Universidad Zaragoza
1542

Trabajo Fin de Grado

Prototipo de comparador de alojamientos Erasmus con software libre

Autor

D. Javier Erlés García

Directora

Dra. Dª Piedad Garrido Picazo

Universidad de Zaragoza

2025

Tabla de Contenido

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas aquellas personas que han contribuido de manera directa o indirecta en la realización de mi Trabajo de Fin de Grado (TFG).

Agradezco en primer lugar a mi tutora Piedad, por su orientación, su apoyo y su comprensión durante todo el proceso y los cursos anteriores.

También quiero agradecer a todos los docentes de la Escuela Universitaria Politécnica de Teruel (EUPT) por su labor brindándome una educación de calidad que me ha permitido desarrollar todos los conocimientos que he adquirido a lo largo del grado.

Además, quiero expresar mi más profundo agradecimiento a los amigos que he tenido la suerte de conocer durante estos años de estudio. Sus palabras de aliento y su constante ánimo han sido fundamentales para superar los desafíos que se han presentado durante este camino.

Por último, quiero dedicar un agradecimiento especial a mi familia, cuyo apoyo incondicional ha sido mi mayor fortaleza. Gracias por estar siempre a mi lado, en los momentos de alegría y en los de dificultad, brindándome el ánimo necesario para seguir adelante.

A todas estas personas, muchísimas gracias por lo que me habéis aportado no sólo durante la elaboración de este TFG sino también a lo largo de todo mi recorrido universitario.

¡Muchísimas gracias a todos!

Resumen y Palabras Clave

El objetivo del presente TFG es la programación de un prototipo de comparador de alojamientos Erasmus basado exclusivamente en software libre.

En este documento se describen todos los pasos llevados a cabo para el desarrollo del prototipo, desde la fase de análisis y diseño donde se determinan todos los requisitos necesarios hasta los diferentes casos de uso y la implementación en un entorno local.

La aplicación ha sido desarrollada con Node.js, esto permitió mantener un entorno unificado basado en Javascript. Para implementar el front-end se ha utilizado Nuxt, un potente framework basado en Vue.js, utilizando técnicas modernas de diseño web como CSS Grid y Flexbox, que permiten estructurar el contenido de forma flexible y escalable, garantizando así un mejor rendimiento y posicionamiento SEO (Search Engine Optimization).

Para gestionar los contenidos dinámicos, se ha implementado una aplicación enmarcada en el back-end del proyecto que utiliza Express.js, un framework minimalista y flexible para Node.js. También se ha utilizado un sistema de base de datos relacional para el almacenamiento y gestión de datos que ha permitido estructurar de forma eficiente la información de alojamientos, reservas y demás entidades.

En resumen, este trabajo ha permitido desarrollar un prototipo funcional de comparador de alojamientos Erasmus, empleando herramientas de software libre y tecnologías modernas. El resultado es una aplicación flexible y escalable que en un futuro podría ampliarse y desplegarse en un entorno real.

Palabras Clave: Comparador de alojamientos, Erasmus, Software libre, Front-end, Back-end.

1. Introducción y Objetivos

Encontrar alojamiento es uno de los mayores retos a los que se enfrentan los estudiantes de Erasmus cuando se van a mudar a otro país para realizar su programa de intercambio. Existen numerosas aplicaciones para buscar alojamiento, pero la mayoría de ellas no están orientadas a estudiantes de intercambio.

La idea de este TFG surge de una experiencia personal cuando tuve que buscar alojamiento para mi programa Erasmus en Irlanda. Durante este proceso tuve numerosos problemas a la hora de encontrar un alojamiento fiable que cumpliera con mis necesidades y presupuesto. La información era bastante limitada y las opciones no se ajustaban a lo que estaba buscando. Muchos de los anuncios estaban desactualizados o eran fraudulentos, lo que generaba una inseguridad a la hora de escoger el alojamiento deseado y afectaba a la experiencia de los estudiantes en su destino Erasmus.

Después, he tenido la suerte de trabajar para una empresa que desarrolla una plataforma similar, orientada a estudiantes de intercambio, por lo que he aprendido mucho sobre como funcionan estas aplicaciones y los retos a los que se enfrentan tanto los estudiantes como los proveedores de alojamiento.

El objetivo principal que se aborda en el proyecto es el de desarrollar un prototipo práctico, sencillo e intuitivo que facilite la búsqueda, filtrado y reserva de alojamientos para estudiantes que formen parte del programa Erasmus. Lo que les proporcionará información confiable y ahorro de tiempo durante este proceso.

El proyecto también tiene como objetivo consolidar conocimientos, obtenidos a lo largo del Grado en Ingeniería Informática (GII) en el campo del desarrollo web, explorando nuevas tecnologías que tengan una gran repercusión en el mercado, con el valor y dificultad añadida de centrarse sólo en el uso de software libre.

A lo largo del documento se describen los pasos llevados a cabo para realización de este proyecto, desde las primeras fases de análisis y diseño, hasta las últimas fases de desarrollo e implementación y pruebas para crear una solución que satisfaga las necesidades de los estudiantes que están buscando un alojamiento en una ciudad y país desconocidos.

2. Estado del Arte

Este apartado recoge un análisis de las diferentes plataformas web orientadas a la búsqueda de alojamiento para estudiantes. El objetivo es identificar características comunes, buenas prácticas, problemas recurrentes de accesibilidad y usabilidad, así como elementos diferenciadores que puedan servir de referencia para el diseño del prototipo desarrollado en este TFG (Trabajo de Fin de Grado).

2.1 Análisis de otros sitios web para buscar alojamiento

No sólo es importante estudiar los diferentes marcos de trabajo y tecnologías existentes en el mercado, sino que también es importante investigar y analizar otros sitios webs similares para observar las diferentes herramientas utilizadas.

Para ello, se han analizado otros sitios webs orientados a la búsqueda de alojamiento, especialmente los que están diseñados para estudiantes en programas de movilidad teniendo en cuenta aspectos como la usabilidad, diseño, filtros, rendimiento y accesibilidad.

La búsqueda de los sitios web no ha supuesto mucha complicación, ya que la mayoría de ellos cuentan con una muy buena optimización para motores de búsqueda y aparecen en las primeras posiciones de los buscadores más famosos, en concreto se van a analizar tres iniciativas: ErasmusPlay, Erasmusu y Uniplaces.

2.1.1 ErasmusPlay

El buscador y comparador de alojamientos EramusPlay está dedicado a estudiantes Erasmus buscando alojamiento en ciudades desconocidas. Permite comparar toda la oferta de alojamientos disponibles, de forma rápida y sencilla. Además, todos los alojamientos que ofrecen en su web provienen de plataformas verificadas.

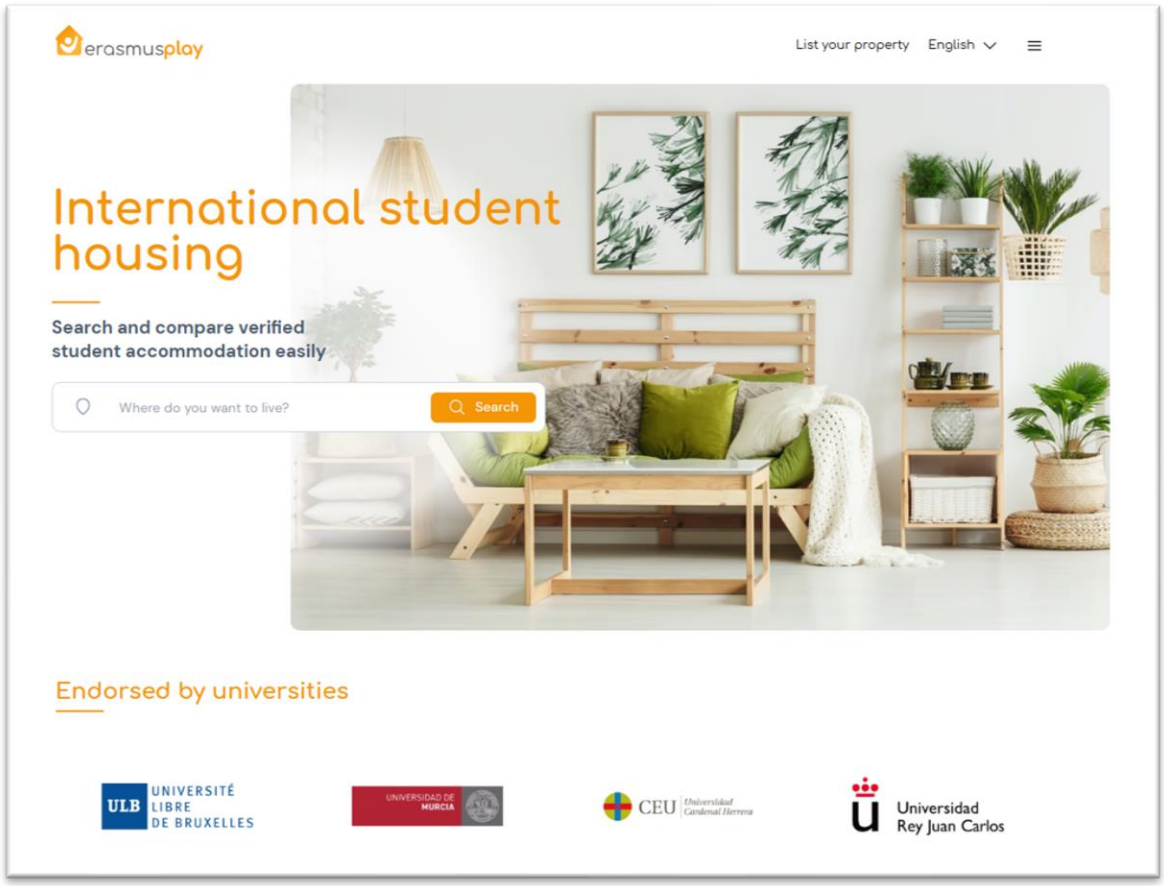


Ilustración 1 - Sitio Web ErasmusPlay

El sitio web utiliza el framework Nuxt, uno de los marcos de trabajo vistos con anterioridad, también utiliza Open Graph, un protocolo que utiliza etiquetas sociales para mejorar el posicionamiento en redes sociales. Otra tecnología utilizada es Google Tag Manager para rastrear eventos y trackear a los usuarios. Además, utiliza servicios de AWS (Amazon Web Services) como Cloudfront, una librería encargada de gestionar estados en aplicaciones de Vue llamada Pinia y un framework de CSS (Cascade Style Sheet) muy utilizado en la actualidad llamado Tailwind CSS.

El contenido de la web es principalmente dinámico, ya que los usuarios pueden interactuar con el comparador de alojamientos y dependiendo de los filtros seleccionados, este ofrecerá una respuesta u otra. También cuenta con contenido estático, como la sección de términos y condiciones, una sección de preguntas e información relevante y landings estáticos con información relevante de diferentes ciudades y países europeos.

El sitio web cuenta con un menú y un selector de idioma, que admite alemán, español, italiano, francés, polaco, portugués, turco y chino. En la página principal se puede encontrar un buscador para comenzar a comparar alojamientos, una breve presentación y varios links a búsquedas de las ciudades europeas más importantes. La sección de búsqueda cuenta con filtros personalizados y un mapa interactivo. Todos los alojamientos cuentan con una sección de detalles dónde se puede apreciar información relevante de estos y realizar una solicitud de reserva. Una vez logueado se pueden utilizar las secciones de alertas, reservas y favoritos, para tener un registro de las acciones realizadas. Además, existen otras secciones interesantes como

una página para gestionar la información personal, una sección donde se permite listar una propiedad, una página con información sobre colaboradores, etc.

Para analizar la **accesibilidad** del sitio se ha utilizado una herramienta que se puede encontrar como una extensión de Google Chrome desarrollada por IBM llamada IBM Accesibility Checker [9], esta herramienta se ha configurado para seguir las pautas WCAG 2.1 con un nivel de análisis AA y unos requisitos adicionales de IBM. Los resultados obtenidos han sido 352 errores encontrados que tienen que ser corregidos y 172 advertencias que no tienen por qué ser errores. Por otra parte, no se han encontrado errores en el 89% de los elementos analizados.

Los principales errores encontrados en el análisis tienen que ver con identificadores duplicados en botones, espacio insuficiente entre objetivos interactivos, falta de etiquetas en formularios, contenido fuera de landmarks y un contraste insuficiente.

Para analizar la **usabilidad** de la página, se han tenido en cuenta las **reglas heurísticas de Jakob Nielsen** [10], en general, se cumplen la mayoría de las reglas desarrolladas por Nielsen.

La página informa a los usuarios sobre el progreso de sus acciones, por lo que proporciona una visibilidad del estado del sistema constantemente. Utiliza un lenguaje familiar dando libertad a los usuarios para que modifiquen los filtros de búsqueda, aunque si no ha realizado el inicio de sesión hay secciones ocultas y funcionalidades que no se pueden realizar.

La interfaz es intuitiva, consistente y cuenta con un diseño limpio, aunque a veces se pueden encontrar errores en las traducciones, pero no son muy frecuentes. Además, la página web cuenta con una página de ayuda donde aparecen preguntas frecuentes e información útil para los estudiantes.

2.1.2 Erasmusu

Erasmusu es un comparador de alojamientos similar a ErasmusPlay, aunque ofrece otras funcionalidades. Permite comparar alojamientos de diferentes ciudades, buscar compañeros de piso, buscar información sobre ciudades, publicar ofertas de trabajo, comparar grados, clases online y entradas en blogs y foros.

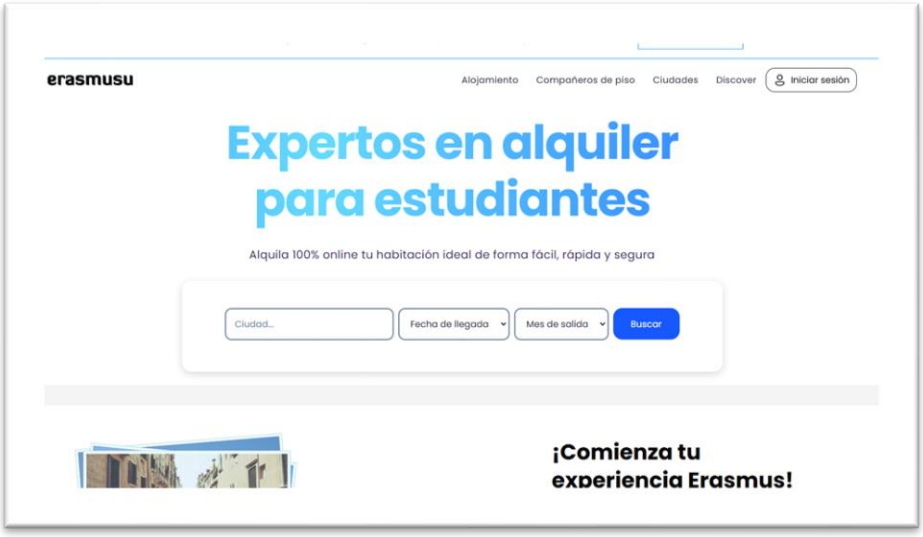


Ilustración 2 - Sitio Web Erasmusu

Para desarrollar la aplicación se ha utilizado el framework de Next.js basado en React del que se habló anteriormente. Al igual que Erasmusplay, se ha utilizado Google Tag Manager para rastrear eventos y Tailwind CSS para diseñar el estilo. Entre otras tecnologías utilizadas se encuentra Cloudflare, que ofrece servicios de red de distribución de contenido, mitigación de DDoS, seguridad de Internet y servicios de servidor de nombres de dominio distribuido. También se han utilizado librerías de JavaScript como JQuery, LazySizes y Modernizr.

La página cuenta con un menú superior que permite navegar entre las cuatro principales secciones de la aplicación (alojamiento, compartir piso, ciudades y discover), acceder a las funciones de perfil y seleccionar los diferentes idiomas que soporta (español, inglés, francés, italiano, portugués, polaco, turco, alemán y holandés). En la página principal aparece una barra de búsqueda con capacidad para añadir filtros de llegada y salida que redirige a la búsqueda. Justo debajo aparece otra barra de búsqueda y un desglose de los alojamientos, masters, compañeros de piso, experiencias, lugares, entradas de blogs, grados, trabajos, entradas en foros y clases particulares ofreciendo muchísima información relevante.

Nuevamente, para analizar la **accesibilidad** del sitio web se ha utilizado la herramienta IBM Accesibility Checker. Los resultados reportados nos muestran 285 errores que necesitan ser corregidos, 48 advertencias a revisar y el porcentaje de elementos sin necesidad de revisión es de un 73%.

Los principales errores encontrados tienen que ver con problemas en formularios donde faltan etiquetas descriptivas, muchos encabezados y párrafos no están dentro de landmarks (como <main> o <section>), existen un bajo contraste de texto lo que puede afectar a usuarios con baja visión, hay presencia de enlaces sin texto descriptivo e imágenes sin texto alternativo y hay algún ID duplicado que puede provocar errores de funcionamiento.

Analizando la **usabilidad** del sitio, se han encontrado varios enlaces rotos que pueden afectar a la experiencia del usuario al loguearse, en los buscadores y en enlaces de sugerencia.

Relacionando estos problemas con **las heurísticas de Nielsen** [10] se puede observar que no se cumplen del todo, los errores encontrados no están explicados correctamente ni se comunica

como proceder. La interfaz mantiene un diseño consistente, aunque los cambios de idioma inesperados debido a errores rompen esta consistencia y el estilo de los diferentes filtros confunde. Los menús, enlaces y secciones son identificables fácilmente, aunque posee un diseño algo saturado de información, sobre todo en la página principal. En otras palabras, la plataforma ofrece bastantes funcionalidades, sin embargo, los errores encontrados generan una experiencia frustrante para los usuarios que habría que mejorar.

2.1.3 Uniplaces

Uniplaces es un comparador de alojamientos bastante similar a los dos casos estudiados anteriormente, aunque su estrategia de marketing está más orientada a ofrecer estancias temporales en ciudades universitarias y ofertas personalizadas, pero no es excluyente y admite otros públicos.

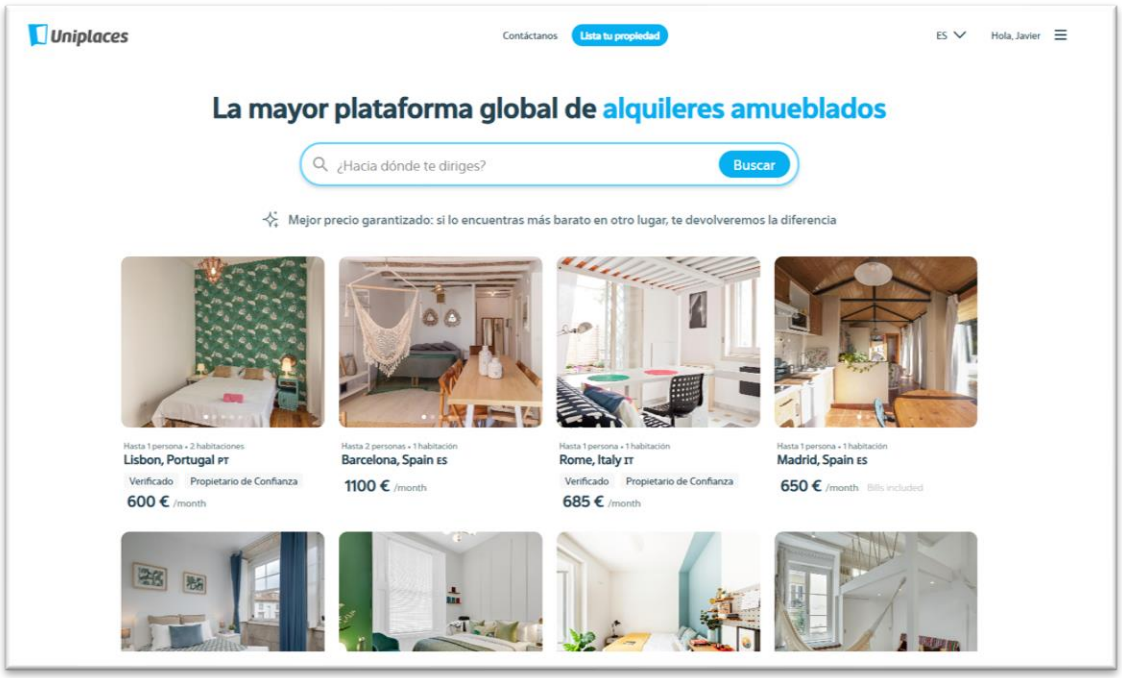


Ilustración 3 - Sitio web Uniplaces

Para desarrollar el sitio web, se han utilizado tecnologías similares a los casos anteriores. El framework principal que se ha utilizado es Next.js con la ayuda del gestor de contenidos Prismic. También se ha utilizado Google Tag Manager y servicios de AWS para la infraestructura y CDN (Content Delivery Network). Entre las librerías de JavaScript utilizadas se pueden encontrar JQuery, Swiper, Moment.js y el framework orientado al estilo 'styled-components' en lugar de Tailwind CSS como ocurre en los casos anteriores.

Los contenidos del sitio web son muy similares a los casos ya estudiados anteriormente, aunque se asemejan más al sitio de ErasmusPlay. Los contenidos dinámicos aparecen en los resultados de búsqueda principalmente. En cambio, los contenidos estáticos se pueden encontrar en las páginas informativas que ofrece, los artículos y blogs sobre ciudades y consejos y los enlaces del pie de página.

El diseño y estructura de la aplicación es bastante similar a la solución ofrecida por ErasmusPlay, en el header o cabecera encontramos el menú que permite navegar entre las diferentes secciones (inicio, lista tu propiedad, favoritos, estancias, ayuda, cómo funciona, etc). Como en ambos casos anteriores, admite varios idiomas: alemán, inglés, español, francés, italiano, holandés, polaco, portugués y chino.

De forma similar que en los casos anteriores el diseño de la página de búsqueda cuenta con una muestra de alojamientos y un mapa interactivo donde aparecen los alojamientos de esa página. Las páginas de detalles ofrecen muchísima información y a diferencia de los casos anteriores, permite la reserva de un establecimiento sin necesidad de estar registrado en la página, lo que mejora mucho la experiencia del usuario.

La **accesibilidad** del sitio se ha analizado de forma análoga a los casos anteriores. La herramienta reporta que existen 222 errores que necesitan ser corregidos, 164 que sería conveniente revisar y el porcentaje de los elementos sin errores ni revisiones necesarias es de un 72%.

Los principales errores detectados están relacionados con problemas en etiquetas y atributos que no están bien asociados, un contraste insuficiente que no cumple con los requisitos de WCAG AA, elementos svg que no proporcionan un nombre accesible, enlaces demasiado pequeños o juntos y uso incorrecto de landmarks y jerarquías de encabezado.

Como en casos anteriores, la **usabilidad** ha sido analizada siguiendo las **heurísticas de Jakob Nielsen** y en este caso se cumplen parcialmente. Por ejemplo en lo que respecta a la consistencia y estándares, el diseño de la interfaz es coherente, aunque se han encontrado errores de idioma que rompen la consistencia del sitio (cambio de idioma a inglés cuando se inicia el proceso de reserva). En los formularios se pueden encontrar validaciones básicas que avisan de la falta de datos clave o si el rango de fechas seleccionado no está disponible en el alojamiento, etc.

2.1.4 Conclusiones

A continuación, se muestra una tabla comparativa que muestra los resultados analizados en los apartados anteriores:

Sitio Web	Tecnologías	Contenido	Accesibilidad	Problemas Usabilidad	Idiomas
Erasmusplay	Nuxt.js OpenGraph Google Tag Manager AWS Cloudfront Pinia Tailwind CSS	Dinámico y estático	352 errores 172 avisos 89% sin error	Consistencia Control y libertad del usuario	Alemán Español Italiano Francés Polaco Portugués Turco Chino

Erasmusu	Next.js Google Tag Manager CloudFlare JQuery LazySizes Modernizr Tailwind CSS	Dinámico y estático	285 errores 48 avisos 73% sin error	Consistencia Prevención de errores Diseño Visibilidad de estado	Español Inglés Francés Italiano Portugués Polaco Turco Alemán Holandés
Uniplaces	Next.js Google Tag Manager AWS CloudFront JQuery Swiper Moment.js	Dinámico y estático	222 errores 164 avisos 72% sin error	Consistencia Prevención de errores	Alemán Inglés Español Francés Italiano Holandés Portugués Chino

Tabla 1 – Comparativa de los sitios web analizados [elaboración propia]

Actualmente, React es la biblioteca de JavaScript más utilizada en el mundo de las tecnologías web. En la **Tabla 2** se puede observar que en el último año ha alcanzado una cuota de uso del 39,5 % situándolo muy por encima de las demás tecnologías. Tiene sentido que dos de las tres páginas web analizadas estén desarrolladas en Next.js, ya que éste es un framework basado en React para crear aplicaciones modernas.

En comparación, Vue parece tener menos popularidad, con una cuota de uso del 15,4 % en el año pasado (**Tabla 2**).

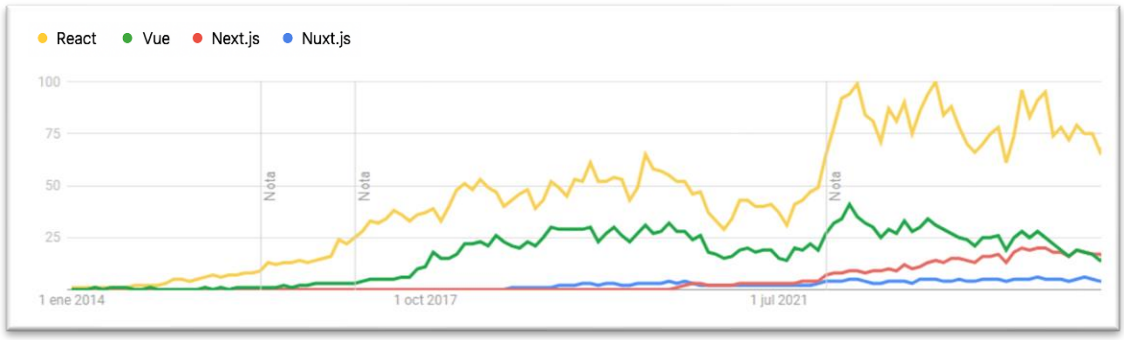


Ilustración 4 - Comparativa a lo largo del tiempo React vs Vue.js vs Next.js vs Nuxt.js [Google Trends][11]

En los sitios web analizados, se puede observar un soporte para varios idiomas en todas las soluciones. Los más comunes son inglés, español, francés e italiano, aunque todos los sitios han incluido idiomas como el chino o el turco, en el caso de ErasmusPlay ambos.

El contenido de todos los sitios web es tanto dinámico como estático, permitiendo mostrar resultados de búsquedas que cambian dependiendo de los filtros que aplican los usuarios y secciones informativas que mantienen su estado.

En cuanto a la **accesibilidad**, los errores son bastante numerosos. ErasmusPlay muestra la mayor tasa de cumplimiento llegando a un 89% de elementos sin errores, mientras que Uniplaces y Erasmusu tienen resultados bastante parecidos, 72% y 73% respectivamente. Estos datos indican que existen posibles mejoras a realizar en todos los sitios web analizados.

Por último, los problemas de **usabilidad** varían según cada sitio. La consistencia es una problemática recurrente en todos los casos, especialmente debido a errores de traducción o cambios inesperados de idioma. Además, Erasmusu y Uniplaces presentan problemas adicionales relacionados con la prevención de errores, como la falta de mensajes claros para guiar al usuario en situaciones problemáticas. En cambio, ErasmusPlay destaca por ofrecer una experiencia más consistente, aunque sigue enfrentando problemas en cuanto a control y libertad del usuario.

A partir del análisis realizado, se ha decidido desarrollar el prototipo con Nuxt 3 como framework principal, debido a su simplicidad, rendimiento y estructura modular, lo que lo convierte en una opción adecuada para proyectos con recursos limitados. A pesar de que Nuxt.js no es una de las tecnologías más utilizadas ofrece una alternativa moderna que permite implementar soluciones rápidas y accesibles.

En cuanto a las mejoras planteadas con respecto a las soluciones existentes, el proyecto se centrará en optimizar la accesibilidad desde el inicio del desarrollo para evitar los errores detectados en las herramientas analizadas, en simplificar la experiencia de usuario con un diseño limpio, navegación clara y filtros adecuados y ofrecer una experiencia personalizada para estudiantes Erasmus.

Estas decisiones se alinean con el objetivo principal del proyecto de diseñar un comparador web de alojamientos centrado en estudiantes Erasmus con una interfaz intuitiva, accesible y funcional con tecnologías libres.

3. Análisis y Diseño

3.1 Introducción

En este apartado se va a describir la estructura y las características del prototipo desarrollado, así como sus requisito funcionales y no funcionales. Este apartado sigue las prácticas recomendadas del estándar IEEE 830-1998 [12] para ofrecer una especificación clara y estructurada.

El objetivo principal, ya comentado anteriormente, es facilitar la búsqueda de alojamientos para estudiantes mediante una plataforma web que sea intuitiva, rápida y accesible sin la necesidad de realizar un registro previo. Mediante este análisis y diseño, se establecen las bases para luego asegurar un desarrollo que cumpla las necesidades del proyecto.

3.1.1 Propósito

El propósito de esta sección es definir con precisión los requisitos y diseño del sistema, para proporcionar una guía estructurada para el desarrollo de éste. Se detallan tanto las funcionalidades del frontend como las del backend, incluyendo los diferentes casos de uso existentes y estableciendo los criterios necesarios para un correcto funcionamiento.

3.1.2 Alcance

El prototipo a desarrollar consiste en una plataforma web que permita a los usuarios buscar alojamientos en diferentes ciudades sin la necesidad de un registro. Esto se realizará a través de un motor de búsqueda y unos filtros personalizables.

La plataforma proporcionará las siguientes funcionalidades principales:

- Búsqueda de alojamientos por ciudad.
- Filtrado avanzado para ajustar los resultados.
- Visualización de detalles para cada alojamiento (imágenes, descripción, ubicación…)
- Formulario de solicitudes (reserva, información y video-tour).

Cabe destacar que el sistema a desarrollar no gestionará pagos, en su lugar se redirigirá a los usuarios a las plataformas de los proveedores de alojamiento para los casos en los que estén disponibles. Tampoco contará con un sistema de registro de usuarios ni permitirá la publicación de alojamientos de forma independiente.

3.1.3 Definiciones, Acrónimos y Abreviaturas

- Usuario: Persona que accede a la plataforma para buscar alojamiento.

- Frontend: Interfaz web con la que interactúan los usuarios para realizar las búsquedas y visualizar alojamientos.
- Backend: Servidor que procesa las solicitudes realizadas por el frontend. Se encarga de gestionar la comunicación con la base de datos.
- RF (Requisito Funcional): Característica o función específica
- RNF (Requisito No Funcional):

3.2 Descripción General del Sistema

Este apartado proporciona una visión general del sistema en el que se detallará el contexto, las funcionalidades principales, el tipo de usuarios, las restricciones y las dependencias de este.

3.2.1 Perspectiva del Producto

El sistema desarrollado es una plataforma web encargada de facilitar la búsqueda de alojamientos para estudiantes. Consiste en una solución independiente que no requiere registro y permitirá a los usuarios navegar entre diferentes opciones de alojamiento utilizando un buscador y unos filtros avanzados.

La arquitectura del sistema sigue un enfoque Cliente-Servidor (C/S), donde:

- Frontend: Es una aplicación web accesible desde cualquier dispositivo con conexión a internet, está encargada de la interfaz de usuario y de la interacción con el backend.
- Backend: Aplicación de servidor que procesa las solicitudes realizadas por el frontend, gestiona la base de datos y maneja la lógica de negocio para ofrecer resultados optimizados.

El sistema no gestiona pagos ni reservas directas, sino que enlaza con plataformas externas para completar el proceso.

3.2.2 Funcionalidades Principales

El sistema cuenta con las siguientes funcionalidades clave:

- Búsqueda de alojamientos por nombre de ciudad

- Filtrado de resultados, permitiendo a los usuario ajustar la búsqueda según:
    - Tipo de alojamiento (habitación privada, compartida, estudio, etc.).
    - Rango de precios.
    - Disponibilidad de fechas.

- Visualización detallada de cada alojamiento, incluyendo:
    - Galería de imágenes.

    - Ubicación en mapa interactivo.
    - Descripción de alojamiento.
    - Precio y condiciones.

- Solicitud de reserva: Se proporcionará un formulario para que los usuarios puedan contactar con el proveedor de alojamiento.

- Información adicional, incluyendo:
    - Sección de preguntas frecuentes.
    - Formulario de contacto para soporte.
    - Términos y condiciones de uso.

3.2.3 Características del Usuario

El sistema está diseñado y orientado para usuarios sin conocimientos técnicos, ofreciendo una interfaz intuitiva y fácil de utilizar. Los principales usuarios serán:

- Estudiantes universitarios en busca de alojamiento temporal o de larga estancia.
- Personas que planean realizar intercambios académicos o prácticas en otras ciudades y necesitan encontrar hospedaje de forma sencilla.
- Usuarios en general que buscan una plataforma centralizada para comparar opciones de alojamiento sin necesidad de un registro previo.

Dado que el sistema no requiere una autenticación, todos los usuarios accederán a la plataforma de forma anónima y sin historial de navegación almacenado.

3.2.4 Restricciones

Para garantizar un buen rendimiento y una experiencia fluida, el sistema contará con las siguientes restricciones:

- Acceso sin registro: No se pedirá a los usuarios crear una cuenta para utilizar la plataforma.
- Tiempo de respuesta: El backend deberá procesar y devolver los resultados de búsqueda en menos de 2 segundos.
- Interfaz responsiva: La plataforma se adaptará a la pantalla de diferentes dispositivos, tanto móviles como tablets y escritorios.

3.2.5 Suposiciones y Dependencias

El correcto funcionamiento del sistema depende de los siguientes factores:

- Conexión a internet: Se asume que los usuarios contarán con acceso a internet para utilizar la plataforma.
- Fuentes externas de datos: La información de los alojamientos proviene de servicios de terceros, por lo que su disponibilidad y actualización dependen de dichas plataformas.
- Integración con mapas: Se utilizarán servicios externos para la visualización de mapas y ubicaciones.
- Sin autenticación de usuarios: No habrá perfiles de usuario ni almacenamiento de preferencias personales.

3.3 Requisitos Específicos

En esta parte de la fase de análisis y diseño se incluyen todos los requisitos tanto del frontend como del backend del sitio web, atendiendo las necesidades del proyecto y llevando a cabo una primera aproximación a lo que será el diseño posterior.

3.3.1 Requisitos Funcionales

Requisitos funcionales frontend

- **RF01.** Existirá únicamente un rol de usuario (no registrado) que tendrá acceso al sitio web por defecto. No será necesario registro para utilizar las funcionalidades básicas del prototipo.

- **RF02.** El sitio web contará con una página de inicio en la que aparecerá una barra de búsqueda, una pequeña introducción con información general sobre como utilizar la página y una pequeña sección con información sobre el proyecto (about us).

- **RF03.** Existirá una página con información legal sobre los términos y condiciones de uso.

- **RF04.** El prototipo incluirá también una página de contacto a disposición de los usuarios con un formulario que incluirá nombre, correo electrónico y el mensaje deseado.

- **RF05.** El sitio web contará con un header fijo que aparecerá en todas las páginas del prototipo con el logo y un selector de idioma. El logo permitirá a los usuarios volver a la página de inicio.

- **RF06.** En la parte inferior de todas las páginas aparecerá un footer con enlaces a las páginas de términos, FAQs (Frequently Asked Questions), contacto y el logo.

- **RF07.** La aplicación permitirá realizar búsquedas de alojamientos introduciendo la ciudad deseada en la barra de búsqueda.

- **RF08.** La barra de búsqueda mostrará sugerencias en tiempo real dependiendo de los caracteres introducidos en ella.

- **RF09.** La barra de búsqueda admitirá fechas de entrada y salida en la página de búsqueda.

- **RF10.** La página de búsqueda contará con una barra de filtros personalizables, un grid en el que aparecerán las tarjetas con los alojamientos y un elemento para paginar los resultados en la parte inferior de ésta.

- **RF11.** Los resultados de la búsqueda aparecerán en orden de relevancia (valor invisible generado al integrar los alojamientos) y cada página admitirá 30 alojamientos.

- **RF12.** Los filtros personalizables admitirán los diferentes tipos de alojamiento (habitación privada, habitación compartida, estudio, casa, residencia, coliving) y precio (máximo y mínimo).

- **RF13.** Los resultados de la búsqueda se actualizarán dinámicamente conforme a los filtros aplicados.

- **RF14.** Las tarjetas de los alojamientos dispondrán de una imagen principal del alojamiento, una descripción con el tipo de alojamiento, la dirección, la estancia mínima y la primera disponibilidad. Además, en la parte inferior contarán con dos botones:
    - Detalles: abrirá en una nueva pestaña la página de detalles del alojamiento.
    - Ver: abrirá en una nueva pestaña la página del proveedor del alojamiento.

- **RF15.** La página de detalles dispondrá de una galería donde visualizar las imágenes, una descripción, una lista con las comodidades, la disponibilidad del año actual y el siguiente, un mapa interactivo con la localización del alojamiento y un formulario para realizar solicitudes.

- **RF16.** El formulario dispondrá de un calendario en el que se seleccionarán las fechas de entrada y salida, tendrá una sección para aceptar los términos y condiciones, se introducirá el correo electrónico, un mensaje, una opción de cómo ha descubierto la página y el tipo de solicitud (reservar ahora, más información y solicitar video tour).

- **RF17.** La disponibilidad se mostrará en un calendario en el que aparecerán los meses disponibles en verde y los meses no disponibles en rojo.

Requisitos funcionales backend

- **RF01.** El backend debe permitir realizar búsquedas de alojamiento en función de la ciudad especificada.

- **RF02.** El sistema debe procesar las solicitudes de alojamiento basadas en los filtros aplicados en el frontend.

- **RF03.** Deberá ofrecer una lista de sugerencias dinámicas basadas en los caracteres introducidos por los usuarios en la barra de búsqueda.

- **RF04.** Las solicitudes de reserva se guardarán en la base de datos, registrando el id del alojamiento y los campos del formulario.

- **RF05.** El backend deberá procesar y reenviar por correo electrónico los mensajes enviados desde el formulario de contacto.

- **RF06.** Las consultas relacionadas con la búsqueda deberán devolver los resultados limitados para optimizar el rendimiento.

- **RF07**. Todas las operaciones exitosas devolverán un código HTTP 200, en caso de fallo interno procesando la solicitud se devolverá un código 500.

3.3.2 Requisitos no funcionales

Requisitos no funcionales frontend

- **RNF01.** La interfaz de usuario será sencilla e intuitiva, permitiendo a los usuarios buscar alojamientos, filtrar resultados y visualizar los detalles. Cada página deberá seguir una estructura lógica.

- **RNF02.** El prototipo deberá ser accesible desde cualquier dispositivo con acceso a Internet, tanto dispositivos móviles y tables como ordenadores.

- **RNF03.** El diseño será totalmente responsivo, para que se adapte a los diferentes tamaños de pantalla y resoluciones.

- **RNF04.** La web deberá soportar múltiples idiomas, principalmente castellano e inglés.

- **RNF05.** Cualquier error que pueda producirse deberá ser manejado, proporcionando mensajes claros y concisos.

- **RNF06.** Las solicitudes que se realicen en el prototipo no podrán superar un tiempo de respuesta de 2 segundos.

- **RNF07.** En toda la página se deberá garantizar que los elementos interactivos como el mapa, la galería y los formularios, funcionen de manera correcta.

Requisitos no funcionales backend

- **RNF01.** El backend deberá ser eficiente y responder con un máximo de 2 segundos por cada solicitud realizada desde el frontend.

- **RNF02.** El backend será sencillo de mantener, con un diseño modular. Cada funcionalidad estará implementada en módulos independientes.

- **RNF03.** Todas las solicitudes que se realicen entre frontend y backend seguirán el formato JSON (JavaScript Object Notation).

- **RNF04.** El backend será compatible con las versiones LTS (Long Term Support) de Node.js para asegurar estabilidad y garantizar soporte a largo plazo.

- **RNF05.** Se deberá manejar concurrencia básica para que el backend sea capaz de procesar al menos 50 solicitudes concurrentes en un entorno local sin pérdida de rendimiento.

- **RNF06.** La arquitectura deberá ser ligera para ejecutarse en entornos locales o servidores básicos sin mucha configuración.

- **RNF07.** El backend deberá mostrar logs de eventos importantes como solicitudes recibidas o errores internos.

- **RNF08.** Las consultas realizadas a la base de datos deberán estar optimizadas para garantizar un buen rendimiento en las búsquedas con o sin filtrado.

3.4 Casos de uso

El lenguaje de Modelado Unificado [16] (UML, por sus siglas en inglés Unified Modeling Language) es un estándar muy utilizado en el ámbito del desarrollo de software para representar de forma gráfica los distintos aspectos de un sistema.

Dentro de UML aparecen los diagramas de casos de uso, una herramienta fundamental que permite representar las principales funcionalidades de un sistema desde el punto de vista del usuario u otros sistemas externos. Estos diagramas muestran los actores que interactúan con el sistema y los diferentes casos de uso o funcionalidades que ofrece.

3.4.1 Casos de uso del frontend

En la **Ilustración 5** se representan los principales casos de uso que ofrece la interfaz de usuario. El usuario puede realizar búsquedas de alojamientos por ciudad, aplicar filtros, visualizar detalles, interactuar con el mapa, realizar solicitudes de reserva, compartir alojamientos y otras funcionalidades.

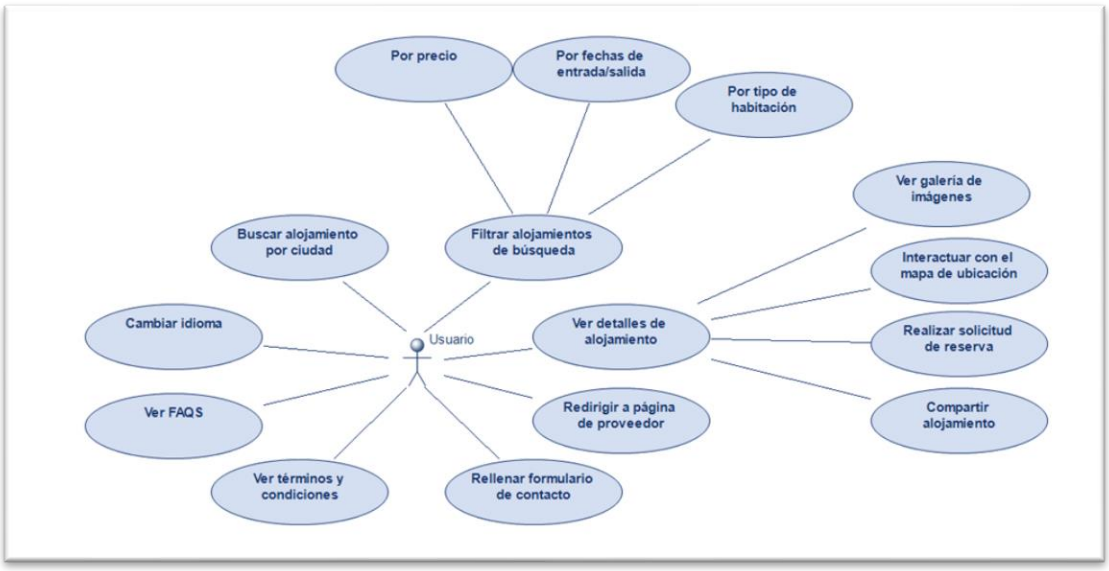


Ilustración 5 Casos de uso del frontend

3.4.2 Casos de uso del backend

En la **Ilustración 6** se representan los principales casos de uso del servidor. Este gestiona las operaciones internas del sistema, es decir, cómo el servidor es responsable de procesar las solicitudes del frontend, enviar listados de alojamientos, devolver los datos de cada alojamiento, gestionar las solicitudes de reserva y procesar los correos de contacto enviados desde la aplicación.

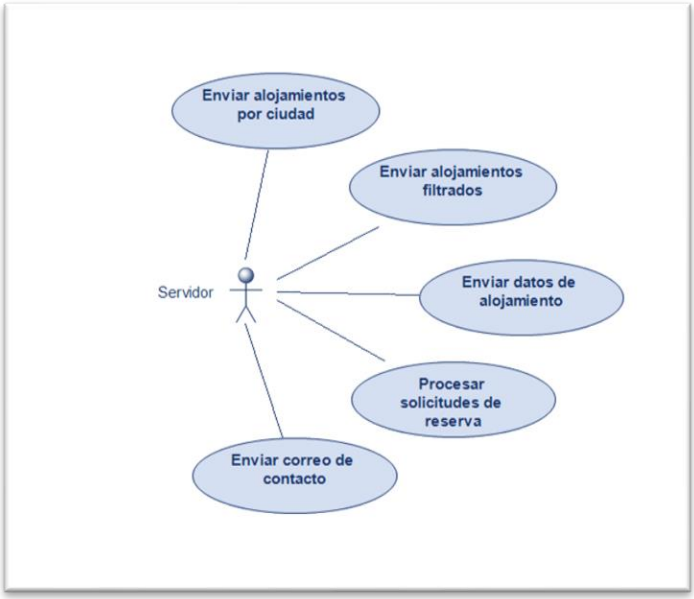


Ilustración 6 Casos de uso del backend

3.5 Primeros diseños

Antes de comenzar con la implementación del sistema, se ha desarrollado una serie de mockups utilizando Balsamiq para mostrar de manera general una aproximación de lo que se va a implementar a continuación, pero con un nivel de detallo bajo. Dichos mockups se pueden consultar en los Anexos 9.1 y 9.2.

4. Desarrollo

En este apartdo se describe el proceso de desarrollo técnico del proyecto, en el que se detallan las herramientas utilizadas, la arquitectura del sistema y las decisiones implementadas en cada una de sus capas, incluyendo las tecnologías estudiadas, el entorno de desarrollo, la implementación del frontend y backend, y el diseño de la base de datos.

En este contexto aparecen los “web framework”, un marco de software que ha sido diseñado para respaldar el desarrollo de aplicaciones web, abarcando servicios, recursos y APIs. Estos marcos proporcionan un estándar a la hora de crear e implementar aplicaciones web.

4.1 Frameworks para desarrollo web

Un “web framework” es un marco de software que tiene como objetivo automatizar la sobrecarga asociada con las actividades comunes que se realizan en un entorno de desarrollo web. Estos se componen de un conjunto de herramientas, bibliotecas y estructuras predefinidas que son la base para construir aplicaciones web.

Algunas de las principales características de los “web framework” son [1]:

- Establecen el flujo de control de un programa y permiten interactuar con él a través de eventos. Por ejemplo, algunos frameworks populares como Express.js permiten añadir “middleware” antes y después de las solicitudes HTTP (HyperText Transfer Protocol), para permitir al desarrollador gestionar autenticación, sesiones y redirecciones, entre otras cosas.

- Uso de sistemas de plantillas web que permite a los desarrolladores trabajar con plantillas para generar de forma automática páginas web personalizadas, simplificando la generación dinámica de contenido HTML (HyperText Metadata Language).

- Almacenamiento en caché para reducir el uso de ancho de banda, la carga del servidor y el “retraso” percibido. Una caché almacena copias de los documentos que pasan por ella para en un futuro evitar redundancia de solicitudes y así mejorar el rendimiento.

- Incluyen herramientas de autenticación y autorización que permiten al servidor web identificar y restringir el acceso a determinadas funciones.

- Acceso, mapeo relacional de objetos y configuración de bases de datos, lo que permite que las aplicaciones funcionen con una variedad de bases de datos sin necesidad de realizar cambios en el código. Además de proporcionar otras utilidades como soporte transaccional y herramientas de migración de bases de datos.

- Mapeo y redirección de URLs (Uniform Resource Language), es el mecanismo por el cual el framework interpreta las URLs. Algunos utilizan patrones mediante expresiones regulares, mientras que otros emplean técnicas de reescritura para traducir la URL.

- Soporte para AJAX, abreviatura de "Asynchronous JavaScript and XML", una técnica que utiliza diferentes tecnologías web en el lado del cliente para crear aplicaciones asincrónicas en las que se pueden enviar y recuperar datos de forma asincrónica sin interferir con la visualización y el comportamiento de la página existente.

- Algunos frameworks proporcionan herramientas para desarrollar y consumir servicios web como por ejemplo soporte para servicios RESTful.

Existen numerosos tipos de web frameworks, cada uno orientado a unas necesidades diferentes. Desde su creación han sido utilizados para construir aplicaciones de uso general, empresarial, de una sola página (SPA, en inglés Single Page Application), APIs RESTful, y cada vez más para aplicaciones dinámicas como redes sociales o herramientas de colaboración en tiempo real.

Según los datos de la encuesta anual de Stack Overflow sobre las tendencias en el mundo de la programación [2], estos son los web frameworks y tecnologías más utilizados en 2024 (los más relevantes):

TECNOLOGÍAS	TIPO	CUOTA DE USO
Node.js	Entorno de ejecución	40.8 %
React	Biblioteca (front)	39.5 %
JQuery	Biblioteca (front)	21.4 %
Next.js	Framework (fullstack)	17.9 %
Express	Framework (back)	17.8 %
Angular	Framework (front)	17.1 %
ASP.NET CORE	Framework (fullstack)	16.9 %
Vue.js	Framework (front)	15.4 %
ASP.NET	Framework (fullstack)	12.9 %
Flask	Framework (back)	12.9 %
Spring Boot	Framework (back)	12.7 %
WordPress	CMS	11.8 %
FastAPI	Framework (back)	9.9 %
NestJS	Framework (back)	5.8 %
Nuxt.js	Framework (fullstack)	3.6 %

Tabla 2 – Tecnologías web más utilizadas (Stack Overflow Survey 2024)

Con estos datos se puede observar que las tecnologías más utilizadas en el mundo son Node.js [3], el entorno en tiempo de ejecución multiplataforma de código abierto para la capa del servidor y la biblioteca de JavaScript de código abierto orientada al desarrollo frontend React [4].

Respecto a los frameworks más utilizados y en los que se enfatizará a continuación, Next.js [5] lidera la tabla con un 17.9%, seguido muy de cerca por Express [6]. Cabe destacar también Vue.js

[7] como otro de los frameworks más utilizados y el intuitivo Vue Framework Nuxt.js [8] que a pesar de tener una cuota de uso bastante baja, 3.6%, su simplicidad y enfoque en la estructuración de proyectos basados en Vue.js lo convierten en una gran opción.

4.1.1 Next.js

Next.js es un framework basado en React que permite la creación de aplicaciones web fullstack, ya que reúne lo mejor de esta biblioteca para el desarrollo frontend y además incluye funcionalidades para backend y renderizado de servidor.

Este framework utiliza componentes React para crear interfaces de usuario y Next.js para funciones y optimizaciones adicionales. Cabe destacar que también abstrae y configura automáticamente las herramientas necesarias para React, como la agupación y la compilación, lo que permite centrarse directamente en la creación de la aplicación sin perder tiempo configurándolo.

Además, cuenta con un sistema de enrutamiento (App Router) que está basado en un sistema de archivos creado sobre componentes de servidor que admite diseños, rutas anidadas, estados de carga y manejo de errores. También permite renderizado del lado del cliente (CSR) y del servidor (SSR), y cuenta con renderizado estático (SSG) y dinámico compatible con Edge y Node.js.

Entre otras características, Next.js ofrece una obtención de datos simplificada utilizando async/await en componentes del servidor y admite memorización, almacenamiento en caché y revalidación. Este framework también es compatible con numerosos estilos como Tailwind CSS e incluye herramientas para optimizar tanto imágenes como fuentes y scripts, mejorando los elementos web básicos y la experiencia de usuario.

4.1.2 Express

A diferencia de Next.js, es un framework minimalista y altamente eficiente para Node.js, la otra tecnología más utilizada durante el año pasado. Cuenta con un diseño minimalista para simplificar la creación de aplicaciones web y servicios backend.

Express es utilizado para numerosas aplicaciones y servicios dentro del ecosistema de Node.js, facilita en gran medida la creación de APIs RESTful robustas para servir datos a aplicaciones frontend, tiene soporte para WebSocket y otras tecnologías en tiempo real y es una buena opción para aplicaciones financieras ya que es capaz de manejar transacciones de alta demanda eficientemente y de forma segura.

Entre las características principales de Express cabe destacar su enfoque minimalista y flexible, un sistema de enrutamiento robusto capaz de organizar rutas en archivos separados, el uso de middlewares para interceptar y procesar solicitudes HTTP, facilita el manejo de operaciones asíncronas que utilizan promesas en JavaScript y es compatible con el motor V8 de Google, garantizando una ejecución rápida y escalable.

Respecto al funcionamiento de Express, es un modelo de C/S típico de las aplicaciones web. Después de recibir una solicitud HTTP, utiliza un sistema de enrutamiento para determinar qué manejador la procesará. A continuación, la solicitud pasa a través de varios middlewares (validación de datos o manipulación) antes de llegar al manejador de rutas y finalmente después de procesar la solicitud envía una respuesta HTTP al cliente.

4.1.3 Vue.js

Vue.js es un marco y un ecosistema que recoge la mayoría de las características comunes en el desarrollo dedicado al frontend y está diseñado para ser flexible y adaptarse a las necesidades del desarrollador, aunque con conocimientos básicos de HTML y JavaScript, cualquier persona puede comenzar a trabajar con Vue debido a su fácil aprendizaje.

Vue cuenta con una gran flexibilidad a la hora de la implementación, permite trabajar con diferentes estrategias de desarrollo como aplicaciones de una sola página (SPA), renderización del lado del servidor (SSR), generación de sitios estáticos (SSG), mejorar HTML estático sin un paso de compilación e incrustar componentes en páginas existentes para así fomentar la reutilización de código.

Los componentes de Vue se pueden crear dos estilos de API diferentes: API de opciones y API de composición. Ambos estilos son totalmente capaces de cubrir casos de uso comunes, la API de opciones se centra en el concepto de una instancia de componente, similar a un entorno de lenguaje OOP (Object-Oriented Progrramming). La API de composición se centra en declarar variables reactivas directamente y componer estados de varias funciones para manejar la complejidad, es más libre y poderosa, aunque requiere comprender la reactividad en Vue.

- Con la API de opciones, se define la lógica en un componente mediante un objeto de opciones ‘data’, ‘methods’ y ‘mounted’. Las propiedades devueltas en data se convierten en reactivas y se podrán acceder mediante ‘this’. Los métodos son funciones que modifican el estado y desencadenan actualizaciones, pueden vincularse como manejadores de eventos en las plantillas. Los hooks del ciclo de vide se llaman en diferentes etapas del ciclo de vida de un componente, en el caso de ‘mounted’ se llamará cuando el componente se monte.
- Con la API de composición se define la lógica de un componente mediante funciones API importadas. En los Single File Component (SFC) se utiliza normalmente con ‘<script setup>’, el ‘setup’ atributo es una sugerencia que realiza transformaciones en tiempo de compilación que permiten usar la API de composición con menos código redundante. Por ejemplo, las importaciones y variables de nivel superior declaradas dentro del ‘<script setup>’ se pueden usar directamente en la plantilla HTML sin necesidad de crear el objeto ‘data’.

Todas estas herramientas hacen que Vue sea más que un simple framework, ya que cuenta con línea de comandos, herramientas de desarrollador, un empaquetador e incluso un framework que extiende sus capacidades por lo que se puede adaptar a muchos proyectos diferentes.

4.1.4 Nuxt.js

Nuxt utiliza convenciones y una estructura de directorios para automatizar tareas repetitivas molestas y permitir que los desarrolladores se centren en el desarrollo de funciones, mientras automatiza muchos aspectos del desarrollo.

Este framework viene con capacidades de renderizado del lado del servidor (SSR, en inglés Server-Side Rendering) integradas sin necesidad de configurar un servidor, lo que garantiza muchos beneficios para las páginas web. Este renderizado reduce considerablemente el tiempo de carga inicial de la página, ya que Nuxt envía una página HTML completamente renderizada al navegador que puede visualizarse de inmediato. Además, los motores de búsqueda pueden indexar mejor las páginas SSR al tener el contenido HTML disponible inmediatamente, en lugar de usar JavaScript en el lado del Cliente. Al estar disponible el contenido inmediatamente después de la carga inicial, mejora la accesibilidad para los usuarios que dependen de lectores de pantalla u otras tecnologías. También cuenta con un almacenamiento en caché más sencillo, ya que las páginas se pueden almacenar en caché en el lado del servidor mejorando aún más el rendimiento. Al ser un marco versátil, ofrece la posibilidad de renderizar estáticamente toda la aplicación en un alojamiento estático con nuxt generate, deshabilitar SSR globalmente o aprovechar la renderización híbrida (routeRules).

Por otra parte, Nuxt proporciona un sistema de módulos para ampliar el núcleo del marco y simplificar integraciones. Estos módulos pueden anular plantillas, configurar cargadores de paquetes web, agregar bibliotecas CSS y realizar muchas otras tareas útiles. Lo mejor de todo es que se pueden distribuir en paquetes npm, para poder reutilizar en diferentes proyectos y compartirlos con la comunidad.

La arquitectura de este framework está caracterizada por un motor central (nuxt) que coordina la interacción entre los diferentes paquetes y herramientas. Además, posee dos paquetes (@nuxt/vite-builder y @nuxt/vite-builder) encargados de compilar y empaquetar el código la aplicación. Al igual que Vue, posee una línea de comandos (nuxi) que simplifica la creación y gestión de proyectos. También cuenta con un motor de servidor (nitro) y un kit de desarrollo (@nuxt/kit) para desarrolladores avanzados y creadores de módulos de Nuxt.

En resumen, Nuxt es un framework gratuito y de código abierto que simplifica el desarrollo full-stack con Vue.js y está centrado en el rendimiento, accesibilidad y flexibilidad ofreciendo una gran opción para desarrollar el prototipo.

4.1.5 Conclusiones

Todas las tecnologías repasadas son válidas para el desarrollo de un sitio web como el del prototipo deseado. Sin embargo, para la elección del framework adecuado se han tenido en cuenta factores como:

- Facilidad de uso y mantenimiento: Las tecnologías seleccionadas no tienen que ser muy complicadas de usar y mantener, para facilitar la curva de aprendizaje y el mantenimiento del sitio web.

- Rendimiento: Dado que la aplicación deberá realizar búsquedas y mostrar contenido dinámico, es muy importante que el framework seleccionado disponga de herramientas para optimizar tiempos de carga y mejorar el rendimiento.

- Flexibilidad y escalabilidad: Es muy importante que los marcos de trabajo ofrezcan buena flexibilidad para adaptarse a las necesidades de la solución y que también sean escalables para asegurar la eficiencia en un entorno con mucho volumen de datos.

- Documentación: Aunque parezca menos importante, una comunidad activa y una amplia documentación son de gran ayuda a la hora de resolver problemas y desarrollar código, ya que cuanto mayor es la comunidad, más información se puede encontrar.

- Familiaridad con la tecnología: Las tecnologías utilizadas y en proceso de aprendizaje son un punto a valorar, a la hora de seleccionar framework, ya que el conocimiento previo reduce significativamente el tiempo de desarrollo e implementación de la solución.

Teniendo en cuenta lo descrito anteriormente, la mejor opción para el desarrollo del frontend es de Vue Nuxt, que a pesar de no tener tanta popularidad garantiza una solución moderna y optimizada con un proyecto ordenado y muchas funcionalidades como el SSR ya integrado que mejoran el rendimiento del prototipo. Para el backend, más sencillo que el frontend, se ha seleccionado el marco minimalista y eficiente Node.js Express. Ambos utilizan o están basados en dos de las tecnologías más empleadas en el mundo del desarrollo web en el último año (Vue.js y Node.js).

4.2 Entorno de desarrollo

Para llevar a cabo el desarrollo del proyecto, se ha configurado un entorno de trabajo moderno y eficiente, compuesto por herramientas ampliamente utilizadas en la industria del desarrollo web. Este entorno ha facilitado la implementación modular del sistema, el control de versiones, la gestión de dependencias y la correcta comunicación entre el frontend y el backend.

El desarrollo se ha realizado sobre un sistema operativo Windows 10. Para la edición del código se ha utilizado Visual Studio Code (VSCode), un editor ligero, extensible y con una amplia comunidad. Este editor ofrece soporte nativo para JavaScript, TypeScript, Vue y otras tecnologías utilizadas en el proyecto, además de integrar funcionalidades como depuración, control de versiones y terminal incorporada.

El lenguaje principal utilizado en todo el proyecto es JavaScript, complementado con TypeScript en ciertas partes del frontend para una mayor robustez y facilidad de mantenimiento. En el

frontend se ha empleado Nuxt 3, un framework de alto nivel basado en Vue.js, que permite desarrollar aplicaciones web con renderizado híbrido (SSR + SPA), gestión automática de rutas y división de código por páginas. Para el backend se ha utilizado Node.js junto con el framework Express, debido a su sencillez para crear APIs RESTful y su gran rendimiento.

Para gestionar el control versiones se ha utilizado Git, con GitHub como repositorio remoto. Para ello se ha utilizado una estructura básica de ramas, manteniendo una rama principal (main) y otra rama secundaria (develop) para el desarrollo de nuevas funcionalidades o corrección de errores. Esto facilitó mantener un historial de cambios ordenado y un trabajo incremental.

Las dependencias tanto en el frontend como en el backend han sido gestionadas con NPM (Node Package Manager). Definiendo los paquetes necesarios en los respectivos archivos package.json, permitiendo la instalación automática de estos en nuevos entornos de desarrollo. Adicionalmente, se utilizó un archivo .gitignore para excluir del control de versiones carpetas innecesarias como node_modules.

El código del proyecto se ha organizado en dos carpetas principales: /frontend y /backend. Esta separación clara entre el cliente y el servidor facilita el mantenimiento, el despliegue por separado en el futuro y una mayor modularidad del sistema. Cada carpeta contiene su propia configuración, scripts de ejecución y dependencias. Además se han utilizado otras herramientas complementarias como Insomnia para probar las rutas de la API de forma rápida y verificar la correcta comunicación entre el frontend y backend. También se ha utilizado ESLint y Prettier en el editor para mantener una base de código limpia y fácil de leer. Otra herramienta utilizada ha sido Trello, para gestionar la planificación y seguimiento de tareas.

4.3 Desarrollo del frontend

El desarrollo del frontend se ha llevado a cabo utilizando Nuxt 3, un framework basado en Vue.js que permite construir aplicaciones modernas, rápidas y escalables. A continuación se detalla cómo se implementaron las diferentes partes del frontend.

4.3.1 Gestión de dependencias y scripts

El proyecto utiliza una serie de librerías que facilitan el desarrollo y mejoran las funcionalidades del frontend. Estas herramientas incluyen bibliotecas para la gestión de estilos, optimización de imágenes, internacionalización, y componentes de interfaz de usuario. Además, se integran utilidades para mejorar la experiencia del desarrollador como formateadores de código y procesadores de estilos. Las más importantes son las siguientes:

- @nuxt/image: Para optimización y manipulación de imágenes, mejora el rendimiento de la aplicación.
- @nuxtjs/i18n: Gestiona la internacionalización facilitando la traducción de texto y la configuración de diferentes idiomas. Implementado en el componente LangSelector.vue para cambiar entre los idiomas disponibles (castellano e inglés)

- @nuxtjs/tailwindcss: Proporciona una forma rápida y eficiente de diseñar interfaces responisvas y modernas. Utilizado en todo el proyecto para personalización de componentes.
- @popperjs/core: Biblioteca para gestionar elementos emergentes, utilizado en componentes como el PopperFilter.vue y la searchBar.vue.
- leaflet: Biblioteca para mapas interactivos que permite mostrar mapas, marcadores y realizar interacciones geográficas. Ha sido utilizada en el componente LeafletMap.vue para mostrar la ubicación de los alojamientos.
- litepicker: Selector de fechas ligero y personalizable que se ha utilizado en el componente LiteCalendar.vue para la selección de fechas.
- nuxt: Framework principal del proyecto que proporciona una estructura modular para construir aplicaciones webs con Vue.js, es la base del frontend incluyendo enrutamiento y Server-Side rendering
- vue: Framework de JavaScript para construir interfaces de usuario reactivas, es la base de todos los componentes del proyecto.
- @nuxtjs/proxy: Permite configurar proxies para redirigir solicitudes HTTP durante el desarrollo.
- postcss: Es utilizado por TailwindCSS para procesar los estilos
- prettier: Herramienta para formatear el código durante el desarrollo.

Para instalar las dependencias del proyecto, se utiliza el gestor de paquetes NPM (Node Package Manager). El comando principal para instalar las dependencias que se encuentran en el archivo package.json es “npm install”.

Además, en el archivo package.json se encuentran predefinidos los scripts que automatizan tareas comunes durante el desarrollo, la construcción y el despliegue del proyecto. Estos scripts utilizan también el gestor de paquetes NPM .A continuación, se explica cada uno de ellos en detalle:

Ilustración 7 Scripts package.json

- “dev”: Este script lanza la aplicación en modo desarrollo, permitiendo realizar cambios en el código y verlos reflejados en tiempo de ejecución sin tener que reiniciar el servidor. Una vez ejecutado, la aplicación se encuentra disponible en http://localhost:3000.

- “build”: Este script genera los archivos optimizados que se utilizarán en un entrono de producción, comprimiendo archivos de JavaScript y CSS y preprocesando imágenes y otros recursos estáticos. Los archivos generados se almacenan en una carpeta de salida.
- “deploy”: Este script combina la construcción del proyecto con el inicio del servidor en producción, construye la aplicación con “build” e inicia el servidor ejecutando el archivo index.mjs situado en la carpeta de salida.
- “generate”: Este script convierte la aplicación en un sitio estático, generando archivos HTML para cada página. Facilita el despliegue en plataformas de hosting estático como Netlify, Vercel o GitHub Pages. Los archivos se almacenan en la carpeta ./dist.
- “preview”: Este script inicia un servidor local para revisar cómo se verá el sitio estático generado con “generate”.
- “postinstall”: Este script se ejecuta automáticamente después de instala las dependencias para generar archivos de configuración necesarios para Nuxt y preparar el entorno para el desarrollo o la construcción.

4.3.2 Arquitectura

El frontend de este proyecto está desarrollado con Nuxt, un potente framework basado en Vue.js que facilita la creación de aplicaciones web modulares, escalables y mantenibles. La arquitectura se ha diseñado para maximizar la reutilización de componentes, para maximizar la reutilización de componentes y para facilitar el mantenimiento.

La estructura de las carpetas y sus principales archivos están explicados a continuación:

- components/: Esta carpeta incluye los componentes de la aplicación, organizados en subcarpetas según sus diferentes funciones.
    - accommodation/: Componentes relacionados con la visualización y gestión de los alojamientos (Actions.vue, Availability.vue, Booking.vue, Data.vue, etc). Incluye subcarpetas como gallery/ para gestionar las imágenes de los alojamientos.
    - basic/: Componentes básicos y genéricos, incluye formularios (Form.vue), checkboxes (Checkbox.vue), popups (Popup.vue) y barras de búsqueda (searchBar.vue) entre otros componentes.
    - booking/: Componentes específicos para la gestión de las reservas (RequestForm.vue y RequestPopup.vue)
    - calendar/: Componentes para la visualización y selección de fechas (Month.vue, MonthSelector.vue y LiteCalendar.vue).
    - head/: Componentes relacionados con la cabecera de la aplicación (LangSelector.vue).
    - icons/: Iconos SVG (Scalable Vector Graphics) como componentes Vue (ArrowLeft.vue, Calendar.vue, Camera.vue, etc).
    - index/: Componentes principales para la página de inicio de la aplicación (AboutUs.vue, Ideal.vue y Main.vue).
    - search/: Componentes específicos para la página de búsqueda que incluyen los filtros de esta, las tarjetas de los alojamientos, la paginación, etc. (Price.vue, Types.vue, FilterRow.vue, AccCard.vue, Pagination.vue, etc.)

- pages/: Cada archivo o carpeta que se encuentra dentro de esta carpeta representa una ruta de la aplicación. Nuxt se encarga automáticamente de generar el sistema de rutas a partir la estructura de esta carpeta.
    - index.vue: Página principal
    - contact.vue, faqs.vue, terms.vue: Páginas estáticas de contacto, preguntas frecuentes y términos.
    - details/[slug].vue, search/[slug].vue: Rutas dinámicas para mostrar los detalles de alojamientos y los resultados de búsqueda dado un parámetro “slug”.
- layouts/: En esta carpeta se encuentran las plantillas generales que utilizan las diferentes pages de la aplicación.
- lang/: En esta carpeta se encuentran los archivos que almacenan los textos de los diferentes idiomas que soporta la apliación en formato JSON (en.json y es.json).
- public/: Almacena recursos estáticos como imágenes y archivos que deben ser accesibles desde el navegador.
- utils/: Incluye funciones y diferentes utilidades que pueden ser utilizadas en el proyecto, facilitando la reutilización.
- types/: En esta carpete se definen diferentes tipos de TypeScript para manejar más fácilmente los datos.

4.3.3 Flujo de datos y estado

El flujo de los datos y la gestión del estado en el frontend están basados en los principios de Vue3 y Nuxt3. Estos principios aprovechan la reactividad y la composición de los componentes para mantener una aplicación organizada y eficiente.

La comunicación entre componentes se realiza utilizando props y eventos. Por ejemplo, el componente basic/searchBar.vue recibe la prop ‘where=”index”’ desde el componente index/Main.vue para adaptar su comportamiento según el contexto. Las props se utilizan para comunicarse de padres a hijos y los eventos de forma inversa. Tras definir los eventos en el componente hijo, el padre los detecta (‘@evento=método’) y realiza el tratamiento para el caso específico.

Para gestionar el estado global de la aplicación, Nuxt3 proporciona composables (funciones reutilizables que encapsulan lógica reactiva). Estos composables pueden almacenar y compartir datos entre diferentes componentes y páginas, como el estado de la búsqueda. Un ejemplo es useAsyncData, este composable permite obtener y gestionar datos de forma asíncrona y reactiva, integrando el estado de la petición y facilitando la actualización de los datos que aparecen en la interfaz.

A continuación, se muestra cómo se ha utilizado en el archivo search/Main.vue para obtener los resultados de búsqueda de alojamientos desde el backend, gestionando el estado de la petición y actualizando automáticamente la interfaz cuando los datos cambian.

```
const { data, status, error, refresh } = useAsyncData(
  'fetchSearchResults',
  async () => {
    const city = route.params.slug || '';
    const query = new URLSearchParams({
      ...route.query,
      city: city,
    }).toString();

    return await $fetch(`/api/accommodations/list?${query}`);
  }
);
```

Ilustración 8 Solicitud del archivo search/Main.vue

Data contiene los resultados de la búsqueda, status indica el estado de la petición, error almacena información sobre los posibles errores y refresh permite volver a lanzar la petición y actualizar los datos.

En archivos como details/[slug].vue o search/[slug].vue, Nuxt utiliza rutas dinámicas para cargar y mostrar los datos según el parámetro de la URL (slug). Esto hace posible que cada página obtenga los datos necesarios (por ejemplo, detalles de un alojamiento) y los pase a los componentes correspondientes.

4.3.4 Estilos y diseño

El diseño visual del frontend se ha implementado principalmente utilizando Tailwind CSS, permitiendo un desarrollo rápido, coherente y responsivo de la interfaz de usuario. La integración de Tailwind se puede observar en todos los componentes, tanto en las clases de los templates como en el uso de la directiva @apply en el apartado de estilos.

El uso de Tailwind permite definir estilos complejos de forma clara y reutilizable, manteniendo el CSS organizado y fácil de modificar. En el archivo tailwind.config.js se permite personalizar la paleta de colores, tamaños, fuentes, etc. De esta forma todos los componentes comparten la misma base de diseño.

Para mantener un diseño responsive se han utilizado prefijos como md:, lg:, xl: en las clases de Tailwind para adaptar el diseño a diferentes dispositivos, asegurando que la aplicación sea completamente responsiva. También se han utilizado sombras y gradientes ('primary-gradient': 'linear-gradient(90deg, rgba(39, 167, 175, 1) 0%, rgba(31, 131, 137,1) 100%)'), fuentes personalizadas (sans: ['"DM Sans"', 'sans-serif']), diferentes colores (primary: 'rgba(31, 131, 137, var(--tw-bg-opacity))'), etc.

El uso de <style scoped lang=”postcss”> en los componentes asegura que los estilos definidos solo afecten al componente correspondiente evitando así conflictos entre ellos.

4.3.5 Rutas y navegación

La gestión de rutas y la navegación en el frontend se realiza de forma automática gracias a las capacidades de Nuxt3. El framework genera el sistema de rutas a partir de la estructura de la carpeta /pages, haciendo mucho más fácil la organización y mantenimiento de la aplicación. Como se ha mencionado anteriormente en el apartado de la arquitectura, cada archivo o subcarpeta dentro de /pages representa una ruta de la aplicación, index.vue corresponde a la ruta principal mientras que contact.vue, faqs.vue y terms.vue generan rutas estáticas como /contact, /faqs y /terms. Los archivos /details/[slug].vue y search/[slug].vue definen rutas dinámicas donde el 'slug' es un parámetro dinámico que se utiliza para obtener la información específica para esos detalles de alojamiento o búsqueda.

Para la navegación entre páginas se utiliza el componente <NuxtLink> de Nuxt que permite cambiar de ruta sin recargar la página, manteniendo la experiencia de una SPA (Single Page Application). Además, en algunos casos se utiliza la navegación programática utilizando el método router.push. Esto permite cambiar de ruta desde el código en JavaScript tras una acción del usuario o para modificar los parámetros de la URL dinámicamente.

4.3.6 Comunicación con el backend

La comunicación entre el frontend y el backend permite que la aplicación muestre información dinámica y responda a las acciones del usuario en tiempo real. El frontend (Nuxt) interactúa con el backend (Node.js + Express) a traveés de peticiones HTTP a endpoints RESTful.

En los componentes y páginas del frontend, estas peticiones son asíncronas para obtener de forma adecuada los datos del backend (listados de alojamientos, detalles, sugerencias, etc). Para llevarlo a cabo se utilizan funciones como $fetch integradas en Nuxt, que permiten realizar solicitudes GET y POST.

Estas peticiones suelen incluir parámetros dinámicos como el slug de un alojamiento, los filtros o los términos de búsqueda introducidos por el usuario. Estos parámetros se construyen a partir del estado de la aplicación y se envían en la query de la petición o en el cuerpo de ésta.

Además, para facilitar el desarrollo y evitar problemas de CORS (Cross-Origin Resource Sharing), se ha configurado un proxy en el archivo 'nuxt.config.ts' utilizando la opción devProxy de Nitro. Gracias a esto todas las peticiones que se realizan a rutas que comienzan por /api son redirigidas automáticamente al backend permitiendo que el frontend y backend se ejecuten en servidores diferentes durante el desarrollo, pero se comuniquen como si estuvieran en el mismo dominio.

En resumen, la comunicación con el backend está perfectamente integrada en el flujo de la aplicación permitiendo que el usuario interactúe con los datos de forma ágil, segura y eficiente.

4.4 Desarrollo del backend

El backend ha sido desarrollado utilizando Node.js y el framework Express.js, con un enfoque modular y organizado para facilitar su mantenimiento y escalabilidad. El objetivo es gestionar las solicitudes y respuestas entre el frontend y la base de datos, proporcionando una API robusta y eficiente.

4.4.1 Estructura

La estructura del backend está bien organizada y sigue las buenas prácticas de desarrollo para garantizar una solución escalable y mantenible. El archivo principal "server.js" actúa como punto de entrada del servidor. En este archivo se inicializa la aplicación Express, se configuran los middlewares, las rutas y la conexión con la base de datos. También se define el puerto en el que el servidor escuchará las solicitudes y se establece la configuración de CORS para permitir la comunicación con el frontend.

En la carpeta "config" se encuentra el archivo "db.js" que gestiona la conexión a la base de datos mediante un pool de conexiones, esto permite manejar múltiples solicitudes simultáneamente de manera eficiente. Se utiliza el paquete "mysql2/promise" para interactuar con la base de datos utilizando operaciones asíncronas (async/await).

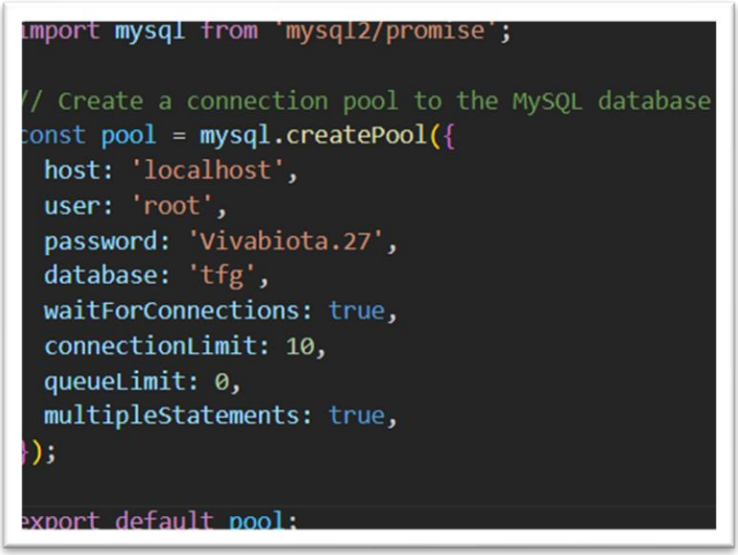


Ilustración 9 Connection pool - db.js

En la carpeta routes se encuentran los diferentes módulos de rutas, cada uno encargado de una funcionalidad de la aplicación:

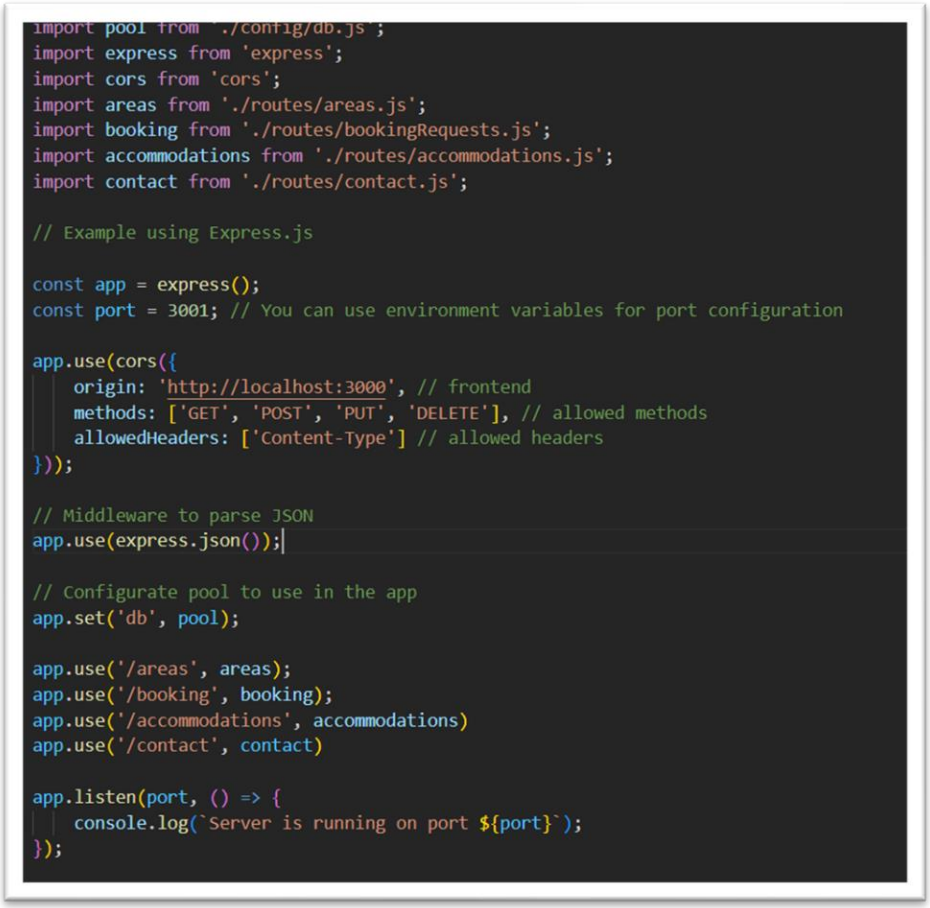
- accommodations.js: Gestiona las peticiones relacionadas con los alojamientos, como la obtención de los detalles, filtrado y listado.
- areas.js: Maneja las peticiones para las sugerencias de áreas en la barra de búsqueda.

- bookingRequests.js: Se encarga de las peticiones para la gestión de solicitudes de reserva realizadas en el frontend.
- contact.js: Gestiona la petición para enviar el correo de contacto.

En el archivo package.json se encuentran definidas las dependencias del proyecto (express, mysql2, cors), los scripts de ejecución y la configuración básica del entorno Node.js permitiendo gestionar fácilmente las bibliotecas necesarias para el correcto funcionamiento del backend.

4.4.2 Configuración del servidor

La configuración del servidor es un aspecto importante para el funcionamiento del backend, la seguridad y la comunicación eficiente. Como se ha mencionado anteriormente el servidor se configura principalmente en el archivo server.js. A continuación, se muestra el código de este archivo y se explican los conceptos más importantes.



```
import pool from './config/db.js';
import express from 'express';
import cors from 'cors';
import areas from './routes/areas.js';
import booking from './routes/bookingRequests.js';
import accommodations from './routes/accommodations.js';
import contact from './routes/contact.js';

// Example using Express.js

const app = express();
const port = 3001; // You can use environment variables for port configuration

app.use(cors({
    origin: 'http://localhost:3000', // frontend
    methods: ['GET', 'POST', 'PUT', 'DELETE'], // allowed methods
    allowedHeaders: ['Content-Type'] // allowed headers
}));

// Middleware to parse JSON
app.use(express.json());

// Configurate pool to use in the app
app.set('db', pool);

app.use('/areas', areas);
app.use('/booking', booking);
app.use('/accommodations', accommodations)
app.use('/contact', contact)

app.listen(port, () => {
    console.log(`Server is running on port ${port}`);
});
```

Ilustración 10 Configuración archivo server.js

Antes de nada, es necesario importar los módulos esenciales para el correcto funcionamiento del servidor. El módulo de Express es el principal framework para la creación del servidor y la gestión de rutas, Cors es un middleware que permite la comunicación entre el frontend y backend permitiendo solicitudes desde otros orígenes y pool es la conexión a la base de datos

mencionada anteriormente. Además se importan también las cuatro rutas de la carpeta /routes para agregarlas a la aplicación de Express.

La primera parte después de las importaciones crea una instancia de Express que servirá como base para definir los middlewares y rutas de la aplicación, se escoge el puerto 3001 para que el backend escuche las solicitudes HTTP entrantes.

Justo debajo se utiliza el middleware CORS (Cross-Origin Resource Sharing) para permitir que el frontend (en este caso http://localhost:3000) pueda comunicarse con el backend. Sin embargo, como se ha mencionado anteriormente, el frontend utiliza un proxy de desarrollo por lo que las peticiones de éste no se envían directamente desde el navegador, sino que pasan primero por el proxy y el navegador interpreta que todas las peticiones provienen del mismo origen y no aplica restricciones CORS. Aunque no sea necesario mantener la configuración de CORS en el backend puede resultar útil en escenarios de producción donde si pueden existir orígenes distintos.

A continuación se añade un middleware 'express.json()' para que el servidor pueda interpretar correctamente las solicitudes con cuerpo en formato JSON, se añade la conexión con la base de datos en la aplicación Express y se agregan también las diferentes rutas mencionadas anteriormente.

4.4.4 Escalabilidad y Seguridad

El backend ha sido diseñado con un enfoque tanto en la escalabilidad como en la seguridad, siguiendo una estructura modular con rutas separadas por funcionalidad y una configuración centralizada de la base de datos, esto permite que el sistema crezca de forma adecuada. Es muy sencillo añadir nuevas rutas o ampliar la lógica presente en éstas sin que llegue a afectar al resto del código.

Utilizando un pool de conexiones a la base de datos se garantiza que la aplicación pueda gestionar múltiples solicitudes de manera eficiente, evitando cuellos de botella y mejorando el rendimiento de carga de la aplicación.

Con respecto a la seguridad de ésta se aplican varias medidas para proteger la aplicación y los datos que están explicadas a continuación:

- Consultas parametrizadas: Todas las consultas utilizan parámetros en lugar de concatenar cadenas, previniendo ataque de inyección SQL (Structured Query Language).
- CORS: Aunque como se ha comentado anteriormente puede no ser imprescindible en desarrollo, el backend incluye configuración CORS para controla qué orígenes pueden acceder a la API, algo esencial en producción.
- Validación de datos: En rutas que reciben datos del usuario (routes/contact.js) se valida que los campos requeridos no sean nulos antes de procesar la solicitud.

- Gestión de credenciales: Las credenciales sensibles, como las que aparecen en el archivo config/db.js deberían gestionarse mediante variables de entorno en un entorno real para evitar su exposición.

4.5 Diseño de la base de datos

El diseño de la base de datos es un elemento clave dentro del desarrollo de esta aplicación, ya que permite estructurar de forma eficiente la información relacionada con los alojamientos y su gestión. Para su elaboración se han seguido las tres fases clásicas de diseño de bases de datos relacionales: diseño conceptual, diseño lógico y diseño físico [17].

4.5.1 Diseño conceptual

En la fase conecptual se identificaron las entidades principales a partir de los requisitos del sistema, es decir, los elementos de información necesarios para cubrir las funcionalidades de la plataforma. Esta etapa se ha realizado sin entrar en detalles técnicos, centrándose únicamente en qué datos debían gestionarse. A continuación, se muestra el diseño conceptual realizado, se han representado como “attributes” los atributos que no son el identificador de las entidades Accommodations y Booking Requests, ya que se recargaría mucho la imagen.

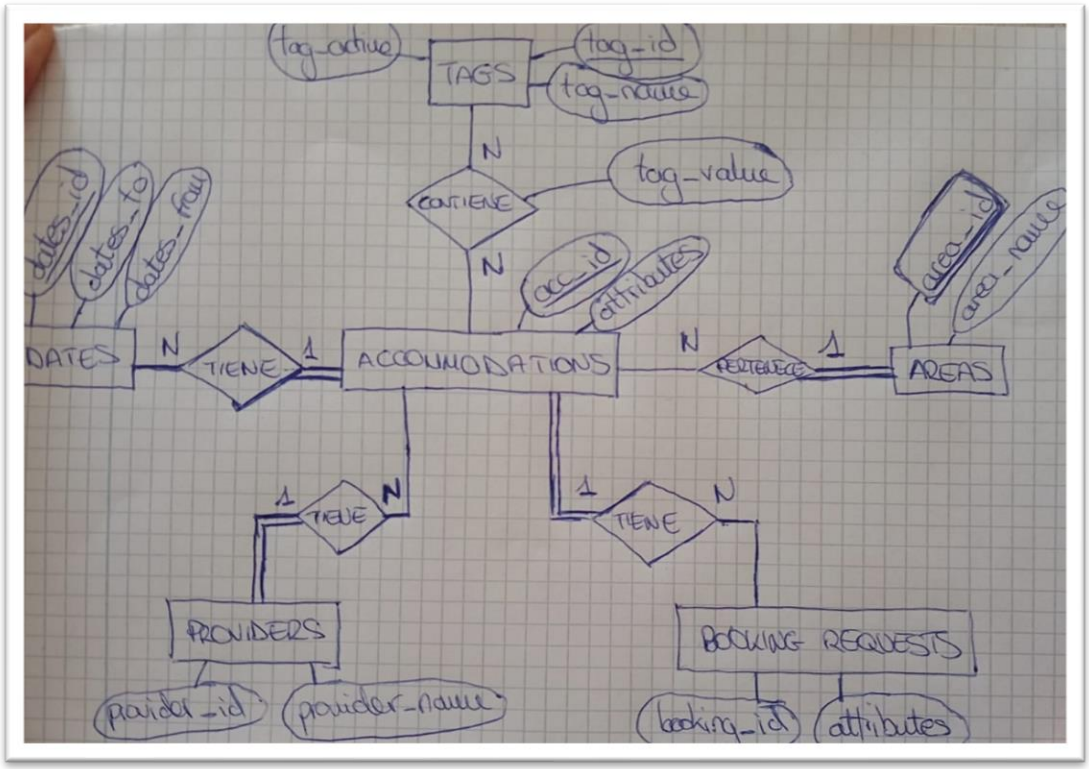


Ilustración 11 Diseño conceptual

4.5.2 Diseño lógico

A partir del modelo físico implementado, se obtiene el siguiente modelo lógico relacional, donde se definen las tablas, atributos, claves primarias, claves foráneas y relaciones entre entidades.

ACCOMMODATIONS
Clave primaria: acc_id (INT)
Atributos:

- acc_type (ENUM)
- acc_provider (clave foránea a PROVIDERS.provider_id)
- acc_bedrooms_count (INT)
- acc_bathrooms_count (INT)
- acc_en_desc (MEDIUMTEXT)
- acc_es_desc (MEDIUMTEXT)
- acc_address (VARCHAR(512))
- acc_latitude (DOUBLE)
- acc_longitude (DOUBLE)
- acc_price (INT)
- acc_deposit (INT)
- acc_img (MEDIUMTEXT)
- min_stay (INT)
- max_stay (INT)
- freq_stay (VARCHAR(45))
- relevance (INT)
- acc_area (clave foránea a AREAS.area_id)

PROVIDERS
Clave primaria: provider_id (INT)
Atributos:

- provider_name (VARCHAR(45))

AREAS
Clave primaria: area_id (INT)
Atributos:

- area_name (VARCHAR(45))

DATES
Clave primaria: dates_id (INT)
Atributos:

- dates_to (DATE)
- dates_from (DATE)
- dates_accommodation (clave foránea a ACCOMMODATIONS.acc_id)

BOOKING_REQUESTS
Clave primaria: booking_id (INT)
Atributos:

- booking_acc (clave foránea a ACCOMMODATIONS.acc_id)
- booking_email (VARCHAR(512))
- booking_date (TIMESTAMP)
- booking_question (VARCHAR(512))
- booking_how (VARCHAR(45))
- booking_enquiry (VARCHAR(45))
- booking_dateTo (DATE)
- booking_dateFrom (DATE)
- booking_lang (VARCHAR(45))

TAGS
Clave primaria: tag_id (INT)
Atributos:

- tag_name (TEXT)
- tag_active (INT)

ACCOMMODATIONS_TAGS
Clave primaria compuesta: (acc_id (INT), tag_id (INT))

- tag_value (INT)
- acc_id (clave foránea a ACCOMMODATIONS.acc_id)
- tag_id (clave foránea a TAGS.tag_id)

4.5.3 Diseño físico

Finalmente, después de realizar los diseños conceptual y lógico se ha obtenido el diseño físico de la base de datos. Esta fase consiste en la implementación práctica del modelo lógico sobre el sistema de gestión de bases de datos (SGBD). Para ello se ha utilizado la herramienta MySQL Workbench [18], que permite la creación visual de modelos de bases de datos, la definición de relaciones entre tablas, generación automática de los scripts SQL de creación de bases de datos y la validación del modelo antes de su implementación final.

A continuación, se muestra una imagen que representa el diseño físico de la base datos:

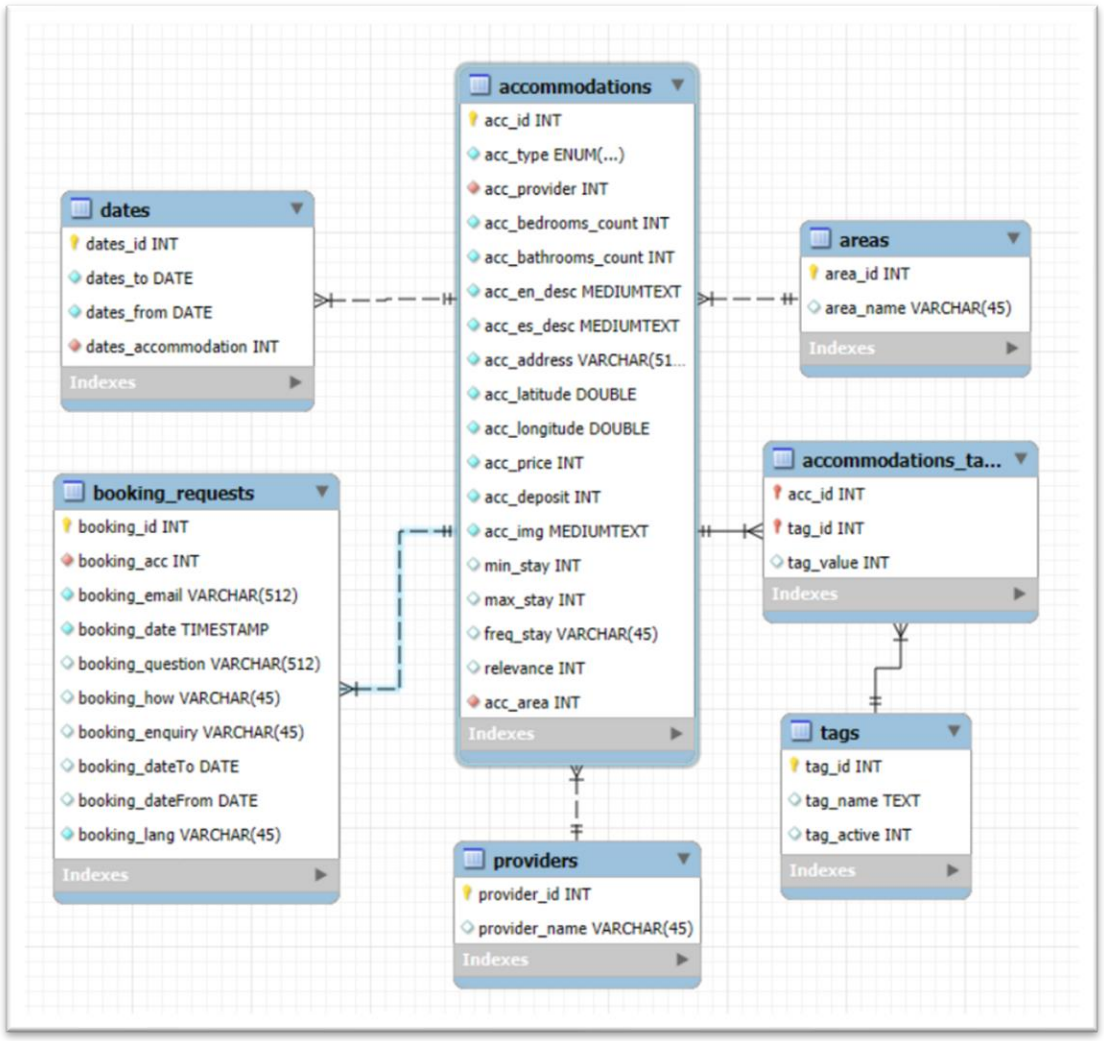


Ilustración 12 Diseño físico de la base de datos

4.5.4 Accommodations

Accommodations es la tabla principal que representa cada alojamiento, está relacionada de forma 1:N con las tablas de dates y booking_requests, es decir, para cada alojamiento habrá 1 o más ocurrencias de estas tablas. También se relaciona de forma N:1 con las tablas providers y areas, es decir, para cada provider habrá 1 o más ocurrencias de alojamientos y finalmente se relaciona con la tabla tags de forma N:N por lo que aparece la tabla accommodations_tags recogiendo como claves primarias las claves de accommodations y tags.

Los atributos de la tabla son los siguientes:

- acc_id: Clave primaria del alojamiento.
- acc_type: Tipo de alojamiento (por ejemplo, habitación, estudio, etc.)
- acc_provider: Relación con el proveedor (clave ajena de providers).
- acc_bedrooms_count, acc_bathrooms_count: Número de habitaciones y baños.
- acc_en_desc / acc_es_desc: Descripción del alojamiento en inglés y español.
- acc_address, acc_latitude, acc_longitude: Dirección y coordenadas del alojamiento.

- acc_price, acc_deposit: Precio y depósito.
- acc_img: Imágenes asociadas (url).
- min_stay / max_stay: Estancias mínima y máxima permitida.
- freq_stay: Frecuencia de pagos (mensual, semanal, etc.)
- relevance: Valor utilizado para ordenar por relevancia.
- acc_area: Relación con las áreas (clave ajena de areas).

4.5.5 Areas

Esta tabla contiene las áreas geográficas que hacen referencia a las diferentes ciudades que aparecen en la aplicación. Los atributos de la tabla son los siguientes:

- area_id: Clave primaria
- area_name: Nombre del área.

Como se ha mencionado en la tabla accommodations, estas dos tablas se relaacionan mediante acc_area.

4.5.6 Providers

Esta tabla registra las entidades o empresas que gestionan los alojamientos. Sus atributos son los siguientes:

- provider_id: Clave primaria.
- provider_name: Nombre del proveedor.

Se relaciona con accommodations mediante acc_provider.

4.5.7 Dates

Representa la disponibilidad de cada alojamiento, permitiendo tener múltiples rangos de fechas por alojamiento.

- dates_id: Clave primaria.
- dates_to, dates_from: Rango de fechas en las que el alojamiento está disponible.
- dates_accommodation: Clave ajena de accommodations.

4.5.8 Booking_requests

Esta tabla se encarga de almacenar las solicitudes de reserva realizadas por los usuarios.

- booking_id: Clave primaria.
- booking_acc: Relación con accommodations (clave ajena de accommodations).

- booking_email: Email del usuario que hace la solicitud.
- booking_date: Fecha de la solicitud.
- booking_question, booking_query: Preguntas o mensajes enviados por el usuario.
- booking_how: Cómo encontró la plataforma.
- booking_dateFrom, booking_dateTo: Fechas en las que el usuario desea reservar.
- booking_lang: Idioma preferido del usuario.

4.5.9 Tags

Contiene características que se pueden asignar a los alojamientos (single bed, double bed, internet, etc.)

- tag_id: Clave primaria.
- tag_name: Nombre de la característica
- tag_active: Indica si la etiqueta está activa (1 = sí, 0 = no)

4.5.10 Accommodations_tags

Es la tabla que representa la relación N:N entre accommodations y tags.

- acc_id: Clave ajena hacia accommodations.
- tag_id: Clave ajena hacia tags.
- tag_value: Representa el valor de la etiqueta (1 = está presente, 2 = puede estar presente).

5. Accesibilidad y Usabilidad

5.1 Accesibilidad

Facilitar el uso de la aplicación a cualquier persona es fundamental. La accesibilidad web [13] busca este objetivo y para ello hay que garantizar que todas las personas, incluidas aquellas con algún tipo de discapacidad, puedan utilizar y entender el sitio web sin ningún problema.

Las normativas WCAG 2 [14] elaboradas por el Grupo de Trabajo de las Pautas de Accesibilidad (AG WG), el cual forma parte de la Iniciativa para la Accesibilidad de la Web (WAI) del World Wide Web Consortium (W3C) [15] que dictan una serie de pautas que explican cómo hacer el contenido web más accesible para todas las personas.

Las WCAG 2 se estructuran en torno a cuatro principios fundamentales:

- **Perceptible**: La información y los componentes que aparecen en la interfaz de usuario deben ser percibidos por cualquier persona, independientemente de sus capacidades sensoriales. Un ejemplo sería ofrecer alternativas textuales para imágenes o subtítulos para vídeos.
- **Operable**: Todos los elementos con los que se pueda interactuar deben poder ser utilizados también por cualquier usuario. Un ejemplo sería que la navegación y las diferentes funcionalidades sean accesibles mediante teclado y evitar elementos que puedan provocar reacciones físicas adversas como animaciones bruscas o destellos.
- **Comprensible**: El contenido y el funcionamiento de la interfaz deben ser fáciles de entender e intuitivas, la información debe aparecer de forma clara y la navegación debe ser predecible.
- **Robusto**: La página web debe estar desarrollada de tal manera que su contenido pueda ser interpretado por la mayor cantidad posible de navegadores y tecnologías de asistencia, garantizando que la accesibilidad se mantenga a lo largo del tiempo y para todos los usuarios.

Para realizar el análisis de accesibilidad del prototipo desarrollado se ha empleado la misma herramienta que se ha utilizado para analizar sitios web del Estado del Arte, el IBM Accesibility Checker.

Utilizando esta herramienta se han analizado las principales páginas del sitio web desarrollado:

- Home: Los resultados obtenidos han sido 5 errores que tienen que ser corregidos y 4 advertencias que necesitan revisión. Por otra parte, no se han encontrado errores en el 93% de los elementos analizados. Entre los errores más relevantes se encuentran:
    - Falta de etiqueta visible en campos de entrada
    - Falta de visibilidad del foco del teclado (algunos botones no muestran claramente cuando están enfocados mediante el uso del teclado)
    - Uso exclusivo del color para transmitir información
- Search: Los resultados obtenidos han sido 5 errores que tienen que ser corregidos y 156 advertencias que hay que revisar. Por otra parte, no se han encontrado errores en el 85% de los elementos analizados. Entre los errores más relevantes se encuentran:

    - Campo de búsqueda sin etiqueta visible
    - Falta de visibilidad del foco del teclado
    - Elementos fuera de regiones landmark
    - Ausencia de roles o etiquetas adecuadas en componentes interactivos
- Details: Los resultados obtenidos muestran 11 errores que han de ser corregidos y 5 advertencias que necesitan revisión. El 95% de los elementos analizados no tienen errores, entre los errores más relevantes se encuentran:
    - Falta de visibilidad del foco del teclado
    - Uso exclusivo del color para transmitir información
    - Contenido significativo en pseudoelementos
    - Estructura semántica mejorable

El análisis de accesibilidad revela que la mayoría de elementos analizados en cada página cumple con los estándares aunque existan algunos aspectos que haya que corregir para garantizar una experiencia inclusiva. Abordar estos aspectos en el futuro mejorará significativamente la accesibilidad del sitio web y lo hará mas usable para personas con diferentes discapacidades.

5.2 Usabilidad

A continuación, se presenta el análisis de usabilidad siguiendo las 10 Heurísticas de Nielsen [10] utilizadas también para analizar los diferentes sitios web en el Estado del Arte.

1. **Visibilidad del estado del sistema**: La aplicación gestiona el estado de las acciones mediante mensajes y cambios visuales en los botones, en algunos casos la falta de foco visible en los botones puede dificultar la percepción del estado para usuarios que navegan con el teclado.
2. **Correspondencia entre el sistema y el mundo real**: Los textos e iconos utilizan un lenguaje claro que facilita la comprensión, los formularios y filtros emplean términos familiares en el sector turístico.
3. **Control y libertad del usuario**: El usuario puede cancelar acciones, volver atrás y modificar filtros o búsquedas de forma fácil. La ausencia de etiquetas visibles en algunos campos puede dificultar a usuarios con tecnología de asistencia.
4. **Consistencia y estándares**: La interfaz de la aplicación mantiene un estilo y estructura coherentes en todas las páginas utilizando componentes reutilizables.
5. **Prevención de errores**: Se han implementado comprobaciones en los formularios para evitar errores comunes a la hora de introducir datos.
6. **Reconocer antes que recordar**: La navegación y las acciones principales son fácilmente reconocibles gracias a menús, iconos y botones visibles. El uso de colores y un buen diseño ayuda a identificar las opciones.
7. **Flexibilidad y eficiencia de uso**: La aplicación permite realizar búsquedas rápidas y filtrar de forma eficiente, adaptándose tanto a nuevos usuarios como a usuarios experimentados. La navegación por teclado podría mejorarse para aumentar la eficiencia de usuarios avanzados o con discapacidad motriz.
8. **Estética y diseño minimalista**: El diseño de la aplicación es limpio y moderno, sin sobrecargar con mucha información y priorizando la claridad visual. La paleta de colores

es limitada y coherente, además los espacios en blanco se utilizan estratégicamente para separar secciones y facilitar la lectura.

9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores**: Los mensajes de error son claros cuando aparecen, aunque sería recomendable mejorar la accesibilidad de estos mensajes para que sean detectados por tecnologías de asistencia para mejorar la experiencia de usuarios que utilicen estas.
10. **Ayuda y documentación**: La aplicación incluye textos de ayuda y descripciones en los formularios, además de una breve guía de como utilizar la página web.

La aplicación cumple en gran medida con los estándares de accesibilidad y usabilidad, ofreciendo una experiencia clara y eficiente para la mayoría de usuario. Aunque se han detectado varios aspectos que hay que mejorar, el prototipo cuenta con una base sólida fácil de mejorar en futuros desarrollo. Corregir estos detalles permitirá que la aplicación sea aún más inclusiva y fácil de utilizar para todo el mundo.

6. Licencia Software y Documental

Copyright © 2025, Javier Erlés García

El presente documento y sus anexos se encuentran liberados bajo los términos de la licencia GFDL. Se permite la copia, distribución y/o modificación de los mismos bajo los términos de la GNU Free Documentation License, versión 1.3, o versiones posteriores publicadas por la Free Software Foundation.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation.

Para más información, puede consultarse: https://www.gnu.org/licenses/

Copyright © 2025, Javier Erlés García

Este programa es software libre: puede redistribuirlo y/o modificarlo bajo los términos de la GNU General Public License, publicada por la Free Software Foundation, ya sea la versión 3 de la Licencia, o (a su elección) cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea útil, pero SIN NINGUNA GARANTÍA; sin siquiera la garantía implícita de COMERCIALIZACIÓN o IDONEIDAD PARA UN FIN DETERMINADO. Véase la GNU General Public License para más detalles.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Para más información: https://www.gnu.org/licenses/

ESTE SOFTWARE ES SUMINISTRADO POR JAVIER ERLÉS GARCÍA Y CUALQUIER GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, PERO NO LIMITANDO A: LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN Y APTITUD PARA UN PROPÓSITO PARTICULAR, SON RECHAZADAS. EN NINGÚN CASO JAVIER ERLÉS GARCÍA O COLABORADORES SERÁN RESPONSABLES POR NINGÚN DAÑO DIRECTO, INDIRECTO, INCIDENTAL, ESPECIAL, EJEMPLAR O CONSECUENCIAL (INCLUYENDO, PERO NO LIMITADO A: LA ADQUISICIÓN O SUSTITUCIÓN DE BIENES O SERVICIOS, LA PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS; O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) O POR CUALQUIER TEORÍA DE RESPONSABILIDAD, YA SEA POR CONTRATO, RESPONSABILIDAD ESTRICTA O AGRAVIO (INCLUYENDO NEGLIGENCIA O CUALQUIER OTRA CAUSA) QUE SURJA DE CUALQUIER MANERA DEL USO DE ESTE SOFTWARE, INCLUSO SI SE HA ADVERTIDO DE LA POSIBILIDAD DE TALES DAÑOS.

7. Conclusiones y Trabajo Futuro

El desarrollo de este prototipo ha permitido cumplir todos los objetivos planteados al inicio del proyecto: diseñar e implementar una plataforma web orientada a Erasmus que facilite la búsqueda y comparación de alojamiento, haciendo uso exclusivo de tecnologías y herramientas de software libre.

Durante la realización del trabajo, se han aplicado los conocimientos que se han aprendido durante el grado, especialmente en áreas como el diseño web, la programación full-stack y la gestión de bases de datos. La elección de Nuxt 3 como framework para el frontend y Express.js como framework del backend ha resultado correcta, ofreciendo una arquitectura adecuada, modular y moderna. Además, se ha implementado una interfaz sencilla, responsive y funcional que es capaz de adaptarse a distintos dispositivos y necesidades del usuario.

El análisis de accesibilidad y usabilidad, así como la comparación de diferentes plataformas ya existentes, ha permitido identificar buenas prácticas y carencias comunes de este tipo de aplicaciones, influyendo de forma positiva en el diseño del prototipo que ha sido desarrollado.

Además de ser un ejercicio académico, este proyecto ha sido una experiencia enriquecedora que ha reforzado mis habilidades técnicas y gestión del tiempo, preparándome para futuros desafíos en el ámbito profesional.

El prototipo desarrollado cumple con las funcionalidades esenciales, existen diversas mejoras que se pueden abordar en el futuro:

- Desplegar ambas partes en un entorno de producción con un dominio propio y configuración de servidores.
- Integración con plataformas reales de alojamiento, para actualizar y obtener mediante APIs todo tipo de alojamientos de diferentes proveedores.
- Implementación de algoritmos para valorar los diferentes alojamientos con respecto los demás alojamientos y las búsquedas realizadas.
- Mejoras de accesibilidad siguiendo los estándares WCAG, con respecto a los errores detectados en los análisis anteriores.
- Testeo automatizado y pruebas de rendimiento para así garantizar una estabilidad del sistema a medida que crecen los usuarios y los datos.

Concluyendo, este proyecto sienta unas bases sólidas para construir una aplicación completa, profesional y adaptada a las necesidades de los estudiantes internacionales.

8. Referencias Bibliográficas

[1] Wikipedia [Internet]. Web framework. Disponible en: https://en.wikipedia.org/wiki/Web_framework

[2] Stack Overflow [Internet]. Most popular technologies. Disponible en: https://survey.stackoverflow.co/2024/technology#most-popular-technologies-webframe

[3] Wikipedia [Internet]. Node.js. Disponible en: https://es.wikipedia.org/wiki/Node.js

[4] React [Internet]. Learn. Disponible en: https://es.react.dev/learn

[5] Next.js [Internet]. Disponible en: https://nextjs.org/docs

[6] EBIS Business Techschool [Internet]. Todo sobre Express.js. Disponible en: https://www.ebiseducation.com/express-js-guia-completa

[7] Vue [Internet]. Guide. Disponible en: https://es.vuejs.org/v2/guide/

[8] Nuxt [Internet]. Docs. Disponible en: https://nuxt.com/

[9] IBM [Internet]. IBM Equal Access Toolkit. Disponible en: https://www.ibm.com/able/toolkit

[10] Nielsen, J. *10 Usability Heuristics for User Interface Design* [Internet]. Nielsen Norman Group. Disponible en: https://www.nngroup.com/articles/ten-usability-heuristics/

[11] Google Trends [Internet]. Disponible en: https://trends.google.es/

[12] IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998, vol., no., pp.1-40, 20 Oct. 1998

[13] ISO/IEC 25010:2011. *Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*. International Organization for Standardization.

[14] W3C WAI [Internet]. Sumario de WCAG 2. Dsiponible en: Sumario de WCAG 2 | Web Accessibility Initiative (WAI) | W3C

[15] W3C [Internet]. Disponible en: W3C

[16] Booch, G., Rumbaugh, J., Jacobson, I. *The Unified Modeling Language User Guide* (2nd ed.). Addison-Wesley, 2005.

[17] Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7ª ed.). Pearson.

[18] MySQL. (2023). *MySQL Workbench Manual*. Oracle Corporation. Disponible en: https://dev.mysql.com/doc/workbench/en/

9. Anexos

9.1 Mockups para pantallas grandes

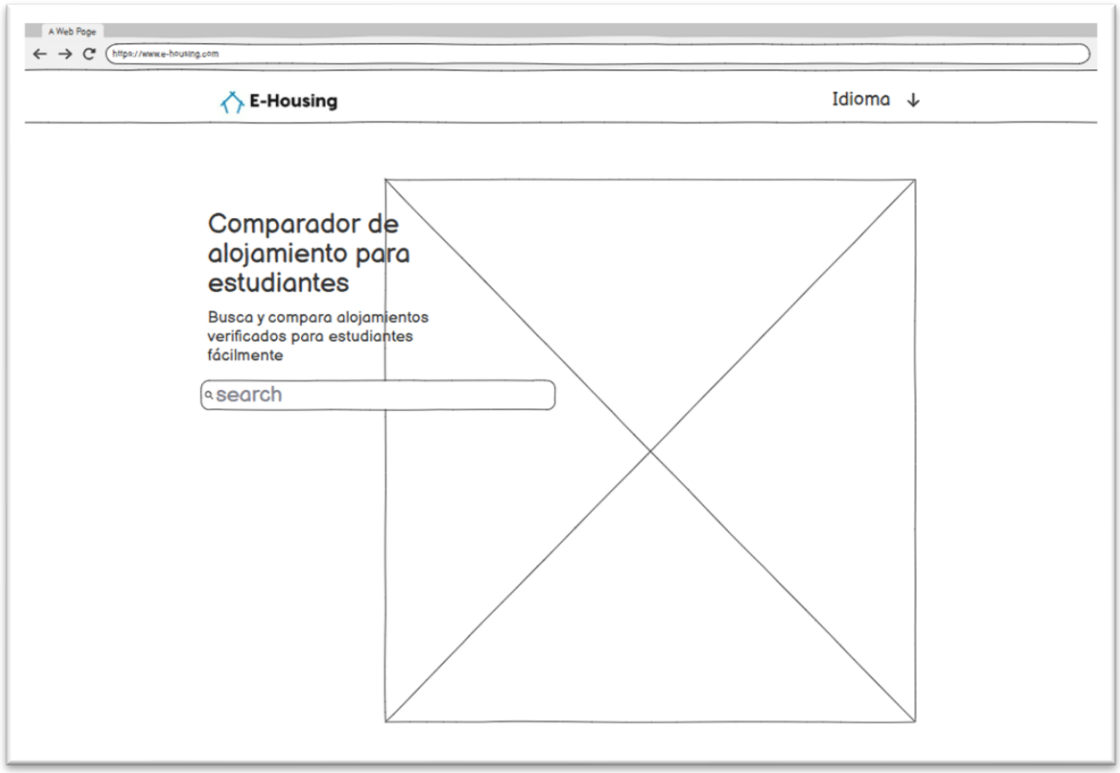


Ilustración 13 Página de inicio

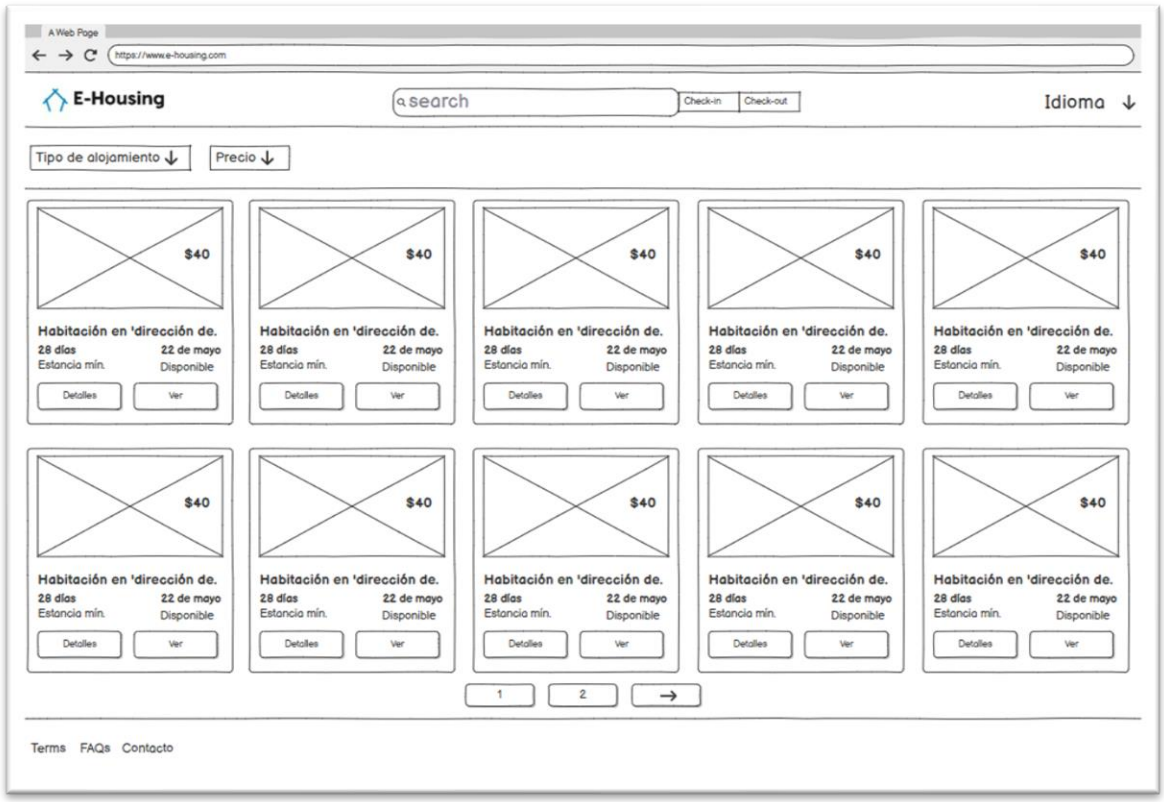


Ilustración 14 Página de búsqueda

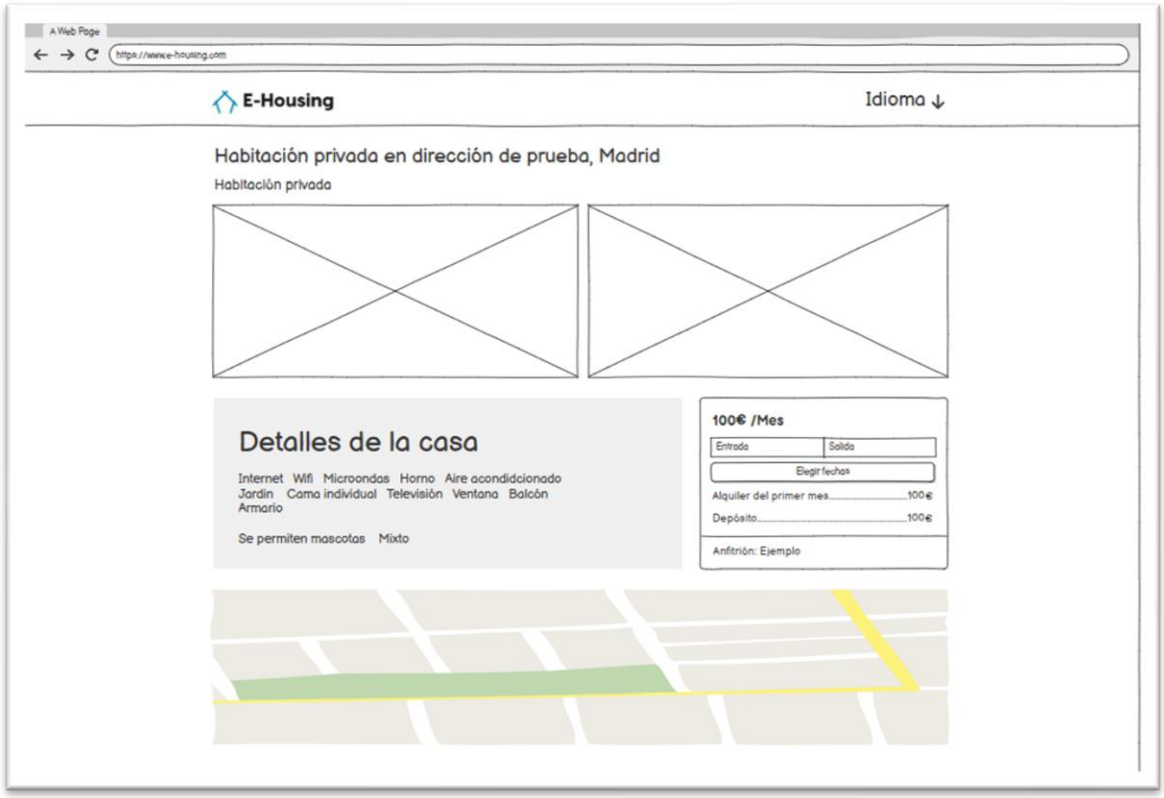


Ilustración 15 Página de detalles

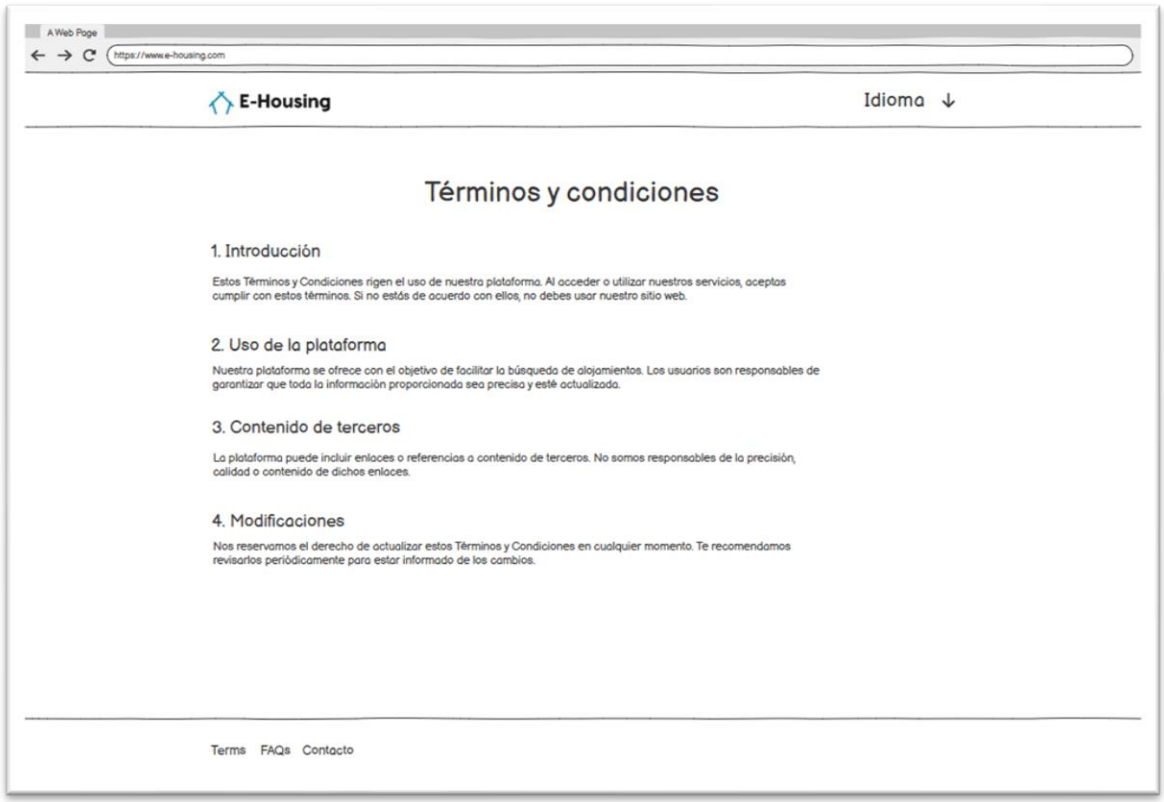


Ilustración 16 Página de términos y condiciones

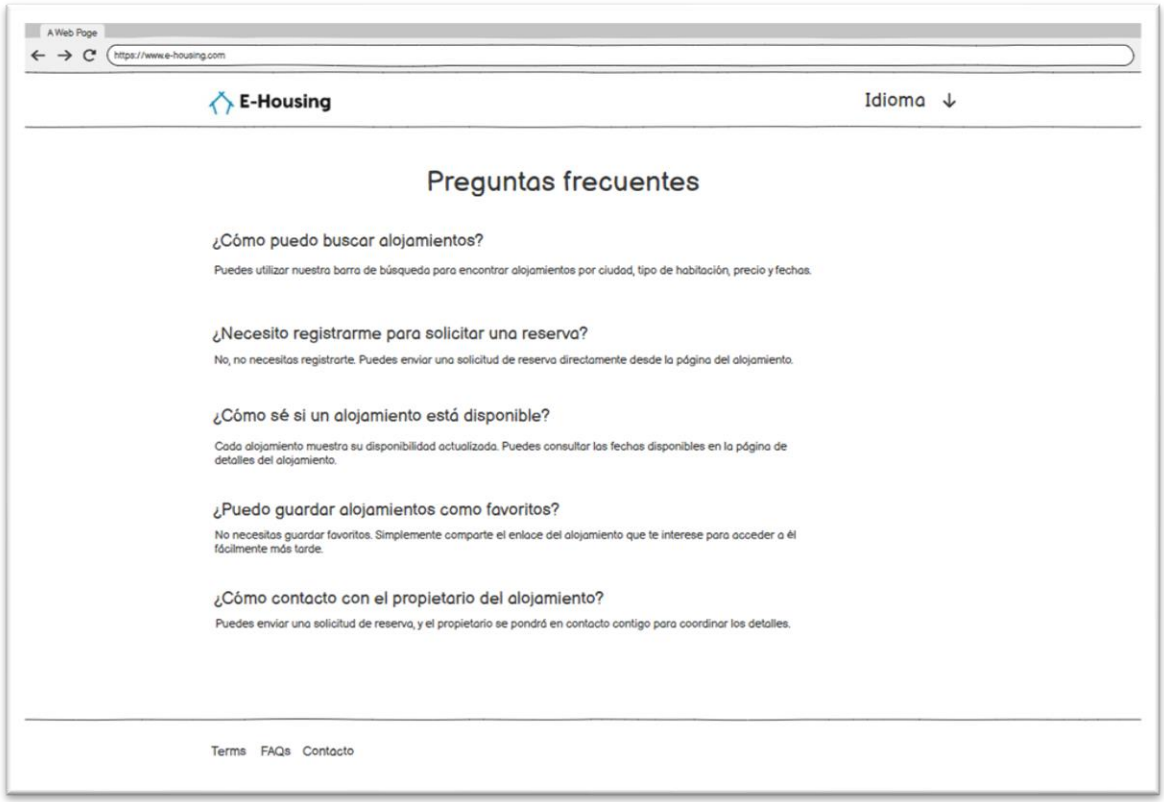


Ilustración 17 Página de FAQs

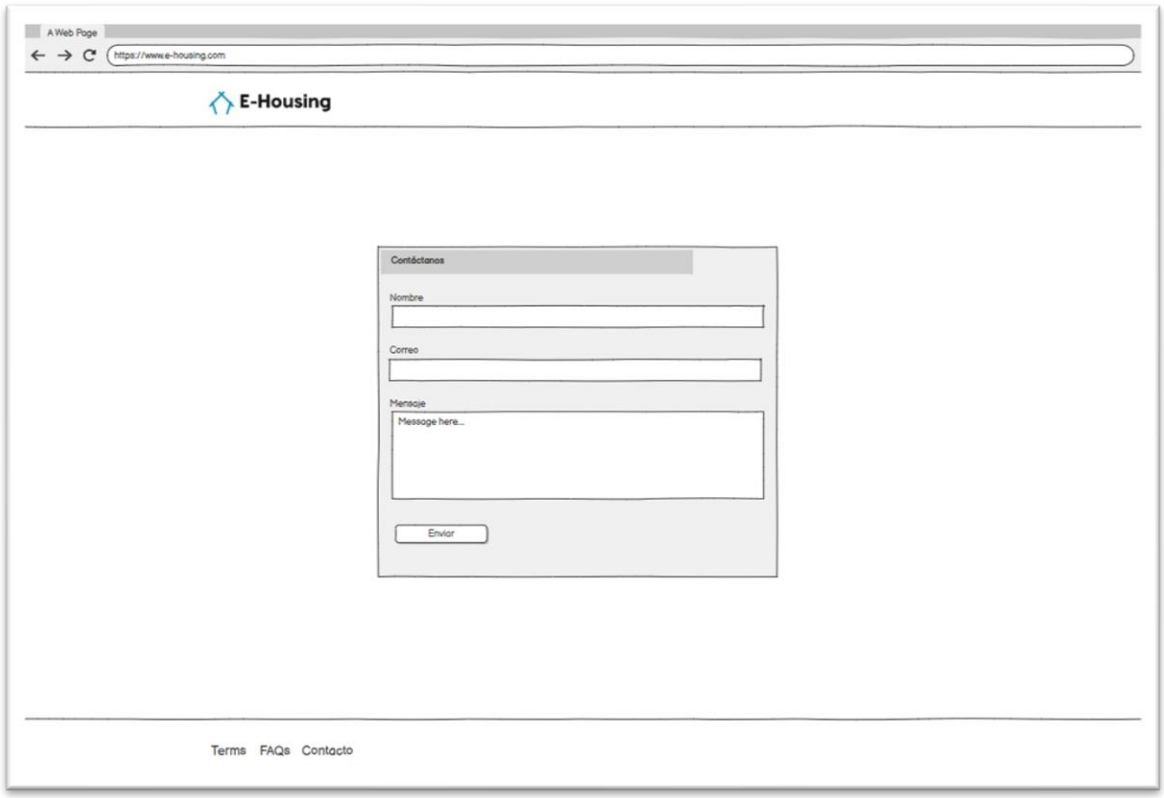


Ilustración 18 Página de contacto

9.2 Mockups para pantallas móvil

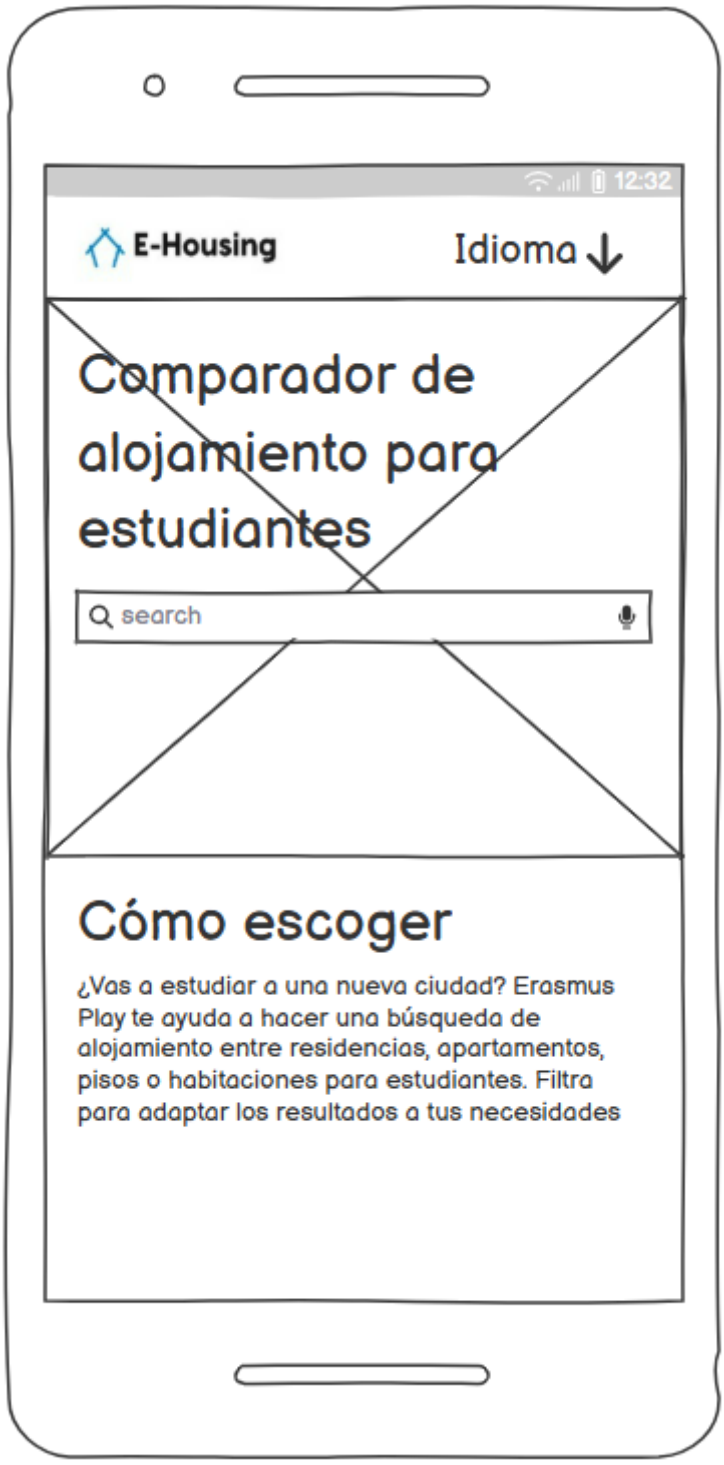


Ilustración 19 Página de inicio

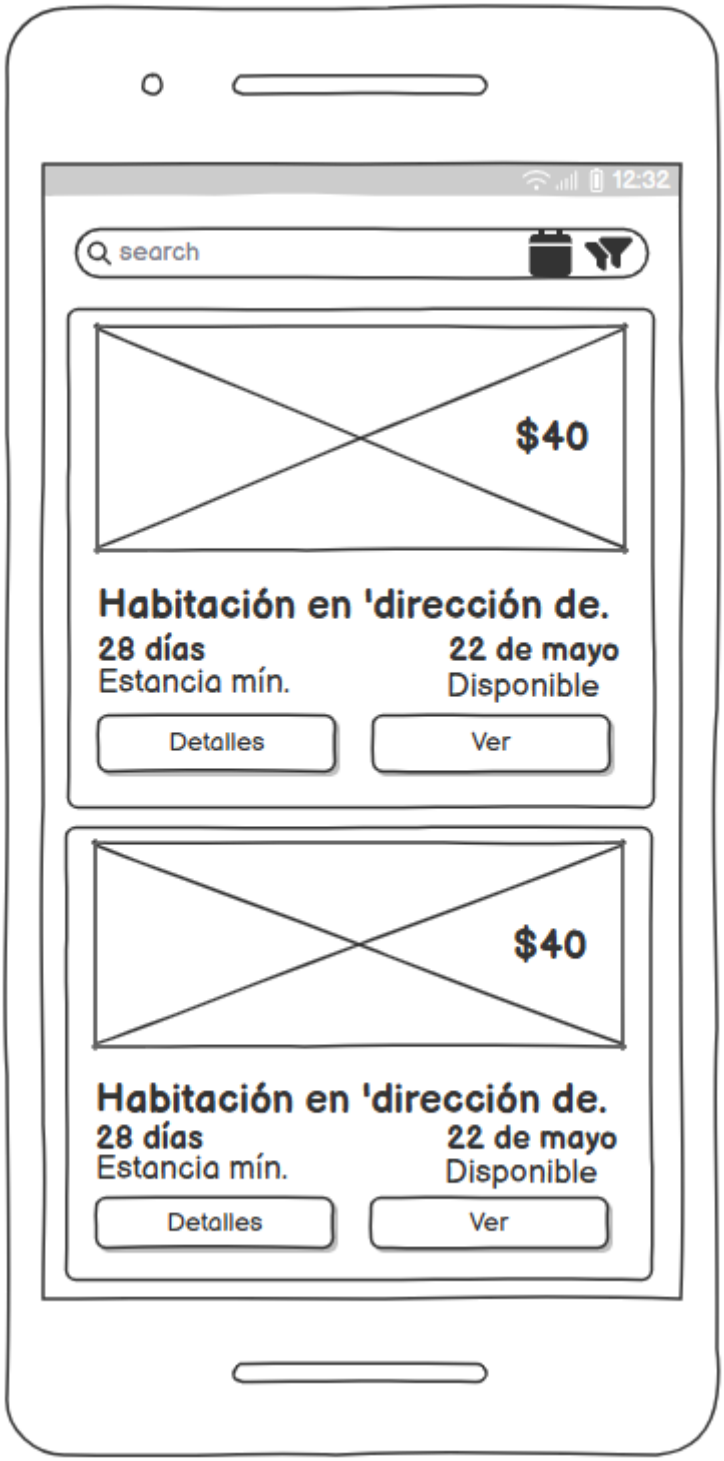


Ilustración 20 Página de búsqueda

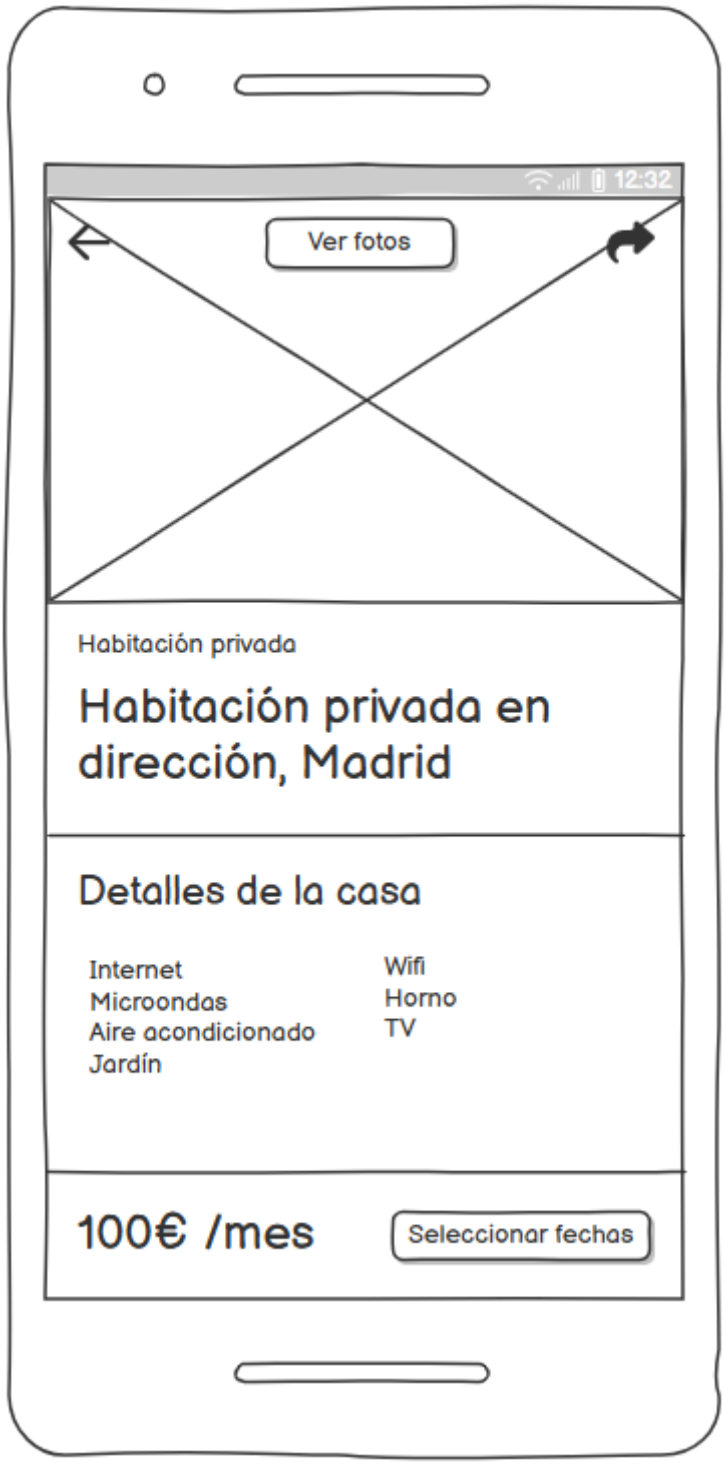


Ilustración 21 Página de detalles

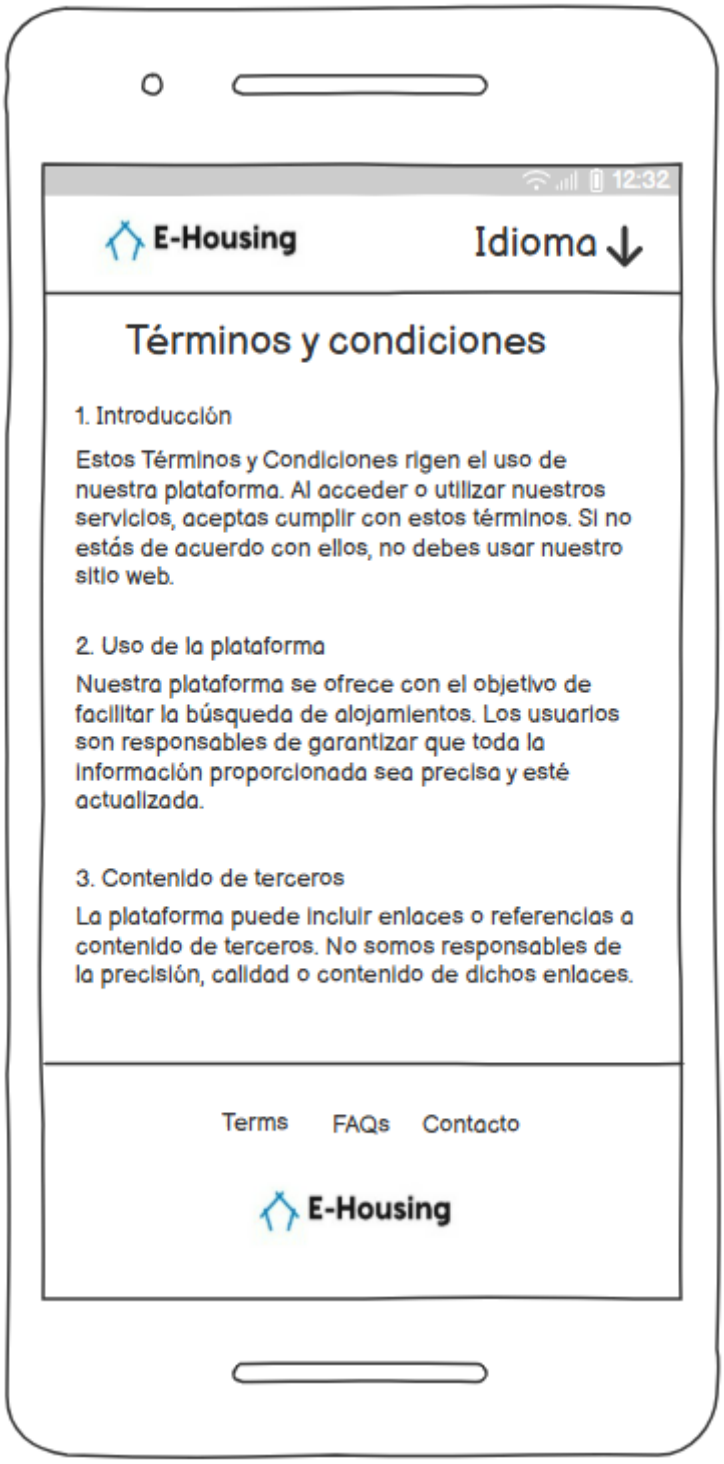


Ilustración 22 Página de términos y condiciones

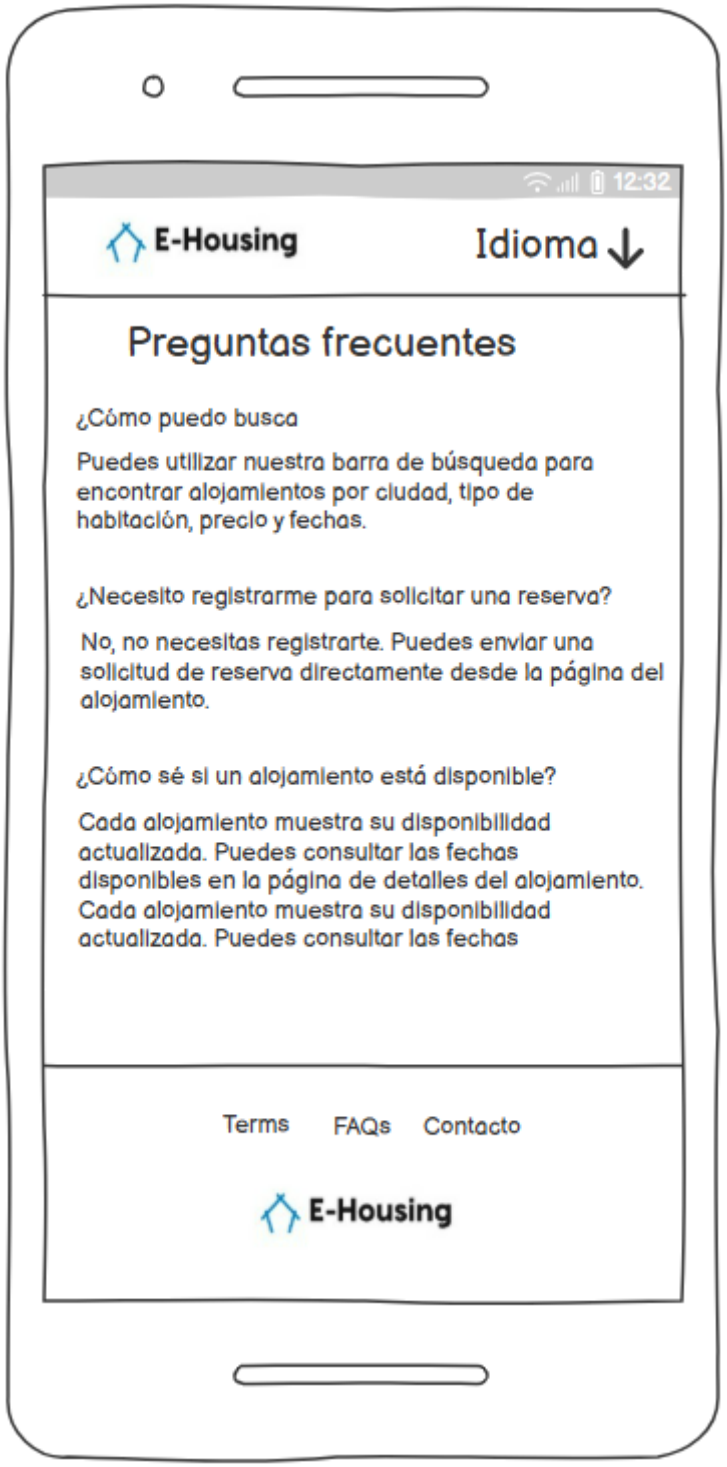


Ilustración 23 Página de FAQs

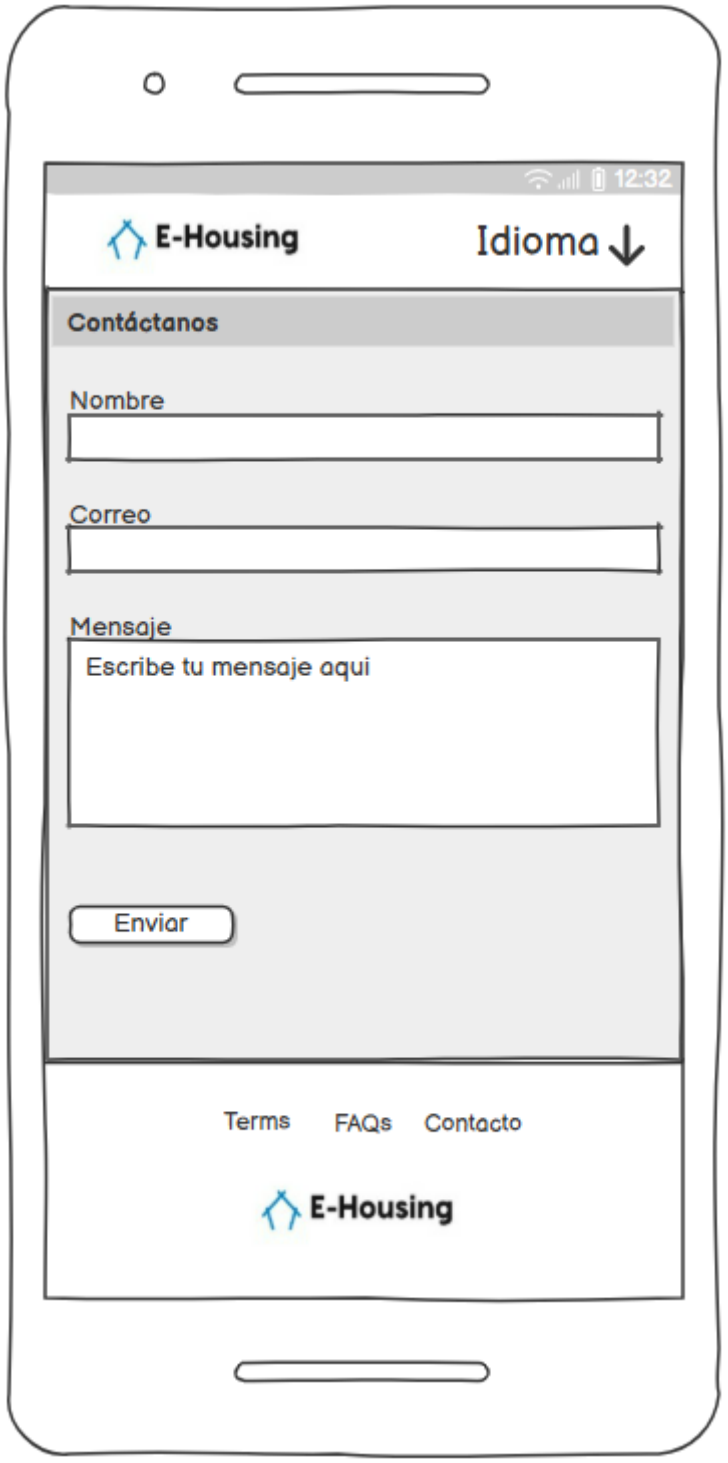


Ilustración 24 Página de contacto