento que he registrado previamente en la aplicación para poder gestionarlo.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero cerrar un alojamiento que he registrado previamente en la aplicación.
<i>Objetivo</i>	Cerrar tu alojamiento.

Cuadro 6.14: HU-<14>: Cerrar tu alojamiento.

HU-<15>	Abrir tu alojamiento
<i>Título</i>	Abrir tu alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder abrir un alojamiento que he registrado y cerrado previamente en la aplicación para poder gestionarlo.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero abrir un alojamiento que he registrado y cerrado previamente en la aplicación.
<i>Objetivo</i>	Abrir tu alojamiento.

Cuadro 6.15: HU-<15>: Abrir tu alojamiento.

HU-<16>	Consultar la disponibilidad de un alojamiento
<i>Título</i>	Consultar la disponibilidad de un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder consultar la disponibilidad de un alojamiento durante un periodo determinado en la aplicación para poder consultarla.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero consultar la disponibilidad de un alojamiento durante un periodo determinado en la aplicación.
<i>Objetivo</i>	Consultar la disponibilidad de un alojamiento en un periodo determinado.

Cuadro 6.16: HU-<16>: Consultar la disponibilidad de un alojamiento.

HU-<17>	Realizar consultas sobre los alojamientos
<i>Título</i>	Realizar consultas sobre los alojamientos en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder realizar consultas sobre los alojamientos en la aplicación usando diferentes parámetros para poder consultarlos.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero realizar consultas sobre los alojamientos en la aplicación usando como parámetros la ubicación, el día de llegada y de fin, número de adultos y de niños, y número de habitaciones.
<i>Objetivo</i>	Realizar consultas sobre los alojamientos usando diferentes parámetros.

Cuadro 6.17: HU-<17>: Realizar consultas sobre los alojamientos.

HU-<18>	Bloquear un alojamiento
<i>Título</i>	Bloquear un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder bloquear un alojamiento en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero bloquear un alojamiento en la aplicación.
<i>Objetivo</i>	Bloquear un alojamiento.

Cuadro 6.18: HU-<18>: Bloquear un alojamiento.

HU-<19>	Desbloquear un alojamiento
<i>Título</i>	Desbloquear un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder desbloquear un alojamiento en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero desbloquear un alojamiento en la aplicación.
<i>Objetivo</i>	Desbloquear un alojamiento.

Cuadro 6.19: HU-<19>: Desbloquear un alojamiento.

HU-<20>	Consultar los alojamientos bloqueados
<i>Título</i>	Consultar los alojamientos que han sido bloqueados en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder consultar los alojamientos que han sido bloqueados en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero consultar los alojamientos que han sido bloqueados en la aplicación.
<i>Objetivo</i>	Consultar los alojamientos bloqueados.

Cuadro 6.20: HU-<20>: Consultar los alojamientos bloqueados.

HU-<21>	Realizar una reserva
<i>Título</i>	Realizar una reserva de un alojamiento en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder realizar una reserva de un alojamiento que ha sido registrado en la aplicación durante un periodo determinado para poder reservarlo.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero reservar un alojamiento que ha sido registrado en la aplicación durante un periodo de tiempo.
<i>Objetivo</i>	Realizar una reserva de un alojamiento durante un periodo determinado.

Cuadro 6.21: HU-<21>: Realizar una reserva.

HU-<22>	Consultar tus reservas
<i>Título</i>	Consultar tus reservas en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder consultar las reservas que he realizado en la aplicación para poder gestionarlas.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero consultar las reservas que he realizado en la aplicación.
<i>Objetivo</i>	Consultar tus reservas.

Cuadro 6.22: HU-<22>: Consultar tus reservas.

HU-<23>	Cancelar una reserva
<i>Título</i>	Cancelar una reserva en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder cancelar una reserva que he realizado en la aplicación con cierta antelación para poder gestionar mis reservas.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero cancelar una reserva que he realizado en la aplicación.
<i>Objetivo</i>	Cancelar una reserva.

Cuadro 6.23: HU-<23>: Cancelar una reserva.

HU-<24>	Valorar una reserva
<i>Título</i>	Valorar una reserva en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder valorar una reserva que he realizado en la aplicación y que ya ha concluido para dejar una opinión sobre el alojamiento.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero valorar una reserva que he realizado en la aplicación y que ya ha concluido.
<i>Objetivo</i>	Valorar una reserva.

Cuadro 6.24: HU-<24>: Valorar una reserva.

HU-<25>	Bloquear una valoración
<i>Título</i>	Bloquear una valoración en la aplicación.
<i>Descripción</i>	Como usuario administrador, quiero poder bloquear una valoración en la aplicación para poder administrarla.
<i>Actor</i>	Usuario administrador.
<i>Acción</i>	Como usuario administrador, quiero bloquear una valoración en la aplicación.
<i>Objetivo</i>	Bloquear una valoración.

Cuadro 6.25: HU-<25>: Bloquear una valoración.

HU-<26>	Seleccionar el tema
<i>Título</i>	Seleccionar el tema en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder seleccionar el tema en la aplicación para poder adecuarlo a mis necesidades.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero seleccionar el tema en la aplicación.
<i>Objetivo</i>	Seleccionar el tema claro o el tema oscuro.

Cuadro 6.26: HU-<26>: Seleccionar el tema.

HU-<27>	Seleccionar el idioma
<i>Título</i>	Seleccionar el idioma de la interfaz en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder seleccionar el idioma de la interfaz en la aplicación para poder adecuarlo a mis necesidades.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero seleccionar el idioma de la interfaz en la aplicación.
<i>Objetivo</i>	Seleccionar el idioma.

Cuadro 6.27: HU-<27>: Seleccionar el idioma.

HU-<28>	Contactar con el servicio técnico
<i>Título</i>	Contactar con el servicio técnico en la aplicación.
<i>Descripción</i>	Como usuario registrado, quiero poder contactar con el servicio técnico en la aplicación a través de un correo electrónico para poder reportar una incidencia.
<i>Actor</i>	Usuario registrado.
<i>Acción</i>	Como usuario registrado, quiero contactar con el servicio técnico en la aplicación.
<i>Objetivo</i>	Contactar con el servicio técnico.

Cuadro 6.28: HU-<28>: Contactar con el servicio técnico.

Capítulo 7

Diseño

La fase de diseño es la etapa en la que se planifican y estructuran las funcionalidades y requisitos previamente definidos para el producto. Durante este proceso, se busca desarrollar un software eficiente y efectivo, priorizando aspectos como la usabilidad, la mantenibilidad y la escalabilidad.

El análisis inicial se centra en la arquitectura general del producto, proporcionando una visión global de su estructura. Posteriormente, se describen los patrones de diseño y los principios fundamentales aplicados durante su desarrollo. Para concluir, se realiza un desglose detallado de cada subsistema.

7.1 Arquitectura general

El sistema desarrollado en este proyecto sigue un modelo cliente-servidor basado en una arquitectura en capas. Este sistema está compuesto de dos subsistemas principales: el subsistema cliente, encargado de la interfaz de usuario y la interacción con los usuarios, y el subsistema servidor, responsable de procesar las solicitudes, gestionar la lógica de negocio y operar sobre la base de datos.

En la figura 7.1 se muestra una representación global del sistema, donde se distinguen los dos subsistemas principales junto con las tecnologías más relevantes utilizadas en cada uno, así como la comunicación entre ellos. En este caso, el subsistema cliente se encarga de proporcionar un punto de acceso para que los usuarios puedan interactuar con la aplicación, mientras el subsistema servidor es el responsable de procesar la lógica de negocio y gestionar los datos relacionados con los usuarios, alojamientos y reservas.

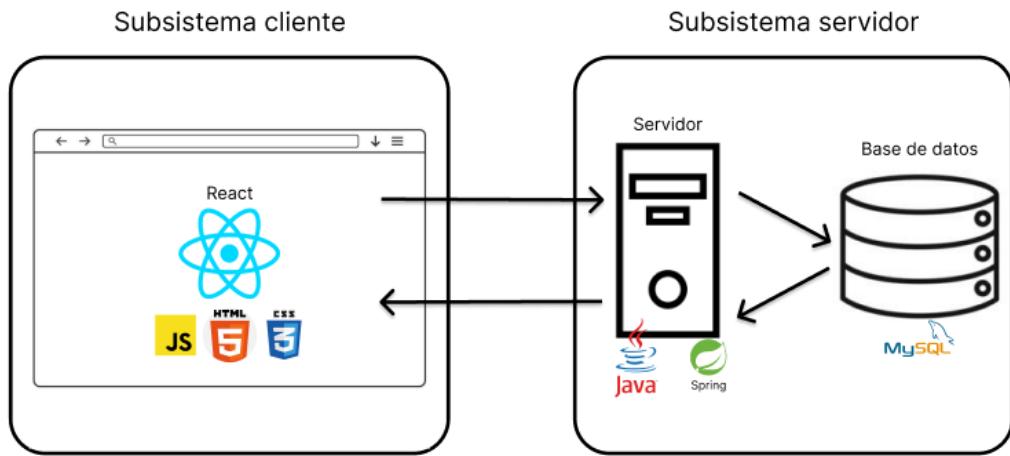


Figura 7.1: Arquitectura general del sistema

7.2 Patrones y principios

La calidad de un producto *software* no solo depende de sus funcionalidades, sino también de su estructura y diseño. Para lograr un código de alta calidad, se emplean patrones y principios de diseño, con el objetivo principal de ofrecer una solución *software* robusta, mantenible y escalable. Estos patrones representan soluciones reutilizables para problemas comunes, adaptándose a las necesidades específicas de cada diseño.

A continuación se muestran los principales patrones y principios de diseño que se emplearon a lo largo del desarrollo de este proyecto.

7.2.1 Diseño por capas

El diseño de una arquitectura en capas [44] permite definir de manera clara y separada las funcionalidades del *software* en distintos niveles. Cada capa se encarga de sus propias funcionalidades, lo que favorece la separación de responsabilidades. Con este enfoque, se busca lograr un alto desacoplamiento entre las capas, de modo que cada una sea independiente de las demás. En este proyecto se estructura la aplicación en una capa de entidades para la representación del modelo de datos, una capa de repositorios para la gestión de la persistencia, una capa de servicios encargada de la lógica de negocio, una capa de controladores para la gestión de solicitudes y una capa para la interfaz de usuario.

7.2.2 Uso de DTOs

Un [DTO](#) [45] es un objeto utilizado para el transporte de datos entre capas. En este proyecto, se emplean para la transmisión y encapsulación de datos entre la capa controlador y la capa cliente. Las interfaces utilizadas en el subsistema cliente se generan a partir de los [DTO\(s\)](#) devueltos por el servidor. El uso de [DTO\(s\)](#) permite separar la lógica de negocio del proceso de transporte de datos.

7.2.3 Repositorios

El uso de repositorios [46] proporciona una capa de abstracción entre la lógica de negocio y las fuentes de datos. A través de interfaces, facilitan la manipulación y el acceso a los datos sin necesidad de conocer los detalles de su almacenamiento y recuperación. En este proyecto, los repositorios se emplean para las operaciones [CRUD](#) de las entidades, así como para realizar consultas [SQL](#) personalizadas y validar la integridad de los datos.

7.2.4 Inyección de dependencias

La inyección de dependencias [47] es una técnica que consiste en proporcionar los objetos o componentes necesarios a una clase sin que esta tenga que crear instancias internamente. En este proyecto, se utiliza para la creación de [bean\(s\)](#) y la inyección de servicios en los controladores.

7.2.5 Principio de encapsulación

El principio de encapsulación [48] se utiliza para controlar el acceso y la modificación de los atributos a los objetos. Esto se logra declarando los atributos como privados y permitiendo su manipulación únicamente a través de métodos públicos, como los *getters* y *setters*.

7.2.6 Principio de inversión de la dependencia

El principio de inversión de la dependencia [49] establece que las clases de alto nivel no deben depender de las clases de bajo nivel, sino que ambas deben depender de abstracciones. Esto se logra mediante la inyección de dependencias [7.2.4](#).

7.2.7 Principio de segregación de interfaces

El principio de segregación de interfaces [50] establece que una clase no debe verse obligada a implementar interfaces que no utiliza. Esto se logra dividiendo las interfaces grandes

en interfaces más pequeñas y específicas, de modo que las clases solo tengan que implementar los métodos que necesitan. En este proyecto, se aplicó este patrón en las interfaces de los servicios.

7.2.8 Principio de responsabilidad única

El principio de responsabilidad única [51] establece que cada clase o módulo del *software* debe tener una única responsabilidad, lo que aumenta la cohesión y genera un código más comprensible y mantenable. En este proyecto, este principio se refleja en la arquitectura, donde cada capa tiene una responsabilidad única.

7.2.9 Principio DRY

El principio **DRY** [52] establece que la información debe tener una representación única y no ambigua. Este principio promueve la reutilización y la escritura de código mantenable y eficiente, evitando la duplicación. De esta manera, se previene que un cambio en un módulo se propague a otros, lo que favorece un desarrollo más eficiente.

7.2.10 Principio KISS

El principio **KISS** [53] establece que las soluciones y los diseños deben mantenerse lo más simples posible, evitando complejidades innecesarias.

7.2.11 Principio YAGNI

El principio **YAGNI** [54] busca evitar la implementación de funcionalidades o características innecesarias para el desarrollo actual del proyecto. Esto previene el gasto de tiempo en el desarrollo de funciones que pueden no ser útiles, reduce la carga de trabajo y disminuye el coste del mantenimiento.

7.3 Arquitectura del subsistema servidor

En esta sección se realizará un análisis de alto nivel del subsistema servidor. Este subsistema se compone de cuatro módulos principales, esenciales para una correcta implementación de la arquitectura cliente-servidor: las entidades persistentes, la capa de repositorios, la capa de servicios y la capa de controladores. En la figura 7.2 se puede observar la distribución general de las capas y los subsistemas a los que pertenecen.

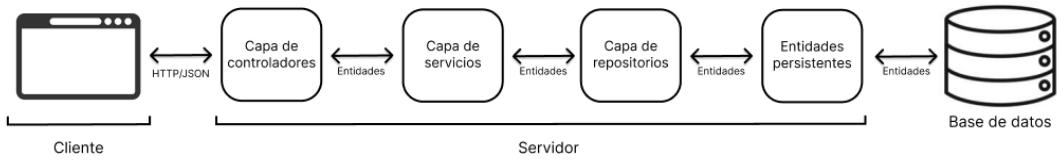


Figura 7.2: Arquitectura general por capas del subsistema servidor

Entidades persistentes

Las entidades persistentes son las encargadas de definir todos los tipos de entidades que se almacenarán en la base de datos de la aplicación. Representan la estructura de datos fundamental del *software*.

Capa de repositorios

La capa de repositorios (o capa de acceso a datos) se comunica directamente con la base de datos para realizar operaciones **CRUD** sobre los recursos, definidos en base a las entidades. Proporciona un nivel adicional de abstracción entre los datos y la lógica de negocio.

Capa de servicios

La capa de servicios define la interfaz de los servicios principales ofrecidos por la aplicación, así como sus implementaciones. Es decir, contiene la lógica de negocio, haciendo uso de los repositorios que interactúan con los recursos solicitados.

Capa de controladores

La capa de controladores se encarga de gestionar la interacción entre el cliente y el servidor. Recibe las solicitudes del usuario, invoca los servicios correspondientes y devuelve las respuestas apropiadas. Su función principal es manejar la comunicación entre la interfaz de usuario y la capa de servicios, asegurando el flujo adecuado de datos y la ejecución de la lógica de negocio.

7.3.1 Entidades persistentes

En la siguiente figura 7.3 se puede ver el diseño de las entidades persistentes. Cada entidad cuenta con un identificador único (*id*), el cual es generado automáticamente por el gestor de la base de datos.

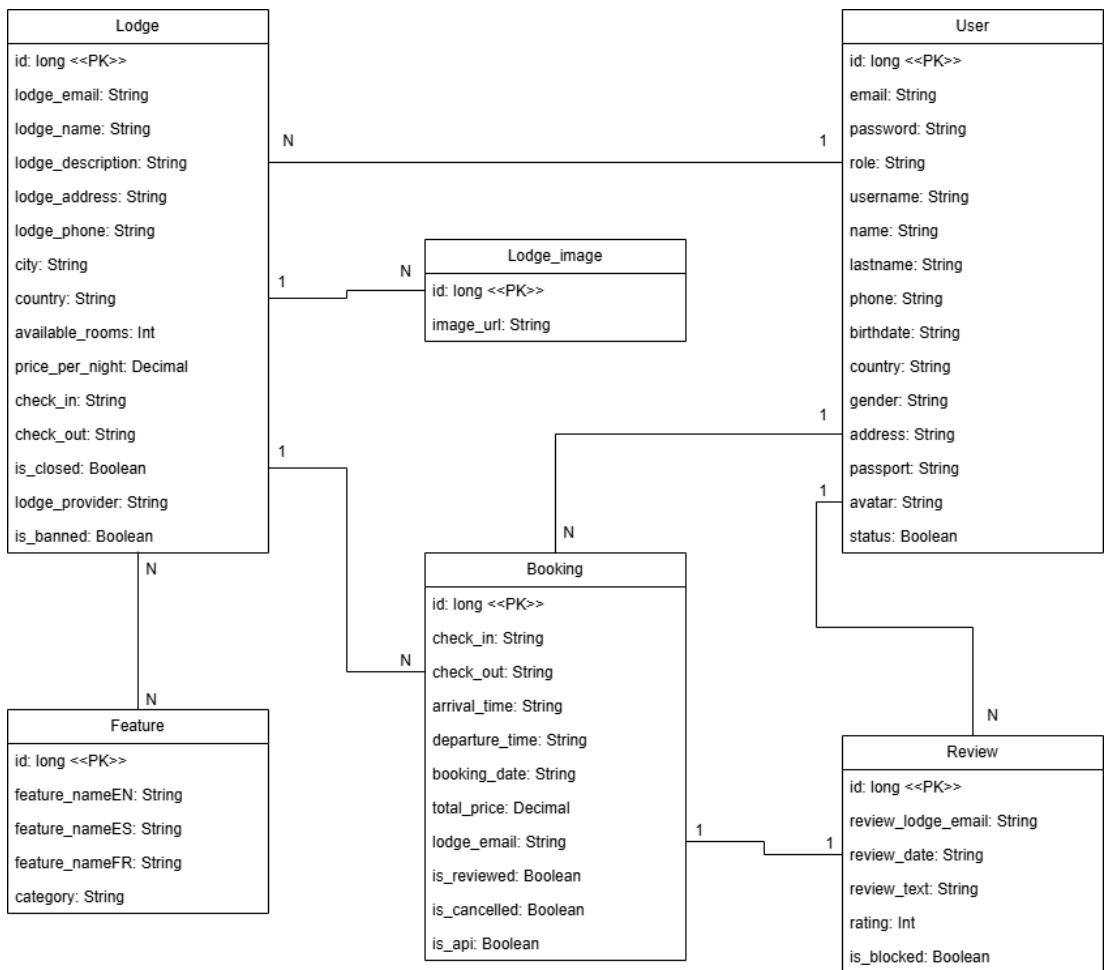


Figura 7.3: Diagrama de entidades

Entidades

- **User**: Entidad que representa al usuario de la aplicación.
- **Lodge**: Entidad que representa a los alojamientos registrados en la aplicación.
- **Lodge_image**: Entidad que representa las imágenes de los alojamientos registrados en la aplicación.
- **Feature**: Entidad que representa las facilidades de los alojamientos.
- **Booking**: Entidad que representa las reservas realizadas para un alojamiento.
- **Review**: Entidad que representa los comentarios escritos por los usuarios sobre las reservas.

Relaciones

- *User-Lodge*: Un usuario puede tener varios alojamientos registrados en la aplicación.
- *User-Booking*: Un usuario puede realizar varias reservas en la aplicación
- *User-Review*: Un usuario puede realizar varios comentarios sobre reservas ya concluidas en la aplicación (un comentario por reserva).
- *Lodge-Lodge_image*: Un alojamiento puede tener asociada varias imágenes en la aplicación.
- *Lodge-Lodge_feature-Feature*: Un alojamiento puede tener varias facilidades en la aplicación.
- *Lodge-Booking*: Un alojamiento puede tener recibir múltiples reservas en la aplicación.
- *Booking-Review*: Una reserva puede recibir un comentario en la aplicación.

El diseño de entidades definido en esta etapa se ha materializado en una base de datos relacional, permitiendo estructurar y gestionar la información de forma eficiente. Este modelo se implementa utilizando MySQL, con la definición específica de tablas, relaciones y restricciones que aseguran la integridad y coherencia de los datos.

Debido a las cardinalidades de algunas relaciones, es necesario crear tablas intermedias en la implementación de la base de datos relacional para almacenar los datos correspondientes. Este aspecto se detallará con mayor profundidad en el siguiente capítulo. A continuación se muestra en la figura 7.4 el diagrama Entidad/Relación.

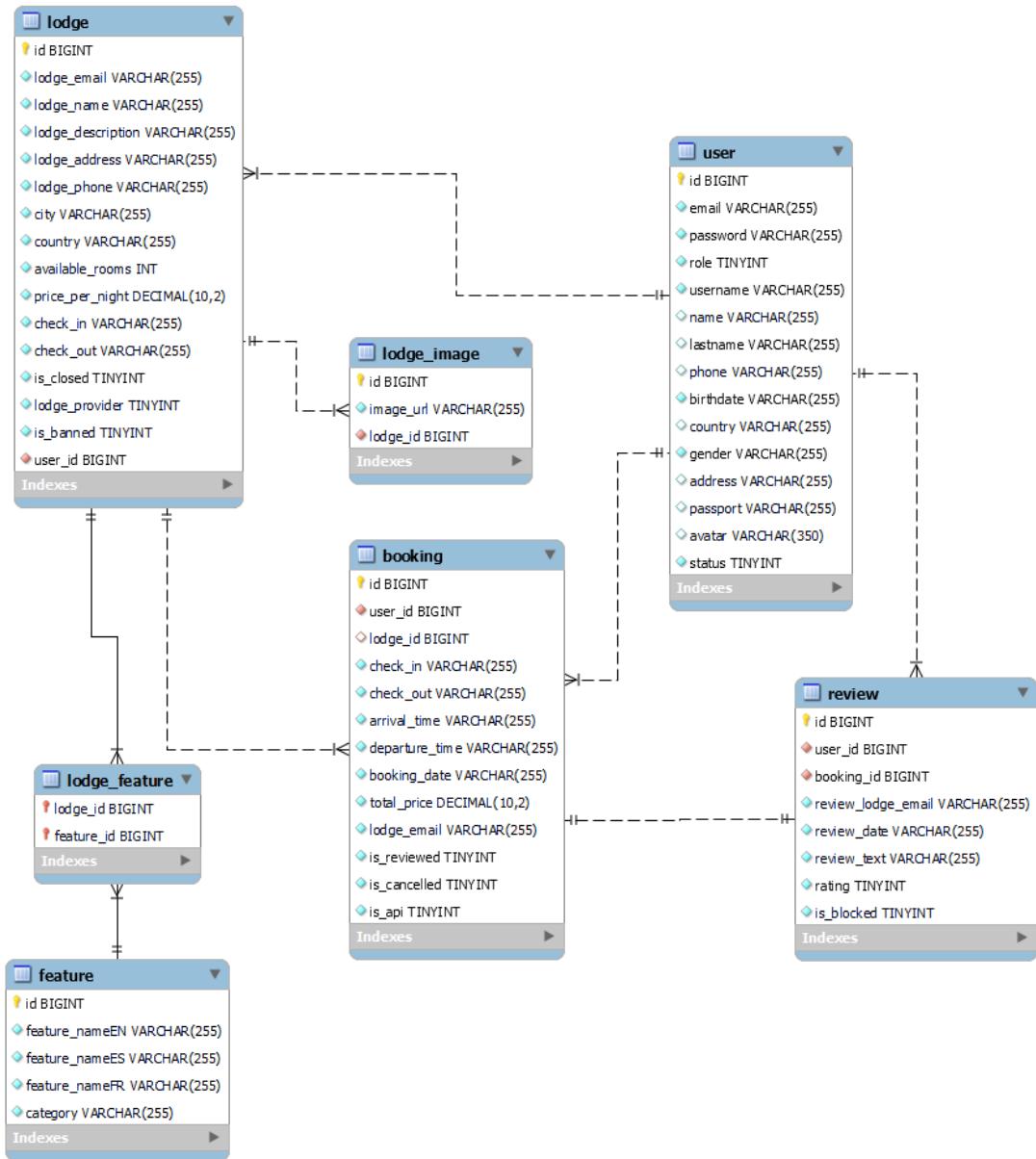


Figura 7.4: Diagrama Entidad/Relación

7.3.2 Capa de repositorios

Esta capa es la encargada de gestionar el acceso y manipulación de los datos. Para ello, en el proyecto se ha utilizado el [framework](#) de persistencia de datos Hibernate junto con la especificación Java Persistence API ([JPA](#)). JPA facilita al programador la interacción con las bases de datos, simplificando y haciendo más eficientes las operaciones de persistencia.

Cada entidad cuenta con un repositorio específico que incluirá las operaciones [CRUD](#), las

cuales serán accesibles desde la capa de servicios. En estos repositorios también se podrán definir consultas personalizadas para satisfacer necesidades específicas, como la obtención de una lista de alojamientos a partir de una ciudad o país y que no estén bloqueados por el administrador. Esto se logrará mediante la anotación `@Query` de JPA, junto con consultas SQL que devuelvan los resultados deseados.

Cada clase de repositorio extiende la interfaz `public interface JpaRepository<T, ID>`, a la cual se le debe proporcionar la entidad que se desea asociar al repositorio "T" y el tipo identificador de la misma "ID". En la figura 7.5 se puede observar un ejemplo de cómo se extiende el repositorio de la entidad "Lodge" desde la interfaz abstracta `JpaRepository`, así como los principales métodos CRUD que esta ofrece para manipular los recursos de la base de datos de la entidad relacionada.

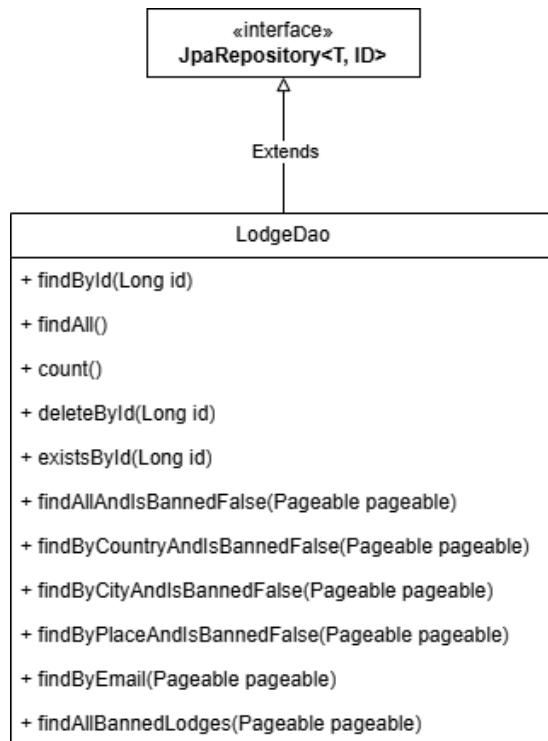


Figura 7.5: Ejemplo de herencia de `JpaRepository`

En la figura 7.6 se puede ver la lista de repositorios de la aplicación.

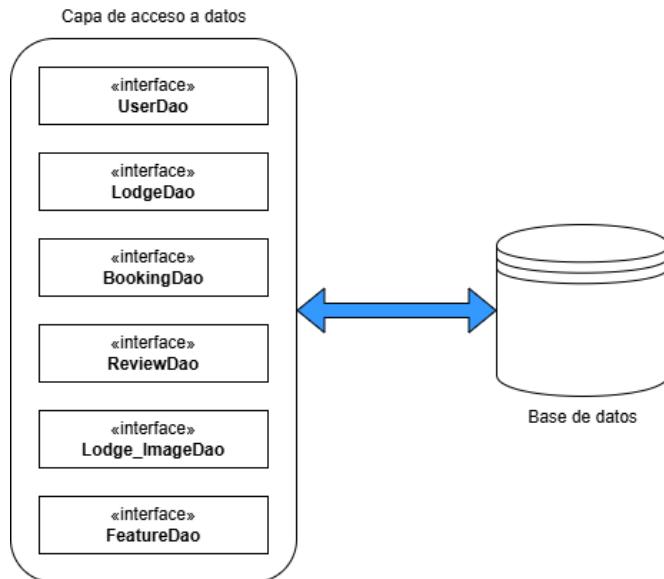


Figura 7.6: Lista de repositorios

A continuación, se presentarán los repositorios en los que se han añadido métodos o consultas específicas, acompañados de una breve explicación de los mismos.

UserDao

- *boolean existsByEmail(String email)*: este método sirve para comprobar la existencia de un usuario a partir de un correo electrónico.
- *Optional<User> findByEmail(String email)*: este método sirve para realizar una consulta personalizada que devuelve un usuario por su *email*
- *Page<User> findAllBannedUsers(Pageable pageable)*: este método sirve para realizar una consulta personalizada que devuelve una página de usuarios que están bloqueados.

LodgeDao

- *Page<Lodge> findAllAndIsBannedFalse(Pageable pageable)*: este método para obtener todos los alojamientos de la aplicación que no están bloqueados.
- *Page<Lodge> findByCountryAndIsBannedFalse(@Param("country") String country, Pageable pageable)*: este método para realizar una consulta personalizada que devuelve una página de alojamientos filtrados por *country* y que no están bloqueados.
- *Page<Lodge> findByCityAndIsBannedFalse(@Param("city") String city, Pageable pageable)*: este método para realizar una consulta personalizada que devuelve una página

de alojamientos filtrados por *city* y que no están bloqueados.

- *Page<Lodge> findByPlaceAndIsBannedFalse(@Param("place") String place, Pageable pageable)*: este método para realizar una consulta personalizada que devuelve una página de alojamientos filtrados por lugar (*city* o *country*) y que no están bloqueados.
- *Optional<Lodge> findByEmail(@Param("email") String email)*: este método sirve para realizar una consulta personalizada que devuelve un alojamiento por su *email*.
- *List<Lodge> findByUserId(Long userId)*: este método sirve para realizar una consulta personalizada que devuelve una lista de alojamientos filtrando por el usuario.
- *Page<Lodge> findAllBannedLodges(Pageable pageable)*: este método sirve para realizar una consulta personalizada que devuelve una página de alojamientos que están bloqueados.

BookingDao

- *List<Booking> findByUserId(Long userId)*: este método sirve para realizar una consulta personalizada que devuelve una lista de reservas filtrando por el usuario.
- *List<Booking> findByLodgeId(Long lodegeId)*: este método sirve para realizar una consulta personalizada que devuelve una lista de reservas filtrando por el alojamiento.

Lodge_ImageDao

- *void deleteAllByLodge(Lodge lodege)*: este método sirve para borrar todas las imágenes a partir de un alojamiento.

ReviewDao

- *List<Review> findByLodgeEmail(@Param("lodgeEmail") String lodegeEmail)*: este método sirve para realizar una consulta personalizada que devuelve una lista de comentarios filtrando por el correo electrónico del alojamiento.

7.3.3 Capa de servicios

La capa de servicios se encarga de definir las funcionalidades de la aplicación que se expondrán al cliente a través de la capa de controladores. Esta capa está compuesta por interfaces que serán utilizadas por los controladores correspondientes, así como por las clases que implementan dichas interfaces, utilizando los repositorios previamente mencionados en la sección 7.3.2. En esta capa se implementa la lógica de negocio de la aplicación. Además, las clases de

implementación realizan la validación de los datos de las solicitudes, lanzando las excepciones necesarias en caso de errores.

En el diagrama de la figura 7.7 se pueden observar las interfaces definidas en el proyecto junto con sus respectivas implementaciones.

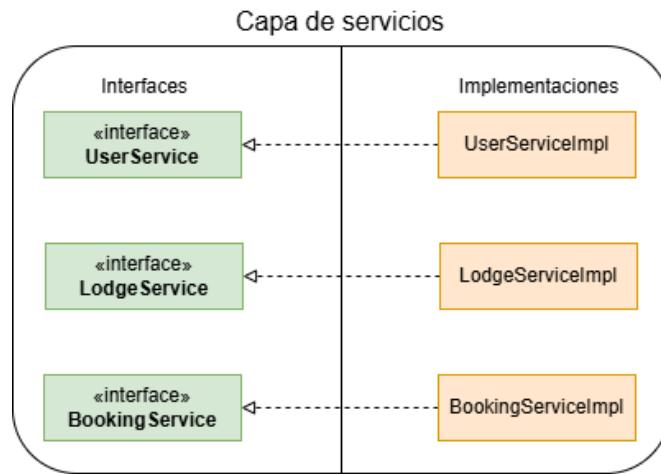


Figura 7.7: Diagrama de servicios

UserService

- *void signUp(User user)*: para crear un nuevo usuario en la aplicación.
- *User login(String email, String password)*: para iniciar sesión en la aplicación.
- *User loginFromId(Long id)*: para iniciar sesión en la aplicación.
- *User updateProfile(Long id, String userName, String name, String lastName, String phone, String birthDateString, String country, String gender, String address, String passport)*: para actualizar el perfil del usuario en la aplicación.
- *void changePassword(Long id, String oldPassword, String newPassword)*: para cambiar la contraseña del usuario en la aplicación.
- *User findById(Long id)*: para encontrar a un usuario a partir del *ID*.
- *User findByEmail(String email)*: para encontrar a un usuario a partir del correo electrónico.
- *void banUser(User admin, String bannedUserEmail)*: para que el administrador pueda bloquear a un usuario en la aplicación

- `void unbannedUser(User admin, String bannedUserEmail)`: para que el administrador pueda desbloquear a un usuario en la aplicación
- `Page<User> findAllBannedUsers(User admin, int page, int size)`: para que el administrador pueda obtener todos los usuarios que han sido bloqueados en la aplicación.

LodgeService

- `Page<Lodge> getLodges(int page, int size)`: para obtener los alojamientos de la aplicación.
- `Page<Lodge> getLodgesByCountry(String country, int page, int size)`: para obtener los alojamientos de la aplicación filtrados por el parámetro `country`.
- `Page<Lodge> getLodgesByCity(String city, int page, int size)`: para obtener los alojamientos de la aplicación filtrados por el parámetro `city`.
- `Page<Lodge> getLodgesByPlace(String place, int page, int size)`: para obtener los alojamientos de la aplicación filtrados por el parámetro lugar (puede ser ciudad o país).
- `Lodge findByEmail(String email)`: para obtener un alojamiento a partir de su correo electrónico en la aplicación.
- `Lodge createLodge(Long userId, String lodgeEmail, String lodgeName, String lodgeDescription, String lodgeAddress, String lodgePhone, String city, String country, int availableRooms, double pricePerNight, String checkIn, String checkOut, List<Long> featureIds, List<String> imageUrls)`: para añadir un alojamiento en la aplicación.
- `Lodge updateLodge(Long userId, String lodgeEmail, String lodgeName, String lodgeDescription, String lodgeAddress, String lodgePhone, String city, String country, int availableRooms, double pricePerNight, String checkIn, String checkOut, List<Long> featureIds, List<String> imageUrls)`: para actualizar los detalles de un alojamiento en la aplicación.
- `List<Feature> getAllFeatures()`: para obtener todas las facilidades disponibles para los alojamientos en la aplicación.
- `List<Lodge> getLodgesById(Long userId)`: para obtener todos los alojamientos a partir de un usuario en la aplicación.
- `void closeLodge(User user, Lodge lodge)`: para cerrar un alojamiento en la aplicación.
- `void openLodge(User user, Lodge lodge)`: para abrir un alojamiento que previamente había sido cerrado en la aplicación.

- *void banLodge(User admin, String bannedLodgeEmail)*: para que el administrador pueda bloquear un alojamiento en la aplicación.
- *void unbanLodge(User admin, String bannedLodgeEmail)*: para que el administrador pueda desbloquear un alojamiento en la aplicación.
- *Page<Lodge> findAllBannedLodges(User admin, int page, int size)*: para que el administrador pueda obtener todos los usuarios que han sido bloqueados en la aplicación.

BookingService

- *Booking createBooking(Long userId, String lodgeEmail, String checkIn, String checkOut, String arrivalTime, String departureTime, double totalPrice, Boolean is_api)*: para que un usuario pueda realizar una reserva en la aplicación.
- *List<Booking> getBookingsByUserId(Long userId)*: para obtener las reservas de un usuario en la aplicación.
- *Boolean checkAvailability(String lodgeEmail, String ArrivalTime, String DepartureTime)*: para comprobar la disponibilidad de un alojamiento en la aplicación.
- *void cancelBooking(Long userId, Long bookingId)*: para cancelar una reserva en la aplicación.
- *Review createReview(Long userId, Long bookingId, String lodgeEmail, String reviewText, int rating)*: para realizar un comentario sobre una reserva en la aplicación.
- *List<Review> getReviewsByLodgeEmail(String lodgeEmail)*: para obtener los comentarios de un alojamiento en la aplicación.
- *Booking getBookingById(Long id)*: para obtener una reserva a partir del parámetro *ID* en la aplicación.
- *Boolean hasReviews(String lodgeEmail)*: para comprobar si un alojamiento tiene reservas en la aplicación.
- *void banReview(User admin, Long reviewId)*: para que el administrador pueda bloquear un comentario en la aplicación.

7.3.4 Capa de controladores

La capa de controladores o de acceso a los servicios se encarga de proporcionar los puntos de entrada que gestionarán las solicitudes **HTTP** realizadas por los clientes, redirigiéndolas

a los servicios correspondientes. Los controladores de esta aplicación permitirán cuatro tipos de peticiones:

- **GET**: para obtener recursos.
- **POST**: para añadir recursos.
- **PUT**: para modificar recursos.
- **DELETE**: para eliminar recursos.

Una vez procesada la solicitud del cliente, los controladores enviarán las respuestas correspondientes. El conjunto de controladores, debidamente anotados con la anotación `@RestController`, conformará la [API](#) del servidor, que está diseñada bajo un enfoque [REST](#). A través de esta [API](#) se gestionarán las peticiones provenientes del subsistema cliente.

En esta capa se implementa la autenticación utilizando *tokens JWT*. Además, mediante el uso de reglas de SpringSecurity y anotaciones específicas, se gestionará el control de acceso. Por otro lado, mediante el uso de anotaciones `@Autowired` de Spring, se realizará la inyección de dependencias de forma automática, permitiendo a los controladores acceder a los servicios requeridos. A continuación, se presentan los controladores disponibles del servidor junto con sus *endpoints* correspondientes.

UserController

Este controlador se encarga de las peticiones relacionadas con la gestión de usuarios, está formado por los siguientes *endpoints*:

- `/api/users/<email>` (GET): para obtener un usuario a partir de su correo electrónico en la aplicación.
- `/api/users/signUp` (POST): para registrar un nuevo usuario en la aplicación.
- `/api/users/login` (POST): para que un usuario inicie sesión en la aplicación.
- `/api/users/loginFromServiceToken` (POST): para iniciar sesión a un usuario en la aplicación.
- `/api/users/updateUser` (PUT): para que un usuario pueda actualizar su perfil en la aplicación.
- `/api/users/changePassword` (POST): para que un usuario pueda cambiar su contraseña en la aplicación.

- `/api/users/banUser/<email>` (POST): para que el administrador pueda bloquear a un usuario en la aplicación.
- `/api/users/unbanUser/<email>` (POST): para que el administrador pueda desbloquear a un usuario en la aplicación.
- `/api/users/findAllBannedUsers` (GET): para que el administrador pueda obtener todos los usuarios que han sido bloqueados en la aplicación.

LodgeController

Este controlador se encarga de las peticiones relacionadas con la gestión de alojamientos y facilidades, está formado por los siguientes *endpoints*:

- `/api/lodges/` (GET): para obtener los alojamientos de la aplicación.
- `/api/lodges/by-country` (GET): para obtener los alojamientos de la aplicación filtrados por el parámetro *country*.
- `/api/lodges/by-city` (GET): para obtener los alojamientos de la aplicación filtrados por el parámetro *city*.
- `/api/lodges/by-place` (GET): para obtener los alojamientos de la aplicación filtrados por el parámetro lugar (ya sea país o ciudad).
- `/api/lodges/<email>` (GET): para obtener un alojamiento en la aplicación a partir de su correo electrónico.
- `/api/lodges/createLodge` (POST): para que un usuario pueda añadir un alojamiento en la aplicación.
- `/api/lodges/getFeatures` (GET): para obtener todas las facilidades disponibles para los alojamientos en la aplicación.
- `/api/lodges/myLodges` (GET): para obtener los alojamientos de un usuario en la aplicación.
- `/api/lodges/closeLodge/<email>` (POST): para que un usuario pueda cerrar un alojamiento en la aplicación
- `/api/lodges/openLodge/<email>` (POST): para que un usuario pueda abrir un alojamiento que previamente había cerrado en la aplicación.
- `/api/lodges/updateLodge` (PUT): para que un usuario pueda actualizar los detalles de su alojamiento en la aplicación.

- `/api/lodges/banLodge/<email>` (POST): para que el administrador pueda bloquear un alojamiento en la aplicación.
- `/api/lodges/unbanLodge/<email>` (POST): para que el administrador pueda desbloquear un alojamiento en la aplicación.
- `/api/lodges/findAllBannedLodges` (GET): para que el administrador pueda obtener todos los alojamientos que han sido bloqueados en la aplicación.

BookingController

Este controlador se encarga de las peticiones relacionadas con la gestión de reservas y comentarios, está formado por los siguientes *endpoints*:

- `/api/bookings/myBookings` (GET): para que un usuario pueda obtener sus reservas en la aplicación.
- `/api/bookings/createBooking` (POST): para que un usuario pueda realizar una reserva en la aplicación.
- `/api/bookings/checkAvailability` (GET): para que un usuario pueda comprobar la disponibilidad de un alojamiento entre determinadas fechas en la aplicación.
- `/api/bookings/cancelBooking/<id>` (POST): para que un usuario pueda cancelar una reserva en la aplicación.
- `/api/bookings/reviewBooking/<id>` (POST): para que un usuario pueda valorar una reserva que ya ha concluido con un comentario en la aplicación.
- `/api/bookings/getTheBooking/<id>` (GET): para obtener una reserva en la aplicación.
- `/api/bookings/hasReviews` (GET): para comprobar si en un alojamiento se han realizado comentarios en la aplicación.
- `/api/bookings/getReviews` (GET): para obtener los comentarios de un alojamiento en la aplicación.
- `/api/bookings/banReview/<id>` (POST): para que el administrador pueda bloquear un comentario en la aplicación.

ImageController

Este controlador se encarga de las peticiones relacionadas con la gestión de imágenes del usuario y de los alojamientos, está formado por los siguientes *endpoints*:

- `/api/images/uploadImage` (POST): para que un usuario pueda establecer un avatar en la aplicación.
- `/api/images/uploadLodgeImage` (POST): para que los alojamientos puedan establecer sus imágenes en la aplicación.

7.4 Arquitectura del subsistema cliente

El subsistema cliente es el responsable de proporcionar a los usuarios acceso a los servicios implementados en el servidor mediante una interfaz gráfica. En este proyecto se utilizó React como plataforma de desarrollo, caracterizada por el uso de componentes reutilizables como pueden ser botones, ventanas modales o desplegables. En el diagrama 7.8 se muestra un ejemplo del funcionamiento del sistema.

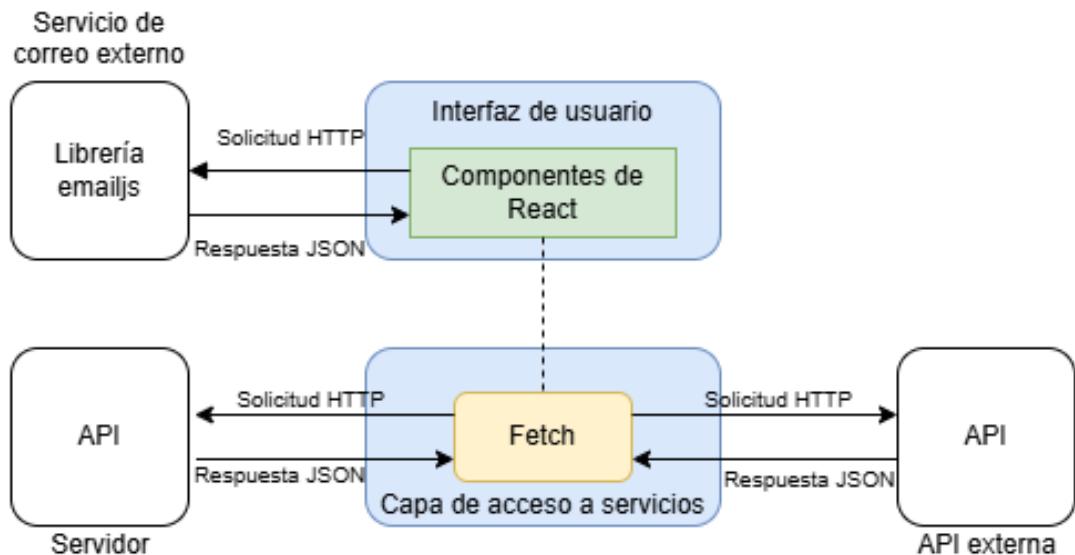


Figura 7.8: Arquitectura general del subsistema cliente

El subsistema cliente se divide en dos partes: la interfaz de usuario y la capa de acceso a servicios. Esta separación permite una distribución clara de responsabilidades, lo que contribuye a la independencia entre la gestión de datos y la presentación en la **IU**.

Interfaz de usuario

La interfaz de usuario se encarga de la representación visual de la aplicación, incluyendo todos los componentes con los que el usuario puede interactuar, como botones, ventanas modales o desplegables, entre otros. En la interfaz de usuario se utilizan componentes tanto

de la librería NextUI mencionada en la sección 3.1.2 como componentes personalizados con el objetivo de ofrecer una experiencia que se ajuste a los requisitos específicos. En el diseño de la interfaz de usuario se considerarán los siguientes objetivos:

- **Usabilidad:** la interfaz de usuario debe ser intuitiva y fácil de usar.
- **Eficiencia:** la interfaz de usuario debe realizar las tareas de manera eficiente, evitando interacciones innecesarias del usuario.
- **Adaptabilidad:** la interfaz se debe adaptar a los diferentes tipos de dispositivos.
- **Estética:** la interfaz debe atractiva visualmente para los usuarios.

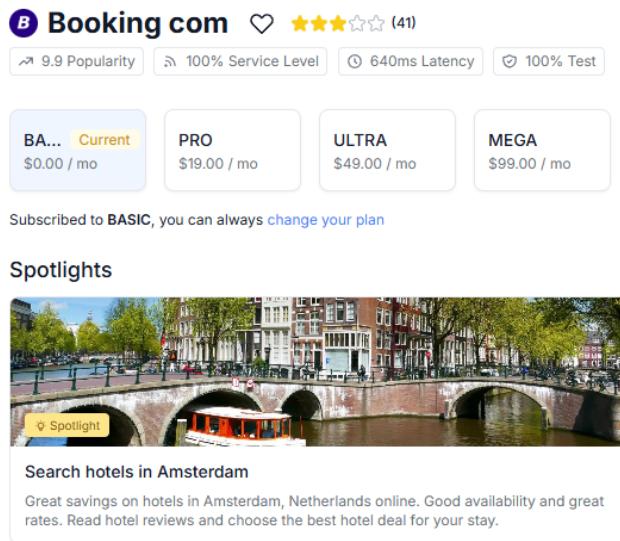
Capa de acceso a servicios

En esta capa, los componentes realizan peticiones [HTTP](#) a la [API](#) del servidor utilizando la función nativa *fetch*, que permite gestionar tanto las respuestas exitosas como los errores. Además, los componentes también pueden realizar peticiones a una [API](#) externa cuando los usuarios realicen ciertas acciones, como por ejemplo, buscar alojamientos en un lugar determinado. En este contexto, *fetch* actúa como la capa que proporciona acceso a los servicios implementados y expuestos en el servidor, los cuales contienen la lógica de negocio. En la figura 7.8 se muestra un diagrama básico que representa la interacción entre los subsistemas mediante esta función, así como los tipos de mensajes que manejan.

Integración con un servicio web externo

Esta es una parte de la capa de acceso a servicios, concretamente cuando los componentes realizan peticiones [HTTP](#) a la [API](#) externa.

En este proyecto, la [API](#) se seleccionó de la página web rapidapi [55], esta plataforma ofrece una gran cantidad de [API](#)(s), que cubren diversas necesidades y servicios. En este proyecto, se utilizó la [API](#) de Booking [56] que cuenta con diferentes planes mensuales, cada uno adaptado a distintas necesidades de uso. En el caso de este proyecto, se escogió el plan gratuito que permite 530 solicitudes al mes.



API Overview

Support: tipsters@rapi.one / t.me/api_tipsters

Booking.com is available in 43 languages and offers more than 28 million reported accommodation listings, including over 6.2 million homes, apartments, and other unique places to stay. Wherever you want to go and whatever you want to do, Booking.com makes it easy and supports you with 24/7 customer support.

Find all hotels, view prices, photos of the hotels, reviews. Find car rental deals. You can make a website like: hotels.com, booking.com, agoda.com

Figura 7.9: API de Booking

Una vez escogida la API y el plan mensual, para integrar la API con el proyecto, se necesita una clave de autenticación (*API_KEY*) que se obtiene al autenticarse en el servicio. Esta clave se utiliza a la hora de realizar las peticiones HTTP a la API externa elegida. A continuación se muestran los *endpoints* de la API que se utilizan en la aplicación.

- */v1/hotels/locations?locale=en-gb&name=<destination>*: para obtener el identificador del lugar donde el usuario quiere realizar la búsqueda.
- */v1/hotels/search?page_number=<pageN>&adults_number=<adults>&room_number=<rooms>&units=metric&checkout_date=<checkOut>&dest_id=<destId>&filter_by_currency=EUR&dest_type=<destType>&checkin_date=<checkIn>&order_by=popularity&locale=en-gb*: para obtener los alojamientos a partir de los parámetros especificados por el usuario.
- */v1/hotels/data?hotel_id=<hotel_id>&locale=<locale>*: para obtener los detalles de un alojamiento a partir de su identificador y del idioma seleccionado por el usuario en la aplicación.
- */v2/hotels/details?locale=<locale>&checkin_date=<checkIn>&hotel_id=<hotel_id>¤cy=EUR&checkout_date=<checkOut>*: para obtener los detalles específicos de un alojamiento a

partir del identificador, el idioma, y de las fechas de llegada y de salida.

- `/v1/hotels/photos?hotel_id=<hotel_id>&locale=en-gb`: para obtener las fotos de un alojamiento a partir de su identificador.

Servicio de correo externo

En este proyecto, se decidió integrar un servicio de correo externo para la funcionalidad de contactar con el servicio técnico, se usó la librería EmailJS [57]. Esta librería permite gestionar el envío de correos electrónicos de manera eficiente sin necesidad de configurar un servidor propio.

Este sistema se utiliza para que los usuarios puedan reportar incidencias de forma directa a través de un formulario en la interfaz de usuario. Los datos proporcionados en el formulario, como el correo electrónico del usuario que quiere mandar una incidencia y el mensaje, son procesados por EmailJS y enviados automáticamente al buzón de correo electrónico configurado para este propósito: `deepdivesl.contact@gmail.com`.

La elección de este servicio se fundamenta en su fácil integración, seguridad y rapidez en la transmisión de la información, optimizando así el proceso de comunicación entre los usuarios y el servicio técnico de la aplicación.

Capítulo 8

Implementación y pruebas

En este capítulo, se profundizará en los sistemas desarrollados, analizando con mayor detalle sus aspectos a nivel técnico.

8.1 Subsistema servidor

En esta sección se tratarán aspectos relevantes de la implementación del subsistema servidor, acompañados de una breve explicación sobre su proceso de compilación y puesta en funcionamiento.

8.1.1 Modelo de datos

El modelo de datos representa de forma estructurada y abstracta la información almacenada en el sistema, específicamente en la base de datos. Este modelo describe cómo se organizan y relacionan las diferentes entidades de la aplicación.

En la figura 8.1 se muestra la distribución de las tablas generadas al crear las entidades del modelo, anotadas con `@Entity` de JPA. Cada tabla tiene un nombre y una lista de atributos con su correspondiente tipo de dato. Para definir los atributos de cada entidad, se utiliza la anotación `@Column`, que asigna una columna en la tabla de datos `@Table` a dicho atributo, estableciendo el tipo de dato adecuado.

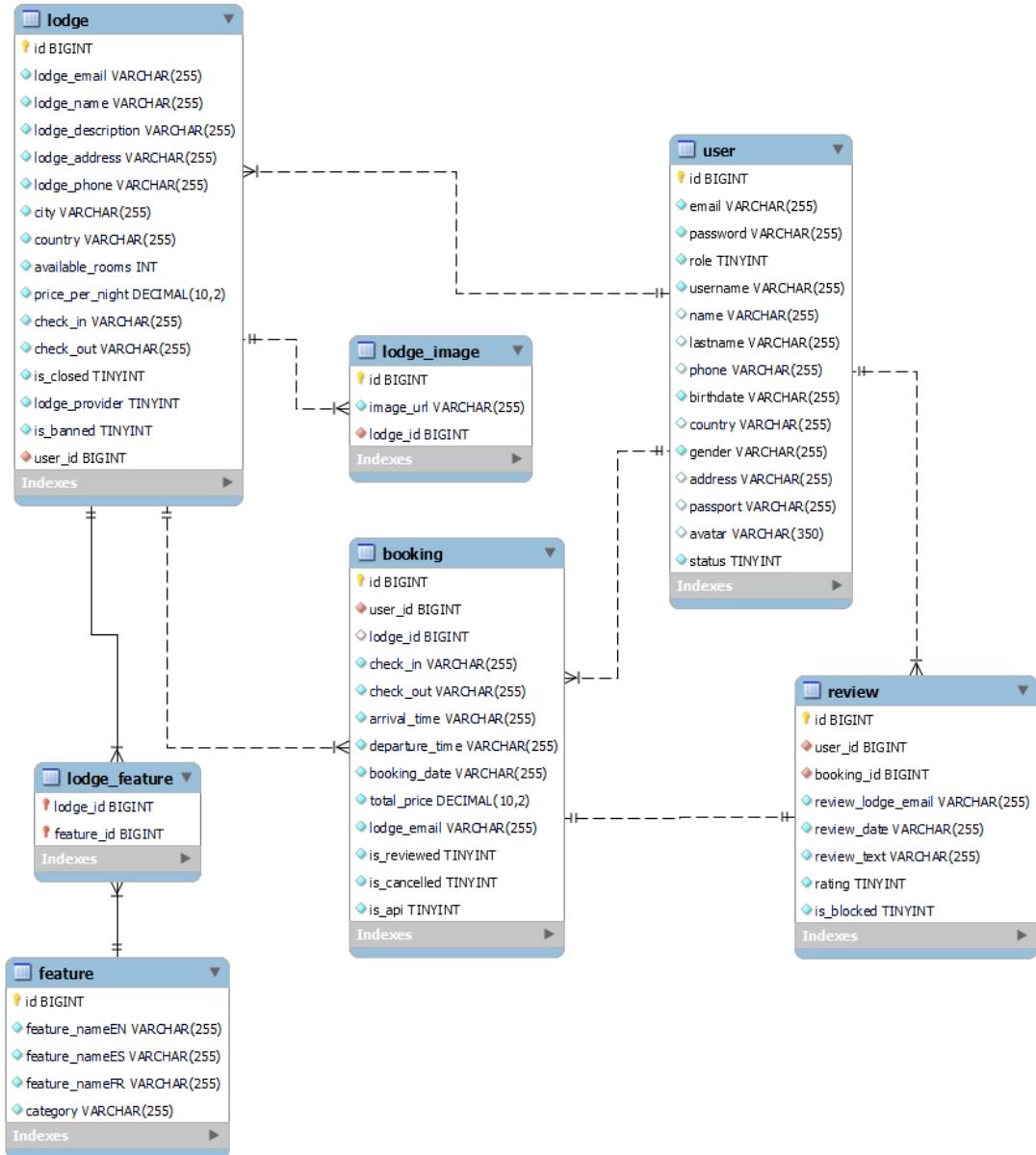


Figura 8.1: Modelo de datos

Para establecer las relaciones entre tablas, se emplean las anotaciones `@OneToMany`, `@ManyToOne` y `@ManyToMany`, según la cardinalidad requerida entre las entidades. Tal como se observa en el modelo de datos, se ha creado una tabla auxiliar para gestionar la información en las relaciones de muchos a muchos (anotadas con `@ManyToMany`), como se puede ver en la relación entre *Lodge* y *Feature*. Esta tabla almacenará las facilidades que tiene cada alojamiento.

8.1.2 Jerarquía de paquetes

En la figura 8.2 se presenta la jerarquía de paquetes que conforma el servidor, la cual implementa parte de la arquitectura en capas.

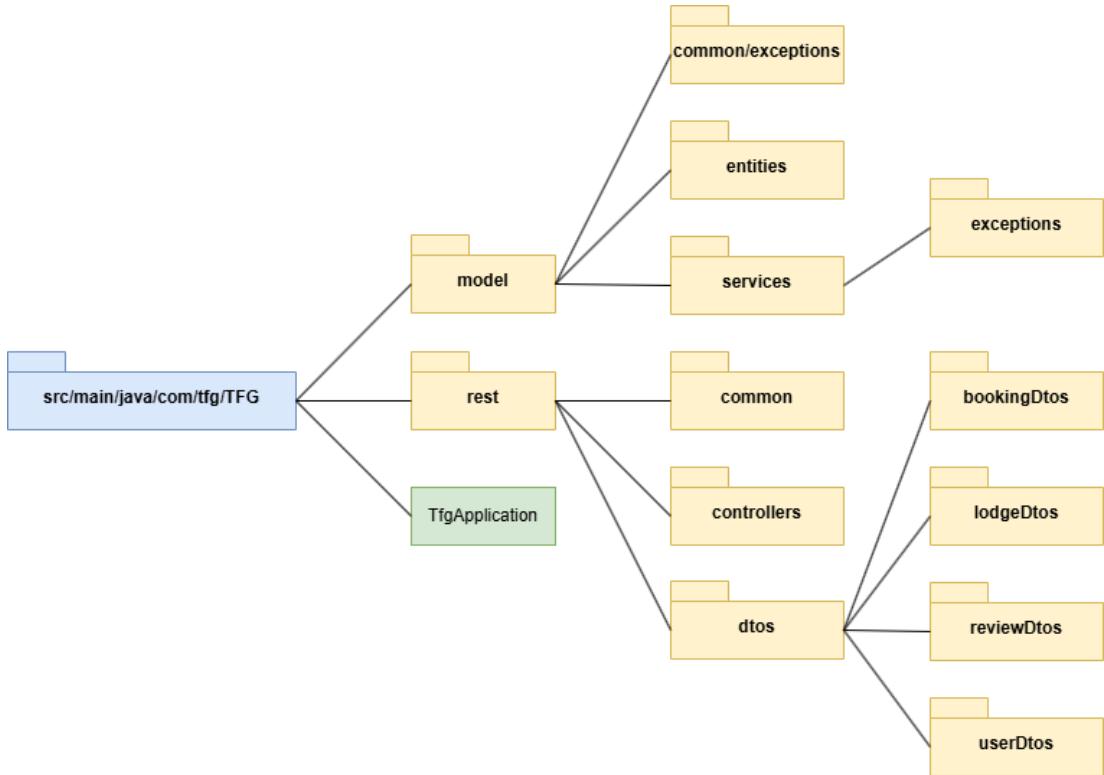


Figura 8.2: Jerarquía de paquetes del subsistema servidor

- **Paquete *model***: incluye las entidades, que representan los objetos persistentes de la aplicación y los servicios asociados, que implementan la lógica de negocio.
- **Paquete *rest***: incluye los controladores y los objetos de transferencia de datos (**DTO**) utilizados para gestionar las solicitudes **HTTP**.
- **Paquete *common/exceptions***: incluye las excepciones básicas utilizadas en la aplicación.
- **Paquete *entities***: incluye las definiciones de todas las entidades persistentes de la aplicación y todos los **DAO(s)** de dichas entidades, gestionando el acceso y manipulación de los recursos de la base de datos.
- **Paquete *services***: incluye las interfaces y las implementaciones de los servicios ofrecidos, los cuales serán expuestos a través de la **API** del servidor.

- **Paquete *exceptions***: incluye las excepciones personalizadas utilizadas en la aplicación
- **Paquete *common***: contiene utilidades generales como validadores, clases auxiliares relacionadas con la seguridad y con **JWT** y funcionalidades relacionadas con **CORS**.
- **Paquete *controllers***: incluye las clases encargadas de gestionar las solicitudes **HTTP** sobre los recursos de la aplicación.
- **Paquete *dtos***: incluye los paquetes que representan los **DTO**(s), los cuales son utilizados para transferir datos entre la capa cliente y la capa controlador.
- **Paquete *bookingDtos***: contiene el **DTO** de reserva y su respectivo **conversor**.
- **Paquete *lodgeDtos***: contiene los **DTO** de alojamiento, imagen de alojamiento y facilidad de alojamiento, y sus respectivos **conversor**(s).
- **Paquete *reviewDtos***: contiene el **DTO** de comentario y su respectivo **conversor**.
- **Paquete *userDtos***: contiene el **DTO** de usuario y su respectivo **conversor**.

La clase *TfgApplication* es la encargada de realizar la configuración automática de la aplicación en SpringBoot. Para ello, está anotada *@SpringBootApplication* y se encarga de realizar la mayoría de las configuraciones necesarias para iniciar la aplicación en el servidor.

8.2 Subsistema cliente

En esta sección se analizarán los principales componentes utilizados en la implementación del subsistema cliente. Estos componentes están diseñados para proporcionar una experiencia positiva al usuario, cada uno con su funcionalidad específica.

8.2.1 Jerarquía de paquetes

En el diagrama de la figura 8.3 se presenta la distribución de paquetes del subsistema cliente.

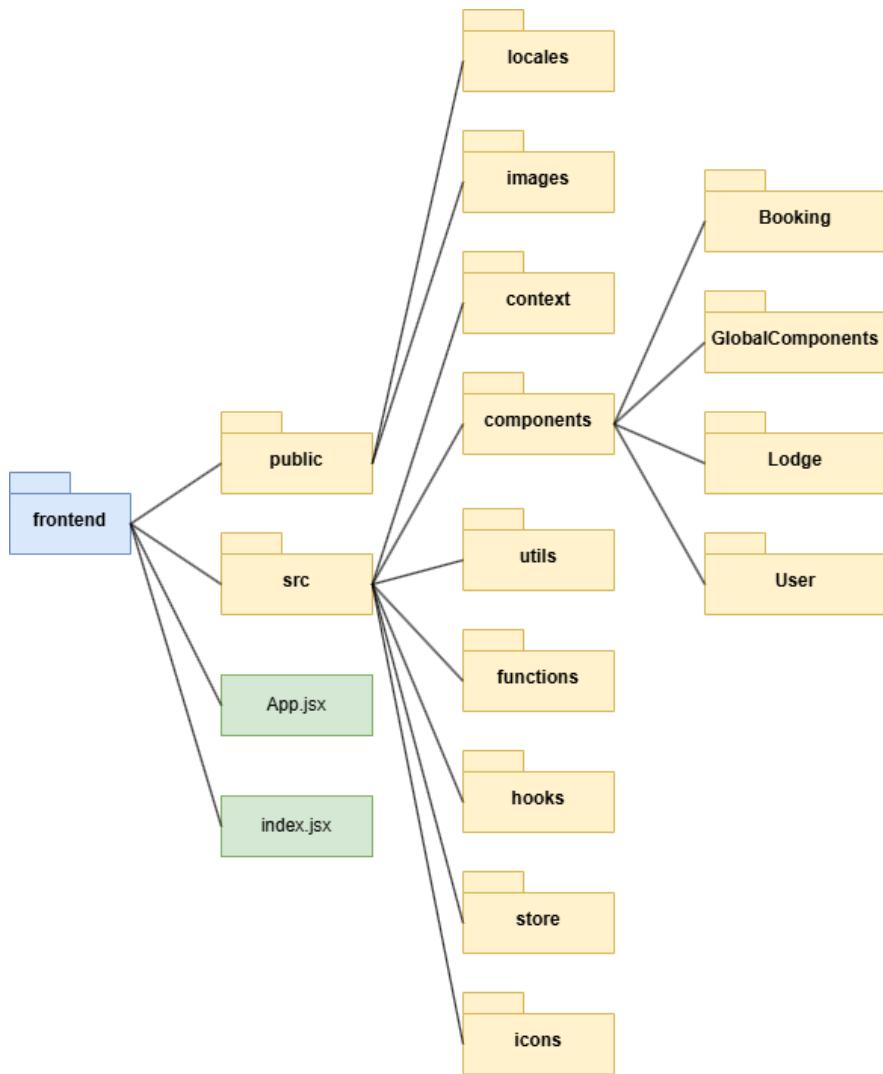


Figura 8.3: Jerarquía de paquetes del subsistema cliente

- **Paquete `frontend`**: directorio raíz del subsistema cliente.
- **Paquete `public`**: incluye archivos estáticos accesibles directamente.
- **Paquete `src`**: incluye el código fuente del subsistema cliente de la aplicación, incluyendo componentes, vistas, estilos, contextos, utilidades y cualquier otro archivo necesario para la lógica y estructura del proyecto.
- **Paquete `locales`**: incluye archivos `JSON` utilizados para la internacionalización.
- **Paquete `images`**: incluye las imágenes y logotipos utilizados en la aplicación.
- **Paquete `context`**: paquete que centraliza la gestión del estado global mediante contex-

tos, como el contexto del usuario y el contexto del tema.

- **Paquete *components*:** incluye los componentes reutilizables de la interfaz de usuario, que representan elementos y bloques estructurales de la aplicación.
- **Paquete *utils*:** incluye archivos JavaScript con utilidades generales y constantes, como listas y mapas utilizados en la aplicación.
- **Paquete *functions*:** incluye funciones utilitarias o auxiliares que realizan tareas específicas en diversas partes de la aplicación.
- **Paquete *hooks*:** incluirá *hook*(s) personalizados para la aplicación, que se utilizan con el objetivo de abstraer y reutilizar una lógica de negocio específica en diversos componentes.
- **Paquete *store*:** incluye las *store*(s) que gestionan diferentes estados específicos de la aplicación, como los usuarios o alojamientos bloqueados, se utilizó la librería Zustand [58].
- **Paquete *icons*:** incluye todos los archivos *SVG* utilizados en la aplicación, empleando estos elementos gráficos para mejorar la interfaz de usuario.
- **Paquete *Booking*:** incluye los componentes relacionados con la funcionalidad de reservas y comentarios.
- **Paquete *GlobalComponents*:** incluye los componentes presentes en toda la aplicación, como la selección de idioma o tema.
- **Paquete *Lodge*:** incluye los componentes relacionados con la funcionalidad de alojamientos.
- **Paquete *User*:** incluye los componentes relacionados con la funcionalidad de usuarios.

El archivo *index.jsx* es fundamental en una aplicación React, ya que actúa como el punto de entrada principal. En este archivo se estructuran los componentes esenciales de la aplicación, como se puede ver en la figura 8.4 esta envuelta en estos proveedores.

```

import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter } from "react-router-dom";
import "./config/i18next.config";
import { AuthContextProvider } from "./context/AuthContext";
import { ThemeContextProvider } from "./context/ThemeContext";
import process from "process";

window.process = process;
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <BrowserRouter>
    <ThemeContextProvider>
      <AuthContextProvider>
        <App />
      </AuthContextProvider>
    </ThemeContextProvider>
  </BrowserRouter>
);

```

Figura 8.4: Archivo index.jsx

- **Componente *BrowserRouter***: gestiona la navegación entre distintas vistas de la aplicación.
- **Componente *ThemeContextProvider***: gestiona el estado relacionado con el tema visual.
- **Componente *AuthContextProvider***: administra el estado de autenticación del usuario, garantizando que la información de acceso esté disponible en toda la aplicación.

El archivo *App.jsx* es el componente que define la estructura de las rutas y la navegación entre las diferentes vistas. Utiliza el componente *Routes* de React Router [59] para gestionar las rutas y redirigir a los usuarios según su estado de autenticación.

8.3 Compilación y puesta en funcionamiento

8.3.1 Servidor

La compilación y ejecución de la aplicación en la parte del servidor se realiza siguiendo los pasos que se describen a continuación:

- Configurar las propiedades deseadas en el archivo de configuración *application.properties*, tales como el método de actualización de las tablas que se utilizará para la creación y

actualización de las tablas con Hibernate y [JPA](#), la [URL](#) de conexión, entre otros.

- Generar el archivo [JAR](#) ejecutable que se genera mediante el uso del comando `mvn clean package`.
- Transferir el archivo [JAR](#) generado al entorno de producción previamente configurado.
- Ejecutar la aplicación [SpringBoot](#) con el comando `java -jar <nombre_del_archivo>.jar`.

8.3.2 Cliente

La compilación y ejecución de la aplicación en la parte del cliente se realiza siguiendo los pasos que se describen a continuación:

- Configurar correctamente las [URL\(s\)](#) que hacen referencia a la [API](#) del servidor en producción.
- Ejecutar la aplicación con el comando `npm start`.

8.4 Pruebas

Las pruebas son técnicas empíricas mediante las cuales se analiza de manera objetiva la calidad y confiabilidad del producto implementado. Las pruebas tienen los siguientes objetivos principales:

- La validación y verificación de los requisitos funcionales y no funcionales.
- La identificación de defectos y errores.
- El aseguramiento de la calidad del producto *software*.

A lo largo del desarrollo del producto se realizan tres tipos de pruebas: pruebas de integración, pruebas sobre la [API](#) del servidor y pruebas de aceptación por parte del cliente sobre la aplicación final.

8.4.1 Pruebas de integración

Durante el proceso de desarrollo de la capa modelo de la aplicación, se realizaron pruebas para verificar el correcto funcionamiento de todos los componentes del *software*, con el fin de lograr una mejor integración del conjunto.

Estas pruebas tienen como objetivo garantizar que los componentes interactúen sin problemas y produzcan resultados consistentes y coherentes. Las clases sometidas a las pruebas son las clases que implementan la lógica de negocio principal, es decir, los servicios.

Estas pruebas fueron implementadas para ser ejecutadas de forma automática utilizando JUnit. En la figura 8.5 se puede observar un diagrama que refleja la estructura de las pruebas empleando `@JUnit`.

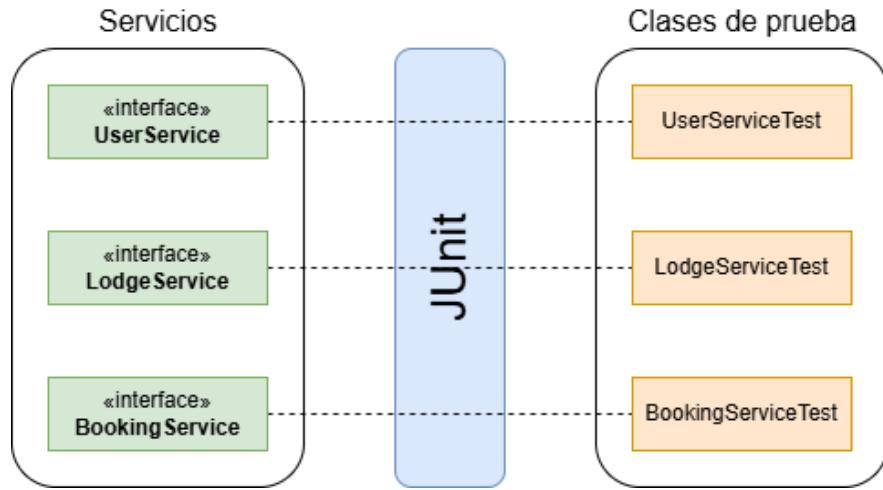


Figura 8.5: Diagrama de pruebas de integración

Para crear un contexto de pruebas, se emplea la anotación `@SpringBootTest`, que configura un entorno de prueba completo para la aplicación, permitiendo que se realicen pruebas sobre los componentes reales, como los servicios y las entidades, sin necesidad de simular las operaciones. De este modo, se garantiza que todas las pruebas se ejecuten en un entorno controlado e independiente, sin modificar la base de datos real gracias a la anotación `@Transactional`, que asegura que los cambios realizados durante la ejecución de las pruebas se reviertan al finalizar cada método de prueba.

La anotación `@Test` se utiliza para señalar que el método contiene una prueba que será ejecutada por JUnit cuando se realicen las pruebas unitarias de la clase correspondiente. JUnit proporciona el uso de aserciones, como por ejemplo `assertEquals`, `assertTrue` o `assertThrows`, entre otras, para verificar el correcto funcionamiento de los métodos. Estas aserciones son declaraciones que permiten comprobar si el resultado de llamar a una operación o método coincide con el valor esperado, comparándolo con el valor real obtenido. Además, se validan los casos en los que se espera que el sistema devuelva algún tipo de excepción, utilizando la aserción `assertThrows`, la cual permite verificar que las excepciones esperadas sean lanzadas correctamente cuando se ejecutan ciertos métodos bajo condiciones específicas.

Para analizar la cobertura de las pruebas y garantizar un alto porcentaje de código probado, se puede tomar como referencia la figura 8.6. En ella, se observa que todos los servicios alcanzan una alta cobertura, la restante se debe a que no se puede probar algunos métodos que

funcionan a partir de acciones en el cliente sobre la API externa, como por ejemplo, reservar un alojamiento que provee la API de Booking.

com.tfg.TFG.model.services

Element	Missed Instructions	Cov.
BookingServiceImpl	81 %	
LodgeServiceImpl	85 %	
UserServiceImpl	88 %	

Figura 8.6: Cobertura de las pruebas de integración

8.4.2 Pruebas sobre los controladores

Para comprobar el correcto funcionamiento de la API del servidor, se utiliza la herramienta software Postman 3.2, el cual permite realizar peticiones a los diferentes *endpoints*. Esto permite hacer de forma gráfica solicitudes HTTP y verificar el correcto funcionamiento de las funcionalidades, así como los tipos de respuesta devueltos.

En la figura 8.7 se puede observar un ejemplo de estas pruebas, en este caso realizando una actualización del perfil de un usuario. Para agilizar estos procesos Postman también permite crear colecciones de peticiones, lo cual es muy útil ya que permite reutilizar las solicitudes en cada prueba de un *endpoint*.

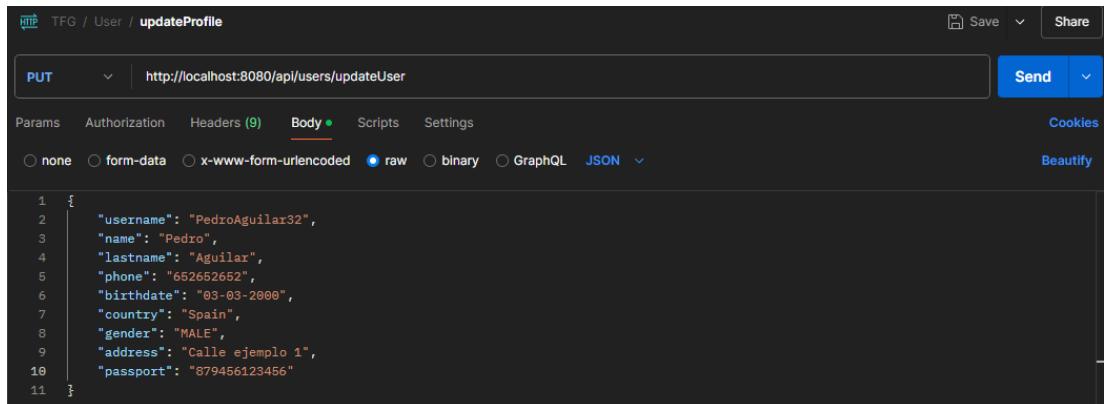


Figura 8.7: Cobertura de las pruebas de integración

8.4.3 Pruebas de aceptación

Las pruebas de aceptación pertenecen a las etapas finales del ciclo de vida de un proyecto, antes de su entrega final y puesta en producción. Estas pruebas son realizadas manualmente

por los usuarios finales o clientes, con el fin de verificar que el sistema cumple con los requisitos y expectativas planeados previamente. Para ello, el usuario interactúa con el *software* a través de la interfaz de usuario, con el fin de comprobar el correcto funcionamiento de las funcionalidades.

Dado que se trata de un proyecto en el que no existe un "cliente real", los usuarios finales serán el alumno y el tutor del proyecto. A continuación se muestran unos ejemplos de pruebas de aceptación en el caso de la gestión de usuarios:

- Comprobar que el usuario final se puede registrar en la aplicación.
- Comprobar que en el caso de que el usuario no proporcione parámetros válidos a la hora de registrarse verá un mensaje de error específico y no se llevará a cabo el registro.
- Comprobar que el usuario puede iniciar y cerrar sesión.
- Comprobar que el cambio de idioma y de tema funcionan correctamente.
- Comprobar que el usuario puede actualizar su avatar y su perfil.
- Comprobar que en el caso de que el usuario no proporcione parámetros válidos a la hora de actualizar el perfil verá un mensaje de error específico y no se llevará a cabo la actualización de los datos del usuario.
- Comprobar que se ve correctamente las diferentes pantallas de la aplicación en concordancia con los datos del usuario.

La conclusión exitosa de estas pruebas confirma la satisfacción del cliente, lo que permite considerar el producto como completado y, en consecuencia, proceder a su puesta en producción.

Capítulo 9

Conclusiones y futuras líneas de trabajo

En este proyecto se llevó a cabo el desarrollo de una aplicación web diseñada para la gestión de alojamientos y reservas. La plataforma permite a los usuarios registrar, administrar y reservar hospedajes de manera eficiente, ofreciendo una experiencia intuitiva tanto para propietarios como huéspedes.

9.1 Conclusiones

- El proyecto ha finalizado con éxito, cumpliendo con los requisitos establecidos inicialmente. Durante su desarrollo, se ha buscado que la solución ofrezca una interfaz gráfica intuitiva y de fácil uso. La aplicación proporciona todas las funcionalidades necesarias para gestionar tu alojamiento, así como realizar reservas en otros alojamientos y posteriormente valorar el resultado de la estancia.
- Durante el desarrollo de este proyecto, se han aplicado los conocimientos adquiridos a lo largo del grado, lo que ha permitido poner a prueba las capacidades del alumno. Este proceso ha servido para consolidar habilidades técnicas y demostrar la capacidad de enfrentarse a desafíos propios del ámbito profesional.
- Se ha adquirido la capacidad de trabajar siguiendo un enfoque basado en metodologías ágiles, lo que resultará de gran utilidad para el alumno al desempeñarse en entornos profesionales donde se apliquen metodologías completas.
- Este proyecto implicó el uso de tecnologías completamente nuevas para el alumno, lo que permitió aumentar su base de conocimientos y fortalecer sus capacidades de cara al futuro. Este proceso no sólo facilitó la adquisición de nuevas competencias técnicas,

sino que también fomentó la adaptabilidad y la capacidad del aprendizaje continuo del alumno.

- Se mejoraron las habilidades previas al desarrollo gracias a la incorporación de nuevas herramientas de *software*. Además, se adquirieron mayores competencias en el diseño de interfaces gráficas, un ámbito en el que se disponía de un conocimiento limitado.
- El proyecto representó un desafío personal para el alumno, quien tuvo que desarrollar una aplicación desde cero por primera vez. Este logro pone de manifiesto su capacidad de enfrentar y superar retos en el ámbito del desarrollo de *software*, demostrando habilidades como la planificación, la resolución de problemas y la adaptación a nuevas exigencias técnicas.

En definitiva, el proyecto cumplió con los objetivos planteados y proporcionó una experiencia valiosa de aprendizaje y crecimiento personal. La aplicación final se convertirá en una herramienta útil como base para futuros proyectos, sirviendo como referencia y punto de partida para desarrollos más avanzados. Este resultado no sólo refleja el esfuerzo y dedicación invertidos, sino también la consolidación de competencias técnicas y profesionales.

9.2 **Futuras líneas de trabajo**

Tras las pruebas finales y la puesta en producción, se sugieren las siguientes posibles mejoras para el producto:

- Se propone la adición de nuevos lenguajes como parte del proceso de internacionalización, con el objetivo de ampliar la accesibilidad y adaptabilidad de la aplicación a usuarios de diversos contextos culturales y lingüísticos.
- Se plantea la amplificación de los métodos de pago disponibles, con el fin de ofrecer mayor flexibilidad y comodidad a los usuarios.
- Se propone implementar un proceso gradual para importar los alojamientos servidos por la [API](#), consolidando así una única fuente de datos que garantice la coherencia, precisión y eficiencia en la gestión de la información.
- Se sugiere ampliar las funcionalidades de la gestión de usuarios mediante la implementación de un sistema de recuperación de contraseñas y la opción de almacenar métodos de pago dentro de la aplicación, con el objetivo de mejorar la experiencia del usuario, garantizando mayor seguridad y optimizar los procesos de autenticación y transacciones.

Apéndices

Apéndice A

Material adicional

En esta sección se incluye un manual de usuario que sirve de guía para la ejecución de distintas funcionalidades implementadas en el proyecto. La información se organiza por módulos:

- Módulo de gestión de usuarios.
- Módulo de gestión de alojamientos.
- Módulo de gestión de reservas.
- Módulo de gestión de comentarios.
- Módulo de administrador.

A.1 Gestión de usuarios

A.1.1 Autenticación y registro de usuarios

Al entrar en la aplicación el usuario encontrará la pantalla de autenticación que se muestra en la figura A.1 donde podrá introducir sus datos para autenticarse.

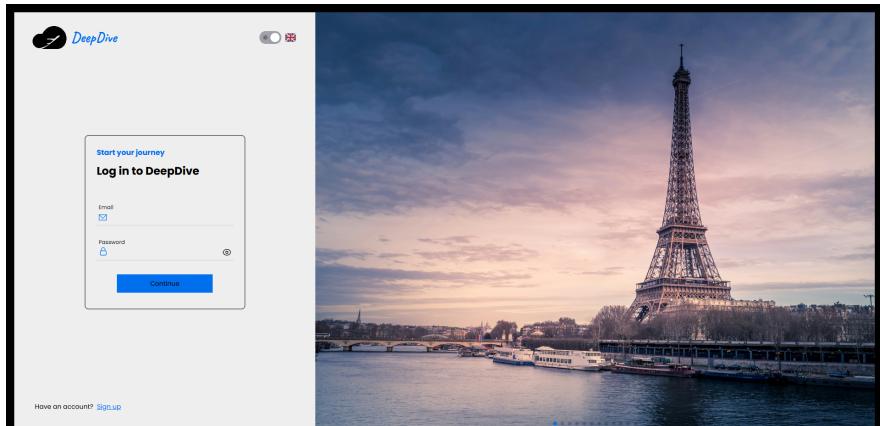


Figura A.1: Autenticación

Si el usuario no dispone de una cuenta en la aplicación, deberá seleccionar la opción *sign up* y el sistema le mostrará el formulario para registrarse mostrado en la figura A.2

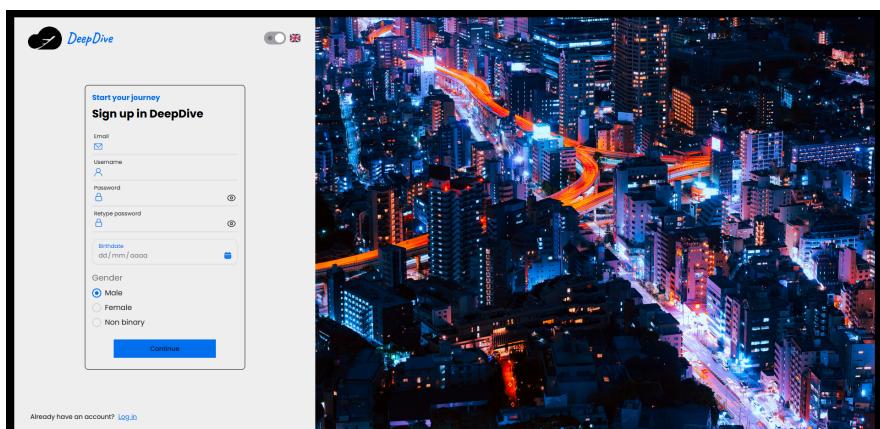


Figura A.2: Formulario de registro

Una vez autenticado, los usuarios podrán ver la pantalla de inicio de la aplicación mostrada en la imagen A.3

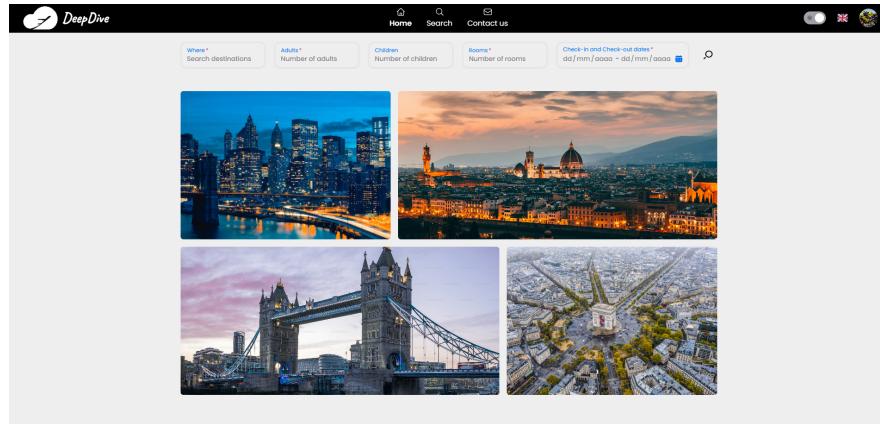


Figura A.3: Página principal de la aplicación

A.1.2 Selección de tema e idioma

Estos dos componentes A.4 se utilizarán para que los usuarios puedan seleccionar tanto el tema como el idioma de la aplicación, estos componentes también estarán presentes en la barra de navegación lo que permite a los usuarios cambiar el tema y el idioma en cualquier sección de la aplicación.

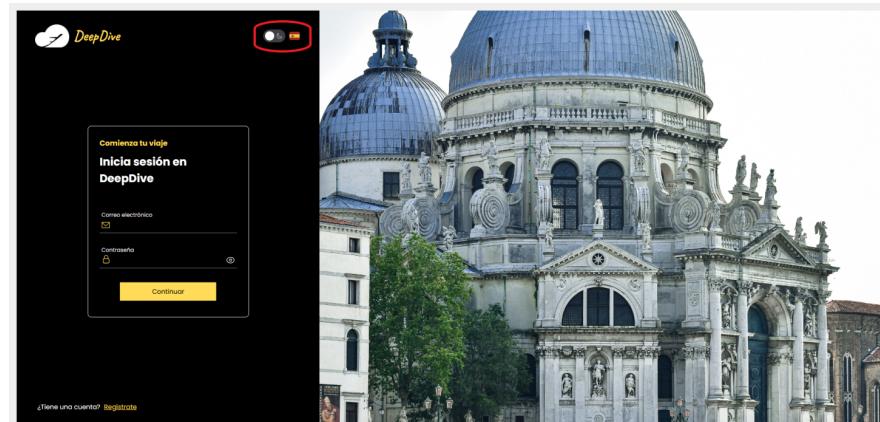


Figura A.4: Selección de tema e idioma

A.1.3 Desplegable de usuario

Este componente permite al usuario acceder a una gran variedad de opciones, se puede ver en la imagen A.5 donde se ubica en la aplicación.

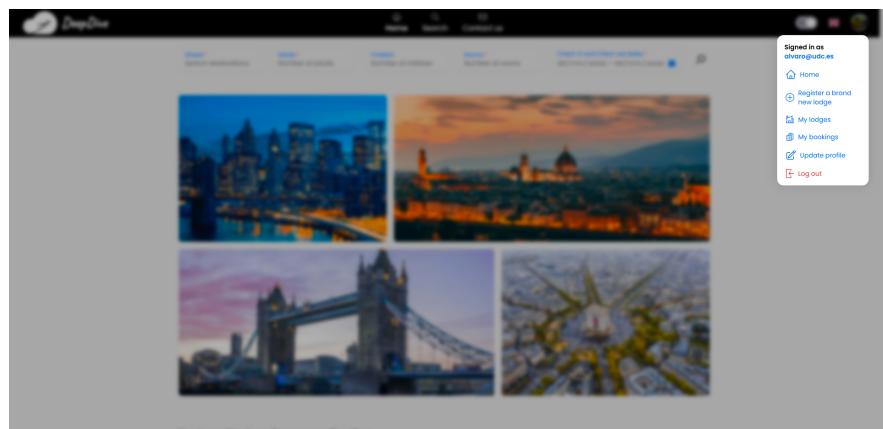


Figura A.5: Desplegable de usuario

Concretamente en la figura A.6 se pueden apreciar las opciones que tiene, como ir al a página principal de la aplicación, añadir un nuevo alojamiento, consultar tus alojamientos o reservas, visualizar o actualizar tu perfil y cerrar sesión.

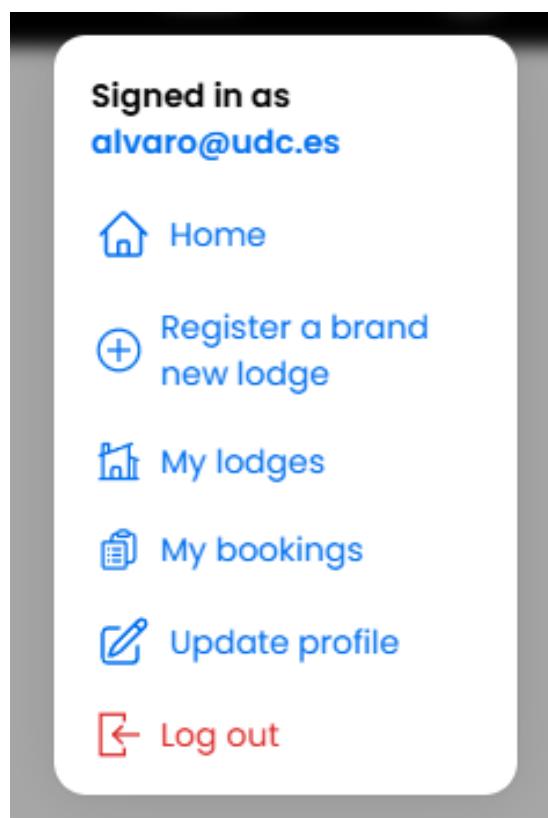


Figura A.6: Desplegable de usuario

A.1.4 Visualización y actualización del perfil

El acceso a la visualización y actualización del perfil se realiza a través del menú desplegable de usuario en la barra de navegación mencionado anteriormente, que se muestra en la imagen A.5.

Una vez seleccionado la opción de *Update profile*, el usuario podrá consultar y modificar los parámetros de su perfil como se puede ver en la imagen A.7

The screenshot displays the 'Update profile' interface of the DeepDive application. It features a sidebar with navigation links like Home, Search, Contact us, and a user icon. The main area has a title 'Update profile'. On the left, there's a form with fields for Username (Alvaro11), Name (Alvaro), Surname (Martinez), Phone (981825252), Address (Address), Passport (123456789), and Birthdate (01/01/2000). On the right, there's a circular placeholder for a profile picture showing a motorcycle. Below the picture are gender selection buttons: Male (selected), Female, and Non binary. At the bottom right is a 'Change password' button, and at the very bottom is a large blue 'Update profile' button.

Figura A.7: Visualizar y actualizar perfil

A.1.5 Cambio de contraseña

A través de la página de visualizar el perfil mostrada en la imagen A.7, el usuario podrá cambiar la contraseña pulsando el botón *Change password*, lo cual redirigirá al usuario al siguiente formulario A.8.

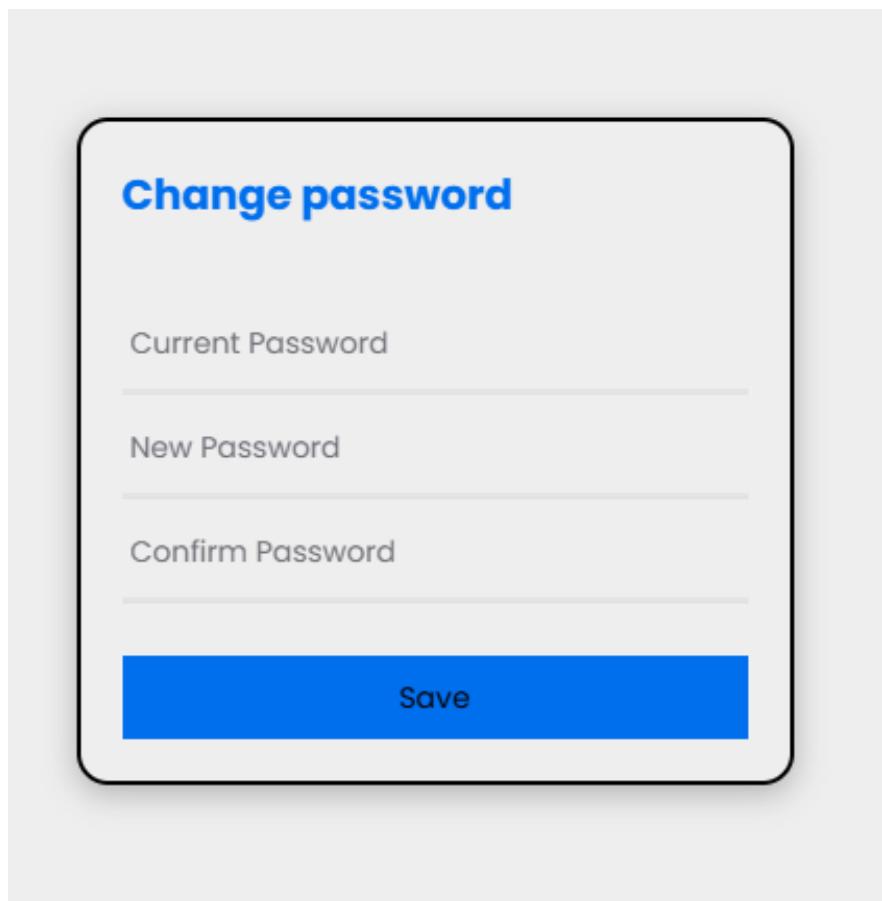


Figura A.8: Cambio de contraseña

A.1.6 Cierre de sesión

El usuario podrá cerrar sesión seleccionando la opción *Log out* en el desplegable del usuario mostrado previamente A.5.

A.1.7 Contactar con el servicio técnico

El usuario podrá contactar con el servicio técnico a través de la opción de *Contact us* en la barra de navegación, lo cual mostrará el siguiente formulario A.9 con los parámetros necesarios para enviar el correo electrónico al servicio técnico.

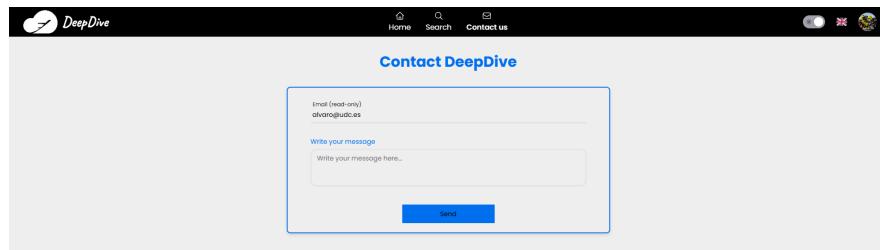


Figura A.9: Contactar con el servicio técnico

A.1.8 Detalles del perfil de un usuario

En determinadas secciones de la aplicación, como en la consulta de los detalles de un alojamiento, se proporciona un enlace que permite acceder a la información del usuario asociado. Como se puede observar en la imagen A.10 el usuario podrá consultar algunos datos, mientras que otros, los más comprometidos, permanecen privados.

The screenshot displays a user profile page titled 'User profile details'. On the left, there is a list of personal information:

- Email: LodgeOwner@apibooking.com
- Username: BookingOwner
- Name: Gleen
- Lastname: D. Fogel
- Country: United States of America
- Address: Private, check your profile.
- Passport: Private, check your profile.
- Birthdate: 03-04-1966

On the right, there is a section for gender selection with three radio buttons: Male (selected), Female, and Non binary. To the right of the gender section is a large blue circular logo with the word 'Booking' in white.

Figura A.10: Detalles de un usuario

A.2 Gestión de alojamientos

A.2.1 Página principal de la aplicación

Como se puede observar en la figura A.11, se muestran cuatro imágenes de destinos populares, donde se ejecutará una búsqueda automática en función de cual selecciones.

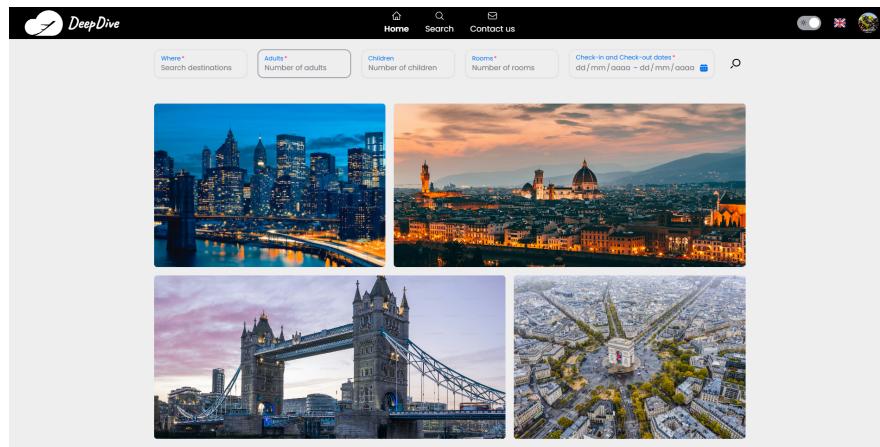


Figura A.11: Página principal de la aplicación

Un resultado de ejemplo de esta búsqueda es el que se puede observar en la figura A.12, donde se seleccionó la imagen de Florencia (Italia), y se realizó una búsqueda de alojamientos en dicho país, como se observa en los resultados, el campo *provider* indica el origen de la fuente de datos del alojamiento. En este sentido, DeepDive corresponde a los alojamientos registrados en la aplicación, mientras que Booking hace referencia a los resultados proporcionados por la [API](#).

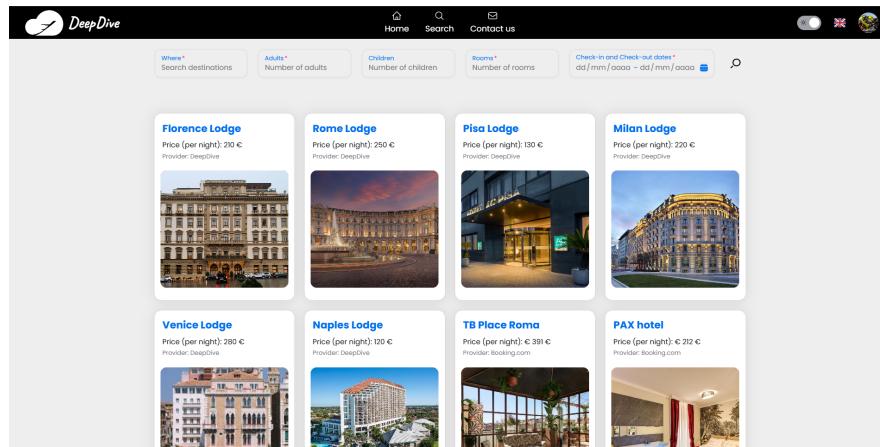


Figura A.12: Resultado de una búsqueda de alojamientos

Si el usuario desplaza la página hacia abajo A.13, se encontrará con una lista de alojamientos registrados en la aplicación.

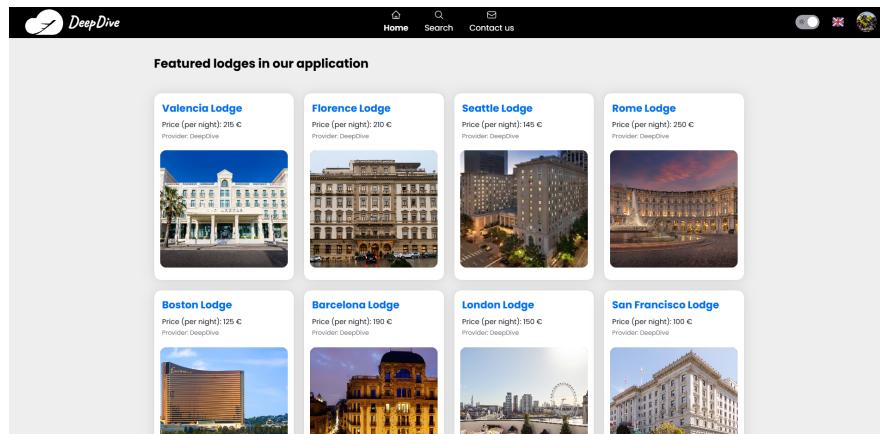


Figura A.13: Página principal de la aplicación

A.2.2 Búsqueda de alojamientos

La aplicación contiene una barra de búsqueda para que el usuario pueda realizar sus consultas personalizadas, como podemos observar en la imagen A.14 hay dos tipos de campos, los obligatorios (*where, adults, rooms* y *check-in and check-out dates*) y los opcionales (*children*). Estos campos son los que se utilizan para optimizar el tiempo del usuario permitiéndole hacer búsquedas específicas basadas en sus intereses.

The search bar interface consists of five input fields and a search icon:

- Where***: Search destinations
- Adults***: Number of adults
- Children**: Number of children
- Rooms***: Number of rooms
- Check-in and Check-out dates***: dd/mm/aaaa – dd/mm/aaaa

Figura A.14: Barra de búsqueda

En las imágenes A.15, A.16 y A.17 se pueden observar los resultados de una búsqueda, y al final de la página encima del Footer se puede observar una barra de navegación entre páginas para ver más resultados.

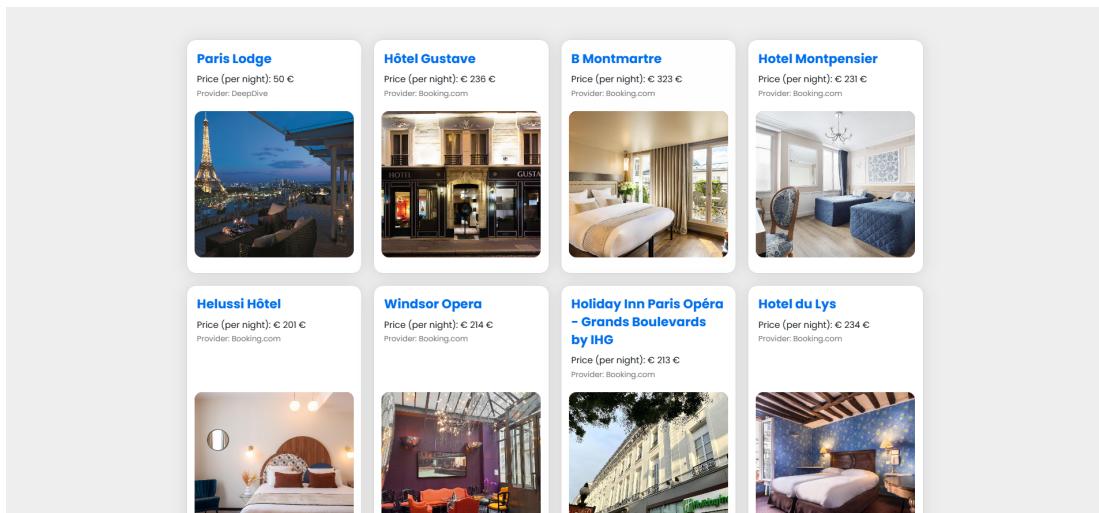


Figura A.15: Ejemplo de búsqueda

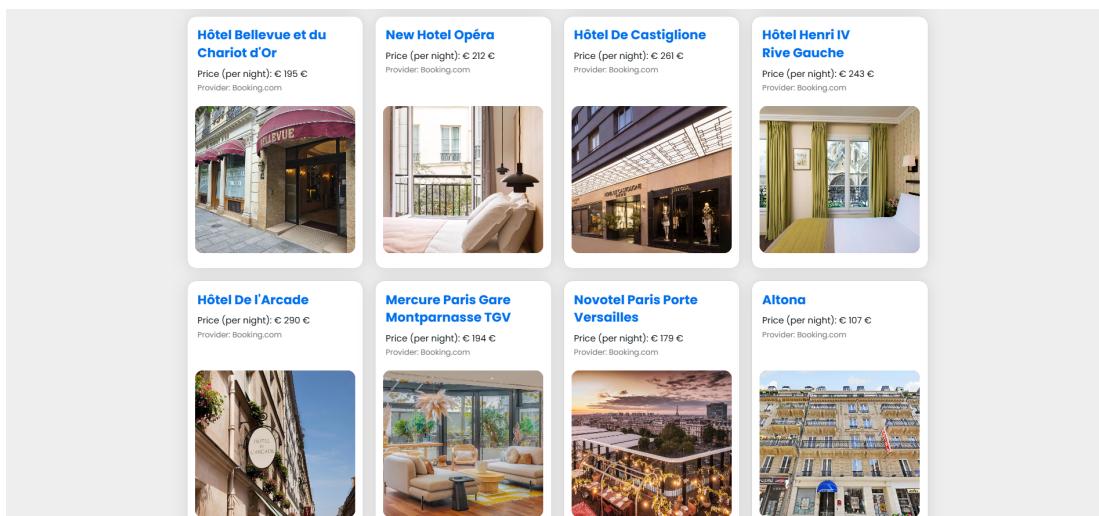


Figura A.16: Ejemplo de búsqueda

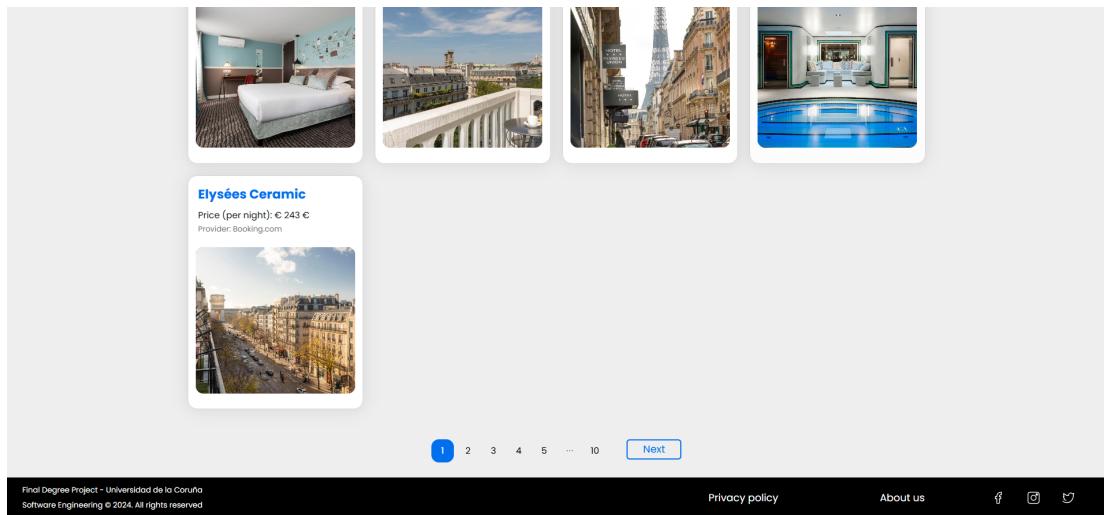


Figura A.17: Ejemplo de búsqueda

A.2.3 Detalles de un alojamiento de DeepDive

Una vez el usuario identifique un alojamiento de su interés, podrá consultar sus detalles haciendo clic en su tarjeta de presentación. En las imágenes A.18 y A.19 se muestran los detalles de un alojamiento registrado en la aplicación. En la primera imagen se observa el usuario que ha subido el alojamiento con un enlace que lleva a su perfil, la dirección del alojamiento, el campo *provider*, y un carrusel de imágenes del alojamiento. También hay un botón en caso de que el alojamiento haya sido valorado y el usuario quiera consultar dichas valoraciones.

Figura A.18: Detalles de un alojamiento de DeepDive

En la segunda sección de los detalles del alojamiento, se presenta la lista de facilidades de las que dispone, así como un desplegable que incluye información relevante, como la descripción, el precio por noche (en euros, libras y dólares), los horarios de *check-in* y *check-out* y los datos de contacto. Además, el usuario puede consultar el número total de habitaciones disponibles y verificar en un formulario si el alojamiento tiene habitaciones disponibles en el intervalo deseado de su viaje.

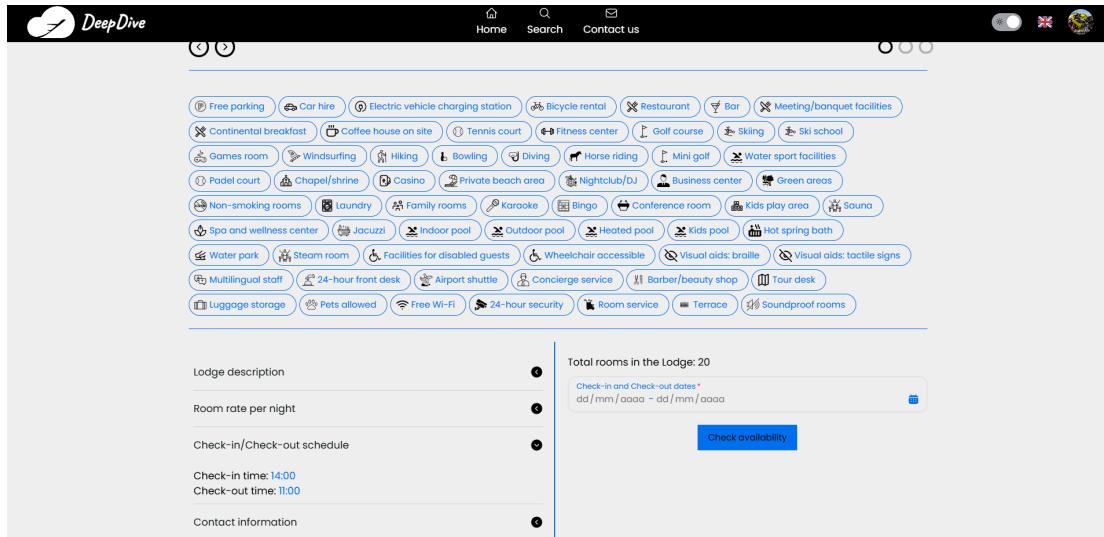


Figura A.19: Detalles de un alojamiento de DeepDive

A.2.4 Detalles de un alojamiento de Booking

De la misma manera que el usuario accede a los detalles de un alojamiento de DeepDive, también puede acceder a los detalles de un alojamiento de Booking. La primera sección de los detalles de un alojamiento de Booking se puede observar en la figura A.20 la estructura es similar a la de un alojamiento de DeepDive.

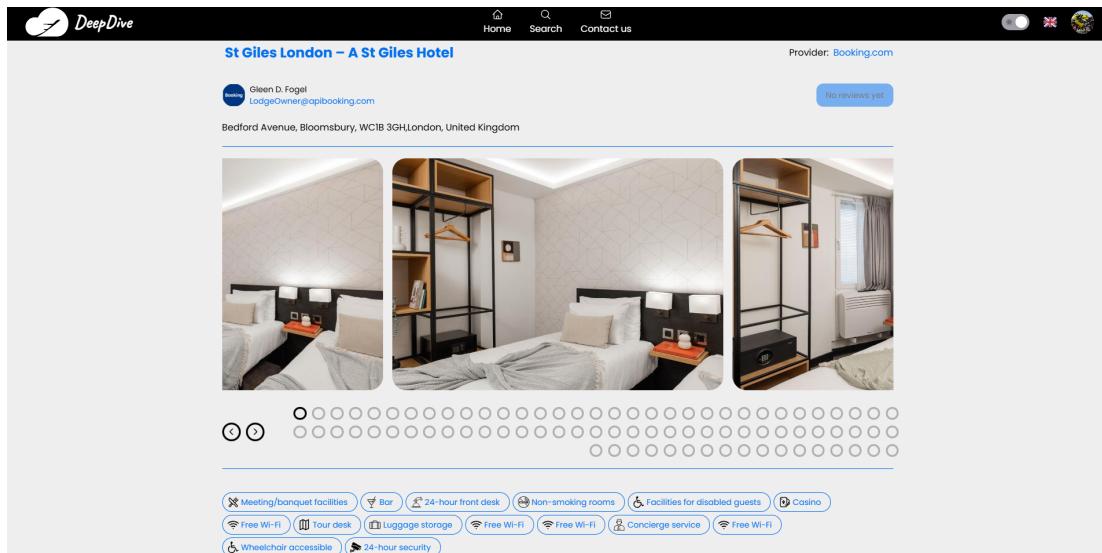


Figura A.20: Detalles de un alojamiento de Booking

En cambio, en la segunda sección de los detalles de un alojamiento de Booking se pueden observar algunas diferencias. Como se puede observar en la imagen A.21, el desplegable contiene otro tipo de información relevante, como pueden ser las coordenadas, con un enlace que permite ver la ubicación en Google Maps [60], los lenguajes que hablan los empleados del alojamiento, la zona horaria y las valoraciones de Booking. También el usuario puede consultar el precio total de la reserva en las fechas seleccionadas previamente y el número total de habitaciones que están disponibles en ese intervalo.

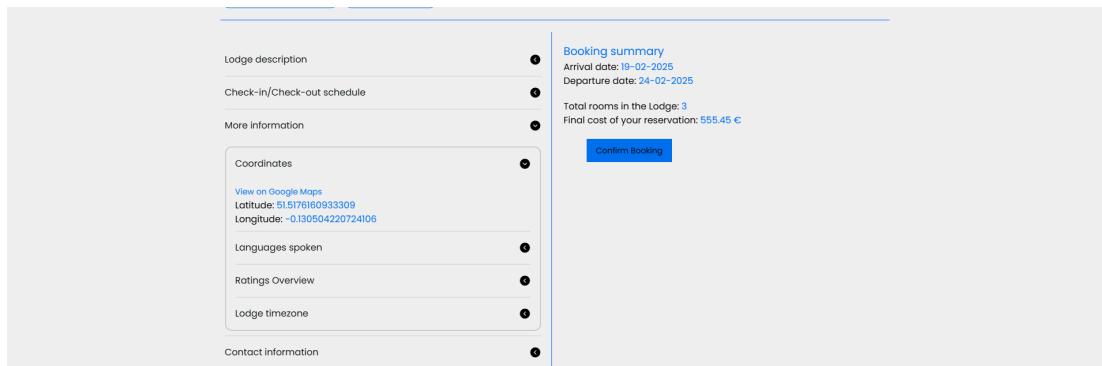


Figura A.21: Detalles de un alojamiento de Booking

A.2.5 Publicación de un alojamiento

En caso de que el usuario quiera añadir su alojamiento en la aplicación, lo podrá hacer accediendo a la opción *Register a brand new lodge* que se muestra en el desplegable de usuario mencionado previamente A.6. Esto nos redirige a un formulario para introducir los datos del

alojamiento, en la primera sección que se muestra en la imagen A.22 se pueden visualizar los distintos campos que el usuario debe completar, así como la imagen de avatar del alojamiento (se ha introducido una de ejemplo para que mostrar la previsualización de la misma).

The screenshot shows a registration form for a lodging. At the top, there's a navigation bar with a logo, 'Deep Dive', and links for 'Home', 'Search', 'Contact us', and a toggle switch. Below the navigation is a title 'Register your accommodation'. The form consists of several sections:

- Lodge name ***: An input field.
- Select the cover image**: A placeholder image of a modern building at night with a 'Deep Dive' watermark, accompanied by a 'Select image' button.
- Description of the Lodge ***: A text area with placeholder text 'Briefly describe your lodge and its features'.
- Check-in/Check-out schedule**: Fields for 'Check-in' (set to '2023-09-15') and 'Check-out' (set to '2023-09-16').
- Price and availability**: Fields for 'Rooms available' (set to '10') and 'Price per night' (set to '100').
- Contact**: Fields for 'Contact email of the lodge *' (placeholder 'lodge@example.com') and 'Phone number of the lodge *' (placeholder '(+34) 91 123 4567').
- Available features and services**: A table listing various amenities with checkboxes and categories:

Feature	Category
Free parking	Vehicle
Car hire	Vehicle
Electric vehicle charging station	Vehicle
Bicycle rental	Vehicle
Restaurant	Food
Bar	Food
Meeting/banquet facilities	Food
Continental breakfast	Food

 A navigation bar at the bottom of the list shows pages 1 through 8.

Figura A.22: Registra un alojamiento

En la siguiente sección, mostrada en la imagen A.23 se puede visualizar la tabla de facilidades, el usuario deberá marcar las que considere apropiadas para su alojamiento.

This screenshot shows the 'Available features and services' section of the registration form. It displays a list of amenities with checkboxes and their corresponding categories:

Feature	Category
Free parking	Vehicle
Car hire	Vehicle
Electric vehicle charging station	Vehicle
Bicycle rental	Vehicle
Restaurant	Food
Bar	Food
Meeting/banquet facilities	Food
Continental breakfast	Food

 A navigation bar at the bottom of the list shows pages 1 through 8.

Figura A.23: Registra un alojamiento

En la última sección, se encuentra una piscina *drag and drop* para que el usuario pueda añadir imágenes del alojamiento. Además, se muestra a continuación una previsualización de imágenes de ejemplo, para mostrar al usuario el botón para eliminar cada imagen.

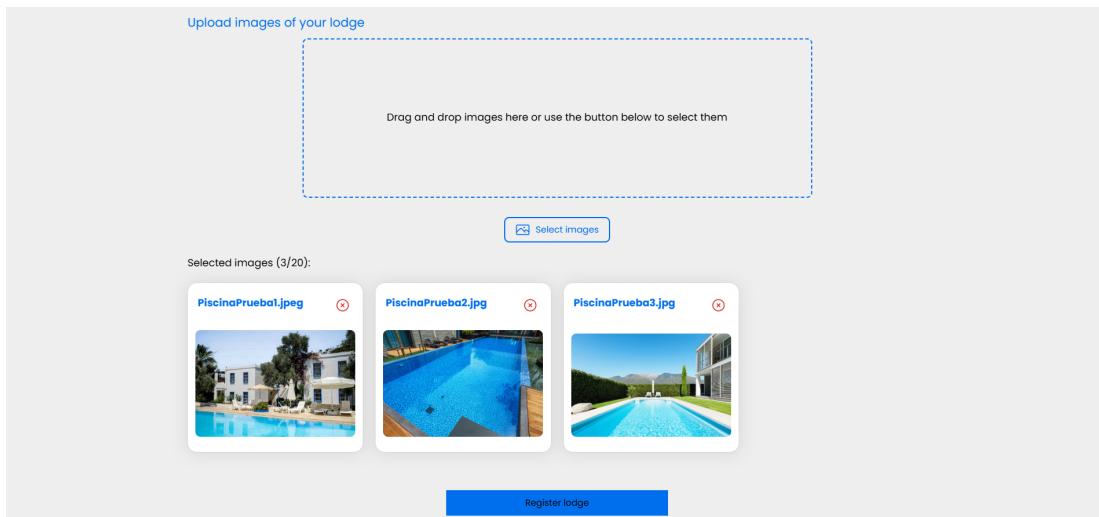


Figura A.24: Registra un alojamiento

A.2.6 Consulta los alojamientos de un usuario

En caso de que el usuario quiera consultar sus alojamientos, lo podrá hacer accediendo a la opción *My lodges* que se muestra en el desplegable de usuario mencionado previamente A.6. Esto redirige al usuario a una página donde puede consultar el estado de sus alojamientos, como se puede apreciar en la imagen A.25

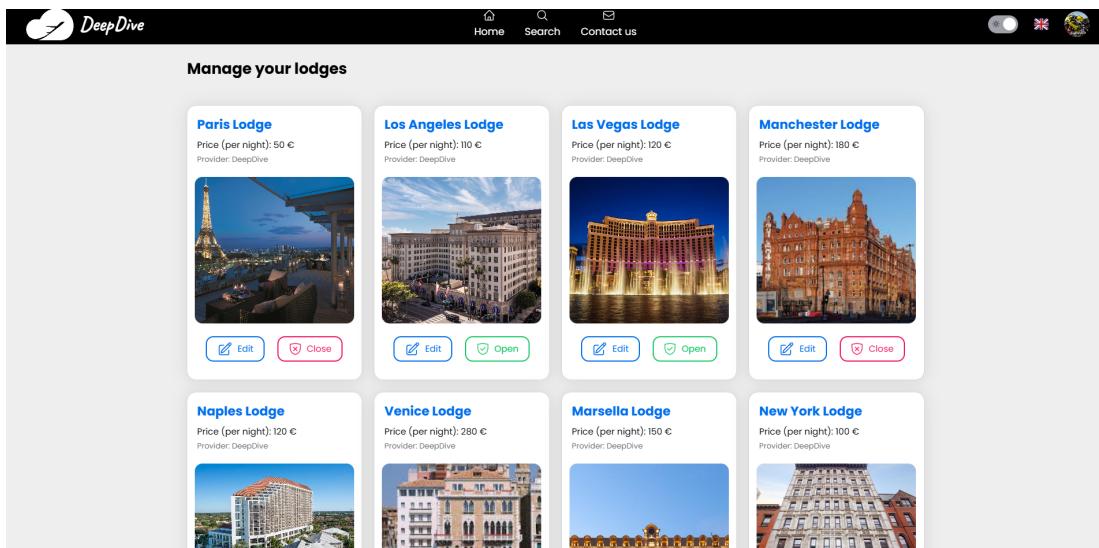


Figura A.25: Consulta los alojamientos de un usuario

A.2.7 Actualización de los detalles de un alojamiento

Un usuario podrá actualizar los detalles de sus alojamientos desde la vista de sus alojamientos mostrada previamente [A.25](#). En caso de pulsar el botón *Edit* que se ve en la imagen el usuario será redirigido a un formulario para actualizar los detalles del alojamiento seleccionado. El formulario es similar al presentado en las imágenes [A.22](#), [A.23](#), y [A.24](#), con la única diferencia que se muestran los datos preexistentes del alojamiento para que el usuario pueda realizar las modificaciones pertinentes con mayor facilidad.

A.2.8 Abrir y cerrar un alojamiento

Un usuario tendrá la opción de abrir o cerrar un alojamiento desde la vista de sus alojamientos mostrada previamente [A.25](#). El cambio actualizará la página para mostrar la información actual de los alojamientos.

A.3 Gestión de reservas

A.3.1 Consultar la disponibilidad de un alojamiento

Cuando un usuario decide el alojamiento en el que desea hospedarse, en la sección de detalles de alojamiento previamente mostrada se incluye un componente que permite elegir el periodo de estadía y verificar la disponibilidad del alojamiento, como se observa en la imagen [A.26](#).

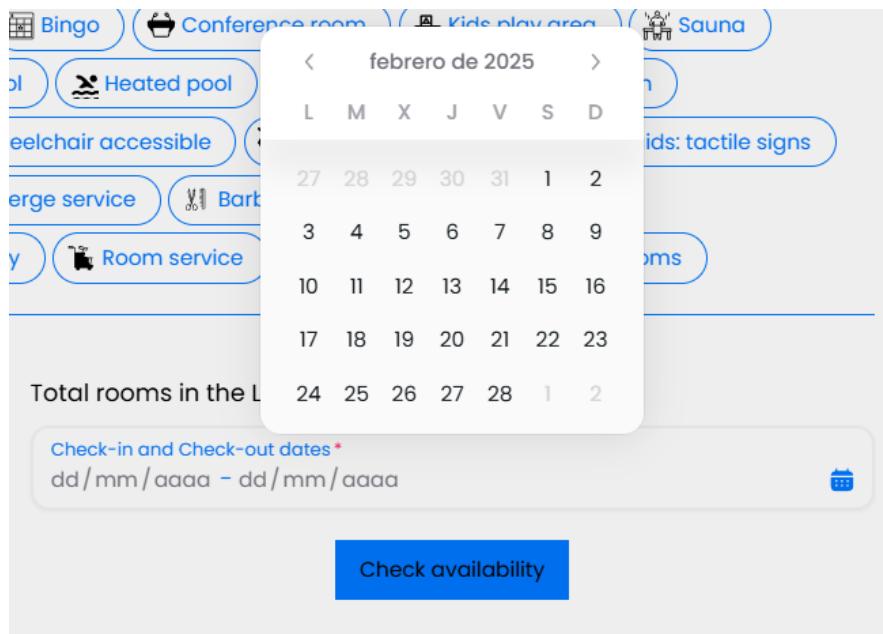


Figura A.26: Consulta la disponibilidad de un alojamiento

A.3.2 Realizar una reserva

Una vez el usuario ha verificado la disponibilidad de un alojamiento o decidido reservarlo tras consultar el precio en el caso de un alojamiento de Booking, tiene la opción de seleccionar el botón de reserva. Al hacerlo, es redirigido al formulario informativo, donde únicamente debe actualizar la información correspondiente al método de pago, como se muestra en las imágenes A.27 y A.28.

En la imagen A.27 que se muestra a continuación se ven los datos del alojamiento elegido e información del viaje.

Lodge name
Florence Lodge

Lodge email
FlorenceLodge@example.com

Price per night
210 €

Address
2020 Piazza della Signoria, Florence, Florence, Italy

Check in time
⌚ 14:00

Check out time
⌚ 11:00

Arrival date
📅 23 / 01 / 2025

Departure date
📅 27 / 01 / 2025

Total days
4 days

Total price
840 €

Figura A.27: Reservar un alojamiento

En la imagen A.28 se muestra la sección donde se debe introducir la tarjeta para realizar el pago.

Card number*

Card holder name*

Expiration date CVC

Confirm booking

Figura A.28: Reservar un alojamiento

Como se aprecia en las imágenes A.29 y A.30, el componente de la tarjeta se actualiza en tiempo real a medida que el usuario ingresa sus datos, lo que permite visualizar de manera inmediata la información proporcionada, asegurando una mayor precisión en la introducción de los datos y facilitando el proceso de pago.

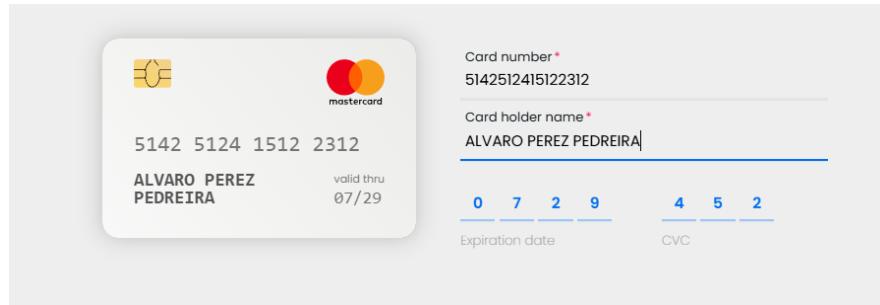


Figura A.29: Método de pago

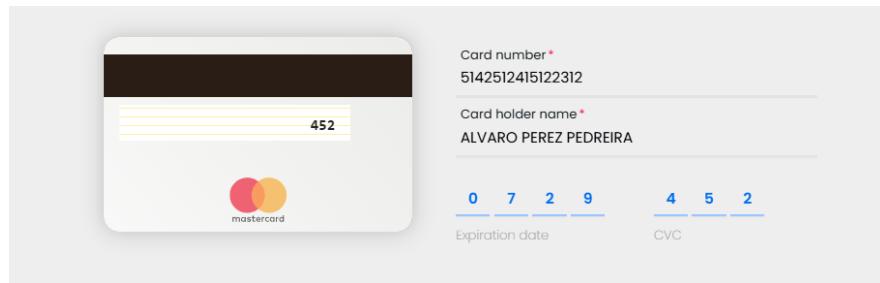


Figura A.30: Método de pago

A.3.3 Consultar tus reservas

Si el usuario desea consultar sus reservas, puede hacerlo a través de la opción *My bookings* del menú desplegable de usuario mostrado previamente A.6. Una vez seleccionada esta opción, el usuario será redirigido a la página que se muestra a continuación A.31.

This figure displays a user's booking history on the Deep Dive website. The page title is 'My bookings'. It lists eight reservations in a grid format:

- ParisLodge@example.com**: Price: 350 €, Status: concluded. Details: Arrival date: 05-12-2022, Departure date: 12-12-2022, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-12-2022. Buttons: 'Rate' (blue) and 'Cancel' (red).
- IALodge@example.com**: Price: 770 €, Status: concluded. Details: Arrival date: 05-12-2022, Departure date: 12-12-2022, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-12-2022. Buttons: 'Rate' (blue) and 'Cancel' (red).
- MadridLodge@example.com**: Price: 350 €, Status: active. Details: Arrival date: 01-01-2025, Departure date: 31-12-2025, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-12-2023. Buttons: 'Rate' (blue) and 'Cancel' (red).
- MadridLodge@example.com**: Price: 350 €, Status: active. Details: Arrival date: 01-01-2025, Departure date: 31-12-2025, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-12-2023. Buttons: 'Rate' (blue) and 'Cancel' (red).
- BarcelonatLodge@example.com**: Price: 350 €, Status: active. Details: Arrival date: 01-01-2025, Departure date: 31-12-2025, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-12-2023. Buttons: 'Rate' (blue) and 'Cancel' (red).
- BarcelonatLodge@example.com**: Price: 350 €, Status: concluded. Details: Arrival date: 01-01-2013, Departure date: 01-01-2013, Check in time: 14:00, Check out time: 11:00, Confirmation date: 01-01-2013. Buttons: 'Rate' (blue) and 'Cancel' (red).
- BarcelonatLodge@example.com**: Price: 350 €, Status: concluded. Details: Arrival date: 01-02-2013, Departure date: 01-02-2013, Check in time: 14:00, Check out time: 11:00, Confirmation date: 02-01-2013. Buttons: 'Rate' (blue) and 'Cancel' (red).
- BarcelonatLodge@example.com**: Price: 350 €, Status: concluded. Details: Arrival date: 01-03-2013, Departure date: 01-03-2013, Check in time: 14:00, Check out time: 11:00, Confirmation date: 03-01-2013. Buttons: 'Rate' (blue) and 'Cancel' (red).

Figura A.31: Consultar las reservas de un usuario

A.3.4 Cancelar una reserva

Si el usuario desea cancelar una reserva, lo podrá hacer a través de la sección mostrada previamente A.31, en caso de no realizarse con la suficiente antelación el usuario recibirá un error.

A.4 Gestión de comentarios

A.4.1 Valorar una reserva

Si el usuario desea valorar una reserva que ya ha concluido, lo podrá hacer a través de la sección mostrada previamente A.31, presionando el botón *Rate*. Esto redirigirá al usuario al siguiente formulario A.32 donde podrá introducir su valoración y su comentario.

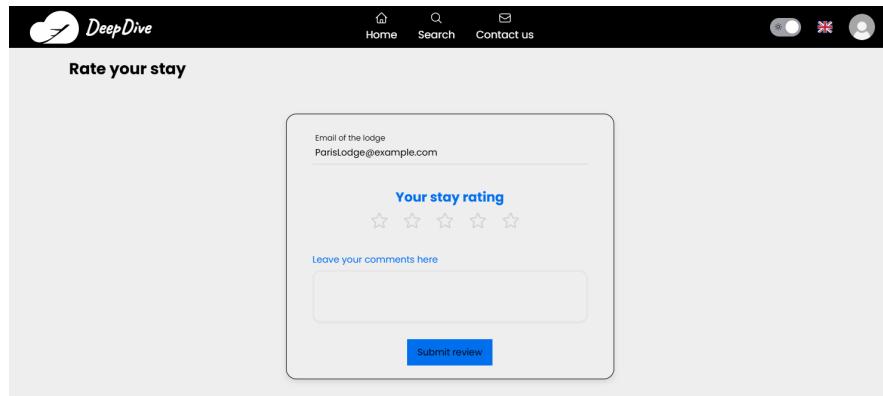
A screenshot of a web browser displaying a review form. The header features the logo 'DeepDive' and navigation links for 'Home', 'Search', and 'Contact us'. On the right side of the header are language selection and user profile icons. The main content area has a light gray background and contains the text 'Rate your stay' in bold. Below this, there is a rectangular input field with the placeholder 'Email of the lodge' and the value 'ParisLodge@example.com'. Underneath the input field is a section titled 'Your stay rating' with five yellow star icons. Below the stars is a text input field with the placeholder 'Leave your comments here'. At the bottom of the form is a blue 'Submit review' button.

Figura A.32: Valorar una reserva

A.4.2 Ver los comentarios de un alojamiento

En caso de que el usuario quiera ver los comentarios de un alojamiento, lo podrá hacer desde los detalles del mismo presionando el botón *View reviews* que se puede apreciar en la imagen A.33.

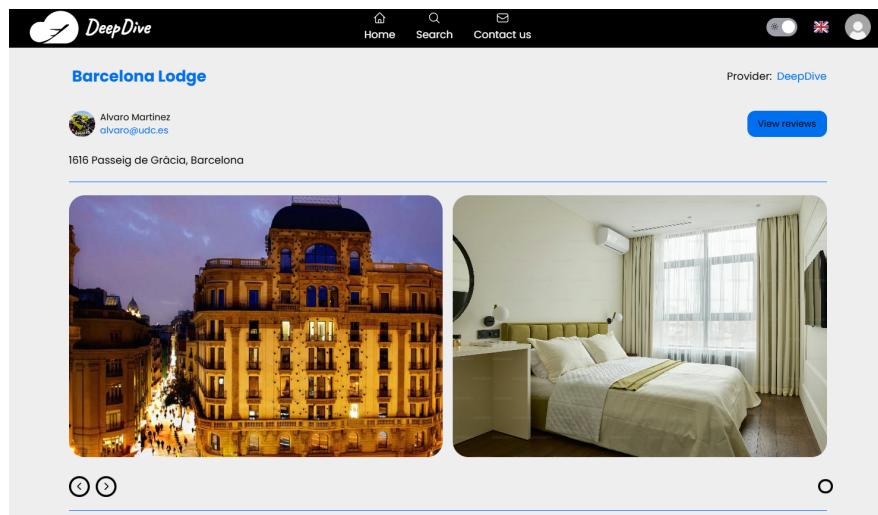


Figura A.33: Ver comentarios de un alojamiento

Una vez presionado ese botón, se redirigirá al usuario a la página que se muestra a continuación donde podrá consultar los comentarios de los diferentes usuarios sobre el alojamiento A.34.

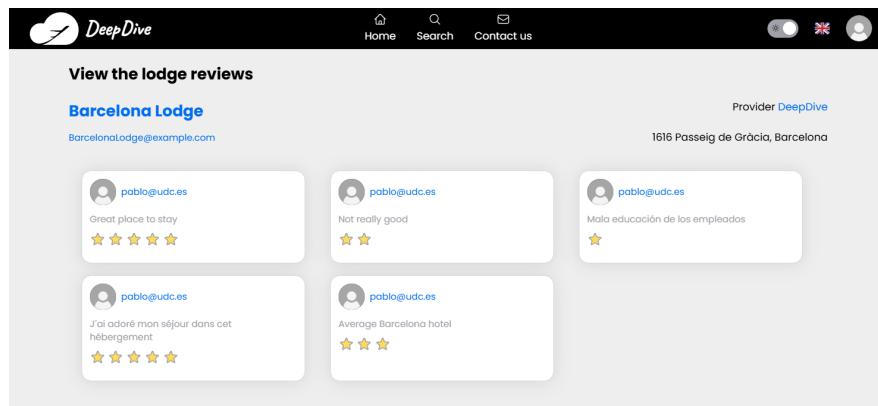


Figura A.34: Ver comentarios de un alojamiento

A.5 Módulo de administrador

En esta sección se mostrará las funcionalidades específicas del administrador.

A.5.1 Desplegable de usuario del administrador

El administrador contará con un menú desplegable de usuario extendido A.35, adaptado a sus responsabilidades para garantizar el correcto funcionamiento de la aplicación.

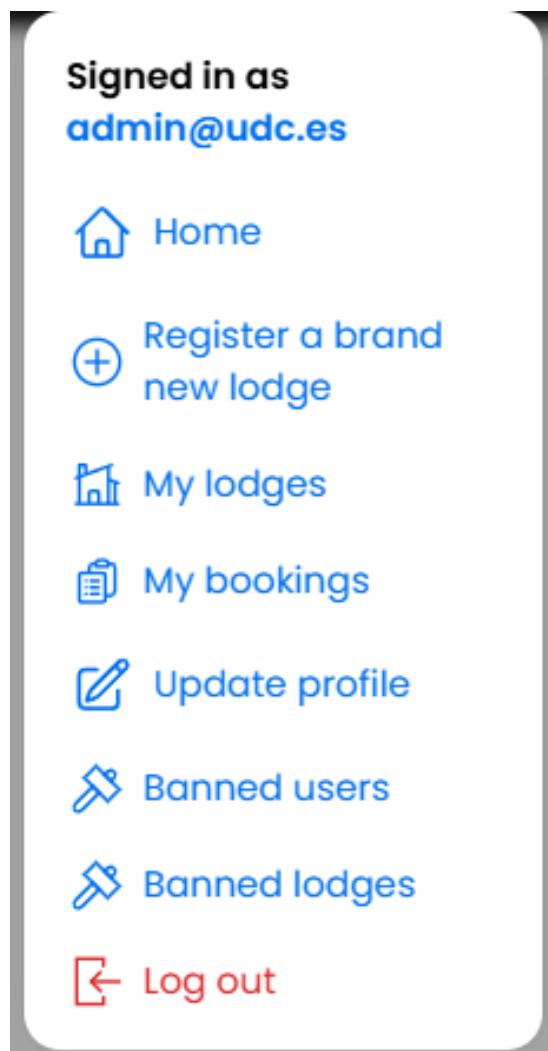


Figura A.35: Desplegable de usuario administrador

A.5.2 Bloquear un usuario

Si el administrador desea bloquear a un usuario, podrá hacerlo a través de la sección de detalles del perfil previamente presentada. La única diferencia es que dispondrá de un botón adicional que le permitirá ejecutar esta acción sobre el usuario seleccionado. Esto se puede observar en la figura A.36.

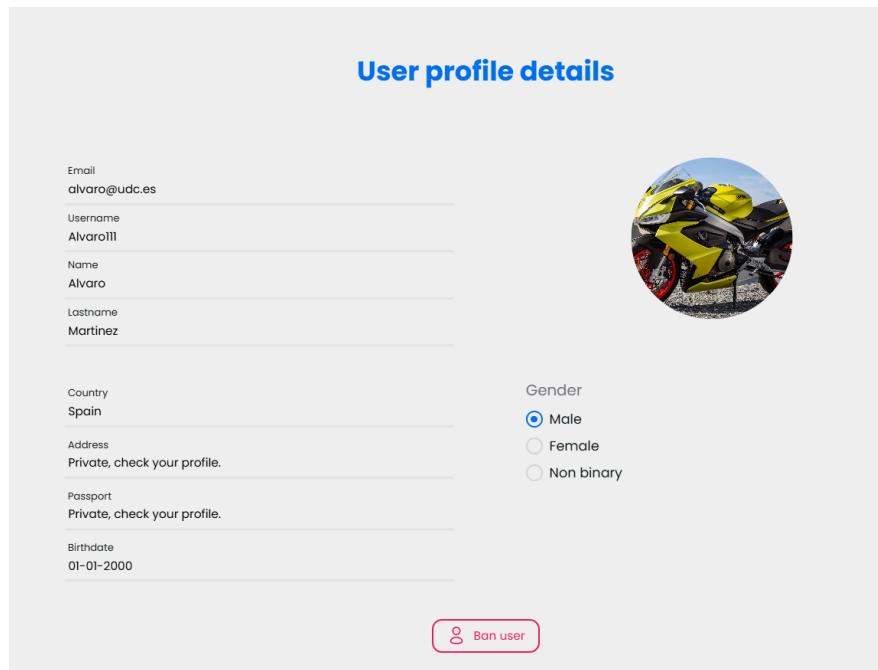


Figura A.36: Bloquear a un usuario

En caso de que un usuario bloqueado intente acceder a la aplicación, verá el error mostrado en la imagen [A.37](#).

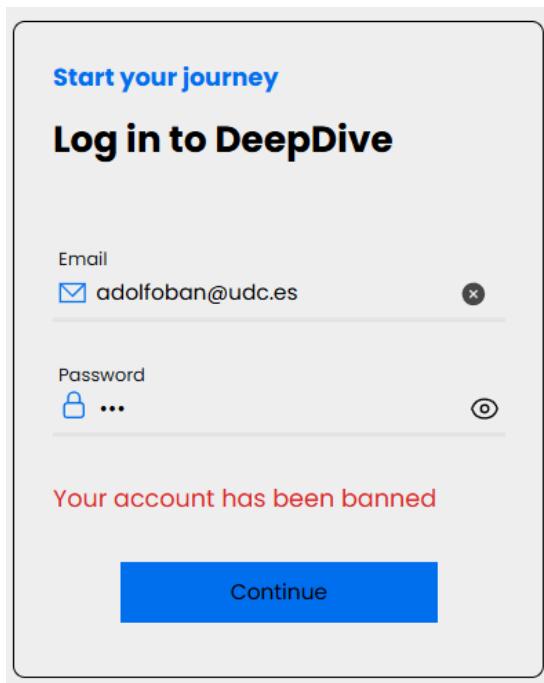


Figura A.37: Error de acceso de un usuario bloqueado

A.5.3 Consultar usuarios bloqueados

Si el administrador desea consultar los usuarios que han sido bloqueados, podrá hacerlo a través de la opción *Banned users* del desplegable de usuario administrador mostrado previamente A.35. Al seleccionar esta opción, el administrador será redirigido a la página correspondiente, como se puede observar en la figura A.38.

DeepDive

Home Search Contact us

Banned users

Email	Username	Full name	Action
adolfoban@udc.es	AdolfoBan	Adolfo Rodriguez	Unban user
carlosban@udc.es	CarlosBan	Carlos Suarez	Unban user
miguelban@udc.es	MiguelBan	Miguel Mata	Unban user

Figura A.38: Lista de usuarios bloqueados

A.5.4 Desbloquear a un usuario

En caso de que el administrador desee desbloquear a un usuario lo podrá hacer a través del botón *Unban user* que aparece debajo de la tarjeta de cada usuario en la sección mostrada

previamente [A.38](#).

A.5.5 Bloquear un alojamiento

Si el administrador desea bloquear un alojamiento, podrá hacerlo a través de la sección de detalles del alojamiento previamente presentada. La única diferencia es que dispondrá de un botón adicional debajo de los detalles del alojamiento que le permitirá ejecutar esta acción sobre el alojamiento seleccionado. Esto se puede observar en la figura [A.39](#).

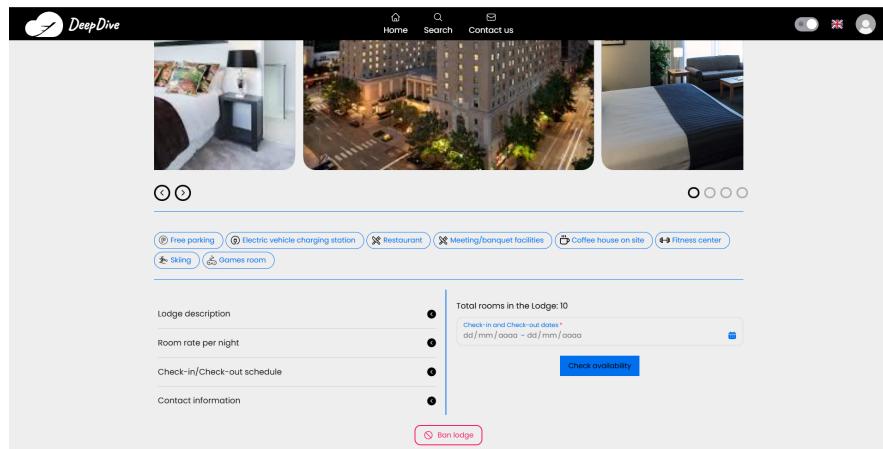


Figura A.39: Bloquear un alojamiento

A.5.6 Consultar alojamientos bloqueados

Si el administrador desea consultar los alojamientos que han sido bloqueados, podrá hacerlo a través de la opción *Banned lodges* del desplegable de usuario administrador mostrado previamente [A.35](#). Al seleccionar esta opción, el administrador será redirigido a la página correspondiente, como se puede observar en la figura [A.40](#).

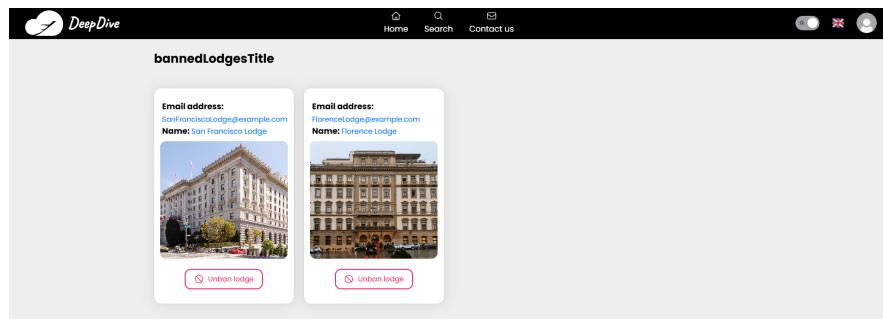


Figura A.40: Lista de alojamientos bloqueados

A.5.7 Desbloquear un alojamiento

En caso de que el administrador desee desbloquear un alojamiento lo podrá hacer a través del botón *Unban lodge* que aparece debajo de la tarjeta de cada alojamiento en la sección mostrada previamente [A.40](#).

A.5.8 Bloquear un comentario

En caso de que el administrador considere que alguna valoración está incumpliendo con las políticas de la aplicación, podrá bloquearla presionando el botón que aparece debajo de cada tarjeta de comentario [A.41](#), funcionalidad disponible únicamente para los usuarios con perfil de administrador.

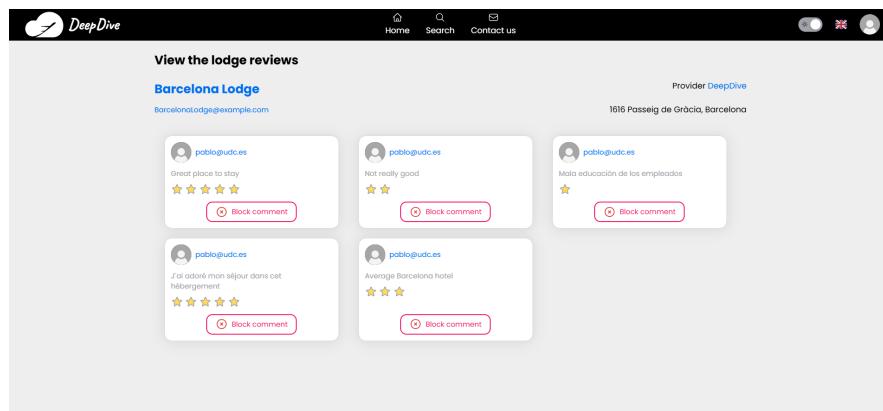


Figura A.41: Bloquear un comentario

Lista de acrónimos

- API** Application Programming Interface. v, 1–3, 11, 16, 28, 29, 53, 60, 64, 65, 69, 74, 76, 79, 88
- CORS** Cross-Origin Resource Sharing. 70
- CRUD** Create Read Update Delete. 48, 50, 53, 54
- CSS** Cascading Style Sheets. 12, 13, 23
- DAO** Data Access Object. 69
- DOM** Document Object Model. 12
- DRY** Don’t Repeat Yourself. 49
- DTO** Data Transfer Object. 48, 69, 70, 109
- HTML** HyperText Markup Language. 12, 13
- HTTP** HyperText Transfer Protocol. 14, 16, 59, 64, 65, 69, 70, 76
- IDE** Integrated Development Environment. 14
- IU** Interfaz de Usuario. 14, 63
- JAR** Java ARchive. 74
- JPA** Java Persistence API. 53, 54, 67, 74
- JSON** JavaScript Object Notation. 11, 71
- JWT** JSON Web Token. 60, 70
- KISS** Keep It Simple, Stupid. 49

ORM Object-Relational Mapping. [10](#)

REST REpresentational State Transfer. [10](#), [60](#)

SGBD Sistema de Gestión de Bases de Datos. [10](#)

SPA Single Page Application. [12](#)

SQL Structured Query Language. [10](#), [15](#), [23](#), [48](#), [54](#)

SVG Scalable Vector Graphics. [72](#)

URL Uniform Resource Locator. [74](#)

YAGNI You Aren't Gonna Need It. [49](#)

Glosario

bean Componente de software reutilizable en Java, diseñado para ahorrar tiempo de desarrollo al facilitar la programación de funcionalidades comunes.. [48](#)

conversor Clase encargada de convertir o transformar los datos entre las entidades y los **DTO(s)**.. [70](#)

framework Conjunto estructurado de conceptos, prácticas, bibliotecas y herramientas que tienen como objetivo servir de base en el desarrollo de software. Su propósito es ofrecer una estructura predefinida que guía y simplifica la creación de aplicaciones.. [9–14, 53](#)

hook Función de React que permite gestionar estado, ciclo de vida o lógica reutilizable en componentes funcionales.. [72](#)

store Objeto que almacena y maneja el estado de la aplicación, permitiendo acceder y actualizar los datos globales desde cualquier componente.. [72](#)

Bibliografía

- [1] Servimedia, “El ine certifica el año récord del turismo en 2023,” *Servimedia*, 2024. [En línea]. Disponible en: <https://www.servimedia.es/noticias/ine-certifica-ano-record-turismo-2023-85-1-millones-llegadas-108662-millones-gasto/4291350>
- [2] T. B. C. for Translational Research, “The evidence of vacations,” *Psychology Today*, 2017. [En línea]. Disponible en: <https://www.psychologytoday.com/us/blog/evidence-based-living/201702/the-evidence-on-vacations>
- [3] W. Tourism, “The science of wellness tourism: Understanding the benefits for your health,” 2024. [En línea]. Disponible en: <https://www.wellnesstourism.com/article/the-science-of-wellness-tourism-understanding-the-benefits-for-your-health>
- [4] D. C. Heymann, “El auge del turismo rural en españa: una oportunidad para el desarrollo rural,” *CaixaBank Research*, 2020. [En línea]. Disponible en: <https://www.caixabankresearch.com/es/analisis-sectorial/agroalimentario/auge-del-turismo-rural-espana-oportunidad-desarrollo-rural>
- [5] Booking, “Booking.com: Hoteles, apartamentos, vuelos, alquiler de coches y más,” 2024, accedido: 2024-12-09. [En línea]. Disponible en: <https://www.booking.com/index.es.html>
- [6] Airbnb, “Airbnb: Alquileres vacacionales, casas, experiencias y lugares,” 2024, accedido: 2024-12-09. [En línea]. Disponible en: <https://www.airbnb.es/>
- [7] Wikipedia, “Java (lenguaje de programación),” 2024, accedido: 2024-12-21. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))
- [8] BIGCODE, “Características de java: ¿qué lo hace especial?” 2024. [En línea]. Disponible en: <https://bigcode.es/caracteristicas-de-java-que-lo-hace-especial/>
- [9] Wikipedia, “Spring boot,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Spring_Boot

- [10] ---, “Spring security,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Spring_Security
- [11] ---, “Mysql,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/MySQL>
- [12] ---, “Hibernate,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/Hibernate>
- [13] ---, “Maven,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/Maven>
- [14] ---, “Junit,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/JUnit>
- [15] ---, “Json web token,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: https://es.wikipedia.org/wiki/JSON_Web_Token
- [16] ---, “Javascript,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/JavaScript>
- [17] JavaScript, “React,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/React>
- [18] Wikipedia, “Javascript,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/JavaScript>
- [19] React, “React - la biblioteca para interfaces de usuario web y nativas,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.react.dev>
- [20] Wikipedia, “Html,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/HTML>
- [21] ---, “Css,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/CSS>
- [22] Tailwind, “Getting started with tailwind css,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://v2.tailwindcss.com/docs>
- [23] Wikipedia, “Protocolo de transferencia de hipertexto,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto
- [24] NextUI, “Make beautiful websites regardless of your design experience.” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://nextui.org/>

- [25] E. Carousel, “Embla carousel. a lightweight carousel library with fluid motion and great swipe precision.” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://www.embla-carousel.com/>
- [26] Swiper, “Swiper. the most modern mobile touch slider.” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://swiperjs.com/>
- [27] Jetbrains, “The leading java and kotlin ide,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: <https://www.jetbrains.com/idea/>
- [28] ——, “Haga realidad su visión del software,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: <https://www.jetbrains.com/es-es/>
- [29] V. S. Code, “Your code editor. redefined with ai.” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://code.visualstudio.com/>
- [30] Wikipedia, “Microsoft,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/Microsoft>
- [31] MySQL, “Mysql,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://dev.mysql.com/doc/refman/8.0/en/mysql.html>
- [32] Node.org, “Ejecuta javascript en cualquier parte,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://nodejs.org/es>
- [33] Postman, “Ai is powered by apis. apis are powered by postman,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://www.postman.com/>
- [34] Git, “git –local-branching-on-the-cheap,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://git-scm.com/>
- [35] Overleaf, “Overleaf. a digital science solution,” 2024, accedido: 2024-12-21. [En línea]. Disponible en: <https://www.overleaf.com/project>
- [36] Wikipedia, “Scrum (desarrollo de software),” 2024, accedido: 2024-12-22. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Scrum_\(desarrollo_de_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software))
- [37] A. W. Services, “¿en qué consiste scrum?” 2024, accedido: 2024-12-22. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/scrum/>
- [38] A. M. Loreto, “Cuánto gana un programador en españa en 2024,” 2024, accedido: 2024-12-22. [En línea]. Disponible en: <https://www.hackaboss.com/blog/cuanto-gana-programador-salario-espana>

- [39] J. Rodríguez, “¿qué es un scrum master? funciones y salario en españa,” 2024, accedido: 2024-12-22. [En línea]. Disponible en: <https://www.mundoposgrado.com/que-es-un-scrum-master-funciones-y-salario-en-espana/>
- [40] ING, “Dime tu situación laboral y te diré qué impuestos deberás pagar,” 2024, accedido: 2024-12-22. [En línea]. Disponible en: <https://www.ing.es/ennaranja/finanzas-personales/que-impuestos-deberas-pagar-trabajador/>
- [41] IntelliJ, “Opciones de suscripción y precios,” accedido: 2025-01-15. [En línea]. Disponible en: <https://www.jetbrains.com/es-es/idea/buy/?section=personal&billing=monthly>
- [42] Anna, “¿cuánto tiempod ura un ordenador portátil? ¿cuándo comprar un ordenador portátil nuevo?” 2024, accedido: 2024-12-22. [En línea]. Disponible en: <https://www.minitool.com/es/respaldar-datos/cuanto-dura-un-ordenador-portatil.html>
- [43] M. Rehkopf, “Historias de usuario con ejemplos y plantilla,” 2024, accedido: 2024-12-24. [En línea]. Disponible en: <https://www.atlassian.com/es/agile/project-management/user-stories>
- [44] Wikipedia, “Arquitectura multicapa,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Arquitectura_multicapa
- [45] ——, “Objeto de transferencia de datos,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Objeto_de_transferencia_de_datos
- [46] E. Barrios, “Patrón repositorio y unidad de trabajo,” 2019. [En línea]. Disponible en: <https://dev.to/ebarrioscode/patron-repositorio-repository-pattern-y-unidad-de-trabajo-unit-of-work-en-asp-net-core-webapi-3-0>
- [47] Wikipedia, “Inyección de dependencias,” 2022, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_de_dependencias
- [48] ——, “Encapsulamiento (informática),” 2024, accedido: 2025-01-15. [En línea]. Disponible en: [https://es.wikipedia.org/wiki/Encapsulamiento_\(inform%C3%A1tica\)](https://es.wikipedia.org/wiki/Encapsulamiento_(inform%C3%A1tica))
- [49] ——, “Principio de inversión de la dependencia,” 2023, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Principio_de_inversi%C3%B3n_de_la_dependencia
- [50] ——, “Principio de segregación de la interfaz,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Principio_de_segregaci%C3%B3n_de_la_interfaz

- [51] ——, “Principio de responsabilidad única,” 2020, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Principio_de_responsabilidad_%C3%BAlica
- [52] ——, “No te repitas,” 2020, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/No_te_repit
- [53] ——, “Principio kiss,” 2024, accedido: 2025-01-15. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Principio_KISS
- [54] ——, “Yagni,” 2020, accedido: 2025-01-15. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/YAGNI>
- [55] rapidapi, “Rapidapi hub: public api marketplace,” accedido: 2025-01-18. [En línea]. Disponible en: <https://rapidapi.com/hub>
- [56] Booking, “Booking.com,” accedido: 2025-01-18. [En línea]. Disponible en: <https://www.booking.com/index.es.html>
- [57] EmailJS, “Send email directly from your code,” accedido: 2025-01-18. [En línea]. Disponible en: <https://www.emailjs.com/>
- [58] Zustand, “Zustand,” accedido: 2025-01-18. [En línea]. Disponible en: <https://zustand-demo.pmnd.rs/>
- [59] React, “React router,” accedido: 2025-01-18. [En línea]. Disponible en: <https://reactrouter.com/>
- [60] Wikipedia, “Google maps,” accedido: 2025-01-18. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Google_Maps