PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

Unidad Académica de Formación Técnica y Tecnológica – PUCE TEC

APLICATIVO WEB PARA LA GESTIÓN DE RESERVAS DE HOSPEDAJES Y TOURS PARA TENAADVENTURE

Proyecto de titulación previo a la obtención del título de: Tecnología Superior en Desarrollo de Software

Autor: Jefferson Bryan Almeida Criollo

Tutor: Ing. Christian Roberto Tapia Gaibor

Quito, Ecuador

2025

Tabla de contenidos

Lista de Tablas

Lista de Ilustraciones

DECLARACIÓN y AUTORIZACIÓN

Yo, Jefferson Bryan Almeida Criollo con C.I. 1500640147 autor del trabajo de

Titulación: “Desarrollo de un Aplicativo Web para la Gestión de Reservas de Hospedajes y Tours para TenaAdventure”, previa a la obtención del título de Tecnólogo Superior en Desarrollo de Software en la Unidad Académica de Formación Técnica y Tecnológica PUCE TEC:

1.- Declaro tener pleno conocimiento de la obligación que tiene la Pontificia Universidad Católica del Ecuador, de conformidad con el artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de graduación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la Pontificia Universidad Católica del Ecuador a difundir a través de sitio web de la Biblioteca de la PUCE el referido trabajo de titulación, respetando las políticas de propiedad intelectual de Universidad.

Quito, febrero 2026

Jefferson Bryan Almeida Criollo

C.I. 1500640147

Agradecimientos Jefferson Almeida

Quiero expresar mi agradecimiento a mi madre que es mi pilar fundamental, siempre me ha apoyado en mis sueños y anhelos, es la persona más importante en mi vida, además quiero agradecerme a mí por no rendirme y persistir en mis sueños, Never Give Up.

Introducción

El proyecto de titulación consiste en el desarrollo y construcción de una página web denominada reservebnb para TenaAdventure, es una empresa de hospedajes y turismo que funciona en las ciudades de Tena, Archidona y Ambato.
Este proyecto busca crear un aplicativo web para la gestión de reservas de hospedajes y tours mediante CRUD, con 2 roles diferenciados como administrador y huésped, además de un dashboard de administración donde pueda aprobar o rechazar las solicitudes.

El proyecto busca fortalecer la presencial digital de la empresa TenaAdventure, ya que actualmente depende de plataformas externas como Airbnb para poder hacer las reservas de los 15 hospedajes que manejan.

Al realizar este aplicativo web se busca obtener autonomía y control sobre sus reservas, además de mayor visibilidad en la red, esto le permite tener una ventaja competitiva y atraer más clientes nacionales e internacionales.

Visto desde el campo técnico, el proyecto es una solución efectiva que aplica tecnologías web modernas para resolver un problema real de la empresa TenaAdventure, esto ayuda a promover el desarrollo local de la empresa TenaAdventure. Para el frontend se va a utilizar lenguajes como HTML, CSS y JavaScript vanilla para una interfaz simple y responsiva, en el backend se usará Node.js y Express para una API REST robusta y para la base de datos se utilizará PostgreSQL como base de datos relacional, además se empaquetará el proyecto en Docker para contenedores y mayor portabilidad. También se utilizará GitHub para el manejo y control de versiones y GitHub Proyects para la planificación y desarrollo del proyecto.
Además se propone Render para el despliegue a producción del aplicativo web.
Estos lenguajes aseguran un desarrollo ágil, seguro y escalable, con énfasis.
Tambien se utilizará la autenticación JWT y manejo de emails para solicitudes, siendo un mecanismo óptimo para la empresa.

Este proyecto busca desarrollar un aplicativo web que automatice reservas, integre roles de usuarios como administrador y huésped, y envíe notificaciones por email, ayudando a reducir el tiempo de procesamiento de solicitudes de reservas a minutos.

Con esta plataforma, se soluciona un requerimiento real en el sector turístico ecuatoriano, ayudando a fomentar la inclusión de empresas pequeñas de turismo y hospedajes.

Problema Técnico/Tecnológico

TenaAdventure es una empresa de hospedajes y turismo que opera actualmente en las ciudades de Tena, Archidona y Ambato en el país de Ecuador, la cual enfrenta desafíos en su operación diaria debido a la falta de una plataforma web propia.

En la actualidad gestiona y renta 15 hospedajes a través de la plataforma Airbnb, por la cual no tiene visibilidad exclusiva y esto le genera dependencia de una plataforma externa, en la cual existen otros hospedajes que son su competencia.

Esto le dificultad poder promocionar tours personalizados y hospedajes que tengan sus propias reglas. En el contexto postpandemia el turismo en Ecuador crecio un 15% anual segun

En un contexto socioeconómico donde el turismo post-pandemia en Ecuador crece un 15% anual (según datos del Ministerio de Turismo), las pymes como TenaAdventure necesitan herramientas digitales para captar huéspedes internacionales y locales, pero carecen de presencia en internet más allá de redes sociales esporádicas.

Técnicamente, el problema se manifiesta en la ausencia de un sistema integrado para gestionar reservas de hospedajes y tours, lo que provoca ineficiencias como duplicidad de datos, errores en inventarios (disponibilidad no actualizada en tiempo real), y falta de trazabilidad en aprobaciones. Los huéspedes no tienen un portal para registrar información personal y ver sus reservas, mientras que los administradores manejan todo manualmente, aumentando el riesgo de pérdidas financieras y descontento de clientes. En el ámbito sociocultural, Tena, una zona amazónica con alto potencial turístico, requiere soluciones que promuevan el eco-turismo sostenible, pero sin una web dedicada, TenaAdventure pierde oportunidades de diferenciarse de competidores urbanos.

Objetivos

Objetivo General

Desarrollar una página web para la gestión de reservas de hospedajes y tours de TenaAdventure, integrando tecnologías modernas como Node.js, Express, PostgreSQL y Docker, para optimizar la presencia digital y operaciones de la empresa.

Objetivo Específicos

- Diseñar la arquitectura del sistema incorporando en el frontend lenguajes como HTML, CSS, JavaScript Vanilla, y en el backend lenguajes como Node.js, Express y para la base de datos PostgreSQL, con estos lenguajes de programación se asegura una mayor escalabilidad y seguridad para un crecimiento futuro, además que protege los datos sensibles de los usuarios.
- Implementar funcionalidades CRUD para hospedajes y tours, y la creación de roles de administrador y huésped con funcionalidades personalizadas, mediante autenticación JWT y con notificaciones vía emails para a mejorar la experiencia de usuario, esto nos ayuda a reducir errores y tiempos de respuestas en las solicitudes.
- Probar el sistema integralmente, mediante aprobaciones manuales y notificaciones, con pruebas integrales para backend con el programa Postman y con usuarios de prueba, para revisar que el sistema funcione correctamente y resuelva los problemas como ineficiencia en la gestión de reservas, duplicidad de datos o falta de trazabilidad.

Marco Conceptual

El marco conceptual del proyecto se desarrolló como un web full stack para el sector turístico, que se enfoca en la gestión de reservas y tours. Este aplicativo web nos asegura un sistema seguro, eficiente y adaptado al contexto turístico ecuatoriano, lo que ayuda a promover la digitalización en regiones amazónicas como el Tena.

Aplicativo Web Full-Stack: Se refiere a un sistema completo que integra frontend para la interfeaz del usuario y backend para logica del servidos y base de datos. El frontend usa HTML para estructura, CSS para estilos responsivos, y JavaScript vanilla para interacciones dinámicas como fetch a APIs, y evitamos frameworks pesados para mantener simplicidad en el proyecto.

Se elige esta forma de desarrollo ligero y accesible, porque es ideal para pymes con recursos limitados, ya que facilita una interfaz intuitiva y sin sobrecarga de dependencias.

Node.js y Express: Node.js es un entorno de ejecución JavaScript en servidor, ideal para apps escalables debido a su modelo asíncrono y no bloqueante (Banks, 2020). Express, como framework minimalista, facilita la creación de APIs REST, manejando rutas, middleware (ej., autenticación JWT) y respuestas JSON (Brown, 2019). En nos permite procesar reservas en tiempo real, con aprobaciones manuales vía endpoints protegidos. Estas tecnologías se eligen por su eficiencia en manejo de alto tráfico variable, además de su integración nativa con JavaScript, lo que reduce la curva de aprendizaje y costos de desarrollo.

PostgreSQL: Base de datos relacional open-source, elegida por su robustez en manejo de transacciones ACID, consultas complejas y escalabilidad (Hollemans, 2023). Soporta tablas para usuarios y administradores.

Se elige sobre otras bases por su soporte gratuito y capacidad para manejar relaciones complejas en sistemas relacionales, esto nos asegura la integridad y eficacia en el manejo de datos.

Docker: Herramienta de contenedores que empaqueta la app (frontend con Nginx, backend Node, DB PostgreSQL) en entornos aislados, asegurando consistencia entre desarrollo y producción (Wilson, 2022). Docker Compose no ayuda a desplegar en Render el proyecto ya que orquesta servicios con dependencias. Además, Docker es portable, y permite despliegues rápidos en entornos cloud incompatibles, esto es ideal para regiones rurales con infraestructura limitada.

JWT (JSON Web Tokens): Mecanismo de autenticación stateless para roles: admins acceden a CRUD y aprobaciones, huéspedes a dashboards. Integra con Express middleware para rutas protegidas (Nguyen, 2024). Es seguro y eficiente con autentificaciones en APIs REST.

Gestión de Reservas en Turismo: Se enfoca en hacer CRUD para recursos de hospedajes y tours, ademas de aprobaciones manuales y notificaciones via email.

Ayudando a reducir la dependencia de plataformas externas como airbnb.

Despliegue en Render: Plataforma PaaS que soporta Docker, GitHub integration, y escalado automático, ideal para pymes turísticas con tráfico variable (Wilson, 2022). Es de fácil instalación y uso gratuito para mantener costos bajos.

Metodología Scrum-like: El enfoque metodológico Scrum-like se basa en iteraciones cortas denominadas sprints, reuniones diarias conocidas como daily stand-ups, revisiones de sprint para evaluar avances y retrospectivas destinadas a mejorar continuamente el proceso. Este método se apoya en un product backlog gestionado a través de GitHub Projects, utilizando user stories como artefactos principales para priorizar y organizar tareas (Schwaber & Sutherland, 2020).

La elección de este método se debe a su adaptabilidad y flexibilidad a la evolución constante de requisitos en proyectos de software, permitiendo ajustes según el feedback que se realizan en un plazo determinado de realización. Además, fomenta la colaboración en equipo y entregas incrementales. GitHub Projects se utilizará para la planificación, seguimiento y asignación de tareas, integrado en tableros Kanban para visualizar el progreso y gestión eficiente del proyecto.

CAPÍTULO I

Levantamiento de Requisitos y Diseño del Sistema

1.1 Antecedentes

El constante crecimiento de la tecnología y ventas en línea ha impulsado a las industrias de hospitalidad a incorporar herramientas tecnológicas para no quedarse atrás de su competencia y poder seguir manteniendo una ventaja competitiva que le permita mejorar el manejo y comunicación con sus clientes.

En Ecuador, particularmente en empresas pequeñas aún se utilizan procesos manuales o plataformas externas para el manejo de sus operaciones.

En este contexto, TenaAdventure vio limitada su capacidad de gestión debido a la falta de una plataforma web propia donde pueda centralizar y gestionar directamente todas sus reservas de tours y alojamientos, por lo cual se dio la necesidad de desarrollar la plataforma web reservebnb para optimizar los procesos internos y presencia digital de la empresa.

1.2 Metodología de desarrollo Scrum

Se aplicó la metodología ágil basada en Scrum para desarrollar el aplicativo web ReserveBnB, se adaptó para un entorno de desarrollo individual. La metodología Scrum se utiliza mucho en proyectos de software, esta nos permite ir entregando funcionalidades de forma incremental y adaptándolas en el avance del proceso.

Con la metodología Scrum se organizó el trabajo en sprints definidos, priorizando las funcionalidades más importantes y haciendo un seguimiento mediante de las tareas realizas y por realizar. Este proyecto fue desarrollado por una sola persona, que utilizó una forma adaptada de Scrum ajustando los roles, pero conservando la esencial de la metodología.

La organización y el seguimiento de los springs, e historias de usuario se realizó a través de GitHub Projects, mediante el tablero Kanban donde se visualiza el estado de las

tareas en tres columnas principales: pendiente (Todo), en curso (In progress) y finalizado (Done).

1.2.1 Roles de la metodología Scrum:

Product Owner: Representado por la gerente del proyecto empresarial TenaAdventure, quien definió los requerimientos, necesidades y funcionalidades requeridas para su negocio.

Scrum Máster: El tutor académico fue quien cumplió esta función, orientando el proceso metodológico y ayudando al cumplimiento de los objetivos establecidos.

Equipo de desarrollo: Está integrado por el autor del proyecto, quien desarrolló el diseño, implementación, pruebas y documentación del sistema.

El desarrollo se planificó por módulos funcionales denominados sprint, esta estrategia permitió ir programando el sistema por etapas, lo que ayuda a revisar el cumplimiento de cada módulo antes de pasar al siguiente.

1.3 Identificación de actores del sistema

Los actores se consideran las personas o cosas externas que participan en el proceso del manejo y funcionamiento del sistema.

Actor Administrador

El Administrador es el actor principal en el manejo de la plataforma web por lo general es el propietario o encargado de administrar el negocio TenaAdventure, tiene acceso a herramientas más avanzadas en el aplicativo, que ingresa mediante acceso exclusivo de su usuario y contraseña.

Las obligaciones principales del administrador son:

- Registrar, editar y eliminar hospedajes disponibles.
- Registrar, editar y eliminar tours.
- Visualizar todas las solicitudes de reserva generadas por los huéspedes.
- Aprobar o rechazar solicitudes de reserva según la disponibilidad y criterios del negocio.

Actor Huésped

El Huésped es el cliente final que utiliza la página web para mirar qué opciones de alojamiento y tours hay disponibles y realiza las solicitudes dependiendo de sus gustos e intereses.

Las principales acciones que puede realizar el huésped son:

- Registrarse llenando un formulario con información básica de contacto.
- Iniciar sesión para poder realizar solicitudes de reserva.
- Revisar la lista de alojamientos y tours disponibles.
- Enviar solicitudes de reserva que consta de fechas, número de personas y tipo de servicio.
- Revisar el estado de sus solicitudes que pueden ser pendiente, aceptada, rechazada.

Actor Programador

- Es el encargado de manejar la actualización, mejoras y mantenimiento del sistema.

1.4 Historias de usuario

Las historias de usuario que se utilizaron para el desarrollo del aplicativo web reservebnb son las siguientes:

Tabla 1

Historias de usuario del sistema Reservebnb

Código	Historia de Usuario
HU-01 – Registro de usuarios	Como huésped, quiero registrarme en la plataforma, para poder solicitar las reservas de hospedajes y tours.
HU-02 – Inicio de sesión	Como usuario sea huésped o administrador, quiero iniciar sesión de forma segura, para acceder a mis recursos.
HU-03 – Visualización de hospedajes	Como huésped, quiero visualizar los hospedajes disponibles, para reservar el que desee.
HU-04 – Visualización de tours	Como huésped, quiero visualizar los tours disponibles, para reservar el que necesite.
HU-05 – Detalle de hospedaje o tour	Como huésped, quiero ver el detalle de un hospedaje o tour, para conocer toda su información.
HU-06 – Solicitud de reserva	Como huésped, quiero enviar una solicitud de reserva de hospedaje o tour, para reservar según mis intereses.
HU-07 – Historial de solicitudes	Como huésped, quiero visualizar el historial de mis solicitudes, para conocer su estado de pendiente, aceptada o rechazada.
HU-08 – Gestión de solicitudes de reserva	Como administrador del negocio, quiero aceptar o rechazar las solicitudes de reserva, para controlar la disponibilidad.
HU-09 – Notificaciones por correo electrónico	Como huésped, quiero recibir notificaciones por correo electrónico, para conocer el estado de las reservas.

HU-10 – Gestión de hospedajes	Como administrador, quiero crear, editar o eliminar hospedajes, para mantener actualizado el inventario.
HU-11 – Gestión de tours	Como administrador, quiero crear, editar y eliminar tours, para mantener actualizado las experiencias turísticas.
HU-12 – Panel administrativo	Como administrador, quiero acceder a un panel administrativo, para gestionar hospedajes, tours y solicitudes de reserva.
HU-13 – Control de acceso por roles	Como administrador, quiero que el sistema permita el acceso según roles, para permitir el acceso a las funciones específicas de cada rol.
HU-14 – Cierre de sesión	Como usuario administrador o huésped, quiero cerrar sesión de forma segura, para proteger mi información personal.

1.5 Casos de uso del sistema

Los casos de uso describen la manera en que interactúan los actores con el software web, es decir permiten describir de manera más detallada el comportamiento del sistema cuando los usuarios utilizan el aplicativo, en nuestro caso se enfoca en la gestión de usuarios, alojamientos y tours.

Caso de uso 1: Registro de usuario

Actor: Huésped
Descripción: El usuario se registra en el sistema mediante la introducción de sus datos personales.
Precondición: El usuario no debe estar registrado previamente.
Postcondición: El usuario se encuentra registrado.

Flujo principal:

1. El usuario llena al formulario de registro.
2. Proceda a introducir nombre, dirección de correo electrónico, contraseña e información de contacto.

3. El sistema valida la información ingresada.
4. El sistema guarda la información en la base de datos.
5. El sistema valida que el registro se haya realizado correctamente.

Caso de uso 2: Inicio de sesión

Actor: Huésped o Administrador
Descripción: Los usuarios inician sesión con sus credenciales seguras, mediante usuario y contraseña.

Precondición: El usuario debe estar registrado.
Postcondición: El usuario tiene acceso a enviar solicitudes de reserva.
Principal Flujo:

1. El usuario ingresa su correo electrónico y contraseña.
2. El sistema procede a verificar de credenciales.
3. El sistema produce un token para la autenticación del usuario.
4. El usuario ingresa al sistema.

Caso de uso 3: Gestión de hospedajes

Actor: Administrador
Descripción: El administrador gestiona los alojamientos del sistema.
Precondición: El administrador debe estar logueado.
Postcondición: Puede hacer CRUD en los hospedajes.
Principal Flujo:

1. El gerente ingresa al panel administrativo.
2. Visualiza, crea, edita o elimina alojamientos.
3. El sistema procesa la solicitud y actualiza la base de datos.

Caso de uso 4: Gestión de tours

Actor: Administrador

Descripción: El administrador gerencia los tours del sistema.

Precondición: El administrador debe estar logueado.

Postcondición: Puede hacer CRUD en los tours.

Flujo Principal:

1. El gerente ingresa al panel de administración.
2. Visualiza, crea, edita o elimina alojamientos.
3. El sistema procesa la solicitud y actualiza la base de datos.

Caso de uso 5: Petición de reserva

Actor: Huésped

Descripción: El huésped envía una solicitud de reserva.

Precondición: El usuario debe estar logueado.

Postcondición: La solicitud de reserva se registra de inicio en estado pendiente.

Flujo principal:

1. El huésped elige un alojamiento o tour.
2. Introduce las fechas y número de personas.
3. El sistema procesa la petición.
4. El sistema asigna el estado de pendiente.

Caso de uso 6: Aprobación o rechazo de solicitudes

Actor: Administrador

Descripción: Permite al administrador aprobar o rechazar solicitudes de reserva.

Precondición: El administrador debe estar autenticado.

Postcondición: La solicitud se actualiza a aprobado o rechazado.

Flujo principal:

1. El gerente revisa las solicitudes de los huéspedes.
2. Elige una solicitud y acepta o rechaza.
3. El sistema actualiza el estado en la base de datos.

1.6 Etapas del proceso

1.6.1 Levantamiento de requerimientos

Se realizo múltiples entrevistas con la Product Owner tanto presenciales como virtuales en las ciudades donde maneja los hospedajes y tours, donde nos explicó el giro de su negocio para poder entender cuál es la mejor solución a sus necesidades.

También se realizó varias reuniones virtuales donde nos enseñó el manejo de la plataforma Airbnb de la cual depende para poder alquilar sus hospedajes, con esto entendimos mejor cuales son los puntos débiles que tiene su negocio y ofrecerle tener un aplicativo web propio.

1.6.2 Análisis del problema y propuesta de solución

Con la información obtenida de la empresa TenaAdventure, se determinó que el problema principal que limita su crecimiento es la dependencia de las plataformas externas para la gestión de reservas de hospedajes y tours, lo que no le permite tener un buen control sobre el manejo de sus hospedajes, además de manejar la mayoría de sus reservas de tours mediante el aplicativo WhatsApp, esto le impide tener un historial ordenado, generándole demoras y perdida en la autonomía del negocio.

Como solución, se propone el desarrollo de un aplicativo web denominado reservebnb, que permite centralizar la información de hospedajes y tours, ayudando a gestionar las solicitudes de reservas con estados de pendiente, aprobado y rechazado, mediante un panel administrativo donde controle con operaciones CRUD el inventario de hospedajes y tours, creando así un manejo eficiente de las operaciones, mejorando la comunicación y tiempo de respuesta con los clientes, además de obtener presencia digital de la empresa.

1.6.3 Diseño del sistema

En esta etapa se diseñó la arquitectura general del sistema, la cual se basa en una arquitectura cliente servidor que se comunica a través de una API REST.

Los actores principales del sistema en los que se dividió son:

1. Frontend se utilizó HTML, CSS y JavaScript sin frameworks.
2. Backend para la lógica del negocio se usó Node.js y Express.
3. Base de datos se utilizó una base relacional PostgresSQL para guardar los datos.

También en la programación se utilizó Docker, Github y Github Project.

En esta fase, se establecieron la lógica del negocio y las tablas de la base de datos, las rutas del backend, la autenticación y permisos de acceso según rol del usuario.

1.6.4 Pruebas del sistema

Se realizaron pruebas funcionales y técnicas después de cada sprint de forma continua para verificar el correcto funcionamiento del proyecto.

Las pruebas comprenden la verificación de la autenticación, la administración de alojamientos y tours, el envío y aprobación de solicitudes de reserva, así como la interacción entre el frontend y el backend.

Además, se realizó pruebas con Postman para probar las APIs del backend, los métodos usados fueron GET, POST, PUT y DELETE.

También se aplicó pruebas de seguridad mediante autentificación JWT, en este proyecto, el backend crea el token con el id y rol del usuario, el frontend lo envía en el header Authorization, y el servidor lo valida con una clave secreta para permitir o negar acciones, por ejemplo, en el caso del administrador si sus credenciales son válidas puede gestionar los hospedajes y tours.

1.7 Creación y gestión del backlog

La creación del backlog se realizó en Github Project en base a las historias de usuario, aquí se planifico las tareas técnicas de desarrollo del proyecto denominado issues.

El backlog se dividió en 7 sprints conformado por 16 issues que facilita la organización, priorización y seguimiento de todas las funcionalidades para lograr el objetivo del proyecto.

Dentro del backlog se dividió en tareas pendientes, en ejecución y finalizadas lo que nos permitió tener una visión clara del estado general del proyecto.

Los estados utilizados en el backlog fueron:

- **To Do:** Issues que todavía falta por hacer.
- **In Progress:** Issues en curso.
- **Done:** Issues finalizados según sus criterios de aceptación.

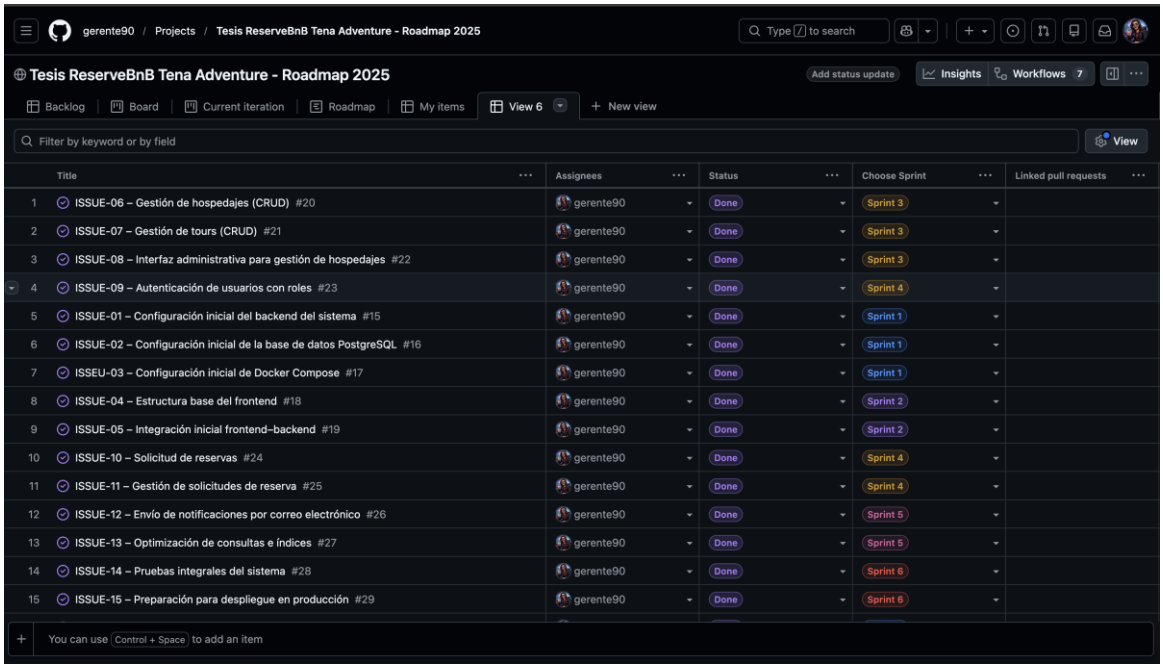


Ilustración 1. Backlog del sistema Reservebnb

1.8 Viabilidad del proyecto

En la viabilidad del proyecto se analizaron los aspectos técnicos, económicos y operativos, para evidenciar que la solución propuesta pueda ser desarrollada, mantenida y utilizada de manera efectiva.

1.8.1 Viabilidad técnica.

El sistema es viable técnicamente debido al uso de tecnologias modernas y de codigo abierto.

Los lenguajes utilizados son en el Backend en Node.js con Express, para el Frontend en HTML, CSS, JavaScript y para la base de datos PostgreSQL relacional.

La comunicación entre el backend y fronted se realizó a través de una API REST de forma transparente, segura y escalable.

Para tener seguridad en los inicios de sesión se basó en JSON Web Tokens (JWT).

Además aplicamos Docker Compose en el frontend, backend y base de datos para automatizar y simplificar el entorno.

1.8.2 Viabilidad económica.

Desde la perspectiva económica, el proyecto resulta viable, porque se usó tecnologías de código abierto, y esto redujo los costos de licencias de software.

También la elaboración del proyecto fue realizada por el autor, evitando los gastos de contratación de personal externo.

Además, con un sistema de gestión de reservas propio se puede optimizar sus precios.

1.8.3 Viabilidad operativa.

El sistema es viable operativamente ya que el personal de TenaAdventure logro manejar con normalidad la plataforma Reservebnb, es decir el diseño desarrollado permitió una navegación natural e intuitiva, además al programar un panel administrativo donde el gerente maneja el control de la página, le volvió facil, cómodo y optimo de usar.

1.9 Planificación de los Sprints

En el desarrollo del sistema se definieron siete sprints, comprendidos entre el 13 de octubre de 2025 y el 08 de febrero de 2026 con una duración de 2 semanas cada uno. Con esta planificación se organizó de forma óptima las actividades del proyecto y contribuyó a asegurar la realización de los objetivos planteados.

Todos los sprint fueron gestionados mediante GitHub Projects, en esta herramienta también se creó los issues en base a las historias de usuario, con esto se visualizó el avance de las tareas y se controló el estado de cada actividad.

Al cierre de cada sprint, las funcionalidades implementadas fueron revisadas por el Scrum Master y una vez cumplidos los criterios de Definition of Done (DoD) se dio por finalizado.

Los Sprint utilizados en el proyecto fueron los siguientes:

Sprint 1 Configuración del Sistema
Fechas: 13 de octubre de 2025 a 20 de octubre de 2025

En este sprint se abordaron las tareas relacionadas con la configuración técnica inicial del sistema. Se desarrolló la estructura base del backend, se configuró la base de datos PostgreSQL para tener la lógica del negocio y se orquestaron los servicios mediante Docker Compose, logrando así las bases técnicas necesarias para el desarrollo del sistema.

Sprint 2 Estructura base del frontend e integración
Fechas: 21 de octubre de 2025 a 03 de noviembre de 2025

En este sprint se desarrolló la estructura base del frontend y se llevó a cabo la primera integración con el backend a través de la API REST. Estas actividades permitieron validar la comunicación entre los componentes del sistema y habilitar las primeras interacciones con los usuarios.

Sprint 3 Gestión de hospedajes y tours
Fechas: 04 de noviembre de 2025 a 17 de noviembre de 2025

Durante este sprint se implementaron las funcionalidades principales que son las que están asociadas a la gestión de hospedajes y tours. Además, se desarrollaron los módulos CRUD correspondientes y las interfaces administrativas necesarias para que el gerente o administrador logre gestionar el catálogo de alojamientos y experiencias turísticas.

Sprint 4 Autenticación y flujo de reservas
Fechas: 18 de noviembre de 2025 a 08 de diciembre de 2025

Este sprint se enfocó en la implementación del sistema de autenticación de usuarios con roles diferenciados entre administrador o huésped y en la consolidación del flujo principal de solicitudes de reserva. También se habilitó a los usuarios huéspedes la posibilidad de realizar solicitudes y a los administradores la capacidad de aprobarlas o rechazarlas, estableciendo el flujo central del sistema de reservas.

Sprint 5 Servicios adicionales y optimización

Fechas: 09 de diciembre de 2025 a 05 de enero de 2026

En este sprint se integraron servicios adicionales, como el envío de notificaciones por correo electrónico, y se realizaron optimizaciones orientadas al rendimiento del sistema. Estas acciones ayudan a mejorar la experiencia del usuario y la eficiencia general del software.

Sprint 6 Pruebas y preparación para el despliegue

Fechas: 06 de enero de 2026 a 26 de enero de 2026

Durante este sprint se ejecutaron pruebas funcionales y de extremo a extremo con el fin de verificar el correcto funcionamiento de las principales funcionalidades del sistema. Asimismo, por petición del Product Owner se subió el código completo a Git Hub y se le entrego después de una capacitación presencial.

Sprint 7 Documentación final

Fechas: 27 de enero de 2026 a 08 de febrero de 2026

El sprint final se centró en la elaboración y finalización del presente documento de tesis. Esta etapa permitió concluir el trabajo realizado y dejar documentado formalmente el proceso de desarrollo del sistema Reservebnb.

1.10 Herramientas y Software

En el desarrollo del sistema se usó diversas herramientas de software para la programación, control, pruebas y despliegue del proyecto.

Las herramientas utilizadas fueron:

Gestión del proyecto: GitHub Projects para la planificación del backlog y sprints.

Control de versiones: Se utilizo GitHub.

Backend: Se uso los lenguajes Node.js y Express.js.

Base de datos: Se uso la base relacional PostgreSQL.

Frontend: Se utilizo los lenguajes HTML, CSS y JavaScript.

Contenedores: Para la contenerización se usó Docker y Docker Compose.

Entorno de desarrollo: Se utilizo el IDE Visual Studio Code.

Pruebas de API: Se realizó con Postman.

Herramientas de soporte: Se probó en distintos navegadores web modernos para pruebas de interfaz.

1.10.1 Flujo de trabajo en Git

Se utilizo comandos de Git para el control de versiones y Github para almacenar el código del repositorio.

Dado que el proyecto fue desarrollado por un solo programador, se mantuvo simple y centralizado.

Trabajando principalmente sobre la rama principal (main), en la cual se desarrolló las funcionalidades del sistema Reservebnb en base a lo planificado en GitHub Projects.

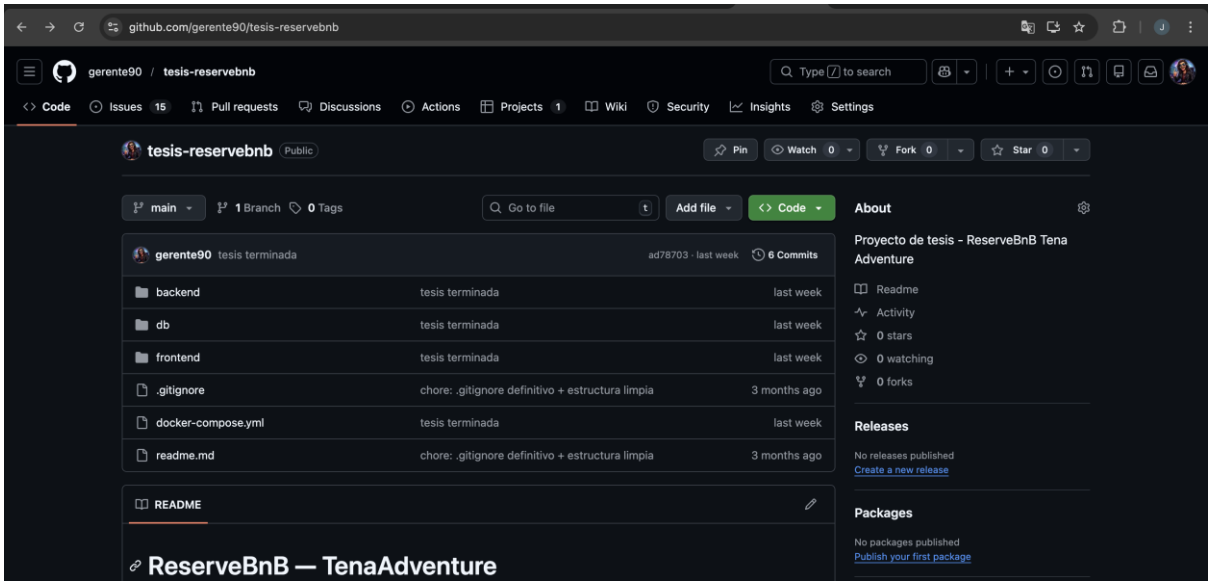


Ilustración 2. Github del sistema Reservebnb

1.11 Recursos de hardware

Requisitos mínimos del equipo son:

- Equipo de cómputo portátil o de escritorio.
- Arquitectura de procesador x64.
- Memoria RAM mínima de 8 GB.
- Almacenamiento mínimo recomendado 120 GB de espacio libre
- Sistema operativo compatible con Node.js, Docker y PostgreSQL como Windows 10, macOs o Linux (Ubuntu 20.04 LTS o superior).

Especificaciones del equipo usado en el proyecto:

- Computador portátil Macbook pro (2025).
- Procesador Apple M4.
- Sistema operativo macOS que es compatible con Node.js, Docker y PostgresSQL.

CAPÍTULO II

Construcción del Sistema

En este capítulo se describe la expansión del sistema ReserveBnB y los aspectos técnicos considerados para su desarrollo. Se detallan las funcionalidades implementadas, las limitaciones del proyecto y las posibles extensiones futuras, con el propósito de definir de manera clara el sistema desarrollado dentro de la modalidad de titulación.

2.1 Alcance

El sistema ReserveBnB se desarrolló como una aplicación web orientada a la gestión de solicitudes de reserva de hospedajes y tours, permitiendo que un administrador realice la aprobación manual de las solicitudes correspondientes a los servicios ofrecidos por TenaAdventure. La plataforma facilita el registro de solicitudes, la asignación de roles y la notificación del resultado al huésped.

El frontend fue implementado utilizando HTML, CSS y JavaScript (vanilla), y cuenta con una interfaz responsiva que permite al huésped enviar solicitudes de reserva y consultar su estado. A su vez, el administrador puede gestionar dichas solicitudes mediante un panel de control.

El backend se desarrolló con Node.js y Express, exponiendo una API REST encargada de la autenticación de usuarios según roles, el registro de solicitudes, la actualización de estados (pendiente, aprobada o rechazada) y la ejecución de las reglas definidas para el flujo de aprobación.

La base de datos se implementó utilizando PostgreSQL, donde se almacena la información correspondiente a usuarios, hospedajes, tours y solicitudes de reserva, junto con su respectivo historial de estados.

El alcance del sistema excluye funcionalidades como transacciones en línea, emisión de facturas, integración automática con calendarios externos (por ejemplo, sincronización directa con Airbnb), automatización total de confirmaciones sin intervención humana y el desarrollo de una aplicación móvil nativa.

No obstante, el sistema fue diseñado con una arquitectura modular, lo que permite la incorporación de nuevos componentes y funcionalidades en el futuro, de acuerdo con las necesidades y crecimiento del negocio.

2.2 Análisis del diseño

El diseño del sistema ReserveBnB se llevó a cabo con base en las especificaciones funcionales definidas en las historias de usuario y en el backlog del proyecto.
Los sprints guiaron el desarrollo de las distintas funcionalidades del sistema, tales como la administración de alojamientos, tours, usuarios y solicitudes de reserva, así como los procesos de aprobación y notificación.

A partir de este análisis, se logró organizar el desarrollo del frontend, backend y base de datos por etapas claramente definidas dentro de cada sprint.

2.3 Mapa de navegación

La interacción en la página web ReserveBnB se divide en dos tipos de usuarios: el huésped (cliente) y el administrador.

Para el cliente, el proceso de navegación inicia en la página principal (index.html), la cual contiene secciones informativas como inicio, habitaciones, tours, servicios, reservaciones y contacto. Desde esta página, el usuario puede revisar los alojamientos y tours disponibles sin necesidad de autenticarse en el sistema.

El usuario tiene la opción de iniciar sesión como huésped o como administrador desde la página principal. Al seleccionar un alojamiento entre las habitaciones disponibles, el sistema redirige al usuario a una página denominada (listing.html), donde se presenta información detallada del hospedaje seleccionado.

El usuario puede alquilar un hospedaje mediante transferencia bancaria, siempre y cuando se encuentre autenticado en la plataforma. Alternativamente, también puede realizar

la reserva a través de la plataforma Airbnb, sin necesidad de iniciar sesión en la página web de ReserveBnB.

Una vez que el usuario huésped se registra y realiza una solicitud de reserva, se habilita una página de historial (historial.html), desde la cual puede dar seguimiento al estado de sus solicitudes de reserva.

Por otro lado, cuando el usuario inicia sesión con el rol de administrador, el sistema lo redirige a la interfaz administrativa (admin.html). Desde esta sección, el administrador puede gestionar el sistema en su totalidad, incluyendo la creación, edición y eliminación de hospedajes y tours, así como la aprobación o rechazo de las solicitudes de reserva pendientes.

De esta manera, el sistema habilita las distintas funcionalidades y páginas de acuerdo con el rol del usuario, garantizando un acceso controlado y una navegación acorde a las responsabilidades asignadas.

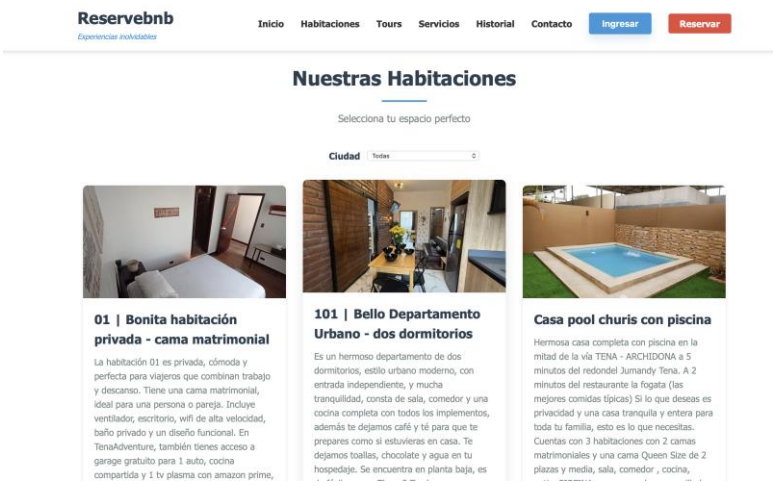


Ilustración 3. Página principal del sistema Reservebnb

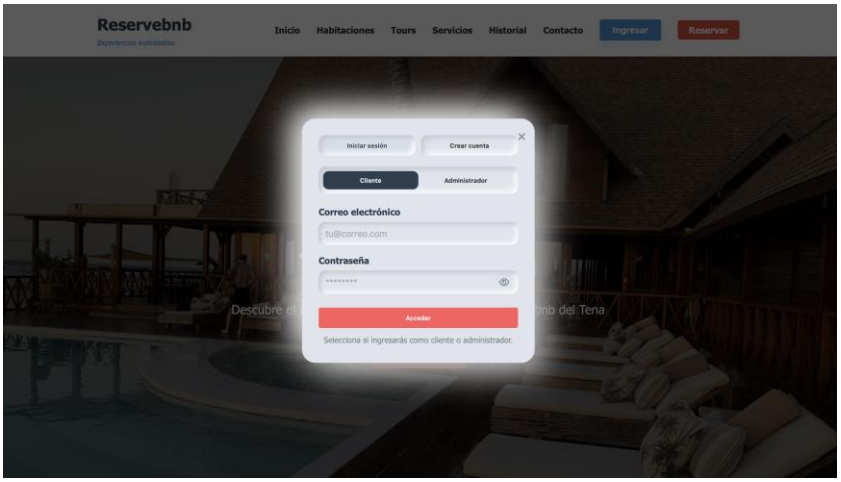


Ilustración 4. Página de login del sistema Reservebnb

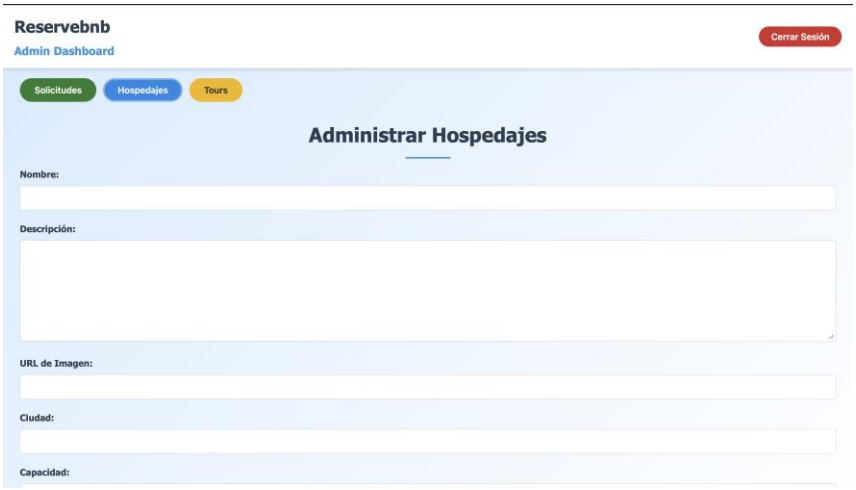


Ilustración 5. Panel administrativo del sistema Reservebnb

2.4 Lógica del negocio

La lógica de negocio del sistema ReserveBnB, desarrollado para la empresa TenaAdventure, se fundamenta en la administración y el control de reservas de hospedajes y tours turísticos, con el objetivo de optimizar los procesos de venta y disponer de una plataforma propia que reduzca la dependencia de servicios externos.

El sistema gestiona el rol de usuario huésped, quien puede autenticarse y realizar solicitudes de reserva de hospedajes o tours. Dichas solicitudes se registran inicialmente con

un estado pendiente y permanecen en ese estado hasta que el administrador las apruebe o rechace. Durante este proceso, el administrador recibe información relevante como fechas solicitadas, cantidad de personas y el recurso requerido.

Por su parte, el usuario administrador dispone de un panel administrativo desde el cual puede aprobar o rechazar solicitudes tanto de hospedajes como de tours. Adicionalmente, cuenta con funcionalidades CRUD que permiten crear, editar y eliminar hospedajes y tours. Una vez que una solicitud es aprobada, el sistema actualiza el estado de la reserva a confirmada.

La estructura lógica del sistema se apoya en una API REST, desarrollada con Node.js y Express, la cual responde a las necesidades operativas del negocio, valida los datos enviados desde el frontend y gestiona el acceso a los distintos endpoints mediante un sistema de autenticación basado en JSON Web Tokens (JWT).

Este enfoque garantiza que únicamente los usuarios autenticados y autorizados puedan ejecutar operaciones críticas dentro del sistema, de acuerdo con el rol asignado.

Además, el sistema integra un módulo de notificaciones por correo electrónico que informa a los usuarios sobre el estado de sus solicitudes de reserva, mejorando la comunicación entre la plataforma y los clientes.

La información se almacena y administra a través de una base de datos PostgreSQL, organizada en tablas relacionales que gestionan usuarios, alojamientos, tours, solicitudes y reservas, lo que asegura la integridad y consistencia de los datos conforme a la lógica del negocio.

En resumen, la lógica de negocio de la plataforma ReserveBnB contribuye a optimizar los procesos internos de TenaAdventure, minimizar errores operativos, mejorar la experiencia de los huéspedes y reducir la dependencia de plataformas externas. Asimismo, el sistema fue diseñado con una arquitectura escalable que permite futuras mejoras y ampliaciones.

2.5 Modelo de base de datos (BDD)

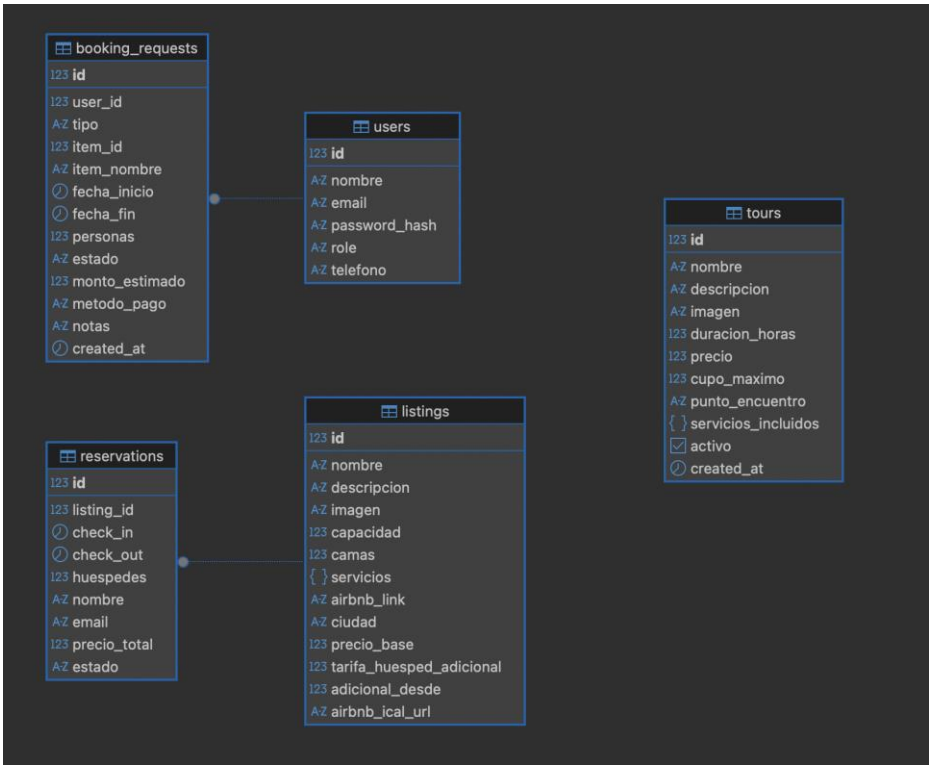


Ilustración 6. Diagrama de la base de datos

En el modelo de la base de datos podemos observar claramente las tablas y atributos de la lógica del negocio.

1. **Tabla users:** Sirve para guardar los usuarios clientes y administradores.
   Atributos: id, nombre, email, password_hash, role, telefono.
   Detalle de los atributos:
   id: identificador único del usuario.
   nombre: nombre completo del usuario.
   email: correo usado para login.
   password_hash: contraseña encriptada.
   role: rol del usuario puede ser admin o huesped.
   telefono: número de contacto es opcional.
   Relaciones:
   Usuario (1) a booking_request (N)

Un usuario puede crear muchas solicitudes en booking_request.
Es decir, es de tipo de 1 a muchos.

2. **Tabla listings:** Sirve para guardar los hospedajes publicados.
   Atributos: id, nombre, descripcion, imagen, capacidad, camas, servicios, airbnb_link, ciudad, precio_base, tarifa_huesped_adicional, adicional_desde, airbnb_ical_url.
   Detalle de los atributos:
   id: identificador único del hospedaje.
   nombre: nombre comercial del hospedaje.
   descripcion: descripción larga.
   imagen: URL de imagen principal.
   capacidad: número máximo de huéspedes.
   camas: número de camas (puede ser distinto de capacidad).
   servicios: lista de servicios incluidos como WiFi, cocina, etc.
   airbnb_link: link directo al anuncio de Airbnb.
   ciudad: ciudad donde está el hospedaje.
   precio_base: precio por noche base.
   tarifa_huesped_adicional: extra por huésped adicional.
   adicional_desde: desde qué número de huéspedes se cobra extra.
   airbnb_ical_url: link iCal para sincronizar disponibilidad.
   Relaciones:
   Listings (1) a reservations (N)
   Un hospedaje puede tener muchas reservas directas en reservations (listing_id)
   Es decir, es de tipo 1 a muchos.
   Listings (1) a booking_requests(N)
   También puede ser referenciado en booking_requests cuando el tipo es listing.
   Es decir, es de tipo 1 a muchos.

3. **Tabla tours:** Sirve para guardar los tours y experiencias.
   Atributos: id, nombre, descripcion, imagen, duracion_horas, precio, cupo_maximo, punto_encuentro, servicios_incluidos, activo, created_at.
   Detalle de los atributos:
   id: identificador único del tour.

nombre: nombre del tour.

descripcion: detalle del tour.

imagen: imagen del tour.

duracion_horas: duración aproximada.

precio: costo por persona o tour.

cupo_maximo: límite de personas.

punto_encuentro: lugar de inicio.

servicios_incluidos: lista de servicios incluidos.

activo: si el tour está disponible.

created_at: fecha de creación.

Relaciones:

Tours(1) a booking_requests(N)

Si el request es tipo tour, varios request pueden apuntar a un mosimo tour.

Es decir, es de tipo 1 a muchos.

4. **Tabla reservations:** Es la tabla de reservas de hospedajes, no tours.

   Atributos: id, listing_id, check_in, check_out, huespedes, nombre, email, precio_total, estado.

   Detalle de los atributos:

   id: identificador único de la reserva.

   listing_id: hospedaje reservado.

   check_in: fecha entrada.

   check_out: fecha salida.

   huespedes: número de personas.

   nombre: nombre del huésped que reservó.

   email: correo del huésped.

   precio_total: total calculado.

   estado: estado de la reserva.

   Relaciones:

   Reservations (N) a listings (1)

   Muchas reservas pueden pertenecer a un hospedaje.

   Es decir, es de tipo muchos a 1

5. **Tabla booking_requests:** Se guardan las solicitudes de pago manual por transferencia bancaria para hospedajes o tours.

   Atributos: id, user_id, tipo, item_id, item_nombre, fecha_inicio, fecha_fin, personas, estado, monto_estimado, metodo_pago, notas, created_at.

   Detalle de los atributos:

   id: identificador único de la solicitud.

   user_id: usuario que la creó.

   tipo: si es listing o tour.

   item_id: id del listing o tour solicitado.

   item_nombre: nombre del hospedaje o tour (copia para referencia).

   fecha_inicio: fecha inicio (si aplica).

   fecha_fin: fecha fin (si aplica).

   personas: número de personas.

   estado: pendiente/aceptada/rechazada.

   monto_estimado: total calculado.

   metodo_pago: cómo pagará (transferencia).

   notas: referencia o comentario.

   created_at: fecha de creación.

   Relaciones:

   Booking_requests (N) a users (1)

   Muchas solicitudes pertenecees a 1 uuario.

   Es decir, es de tipo muchos a 1.

   Booking_requests (N) a listings/tours (1)

   Muchas solicitudes apuntan a un hospedaje o tour según el tipo.

   Es decir, es de tipo muchos a 1.

   No hay relaciones de muchos a muchos directas en esta base de datos.

   Resumen de claves foráneas

1. **booking_requests.user_id → users.id**

Cada solicitud pertenece a un usuario.

2. **reservations.listing_id → listings.id**
   Cada reserva pertenece a un hospedaje.
3. **booking_requests.item_id → listings.id**
   Cuando es de tipo listing la solicitud apunta al hospedaje solicitado.
4. **booking_requests.item_id → tours.id**
   Cuando es de tipo tour la solicitud apunta al tour solicitado.

La relación booking_requests.item_id es polimórfica, es decir puede apuntar a listings o tours, por eso no hay una FK estricta en la base para ese campo, la validación se hace por el campo tipo.

2.6 Requisitos Funcionales

Los requisitos funcionales es la funcionalidad principal del sistema, es decir las acciones que ofrece cumplir con los objetivos del proyecto.

El sistema debe permitir:

- Registro de los usuarios huéspedes en la plataforma.
- Inicio de sesión en el login mediante usuario y contraseña.
- Poder loguearse según el rol del usuario como administrador o huesped.
- Poder revisar los alojamientos y tours en la página principal sin necesidad de loguearse.
- Puede consultar el detalle completo de un hospedaje o tour.
- El huesped puede enviar solicitudes de reserva una vez logueado.
- Por defecto se registra las solicitudes de reserva en estado de pendiente.
- El administrador puede rechazar o aceptar las solicitudes.
- Las reservas aceptadas se registran como confirmadas.
- El huésped logueado puede visualizar sus solicitudes en la página historial.
- El administrador puede gestionar los hospedajes y tours mediante operaciones CRUD.
- El administrador puede visualizar las solicitudes de reserva en el panel de administración.
- Ambos roles pueden cerrar sesión de forma segura, mediante el botón cerrar sesión.

2.7 Requisitos No Funcionales

Los requisitos no funcionales son los que definen las características de rendimiento, calidad y seguridad del sistema.

El sistema cumple con los siguientes aspectos:

- Utiliza autenticación segura basada en JWT.
- Protege las rutas del sistema según sus roles o permisos.
- Garantiza la seguridad de los datos personales en la base de datos PostgreSQL.
- Permite el acceso simultaneo de varios usuarios.
- Tiene un diseño responsive adaptado a dispositivos móviles y computadoras.
- Mantiene respuestas rápidas en las consultas de hospedajes y tours.
- Valida los datos correctos de usuarios.
- La arquitectura de la programación separado en front-end, back-end y base de datos, ayuda en el mantenimiento y escalabilidad del proyecto.
- Usa contenedores Docker para el despliegue a producción de manera más optima.
- Con sus lenguajes de programación de vanguardia, permite la compatibilidad en navegadores web modernos.

CAPÍTULO III

Pruebas y Estabilización del Sistema

3.1 Reuniones con el Product Owner

Se realizo reuniones recurrentes con la Product Owner Dra. Luzmila Criollo, tanto de forma virtual como presencial.
El involucramiento del Product Owner fue un paso esencial, en las pruebas realizadas en el transcurso y término de cada Scrum, ya que nos ayudó resolviendo las dudas surgidas en el desarrollo del sistema.

Sobre todo, el Product Owner se involucró directamente en revisar si cada funcionalidad desarrollada, le resolvía las necesidades reales del negocio.

Dado que el sistema fue desarrollado por un unico programador la comunicación fue fácil y efectiva con el Product Owner y Scrum Máster.

La mayoría de las reuniones se realizaron de forma virtual donde se revisaron aspectos como el flujo de reservas, validación de fechas disponibles, gestión de usuarios, el funcionamiento correcto del panel administrativo, y la interfaz de usuario en general.

Las observaciones y sugerencias realizadas fueron aplicadas de forma óptima y progresiva mediante el transcurso del desarrollo del sistema.

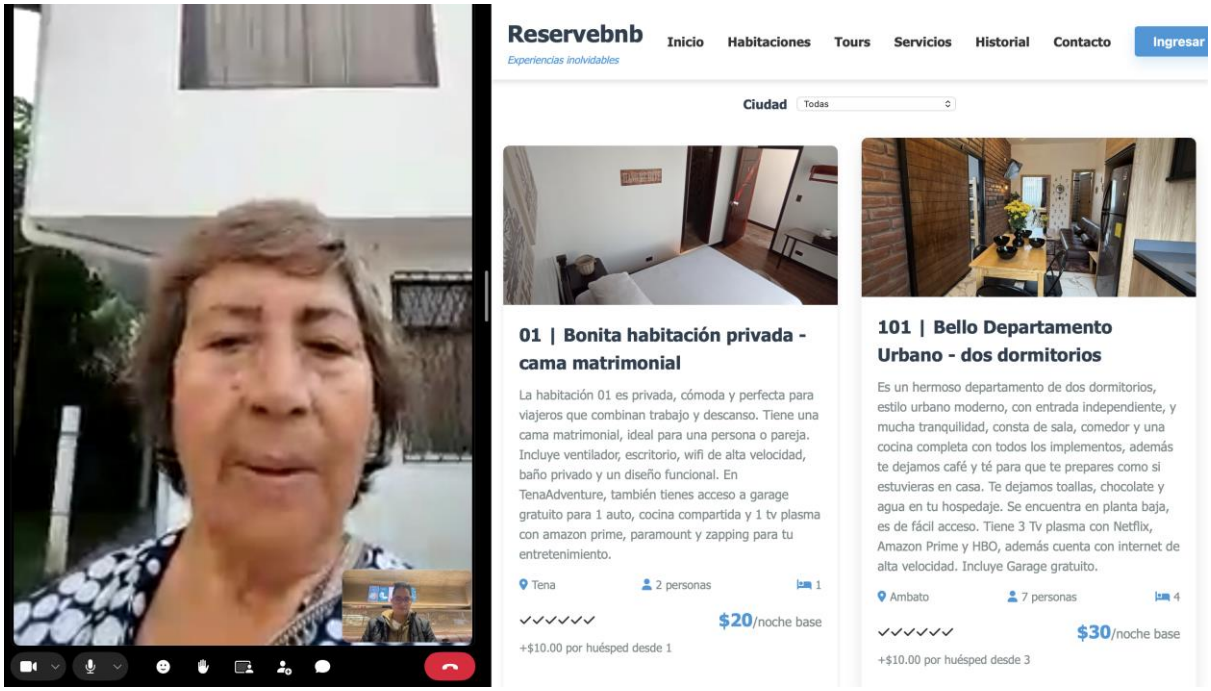


Ilustración 7. Revisión del proyecto con la Product Owner.

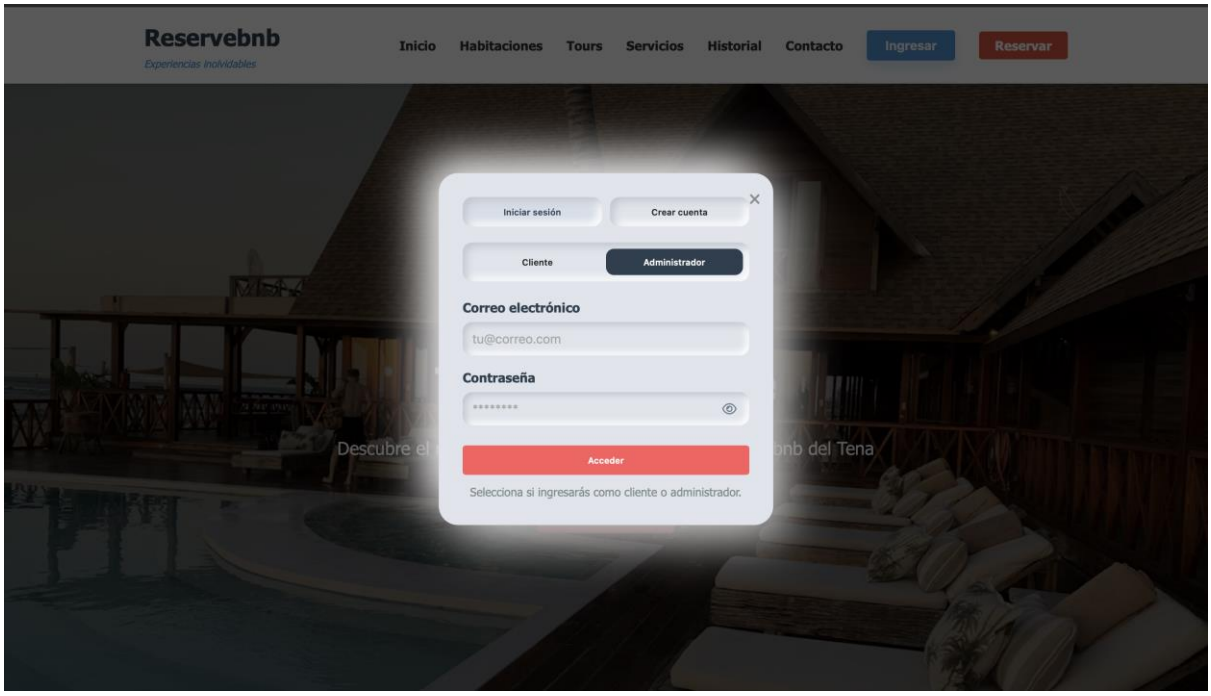
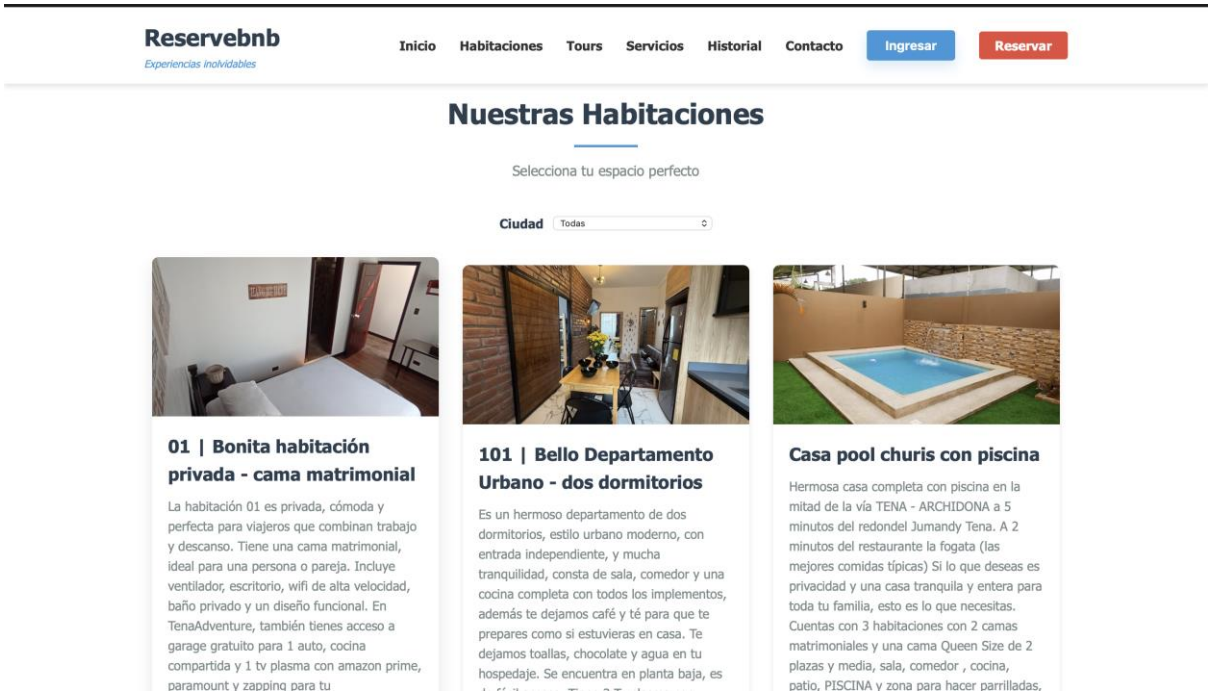


Ilustración 8. Inicio de sesión del sistema Reservebnb.



Ilustracion 9. Creación correcta de Hospedajes.

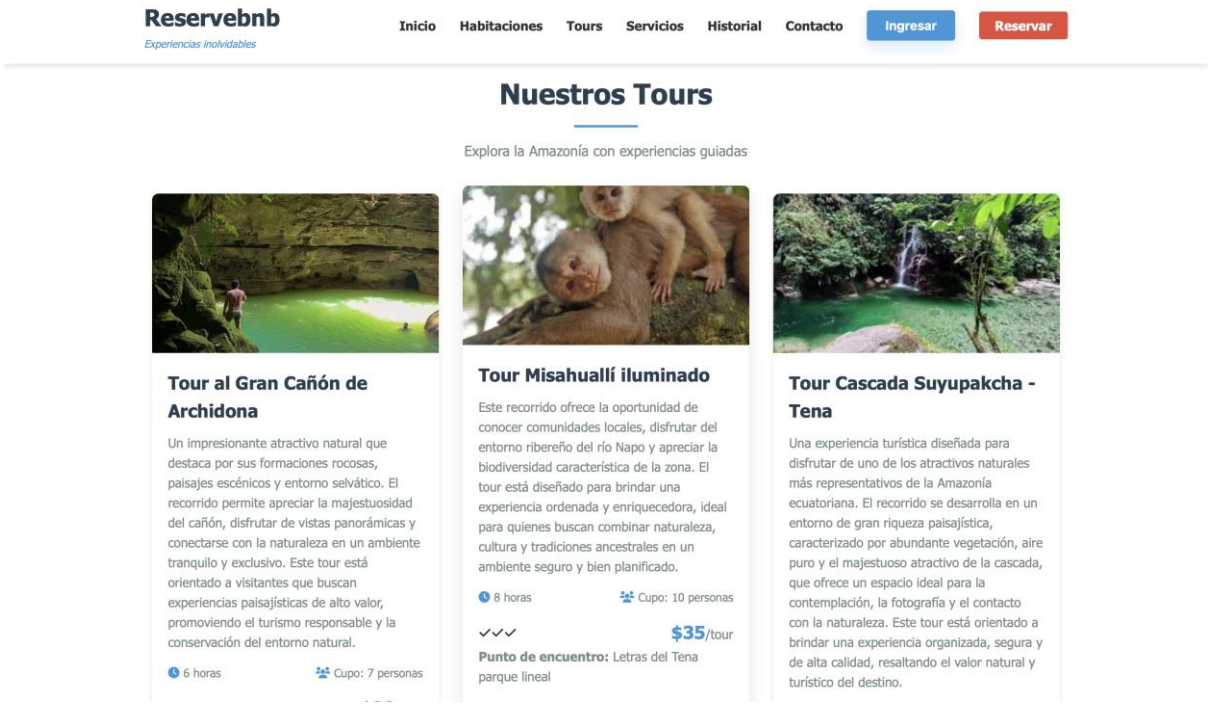


Ilustración 10. Creación correcta de Tours.

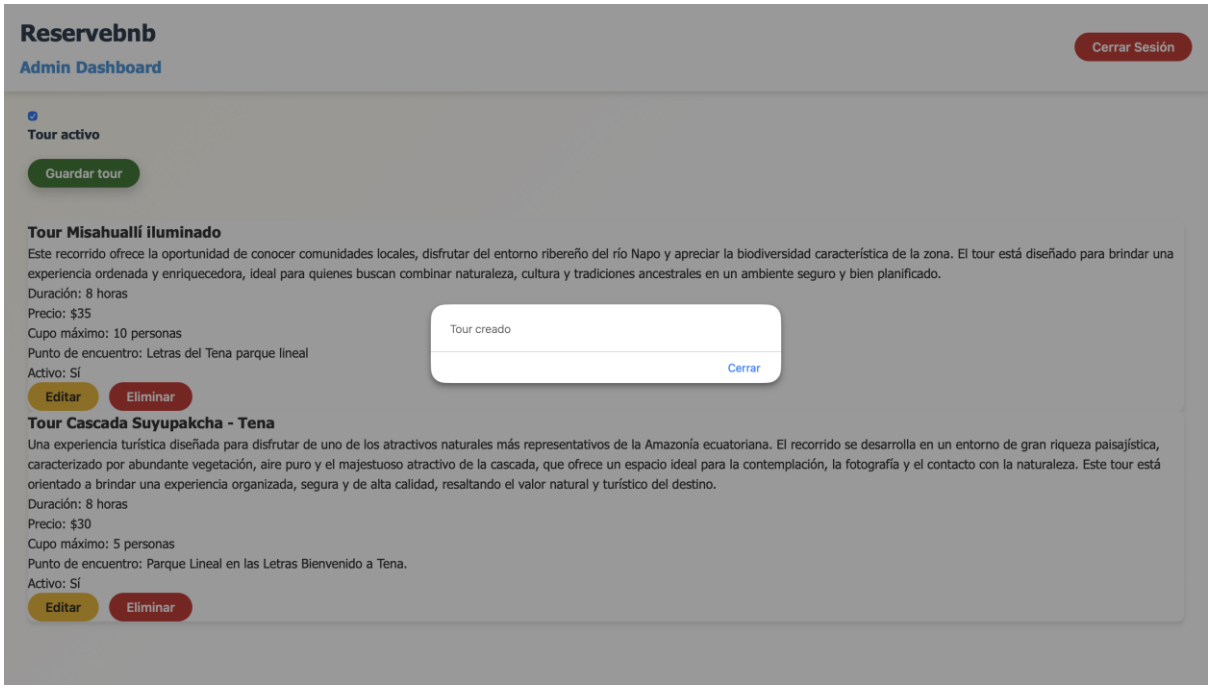


Ilustración 11. Funcionamiento correcto del panel administrativo.

3.2 Tipos de pruebas realizadas

3.2.1 Pruebas funcionales

Se probo cada funcionalidad para verificar que los requisitos establecidos fueron cumplidos, y un Sprint se consideraba finalizado solo si cumplía con la Definition of Done.

Se probo lo siguiente:

- Inicio y registro de sesión con usuario administrador y huésped.
- Visualización, creación, edición y eliminación de hospedajes y tours con usuario administrador.
- Visualización y envío de solicitudes de reserva realizadas por el huésped.
- Manejo de solicitudes de reserva en el panel administrativo por parte del administrador.

3.2.2 Pruebas de integración

Este tipo de prueba comprobó el correcto funcionamiento de comunicación entre los diferentes módulos del sistema que son el frontend, backend y la base de datos.

Con Postman y el aplicativo web se comprobó que el frontend consumía correctamente la API y que los datos se leían y guardaban en la base de datos.

Las pruebas realizadas fueron:

- El frontend y el backend se comunicaron correctamente mediante API REST.
- La integración del backend con la base de datos PostgreSQL fue efectiva.
- Endpoints consumidos correctamente desde el frontend.
- Se guardo la información en la base de datos cuando se realizaba operaciones CRUD.

3.2.3 Pruebas de interfaz

Esta prueba analizó la usabilidad y experiencia de usuario, comprobando que el aplicativo web sea intuitivo y fácil de usar.

Se evaluó lo siguiente:

- La navegación por la página fue clara e intuitiva.
- Visualización ordenada de hospedajes y tours, en la pantalla principal.
- Diseño responsive para dispositivos móviles y web.
- Facilidad en el registro de usuarios, inicio de sesión y solicitudes de reserva.

3.2.4 Pruebas de rendimiento

Se medieron los tiempos de respuesta en consultas de hospedajes y tours visualmente y con Postman se midió los tiempos de respuesta de los endpoints.

Las pruebas ejecutadas fueron:

- Medición de los tiempos de respuesta en consultas de hospedajes y tours.

- Observación en el rendimiento del backend con consultas frecuentes.

3.2.5 Pruebas de seguridad

Se verifico que mediante la lógica del sistema y Postman que exista validación de JWT, rutas protegidas, restricción de acceso por roles y cifrado de contraseñas con bcrypt.

3.3 Herramientas utilizadas

Postman: Se realizo pruebas de los endpoints de la API REST y valido las respuestas en formato JSON.

Navegador web: Se valido visualmente las funcionalidades del sistema.

Consola y logs del servidor: Se utilizo para depuración del código y seguimiento de errores.

3.4 Planilla de pruebas con Sprint

Tabla 2

Pruebas en base a los Sprint

Sprint	Tarea	Descripción	Tipo de prueba	Responsable	Resultado esperado	Estado
S1	Configuración inicial del backend	Creación del proyecto con Node.js y Express	Revisión técnica	gerente90	El Backend se ejecutó en el entorno local	ok
S1	Configuración de la base de datos	Creación de la base de datos con PostgreSQL	Prueba de integración	gerente90	La Conexión fue exitosa entre el backend y la base de datos	ok

S1	Configuración del Docker	Configuración inicial del Docker y Docker Compose	Prueba de entorno	gerente90	Los servicios se ejecutaron sin errores	ok
S2	Estructura base del frontend	Creación de estructura HTML, CSS y JavaScript	Prueba funcional	gerente90	La interfaz se carga correctamente en el navegador	ok
S2	Integración frontend–backend	Consumir la API REST desde el frontend	Prueba de integración	gerente90	La comunicación fue correcta entre el frontend y backend	ok
S3	Autenticación de usuarios	Implementación de roles y login con JWT	Prueba de seguridad	gerente90	Acceso restringido según el rol	ok
S3	Registro de usuarios	Registro de huéspedes en la plataforma	Prueba funcional	gerente90	Los usuarios fueron registrados exitosamente	ok
S4	Gestión de hospedajes	CRUD de hospedajes desde el panel administrativo	Prueba funcional	gerente90	Hospedajes visualizados, creados, editados y eliminados.	ok
S4	Gestión de tours	CRUD de tours	Prueba funcional	gerente90	Los Tours hacen CRUD exitosamente	ok
S4	Solicitud de reservas	Envío de solicitudes del huésped	Prueba funcional	gerente90	Solicitudes registradas en estado inicial de pendiente	ok
S4	Gestión de solicitudes	Aprobar o rechazas reservas	Prueba funcional	gerente90	Las reservas se actualizan correctamente	ok
S5	Historial de solicitudes	Visualización del historial	Prueba funcional	gerente90	El historial se muestra	ok

		por parte del huésped			exitosamente	
S5	Notificaciones por correo	Envío de correos por estados de reserva	Prueba de integración	gerente90	Correos enviados y validados	ok
S6	Pruebas de rendimiento	Se verifico los tiempos de respuesta	Prueba de rendimiento	gerente90	Tiempos de respuesta adecuados	ok
S6	Pruebas de seguridad	Validación de rutas protegidas y roles	Prueba de seguridad	gerente90	Accesos autorizados y restringidos validados	ok
S7	Documentación y despliegue	Documentación final y entrega para producción	Revisión final	gerente90	Codigo entregado y capacitado	ok

3.4 Estabilización del sistema

Antes de la entrega se realizaron las siguientes actividades:

- Ajustes de código y mejora de diseño en formularios de registro, inicio de sesión y solicitud de reservas.
- Optimización de consultas a la base de datos PostgreSQL.
- Corrección de rutas en el frontend al cerrar sesión.
- Mejora en el manejo de respuestas del backend ante datos no encontrados.
- Creación de usuarios de administrador para el manejo de panel administrativo.
- Se implemento una lógica de código en el calendario de reserva del aplicativo sincronizado con el calendario de Airbnb del Product Owner.
- Se mejoro el diseño en colores y forma del panel administrativo.

Conclusiones

Cumplimiento del objetivo del proyecto:

El desarrollo del sistema web Reservebnb para la empresa Tena Adventure cumplió con el objetivo de crear una plataforma web para la gestión y control de reservas de hospedajes y tours, ofreciendo una solución funcional que automatizó procesos y redujo la dependencia de plataformas externas.

Optimización del sistema de reservas

Con el sistema desarrollado el gerente puede administrar y gestionar los hospedajes y tours desde un panel administrativo propio, mejorando el control y evitando depender de plataformas de terceros, que generaban desorganización y dependencia tecnológica de plataformas externas.

Mejora en la experiencia del usuario

Durante las pruebas el gerente evidencio una navegación sencilla e intuitiva lo que le permite un fácil manejo de la plataforma tanto para huéspedes como administradores, obteniendo información detallada y visualización de los estados de las reservas.

Escalabilidad y proyección futura

La arquitectura modular basada en API REST y el uso de tecnologías modernas con Node.js, PostgreSQL y Docker, permite al sistema continuar con futuras ampliaciones por ejemplo incorporación de pagos en línea, reportes avanzados, integración completa con calendarios externos o una aplicación móvil.

Recomendaciones

- Implementar una pasarela de pagos en línea dentro de la aplicación web.
- Desarrollar una versión móvil para el proyecto.
- Mantener actualizado las dependencias para el correcto funcionamiento del sistema tanto en frontend como backend.
- Revisar y optimizar la base de datos conforme aumente el volumen de consultas.
- Optimizar el diseño del backend con más funcionalidades como reportes o estadísticas para mejorar la toma de decisiones.
- Mantener el repositorio de GitHub actualizado para facilitar el mantenimiento.
- Generar una visualización a través del panel de control de los usuarios huéspedes registrados con información de contacto para tener una base de Marketing.

Referencias bibliográficas

Node.js. (s.f.). Introduction to Node.js. Recuperado el 28 de mayo de 2025 de https://nodejs.org/en/learn/getting-started/introduction-to-nodejs

Express. (s.f.). Express – Node.js web application framework. Recuperado el 28 de mayo de 2025 de https://expressjs.com/

PostgreSQL. (s.f.). What is PostgreSQL? Recuperado el 28 de mayo de 2025 de https://www.postgresql.org/docs/current/intro-whatis.html

Docker. (s.f.). Docker documentation. Recuperado el 28 de mayo de 2025 de https://docs.docker.com/

Mozilla Developer Network. (s.f.). JSON Web Tokens (JWT). Recuperado el 28 de mayo de 2025 de https://developer.mozilla.org/es/docs/Glossary/JWT

Postman. (s.f.). API testing documentation. Recuperado el 28 de mayo de 2025 de https://learning.postman.com/docs/introduction/overview/

Ministerio de Turismo del Ecuador. (2024). Informe anual de turismo en Ecuador. Recuperado de https://www.turismo.gob.ec

Brown, E. (2019). Web development with Node and Express: Leveraging the JavaScript stack. O'Reilly Media.

Banks, A. (2020). Node.js web development: Server-side web development made easy with Node 14 using practical examples (5th ed.). Packt Publishing.

Hollemans, M. (2023). PostgreSQL for Node.js developers: A practical guide. Independently published.

Wilson, D. (2022). Docker for developers: Develop and run your application with Docker containers using DevOps tools for continuous delivery. Packt Publishing.