



WeekendHouse

Plataforma web de gestión y reservas de casas
rurales

Autor

Daniel Osuna Molero



IES Francisco Ayala

2º Desarrollo de Aplicaciones Web

Granada, Junio de 2025

1. Abstract.....	4
2. Justificación.....	4
2.1 Características generales.....	4
2.2 Restricciones generales.....	5
2.3 Aspectos a cubrir, junto con los que no se van a tratar.....	5
2.3.1 Aspectos cubiertos.....	5
2.3.2 Aspectos no cubiertos.....	5
2.4 Estudio de las prestaciones de la herramienta frente a la competencia.....	5
3. Justificación de la tecnología empleada.....	6
3.1 Frontend.....	6
3.2 Backend.....	7
3.3 Base de datos.....	7
3.4 Almacenamiento y configuración.....	7
4. Requerimientos hardware y software.....	8
4.1 Requerimientos del cliente.....	8
4.1.1 Hardware.....	8
4.1.2 Software.....	8
4.2 Requerimientos del servidor (entorno de desarrollo y ejecución).....	8
4.2.1 Hardware.....	8
4.2.2 Software.....	8
4.3 Dependencias del proyecto.....	9
5. Análisis y diseño.....	9
5.1 Análisis general del sistema.....	9
5.2 Diagrama de casos de uso.....	10
5.3 Diagrama de clases.....	11
5.4 Mapa de navegación (estructura de la web).....	12
5.5 Base de datos.....	13
5.5.1 Modelo Entidad-Relación.....	13
5.5.2 Diseño lógico de tablas.....	13
6. Implementación.....	14
6.1 Frontend (Cliente).....	14
6.2 Backend (Servidor).....	15
6.3 Funcionalidades destacadas.....	15
7. Evaluación y prueba.....	16
7.1 Pruebas de la aplicación web.....	16
7.1.1 Validación de formularios.....	16
7.1.2 Retroalimentación al usuario.....	16
7.1.3 Pruebas de navegación.....	17
7.2 Pruebas de la base de datos.....	17

7.2.1 Integridad de los datos.....	17
7.2.2 Pruebas con ejemplos reales.....	17
7.3 Batería de ejemplos.....	17
8. Manual de estilos.....	18
8.1 Sketches.....	18
8.1.1 Inicio.....	19
8.1.2 Perfil Casa Rural.....	20
8.1.3 Blog.....	21
8.1.4 Gestión Propietarios.....	22
8.1.5 Pasarela de Pago.....	23
8.2 Criterios de accesibilidad.....	24
8.3 Criterios de usabilidad.....	24
8.4 Tipografía.....	24
8.5 Mapa de colores del proyecto.....	24
8.6 Dispositivos / vistas para las que se ha diseñado el proyecto.....	25
9. Software utilizado.....	25
9.1 Editor de código y entorno de desarrollo.....	25
9.2 Frontend.....	26
9.3 Backend.....	26
9.4 Control de versiones.....	26
9.5 Documentación y diseño.....	26
9.6 Navegadores para pruebas.....	27
10. Mejoras posibles y aportaciones.....	27
10.1 Mejoras en funcionalidad.....	27
10.2 Mejoras en la organización del código.....	28
10.3 Mejoras en la organización de la información.....	28
10.4 Ideas futuras / aportaciones.....	28
11. Bibliografía.....	28

1. Abstract

WeekendHouse es una plataforma web diseñada para facilitar la gestión y reserva de casas rurales. El proyecto permite a los propietarios gestionar sus alojamientos y a los usuarios realizar reservas de forma sencilla. Entre las principales funcionalidades se incluyen: registro e inicio de sesión de usuarios y propietarios, gestión completa de las casas (creación, edición, eliminación de anuncios), gestión de reservas por parte de los propietarios, subida y visualización de imágenes, valoraciones públicas de las casas por parte de los usuarios, y perfil personalizado para cada usuario. El desarrollo de la aplicación se ha realizado utilizando tecnologías de desarrollo web, con Vue.js para el frontend y PHP con MySQL para el backend. El proyecto busca ofrecer una experiencia de usuario intuitiva y agradable tanto para ordenadores como para dispositivos móviles.

WeekendHouse is a web platform designed to simplify the management and booking of rural houses. The project allows property owners to manage their accommodations and users to easily make bookings. The main features include: user and owner registration and login, full management of properties (creation, editing, deletion of listings), booking management by property owners, photo uploading and viewing, public reviews of properties by users, and personalized user profiles. The application has been developed using modern web technologies, with Vue.js for the frontend and PHP with MySQL for the backend. The project aims to provide an attractive, fast, and pleasant user experience on both desktop and mobile devices.

2. Justificación

2.1 Características generales

Como está definido en el Abstract WeekendHouse es una plataforma web que ofrece las siguientes características:

- Buscador por nombre y listado de casas rurales.
- Ficha detallada de cada casa con fotos, servicios y precio.
- Sistema de reservas online con calendario.
- Pasarela de pago simulada.
- Sistema de valoración pública por parte de los usuarios tras la estancia.
- Panel de gestión para propietarios (añadir, editar y eliminar casas; gestionar reservas).
- Perfil personalizado para los usuarios (historial de reservas, gestión de foto de perfil, valoraciones realizadas).
- Diseño responsive para ordenador y para móvil.

2.2 Restricciones generales

- No se ha implementado pasarela de pago real. La actual pasarela es una simulación.
- No se han integrado APIs de terceros como Google Maps.
- No se ha implementado un sistema de notificaciones en tiempo real ni un sistema de alertas.
- La seguridad se ha trabajado a nivel básico: protección de contraseñas, validación de formularios.
- No se ha desarrollado una App móvil nativa; sí un diseño responsive que permite el uso desde móviles.

2.3 Aspectos a cubrir, junto con los que no se van a tratar

2.3.1 Aspectos cubiertos

- Registro e inicio de sesión para usuarios y propietarios.
- Gestión completa de casas por parte del propietario.
- Sistema de reservas funcional con gestión de estados (pendiente, confirmada, cancelada).
- Subida y gestión de imágenes de las casas.
- Valoraciones públicas de las casas.
- Perfil de usuario con personalización de foto de perfil y visualización de reservas y valoraciones.

2.3.2 Aspectos no cubiertos

- Integración con redes sociales.
- Inteligencia artificial o Chatbot.
- Notificaciones push o sistema de alertas.
- Gestión multizona (por el momento se trabaja sobre la zona Sur de Córdoba).
- Sistema de ranking o recomendaciones automáticas.
- Sistema avanzado de estadísticas para propietarios.

2.4 Estudio de las prestaciones de la herramienta frente a la competencia

Actualmente existen soluciones muy completas en el ámbito del alquiler turístico de casas rurales, como:

- **Airbnb**: plataforma global, interfaz muy avanzada, alta integración de sistemas de pago y comunicación. Sin embargo, no está especializada en el turismo rural local ni ofrece una

plataforma personalizada para un área concreta como sí pretende WeekendHouse.

- **EscapadaRural**: plataforma especializada en turismo rural en España. Ofrece una plataforma completa de gestión para propietarios y un potente buscador. Sin embargo, suele tener costes de publicación para los propietarios, y no siempre permite una personalización completa de la experiencia de usuario.
- **WeekendHouse** aporta un enfoque local y personalizable, pensado como herramienta gratuita para propietarios de la zona que no desean depender de grandes plataformas de pago. Su interfaz está simplificada y orientada a la experiencia de usuario directa. Además, permite que cada propietario controle de manera completa sus propiedades y sus reservas.

3. Justificación de la tecnología empleada

3.1 Frontend

Para la parte del cliente se ha utilizado Vue.js (versión 3), un framework progresivo de JavaScript que permite construir interfaces reactivas y modulares con facilidad. Se eligió Vue por las siguientes razones:

- Permite una arquitectura por componentes, ideal para dividir la plataforma en secciones reutilizables como LoginModal, vPerfil, vReservar, etc.
- Su sintaxis es sencilla y declarativa, lo que facilita el desarrollo y mantenimiento del código.
- Posee una amplia comunidad, buena documentación y un ecosistema estable con herramientas como Vue CLI, Vuex o Vue Router.
- Facilita la creación de interfaces responsive y reactivas, adaptables a dispositivos móviles.

Además, se ha usado:

- **V-Calendar**: para el componente de selección de fechas en la reserva.
- **Vuex**: para la gestión global del estado de usuario autenticado.
- **CSS personalizado**: con estilos adaptados a un diseño moderno, limpio y coherente con la temática rural.
- **Vue Router** : control de rutas y navegación entre vistas.

3.2 Backend

Para la parte del servidor se ha utilizado PHP sin framework, por los siguientes motivos:

- Es un lenguaje ampliamente utilizado en desarrollo web y compatible con servidores locales como XAMPP
- Permite una curva de aprendizaje baja y una integración directa con MySQL sin necesidad de capas intermedias complejas.
- Ha permitido crear un backend funcional con rutas como registro.php, login.php, crearCasa.php, reserva.php, etc., simplificando el desarrollo.

Los endpoints PHP gestionan peticiones HTTP (GET, POST, PUT) y devuelven respuestas en formato JSON para ser consumidas por el frontend Vue.

3.3 Base de datos

Se ha utilizado **MySQL**, una base de datos relacional de código abierto muy extendida en proyectos web. Las razones para su elección son:

- Buena integración con PHP mediante extensiones como mysqli.
- Soporte para claves foráneas, integridad referencial y tipos de datos estructurados.
- Facilidad para gestionar mediante herramientas como **phpMyAdmin**.

La estructura de la base de datos incluye tablas como: usuario, casa, reserva y valoración con relaciones bien definidas entre ellas

3.4 Almacenamiento y configuración

- **XAMPP** se ha utilizado como entorno de desarrollo local para simular un servidor completo con Apache, PHP y MySQL.
- Las imágenes se almacenan en carpetas públicas accesibles desde el frontend:
 - /fotos/ para casas
 - /fotos_perfil/ para usuarios
- Se ha definido un archivo config.js en el frontend para gestionar de forma centralizada las rutas base del backend (API_BASE) e imágenes (IMG_BASE, IMG_PERFIL_BASE), lo que ha facilitado la transición de entorno local a posibles despliegues externos.

4. Requerimientos hardware y software

4.1 Requerimientos del cliente

Para acceder y utilizar la plataforma WeekendHouse como usuario final, se requiere un dispositivo con las siguientes características mínimas:

4.1.1 Hardware

- Procesador de al menos 1 GHz
- 2 GB de memoria RAM
- Resolución de pantalla mínima: 1024×600 px (diseño responsive para móviles)
- Conexión a Internet.

4.1.2 Software

- Navegador web actualizado compatible con ES6 y soporte para JavaScript moderno
- Recomendados: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.
- Soporte para cookies y almacenamiento local (localStorage) para mantener la sesión

4.2 Requerimientos del servidor (entorno de desarrollo y ejecución)

Para ejecutar el backend y la base de datos, se ha utilizado un servidor local mediante XAMPP, que incluye Apache, PHP y MySQL. Los requisitos son:

4.2.1 Hardware

- Procesador multinúcleo (recomendado: 4 núcleos)
- 4 GB de RAM
- Al menos 1 GB de espacio en disco para alojar la base de datos, imágenes y scripts PHP

4.2.2 Software

- XAMPP o stack equivalente con:
 - Apache HTTP Server (v2.4.x)
 - PHP (≥ 7.4)
 - MySQL (≥ 5.7)
- Sistema operativo compatible: Windows 10, macOS o cualquier distribución Linux moderna

Aunque el desarrollo se ha hecho localmente con XAMPP, la web ha sido hosteada con ayuda de Infinity Free en un portal web para que sea accesible globalmente en la dirección:

<https://weekendhouse.great-site.net/>

4.3 Dependencias del proyecto

WeekendHouse se apoya en un conjunto de tecnologías y herramientas modernas tanto para el desarrollo frontend como backend. En el lado del cliente se ha utilizado Vue 3 junto con bibliotecas auxiliares como *Vuex* (para el estado global), *Vue Router* (para el enrutado de vistas), *V-Calendar* (para selección de fechas) y *Vue Toastification* (para notificaciones visuales). Además, Axios y Fetch permiten la comunicación con la API.

En el servidor, el backend está implementado en PHP sin frameworks externos, utilizando la librería *mysqli* para conectar con MySQL y organizar las funcionalidades en scripts independientes dentro de la carpeta */api*.

Para el desarrollo y la gestión, se han empleado herramientas como Visual Studio Code como editor principal, phpMyAdmin para administrar la base de datos y Chrome DevTools para la depuración frontend.

5. Análisis y diseño

5.1 Análisis general del sistema

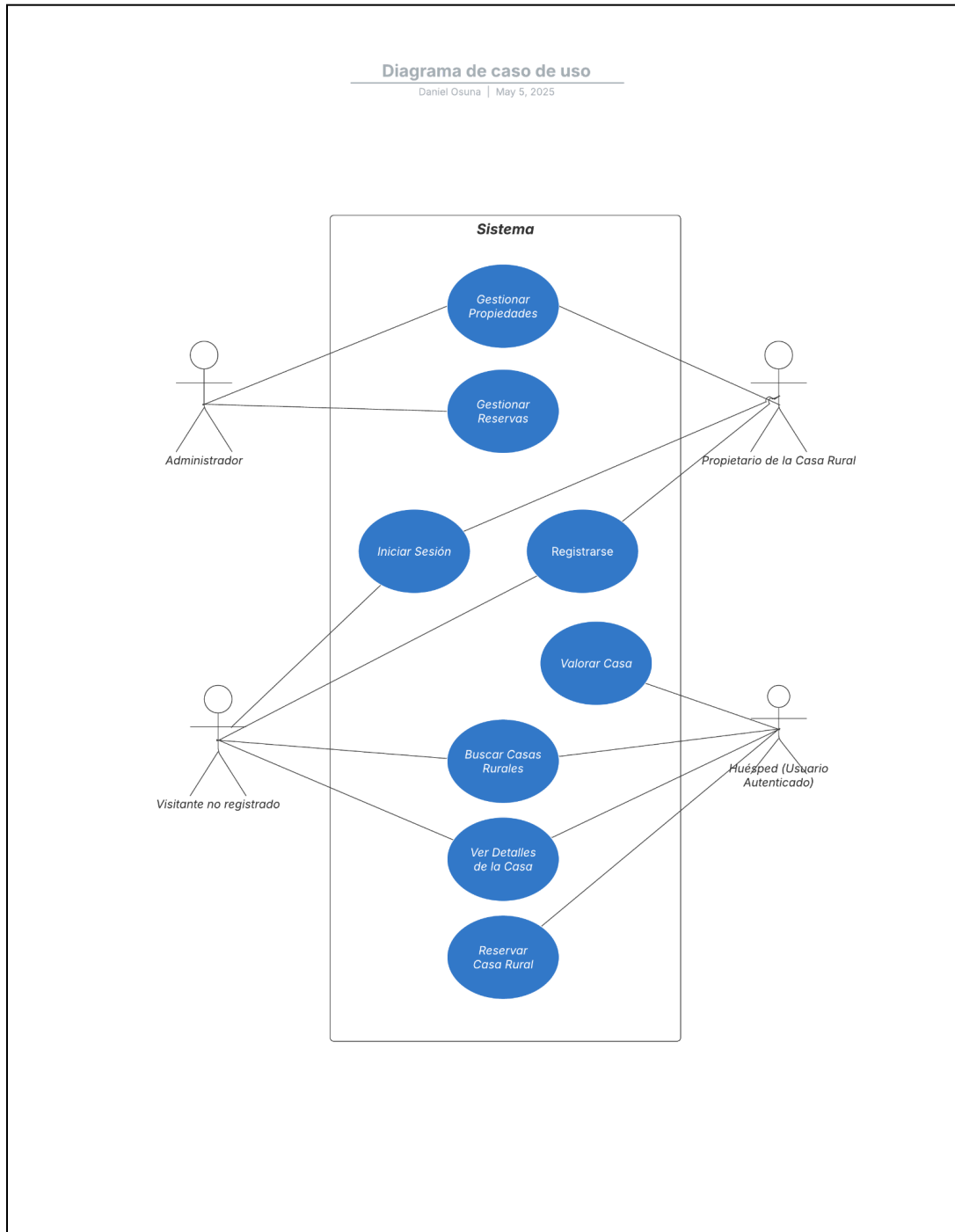
El sistema WeekendHouse es una plataforma web orientada a la gestión y reserva de casas rurales. Está pensado para satisfacer dos perfiles principales de usuario:

- **Usuarios generales / huéspedes:** usuarios que navegan por la plataforma, buscan casas, hacen reservas y publican valoraciones.
- **Propietarios:** usuarios registrados que publican sus casas rurales, gestionan sus anuncios y controlan las reservas recibidas.

Además, se contempla un rol de administrador que podría tener acceso a tareas de control, aunque en esta versión inicial el foco se centra en propietarios y huéspedes.

5.2 Diagrama de casos de uso

El siguiente diagrama muestra los principales casos de uso contemplados en el sistema:

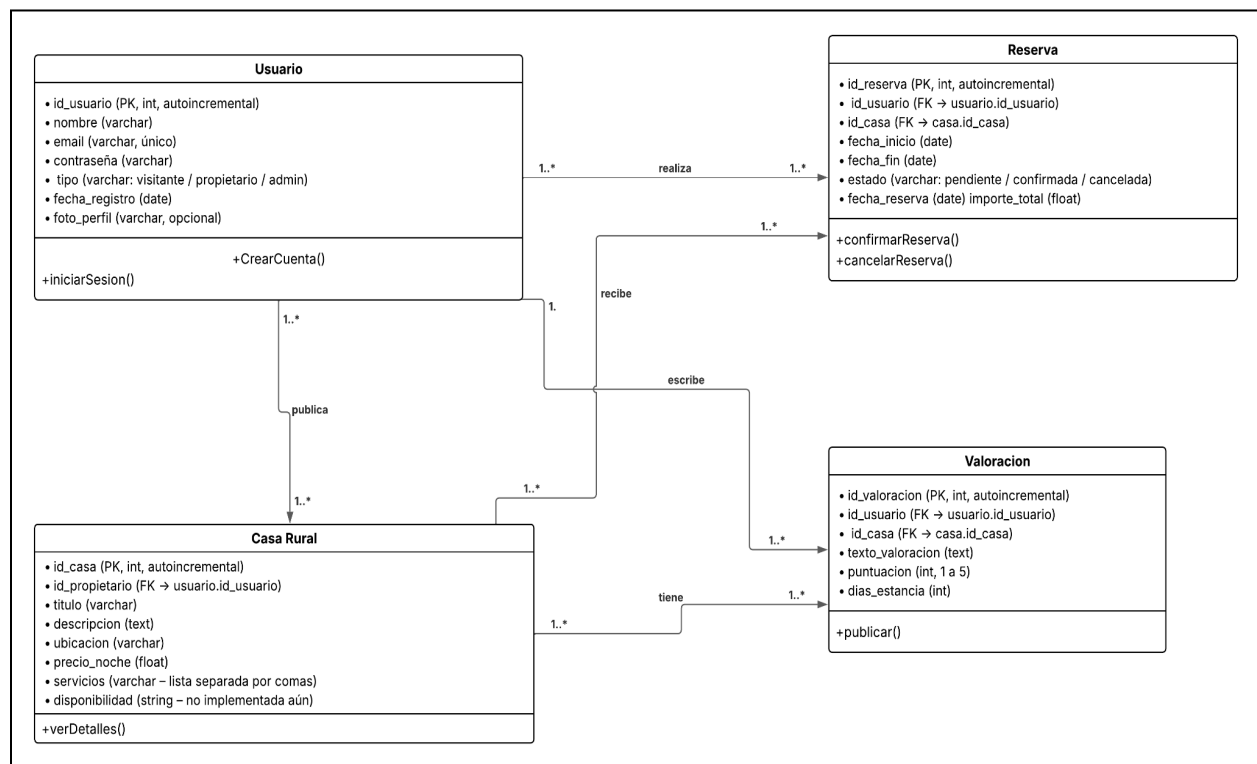


Los principales casos de uso son:

- Registro de nuevos usuarios (visitante no registrado)
- Inicio de sesión
- Gestión de perfil de usuario
- Visualización de casas rurales
- Valoración de casas rurales
- Reserva de casas rurales
- Gestión de propiedades : Edición y creación (propietarios)
- Gestión de reservas : Confirmación o cancelación (propietarios)

5.3 Diagrama de clases

El siguiente diagrama de clases muestra las entidades principales del sistema y sus relaciones:



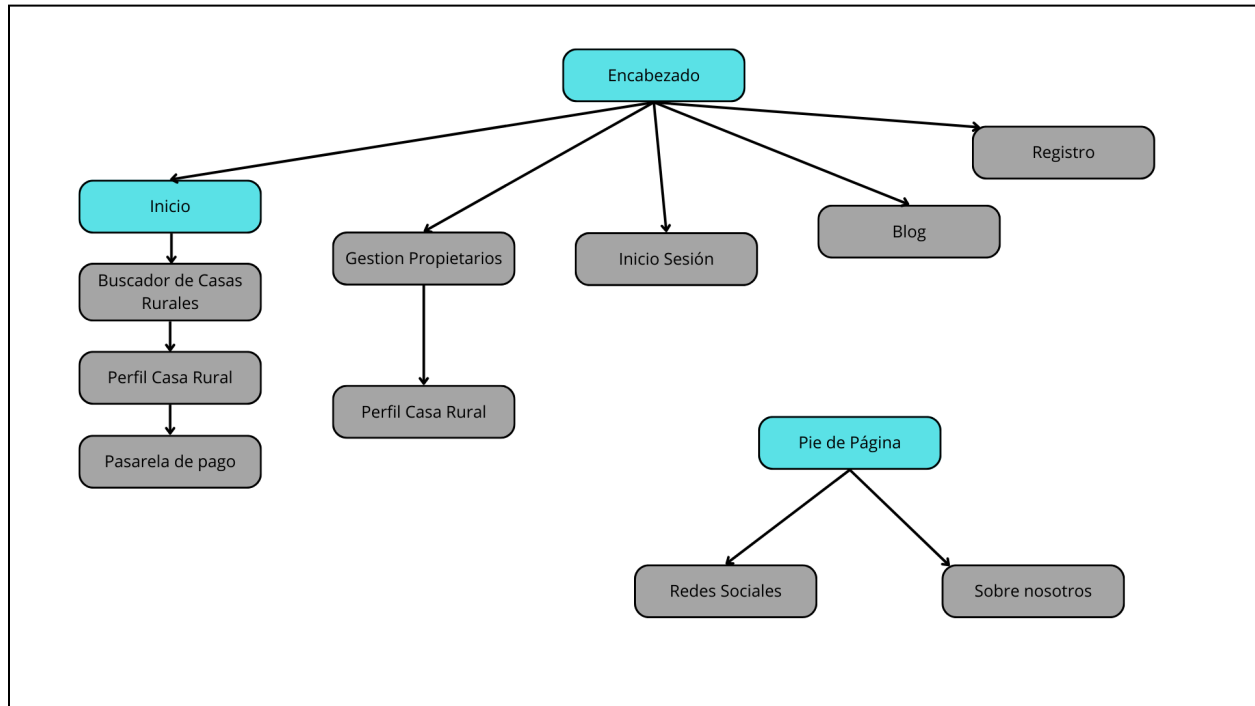
Entidades principales:

- **Usuario**: representa tanto al huésped como al propietario. Almacena datos básicos y tipo de usuario.
- **Casa Rural**: entidad que describe una casa rural, sus atributos y su relación con el propietario.

- **Reserva:** representa la reserva realizada por un huésped.
- **Valoración:** permite que los huéspedes valoren las casas que han reservado.

5.4 Mapa de navegación (estructura de la web)

A nivel de interfaz y experiencia de usuario, la aplicación se ha estructurado de la siguiente forma:



La navegación está pensada para que:

- Los visitantes no registrados puedan explorar el catálogo de casas y registrarse o iniciar sesión.
- Los usuarios autenticados tengan acceso adicional a su perfil y reservas.
- Los propietarios tengan acceso adicional a la gestión de sus propiedades y reservas.

5.5 Base de datos

La base de datos empleada es relacional (MySQL).

5.5.1 Modelo Entidad-Relación

El modelo de base de datos refleja directamente el diagrama de clases anterior. Las principales tablas son:

- **usuario**: guarda los datos de los usuarios.
- **casa**: almacena la información de las casas rurales.
- **reserva**: almacena las reservas hechas por los usuarios.
- **valoración**: almacena las valoraciones de las casas.

5.5.2 Diseño lógico de tablas

Tabla	Atributos
usuario	<ul style="list-style-type: none"> ● id_usuario (PK, int, autoincremental) ● nombre (varchar) ● email (varchar, único) ● contraseña (varchar) ● tipo (varchar: visitante / propietario / admin) ● fecha_registro (date) ● foto_perfil (varchar, opcional)
casa	<ul style="list-style-type: none"> ● id_casa (PK, int, autoincremental) ● id_propietario (FK → usuario.id_usuario) ● titulo (varchar) ● descripcion (text) ● ubicacion (varchar) ● precio_noche (float) ● servicios (varchar – lista separada por comas) ● disponibilidad (string – no implementada aún)
reserva	<ul style="list-style-type: none"> ● id_reserva (PK, int, autoincremental) ● id_usuario (FK → usuario.id_usuario) ● id_casa (FK → casa.id_casa) ● fecha_inicio (date) ● fecha_fin (date) ● estado (varchar: pendiente / confirmada / cancelada) ● fecha_reserva (date) importe_total (float)
valoración	<ul style="list-style-type: none"> ● id_valoracion (PK, int, autoincremental) ● id_usuario (FK → usuario.id_usuario) ● id_casa (FK → casa.id_casa) ● texto_valoracion (text) ● puntuacion (int, 1 a 5) ● dias_estancia (int)

6. Implementación

El desarrollo de WeekendHouse se ha llevado a cabo mediante una arquitectura cliente-servidor, integrando herramientas web modernas

6.1 Frontend (Cliente)

La interfaz del usuario se ha construido como una Single Page Application (SPA) utilizando Vue.js (v3.5.16). Todo el frontend se organiza en componentes modulares reutilizables que permiten mantener una estructura clara y escalable.

- **Composición:** Cada sección de la plataforma (inicio, perfil, gestión, reserva...) se ha implementado como un componente Vue independiente. Los elementos comunes como cabecera y pie se agrupan en componentes globales (appHeader.vue, appFooter.vue) reutilizables en todas las vistas.
- **Estilos:** Los estilos se han programado en cada componente con scoped CSS, incorporando gradientes, transiciones suaves y una paleta visual coherente.
- **Interactividad:** Se han integrado librerías como vue-toastification para notificaciones personalizadas y formularios controlados mediante v-model, con validación en cliente usando atributos HTML5 (required, pattern) y scripts en JavaScript.
- **Configuración dinámica:** Se ha centralizado la gestión de rutas y recursos mediante un archivo config.js, que define constantes para la API (API_BASE) y los caminos de imágenes (IMG_BASE, IMG_PERFIL_BASE), lo que facilita el despliegue en diferentes entornos.

6.2 Backend (Servidor)

El backend está programado en PHP (versión 8.x) sin framework, con una API REST que gestiona todas las operaciones de la plataforma y responde en formato JSON.

- **Gestión de peticiones:** Cada funcionalidad está implementada como un script PHP en la carpeta /api/, que responde a peticiones HTTP (GET, POST, PUT, DELETE).

- **Base de datos:** Se ha utilizado PDO para conectarse a MySQL mediante una conexión centralizada (conexion.php). Todas las consultas se ejecutan con prepared statements para prevenir inyecciones SQL.
- **Seguridad:** Las contraseñas se almacenan cifradas mediante password_hash. Se validan y sanitizan todas las entradas del usuario antes de ser procesadas. Además, se controla el tipo y formato de las imágenes subidas, y se verifica el estado de sesión mediante localStorage en el cliente.

6.3 Funcionalidades destacadas

- **Formularios:** Todos los formularios están integrados con Vue mediante v-model y validaciones duales (frontend y backend). Se proporciona retroalimentación al usuario mediante alertas visuales.
- **Comunicación cliente-servidor:** El frontend realiza las llamadas a la API mediante fetch y axios, según el contexto. Se gestionan respuestas y errores usando promesas (.then/.catch) y bloques try/catch.
- **Plantillas y vistas:** La estructura visual se basa en componentes dinámicos como valoracionCard.vue (para mostrar valoraciones) o plantillas específicas para mostrar casas y reservas.

7. Evaluación y prueba

El proceso de evaluación y pruebas de la plataforma *WeekendHouse* se ha realizado de manera exhaustiva tanto a nivel funcional como a nivel visual, con el objetivo de garantizar que la experiencia de uso sea fluida, coherente y libre de errores. Se ha testado tanto la aplicación web como la base de datos.

7.1 Pruebas de la aplicación web

7.1.1 Validación de formularios

Se han implementado controles de validación tanto en el cliente (HTML5 + JavaScript) como en el servidor, especialmente en los formularios críticos como:

Formulario	Validaciones principales
Registro de usuario	Email en formato válido, contraseña coincidente, campos obligatorios
Inicio de sesión	Verificación de existencia y coincidencia de credenciales
Creación de casa	Título obligatorio, precio válido, servicios y ubicación definidos
Proceso de reserva	Fechas válidas, mínimo de días de estancia, número de teléfono con patrón de 9 dígitos

7.1.2 Retroalimentación al usuario

La interacción con el usuario se ha enriquecido mediante notificaciones visuales gracias a vue-toastification. Estas alertas informan en tiempo real del resultado de las acciones más comunes:

- Registro exitoso o con error
- Inicio de sesión incorrecto
- Reserva realizada correctamente
- Valoración enviada
- Alta y edición de casas
- Actualización de foto de perfil

7.1.3 Pruebas de navegación

Se han verificado todos los flujos posibles de navegación entre vistas, asegurando:

- La inexistencia de enlaces rotos o inaccesibles.
- El comportamiento correcto tanto en la navegación pública (usuarios no autenticados) como privada (usuarios registrados y propietarios).

Estas pruebas garantizan que los usuarios puedan moverse por la aplicación sin obstáculos ni errores de ruta.

7.2 Pruebas de la base de datos

7.2.1 Integridad de los datos

Se ha comprobado que las inserciones y actualizaciones en las tablas usuario, casa, reserva y valoración respetan las restricciones de integridad:

- Claves primarias auto-incrementadas.

- Claves foráneas correctas en las reservas y valoraciones.
- Eliminar una casa implica eliminar sus imágenes y sus valoraciones asociadas correctamente.

7.2.2 Pruebas con ejemplos reales

- Se han insertado usuarios de prueba y propietarios.
- Se han creado casas reales con imágenes.
- Se han realizado reservas desde el flujo de pago.
- Se han insertado valoraciones y visualizado tanto en la vista de la casa como en el blog.

7.3 Batería de ejemplos

La batería de ejemplos de test incluye casos como:

Escenario	Resultado esperado
Registro con datos válidos	Usuario creado y logueado automáticamente
Login con usuario existente	Login correcto, redirección a inicio
Crear casa rural con fotos	Casa visible en la galería y en gestión
Realizar reserva	Reserva creada, visible para el propietario
Valorar una casa que se ha reservado	Valoración visible tanto en la casa como en el blog
Cancelar reserva por el propietario	Reserva eliminada correctamente de la BD

8. Manual de estilos

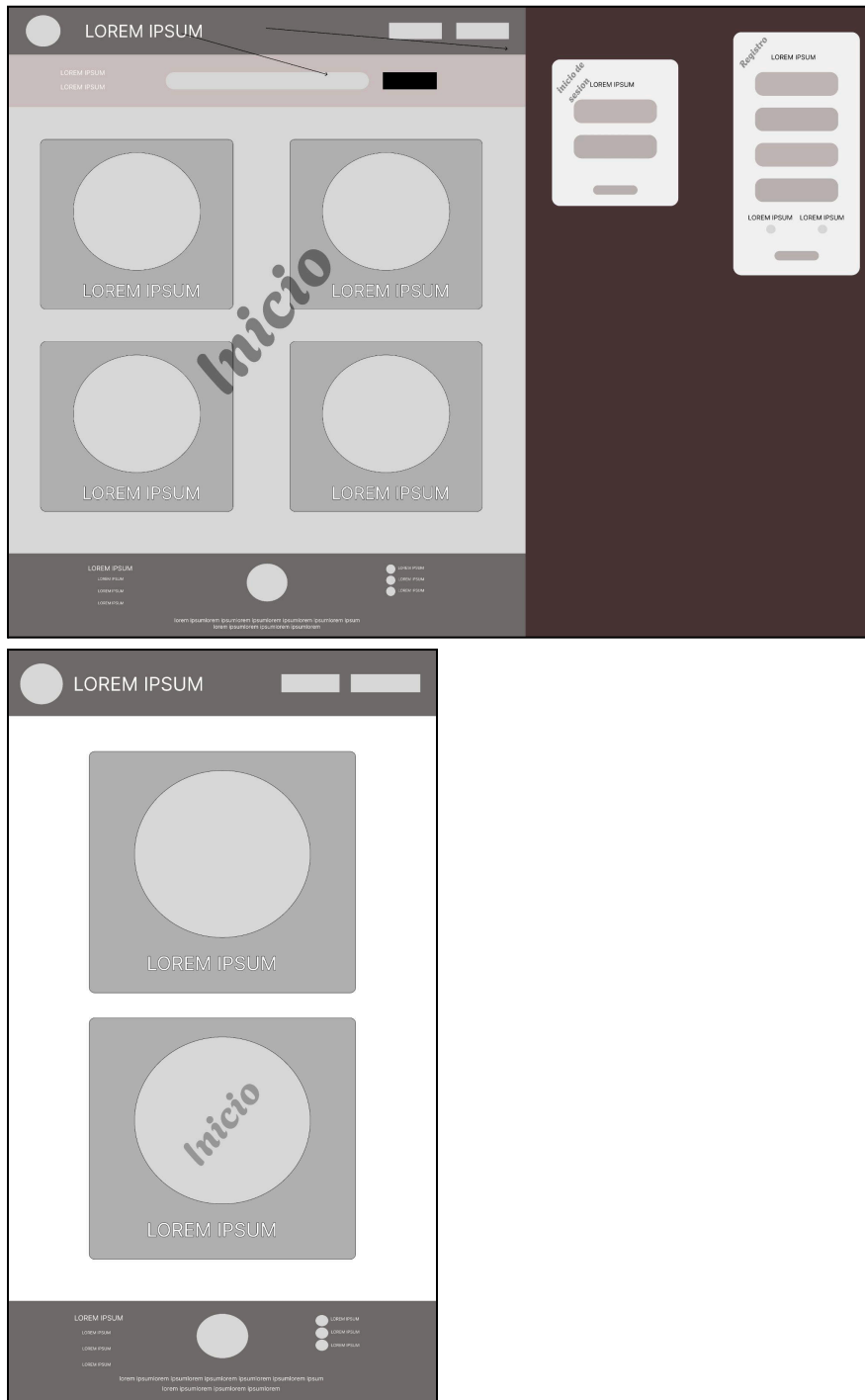
8.1 Sketches

El diseño inicial de la interfaz se basó en wireframes y sketches elaborados en herramientas de prototipado rápido, donde se definieron los siguientes aspectos:

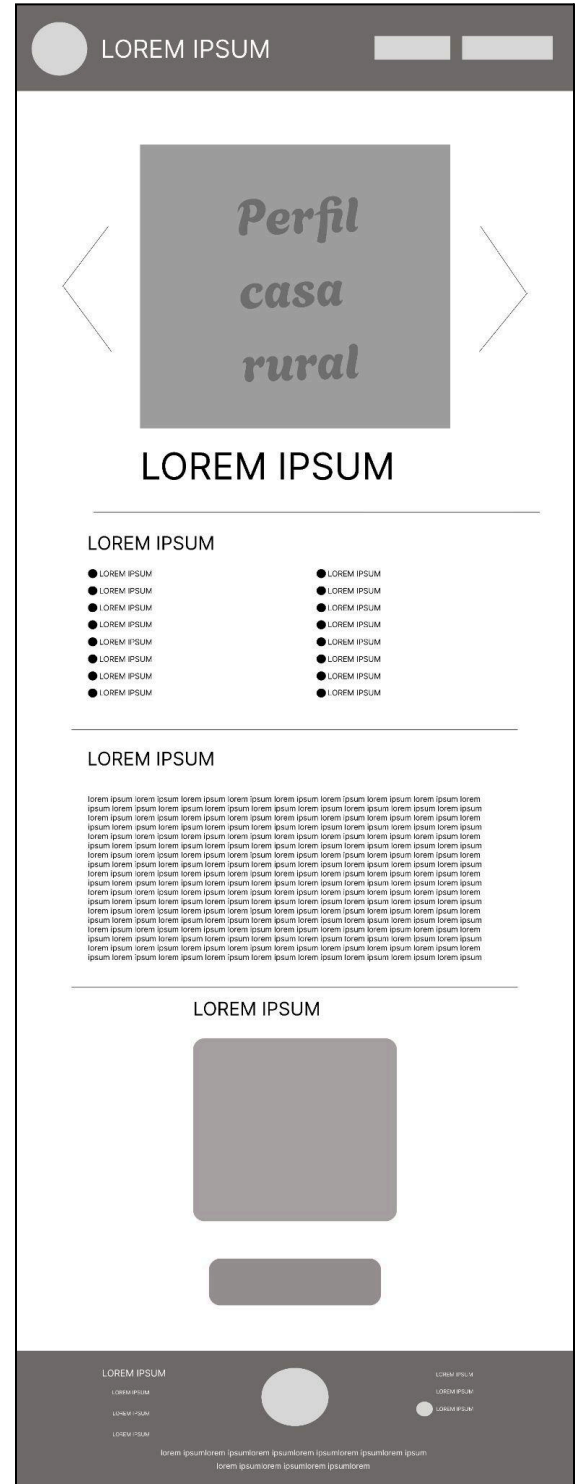
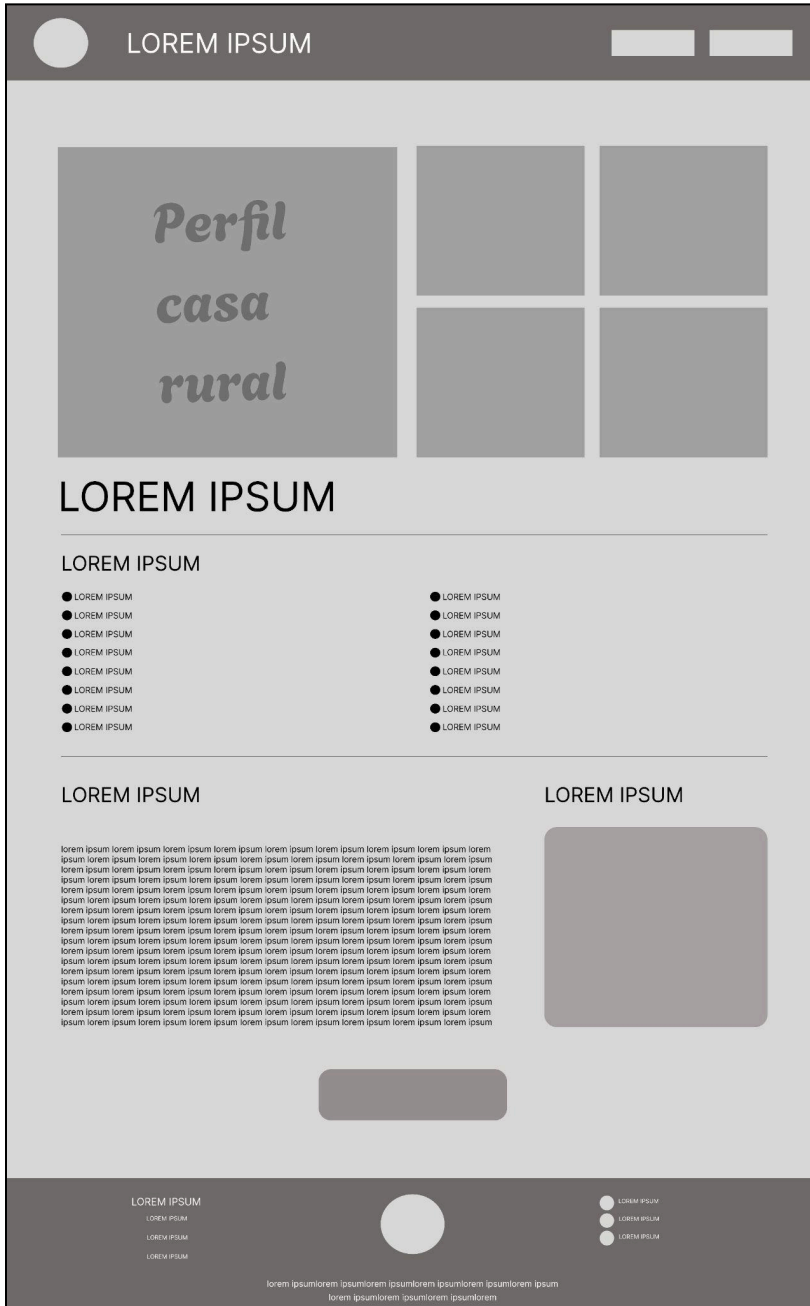
- Estructura básica de la página de inicio con galería de casas.

- Página de detalle de casa con galería de imágenes y formulario de reserva.
- Gestión de casas para el propietario con cards representando cada alojamiento.
- Gestión de reservas en vista tipo lista.
- Blog de valoraciones con estructura de tarjetas.
- Perfil de usuario con información personal, reservas y valoraciones realizadas.

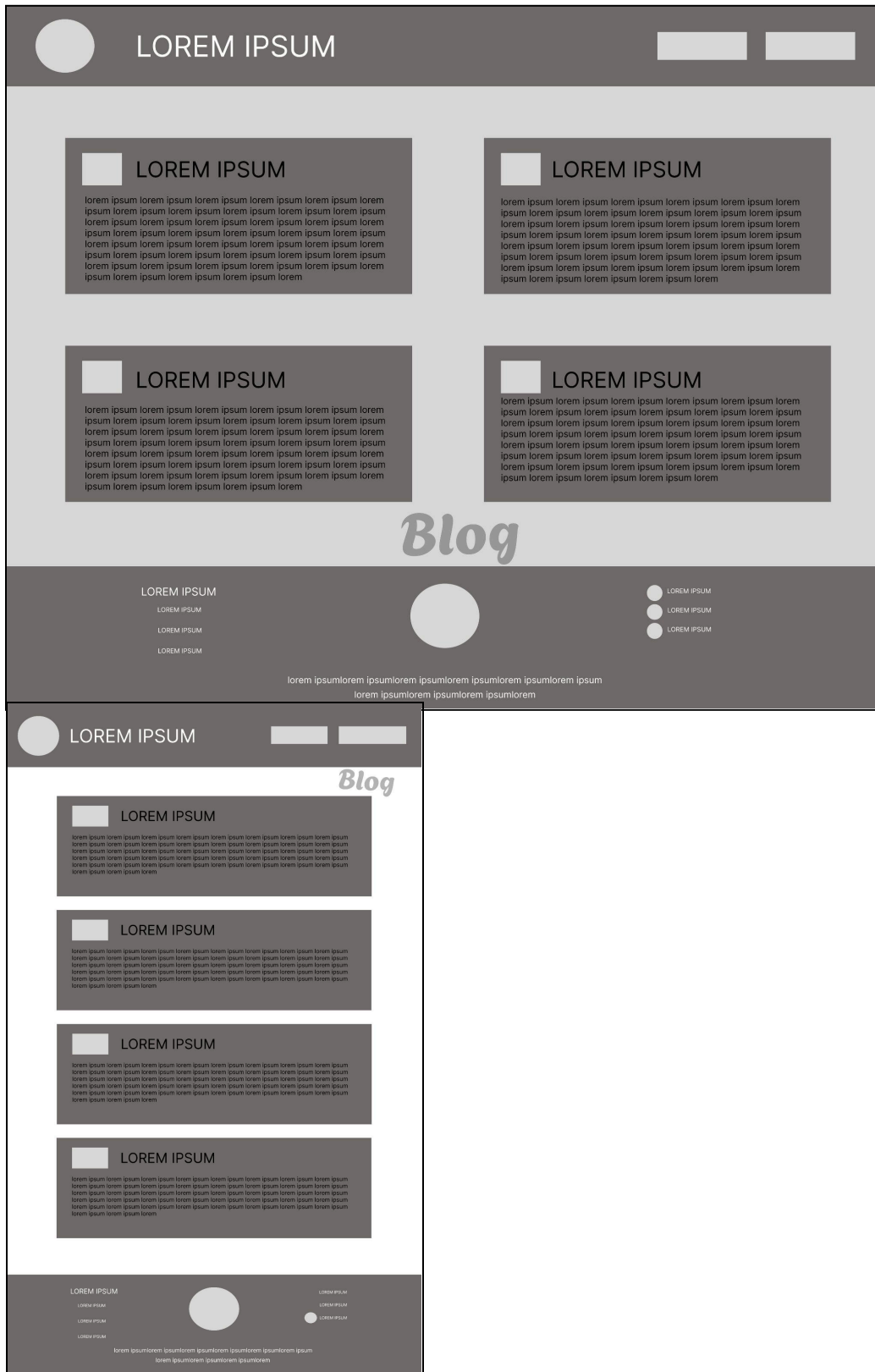
8.1.1 Inicio



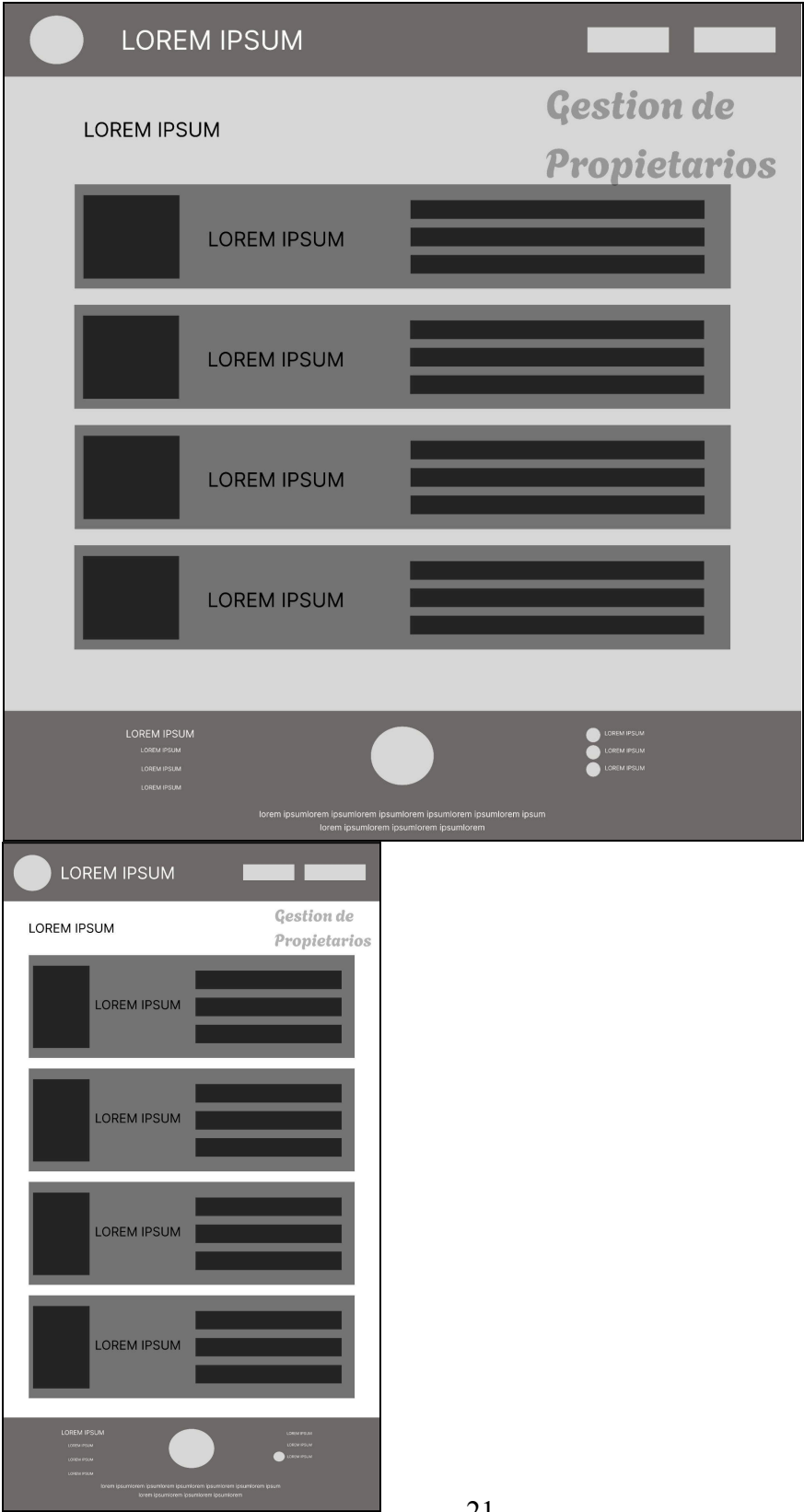
8.1.2 Perfil Casa Rural



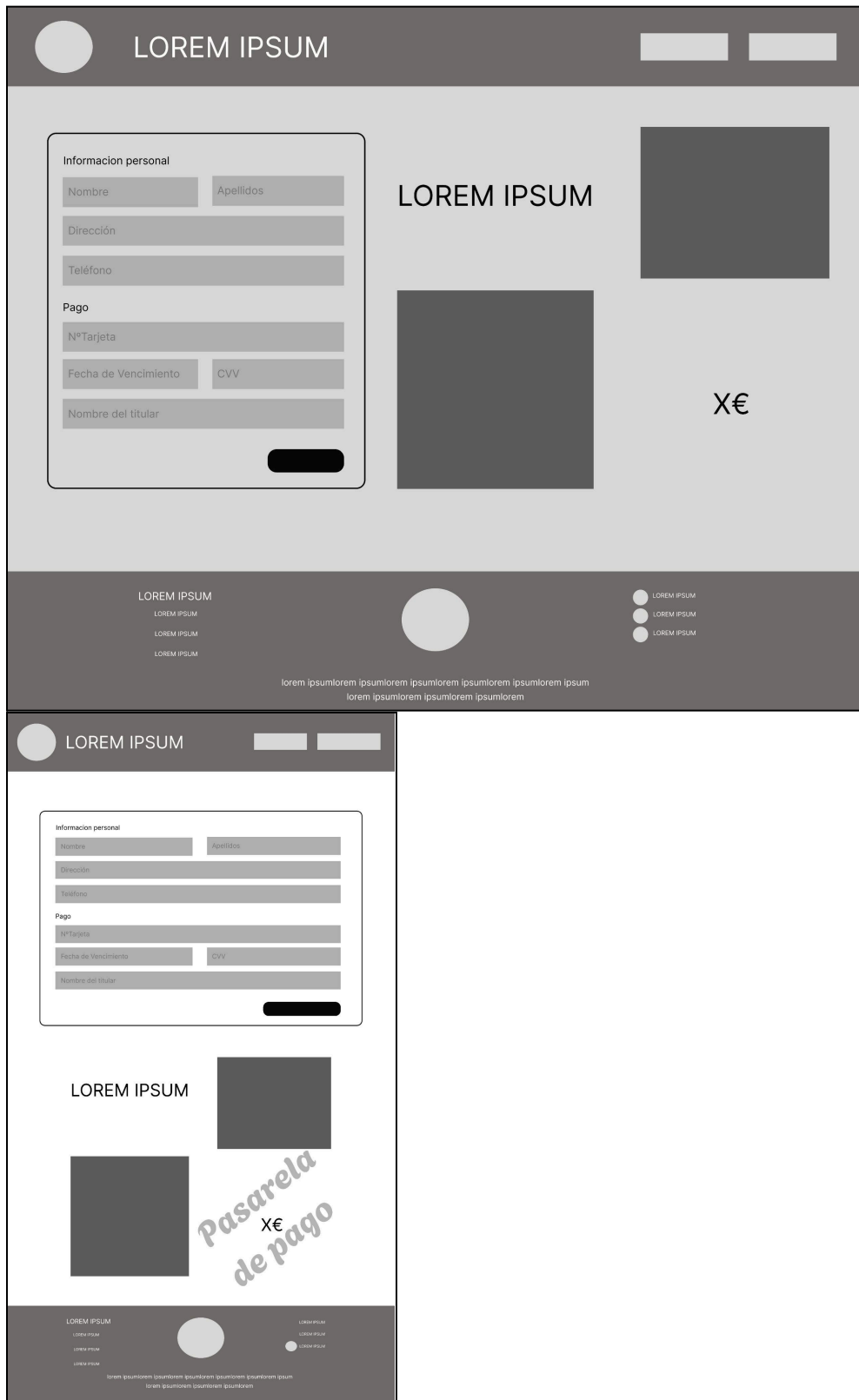
8.1.3 Blog



8.1.4 Gestión Propietarios



8.1.5 Pasarela de Pago



The image displays two wireframe versions of a payment gateway interface. The top wireframe represents a desktop layout, while the bottom one represents a mobile layout. Both versions feature a header with a logo placeholder (LOREM IPSUM) and navigation elements. The main content area is divided into a left sidebar for user information and a right section for payment details. The sidebar includes fields for personal information (Name, Surname, Address, Phone) and payment information (Card Number, Expiry Date, CVV, Cardholder Name). The right section contains a large placeholder for a card image, a price display (X€), and a payment button. The mobile version adapts these elements for a smaller screen, with the payment button and price display being more prominent.

Desktop View:

- Header:** LOREM IPSUM
- Personal Information:**
 - Nombre
 - Apellidos
 - Dirección
 - Teléfono
- Pago:**
 - Nº Tarjeta
 - Fecha de Vencimiento
 - CVV
 - Nombre del titular
- Payment Details:**
 - LOREM IPSUM
 - X€

Mobile View:

- Header:** LOREM IPSUM
- Personal Information:**
 - Nombre
 - Apellidos
 - Dirección
 - Teléfono
- Pago:**
 - Nº Tarjeta
 - Fecha de Vencimiento
 - CVV
 - Nombre del titular
- Payment Details:**
 - LOREM IPSUM
 - X€

Pasarela de pago

8.2 Criterios de accesibilidad

- Uso de etiquetas <label> correctamente asociadas a los campos de los formularios.
- Textos alternativos (alt) en todas las imágenes.
- Contraste de color suficiente entre fondo y texto.
- Navegación accesible mediante teclado (tabulación lógica).
- Formularios validados con mensajes de error claros.
- Uso de elementos semánticos HTML5 (header, nav, main, footer...).

8.3 Criterios de usabilidad

- Interfaz clara y coherente entre páginas.
- Botones con texto descriptivo y fácil interacción.
- Confirmación visual en acciones importantes (crear, eliminar).
- Transiciones suaves entre vistas.
- Responsive: la página se adapta correctamente a diferentes tamaños de pantalla.
- Feedback inmediato tras el envío de formularios mediante toasts.

8.4 Tipografía

- Fuente principal: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif
- Usada en:
 - Todo el texto de la interfaz.
 - Títulos, botones y enlaces.

Motivo de elección: fuente sans-serif, moderna y de alta legibilidad en pantalla.

8.5 Mapa de colores del proyecto

Elemento / Uso	Color (nombre)	RGB	Hexadecimal
Fondo degradado Header/Footer	Verde menta → blanco	rgb(96, 226, 155) → rgb(255, 255, 255)	#60e29b → #ffffff
Botón principal	Verde oscuro	rgb(39, 174, 96)	#27ae60
Botón hover	Verde más oscuro	rgb(33, 145, 80)	#219150
Fondo formularios	Blanco	rgb(255, 255, 255)	#ffffff
Texto principal	Gris oscuro	rgb(34, 34, 34)	#222222

Texto secundario	Gris medio	rgb(102, 102, 102)	#666666
Cards (valoraciones) fondo	Blanco	rgb(255, 255, 255)	#ffffff

8.6 Dispositivos / vistas para las que se ha diseñado el proyecto

- Escritorio (Desktop): resolución mínima 1280px, adaptado hasta 1920px.
- Tablet: adaptado a dispositivos de 768px a 1280px.
- Móvil: adaptado a pantallas a partir de 360px de ancho.

Tecnología usada para el diseño responsive:

- CSS Grid + Flexbox + Media Queries específicas para los puntos de ruptura descritos.
- Vue.js para la estructura de componentes.

9. Software utilizado

Durante el desarrollo del proyecto *WeekendHouse*, se han empleado diversas herramientas de software tanto para la implementación como para la documentación, diseño y pruebas. A continuación se detallan las principales:

9.1 Editor de código y entorno de desarrollo

- **Visual Studio Code:** Utilizado como editor principal para el desarrollo del frontend (Vue.js), configuración de estilos y edición de código PHP en el backend.
- **XAMPP:** Usado como servidor local para el backend, ofreciendo Apache + PHP + MySQL. Permite simular un entorno de servidor de producción en local.

9.2 Frontend

- **Vue.js (versión 3):** Framework de JavaScript utilizado para la construcción de la interfaz de usuario. Permite el desarrollo de componentes reutilizables y una arquitectura reactiva.
- **Vue Router:** Librería de enrutado empleada para la navegación entre las diferentes vistas de la aplicación.

- **V-Calendar:** Componente de calendario utilizado para la selección de fechas en el formulario de reservas.
- **Vue Toastification:** Librería empleada para mostrar notificaciones de manera elegante y no intrusiva en la aplicación.
- **Font Awesome:** Utilizada para incluir iconos en el diseño de la interfaz, mejorando la experiencia visual del usuario.

9.3 Backend

- **PHP:** Lenguaje de programación utilizado para el desarrollo del backend de la aplicación. Los endpoints de la API REST se han implementado en PHP.
- **MySQL:** Sistema de gestión de bases de datos relacional utilizado para almacenar toda la información del sistema (usuarios, casas, reservas, valoraciones, etc.).
- **phpMyAdmin:** Herramienta gráfica para la gestión de la base de datos MySQL. Facilita el trabajo con las tablas, consultas y mantenimiento de la base de datos.

9.4 Control de versiones

- **Git:** Utilizado para el control de versiones del código fuente del proyecto.
- **GitHub:** Plataforma utilizada para alojar el repositorio remoto y mantener una copia segura del código del proyecto.

9.5 Documentación y diseño

- **Google Docs:** Empleado para la redacción de la memoria del proyecto y la documentación técnica.
- **Figma:** Utilizado para el diseño previo de la interfaz, la creación de mockups y la planificación de la experiencia de usuario.
- **LucidChart :** Usado para la creación de los diferentes diagramas de clases , entidad relación, menú de navegación.

9.6 Navegadores para pruebas

- **Google Chrome:** Navegador principal utilizado durante el desarrollo para pruebas y depuración de la aplicación.
- **Mozilla Firefox y Microsoft Edge:** Navegadores alternativos utilizados para comprobar la compatibilidad y el correcto funcionamiento de la aplicación en diferentes entornos.

10. Mejoras posibles y aportaciones

El desarrollo de *WeekendHouse* ha sido iterativo e incremental, y aunque la plataforma en su versión actual es completamente funcional, se identifican diversas mejoras y posibles futuras funcionalidades que ayudan a la experiencia del usuario y aumentarían el valor del sistema.

10.1 Mejoras en funcionalidad

- **Gestión avanzada de imágenes**
Sustituir el almacenamiento local actual de imágenes por almacenamiento en la base de datos o en un servicio de almacenamiento externo (AWS S3, Firebase Storage, etc.). Esto facilita la gestión y portabilidad de la aplicación.
- **Calendario con control de disponibilidad**
Implementar un sistema de calendario avanzado que muestre los días ocupados de cada casa en tiempo real.
- **Pasarela de pago real**
Integrar una pasarela de pago como Stripe o PayPal para permitir reservas con pago seguro en línea.
- **Sistema de notificaciones por email**
Enviar notificaciones automáticas a los usuarios y propietarios cuando se realiza o actualiza una reserva.
- **Buscador avanzado en el blog y catálogo de casas**
Mejorar el buscador actual incorporando filtros por precio, ubicación, servicios, valoración media, etc.
- **Valoraciones más completas**
Añadir la posibilidad de incluir imágenes en las valoraciones y permitir respuestas de los propietarios.

10.2 Mejoras en la organización del código

- Refactorizar y modularizar el código JS en servicios reutilizables.
- Unificar el uso de fetch y axios (actualmente coexisten ambos) para un estilo más consistente.

10.3 Mejoras en la organización de la información

- Añadir panel de administración para visualizar analíticas básicas: número de reservas, casas más populares, ingresos estimados.

- Historial de reservas en el perfil del usuario.
- Posibilidad de editar o eliminar valoraciones propias.

10.4 Ideas futuras / aportaciones

- Posible conversión a PWA (Progressive Web App) para permitir uso offline y notificaciones push.
- Aplicación móvil complementaria utilizando Vue 3 + Capacitor o React Native.
- Multi-idioma para facilitar el acceso a turistas internacionales.

11. Bibliografía

- Vue.js. (n.d.). <https://vuejs.org/>
- V-Calendar. (n.d.). <https://vcalendar.io/>
- Toastification. (n.d.). <https://vue-toastification.maronato.dev/>
- MDN Web Docs. (n.d.). https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design
- PHP Manual. (n.d.). <https://www.php.net/docs.php>
- Supabase. (n.d.). <https://supabase.com/docs/guides/api/rest>