

Ciclo Formativo de Grado superior

PROYECTO DE DESARROLLO DE APLICACIONES WEB

Desarrollo web de casa rural HERANIKA

Alumno: Daniel Herrera Romera

Línea: Lenguajes de Marcas y Sistemas de Gestión de Información
 Desarrollo web Entorno cliente
 Diseño de interfaces web
 Desarrollo web en entorno servidor

Tutor del proyecto: Andrés Leonardo Acosta Castillo

CURSO 2024-2025

ÍNDICE

RESUMEN	3
JUSTIFICACIÓN	4-5
MARCO TEÓRICO	6-7
OBJETIVOS	8
DESARROLLO DEL PROYECTO	9-25
CONCLUSIONES Y FUTURAS MEJORAS	26-27
BIBLIOGRAFÍA	28

RESUMEN

Objetivo principal:

El objetivo de este proyecto es desarrollar una página web funcional para la finca rural **Heranika**, que facilite la gestión de reservas, pagos online y la interacción de los usuarios con el servicio. La plataforma proporciona un sistema de reservas intuitivo, una pasarela de pagos mediante PayPal, un calendario interactivo de disponibilidad, una sección para gestionar las opiniones de los clientes y un apartado de contacto y preguntas frecuentes.

Tecnologías empleadas:

- **Frontend:** HTML, CSS, JavaScript
- **Backend:** Node.js
- **Base de datos:** MongoDB
- **Pasarela de pagos:** PayPal API

Alcance y funcionalidades clave:

La página web permite a los usuarios ver toda la información disponible de esta propiedad, hacer reservas con comprobaciones de disponibilidad, realizar pagos en línea seguros y gestionar sus reservas desde su cuenta personal. Además, los usuarios pueden dar opiniones sobre su estancia en esta propiedad, y el sistema permite la gestión fluida y segura de la información del usuario.

JUSTIFICACIÓN

La idea de este proyecto surgió a partir de una necesidad real detectada en mi entorno. Un familiar cercano gestiona una finca rural llamada **Heranika**, que se alquila como alojamiento vacacional para turistas. Sin embargo, hasta el momento no contaba con una página web que facilitara las reservas ni presentara de manera clara y atractiva la información del alojamiento.

Al observar esta carencia, vi la oportunidad de desarrollar una solución que le ayudara a mejorar la gestión del negocio, ya que actualmente las reservas se realizan de forma manual o mediante canales poco eficientes, como llamadas telefónicas o correos electrónicos. Además, una página web permitiría dar mayor visibilidad a la finca y llegar a un público más amplio. Al mismo tiempo, decidí aprovechar esta ocasión para enfocar mi Trabajo de Fin de Grado en un proyecto práctico, útil y con aplicación real.

El principal problema detectado es la falta de una plataforma digital que permita a la finca Heranika gestionar de manera eficiente las reservas, mejorar la atención al cliente y ampliar su visibilidad entre posibles visitantes.

La web desarrollada responde a esta necesidad ofreciendo una solución completa y accesible: cualquier persona interesada puede consultar la información del alojamiento, revisar la disponibilidad en un calendario interactivo, realizar una reserva y efectuar el pago de forma segura desde su hogar. Además, los usuarios pueden dejar opiniones sobre su estancia, contactar fácilmente con el alojamiento y acceder a sus reservas si están registrados, lo que mejora la experiencia del cliente y optimiza el funcionamiento del negocio.

Este trabajo me ha servido para poner en práctica todo lo que he aprendido durante el ciclo, y además me ha motivado mucho saber que lo que estoy haciendo tendrá un uso real y ayudará a alguien cercano. Por eso considero que este proyecto tiene valor tanto a nivel académico como personal.

Por otra parte, la importancia de la utilidad de esta página web aporta una comodidad y seguridad para el profesional y el cliente debido a las herramientas introducidas en la web. Le aporta a la vez una visibilidad que anteriormente no poseía, que el usuario pueda ver tanto imágenes reales de todas las instalaciones como comentarios de huéspedes anteriores donde han podido dejar sus reseñas y así decidirse en alquilar la finca con una mayor transparencia de esta manera, se optimiza la atención al cliente y se amplía el alcance del negocio a un público más amplio.

Este proyecto cobra especial peso por varios motivos. En primer lugar responde a la necesidad de digitalización del sector rural, que muchas veces se encuentra rezagado en comparación con otros ámbitos turísticos. Y en segundo lugar, mejora la experiencia del usuario al proporcionar una herramienta intuitiva, accesible desde cualquier dispositivo.

Desde el punto de vista técnico, el uso de tecnologías modernas como **Node.js** para el backend y **MongoDB** como base de datos, garantiza un rendimiento eficiente y escalable. El frontend, desarrollado con **HTML**, **CSS** y **JavaScript**, ofrece una interfaz clara y adaptativa. Además, la integración con la pasarela de pagos **PayPal** asegura transacciones seguras y rápidas, algo fundamental en una plataforma de reservas online.

En conclusión, el desarrollo de esta página web para la finca Heranika no solo cubre una necesidad operativa, sino que se convierte en un elemento estratégico para su crecimiento, profesionalización y posicionamiento en el mercado del turismo rural. Se trata de una solución tecnológica accesible, útil y con proyección a futuro, alineada con las expectativas de los usuarios actuales y con las exigencias del entorno digital en el que vivimos.

MARCO TEÓRICO

Situación actual y soluciones existentes

A nivel general, existen plataformas como **Booking** o **Airbnb**, que ofrecen sistemas completos para la gestión de alojamientos, permitiendo a los propietarios automatizar las reservas, recibir pagos online y llegar a un gran número de usuarios. Sin embargo, estos servicios imponen ciertas limitaciones: el diseño no es personalizable, las comisiones por reserva pueden ser elevadas (y el contacto con los clientes queda en muchos casos intermediado por la propia plataforma). Esto limita la autonomía del propietario y dificulta la adaptación a necesidades particulares.

Por ello, desarrollar una **plataforma propia** supone una alternativa real y beneficiosa. En el caso de la finca Heranika, una web personalizada permite reflejar mejor la esencia del lugar, adaptarse a su modelo de gestión y crear un canal de comunicación directo con los clientes. A nivel práctico, esta página web aporta un valor añadido tanto para el propietario como para los usuarios, al mejorar la organización, aumentar la visibilidad y ofrecer una experiencia de navegación cómoda, clara y funcional.

Tecnologías utilizadas

Desde el punto de vista técnico, he optado por un conjunto de herramientas modernas, accesibles y perfectamente integrables entre sí, que permiten desarrollar una plataforma funcional y dinámica.

Frontend: HTML, CSS y JavaScript

La interfaz de la web ha sido construida con **HTML5** para la estructura, **CSS3** para los estilos visuales y **JavaScript** para la interactividad. Estas tecnologías permiten crear una experiencia de usuario clara, visualmente atractiva y adaptada a distintos dispositivos. El objetivo era ofrecer una navegación sencilla e intuitiva, para que cualquier persona pueda acceder fácilmente a la información del alojamiento, realizar una reserva o contactar con el propietario sin dificultades técnicas.

Backend: Node.js

Para la parte del servidor se ha utilizado **Node.js**, un entorno de ejecución que permite desarrollar aplicaciones web eficientes y rápidas utilizando JavaScript también en el backend. Node.js destaca por su rendimiento y por facilitar la creación de servicios web que responden de manera ágil a las solicitudes de los usuarios.

Base de datos: MongoDB

Los datos de la aplicación, como los usuarios, las reservas, la disponibilidad o las opiniones, se almacenan en **MongoDB**, una base de datos que permite estructurar la información de forma flexible y dinámica. Su integración con Node.js mediante herramientas como Mongoose facilita la validación de datos y mejora la seguridad del sistema.

Pasarela de pagos: PayPal API

Uno de los elementos clave de la plataforma es la **pasarela de pagos** integrada mediante la **API de PayPal**, que permite realizar transacciones seguras desde la misma página web. Gracias a esta herramienta, los usuarios pueden pagar con confianza utilizando su cuenta PayPal, lo cual añade comodidad y profesionalidad al proceso de reserva.

Aporte del proyecto y utilidad real

Este proyecto tiene un valor especial por su carácter práctico y personal. No se trata solo de un ejercicio académico, sino de una solución real que ayudará directamente a un familiar a mejorar la gestión de su negocio. A través de la web, los usuarios pueden consultar la disponibilidad de la finca mediante un calendario interactivo, hacer reservas y pagos online, leer opiniones de otros clientes, contactar fácilmente con el alojamiento y acceder a su perfil para revisar o modificar sus reservas. Todo ello contribuye a optimizar la atención al cliente y a mejorar la imagen del alojamiento.

Además, la posibilidad de mostrar imágenes reales, testimonios verificados y detalles del entorno genera mayor confianza y transparencia. Esto resulta fundamental en el sector rural, donde muchos negocios aún no han dado el paso hacia la digitalización.

En conclusión, esta página web representa una herramienta completa y útil que responde a una necesidad concreta, pero también ofrece posibilidades de crecimiento futuro. La estructura técnica utilizada permite implementar nuevas funcionalidades a medida que el negocio lo necesite. En definitiva, se trata de una solución que aporta valor a nivel operativo, estratégico y emocional, alineada con los retos actuales de la digitalización y el turismo rural.

OBJETIVOS

El proyecto se desarrolla con el propósito de mejorar la gestión del alojamiento rural Heranika mediante una plataforma digital. Para ello, se establece un objetivo general y varios objetivos específicos que guían y estructuran su desarrollo.

Objetivo general:

Desarrollar una página web funcional y personalizada para la finca rural Heranika que permita gestionar reservas de manera eficiente, facilitar el pago online, mejorar la comunicación con los usuarios y aumentar la visibilidad del alojamiento en internet.

Objetivos específicos:

1. **Diseñar una interfaz web clara, atractiva y adaptativa**, que permita al usuario consultar la información del alojamiento, ver imágenes reales de las instalaciones, y navegar de forma intuitiva desde cualquier dispositivo (ordenador, móvil o tablet).
2. **Implementar un sistema de reservas online con calendario interactivo**, que permita comprobar la disponibilidad en tiempo real y realizar la reserva de forma cómoda, sin necesidad de contacto telefónico o por correo.
3. **Integrar una pasarela de pagos segura mediante la API de PayPal**, que facilite a los usuarios realizar el pago de su reserva de forma rápida y protegida.
4. **Desarrollar una sección privada para los usuarios registrados**, desde la cual puedan gestionar sus reservas y consultar el historial de reservas.
5. **Crear un sistema de opiniones o valoraciones**, donde los clientes puedan dejar reseñas sobre su experiencia, ayudando a mejorar la página web y aportando confianza a futuros visitantes.
6. **Permitir al administrador de la finca gestionar los contenidos del sitio web**, incluyendo la modificación de disponibilidad, la gestión de usuarios y reservas, y el control general de la plataforma de manera sencilla.

Estos objetivos han guiado el desarrollo del proyecto desde su planteamiento inicial hasta la implementación final, con el propósito de construir una herramienta útil, práctica y alineada con las necesidades reales del negocio rural Heranika.

DESARROLLO

1. Análisis y Requisitos

1.1 Requisitos Funcionales (RF)

Código	Requisito
RF1	Registro, login y gestión de sesiones para usuarios.
RF2	Visualización de información de la finca.
RF3	Consulta de disponibilidad mediante calendario.
RF4	Sistema de reservas con validaciones.
RF5	Pago online a través de PayPal.
RF6	Publicación y moderación de opiniones.
RF7	Gestión de usuarios y reservas desde el panel admin.
RF8	Sistema de contacto y FAQ.

1.2 Requisitos No Funcionales (RNF)

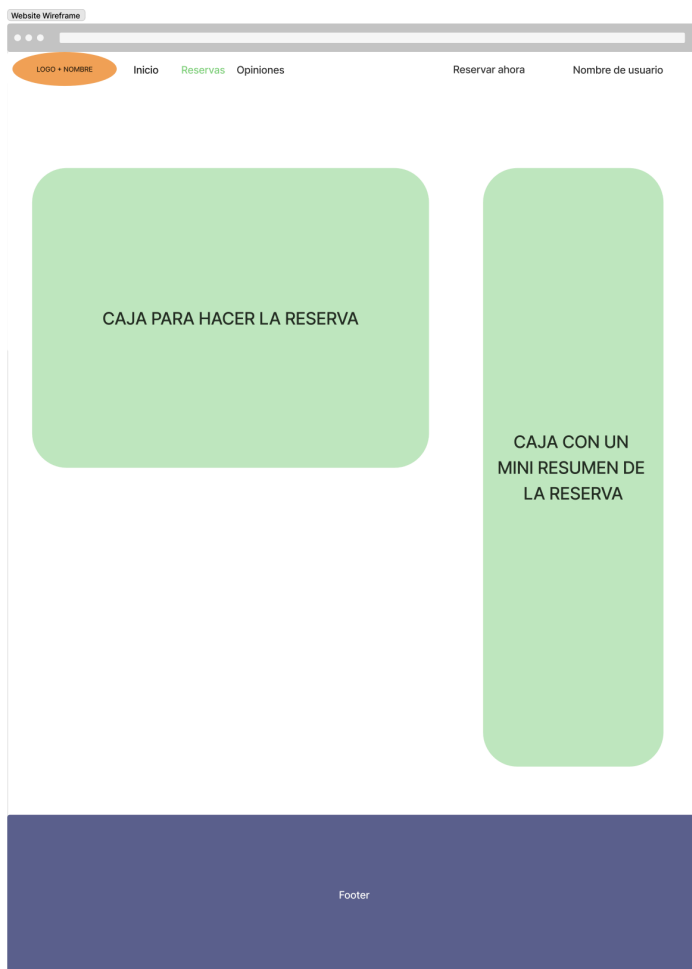
Código	Requisito
RNF1	El sistema debe cargar en menos de 2 segundos.
RNF2	Soporte para móviles, tablets y escritorio.
RNF3	Cifrado de contraseñas y tokens JWT para autenticación.
RNF4	Escalabilidad del backend y la base de datos.
RNF5	Interfaz accesible y clara.

2. Diseño

2.1 Wireframe del sitio

(WireFrame creado con figma)

[Enlace a Figma](#)



[LOGO]

[Barra navegación]

[Información principal]

[Calendario]

[Botón "Reservar"]

[Footer]

2.2 Arquitectura de la Aplicación

Se ha seguido una arquitectura **MVC** (Modelo-Vista-Controlador):

```

heranika/
├── node_modules/           # Dependencias del proyecto (manejadas por pnpm)
├── fotos-campo/           # Imágenes originales del terreno (fuera del src)
│   └── *.jpg
├── package.json            # Configuración principal del proyecto y scripts
├── package-lock.json       # Lockfile de dependencias (para npm)
├── pnpm-lock.yaml          # Lockfile específico de pnpm
├── tests/                  # Pruebas automatizadas
│   └── booking.test.js
└── src/                    # Código fuente principal de la aplicación
    ├── app.js              # Punto de entrada de la aplicación
    ├── config/             # Configuración general
    │   ├── database.js     # Conexión a MongoDB mediante Mongoose
    │   └── testConnection.js # Utilidad para probar la conexión a la BD
    ├── middleware/         # Middlewares personalizados
    │   ├── auth.js         # Autenticación y autorización
    │   ├── rateLimiter.js  # Límite de peticiones por IP
    │   └── validator.js    # Validación de formularios y datos
    ├── models/             # Modelos Mongoose (BD)
    │   ├── Booking.js      # Modelo de reservas
    │   ├── Review.js       # Modelo de opiniones
    │   └── User.js         # Modelo de usuarios
    ├── routes/             # Definición de rutas de Express
    │   ├── auth.js         # Registro e inicio de sesión
    │   ├── bookings.js     # Gestión de reservas
    │   ├── contact.js      # Formulario de contacto
    │   ├── index.js        # Rutas principales
    │   ├── payment.js      # Integración con sistema de pagos
    │   └── reviews.js      # Opiniones de usuarios
    └── utils/              # Funciones de utilidad
        ├── emailService.js  # Envío de correos electrónicos
        └── errorLogger.js   # Registro de errores
  
```

public/	# Archivos públicos accesibles desde el cliente
css/	# Estilos CSS organizados por componentes
*.css	# Ficheros base, utilidades y específicos
components/	# Estilos de componentes reutilizables
js/	# Scripts de frontend
*.js	
images/	# Imágenes generales de la web
banner.avif	
pruebaBanner.jpg	
Bienvenidos.png	
fotos-campo/	# Galería de imágenes del alojamiento
*.jpg	
views/	# Plantillas EJS (vistas renderizadas en servidor)
auth/	# Vistas de autenticación
bookings/	# Vistas relacionadas con reservas
contact/	# Formulario de contacto
home/	# Página principal
reviews/	# Sección de opiniones
partials/	# Componentes parciales (navbar, footer, etc.)
layout.ejs	# Layout principal reutilizable

3. Desarrollo

3.1 Registro de usuario (fragmento de código)

// models/User.js

```
const mongoose = require('mongoose');
const bcrypt = require('bcrypt');

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    unique: true,
    trim: true,
    lowercase: true
  },
  password: {
    type: String,
    required: true
  }
}, {
  timestamps: true
});
```

Este código define el modelo de Usuario (**User**) para una aplicación Node.js usando Mongoose.

Así se ve la estructura del usuario:

name: Nombre de usuario (obligatorio)

email: Correo electrónico (único y obligatorio)

password: Contraseña (obligatoria)

timestamps: Añade automáticamente createdAt y updatedAt

3.2 Gestión de reservas

// models/Bookings.js

```
const mongoose = require('mongoose');

const bookingSchema = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  checkIn: {
    type: Date,
    required: true
  },
  checkOut: {
    type: Date,
    required: true
  },
  guests: {
    adults: {
      type: Number,
      required: true
    },
    children: {
      type: Number,
      default: 0
    }
  },
  totalPrice: {
    type: Number,
    required: true
  }
}, { timestamps: true });

module.exports = mongoose.model('Booking', bookingSchema);
```

Este código define el modelo de Reserva (**Booking**) para una aplicación de reservas. Así se ve la estructura de las reservas:

user: Referencia al ID del usuario que hace la reserva (obligatorio)

checkIn y checkOut: Fechas de entrada y salida (ambas obligatorias)

guests: Objeto que contiene:

adults: Número de adultos (obligatorio)

children: Número de niños (opcional, por defecto 0)

totalPrice: Precio total de la reserva (obligatorio)

3.3 Opiniones de usuarios

// models/Opinion.js

```
const mongoose = require('mongoose');

const reviewSchema = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  title: {
    type: String,
    required: true
  },
  content: {
    type: String,
    required: true
  },
  rating: {
    type: Number,
    required: true,
    min: 1,
    max: 5
  }
}, { timestamps: true });

module.exports = mongoose.model('Review', reviewSchema);
```

Este código define el modelo de Reseña (**Review**) para la aplicación. Así se ve la estructura de las opiniones:

user: Referencia al ID del usuario que escribe la reseña (obligatorio)

title: Título de la reseña (obligatorio)

content: Contenido de la reseña (obligatorio)

rating: Valoración del 1 al 5 (obligatorio, con validación de rango)

timestamps: Añade automáticamente createdAt y updatedAt

3.4 Ruta POST para crear reservas (/bookings/)

- Verifica autenticación del usuario.
- Valida campos obligatorios.
- Valida formato y lógica de fechas.
- Verifica disponibilidad.

Validaciones principales:

```
// Validar que la fecha de check-in no sea en el pasado
if (checkIn < today) {
  return res.status(400).json({
    success: false,
    error: 'La fecha de entrada no puede ser en el pasado'
  });
}

// Validar que la fecha de salida sea posterior a la de entrada
if (checkOut <= checkIn) {
  return res.status(400).json({
    success: false,
    error: 'La fecha de salida debe ser posterior a la de entrada'
  });
}

// Validar número de huéspedes
const adults = parseInt(req.body.adults);
const children = parseInt(req.body.children) || 0;
const totalGuests = adults + children;

if (isNaN(adults) || adults < 1) {
  return res.status(400).json({
    success: false,
    error: 'Debe haber al menos un adulto en la reserva'
  });
}
```

Ruta para verificar la disponibilidad (/bookings/disabled-dates/):

- Obtiene todas las reservas futuras.
- Genera un Array de fechas ocupadas.
- Devuelve las fechas no disponibles en el calendario.

Seguridad:

- Middleware **requireAuth** para proteger rutas.
- Validación de datos de entrada.
- Manejo de errores detallado.

Estructura de datos de la reserva:

- Usuario (referencia).
- Fechas de check-in y check-out.
- Número de adultos y niños.
- Precio total.

Este código es fundamental para el sistema de reservas, asegurando que:

- Las reservas sean válidas.
- No haya conflictos de fechas.
- Los datos de su usuario estén protegidos.

3.5 Explicación de como crear reseñas (/reviews/)**Rutas principales:**

- **GET /:** Muestra todas las reseñas ordenadas por fecha (más recientes primero).
- **POST /:** Crea una nueva reseña (requiere autenticación).
- **PUT /:id:** Actualiza una reseña existente (solo el dueño).
- **DELETE /:id:** Elimina una reseña existente (solo el dueño).

Características clave:

- Seguridad:

```
// Verificar si la reseña existe y pertenece al usuario
if (!review || review.user.toString() !== req.session.userId) {
  return res.status(403).json({
    error: 'No tienes permiso para editar esta reseña'
  });
}
```

Flujo de datos:

- Creación:

```
// Crear la nueva reseña
const review = new Review({
  user: req.session.userId,
  title,
  content,
  rating: parseInt(rating)
});

await review.save();
```

- Consulta:

```
router.get('/', async (req, res) => {
  try {
    const reviews = await Review.find()
      .populate('user', 'name id') // Asegúrate de incluir el id
      .sort({ createdAt: -1 });
    res.render('reviews/index', {
      reviews,
      user: req.session.user // Asegúrate de pasar el usuario a la vista
    });
  } catch (error) {
    console.error('Error al obtener las opiniones:', error);
    res.status(500).send('Error al cargar las opiniones');
  }
});
```

4. Guía de estilos

1. Paleta de Colores

Colores Principales

- **Primario:** #2A9D8F (Verde azulado)
- **Secundario:** #264653 (Azul oscuro)

Colores de Feedback

- **Éxito:** #10B981 (Verde esmeralda)
- **Advertencia:** #F59E0B (Ámbar)
- **Peligro:** #EF4444 (Rojo)
- **Información:** #3B82F6 (Azul)

Escala de Grises

--gray-100: #F8F9FA;
--gray-200: #E9ECEF;
--gray-300: #DEE2E6;
--gray-400: #CED4DA;
--gray-500: #ADB5BD;
--gray-600: #6C757D;
--gray-700: #495057;
--gray-800: #343A40;
--gray-900: #212529;

2. Tipografía

Fuentes

- **Principal (sans-serif):** [Poppins](#) (400, 500, 700, 800)
- **Secundaria (serif):** [Playfair Display](#) (400, 700)
- **Monoespaciada:** [JetBrains Mono](#) (400)

Tamaños de Fuente

```
--text-xs: 0.75rem;  
--text-sm: 0.875rem;  
--text-base: 1rem;  
--text-lg: 1.125rem;  
--text-xl: 1.25rem;  
--text-2xl: 1.5rem;  
--text-3xl: 1.875rem;  
--text-4xl: 2.25rem;  
--text-5xl: 3rem;  
--text-6xl: 3.75rem;
```

Jerarquía de Encabezados

- [h1](#): 2.25rem, font-weight: 800
- [h2](#): 1.875rem, font-weight: 700
- [h3](#): 1.5rem, font-weight: 700
- [h4](#): 1.25rem, font-weight: 700
- [h5](#): 1.125rem, font-weight: 500
- [h6](#): 1rem, font-weight: 500

3. Botones

Estilo Base

```
.btn {  
  padding: 0.75rem 1.5rem;  
  border-radius: var(--radius-md);  
  font-family: var(--font-sans);  
  font-weight: 600;  
  transition: all 0.3s ease;  
}
```

Variantes

- **Primario**
Fondo: `var(--primary-color)`
Texto: blanco
Borde: ninguno
- **Secundario**
Fondo: transparente
Texto: `var(--primary-color)`
Borde: `1px solid var(--primary-color)`

- **Glass (vidrio esmerilado)**

```
.btn-glass {  
  background: rgba(255, 255, 255, 0.1);  
  backdrop-filter: blur(10px);  
  border: 1px solid rgba(255, 255, 255, 0.2);  
  color: var(--light-color);  
}
```

Estados

- `:hover`: Elevación suave (`transform: translateY(-2px)`)
- `:active`: Presionado (`transform: translateY(0)`)

4. Espaciado

Sistema de Espaciado

```
--space-xs: 0.2rem;  
--space-sm: 0.4rem;  
--space-md: 0.8rem;  
--space-lg: 1.2rem;  
--space-xl: 1.6rem;  
--space-2xl: 2rem;  
--space-3xl: 2.4rem;
```

5. Sombras

```
--shadow-sm: 0 1px 2px 0 rgb(0 0 0 / 0.05);  
--shadow: 0 1px 3px 0 rgb(0 0 0 / 0.1), 0 1px 2px -1px rgb(0 0 0 / 0.1);  
--shadow-md: 0 4px 6px -1px rgb(0 0 0 / 0.1), 0 2px 4px -2px rgb(0 0 0 / 0.1);
```

6. Bordes

```
--radius-sm: 0.25rem;  
--radius-md: 0.375rem;  
--radius-lg: 0.5rem;  
--radius-full: 9999px;
```

7. Transiciones

```
--transition-all: all 0.3s ease;
```

8. Puntos de Ruptura (Breakpoints)

- sm: 640px
- md: 768px
- lg: 1024px

9. Componentes Específicos

Tarjetas

- Fondo: `var(--surface-color)`
- Borde: `1px solid var(--gray-200)`
- Sombra: `var(--shadow)`
- Bordes: `var(--radius-lg)`

Formularios

- Inputs con borde sutil
- Estados de validación (éxito, error) con colores de feedback
- Etiquetas claras y mensajes de ayuda accesibles

Navegación

- Barra superior fija
- Efecto glass en modo móvil
- Transiciones suaves para menús desplegables

10. Accesibilidad

- Contraste adecuado entre texto y fondo
- Estados de foco visibles
- Navegación por teclado en todos los componentes interactivos

5. Usabilidad y experiencia de usuario

- **Accesibilidad:** navegación por teclado y etiquetas ARIA.
- **Adaptabilidad:** diseño responsive mediante media queries.
- **Flujo de usuario optimizado:** desde la búsqueda de fechas hasta el pago.
- **Errores controlados:** mensajes claros en validaciones de formularios.

6. Pruebas realizadas

A lo largo del desarrollo he llevado a cabo distintos tipos de pruebas para asegurar el correcto funcionamiento de la web y una buena experiencia de usuario.

Pruebas unitarias: he probado funciones individuales del backend, como la validación de datos y la gestión de rutas protegidas.

Pruebas de integración: he verificado que los distintos componentes (backend, base de datos y pasarela de pagos PayPal) funcionaran correctamente de forma conjunta.

Pruebas funcionales: probé las funcionalidades principales (registro, login, reservas, opiniones, etc.) simulando casos reales.

Pruebas de usabilidad: usuarios reales probaron la web y, a partir de sus comentarios, he podido mejorar algunos aspectos, como algunos estilos de la web.

Pruebas de rendimiento: analicé los tiempos de carga y respuesta, optimizando algunos aspectos como el uso de imágenes y scripts.

Estas pruebas han sido clave para detectar errores, mejorar la calidad de la aplicación y preparar la web para su uso real.

7. Despliegue

7.1 Proceso

- Subida del repositorio a GitHub.
- Configuración de variables en `.env`.
- Creación de cuenta MongoDB Atlas y Render.
- Despliegue continuo con GitHub → Render.

7.2 Documentación

- Manual de usuario final en PDF.
- Documentación técnica (API REST, estructura del proyecto).
- Readme en el repositorio.

8. Conclusiones y futuras mejoras

La creación de esta página web para el alojamiento rural Heranika resultó ser una vivencia enriquecedora, tanto en lo técnico como en lo personal. Desde el inicio, tenía una meta clara: construir una plataforma práctica que facilitara las reservas, los pagos y la comunicación entre el alojamiento y los huéspedes. Hoy, puedo confirmar que he alcanzado satisfactoriamente los objetivos principales.

El sitio web brinda la oportunidad de consultar información detallada del alojamiento, verificar la disponibilidad a través de un calendario interactivo, hacer reservas, pagar de manera segura mediante PayPal, registrarse, acceder a la cuenta personal, publicar reseñas y administrar las reservas propias. Para lograrlo, se empleó un conjunto de tecnologías moderno y apropiado: HTML, CSS y JavaScript en el frontend, Node.js en el backend, y MongoDB como base de datos. Se integró la API de PayPal para posibilitar pagos seguros y profesionales.

Durante el desarrollo, tuve varios desafíos. Uno de los más importantes fue la configuración del despliegue en Render, ya que aparecieron problemas técnicos relacionados con la gestión de rutas y la conexión con la base de datos. Aunque el despliegue aún no está completamente resuelto, estoy analizando diferentes alternativas y ajustes de configuración para encontrar una solución estable y funcional. También surgieron dificultades con la integración de PayPal, sobre todo en el entorno de pruebas (sandbox), debido a errores de validación de los datos que devolvía la API. Estos se resolvieron después de varias pruebas, depuración y revisión de la documentación oficial.

En el plano personal, este proyecto ha sido muy motivador. Saber que el desarrollo se aplicará en un caso real y cercano aumentó mi compromiso y me ayudó a consolidar los conocimientos adquiridos durante el ciclo. También me sirvió para mejorar mis habilidades en la resolución de problemas reales y el trabajo autónomo, enfrentándome a situaciones similares a las que pueden surgir en un entorno laboral.

Con respecto a las mejoras y ampliaciones futuras, se proponen varias funcionalidades que agregarían un valor significativo a la plataforma:

Editar el perfil de usuario: permitir a los usuarios registrados modificar su información personal (nombre, correo, contraseña, añadir una imagen de perfil).

Cancelar reservas: incluye una opción para que los usuarios puedan cancelar sus reservas bajo ciertas condiciones, lo que incrementa la flexibilidad y mejora la experiencia del cliente.

Notificaciones por correo electrónico: implementar un sistema de envío de correos automáticos después de la confirmación o cancelación de una reserva, para mantener al usuario informado en todo momento.

Panel de administración: crear una interfaz específica para el propietario, desde la cual pueda gestionar reservas, editar información de la finca, visualizar estadísticas y responder a opiniones.

Internacionalización: incorporar soporte multilingüe para llegar a un público más amplio, sobre todo turistas extranjeros.

Cambiar a plataformas de desarrollo actuales: pensar en emplear React o Next.js para optimizar la eficiencia de la interfaz y promover la ampliación del proyecto.

Finalmente, el proyecto ha cumplido las metas propuestas y está listo para seguir expandiéndose. Se trata de una alternativa genuina, práctica y con mucho margen de mejora, acorde con las exigencias del turismo rural y las costumbres digitales de hoy.

8. Bibliografía

Enlaces web

Developer PayPal. (n.d.). API Overview. <https://developer.paypal.com/docs/api/overview/>

GetBootstrap. (n.d.). Bootstrap Documentation. <https://getbootstrap.com/>

GitHub. (n.d.). GitHub: Where the world builds software. <https://github.com/>

Node.js. (n.d.). Node.js Documentation. <https://nodejs.org/docs/latest/api/>


npm. (n.d.). npm - Node Package Manager. <https://www.npmjs.com/>

Reddit. (n.d.). Reddit. <https://www.reddit.com/>

W3Schools. (n.d.). W3Schools Online Web Tutorials. <https://www.w3schools.com/>

Videos informativos



Midudev. (2024, junio 13).

 Aprende Autenticación de Usuario, Sesión, Cookies y JWT con Node.js

Midudev. (2024, junio 20).

 CURSO DE CSS COMPLETO: De Principiante a Front-End 2024

Estigarribia Canese, R. (2025, abril 26).

  Cómo Crear un Calendario Interactivo con FullCalendar y NodeJS | Tutorial Paso a Paso

