

A stylized, dark blue mountain range graphic that curves across the top and bottom of the page, framing the central text. The mountains are represented by sharp, angular peaks and valleys.

AGENCIA DE VIAJES ENTRE VALLES Y MONTAÑAS

**I.E.S BERNALDO DE QUIRÓS
CURSO 2023-2024
NOEMÍ JUAN PÉREZ**

ÍNDICE DE LA MEMORIA

1.	INTRODUCCIÓN	1
1.1	Presentación y Objetivos.	1
1.2	Contexto.	1
1.3	Planteamiento del Problema (la idea).	1
1.4	Análisis de costes.	2
1.5	Plan de Financiación.	2
1.6	Plan de recursos humanos.	2
1.7	Plan de prevención de riesgos.	3
2.	ESPECIFICACIÓN DE REQUISITOS	4
2.1	Introducción.	4
2.2	Descripción general.	4
2.3	Requisitos Específicos	4
2.3.1	Requerimientos Funcionales.	4
2.3.2	Requerimientos de Interfaces Externas.	5
2.3.3	Requerimientos de Rendimiento.	9
2.3.4	Obligaciones del Diseño.	10
2.3.5	Atributos.	11
3.	ANÁLISIS	13
3.1	Introducción.	13
3.2	Diagrama de Clases.	13
3.3	Diagrama de Casos de Uso.	14
4.	DISEÑO	17
4.1	Introducción.	17
4.2	Capa de Presentación.	17
4.3	Capa de Negocio o Lógica de la Aplicación.	20
4.4	Capa de Persistencia o Datos.	22
5.	IMPLEMENTACIÓN	24
5.1	Tecnologías utilizadas en el desarrollo del proyecto.	24
5.2	Descripción del Proyecto.	25

5.2.1	Capa de Presentación.	25
5.2.2	Capa de Negocio o Lógica de la Aplicación.	29
5.2.3	Capa de Persistencia o de Datos.	32
6.	EVALUACIÓN	34
6.1	Introducción.	34
6.2	Validaciones de páginas de Estilo.	34
6.3	Validación de Enlaces.	35
6.4	Validación de la Resolución.	36
6.5	Validación de Navegadores.	36
7.	CONCLUSIÓN	37
7.1	Valoración Personal del Trabajo Realizado (análisis DAFO + análisis CAME).	37
7.2	Posibles Ampliaciones.	39
8.	Referencias	40

1. INTRODUCCIÓN

1.1 Presentación y Objetivos.

Descripción:

La plataforma web desarrollada, llamada "Entre Valles y Montañas", tiene como objetivo principal ofrecer a los usuarios una experiencia única al descubrir Asturias como destino turístico. La intención es cubrir una demanda no satisfecha de información digitalizada, facilitando así la planificación y reserva de viajes en la región.

Objetivos específicos:

- **Atraer turistas:** Dirigirse a un público aventurero que busca desde alojamiento hasta actividades extremas, ofreciendo una planificación completa del viaje.
- **Promocionar negocios:** Ofrecer a empresas locales de alojamiento y actividades una mayor visibilidad y potencial de reservas mediante la plataforma.
- **Mejorar la experiencia del usuario:** Implementar funciones como búsqueda avanzada, filtrado de resultados, visualización de detalles específicos, gestión de reservas, registro de usuarios y procesos de pago seguro.

1.2 Contexto.

Situación actual:

- En Asturias, el turismo ha crecido significativamente, con un aumento del 35% en visitantes en los últimos dos años. Sin embargo, la necesidad de digitalizar la información turística es evidente para facilitar la planificación de viajes.

Oportunidad:

- Crear una plataforma que satisfaga esta demanda de información digitalizada y optimice la experiencia de los visitantes, haciendo más accesible y cómoda la planificación de sus viajes a Asturias.

1.3 Planteamiento del Problema (la idea).

Problema:

- La falta de una plataforma web completa y actualizada que concentre toda la información turística sobre Asturias complica la planificación de viajes para los turistas.

Solución:

- Desarrollar "Entre Valles y Montañas", una plataforma web que reúna información detallada y actualizada sobre alojamientos, actividades y experiencias en Asturias, brindando así una solución completa para la planificación y reserva de viajes.

1.4 Análisis de costes.

Costes de desarrollo:

- **Salarios:** Pago a desarrolladores y diseñadores web.
- **Infraestructura:** Costes de servidores y alojamiento web.
- **Licencias:** Pago por licencias de software necesarias para el desarrollo.

Costes operativos:

- **Mantenimiento:** Costes continuos para mantener y actualizar la plataforma.
- **Marketing:** Gastos en publicidad y promoción de la plataforma.
- **Atención al cliente:** Costes asociados a proporcionar soporte a los usuarios.

1.5 Plan de Financiación.

Fuentes de financiación:

- **Inversión propia:** Capital inicial aportado por mí misma.
- **Subvenciones públicas:** Fondos obtenidos de programas gubernamentales para el desarrollo de proyectos tecnológicos.

Plan de inversión:

- **Inicial:** Inversión en el desarrollo y lanzamiento de la plataforma.
- **Continua:** Fondos destinados a marketing, promoción y mejoras continuas de la plataforma.

1.6 Plan de recursos humanos.

Equipo inicial:

- En esta fase, asumiré todas las funciones: desarrollo, diseño, marketing y atención al cliente.

Plan de contratación:

- **Personal cualificado:** Se contratará a medida que la plataforma genere beneficios.
 - **Áreas:** Desarrolladores web, diseñadores web, expertos en marketing digital y personal de atención al cliente.
 - **Subcontratación:** Se considerará para tareas puntuales o áreas especializadas.

1.7 Plan de prevención de riesgos.

Riesgos potenciales:

- Baja demanda de la plataforma web.
- Competencia de otras plataformas web turísticas.
- Dificultades técnicas en el desarrollo o mantenimiento de la plataforma web.

Medidas de prevención:

- Realizar un estudio de mercado para evaluar la demanda potencial de la plataforma web.
- Diferenciar la plataforma web de la competencia ofreciendo un servicio único y de alta calidad.
- Implementar un plan de mantenimiento preventivo para la plataforma web.

2. ESPECIFICACIÓN DE REQUISITOS

2.1 Introducción.

A continuación, se detallarán los requisitos funcionales que debe cumplir la aplicación web, así mismo, también se identificará a el actor o actores del sistema que podrán realizar cada uno de los requisitos, de cara a definirlos más adelante en profundidad, se le otorgará a cada requisito un identificador único.

2.2 Descripción general.

La plataforma proporcionará información detallada y actualizada sobre destinos turísticos y actividades en Asturias, permitiendo a los usuarios planificar y reservar sus viajes o actividades de manera eficiente.

2.3 Requisitos Específicos.

2.3.1 Requerimientos Funcionales.

RQ1. El usuario se podrá registrar en el sistema, esta acción podrá ser realizada por cualquier usuario que no esté registrado previamente.

RQ2. El usuario podrá visualizar los alojamientos disponibles en la web, esto podrá ser realizado por cualquier usuario.

RQ3. El usuario podrá iniciar sesión en el sistema, esto podrá ser realizado por aquellos usuarios registrados previamente en el mismo.

RQ4. El usuario podrá cerrar sesión, para ello previamente debe ser un usuario registrado y haber iniciado sesión.

RQ5. El usuario podrá modificar su contraseña, debe estar logueado para realizar esta acción.

RQ6. El usuario podrá visualizar las actividades de la web, esto podrá ser realizado por cualquier usuario.

RQ7. El usuario podrá filtrar las actividades de la web en función de diferentes filtros.

RQ8. El usuario podrá filtrar los alojamientos de la web en función de diferentes filtros, cualquier usuario lo podrá realizar.

RQ9. El usuario podrá visualizar las entradas del blog de la aplicación, lo realizará cualquier usuario.

RQ10. El usuario podrá realizar la reserva de un alojamiento, previamente debe haber realizado el pago y estar logueado en el sistema.

RQ11. El usuario podrá realizar la reserva de una actividad, previamente debe estar logueado en el sistema y haber realizado el pago.

RQ12. El sistema enviará un correo electrónico al usuario cuando modifique su contraseña.

RQ13. El sistema no permitirá loguearse al usuario cuando su email o contraseña sea incorrectos.

RQ14. El sistema validará el formato de email introducido a la hora de registrarse.

RQ15. El sistema validará que la contraseña del usuario tenga una combinación de longitud 8 entre las que debe haber al menos una mayúscula y un número, a la hora de registrarse.

RQ16. El usuario podrá ver sus reservas tanto de actividades como de alojamientos, para ello debe haberse logueado.

RQ17. El usuario administrador dispondrá de una pantalla donde poder dar de alta nuevos alojamientos, debe estar logueado con la cuenta de administrador del sistema.

RQ18. El usuario administrador dispondrá de una pantalla donde poder dar de alta nuevas actividades, debe estar logueado con la cuenta de administrador del sistema.

RQ19. El usuario administrador dispondrá de una pantalla donde poder dar de alta nuevas entradas del blog, debe estar logueado con la cuenta de administrador del sistema.

2.3.2 Requerimientos de Interfaces Externas.

2.3.2.1 Interfaces de los Usuario.

IU1. Interfaz de Usuario Intuitiva

- La interfaz es intuitiva y fácil de usar, construida con React y utilizando componentes de Mantine UI para consistencia visual.
- En esta interfaz, se han empleado elementos visuales claros y organizados, como menús desplegables y botones destacados, para facilitar la navegación. Se ha diseñado pensando en la experiencia del usuario, con una disposición coherente de los elementos y una estructura de página que guía al usuario de manera intuitiva a través de las diferentes secciones del sitio. Además, se han implementado indicadores visuales y mensajes claros para orientar al usuario durante el proceso de reserva y pago.

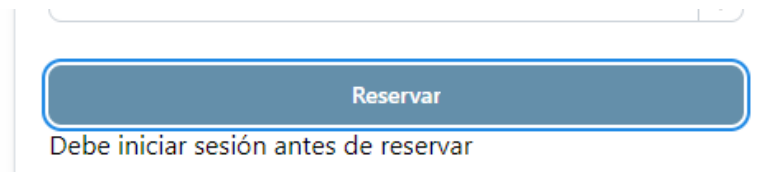


Figura 1. Indicadores de guía.

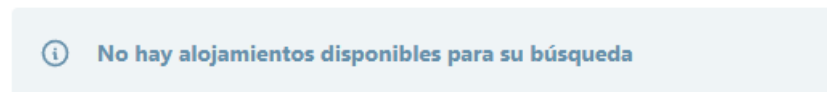


Figura 2. Información de los filtros.

2.3.2.2 Interfaces Hardware.

IH1. Compatibilidad con Dispositivos Habituales

- Se garantiza que la plataforma sea compatible con una amplia variedad de dispositivos, desde ordenadores de escritorio hasta dispositivos móviles.
- Se ha optimizado el diseño y la funcionalidad de la plataforma para adaptarse a diferentes tamaños de pantalla y resoluciones, asegurando una experiencia de usuario consistente en todos los dispositivos. Se han realizado pruebas exhaustivas en diferentes dispositivos y se ha ajustado el diseño y la disposición de los elementos para garantizar una visualización óptima en cada dispositivo.

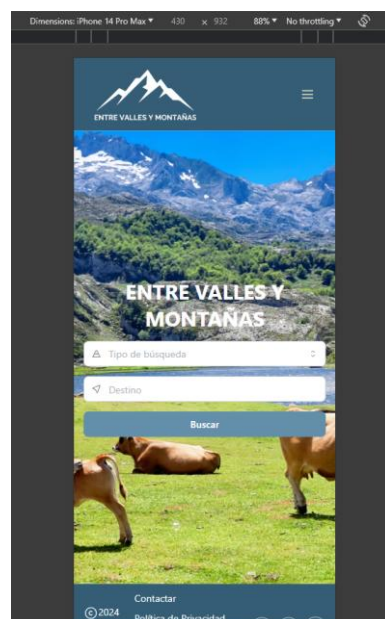


Figura 3. Versión móvil responsiva.



Figura 4. Versión tablet responsiva.



Figura 5. Versión ordenador responsiva.

2.3.2.3 Interfaces Software.

IS1. Compatibilidad con Navegadores Web

- Asegurar que la plataforma funcione correctamente en los principales navegadores web.
- Se ha probado la plataforma en una variedad de navegadores web, incluyendo Google Chrome, Mozilla Firefox, Safari y Microsoft Edge, para garantizar su correcto funcionamiento y renderizado. Se han utilizado estándares web modernos y se ha optimizado el código para asegurar una compatibilidad óptima con cada navegador. Además, se han realizado ajustes específicos para abordar posibles problemas de compatibilidad y garantizar una experiencia de usuario sin problemas.

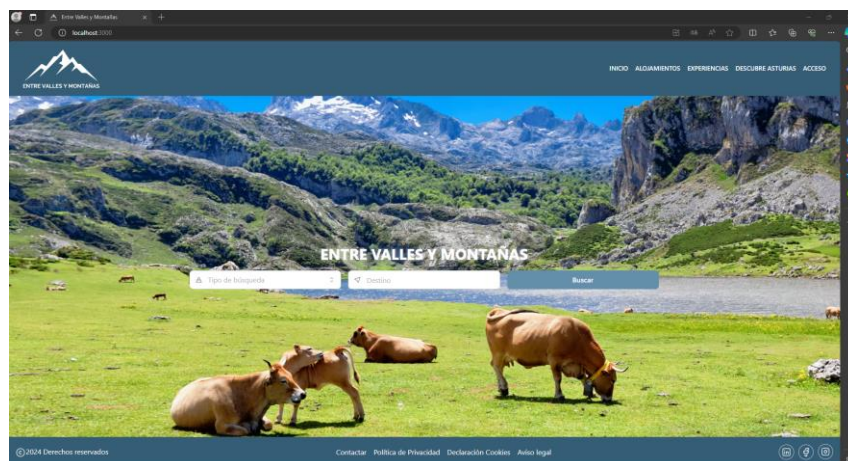


Figura 6. Navegador Microsoft Edge.

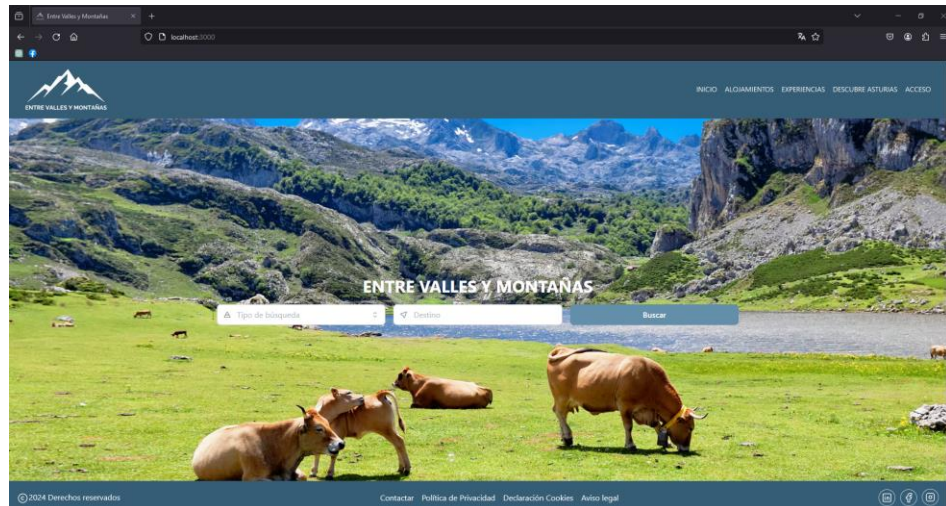


Figura 7. Navegador Mozilla Firefox.

2.3.2.4 Interfaces de Comunicaciones.

IC1. Seguridad en Comunicaciones con Stripe

- Se protege la seguridad de las transacciones de pago realizadas a través de Stripe.
- Cuando un usuario realiza una reserva de alojamiento o actividad, la plataforma se comunica de forma segura con Stripe para procesar el pago. Se implementa la integración con Stripe utilizando protocolos de comunicación seguros como HTTPS, garantizando así la protección de la información sensible del usuario durante la transmisión. Además, se utilizan tokens de seguridad para proteger los datos de la tarjeta del usuario durante la transacción y se realizan pruebas exhaustivas para verificar la seguridad y la integridad de las comunicaciones con Stripe.

Número de noches	4
Huéspedes	2
Precio Total	480 €
Número de tarjeta	4242 4242 4242 4242
Caducidad	CVC
02 / 25	123
País	España
Pagar Reserva	

Figura 8. Métodos de pago.

2.3.3 Requerimientos de Rendimiento.

RR1. Rendimiento y Eficiencia del Sistema

- **Tiempo de respuesta de la página**
 - Carga inicial de la página: Cualquier página del sistema debe cargar en un tiempo inferior a 2 segundos. Esto garantiza que los usuarios puedan acceder rápidamente a la información que necesitan sin experimentar demoras frustrantes.
 - Interacciones de usuario: Las acciones del usuario, como aplicar filtros o navegar entre páginas, deben reflejarse en menos de 1 segundo para asegurar una experiencia de usuario fluida.



Figura 9. Tiempo de respuesta Fron-end.

- **Tiempo de respuesta del servicio backend.**
 - Consultas de datos: Todos los servicios backend relacionados con la obtención de datos, como la visualización de alojamientos y actividades, deben responder en un tiempo inferior a 2 segundos.
 - Procesamiento de reservas: Funciones críticas como la reserva de alojamientos y actividades, incluyendo el procesamiento de pagos a través de Stripe, deben completarse en menos de 2 segundos.
 - Autenticación: Las operaciones de inicio y cierre de sesión deben responder en menos de 1 segundo.

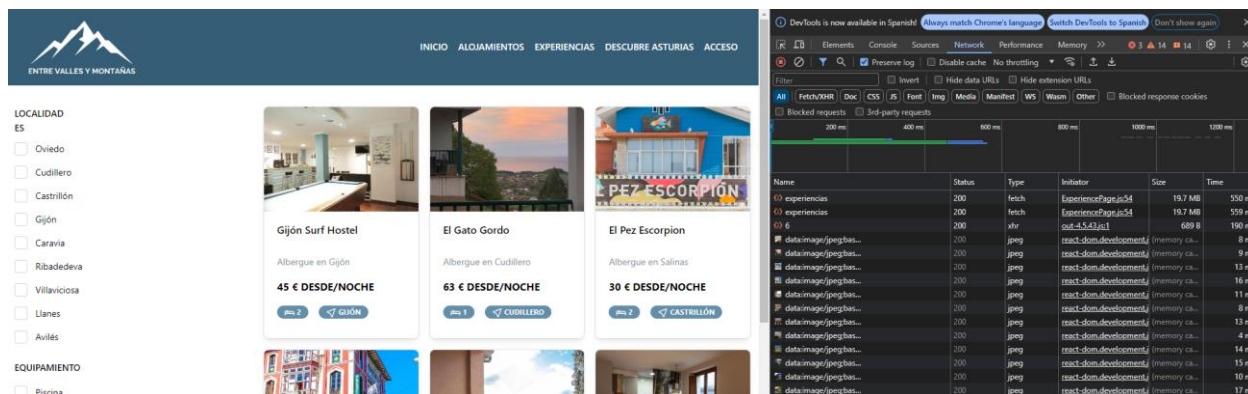


Figura 10. Tiempo de respuesta del Back-end.

2.3.4 Obligaciones del Diseño.

2.3.4.1 Estándares Cumplidos.

OD1. Cumplimiento de Estándares de Desarrollo Web

- El código JavaScript sigue las mejores prácticas y estándares de desarrollo web.
- Se realiza una validación del código utilizando herramientas como JSLint o ESLint para garantizar la calidad y corrección del código JavaScript.
- Además de las herramientas mencionadas, se utilizan estándares de accesibilidad web (WCAG) para asegurar que la plataforma sea accesible para todos los usuarios, incluyendo aquellos con discapacidades.
- El análisis de calidad de la aplicación se lleva a cabo utilizando Lighthouse, una herramienta automatizada de código abierto. Esta herramienta verifica el rendimiento, la accesibilidad, las mejores prácticas y el SEO de la aplicación, garantizando que se cumplan todos los estándares establecidos.

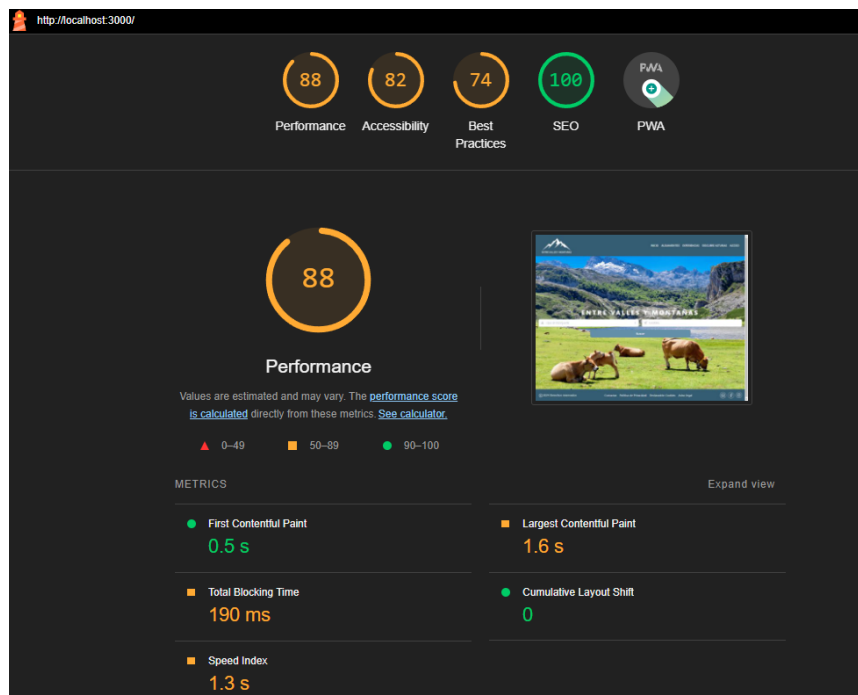


Figura 11. Lighthouse medidor calidad de una web.

2.3.4.2 Limitaciones Hardware.

OD2. Consideraciones de Hardware

- Se tienen en cuenta las limitaciones de hardware para garantizar un rendimiento óptimo en una amplia gama de dispositivos y configuraciones.
- Pruebas de rendimiento en diferentes dispositivos y configuraciones, optimización de recursos para minimizar el uso de memoria y CPU.

2.3.5 Atributos.

2.3.5.1 Seguridad.

AS1. Seguridad de la Información

- Se implementan medidas de seguridad robustas para proteger la información del usuario y los datos de pago.
- Encriptación de datos utilizando algoritmos seguros, gestión adecuada de sesiones y tokens de autenticación, auditorías de seguridad periódicas.

2.3.5.2 Facilidades de Mantenimiento.

AF1. Mantenibilidad del Código

- Se sigue un enfoque de diseño limpio y modular para facilitar el mantenimiento y la actualización del código de la aplicación.
- Se implementó una arquitectura basada en el patrón de diseño MVC (Modelo-Vista-Controlador), combinada con una API REST en el backend, para separar la lógica de negocio de la presentación, facilitando así la integración del backend con cualquier capa visual al hacerlas independientes.

2.3.5.3 Portabilidad.

AP1. Portabilidad de la Plataforma

- La aplicación se desarrolla utilizando tecnologías y frameworks compatibles con diferentes sistemas operativos y entornos de ejecución.
- Uso de herramientas de desarrollo multiplataforma como React.js, evitando dependencias específicas del sistema operativo.

2.3.5.4 Otros Requerimientos.

AO1. Cumplimiento Legal y Normativo

- La plataforma cumple con todas las leyes y regulaciones aplicables, incluyendo la protección de datos y la privacidad del usuario.
- Revisión de las leyes y regulaciones relevantes, implementación de políticas y procedimientos adecuados, colaboración con expertos legales en materia de protección de datos.

3. ANÁLISIS

3.1 Introducción.

En esta sección, exploraremos el modelo de clases del sistema, tanto con relaciones entre ellas como sin relaciones, para comprender la estructura y la organización de los datos. Además, examinaremos el diagrama de casos de uso para identificar cómo los actores interactúan con las diversas funcionalidades del sistema. A través de este análisis, buscamos obtener una visión clara y detallada de las características y el flujo de trabajo del sistema, lo que servirá como base para su desarrollo y diseño.

3.2 Diagrama de Clases.

- **Modelo de Clases Relacionales.**

Este diagrama muestra las clases con relaciones entre ellas. También indica mediante líneas el tipo de relación, en este caso se han usado para todas el de asociación. Este tipo es el llamado uno a muchos o uno a n.

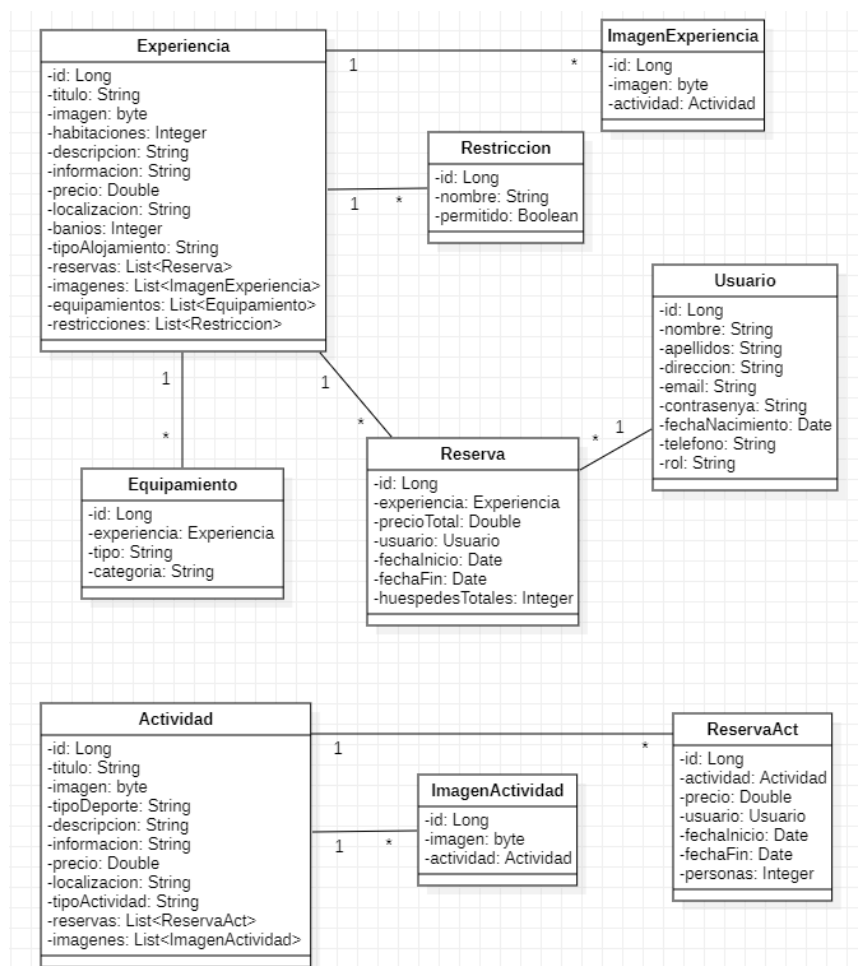


Figura 12. Diagrama de Clases

- **Modelo de Clases sin Relacionales.**

Este modelo muestra las características individuales de las clases que no tienen relación con ninguna otra. Simplemente aparecen en el sistema como objeto que pasa la información del front-end al back-end.

Estas clases se utilizan como DTO (Data Transfer Object) y su función es intercambiar datos entre la capa de presentación y la capa de lógica de la aplicación.

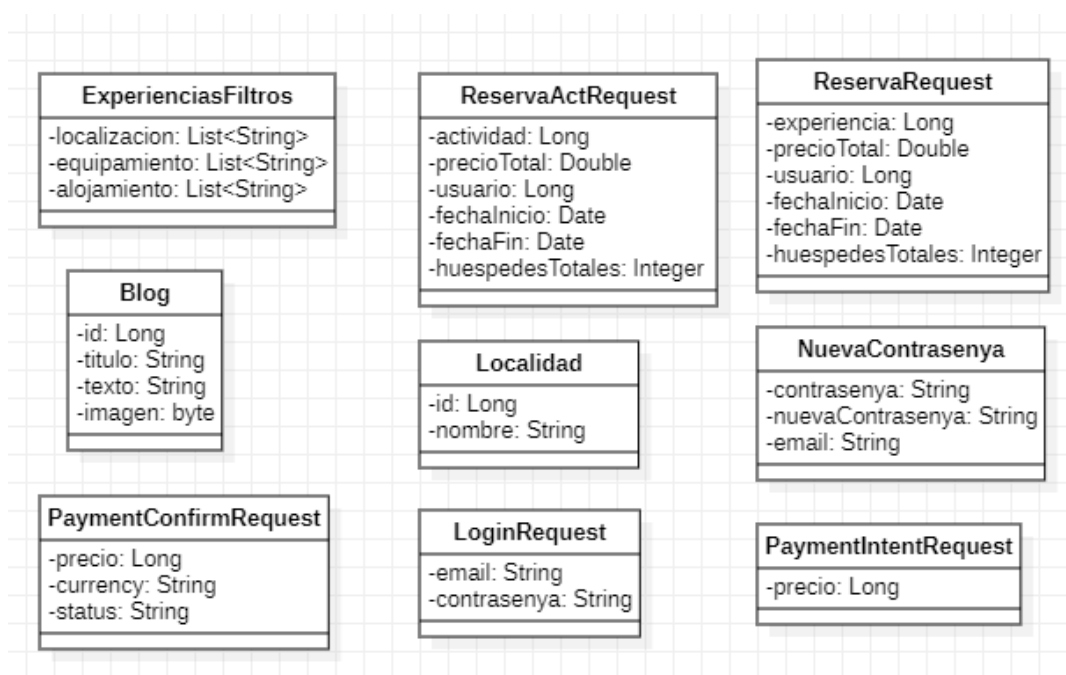


Figura 13. Diagrama de Clases DTO

3.3 Diagrama de Casos de Uso.

Un diagrama de casos de uso identifica los componentes principales y las acciones en un sistema. Los componentes principales, llamados "actores", y las acciones, llamadas "casos de uso", muestran cómo los actores interactúan con cada acción.

- **Actores:** estos van a representar a personas que van a interactuar con la web del proyecto.

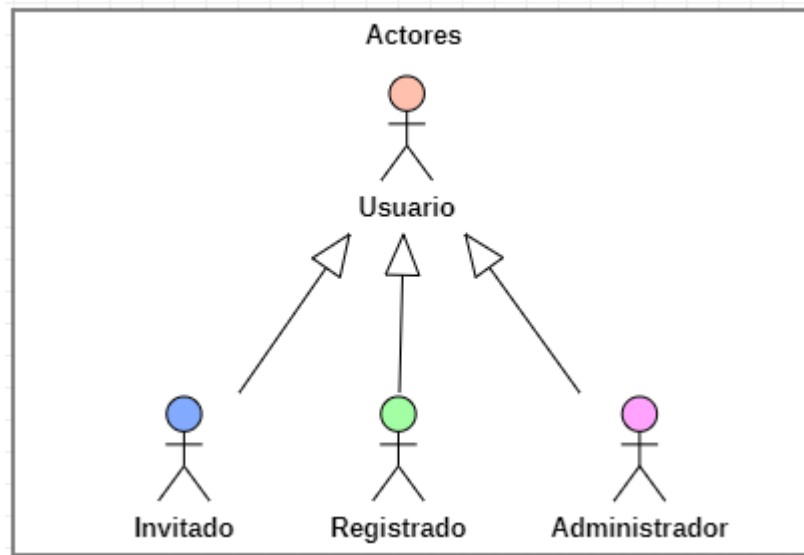


Figura 14. Tipos de actor

- **Tipos de usuario.**
 - **Usuario:** Este usuario es el principal, puede buscar alojamientos y actividades. Además, tiene la opción de registrarse en el sistema y poder leer el blog. Los otros actores tendrán características específicas que solo ellos pueden realizar. En este contexto, todas las acciones que puede realizar el usuario también están disponibles para el actor invitado.

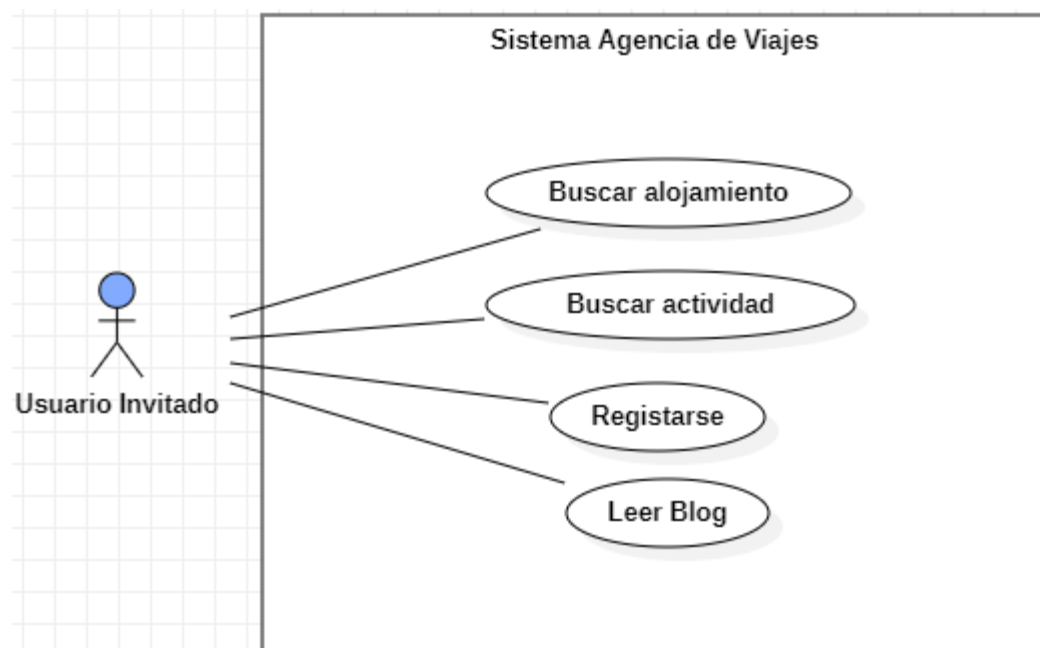


Figura 15. Casos de Uso Usuario Invitado

- **Usuario registrado:** Este actor es un invitado que se ha registrado en el sistema. Las acciones permitidas a un usuario registrado son: iniciar sesión en el sistema, después de estar logueado puede reservar actividad, reservar alojamiento, cambiar contraseña y realizar el pago después de hacer una reserva.

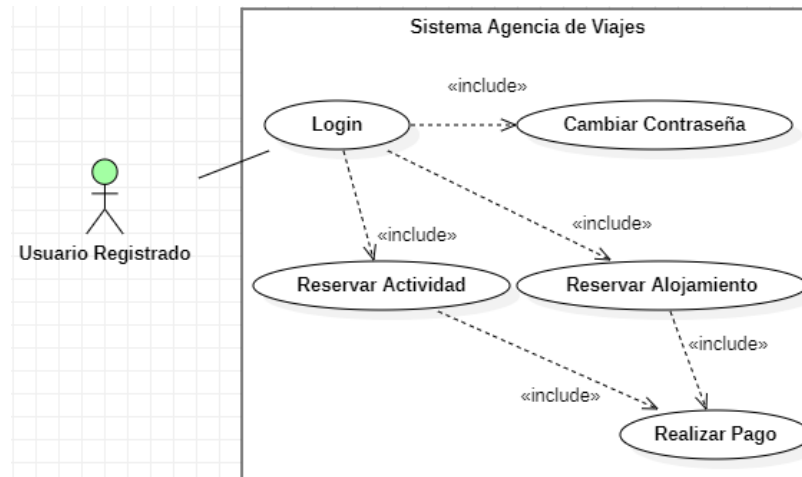


Figura 16. Casos de Uso Usuario Registrado

- **Usuario administrador:** Este actor es un usuario con privilegios especiales en el sistema. Además de las acciones permitidas a un usuario registrado, el administrador puede realizar actividades exclusivas de su rol. Para acceder a estas funciones, el administrador debe iniciar sesión con sus credenciales de administrador. Una vez autenticado, puede publicar nuevas entradas en el blog, crear nuevos alojamientos o actividades y también interactuar con todas las funcionalidades disponibles para los usuarios.

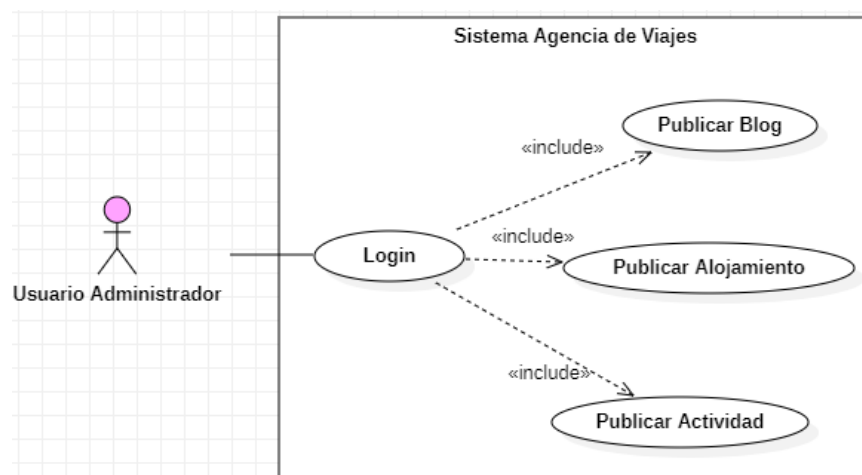


Figura 17. Casos de Uso Usuario Administrador

4. DISEÑO

4.1 Introducción.

En esta sección se describe la arquitectura de diseño de la plataforma "Entre Valles y Montañas", detallando cada una de las capas principales: Presentación, Negocio y Persistencia.

4.2 Capa de Presentación.

La capa de presentación se centra en ofrecer una experiencia de usuario fluida y atractiva. Para lograrlo, se utiliza el framework frontend React.js junto con la librería Mantine de componentes visuales.

Mantine complementa esto con una variedad de componentes predefinidos que simplifican el diseño y permiten una rápida iteración. Además, proporciona clases CSS predefinidas para garantizar que la interfaz sea fácilmente adaptable a diferentes dispositivos y tamaños de pantalla, asegurando así la accesibilidad desde cualquier dispositivo.

A continuación, se muestra como se han utilizado la capacidad de reutilización de los componentes que ofrece React, para diseñar la aplicación en pequeños componentes independientes.

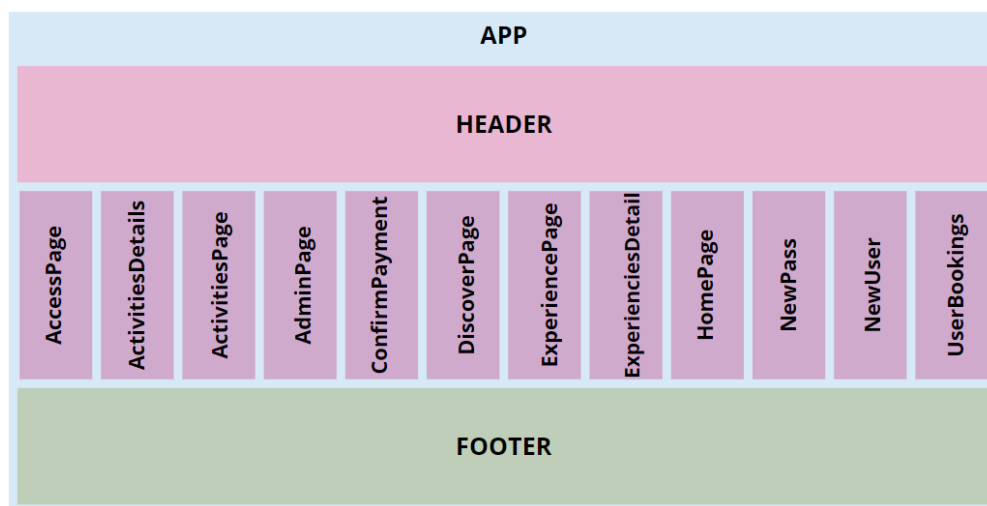


Figura 18. Diseño de la estructura por componentes

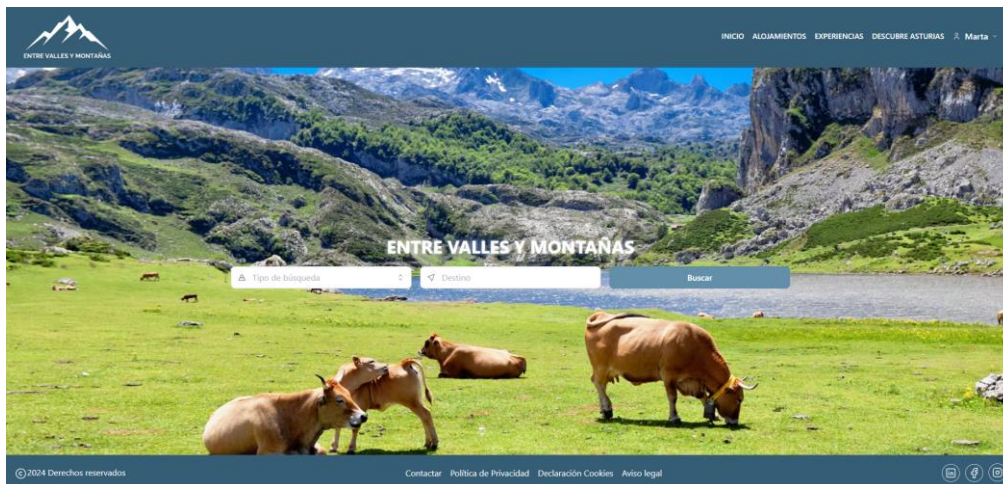


Figura 19. Pantalla de Inicio

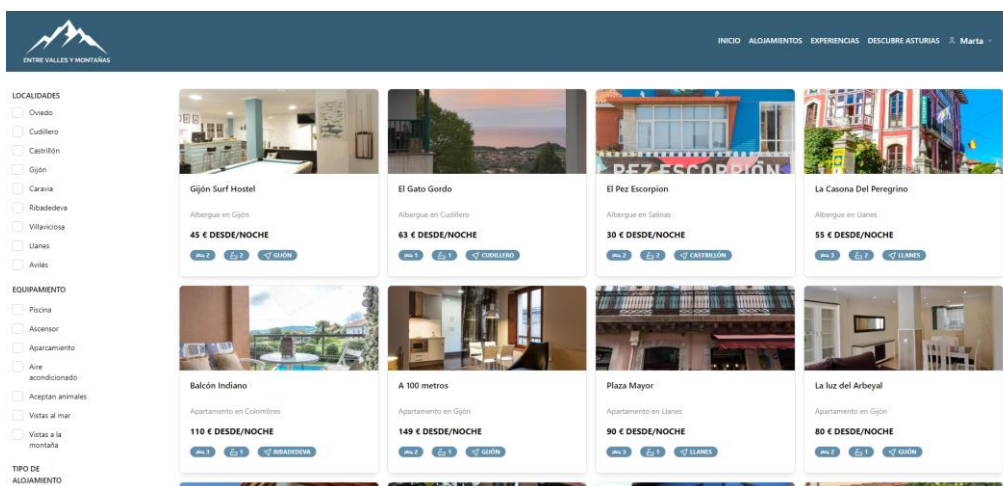


Figura 20. Pantalla de Alojamientos

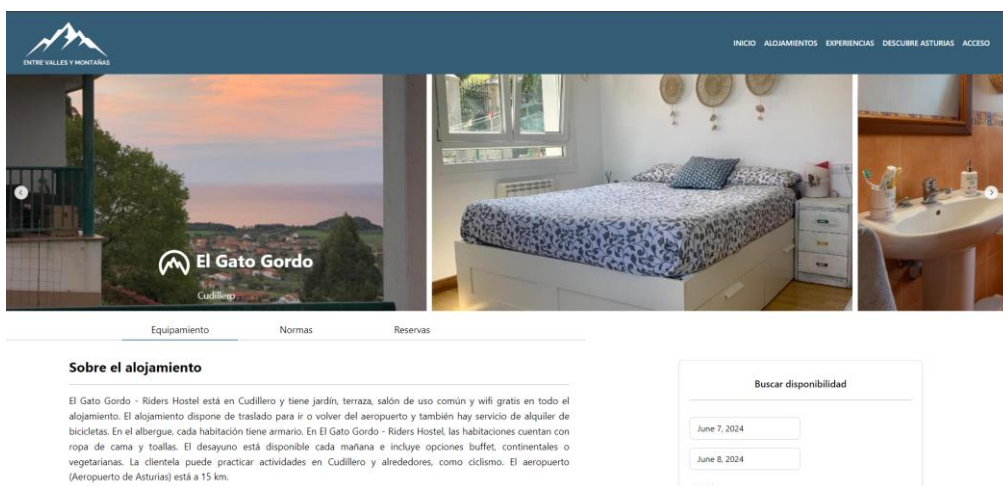


Figura 21. Pantalla de detalles del Alojamiento

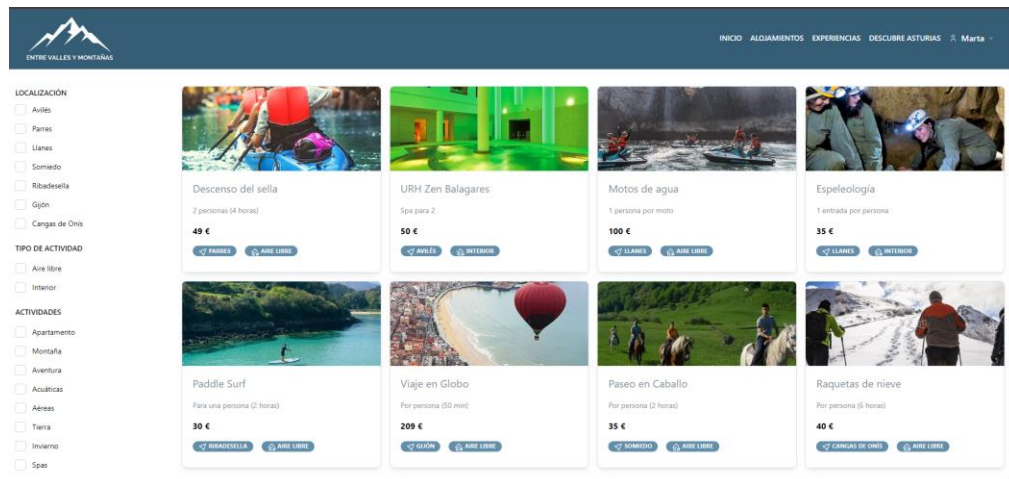


Figura 22. Pantalla de Experiencias



Figura 23. Pantalla del Blog

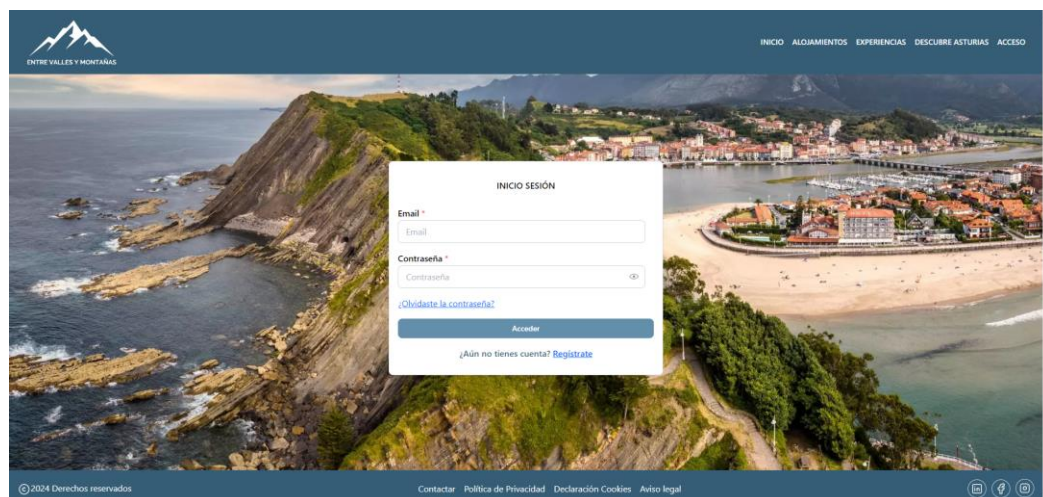


Figura 24. Pantalla de Acceso

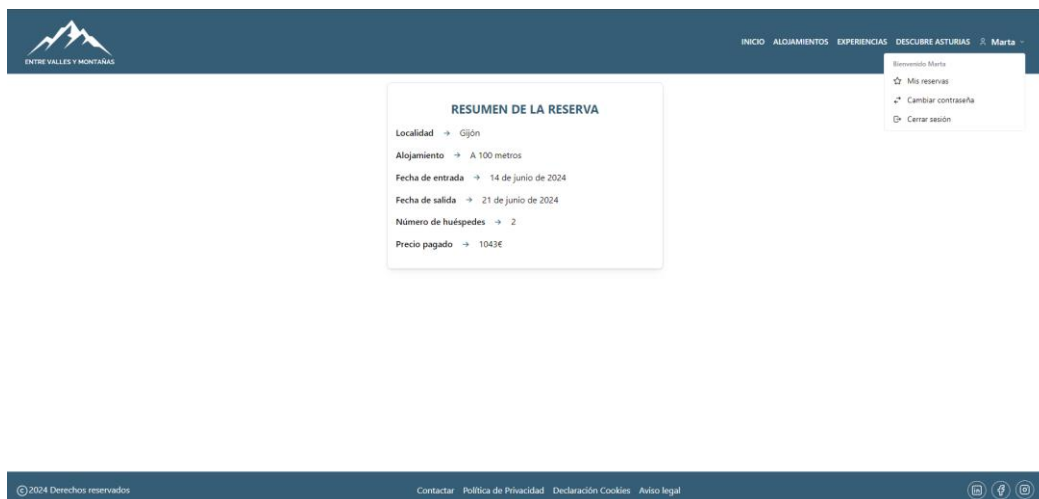


Figura 25. Pantalla de Reservas

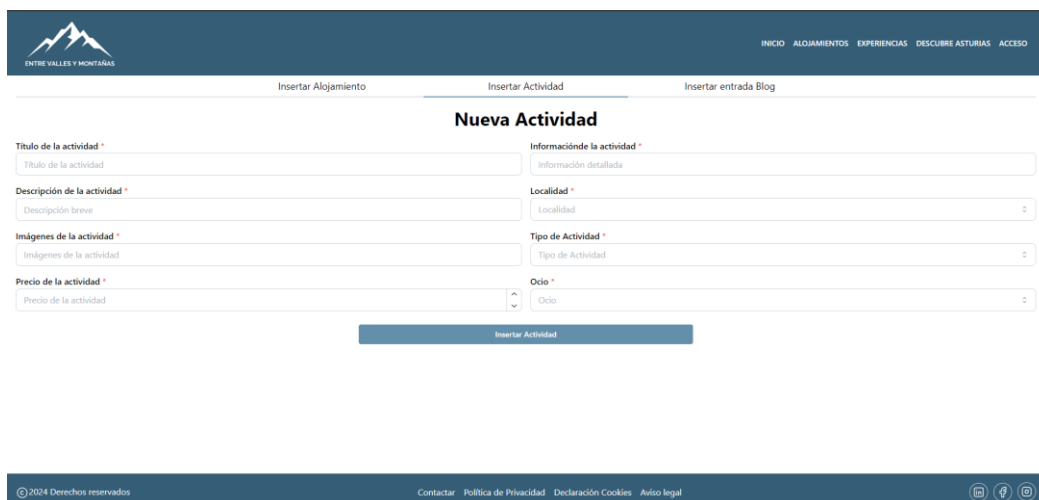


Figura 26. Pantalla de Insertar Actividad

4.3 Capa de Negocio o Lógica de la Aplicación.

La capa de negocio o lógica de la aplicación se encuentra en el backend del proyecto y se desarrolla utilizando SpringBoot junto con Java. Esta capa se encarga de procesar las solicitudes del cliente, manejar la lógica del negocio y coordinar las operaciones entre la capa de presentación y la capa de persistencia.

Esta capa implementa funcionalidades clave como la autenticación de usuarios, la gestión de reservas y la lógica de negocio relacionada con la planificación y reserva de viajes en la región de Asturias. Utilizando SpringBoot, se facilita la creación de APIs RESTful que permiten la comunicación eficiente entre el frontend y el backend de la aplicación.

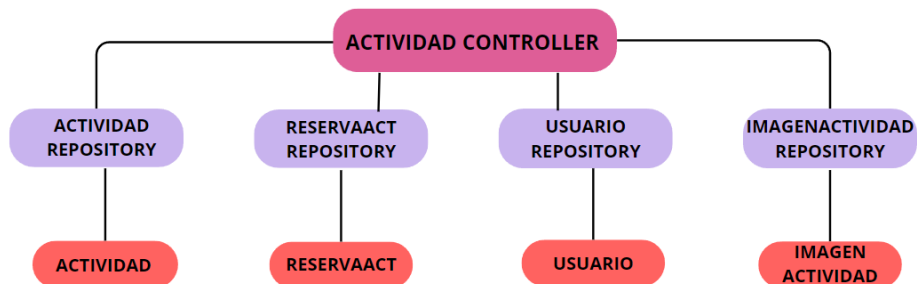


Figura 27. Diagrama del flujo de la Actividad



Figura 28. Diagrama del flujo de la Experiencia

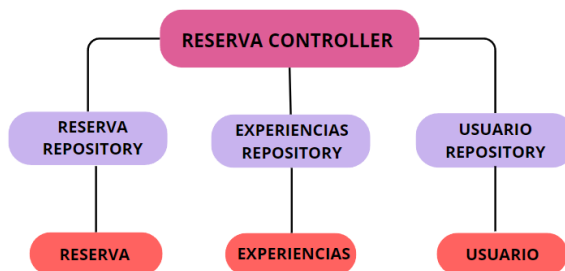


Figura 29. Diagrama del flujo de la Reserva

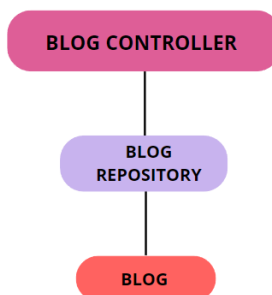


Figura 30. Diagrama del flujo del Blog

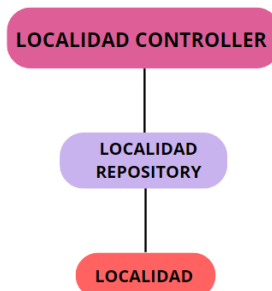


Figura 31. Diagrama del flujo de la Localidad

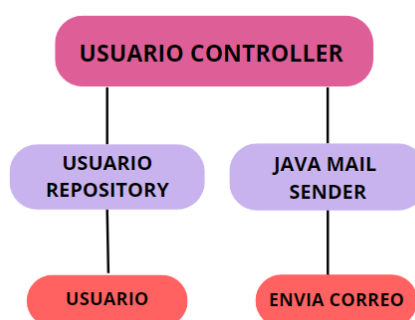


Figura 32. Diagrama del flujo del Usuario

4.4 Capa de Persistencia o Datos.

La capa de persistencia o datos se encarga de gestionar el almacenamiento y recuperación de la información necesaria para el funcionamiento de la aplicación. Se utiliza una base de datos relacional MySQL para almacenar datos como información de usuarios, alojamientos, actividades y reservas.

La comunicación con la base de datos se realiza a través de una capa de acceso a datos que utiliza tecnologías como JPA para interactuar con la base de datos de manera eficiente y segura. Esta capa garantiza la integridad y consistencia de los datos almacenados, así como la optimización de las consultas para mejorar el rendimiento de la aplicación.

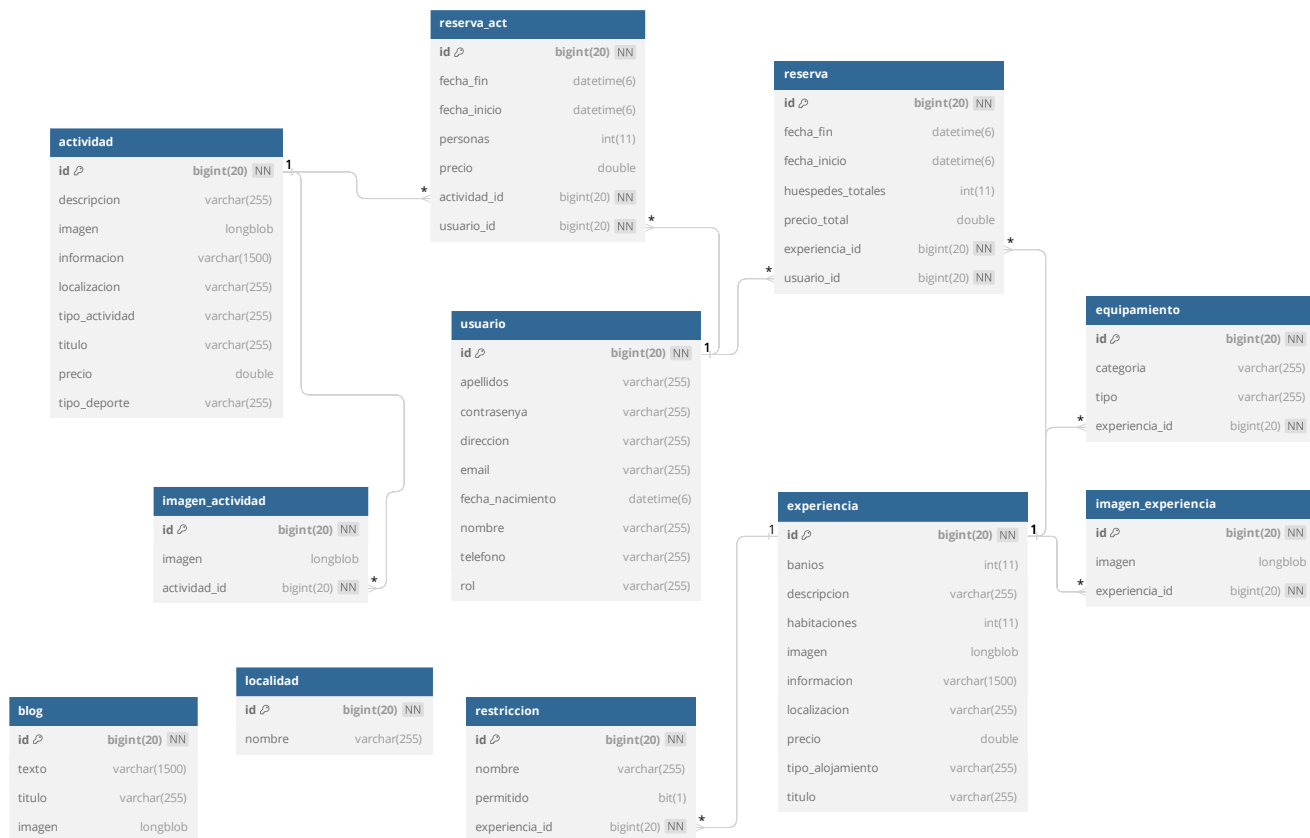


Figura 33. Diagrama de Entidades BBDD

5. IMPLEMENTACIÓN

5.1 Tecnologías utilizadas en el desarrollo del proyecto.

Tecnologías capa de presentación (Frontend)

- **React.js:** framework de JavaScript para construir interfaces de usuario interactivas.
- **HTML5:** El estándar para estructurar y presentar contenido en la web.
- **CSS3:** utilizado para estilizar y diseñar la apariencia de las páginas web.
- **JavaScript:** lenguaje de programación principal utilizado en el frontend para agregar interactividad.
- **Mantine:** librería de componentes React para la construcción de interfaces de usuario modernas y accesibles.
- **JSX:** Una extensión de JavaScript que permite escribir HTML dentro de JavaScript.

Tecnologías capa de negocio (Backend)

- **SpringBoot:** Un framework de Java que simplifica el desarrollo de aplicaciones Java EE y Spring.

Tecnologías capa de datos.

- **MySQL:** Un sistema de gestión de bases de datos relacional de código abierto.
- **PostgreSQL:** Otro sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez y funcionalidades avanzadas.

Librerías para procesamiento de pagos

- **Stripe:** Una plataforma de pagos en línea que permite a las empresas aceptar pagos por Internet.

Envío de correos.

- **Servidor GMAIL SMTP:** servicio de envío de correos electrónicos utilizando el servidor SMTP de Gmail para enviar correos electrónicos desde la aplicación.

Despliegue de la aplicación

- **Render:** Un servicio de alojamiento en la nube que facilita el despliegue de aplicaciones web.

Gestión de Dependencias.

- **NPM:** El sistema de gestión de paquetes de Node.js utilizado para instalar y gestionar las dependencias del proyecto en el frontend.
- **Maven:** Una herramienta de gestión de proyectos de software utilizada principalmente para la construcción y gestión de proyectos Java.

Control de Versiones.

- **GitHub:** Una plataforma de desarrollo colaborativo de software que utiliza el sistema de control de versiones Git para el seguimiento de cambios en el código fuente.

5.2 Descripción del Proyecto.

5.2.1 Capa de Presentación.

La capa de presentación como he comentado anteriormente, esta realizada en React con la librería de componentes visuales Mantine, a continuación, explico algunos detalles de implementación de dicha capa.

Comunicación con la capa de negocio

La capa de presentación se comunica con la capa de lógica realizando peticiones HTTP a los servicios expuestos por SpringBoot, para ello, se usan llamadas fetch tanto GET como POST para poder enviar y recoger datos del servidor.

```
fetch(`${url()}/actividad/~ + params["id"]`)
  .then(response => response.json())
  .then(data => {
    console.log(data);
    setExperiencia(data);
  })
  .catch(error => console.error('Error fetching users:', error));
```

Figura 34. Ejemplo petición HTTP - Capa Presentación

Comunicación con el sistema de pagos Stripe

La aplicación, cuando un usuario reserva un alojamiento o una actividad, se comunica con Stripe para realizar el pago, Stripe proporciona algunas tarjetas de ejemplo para pruebas, alguna de las cuales se añade a continuación:

Marca	Número	CVC	Fecha Caducidad
Visa	4242424242424242	3 dígitos aleatorios	Cualquier Fecha Futura
Visa (débito)	4000056655665556	3 dígitos aleatorios	Cualquier Fecha Futura
Mastercard	5555555555554444	3 dígitos aleatorios	Cualquier Fecha Futura

Se ha generado un API Key en la plataforma de Stripe

Desarrolladores

Resumen Claves de API Webhooks Eventos Registros Aplicaciones

Claves de API

[Más información sobre la autenticación de API →](#)

Claves estándar

Crea una clave que desbloquea el acceso completo a la API, lo que permite una amplia interacción con tu cuenta. [Más información](#)




NOMBRE	TOKEN	ÚLTIMA VEZ QUE SE USÓ	DE CREACIÓN
Clave publicable	pk_test_51PFatYRxV1RVND84NS81925iyGjIsEdL10igwkgPlHcPwWwxBcXxm5eN9vnOqjp5fsg1FaY4nuFoSqNAAp9uu2jY002xmZtHtR	23 may.	 12 may. ...
Clave secreta	 <input type="button" value="Revelar la clave de prueba"/>	23 may.	 12 may. ...

Figura 35. Configuración Stripe

Con esto, se ha generado un servicio de backend, al cual React invoca para hacer la autorización previa con Stripe

```
@PostMapping("/payment")
public ResponseEntity<String> createPaymentIntent(@RequestBody PaymentIntentRequest intent) {
    Stripe.apiKey = stripeSecretKey;

    try {
        PaymentIntent paymentIntent = PaymentIntent.create(
            new PaymentIntentCreateParams.Builder()
                .setCurrency("eur")
                .setAmount(intent.getPrecio()) // Monto en centavos
                .build()
        );

        // Devuelve el Client Secret
        return ResponseEntity.ok(paymentIntent.getClientSecret());
    } catch (StripeException e) {
        e.printStackTrace();
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Error al crear el PaymentIntent");
    }
}
```

Figura 36. Servicio Backend

Se invoca a este servicio desde el Frontend

```
fetch(`${url()}/payment`, requestOptions)
    .then(response => response.text())
    .then(data => {
        console.log(data);
        setOptions({clientSecret: data});
        console.log(options);
    })
    .catch(error => console.error('Error fetching users:', error));
```

Figura 37. Servicio Fronted

Una vez el cliente confirma el pago, se llama a la plataforma de Stripe para registrar el mismo.

```
const stripePayment = async (data) => {
  const { error } = stripe.confirmPayment({
    // 'Elements' instance that was used to create the Payment Element
    elements,
    confirmParams: {
      return_url: `${window.location.origin}`,
    },
    redirect: 'if_required'
  }).then(function(result) {
    if (result.error) {
      // Show error to your customer (for example, payment details incomplete)
      console.log(result.error.message);
    } else {
      console.log(result);
      navigate(`/confirmPayment/${data.id}/${user.id}?payment_intent=${result.paymentIntent.id}&tipo=actividad`);
    }
  });
};
```

Figura 38. Confirmación de pago

Navegación entre pantallas

Para la navegación entre pantallas, se ha utilizado el router de React y las rutas se definen en el archivo App.js el cual redirige a una pantalla u otra en función de la URL actual.

```
function App() {
  return (
    <UserProvider>
      <Router>
        <Routes>
          <Route exact path="/" element={<HomePage/>} />
          <Route exact path="/digs" element={<ExperiencePage/>} />
          <Route exact path="/experiencias/:id" element={<ExperienciasDetail/>} />
          <Route exact path="/activities" element={<ActivitiesPage/>} />
          <Route exact path="/activities/:id" element={<ActivitiesDetail/>} />
          <Route exact path="/discover" element={<DiscoverPage/>} />
          <Route exact path="/access" element={<AccessPage/>} />
          <Route exact path="/newuser" element={<NewUser/>} />
          <Route exact path="/newpass" element={<NewPass/>} />
          <Route exact path="/bookings" element={<UserBookings/>} />
          <Route exact path="/confirmPayment/:reservaId/:userId" element={<ConfirmPayment/>} />
          <Route exact path="/admin" element={<AdminPage/>} />
        </Routes>
      </Router>
    </UserProvider>
  );
}
```

Figura 39. Rutas de Pantallas

Guardado datos usuario logueado

Para guardar los datos del usuario logueado, se ha utilizado un hook de React, los hook de React, permiten mantener datos entre diferentes componentes / pantallas de la aplicación, permitiendo no perder la sesión del usuario.

```
import React, { useState, useEffect, createContext, useContext } from 'react'; 7k (gzipped: 2.8k)

// Creamos un contexto para el usuario conectado
const UserContext = createContext();

// Creamos un componente de proveedor que almacenará el usuario conectado y proporcionará funciones para establecerlo y obtenerlo
export const UserProvider = ({ children }) => {
  const [user, setUser] = useState({});

  // Aquí puedes agregar lógica para obtener el usuario de tu backend o cualquier otro método de autenticación que estés utilizando
  const updateUser = (newUser) => {
    console.log(newUser);
    setUser(newUser);
  };

  return (
    <UserContext.Provider value={{ user, updateUser }}>
      {children}
    </UserContext.Provider>
  );
};

// Hook personalizado para acceder al usuario conectado en cualquier componente
const useUser = () => {
  const context = React.useContext(UserContext);
  if (context === undefined) {
    throw new Error('useUser debe ser usado dentro de un UserProvider');
  }
  return context;
};

export { useUser };
```

Figura 40. Mantener el Usuario entre pantallas

Pantalla ejemplo React

Se muestra un ejemplo de una pantalla de la aplicación construida con Mantine

```
return (
  <>
    <Header></Header>
    {blogs ? (
      <Container size='xxl' mb='md' mt='md'>
        <Title order={1} className='blogTitle' mb='md'>Descubre Asturias</Title>
        <Grid justify='center' align='center' gutter='md' overflow='hidden'>
          {blogs.map((element, index) => {
            if (index % 2 == 0) {
              return (
                <<Grid.Col span={{ base: 12, xs: 4 }} className='imageColumn'>
                  <img src={element.url} alt='Imagen 1' className='roundedImage' />
                </Grid.Col>
                <<Grid.Col span={{ base: 12, xs: 8 }} style={{ padding: '20px' }}>
                  <Title order={3} className='title'>{element.titulo}</Title>
                  <Text className='text-align-justify'>{element.texto}</Text>
                </Grid.Col></>
              )
            } else {
              return (
                <<Grid.Col span={{ base: 12, xs: 8 }} style={{ padding: '20px' }}>
                  <Title order={3} className='title'>{element.titulo}</Title>
                  <Text className='text-align-justify'>{element.texto}</Text>
                </Grid.Col>
                <<Grid.Col span={{ base: 12, xs: 4 }} className='imageColumn'>
                  <img src={element.url} alt='Imagen 2' className='roundedImage' />
                </Grid.Col></>
              )
            }
          })}
        </Grid>
      </Container>
    ) : (null)}
    <Footer></Footer>
  </>
);
```

Figura 41. Código usando librería Mantine

Como se puede ver, se ha establecido el contenido de la página entre los componentes Header y Footer, los cuales como vimos en los diagramas anteriormente, son componentes reutilizables en todas las pantallas de la aplicación.

5.2.2 Capa de Negocio o Lógica de la Aplicación.

La capa de negocio de la aplicación se ha realizado exponiendo servicios HTTP con SpringBoot como framework.

Ejemplo Flujo Servicio Expuesto

A continuación, vamos a tomar el servicio de "detalle alojamiento" para explicar cómo se ha realizado la implementación de la capa de negocio.

En primer lugar, cuando llega la petición, llegará a la capa de controladores web.

```
@GetMapping("/experiencias/{id}")
public @ResponseBody Experiencia experienciasDetail(@PathVariable("id") Long id) {
    Experiencia experiencia = this.experienciasRepository.findById(id).get();
    for(Reserva r:experiencia.getReservas()) {
        r.setExperiencia(null);
    }
    return experiencia;
}
```

Figura 42. Consultas Backend

En este caso, la petición recibe desde el Frontend el id de la experiencia que se desea obtener de la base de datos.

Posteriormente el controlador, invoca al repositorio JPA pasando el id de la entrada.

```
@Repository
public interface ExperienciasRepository extends JpaRepository<Experiencia, Long>{

    @Query(value = "SELECT e FROM Experiencia e WHERE e NOT IN " +
        "(SELECT r.experiencia FROM Reserva r WHERE r.fechaInicio BETWEEN :fechaInicio AND :fechaFin AND r.fechaFin BETWEEN :fechaInicio AND :fechaFin)")
    public List<Experiencia> filterByBookDates(@Param("fechaInicio")Date fechaInicio,@Param("fechaFin")Date fechaFin);

    @Query(value = "SELECT e FROM Experiencia e LEFT JOIN e.equipoamentos eq WHERE (:localizacion IS NULL OR e.localizacion IN (:localizacion)) AND (:alojamiento IS NULL OR e.tipoAlojamiento = :alojamiento) AND (:equipamiento IS NULL OR e.equipoamentos IN (:equipamiento))")
    public List<Experiencia> filter(@Param("localizacion")List<String> localizacion, @Param("alojamiento")List<String> alojamiento, @Param("equipamiento")List<String> equipamiento);
}
```

Figura 43. Querys Backend

Este repositorio JPA, esta ligado a la entidad de Experiencia, la cual mediante anotaciones de JPA, tiene relación con la tabla de Experiencia de BBDD, JPA proporciona querys ya hechas por defecto, como es el caso del método findById, save, findAll utilizados en diferentes puntos de la aplicación.


```

14 @Entity
15 public class Experiencia {
16
17     @Id
18     @GeneratedValue(strategy = GenerationType.AUTO)
19     private Long id;
20
21     private String titulo;
22     @Lob
23     @Column(columnDefinition = "LONGBLOB")
24     private byte[] imagen;
25
26     private Integer habitaciones;
27
28     private String descripcion;
29     @Column(length = 1500)
30     private String informacion;
31
32     private Double precio;
33
34     private String localizacion;
35
36     private Integer banios;
37
38     private String tipoAlojamiento;
39
40
41     @OneToMany(mappedBy="experiencia")
42     private List<Reserva> reservas;
43
44     @OneToMany(mappedBy="experiencia")
45     private List<ImagenExperiencia> imagenes;
46
47     @OneToMany(mappedBy="experiencia")
48     private List<Equipamiento> equipamientos;

```

Figura 44. Anotaciones JPA

Envío de correos al usuario

Me he basado en el sistema de envío de correos, que ya proporciona Java por defecto (JavaMailSender)

```

public void sendSimpleMessage(
    String to, String subject, String text) {
    String result = null;
    MimeMessage message = emailSender.createMimeMessage();
    try {
        MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
        ClassPathResource resource = new ClassPathResource("static/cambioContrasenya/index.html");
        byte[] fileData = FileCopyUtils.copyToByteArray(resource.getInputStream());
        String htmlMsg = new String(fileData, "UTF-8");
        helper.setTo(to);
        helper.setFrom("entrevallesymontanas@gmail.com");
        helper.setSubject(subject);
        helper.setText(htmlMsg, true);
        // Configurar la imagen incrustada
        ClassPathResource image = new ClassPathResource("static/cambioContrasenya/images/image-1.png");
        helper.addInline("image-1.png", image);
        ClassPathResource image2 = new ClassPathResource("static/cambioContrasenya/images/image-2.png");
        helper.addInline("image-2.png", image2);
        ClassPathResource image3 = new ClassPathResource("static/cambioContrasenya/images/image-3.png");
        helper.addInline("image-3.png", image3);
        ClassPathResource image4 = new ClassPathResource("static/cambioContrasenya/images/image-4.png");
        helper.addInline("image-4.png", image4);
        ClassPathResource image5 = new ClassPathResource("static/cambioContrasenya/images/image-5.gif");
        helper.addInline("image-5.gif", image5);
        result = "success";
        emailSender.send(message);
    } catch (MessagingException | IOException e) {
        throw new MailParseException(e);
    } finally {
        if (result != "success") {
            result = "fail";
        }
    }
}

```

Figura 45. Sistema JavaMailSender

Lo que hice fue configurar el HTML a enviar en dicho correo, como recurso de la aplicación.

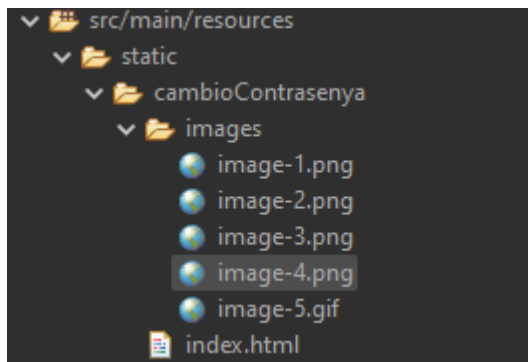


Figura 46. Configuración HTML

Maven como gestor de dependencias

Para usar las diferentes librerías de terceros, se han configurado en el archivo pom.xml de Maven, a continuación, se muestra un fragmento del mismo

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
    <version>3.1.5</version>
  </dependency>
  <dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.3.8</version>
  </dependency>
  <dependency>
    <groupId>com.stripe</groupId>
    <artifactId>stripe-java</artifactId>
    <version>20.86.0</version> <!-- Reemplaza con la última versión -->
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Figura 47. Dependencias Maven

Configuración para diferentes entornos

Dado que, al desplegar la aplicación, no se contará con el mismo tipo de BBDD ni esta estará en el mismo lugar que en local, se han añadido dos perfiles de Spring para arrancar la aplicación con diferentes configuraciones.

```
spring.application.name=entrevalles
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/entrevalles
spring.datasource.username=root
spring.datasource.password=alumno
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
server.port=8081
stripe.secretKey=sk_test_51PFatYRxV1RVND84dm85UsLq901w8DIN0zYCBaCfjJdAMy96fjkaSQ2xnt4HtPPQWtcJXbP9F0ovcJxSo0YuqaVR00JSiNTAN1
```

Figura 48. Configuración para local

```
spring.application.name=entrevalles
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform = org.hibernate.dialect.PostgreSQL94Dialect
spring.datasource.url=jdbc:postgresql://pg-cp51sn21hbls73f8kh70-a.frankfurt-postgres.render.com:5432/entrevalles
spring.datasource.username=root
spring.datasource.password=4uwAz5Z6u0eLcnTqBTahmmnrI32K4nPA
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
stripe.secretKey=sk_test_51PFatYRxV1RVND84dm85UsLq901w8DIN0zYCBaCfjJdAMy96fjkaSQ2xnt4HtPPQWtcJXbP9F0ovcJxSo0YuqaVR00JSiNTAN1
```

Figura 49. Configuración para despliegue

Al desplegar la aplicación, se especifica el profile de deploy, para que la aplicación arranque con la configuración de la BBDD remota PostgreSQL.

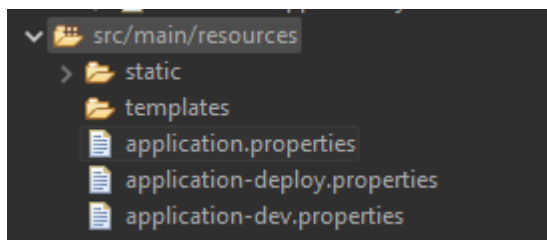


Figura 10. Archivo deploy

5.2.3 Capa de Persistencia o de Datos.

Anotaciones JPA

A continuación, se explican las diferentes anotaciones utilizadas en JPA para relacionar los objetos del backend a la BBDD.

@Entity: Establece un objeto como entidad JPA, si no se especifica el nombre de la tabla generada es el mismo que el del objeto.

@Id: Establece esta propiedad como id de la tabla.

@GeneratedValue(strategy = GenerationType.AUTO): Se indica que el id se autogenera por la BBDD.

@OneToMany(mappedBy="experiencia"): Anotación para hacer una relación 1:n con otra entidad JPA.

@Lob: Indica que el campo es de tipo BLOB, para poder guardar en esa columna archivos, imágenes etc de gran tamaño.

@ManyToOne: Relación n a 1 con otra entidad JPA

@JoinColumn(name="experiencia_id", nullable=false): Indica la columna que se creará como foreign key en una relación n a 1.

Esquema tablas BBDD

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
<input type="checkbox"/> actividad	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> actividad_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> blog	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	latin1_swedish_ci	128.0 KB	-
<input type="checkbox"/> blog_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> equipamiento	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	latin1_swedish_ci	32.0 KB	-
<input type="checkbox"/> equipamiento_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> experiencia	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> experiencia_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> imagen_actividad	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	8	InnoDB	latin1_swedish_ci	272.0 KB	-
<input type="checkbox"/> imagen_actividad_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> imagen_experiencia	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	17	InnoDB	latin1_swedish_ci	1.5 MB	-
<input type="checkbox"/> imagen_experiencia_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> localidad	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	77	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> localidad_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> reserva	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	latin1_swedish_ci	48.0 KB	-
<input type="checkbox"/> reserva_act	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	latin1_swedish_ci	48.0 KB	-
<input type="checkbox"/> reserva_act_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> reserva_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> restriccion	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	latin1_swedish_ci	32.0 KB	-
<input type="checkbox"/> restriccion_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> usuario	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	latin1_swedish_ci	16.0 KB	-
<input type="checkbox"/> usuario_seq	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	1	InnoDB	latin1_swedish_ci	16.0 KB	-
22 tablas	Número de filas	127	InnoDB	latin1_swedish_ci	2.3 MB	0 B

Figura 11. Base de Datos

Las tablas terminadas en _seq son los ID's de las tablas autogenerados en forma de secuencia.

6. EVALUACIÓN

6.1 Introducción.

La evaluación del sistema se realizó a través de pruebas de funcionalidad, rendimiento y compatibilidad para asegurar su correcto funcionamiento y usabilidad.

6.2 Validaciones de páginas de Estilo.

Se utilizaron herramientas de validación de CSS y HTML para asegurar que las páginas cumplen con los estándares web y son accesibles. La herramienta principal utilizada fue el **W3C Markup Validation Service**. Durante estas validaciones, se identificaron y corrigieron varios errores y advertencias, garantizando que el código de las páginas sea semánticamente correcto y accesible.

A continuación, se muestra un ejemplo de la validación de una página de la plataforma.

Información de CSS válida

```
.mainContainerHome {
  overflow-y : hidden;
  width : 100vh;
}

.bodyHome {
  background-image : url("/public/assets/lagos.avif");
  background-size : cover;
  background-repeat : no-repeat;
  background-position : center;
  display : flex;
  justify-content : center;
  align-items : center;
  height : 80vh;
  width : 100%;
  margin : 0;
  padding : 0;
}

.titleHome {
  background : radial-gradient(ellipse at bottom, #fff, transparent, transparent) 50% 100% / 50% 50% no-repeat;
  background-clip : text;
  color : #ffffff;
  font-size : 12vw;
  font-family : "Raleway", sans-serif;
  animation : reveal 3000ms 200ms ease-in-out forwards, glow 2500ms 2000ms linear infinite;
}

@keyframes reveal {
  80% {
    letter-spacing : 8px;
  }

  100% {
    background-size : 300% 300%;
  }
}

@keyframes glow {
  40% {
    text-shadow : 0 0 8px #fff;
  }
}
```

Figura 12. Validador de CSS W3C

6.3 Validación de Enlaces.

Se realizaron pruebas exhaustivas para verificar que todos los enlaces en la plataforma funcionan correctamente y llevan a las páginas esperadas.

No se encontraron enlaces rotos en la versión final de la plataforma, garantizando así una navegación fluida para los usuarios.

Para la validación de los enlaces, se utilizó **W3C Link Checker**, una herramienta gratuita proporcionada por W3C que permite verificar los enlaces en el sitio web. Esta herramienta ayudó a identificar y corregir cualquier enlace potencialmente problemático, asegurando la integridad y funcionalidad de todos los enlaces en la plataforma.

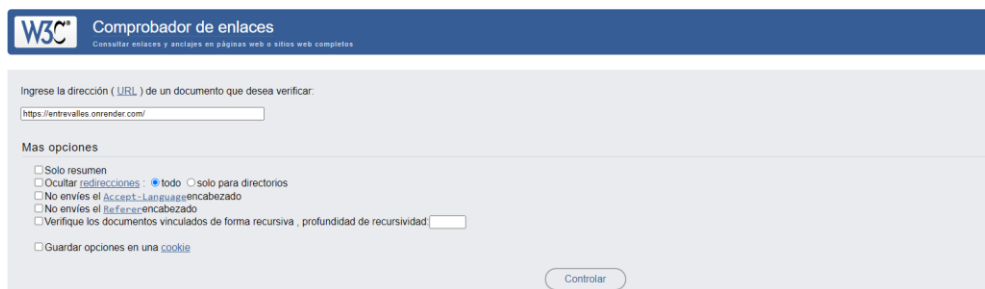


Figura 13. Validación Links W3C

Processing <https://entrevalles.onrender.com/>

Go to [the results](#).

For reliable link checking results, check [HTML validity](#) and [CSS validity](#) first.

Back to the [link checker](#).

Status: Done. Document processed in 12.65 seconds.

```
Parsing...
done (1 lines in 0.00 seconds).
Checking anchors...
done.

Checking link https://entrevalles.onrender.com/logo.ico
HEAD https://entrevalles.onrender.com/logo.ico fetched in 2.36 seconds

Checking link https://unpkg.com/leaflet@1.0.1/dist/leaflet.css
HEAD https://unpkg.com/leaflet@1.0.1/dist/leaflet.css fetched in 1.13 seconds

Checking link https://entrevalles.onrender.com/static/css/main.1809660d.css
HEAD https://entrevalles.onrender.com/static/css/main.1809660d.css fetched in 1.00 seconds
```

Results

Links

Valid links!

Anchors

Found 1 anchor.

Valid anchors!

Checked 1 document in 12.71 seconds.

Figura 14. Links válidos por W3C

6.4 Validación de la Resolución.

Se llevaron a cabo pruebas en diferentes resoluciones de pantalla para asegurar que la interfaz es responsiva y usable en distintos dispositivos, desde móviles hasta monitores de escritorio. Se utilizaron herramientas de desarrollo en navegadores web (DevTools) y servicios en línea como BrowserStack para simular diferentes dispositivos y resoluciones. La interfaz se ajustó adecuadamente en resoluciones comunes como 1920x1080, 1366x768, y 375x667 (iPhone X), asegurando una experiencia de usuario consistente y agradable.

Estas comprobaciones específicas se realizaron dentro del apartado [Interfaces Hardware](#), como se detalla en las figuras 3, 4 y 5 del documento.

6.5 Validación de Navegadores.

Se realizaron pruebas en múltiples navegadores web (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge) para asegurar la compatibilidad y el correcto funcionamiento en todos ellos. Las pruebas incluyeron la verificación del rendimiento, la correcta renderización de los elementos de la página. Las pruebas demostraron que la plataforma es completamente funcional en todas las versiones modernas de estos navegadores.

Se muestran las comprobaciones en la Figura 6 y 7 de este documento en el apartado [IH1. Compatibilidad con Dispositivos Habituales](#)

7. CONCLUSIÓN

7.1 Valoración Personal del Trabajo Realizado (análisis DAFO + análisis CAME).

Análisis DAFO

El análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) es una herramienta clave para evaluar el estado actual del proyecto y planificar estrategias futuras. A continuación, se presenta el análisis para la plataforma "Entre Valles y Montañas":

Debilidades

- Dependencia de una sola persona: En la fase inicial, el proyecto depende en gran medida de un único desarrollador, lo que puede limitar la capacidad de abordar múltiples tareas simultáneamente.
- Falta de experiencia previa en algunos aspectos: Aunque se posee un buen conocimiento técnico, hay áreas del proyecto, como el marketing digital y la gestión empresarial, donde la experiencia es limitada.
- Recursos limitados: La financiación inicial es pequeña, lo que puede restringir el alcance y la rapidez del desarrollo y promoción del proyecto.

Amenazas

- Competencia: Existen varias plataformas establecidas en el mercado que ofrecen servicios similares, lo que puede dificultar la captación de usuarios y socios comerciales.
- Cambios en las preferencias del mercado: Las tendencias en el turismo pueden cambiar rápidamente, afectando la relevancia de la plataforma.
- Problemas técnicos: Cualquier falla en la seguridad o en la funcionalidad de la plataforma podría afectar la confianza de los usuarios y la reputación del proyecto.

Fortalezas

- Visión clara del proyecto: Se tiene una visión bien definida y una clara comprensión de los objetivos a alcanzar.
- Competencia técnica: El uso de tecnologías modernas y eficientes, como React.js y SpringBoot, garantiza un producto robusto y escalable.
- Enfoque en la experiencia del usuario: La plataforma está diseñada para ser intuitiva y fácil de usar, mejorando la satisfacción del cliente.

Oportunidades

- Crecimiento del turismo en Asturias: El aumento constante del turismo en la región presenta una gran oportunidad para atraer usuarios a la plataforma.

- Digitalización del sector turístico: La tendencia hacia la digitalización en el turismo favorece la adopción de soluciones como la propuesta por la plataforma.
- Colaboraciones con negocios locales: Existe un gran potencial para establecer alianzas con negocios locales, ofreciendo un valor añadido tanto a los usuarios como a los socios comerciales.

Análisis CAME

El análisis CAME (Corregir, Afrontar, Mantener y Explotar) se utiliza para desarrollar estrategias basadas en el análisis DAFO. A continuación, se presentan las estrategias derivadas del análisis:

Corregir (Debilidades)

- Aumentar el equipo de trabajo: Contratar personal adicional para diversificar las competencias y reducir la dependencia de una sola persona.
- Formación continua: Invertir en la formación en áreas críticas como el marketing digital y la gestión empresarial para fortalecer las habilidades del equipo.
- Buscar financiamiento adicional: Explorar opciones de financiamiento, como inversionistas externos o subvenciones, para ampliar los recursos disponibles.

Afrontar (Amenazas)

- Análisis de la competencia: Realizar un seguimiento constante de las actividades de la competencia para identificar oportunidades de diferenciación.
- Adaptabilidad: Mantenerse al tanto de las tendencias del mercado y estar preparado para adaptar la plataforma según las preferencias cambiantes de los turistas.
- Fortalecer la seguridad: Implementar y mantener prácticas de seguridad robustas para proteger los datos de los usuarios y la integridad de la plataforma.

Mantener (Fortalezas)

- Desarrollo continuo: Continuar mejorando la plataforma con nuevas funcionalidades y actualizaciones basadas en las necesidades de los usuarios.
- Calidad técnica: Mantener el uso de tecnologías modernas y eficientes, asegurando un producto sólido y fiable.
- Enfoque en el usuario: Seguir centrando el desarrollo en mejorar la experiencia del usuario y la usabilidad de la plataforma.

Explotar (Oportunidades)

- Marketing enfocado: Aprovechar el crecimiento del turismo en Asturias mediante campañas de marketing dirigidas específicamente a turistas potenciales.
- Alianzas estratégicas: Establecer colaboraciones con negocios locales para ofrecer paquetes exclusivos y promociones a los usuarios de la plataforma.

- Innovación: Introducir características innovadoras que aprovechen la digitalización del sector turístico, como experiencias de realidad aumentada o recomendaciones personalizadas.

7.2 Posibles Ampliaciones.

Implementación de una aplicación móvil:

Desarrollo de una app para iOS y Android que permita a los usuarios acceder a la plataforma desde sus dispositivos móviles, mejorando la accesibilidad y conveniencia.

Integración con sistemas de transporte:

Incorporación de información sobre horarios y rutas de transporte público y privado para facilitar la planificación de rutas turísticas.

Expansión de la plataforma:

Ampliación de la plataforma para incluir información y servicios turísticos de otras regiones, creando una red de destinos que ofrezca una experiencia turística completa.

8. Referencias

A continuación, se presentan las páginas web visitadas donde se recopiló toda la información que ayudó en el desarrollo del código de la aplicación web y en la elaboración de la documentación:

[Sitio web oficial de React](#) - Documentación oficial para el desarrollo con React.

[Documentación de React Router](#) - Guías y ejemplos para la navegación con React.

[W3Schools](#) - Recursos para HTML, CSS, JavaScript y otras tecnologías web.

[Stack Overflow](#) - Comunidad en línea de desarrolladores donde se encontraron soluciones a problemas específicos.

[MDN Web Docs](#) - Recursos tecnologías web como HTML, CSS y JavaScript.

[W3C Markup Validation Service](#): Herramienta utilizada para validar CSS y HTML.

[W3C Link Checker](#): Herramienta gratuita para verificar los enlaces en el sitio web.

[BrowserStack](#): Servicio en línea para simular diferentes dispositivos y resoluciones.

[Mantine](#): Librería de componentes visuales del Frontend

[Stripe](#): Sistema de pagos para hacer las reservas en la aplicación

[Maven](#): Gestor de dependencias backend del proyecto

[Lighthouse](#): Herramienta automatizada de código abierto para verificar el rendimiento, la accesibilidad, las mejores prácticas y el SEO de la aplicación.