ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA WEB TURÍSTICO PARA LA PARROQUIA DE NONO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

SANDY CAROLINA DONOSO CAÑAR

DIRECTOR: DR. JULIO CÉSAR CAIZA ÑACATO

Quito, enero 2023

AVAL

Certifico que el presente trabajo fue desarrollado por Sandy Carolina Donoso Cañar, bajo mi supervisión.

Dr. JULIO CÉSAR CAIZA ÑACATO

DECLARACIÓN DE AUTORÍA

Yo, Sandy Carolina Donoso Cañar, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

SANDY CAROLINA DONOSO CAÑAR

DEDICATORIA

A mi padre.

AGRADECIMIENTO

A la Virgen del Camino que siempre me guía y cuando me siento perdida me devuelve al camino.

Quiero agradecer a mi familia a mi papi Edison, a mi mami Gladys y a mi hermano Esteban quienes siempre han estado a mi lado cuidándome y mostrándome su apoyo incondicional.

A mis tías, que siempre me han mostrado su cariño, han creído en mí y me han defendido a pesar de todo.

A todos los amigos de la universidad, en especial a Bibi, Daya, Kchy, Paulcillo, Jimmy, Jorge, Javi y Juanito con quienes compartí diversos momentos buenos y malos. Gracias por todo.

A mis amigas del colegio que siempre las llevo en mi corazón, no importa cuantos años pasen, con ustedes se siente que no ha trascurrido el tiempo.

A Sammy, Oscar y Gina que me ayudaron en la elaboración de este proyecto ya sea compartiéndome sus conocimientos, con sus consejos o con el escrito. Sin su ayuda no hubiera podido terminar.

Finalmente, agradezco a mi director, el doctor Julio Caiza, por su guía, por su tiempo y sobre todo por su paciencia. En este viaje he aprendido mucho gracias a usted tanto la parte técnica como sobre mí misma, gracias.

Non, je ne regrette rien

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo describe el desarrollo de un sistema web turístico para la parroquia de Nono en Ecuador, que permite a los turistas ver información de las actividades ofertadas por diferentes establecimientos y reservarlas. Estas actividades, sus establecimientos ofertantes y sus reservas, son gestionadas por los encargados de los establecimientos.

El desarrollo de este sistema ha seguido la metodología Kanban y se ha hecho uso del stack MERN (*MongoDB*, *Express*, *React* y *NodeJS)*. Este proceso se describe en cinco capítulos:

El primer capítulo presenta el fundamento teórico necesario para llevar a cabo este proyecto. Esto incluye la metodología Kanban, el stack MERN, y la arquitectura Modelo-Vista-Controlador.

El segundo capítulo presenta la metodología utilizada para la realización del sistema; particularmente aborda el análisis y diseño. Se presentan los requerimientos funcionales y no funcionales, diagramas de casos de uso, de secuencia, de clases y los bosquejos para la interfaz gráfica. También presenta la implementación del sistema web, esto incluye herramientas de desarrollo, servidor, base de datos y un ejemplo de implementación para la gestión de categorías.

El tercer capítulo contiene los resultados de las pruebas de funcionalidad del sistema web realizadas por diferentes usuarios, el resumen de encuestas realizadas a dichos usuarios y los errores encontrados con sus respectivas correcciones.

Por último, en el cuarto capítulo se exponen las conclusiones y recomendaciones obtenidas en el desarrollo de este trabajo de titulación.

PALABRAS CLAVE: Reservas, sistema web, Stack MERN, metodología Kanban,

ABSTRACT

This work describes the development of a tourism web system for the town of Nono in Ecuador. It allows tourists to see information on the activities offered by different establishments and book them. The managers of the establishments manage activities, their offering establishments and reservations.

The development of this system has followed the Kanban methodology and the MERN stack (MongoDB, Express, React and NodeJS). This process is described in five chapters:

The first chapter presents the theoretical foundation necessary to carry out this project. This includes the Kanban methodology, the MERN stack, and the Model-View-Controller architecture.

The second chapter presents the methodology used to carry out the system; particularly addresses analysis and design. Functional and non-functional requirements, use case diagrams, sequence diagrams, class diagrams, and sketches for the graphical interface are presented. It also presents the implementation of the web system, this includes development tools, server, database and an implementation example for category management.

The third chapter contains the results of the tests of the web system carried out by different users, the summary of surveys carried out and the errors found with their respective corrections.

Finally, in the fourth chapter the conclusions and recommendations obtained in the development of this work are presented.

KEYWORDS: Reservations, web system, MERN stack, Kanban methodology

1. INTRODUCCIÓN

San Miguel de Nono es una parroquia ubicada en la Provincia de Pichincha, que debido a su naturaleza y a la cercanía de la ciudad tiene gran potencial turístico. Considerando esta situación varios lugareños han optado por dedicarse a actividades turísticas como lo son: servicio de alimentación, pesca deportiva, alojamiento, paseos en reservas naturales, entre otros, que han ido creciendo con el paso de los años.

No obstante, este crecimiento se detuvo debido a la pandemia producida por el virus COVID-19, y por las medidas sanitarias que se habían adoptado. Una de esas medidas fue la disminución de aforo; por lo que era necesario realizar un control del mismo.

Es importante agregar, el hecho de que varios de los negocios no tienen visibilidad a través de Internet y por ende los turistas se enteran de las actividades y productos ofertados, incluso del aforo permitido en cada local, cuando ya se encuentran en el establecimiento.

Por lo mencionado anteriormente, el presente trabajo tiene como finalidad el desarrollo de un sistema web que podrá ser usado por los diferentes establecimientos de la parroquia de Nono que ofrezcan actividades turísticas. Se les brindará ventajas como: administración, ya que podrán controlar las actividades que deseen mantener publicadas, así el turista interesado va a tener diferentes opciones al momento de visitar la parroquia; dinamismo de contenido, ya que ellos podrán definir la información de la actividad que quieren mostrar; control de bioseguridad a través de la gestión de reservas y de aforo.

1.1. OBJETIVOS

El objetivo general de este Trabajo de Titulación es desarrollar un sistema web turístico para la parroquia de Nono.

Los objetivos específicos de este Trabajo de Titulación son:

- Analizar los conceptos fundamentales para el desarrollo del proyecto propuesto.
- Diseñar los módulos del sistema especificados en el alcance.
- Implementar los módulos del sistema según el diseño realizado.
- Analizar los resultados obtenidos en las pruebas realizadas.

1.2. ALCANCE

Este trabajo de titulación propone el desarrollo de un sistema web que permita tener una mayor visibilidad de las actividades turísticas que se llevan a cabo en la parroquia de Nono. El sistema va a permitir a los turistas generar reservas. Conjuntamente, el sistema va a permitir la administración de los establecimientos y las actividades propuestas.

Para llevar a cabo el sistema, se necesitará analizar el contexto, determinar los requerimientos, diseñar la solución, implementar, y finalmente realizar pruebas con los usuarios.

Se tendrán tres tipos de roles:

- **Administrador:** se encarga de la administración general del sistema web. Va a poder gestionar las categorías (o tipos) de actividades. Va a poder realizar la gestión de los encargados del establecimiento. Además, podrá iniciar sesión. Un administrador principal va a estar creado desde el inicio y es quien tendrá la facultad de crear y gestionar nuevos administradores.
- **Encargado de establecimientos:** se encarga de la administración de los establecimientos en el sistema. El encargado puede gestionar más de un establecimiento y las actividades de cada uno. El encargado va a poder registrarse en el sistema e iniciar sesión.

  Además, este rol va a poder administrar reservas pendientes y confirmadas. Las reservas pendientes son aquellas que aún no tiene una confirmación. Si no se acepta el horario de una reserva, posteriormente quedará libre para otro turista interesado. Las reservas confirmadas son aquellas aceptadas, tanto por parte del cliente como por parte del encargado; este horario ya no se encontrará disponible en el sistema web.
- **Turista:** Este rol no inicia sesión en el sistema. Va a poder ingresar en la plataforma y podrá visualizar las actividades. Si el cliente desea reservar, se gestiona su petición dependiendo de los diferentes escenarios propuestos en la generación de reservas detalladas en la Figura 1.1. Independientemente del escenario, el turista interesado va a llenar un formulario con al menos nombres, apellidos, números telefónicos, cantidad de reservas o cantidad de platos.

El sistema contará con los siguientes módulos:

Módulo del administrador:

- **Inicio de sesión en el sistema:** el administrador podrá iniciar sesión mediante el ingreso del correo y su contraseña.
- **Gestionar categorías de actividades:** las categorías van a permitir la clasificación de las actividades para poder organizarlas. El administrador va a poder realizar las siguientes acciones: ingresar, modificar, borrar y visualizar categorías. Cada categoría va a contar al menos con un nombre y una descripción.
- **Gestionar encargados de establecimiento:** el administrador va a poder realizar las siguientes acciones: ingresar (o aceptar), modificar, borrar y visualizar a los encargados de los establecimientos.
- **Gestionar administradores:** esta acción solo podrá ser realizada por el administrador principal; este va a poder realizar las siguientes acciones: ingresar, modificar, borrar y visualizar a los usuarios administradores.

Módulo de turista:

- **Visualizar actividades:** el turista interesado va a poder visualizar las categorías con sus respectivas actividades. En cada una podrá visualizar información básica de las actividades y los horarios disponibles para reservar.
- **Generar reservas:** para poder generar una reserva, el cliente va a tener que seleccionar una actividad y un horario disponible. Se va a presentar un formulario donde se va a solicitar datos de la reserva. Una vez ingresados todos los datos, se va a generar una reserva pendiente. Esta reserva pendiente cambiará a reserva confirmada dependiendo de dos factores: la duración de la actividad y la cantidad de reservas; esta información debió ser ingresada en el formulario previo. El sistema validará estas opciones como se muestra en la Figura *1.1* El sistema modificará el estado de reserva o se esperará confirmación por parte del encargado del establecimiento.

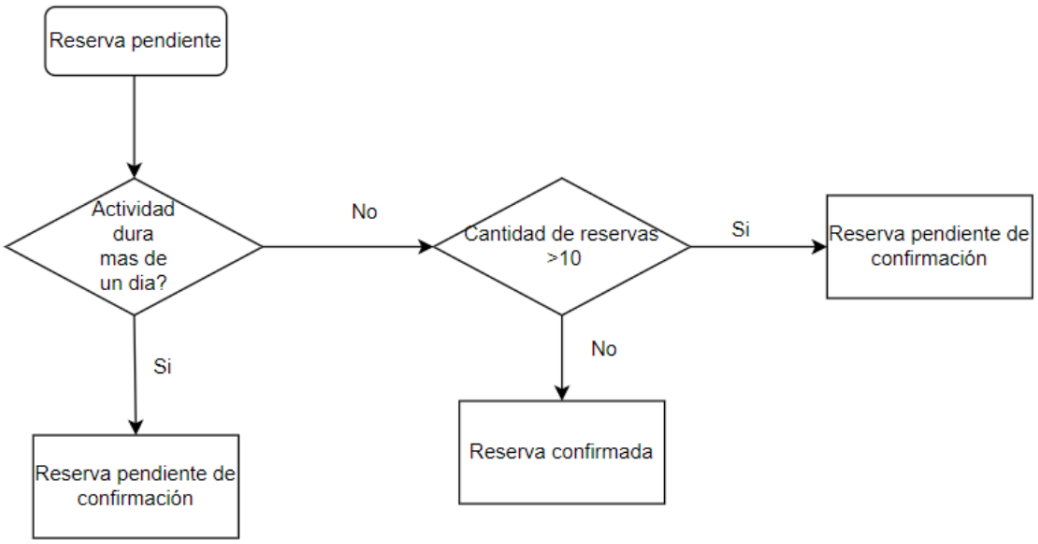

Figura 1.1. Diagrama de flujo de la gestión de reserva pendiente.

Módulo del encargado de establecimiento:

- **Registro:** los encargados de los establecimientos se registrarán con sus datos, entre ellos se incluirán un correo y una contraseña para autenticarse en la aplicación.
- **Inicio de sesión en el sistema:** el encargado del establecimiento podrá iniciar sesión mediante el ingreso del correo y su contraseña.
- **Gestionar establecimientos:** en este módulo el encargado de los establecimientos va a poder ingresar, modificar, borrar y visualizar establecimientos. Cada establecimiento al menos contará con su nombre, descripción, ubicación y redes sociales.
- **Gestionar actividades:** en este módulo el encargado de los establecimientos va a poder ingresar, modificar, borrar y visualizar actividades por establecimiento. Cada actividad contará con un título, una descripción, la oferta de horarios para reserva y fotografías para la presentación en la plataforma.
- **Gestionar reservas pendientes:** en este módulo el encargado de los establecimientos va a poder agregar, convertir reservas pendientes en confirmadas, modificar, borrar y visualizar reservas pendientes. Cuando se genere una reserva, su estado inicial va a ser pendiente, si la reserva cumple con los requisitos mostrados en la Figura 1.1 se creará un mensaje con toda la información del turista interesado. Se dará la opción de enviar mensaje por WhatsApp; la cual al seleccionarse va a abrir la aplicación para que se envíe la información al teléfono de

contacto del establecimiento. Una vez se haya dialogado y llegado a un acuerdo de reserva, el encargado del establecimiento confirmará la reserva mediante el sistema web. El estado de reserva pendiente permanecerá así durante un tiempo de 2 días máximo; después de eso se va a liberar ese horario para otras personas interesadas.

- **Gestionar reservas confirmadas:** La reserva confirmada va a cumplir con los requisitos establecidos en la Figura 1.1. El horario de la reserva ya no estará disponible en el sistema web. Adicionalmente, se va a poder borrar y visualizar reservas confirmadas. Además, se van a poder agregar reservas confirmadas a través de otros medios que tenga el encargado (por ejemplo, a través de llamadas telefónicas).

El sistema tendrá una arquitectura cliente-servidor, en la cual el servidor gestionará peticiones del cliente y enviará respuestas que puedan ser presentadas al usuario final del sistema. Además, se encargará de gestionar toda la información en una base de datos.

Por otra parte, el lado del cliente interactuará de forma directa con el usuario mediante una interfaz gráfica (GUI) que, dependiendo de su rol, le permitirá acceder a los diferentes módulos. Para el proceso de codificación del servicio web, se usará el stack MERN, cuyas siglas provienen de *MongoDB*, *Express*, *React* y *NodeJS* [1]. El proyecto aplicará esta herramienta para el desarrollo completo del frontend y *backend*. La fluidez que provee este stack se debe al uso único de *Javascript* como lenguaje de programación.

La arquitectura de la aplicación se muestra en la Figura *1.1*Figura *1.2*.

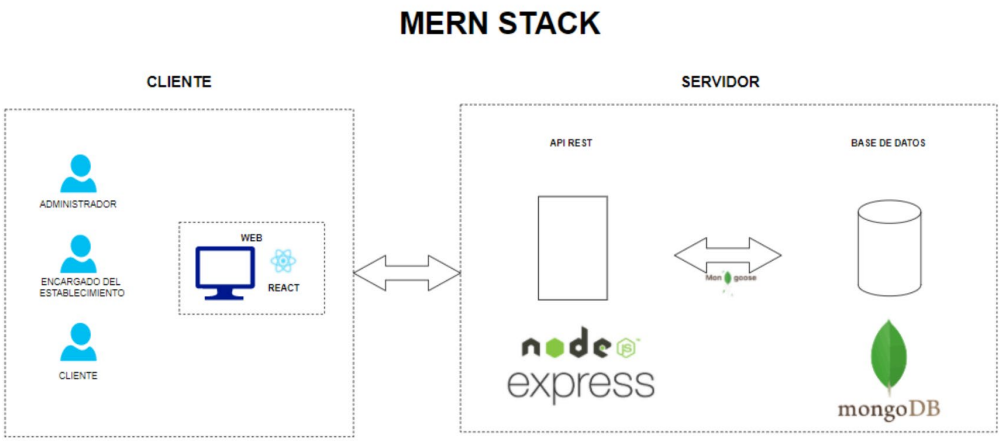

Figura 1.2. Arquitectura del sistema.

Para seguir un proceso de desarrollo ordenado, definido en tiempos y con actividades específicas, se utilizará como base la metodología de desarrollo ágil Kanban.

Las funcionalidades del sistema se validarán a través de pruebas de funcionamiento. El presente trabajo de titulación tendrá un producto final demostrable.

1.3. MARCO TEÓRICO

1.3.1. METODOLOGÍA KANBAN

Kanban es una metodología ágil que permite la administración de flujo de tareas, cuyo origen se dio en Toyota y sus procesos de producción "JUST-IN-TIME" (JIT) [2]. La palabra Kanban tiene origen japonés y significa tarjeta con signos o señal visual.

Funcionamiento de Kanban

El método Kanban divide el trabajo en tareas pequeñas, con el objetivo de permitir la visualización y el control del flujo de trabajo que tiene el equipo. En particular:

- Permite tener una visión general y consolidada de las tareas y el estado en que se encuentran.
- Facilita la detección de problemas que se generen durante la toma de decisiones o en la implementación de alguna de las tareas.
- Permite visualizar los cuellos de botella en tiempo real.
- Posibilita una correcta asignación de tareas al equipo de trabajo debido a la visión amplia que provee.

Los equipos de desarrollo de software utilizan esta metodología ágil para aprovechar los principios de JIT, ajustando la cantidad de trabajo en curso a la capacidad del equipo [3].

Tablero Kanban

El método Kanban hace uso de una herramienta esencial conocida como tablero Kanban. Esta herramienta proporciona un sistema visual en el cual se muestra las tareas del proyecto [3], en forma de tarjetas (Figura *1.3*), organizadas a través de columnas, que podrían representar el estado de una tarea (p.ej. "Por hacer, "En proceso" y "Hecho) (Tabla *1.1*). De este modo, el tablero permite que el equipo se concentre únicamente en las tareas que se requieran, de una manera fluida y constante. Hay que tener en cuenta que el tablero debe actualizarse constantemente en cada etapa del proyecto.

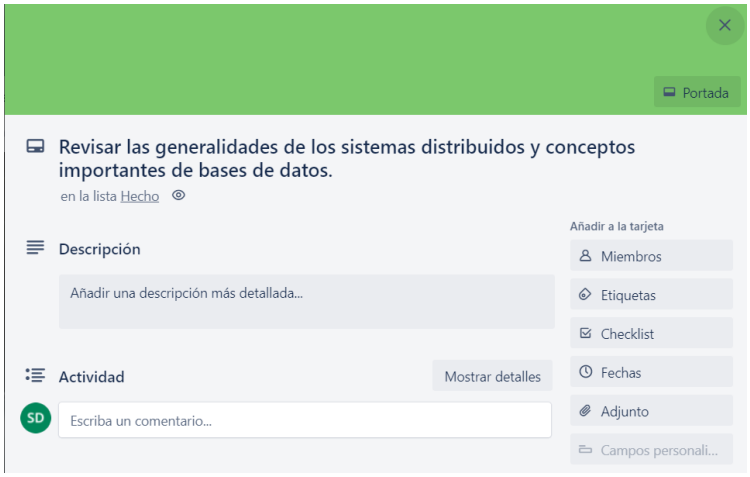

Figura 1.3. Tarjetas de Kanban.

Tabla 1.1. Columnas Kanban.

Por hacer	**En proceso**	**Hecho**
Tareas pendientes	Tareas que se están realizando en el momento	Tareas finalizadas

1.3.2. PROGRAMACIÓN POR CAPAS

Para crear un sistema de software existen múltiples lenguajes y prácticas de programación que pueden aplicarse para que el desarrollo sea organizado, continuo y admita posteriores mejoras. Un ejemplo de ello es la programación por capas, que consiste en separar las partes que conforman el programa y que cada una de ellas cumpla un rol simple, de tal manera que sus funciones o métodos sean invisibles al resto [4].

1.3.3. HERRAMIENTAS DE DESARROLLO

Para llevar a cabo el proyecto se va a aplicar el Stack MERN que define las tecnologías de desarrollo para implementar una arquitectura de un sistema de software de tres capas: servidor web (frontend), el servidor de aplicaciones y la base de datos. Las siglas del Stack MERN provienen de las tecnologías *MongoDB*, *Express*, *React* y *NodeJS*, ilustradas en la Figura *1.1*.

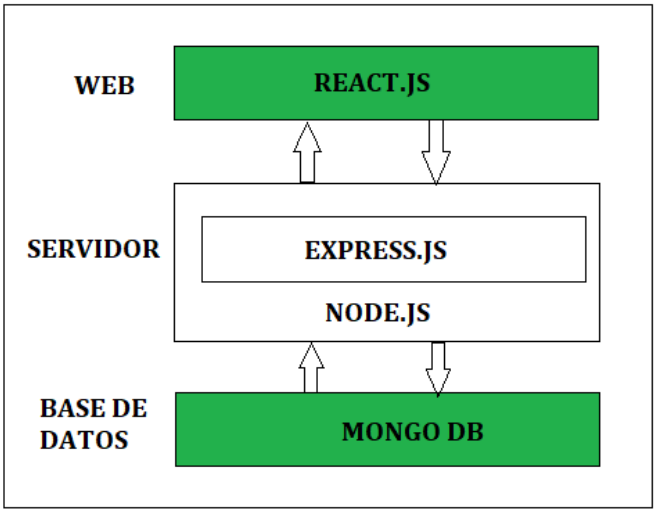

Figura 1.4. Arquitectura del Stack Mern.

Entre las ventajas que proporciona el Stack Mern [1] se puede mencionar:

- Proporciona herramientas para todo el ciclo de desarrollo del proyecto.
- Se basa en tecnologías considerablemente respaldadas.
- Trabaja solo con el lenguaje de programación *Javascript*.

1.3.4. BASE DE DATOS

MongoDB

MongoDB[1] es una base de datos no relacional, orientada a documentos, que proporciona alto rendimiento, disponibilidad y fácil escalabilidad [5]. Cada registro de esta base de datos es un documento que consta de secciones campo-valor. El conjunto de varios documentos se conoce como colecciones.

Por ejemplo, un documento *película* puede tener pares campo-valor como los mostrados en la Figura 1.5. Cabe indicar que toda información es guardada en formato BSON (*Binary Javascript Object Notation*), que es similar a los objetos JSON (*Javascript Object Notation*).

[1] Proviene del término *humongous* que significa enorme.

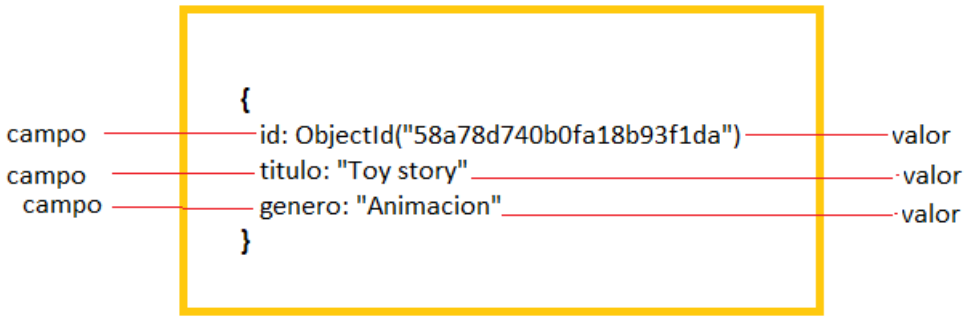

Figura 1.5. Documento de ejemplo en MongoDB.

Para representar relaciones entre elementos es posible realizar incrustaciones o usar referencias [5]. Una incrustación consiste en guardar información relacionada en el mismo documento del modo mostrado en la Figura *1.6*; donde se puede ver un ejemplo de dos incrustaciones. De esa manera, cuando se realizan operaciones CRUD (*create, read, update, and delete*), se emiten menos consultas y actualizaciones.

Figura 1.6. Ejemplo de incrustación de datos.

Para definir relaciones a través de referencias, se hace uso de un identificador de un documento apuntado desde otro documento. Es usada cuando se desea evitar duplicidad de datos [5].En la Figura *1.7*. se muestra un ejemplo.

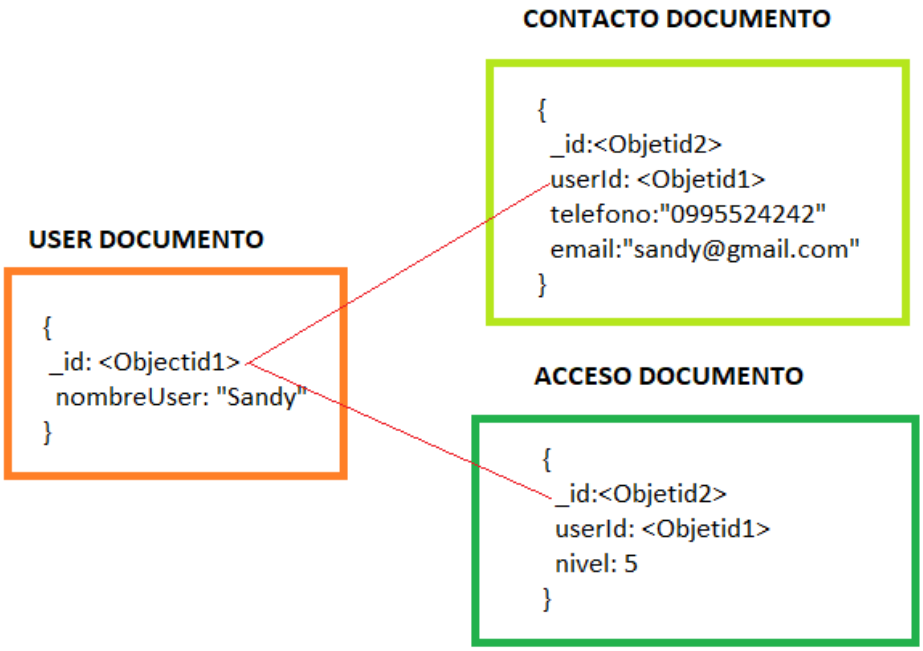

Figura 1.7. Ejemplo de uso de referencias.

Al ser *MongoDB* una base de datos NoSQL, los documentos son almacenados sin un esquema predefinido. Para establecer un esquema puede usarse *Mongoose*, que es una librería de modelado de datos de objetos ODM (*Object Document Mapper*) para *MongoDB* y *Node.js*. Su función es administrar las relaciones entre los datos, definir modelos, proporcionar validación de esquema, creación de consultas, etc [6].

1.3.5. SERVIDOR

El servidor está encargado de proveer servicios, locales o en red, a la capa Web (que se ejecuta en el lado del usuario) e interactuará con la capa de Base de Datos explicada previamente. El servidor recibe solicitudes HTTP de la capa Web y proporciona una respuesta HTTP, como una página HTML o JSON [7].

Node.js y Express

Node.js es un entorno de ejecución de código abierto y multiplataforma, que está construido sobre el motor de *Javascript V8* de Google Chrome [8], que se ejecuta en el Servidor y que contiene aplicaciones JS (de *backend*) que ofrecen los servicios a ser consumidos por la capa Web.

Para crear las aplicaciones JS, se puede hacer uso de un framework como *Express*, que facilita el manejo de solicitudes y respuestas HTTP, a través de un modelo de enrutamiento

de URL, donde se hace coincidir una URL entrante con un servicio específico [1]. Además, permite partir de un código ya preparado que se encarga de diversos subprocesos como la gestión de cookies, sesiones y usuarios.

Otras ventajas importantes de *Express* incluyen el ser asíncrono, eficaz, rápido, escalable y que promueve la reutilización del código.

1.3.6. WEB

Es la capa encargada de presentar el sistema web mediante una interfaz gráfica al usuario final. Esta capa envía peticiones del usuario hacia el servidor y este dirige las respuestas a la capa web que va a mostrar de manera visual el resultado al usuario. Para poder crear la interfaz gráfica se va a utilizar la librería *React* que permite la construcción de interfaces complejas a través de componentes simples, conectarlos a datos en el servidor *backend* y representarlos como HTML.

React

Es una librería creada por Facebook y utilizada para desarrollar páginas y aplicaciones con interfaces de usuario interactivas y sencillas [9].

Su estructura está basada en objetos *Javascript* que permiten dividir los elementos de la interfaz gráfica en piezas reutilizables e independientes, de tal forma que cada componente gestiona su propio estado cuando los datos cambian. No obstante, además de *Javascript*, como lenguaje de programación se utiliza un pseudo-lenguaje llamado JSX, que es una mezcla de JS y HTML utilizada para que el código coexista en un solo componente [9].

La versión 16.8 de *React* tiene una nueva característica llamada *Hooks*, que es el conjunto de funciones simples de *Javascript* utilizadas para aislar la parte reutilizable de un componente funcional que permiten crear, acceder a estados y gestionar los efectos secundarios.

A continuación, en la Tabla *1.1*, se muestra una lista de las bibliotecas de *React* que se usaron en el proyecto.

Tabla 1.2. Librerías.

Librería	Definición
React **Bootstrap**	Es una implementación de los componentes de *Bootstrap* (un marco CSS gratuito y de código abierto para el desarrollo web front-end y móvil) usando *React* [10].
Formik	Es la biblioteca de formularios de código abierto para *React* y *React Native* [11].
Yup	Es un generador de esquemas de *Javascript* para el análisis y la validación de valores [12].
Axios	Es una biblioteca de cliente HTTP que le permite realizar solicitudes a un servidor determinado [13].

1.3.7. STACK MERN Y MVC

Las aplicaciones MERN Stack siguen un patrón de arquitectura de 3 capas que se basa en un modelo MVC[2] típico.

A continuación, se describirá como se relaciona el Stack MERN con la arquitectura MVC (Figura *1.8*).

- **Modelo:** representa la estructura de los datos, el formato y las restricciones con las que se almacenan. Todos los datos de la aplicación se almacenan en la base de datos *MongoDB.*
- **Controlador:** su lógica es programada mediante *Express* y *Node.js*. Este nivel representa el servidor de aplicaciones que actuará como puente de comunicación para el cliente y la base de datos.
- **Vista:** *React* sirve como la Vista en la arquitectura MVC. Este nivel de la arquitectura es con el que el usuario interactuará para acceder a las funcionalidades de nuestra aplicación.

[2] MVC: Modelo Vista Controlador

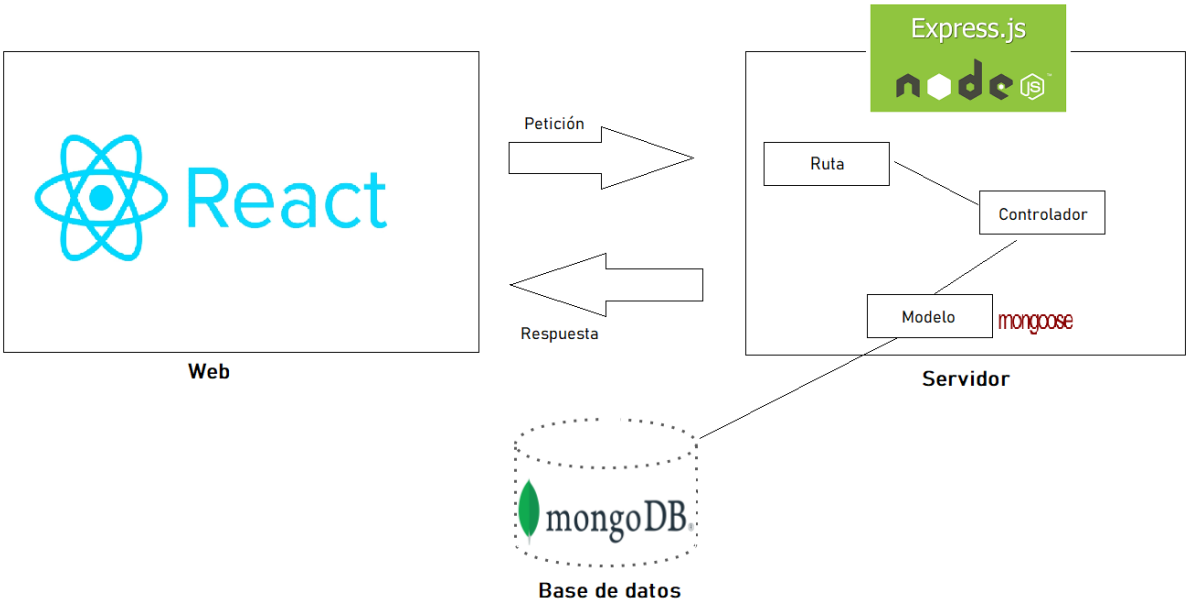

Figura 1.8. Sack MERN y MVC.

1.3.8. HERRAMIENTAS

En esta sección se describen los programas que se usaron en el desarrollo del proyecto.

Visual Studio Code

Visual Studio Code es un editor de código ligero disponible para Windows, macOS y Linux, que tiene soporte para *Javascript*, *TypeScript* y *Node.js*, que permite trabajar con un código individual o con sistemas de archivos basados en carpetas. Además, cuenta con depuradores que se pueden utilizar mediante extensiones [10].

Postman

Postman es una plataforma enfocada en crear peticiones sobre APIs (*Application Programming Interface*) de una forma sencilla para así probar dichas APIs [11] . Este programa tiene métodos que permiten tomar acción ante peticiones bien conocidas como:

- GET: obtener información
- POST: añadir información
- PUT: editar la información
- DELETE: eliminar información

Github

Es una plataforma que permite guardar código fuente y permite control de versiones y la colaboración entre usuarios [12].

2. METODOLOGÍA

En el presente capítulo se describe el análisis y el diseño del sistema. Para la primera etapa fue necesario determinar la situación actual de la parroquia, por lo que se recolectó información mediante entrevistas a encargados de dos negocios, de tal forma que pudo establecerse los requerimientos. Posteriormente, se realizaron los diagramas de casos de uso, secuencia y clases con el fin de modelar el sistema web. Finalmente, se diseñaron las interfaces gráficas.

En la Figura *2.1* se muestra el conjunto de tareas del tablero Kanban que se van a ir llevando a cabo a lo largo del desarrollo del sistema web. En cada sección se va a mostrar actualizaciones.

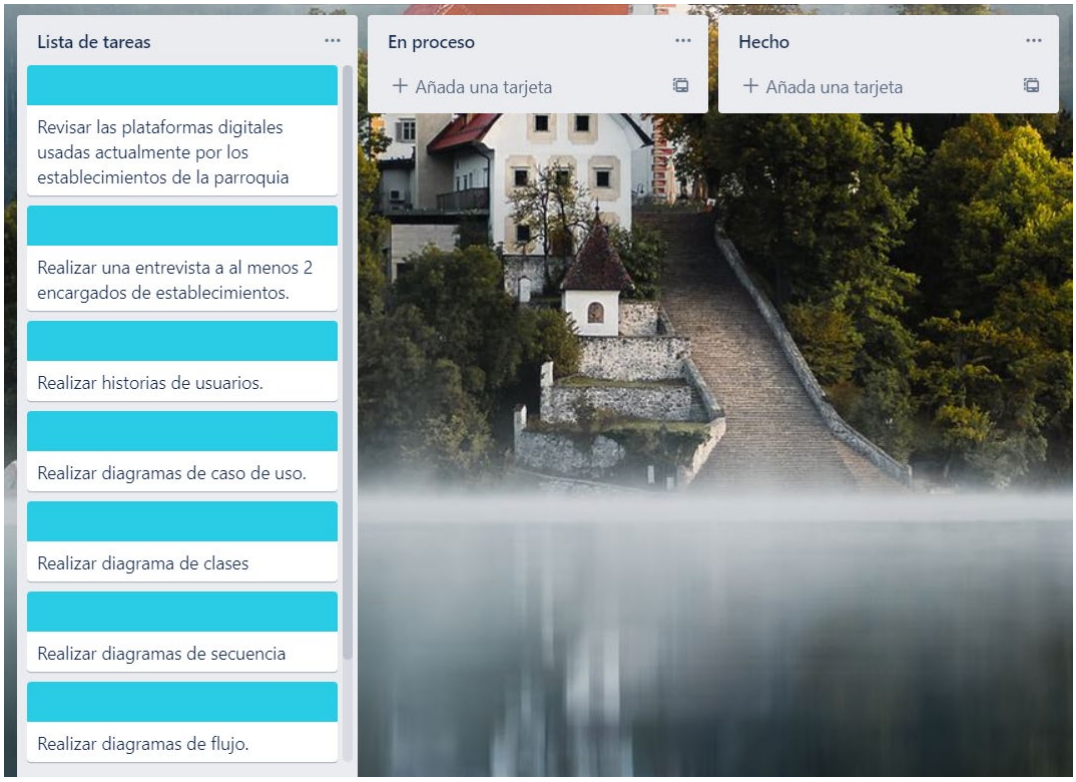

Figura 2.1 Tablero Kanban, metodología.

2.1. ANÁLISIS

2.1.1. SITUACIÓN ACTUAL DE LA PARROQUIA

La Parroquia de San Miguel de Nono se encuentra ubicada en la Provincia de Pichincha, a 18 kilómetros hacia el Noroccidente del Distrito Metropolitano de Quito. Según datos suministrados por el Instituto Nacional de Estadística del Ecuador (INEC) en el 2010 [13] , las ocupaciones de los residentes abarcaban principalmente el sector primario, representado por los peones y jornaleros con el 27,19%, empleados privados con el 22,36%, y por los que trabajan “por cuenta propia” con el 27,83%. Este 27,83% refleja la gran iniciativa que tienen los pobladores para generar sus propios negocios.

Entre estas actividades por cuenta propia se incluyen varias actividades turísticas que, debido al lugar privilegiado tanto por su naturaleza, como por la cercanía a la ciudad, han tenido gran acogida entre los visitantes. Las actividades turísticas incluyen: servicio de alimentación, alojamiento, paseos en reservas naturales, entre otras, que han ido creciendo con el paso de los años. Junto con este crecimiento, los ingresos por este rubro también lo han hecho.

Sin embargo, debido a la pandemia producida por el virus COVID-19, y por las medidas sanitarias temporales que se han venido adoptando desde el 4 de mayo del 2020 [14], ese crecimiento se detuvo. Incluso hubo un decremento en los ingresos generados por dichas actividades. Aunque no se tienen datos de la parroquia, tenemos datos a nivel nacional donde se puede apreciar una reducción del 52,4% [15] (equivalente a 2.141 millones de USD) de los ingresos percibidos por actividades turísticas del año 2020 respecto al año 2019. En este contexto, los negocios de Nono han experimentado varios problemas. Debido a las medidas de confinamiento, se detuvieron las visitas de turistas a la parroquia. Levantadas las medidas de confinamiento, hubo disminución de aforo; por lo que se necesitaba realizar un control del mismo. A esto se suma, el hecho de que varios de los negocios no tienen visibilidad a través de Internet y por ende los turistas se enteran de las actividades y productos ofertados, incluso del aforo permitido en cada local, cuando ya se encuentran en el establecimiento.

Una vez planteado el panorama actual, se procedió a realizar entrevistas dirigidas a dueños de dos establecimientos: Cascadas de Guagrapamba y Alpasungana. La información obtenida se muestra en la Tabla *2.1*.

Tabla 2.1 Resumen de entrevistas a dueños de establecimientos.

	Cascadas de Guagrapamba	**Alpasungana**
Encargada	Carmen Beatriz Salas Cargua	Margiory Piedad Donoso Chacon
Edad	63 años	46 años
Negocio establecido desde	2010	2010
Actividades	Caminatas, picnic, almuerzos, desayunos, acampadas.	Restaurante y heladería.
Visibilidad del negocio	Redes sociales, pagina web	Redes sociales y boca a boca.
Reservas	Sí, con adelanto previo.	Sí, con adelanto previo.
Formas de contacto	Teléfono, redes sociales y correo electrónico.	Teléfono y redes sociales.
Horario de atención	Fin de semana y feriado. Con reserva se puede abrir otros días.	Fin de semana y feriado.

Gracias a las entrevistas y a la revisión de sus plataformas digitales fue posible realizar las historias de usuario.

En la Figura 2.2 se muestra la actualización del tablero Kanban.

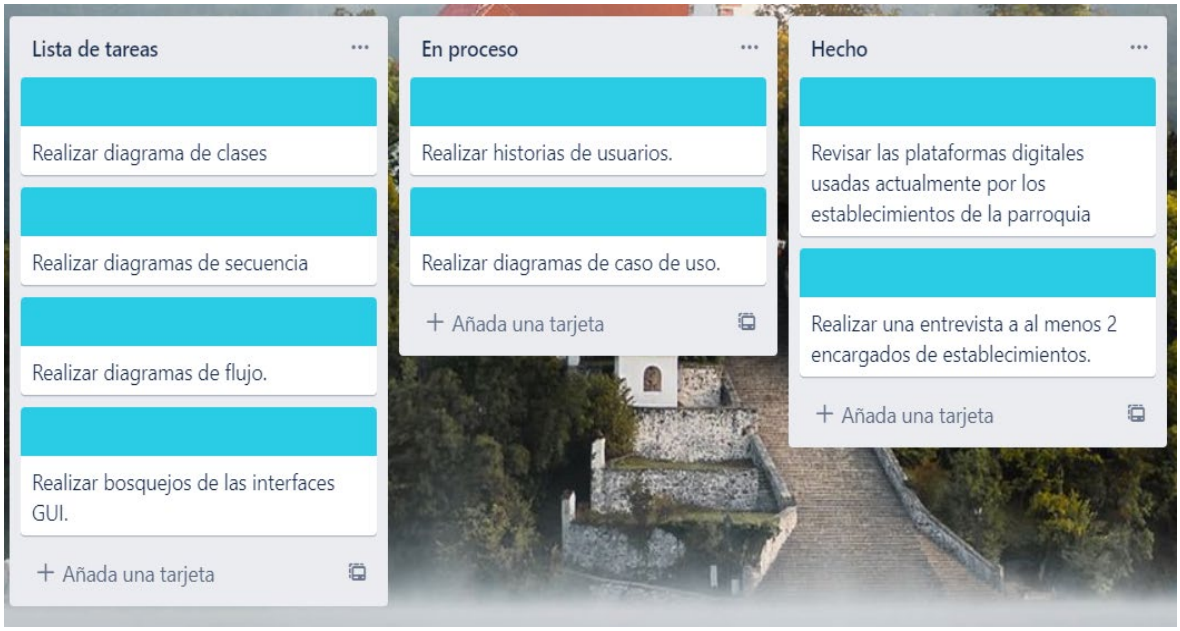

Figura 2.2. Actualización tablero Kanban, diseño.

2.1.2. HISTORIAS DE USUARIO

Se definió una plantilla simple de historia de usuario[3] (Tabla 2.2), cuyos campos son:

1. **ID:** identificador de la historia de usuario. Se adoptan las siglas H, seguidas de un número de dos dígitos.
2. **Nombre:** nombre descriptivo de la historia de usuario, en línea con el requerimiento del usuario.
3. **Usuario:** persona que va a realizar las acciones definidas en la historia de usuario.
4. **Programador:** encargado de desarrollar la funcionalidad en el sistema.
5. **Descripción:** texto que consolida brevemente el propósito del requerimiento.
6. **Criterios de aceptación:** requisitos a cumplir para aceptar la funcionalidad programada.

[3] Es una descripción general de una función de software realizada desde el punto de vista del usuario final [22]

Tabla 2.2. Historia de usuario ejemplo.

HISTORIA DE USUARIO	**H# (1)**
Nombre: (2)	
Usuario: (3)	
Programador responsable: (4)	
Descripción: (5)	
Criterios de aceptación: (6)	

A continuación, son detalladas las historias de usuario en un orden que favorezca el entendimiento del lector, y agrupadas dentro de los módulos definidos.

Se van a mostrar las historias de usuario más importantes de cada módulo, el resto se va a encontrar en los anexos.

Módulo del administrador

Dentro del módulo del administrador se va a manejar dos perfiles de usuario administrador principal[4] y administrador. A continuación, se detallan las historias de usuario para cada uno de ellos.

- **Administrador principal**

Tabla 2.3. Historia de usuario 1 - Gestionar administradores.

HISTORIA DE USUARIO	H1
Nombre: Gestionar administradores	
Usuario: Administrador principal	
Programador responsable: Sandy Donoso	
Descripción: como usuario administrador principal quiero gestionar a los administradores para controlar o restringir el acceso de usuarios a la plataforma.	

[4] Administrador principal: este rol va a estar creado desde el inicio y es quien tendrá la facultad de crear y gestionar nuevos administradores, adicional de las otras acciones que realiza el administrador general.

Criterios de aceptación. El administrador puede realizar las siguientes acciones: • Visualizar lista de administradores. • Crear administrador. • Modificar administrador. • Borrar administrador. • Visualizar administrador a detalle.

Las historias de usuario 2 y 3 se muestran en el Anexo A.

- **Administrador y Administrador principal**

Tabla 2.4. Historia de usuario 4 - Gestionar solicitudes de posibles encargados de establecimientos.

HISTORIA DE USUARIO	H4
Nombre: Gestionar solicitudes de posibles encargados de establecimientos	
Usuario: Administrador y Administrador principal	
Programador responsable: Sandy Donoso	
Descripción: como usuario administrador deseo gestionar las solicitudes que lleguen de posibles encargados de establecimientos que hayan realizado el registro a través del formulario respectivo. Quiero poder aceptar o rechazar la solicitud.	
Criterios de aceptación. El administrador puede realizar las siguientes acciones: • Visualizar lista de solicitudes de encargados de establecimientos pendientes. • Aceptar solicitud de encargado de establecimiento, en cuyo caso, pasa a la lista encargados de establecimientos. • Rechazar solicitud de encargado de establecimiento.	

Tabla 2.5. Historia de usuario 5 - Inicio de sesión en el sistema.

HISTORIA DE USUARIO	H5
Nombre: Inicio de sesión en el sistema	

Usuario: Administrador y Administrador principal
Programador responsable: Sandy Donoso
Descripción: Como administrador quiero ingresar al sistema para poder usar las opciones de gestión según mi perfil.
Criterios de aceptación: • El sistema debe pedir el correo electrónico y la contraseña para realizar el inicio de sesión del administrador. • El sistema tiene que validar que el correo electrónico y la contraseña no estén vacíos y coincidan con lo registrado. De no ser así debe mostrarse un mensaje de error que indique “correo electrónico o contraseña incorrectos”. • El sistema debe validar que se ingrese como perfil administrador o administrador principal.

Módulo del encargado de establecimiento

Dentro de este módulo se tendrá el perfil encargado de establecimiento. A continuación, se detallan las historias de usuario para este perfil.

- **Encargado de establecimiento (Ingreso de datos al sistema)**

Tabla 2.6. Historia de usuario 6 - Registro en el sistema.

HISTORIA DE USUARIO	H6
Nombre: Registro en el sistema	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero poder registrarme en el sistema con mis datos.	
Criterios de aceptación. • El registro debe pedir datos al encargado del establecimiento: nombres completos, dirección de correo electrónico y contraseña. • Cada campo debe ser validado antes de su registro.	

Las historias de usuario 7, 8, 9, 10 y 11 se muestran en el Anexo A.

Las siguientes historias de usuario se las realiza considerando la existencia de reservas solicitadas a través del módulo turista.

Tabla 2.7. Historia de usuario 12 - Reserva.

HISTORIA DE USUARIO	H12
Nombre: Reserva	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero que las reservas, cuando cumplan con determinadas condiciones, se confirmen de manera automática; y cuando no, que queden pendientes. Cuando no se cumplan las condiciones, se debe abrir un chat con el turista para llegar a establecer condiciones específicas previo a su reserva.	
Criterios de aceptación: • Las reservas que se guarden automáticamente como confirmadas van a ser aquellas donde la cantidad de reservas sea menor a 10 y la actividad escogida dure menos de un día. • La reserva que se guarden como pendientes van a ser aquellas cuya cantidad de reserva sea mayor a 10 o la actividad escogida dure más de un día. • Si la reserva es pendiente, se va a abrir un link de WhatsApp con la información del formulario para que el turista establezca una conversación con el encargado, para así llegar a un acuerdo de confirmación o no.	

Tabla 2.8. Historia de usuario 13 - Gestión de reservas confirmadas.

HISTORIA DE USUARIO	H13
Nombre: Gestión de reservas confirmadas	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	

Descripción: como usuario encargado de establecimiento deseo eliminar las reservas confirmadas o ingresar nuevas de manera manual. Las reservas confirmadas no pueden editarse.
Criterios de aceptación: • Visualizar lista de reservas confirmadas. • Crear reserva confirmada. • Borrar reserva confirmada. • Visualizar reserva confirmada a detalle.

Tabla 2.9. Historia de usuario 14 - Gestionar reservas pendientes.

HISTORIA DE USUARIO	H14
Nombre: Gestionar reservas pendientes	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero gestionar las reservas que se encuentren en estado pendiente.	
Criterios de aceptación: • Visualizar lista de reservas pendientes. • Aceptar reserva pendiente (pasa a reserva confirmada). • Modificar reserva pendiente. • Borrar reserva pendiente. • Visualizar reserva pendiente a detalle.	

Módulo de turista

En este módulo se tendrá las actividades a desarrollarse desde el perfil turista. Este perfil representa un visitante de la web que no requiere autenticarse.

- **Turista**

Tabla 2.10. Historia de usuario 15 - Generar reservas.

HISTORIA DE USUARIO	H15
Nombre: Generar reservas	
Usuario: Turista	
Programador responsable: Sandy Donoso	
Descripción: como usuario turista quiero ver información detallada de las actividades ofertadas por un establecimiento, y reservar a través de un formulario para asegurar un cupo.	
Criterios de aceptación: • Visualizar categorías. • Visualizar actividades por categorías. • Visualizar perfil de la actividad. • Visualizar horarios disponibles. • Realizar reserva mediante formulario. El formulario al menos va a contener nombres, apellidos, teléfono, correo electrónico y cantidad de reservas (cantidad de personas que van a ir o cantidad de platos). • Establecer términos específicos a través de Whatsapp.	

En la Tabla *2.11* se muestra el detalle de los requisitos que se tendrán en cuenta por cada historia de usuario.

Tabla 2.11. Listado de requisitos e historias de usuario.

Historia de usuario	**Número Requerimiento**	**Descripción**
H1	R1	Visualizar lista de administradores.
H1	R2	Crear administrador.
H1	R3	Modificar administrador.
H1	R4	Borrar administrador.

H1	R5	Visualizar administrador a detalle.
H2	R6	Visualizar lista de categorías.
H2	R7	Ingresar categoría.
H2	R8	Modificar categoría.
H2	R9	Borrar categoría.
H2	R10	Visualizar categoría a detalle.
H3	R11	Visualizar lista de encargados de establecimientos.
H3	R12	Crear encargado de establecimientos.
H3	R13	Modificar encargado de establecimiento.
H3	R14	Borrar encargado de establecimiento.
H3	R15	Visualizar encargado de establecimiento a detalle.
H4	R16	Visualizar lista de solicitudes de encargados de establecimientos pendientes.
H4	R17	Aceptar solicitud de encargado de establecimiento
H4	R18	Rechazar solicitud de encargado de establecimiento
H5	R19	Inicio de sesión del administrador.
H6	R20	Registro del encargado del establecimiento
H7	R21	Inicio de sesión del encargado.
H8	R22	Visualizar lista de establecimientos.
H8	R23	Crear establecimiento.
H8	R24	Modificar establecimiento.
H8	R25	Borrar establecimiento.
H8	R26	Visualizar establecimiento a detalle.
H9	R27	Visualizar lista de actividades.
H9	R28	Crear actividad.

H9	R29	Modificar actividad.
H9	R30	Borrar actividad.
H9	R31	Visualizar actividad a detalle.
H10	R32	Visualizar lista de horarios.
H10	R33	Crear horario.
H10	R34	Borrar horario.
H10	R35	Visualizar horario a detalle.
H11	R36	Visualizar lista de platos.
H11	R37	Crear plato.
H11	R38	Modificar plato
H11	R39	Borrar plato.
H11	R40	Visualizar plato a detalle.
H13	R41	Visualizar lista de reservas confirmadas.
H13	R42	Crear reserva confirmada.
H13	R43	Borrar reserva confirmada.
H13	R44	Visualizar reserva confirmada a detalle.
H14	R45	Visualizar lista de reservas pendientes.
H14	R46	Aceptar reserva pendiente (pasa a reserva confirmada).
H14	R47	Modificar reserva pendiente.
H14	R48	Borrar reserva pendiente.
H14	R49	Visualizar reserva pendiente a detalle.
H15	R50	Visualizar categorías.
H15	R51	Visualizar actividades por categorías.
H15	R52	Visualizar perfil de la actividad.

H15	R53	Visualizar horarios disponibles.
H15	R54	Realizar reserva mediante formulario
H12	R55	Colocar reserva en estado pendiente o confirmada.
H15	R56	Establecer términos específicos a través de Whatsapp.

2.1.3. REQUERIMIENTOS NO FUNCIONALES

En la Tabla *2.12* se detallan los requerimientos no funcionales.

Tabla 2.12. Requerimientos no funcionales.

Número de requerimiento	Descripción
RNF1	Se debe usar Whatsapp como servicio de mensajería para el intercambio de mensajes entre el encargado y el cliente.

2.1.4. DIAGRAMA DE CASOS DE USO

Este es un diagrama de comportamiento que representa la manera en que los usuarios interactúan con el sistema [16] y está conformado por: actores (quienes utilizan el sistema), casos de uso (elipses que indican la acción a realizarse en el sistema) y sistema (se utiliza para definir el alcance del caso de uso y se dibuja como un rectángulo). Se ha identificado cuatro actores en el sistema: administrador principal, administrador, encargado de establecimiento y turista.

El administrador principal va a poder autenticarse, gestionar administradores, gestionar encargados de establecimientos, categorías y gestionar solicitudes de encargados (Figura 2.3). Como administrador va a poder realizar las mismas acciones que el administrador principal, excepto gestionar administradores (Figura *2.4*). Por su parte, el encargado del establecimiento podrá autenticarse, gestionar establecimientos, reservas, actividades y horarios (Figura *2.5*). En caso de que la actividad sea comida, además de las acciones anteriores, el encargado va a poder gestionar platos (Figura *2.6*). Finalmente, el turista será capaz de visualizar las actividades y generar la reserva de aquella que le interese (Figura *2.7*).

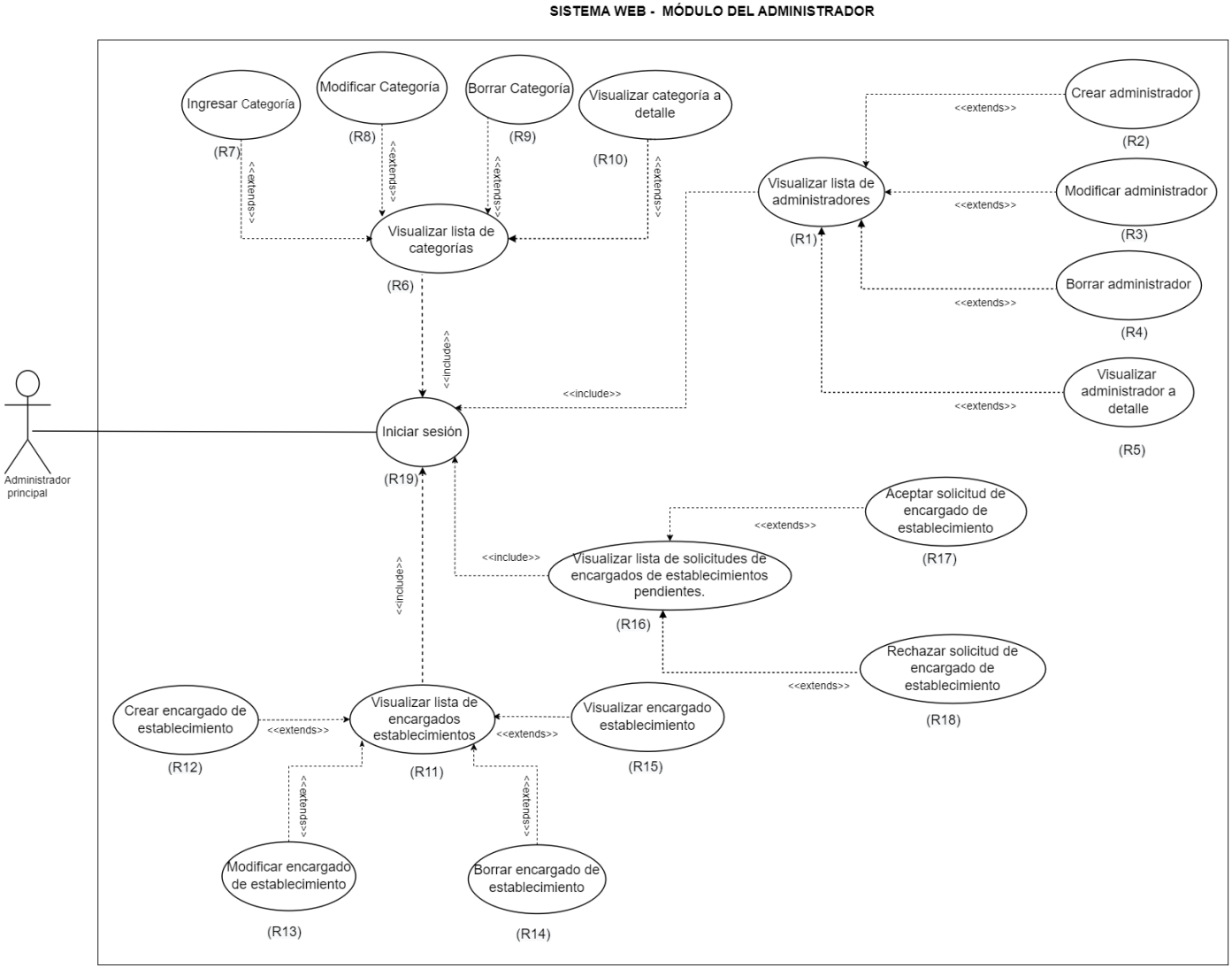

Figura 2.3. Diagrama de caso de uso del administrador principal.

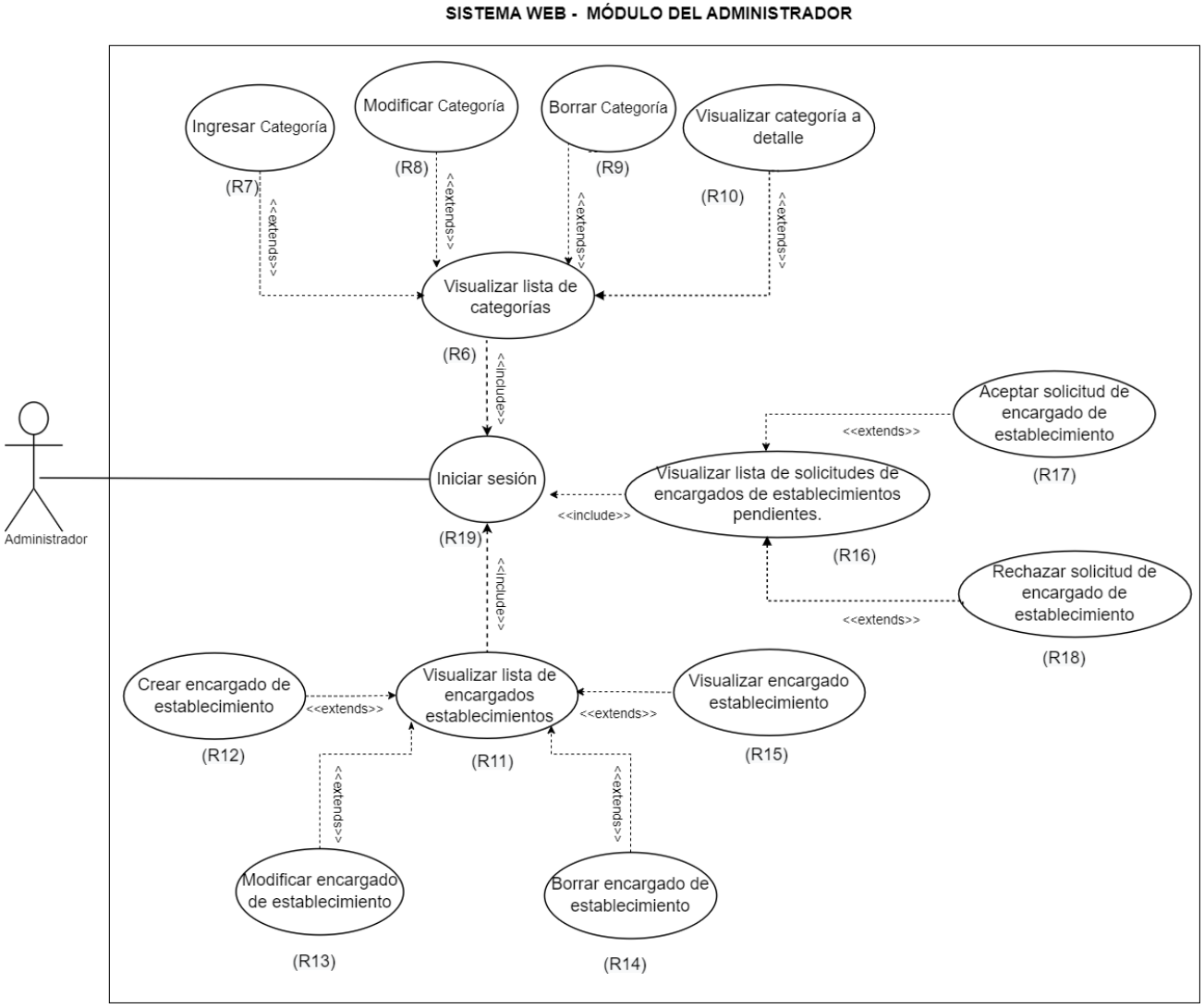

Figura 2.4. Diagrama de caso de uso del administrador.

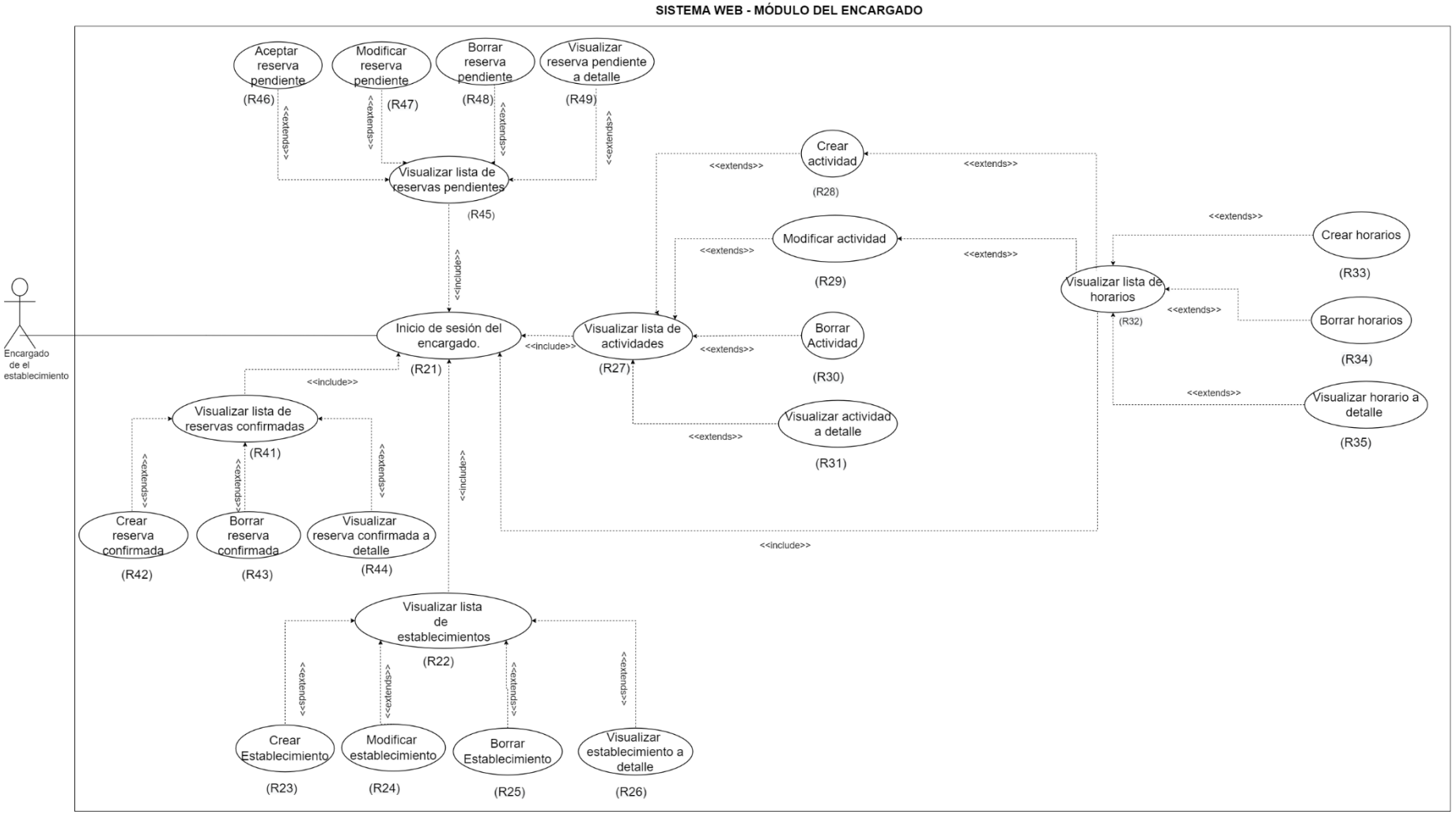

Figura 2.5. Diagrama de casos de uso del encargado de un establecimiento, todas las actividades.

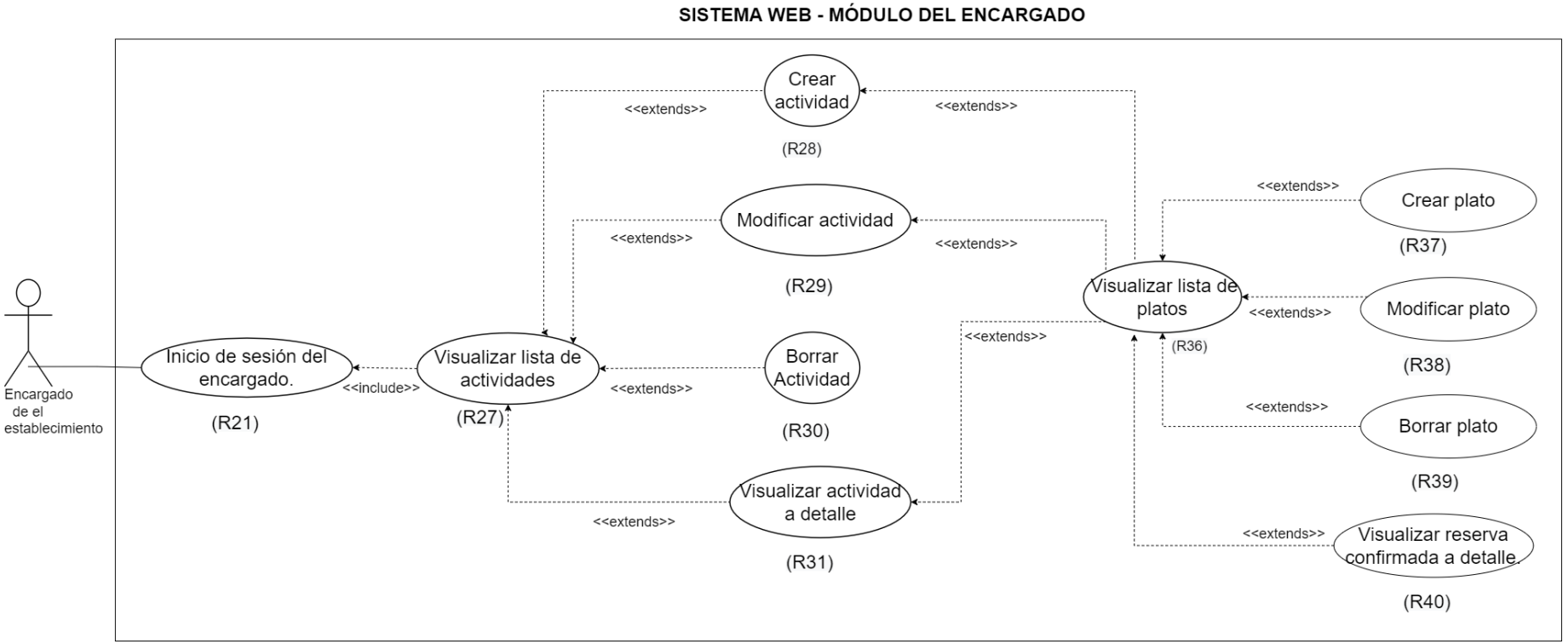

Figura 2.6. Diagrama de casos de uso del encargado de un establecimiento, actividad alimentación.

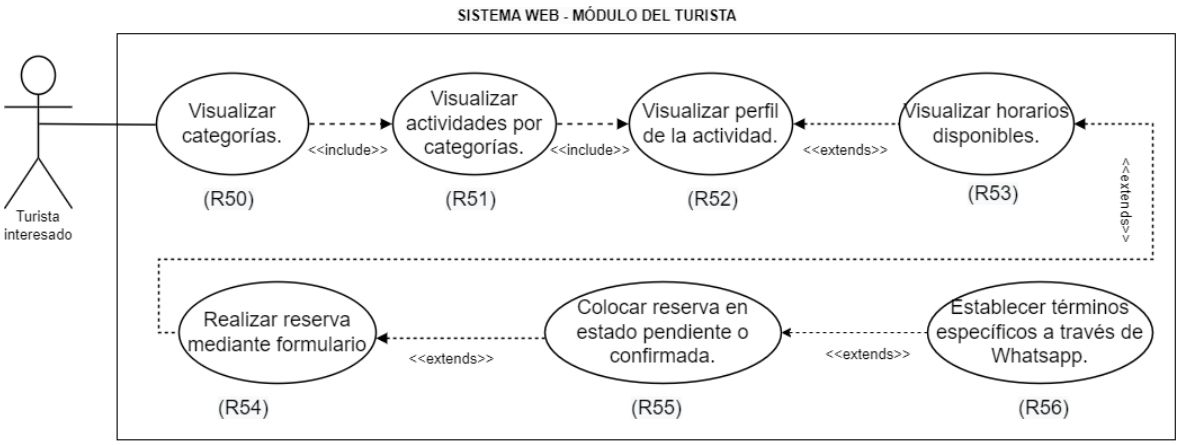

Figura 2.7. Diagrama de caso de uso del turista.

2.2. DISEÑO

Una vez finalizados los diagramas de casos de uso se procede con la realización de los diagramas de clases de secuencia y de flujo, se actualiza el tablero Kanban (Figura 2.8).

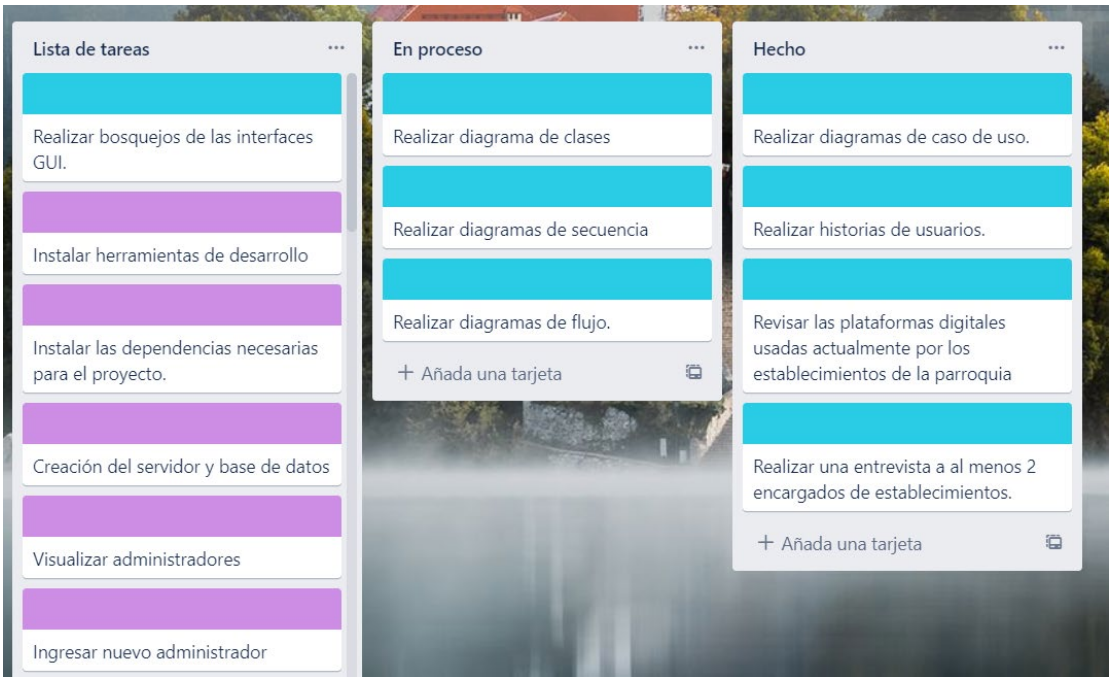

Figura 2.8. Actualización del tablero Kanban, diagrama de casos de uso.

2.2.1. DIAGRAMA DE SECUENCIA

El diagrama de secuencia es una representación simultánea de elementos que interactúan dentro del sistema, así como de los mensajes que intercambian entre ellos [17].

En los siguientes párrafos, se muestran los procesos más relevantes alineados al objetivo de este proyecto: realizar una reserva con confirmación manual o realizar una reserva con confirmación automática.

En el caso de reserva con confirmación manual, el turista ingresa a la página web (Figura *2.9*) donde visualizará las categorías disponibles, él escoge alguna de su interés y se va a redireccionar a una página donde se muestren las actividades correspondientes a esa categoría. El turista puede escoger alguna de esas actividades para poder verla a detalle. Estando ahí, el cliente podrá iniciar el proceso de reserva, para lo cual se le van a mostrar los horarios disponibles. Luego que el turista escoge el horario, se despliega un formulario para que llene su información de contacto. En este caso (reserva mayor a 10 o actividad escogida de más de un día), la reserva se guarda como pendiente, por lo que tiene que ser confirmada manualmente por el encargado. Para esto, al turista se le va a abrir automáticamente su aplicación de WhatsApp con la información del formulario para que envíe un mensaje al encargado del establecimiento. Finalmente, una vez que ambas partes lleguen a un acuerdo, el encargado coloca manualmente la reserva como confirmada.

Por otra parte, si la actividad escogida por el turista dura menos de un día y la cantidad de reserva es menor a 10, se va a guardar como reserva confirmada automáticamente (Figura *2.10*).

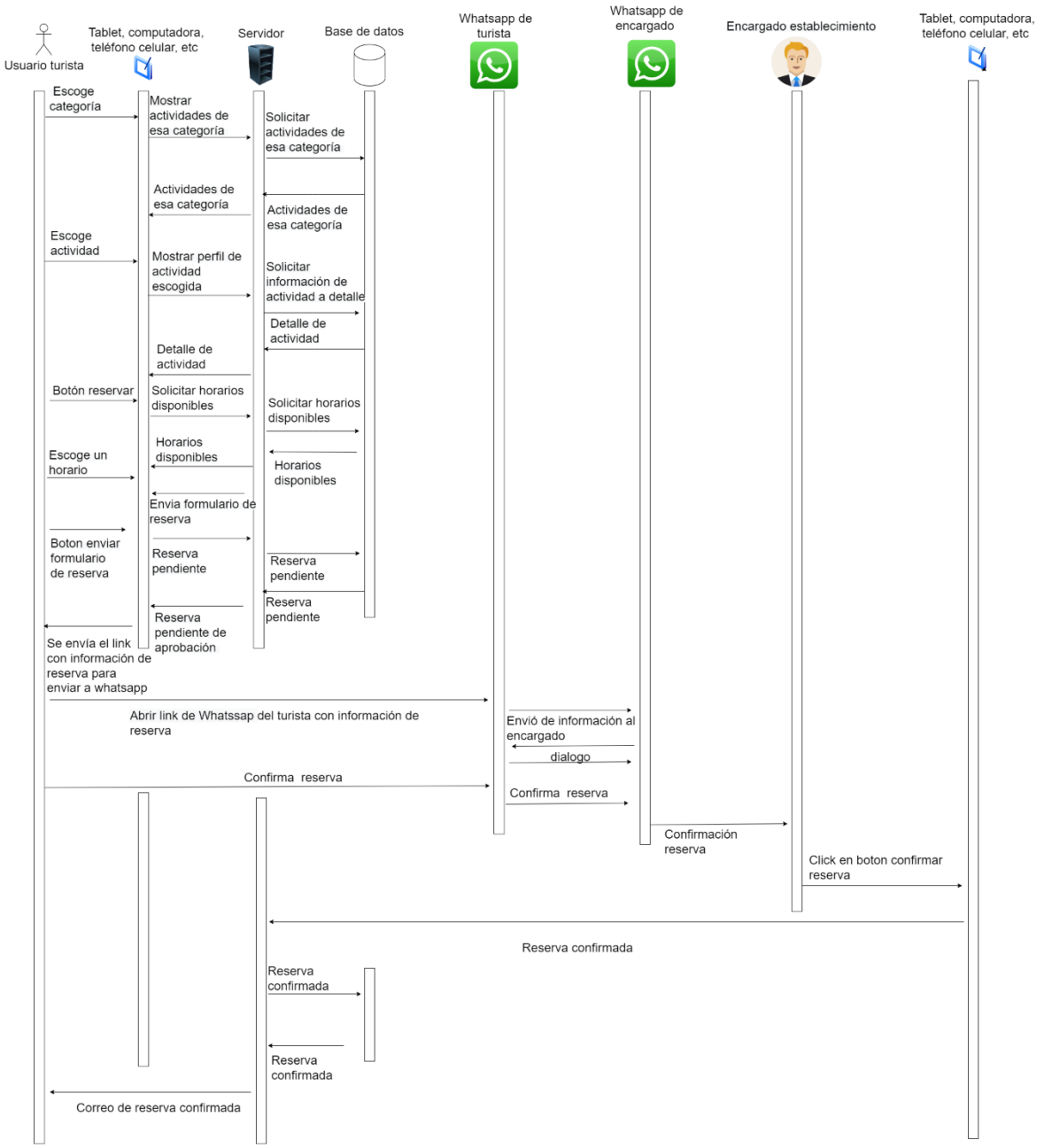

Figura 2.9. Diagrama de secuencia de una reserva manual.

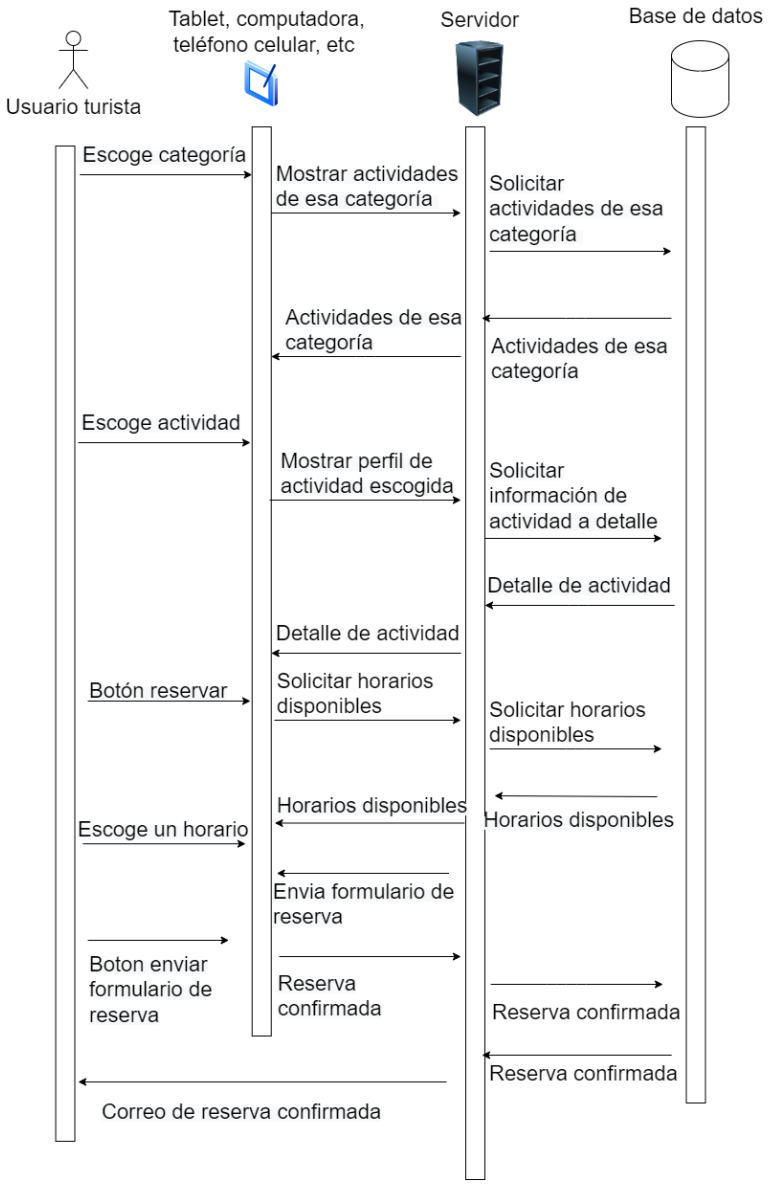

Figura 2.10. Diagrama de secuencia de una reserva automática.

2.2.2. DIAGRAMA DE CLASES

Es un diagrama que muestra las clases y relaciones. Una clase es la descripción de una serie de objetos que tienen las mismas características. En cambio, las relaciones son los vínculos que hay entre las clases y están representados en el diagrama con una línea que los conecta [18].

El diagrama de clases del sistema web (Figura 2.11) cuenta con 12 clases que representan las entidades: persona, administrador, turista, encargado de establecimiento, solicitud de encargado de establecimiento, actividad, establecimiento, categoría, plato, horario de actividad, reserva y platos reserva.

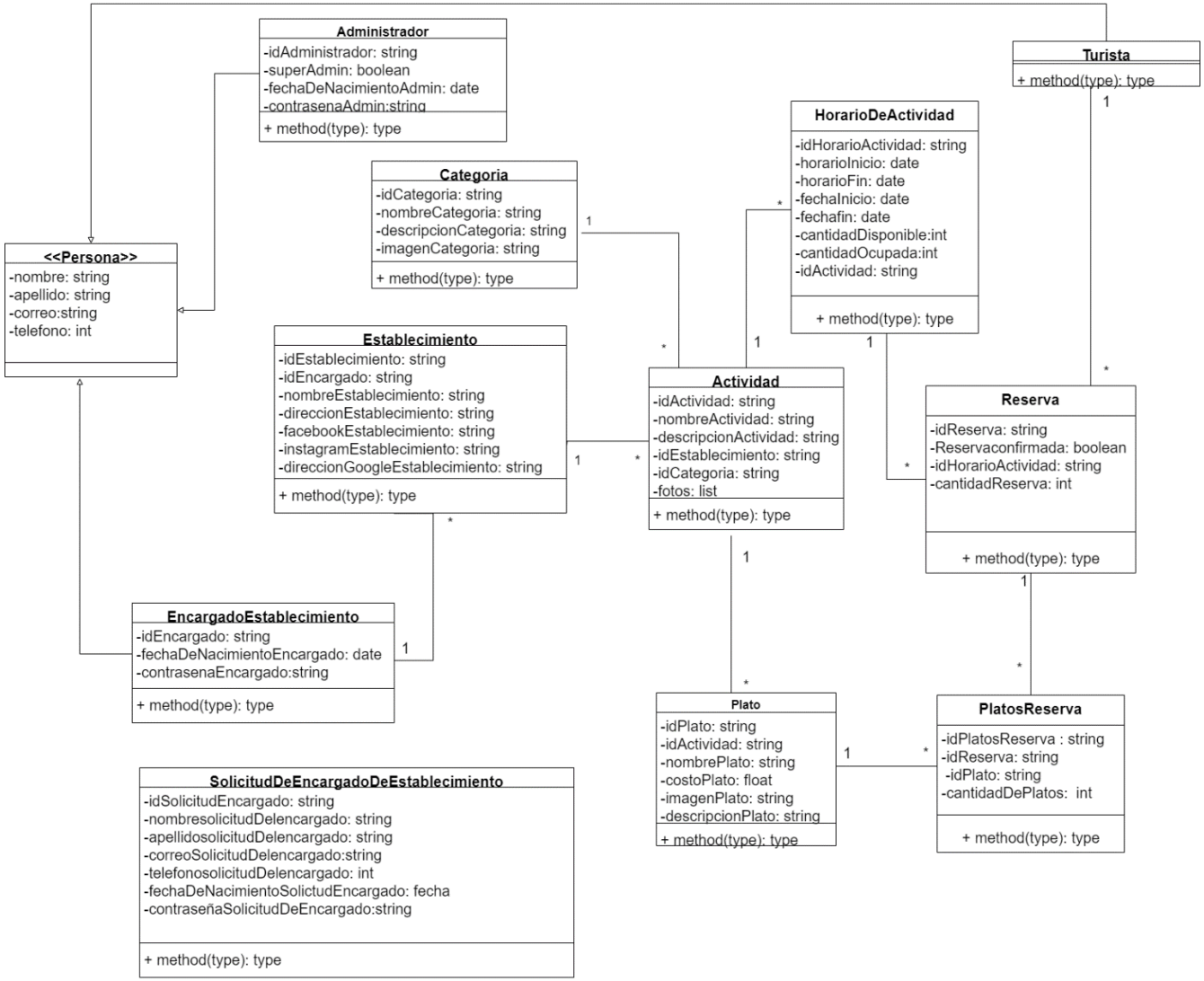

Figura 2.11. Diagrama de clases del sistema web.

2.2.3. BOSQUEJOS

Esta sección detalla los bosquejos de la interfaz gráfica para el sistema web. Se actualiza el tablero Kanban con las nuevas tareas en proceso y finalizadas (Figura *2.12*).

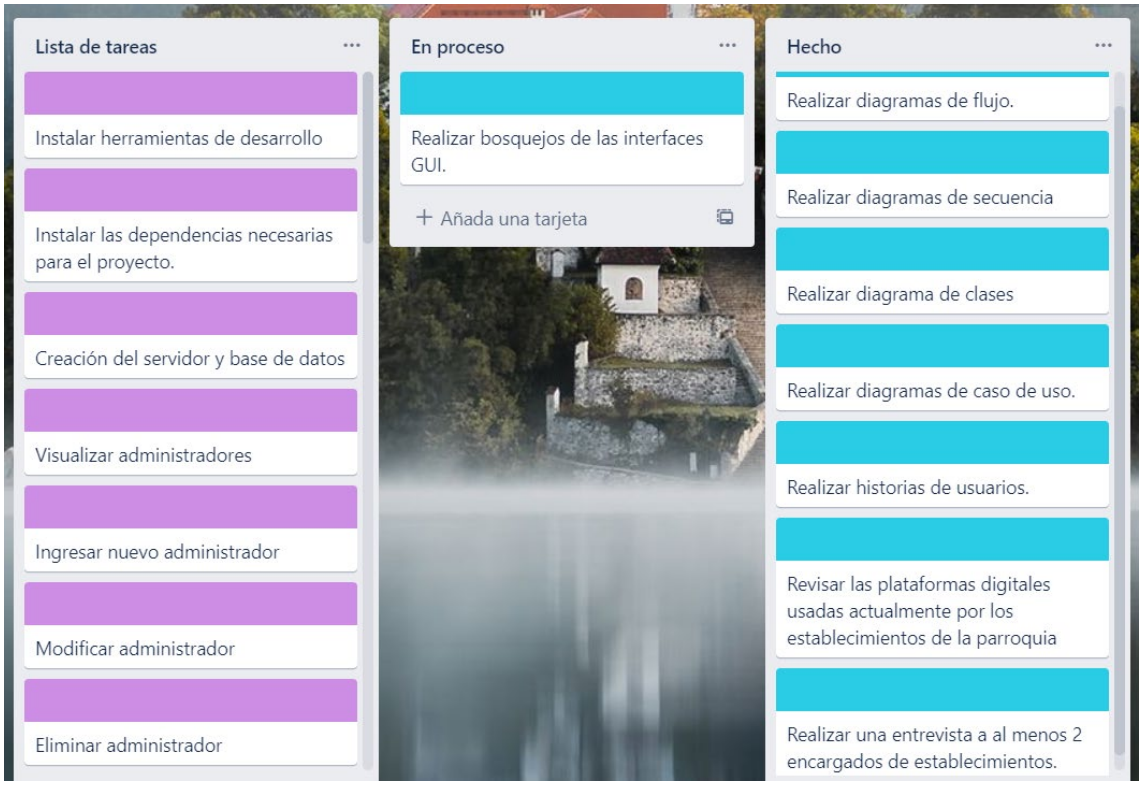

Figura 2.12. Actualización del tablero Kanban, bosquejos.

Para comenzar, se muestra el ingreso al sistema web (Figura 2.13) donde los usuarios, administrador principal, administradores y encargados de establecimientos tienen la opción de iniciar sesión para poder revisar los módulos de gestión de acuerdo a su perfil. Además, si una persona se encuentra interesada en ser encargada de un establecimiento tiene la opción de registrarse en el sistema (Figura 2.14) para su posterior revisión y aprobación.

Figura 2.13. Inicio de sesión.

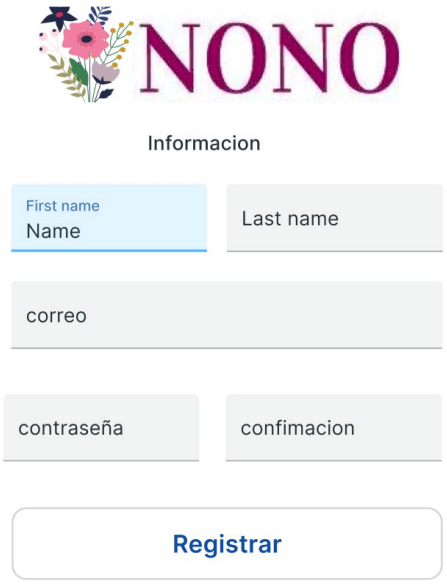

Figura 2.14. Registro del encargado de establecimiento.

Módulo del administrador

Este módulo cuenta con dos perfiles: perfil administrador principal, el cual puede gestionar encargados, categorías y administradores, y perfil administrador el cual puede realizar las mismas funciones que el administrador principal excepto la gestión de administradores.

En la Figura 2.15 se muestra la interfaz de usuario para la gestión de administradores, mediante la cual el administrador principal podrá crear, editar, eliminar y visualizar administradores.

Figura 2.15. Interfaz del administrador principal para gestión de administradores.

En la Figura 2.16 se observa la interfaz de usuario para la gestión encargados, mediante la cual el administrador podrá crear, editar, eliminar y visualizar encargados.

Figura 2.16. Interfaz del administrador para gestión de encargados.

Desde la interfaz de usuario para gestión de categorías el administrador podrá crear, editar, eliminar y visualizar categorías como se observa en la Figura 2.17.

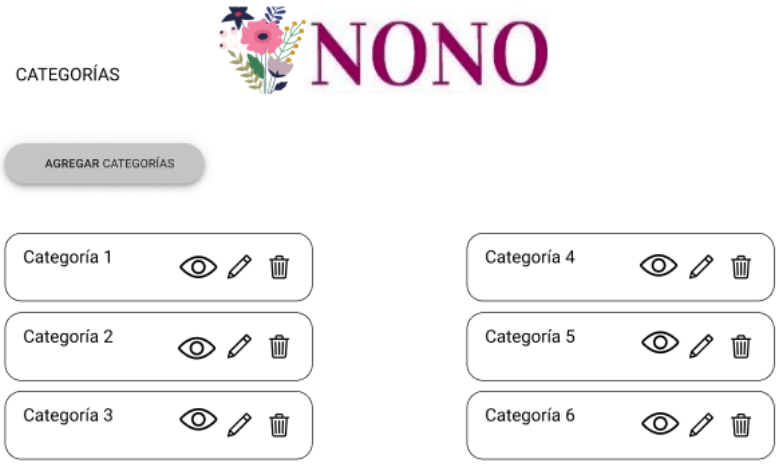

Figura 2.17. Interfaz del administrador para gestión de categorías.

Módulo encargado de establecimiento

A continuación, se muestran las diferentes interfaces de la aplicación web a las que tendrá acceso el encargado de establecimiento.

Primero, se muestra la gestión de establecimientos, en la que el encargado puede visualizar una lista de los establecimientos que se encuentran a su cargo (Figura 2.18) o agregar un nuevo establecimiento (Figura 2.19). Cada establecimiento de la lista cuenta con la opción de editar, ver a detalle y eliminar.

Figura 2.18. Gestión de establecimientos.

Figura 2.19. Ingresar datos del establecimiento.

El encargado del establecimiento también tiene la opción de gestionar actividades. Para esto, se mostrará la lista de actividades disponibles (Figura 2.20), las cuales van a poder ser editadas, eliminadas o vistas a detalle. Asimismo, se puede ingresar una actividad. En la Figura 2.21 se muestra el proceso de ingreso de una actividad que incluye el ingreso de datos como nombre, descripción, la categoría a la que pertenece y el ingreso de imágenes (Figura 2.22).

Figura 2.20. Lista de actividades disponibles.

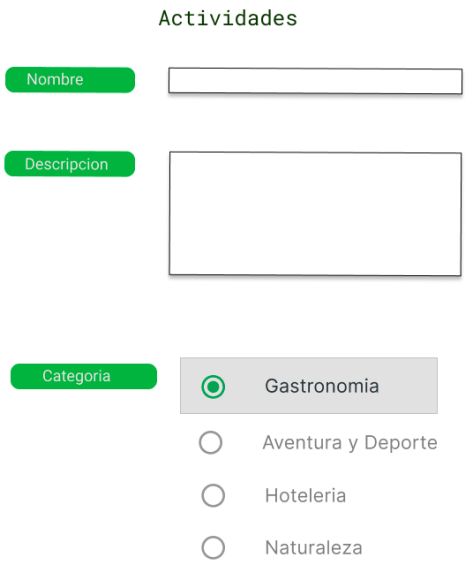

Figura 2.21. Ingreso de datos para la creación de una actividad.

Figura 2.22. Ingreso de imágenes de una actividad

Asociada a una actividad, se tiene una lista de horarios (Figura 2.23). Aquí se tiene la opción de agregar un nuevo horario, ver o eliminarlo (Figura 2.24).

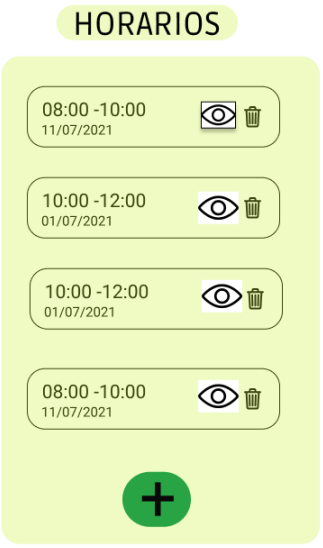

Figura 2.23. Lista de horarios.

Figura 2.24. Ingreso de un nuevo horario.

Cuando un cliente ha generado una reserva, el encargado del establecimiento tiene también la facultad de gestionar las reservas pendientes (Figura 2.25) y confirmadas (Figura 2.26).

Figura 2.25. Lista de reservas confirmadas.

Figura 2.26. Lista de reservas pendientes.

Módulo turista

A continuación, se muestra la Figura 2.27 que corresponde a la página principal del sistema web, a la cual tienen acceso a todos los usuarios incluyendo los turistas interesados. Esta página muestra las categorías registradas previamente por el administrador. El usuario turista puede seleccionar una de las categorías de su interés para que, en la siguiente página (Figura 2.28), se muestren las actividades disponibles de dicha categoría.

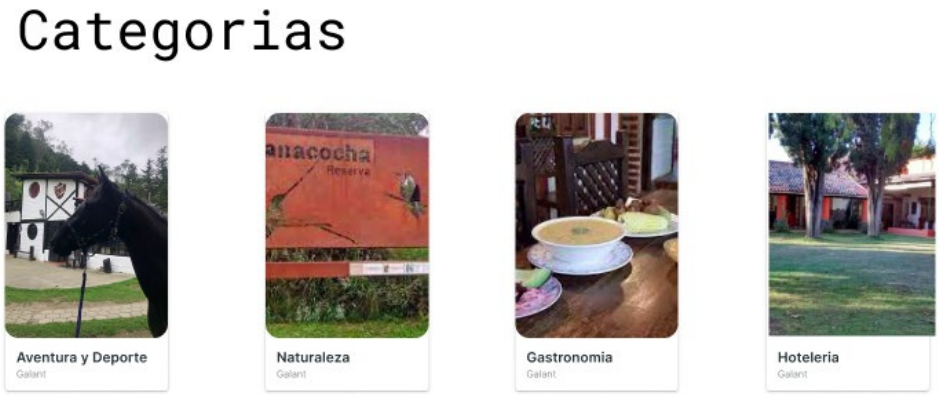

Figura 2.27. Página web con las categorías disponibles.

Figura 2.28. Página web con las actividades disponibles.

A continuación, el usuario turista interesado puede ingresar a cualquiera de las actividades registradas para ver más información de la misma (Figura 2.29).

Figura 2.29. Perfil de la actividad seleccionada.

El usuario puede realizar una reserva de una actividad; y solo si la categoría de la actividad es alimentación, se abre una ventana con el menú de platos disponible (Figura 2.30). El proceso de reserva continúa al visualizar los horarios disponibles para reservar (Figura 2.31).

Figura 2.30. Menú de platos.

Figura 2.31. Lista de horarios disponibles.

Una vez el turista escoja un horario, se va a desplegar un formulario (Figura 2.32) donde va a colocar sus datos para la reserva.

Figura 2.32. Formulario para realizar reserva.

Finalizados los bosquejos se procede con la implementación del programa.

2.3. IMPLEMENTACIÓN

En esta sección se detalla el proceso seguido para la implementación del sistema web. Esta sección se divide en tareas (Figura 2.33) que incluyen la instalación de herramientas de desarrollo, creación del servidor, creación de la base de datos e implementación del sistema web.

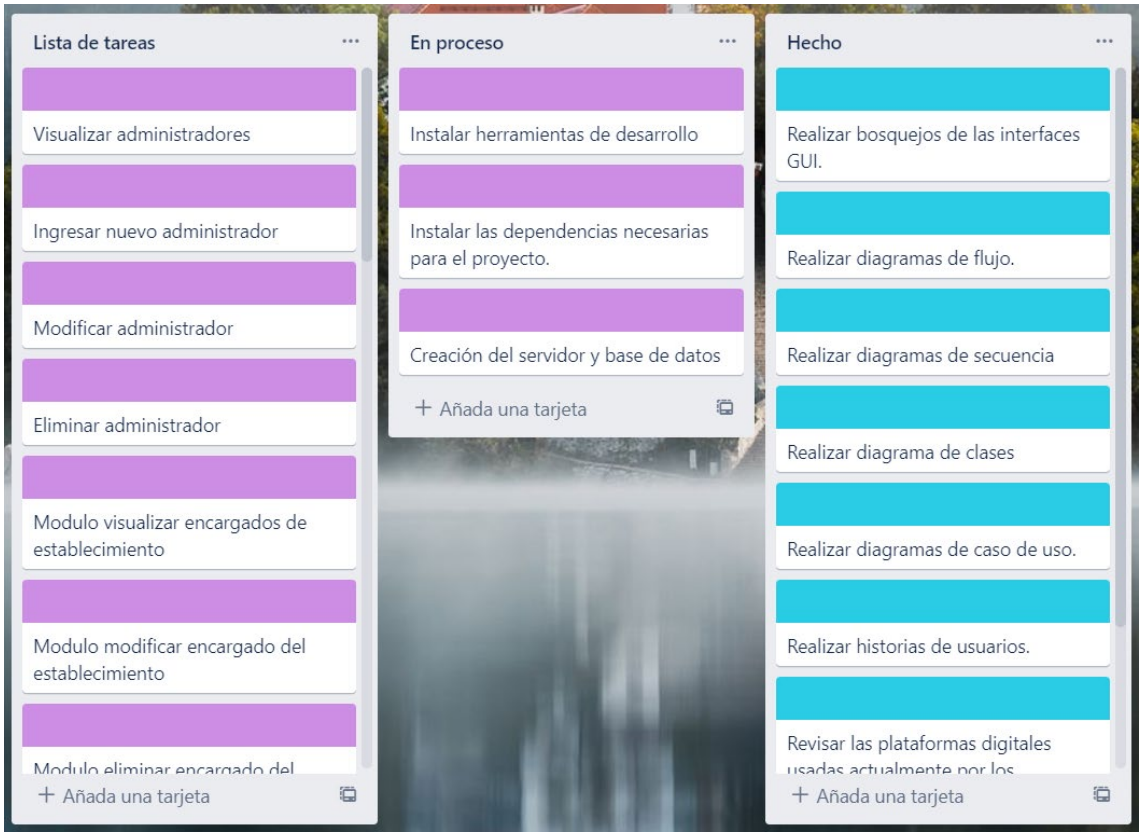

Figura 2.33. Actualización del tablero Kanban, implementación.

2.3.1. HERRAMIENTAS DE DESARROLLO

Visual Studio Code

La instalación de Visual Studio Code se la realiza desde la página oficial (Figura 2.34), en la cual se obtiene el instalador para el sistema operativo correspondiente.

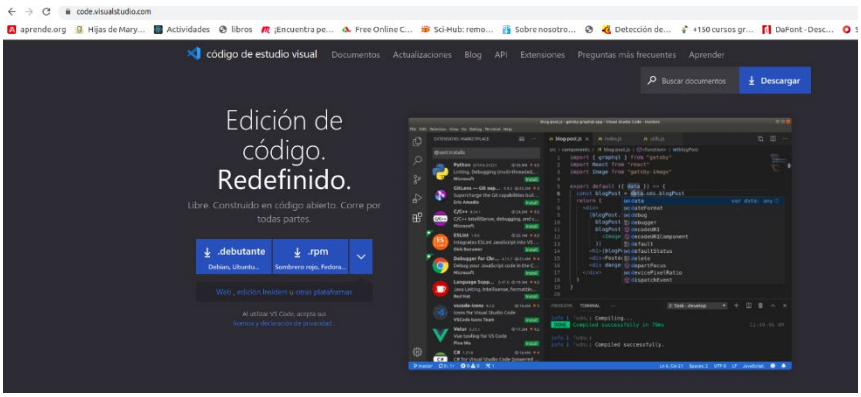

Figura 2.34. Página oficial de Visual Studio Code.

GitHub

Para mantener un control de versionamiento y respaldo del código fuente del prototipo de plataforma se levantaron dos repositorios privados dentro de la plataforma Github, los cuales se enlazaron al proyecto de *backend* y de *fronted*.

Node.js

Para instalar *Node.js* se usó el comando *sudo apt install nodejs* en un terminal (Figura 2.35). Esta instalación incluye el gestor de paquetes de *Node.js,* NPM (*Node Package Manager)* que aloja una gran cantidad de paquetes privados y públicos.

```
sandy@sandy-Inspiron-5570:~$ sudo apt install nodejs
[sudo] contraseña para sandy:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
nodejs ya está en su versión más reciente (16.13.0-deb-1nodesource1).
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  gyp javascript-common libc-ares2 libfwupdplugin1 libjs-inherits
  libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libpython2-stdlib
```

Figura 2.35. Instalación de Node.

Una vez instaladas las herramientas básicas, se estableció el ambiente local de desarrollo para el *fronted* (Figura 2.36) y *backend* (Figura 2.37).

Para la interfaz de usuario se creó una aplicación *Reactjs* para dar una estructura básica al *fronted*. En cuanto al *backend*, se creó una carpeta llamada *src*. y un archivo *env*. En la carpeta *src* se definieron tres archivos: *index.js*, *app.js* y *database.js* que van a ser usados en la implementación del servidor y de la base de datos.

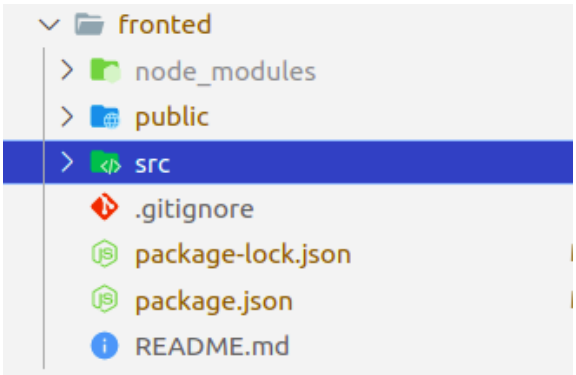

Figura 2.36. Carpeta Fronted.

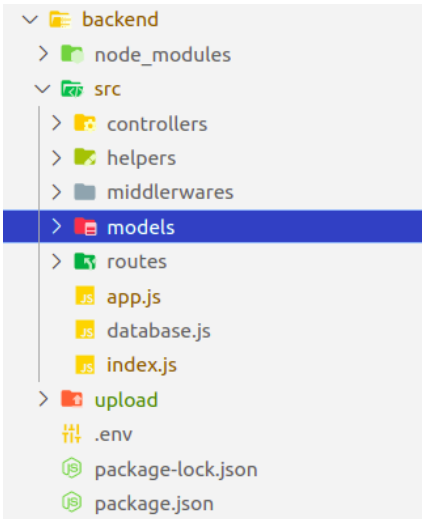

Figura 2.37. Carpeta Backend.

Dentro de la carpeta *backend* se ejecuta en un terminal el comando *node npm init –y* para crear el archivo *package.json* (Figura 2.38), este archivo va a tener información sobre el proyecto como: el nombre, la versión y los módulos. Los módulos instalados fueron: *Express, mongoose, cors*[5]*, dotenv*[6] *y Express-fileupload*[7]. Según se avance con desarrollo del sistema web, se van a ir instalando más módulos.

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "dev": "nodemon src/index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "2.8.5",
    "dotenv": "12.0.3",
    "express": "4.17.2",
    "express-fileupload": "1.3.1"
  },
```

Figura 2.38. Archivo package.json.

MongoDB

Para la creación de la base de datos, se descarga *MongoDB*, para esto se utiliza el comando *sudo apt install -y MongoDB* (Figura 2.39). Además, se instaló el visor de base de datos *Robo 3T*.

```
sandy@sandy-Inspiron-5570:~$ sudo apt install -y mongodb
[sudo] contraseña para sandy:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
mongodb ya está en su versión más reciente (1:3.6.9+really3.6.8+90~g8e540c0b6d
0ubuntu5.3).
Los paquetes indicados a continuación se instalaron de forma automática y ya n
 son necesarios.
  gyp javascript-common libc-ares2 libfwupdplugin1 libjs-inherits
  libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer libpython2-stdlib
  libpython2.7-minimal libpython2.7-stdlib libuv1-dev node-abbrev node-ajv
  node-ansi node-ansi-align node-ansi-regex node-ansi-styles node-ansistyles
  node-aproba node-archy node-are-we-there-yet node-asap node-asn1
```

Figura 2.39. Instalación de *MongoDB*.

[5] Es un mecanismo que permite que un servidor indique cualquier origen (dominio, esquema o puerto) que no sea el suyo propio desde el cual un navegador debería permitir la carga de recursos [18].
[6] Módulo de dependencia que permite crear variables de entorno [19].
[7] Middleware de express para cargar archivos [20].

2.3.2. SERVIDOR

Para la creación del servidor se usará el archivo *app.js* que va a contener la configuración de los middlewares[8]. Inicialmente se importan las librerías necesarias como se observa en las 3 primeras líneas de la Figura 2.40. Posteriormente, en la línea 4 se crea una aplicación *Express* con el objeto *app* que posee métodos para el enrutamiento de las peticiones HTTP y configuración del middleware.

En la línea 6 se establece el puerto, se define la variable *port* y el valor asignado va a ser la variable de entorno que se llama mediante *process.env.port* (Figura 2.41). En caso de no existir esta variable de entorno, se va a ocupar el puerto 4000.

De la línea 7 a la 9 se llama a la función *app.use* para cargar y especificar las funciones de middleware. Finalmente, se exporta el archivo en la línea 15 para posteriormente importarlo desde el archivo *index.js*.

```
1  const express = require("express");
2  const cors = require("cors");
3  const fileUpload = require("express-fileupload");
4  const app = express();
5
6  app.set("port", process.env.PORT || 4000);
7  app.use(cors());
8  app.use(express.json());
9  app.use(
10   fileUpload({
11     useTempFiles: true,
12     tempFileDir: "./upload",
13   })
14 );
15 module.exports = app;
```

Figura 2.40. Archivo app.js.

```
1  MONGODB_URI=mongodb://localhost/database
2  SECRET_JW_SEED=Esto-Es-uN-$3cr3t0
3  PORT=4000
```

Figura 2.41. Archivo env.

A continuación, en el archivo *index.js* (Figura 3.10) se arranca el servidor mediante la función *app.listen* la cual se va a ejecutar mediante el método *main*.

[8] Software que se sitúa entre el sistema operativo y las aplicaciones que se ejecutan en él. Conecta dos aplicaciones para que se puedan pasar fácilmente datos [21].

```
1  const app = require("./app");
2  require("dotenv").config();
3  require("./database");
4  async function main() {
5    await app.listen(app.get("port"));
6    console.log("server on port", app.get("port"));
7  }
8  main();
9
```

Figura 2.42. Archivo index.js.

2.3.3. BASE DE DATOS

En esta sección se creará la base de datos en el ambiente de desarrollo local. Esta creación la realiza el servidor en función de los parámetros definidos a través de la librería *mongoose*, la cual nos permite conectar, gestionar y controlar la base de datos mediante el archivo *database.js*.

En la línea 3 de la Figura 2.43 se define una constante que representa el URI (*Uniform Resource Identifier*) que va a contener el string de conexión a la base de datos. En la línea 5 se ejecuta el método *connect* desde *Mongoose* el cual permite conectar a una instancia de *MongoDB.* En la línea 9 se crea la conexión a la base de datos y finalmente en la línea 10 se verifica el estado de la conexión.

```
1  const mongoose=require('mongoose');
2
3  const URI = process.env.MONGODB_URI ? process.env.MONGODB_URI:
4   'mongodb://localhost/databaseCrud';
5  mongoose.connect(URI,{
6      useNewUrlParser:true,
7  });
8  const connection=mongoose.connection;
9  connection.once('open',()=>{
10     console.log('DB is connected');
11 });
```

Figura 2.43. Archivo database.js.

En la Figura 2.44 se muestra la conexión establecida entre el servidor y la base de datos.

```
sandy@sandy-Inspiron-5570:~/Documentos/CRUDS COMPLETOS/backend$ npm run dev

> backend@1.0.0 dev
> nodemon src/index.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
server on port 4000
DB is connected
```

Figura 2.44. Ejecución de servidor y conexión de la base de datos.

2.4. EJEMPLO DE IMPLEMENTACION PARA LA GESTION DE CATEGORIAS

En la Figura 2.45 se muestra la actualización del tablero Kanban con las actividades finalizadas.

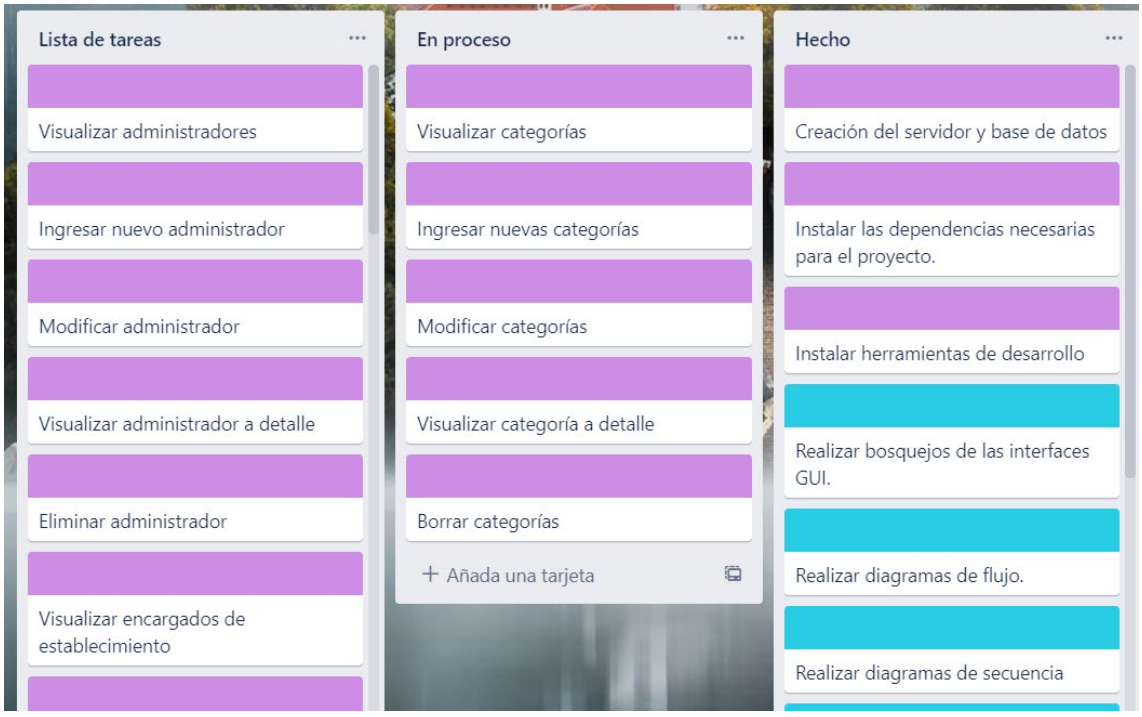

Figura 2.45. Actualización del tablero Kanban, implementación de gestión categorías.

A continuación, se muestra un ejemplo de implementación correspondiente a la gestión de categorías (Figura 2.46) dentro del módulo del administrador. Este proceso puede ser extrapolado a aquellos procesos de gestión de otros elementos, como lo son la implementación de: gestión de actividades, gestión de horarios, gestión de encargados, gestión de administradores, gestión de platos y gestión de reservas.

Figura 2.46. Pantalla con la gestión de categorías.

Dentro del servidor se definirán los modelos, controladores y las rutas necesarias para establecer comunicación entre cliente-servidor.

2.4.1. MODELO DE CATEGORÍA

Primero se define la estructura del documento que se va a guardar en la base de datos, para esto se crea un archivo *categoria.js*, del cual se exportará el modelo para poder consumirlo desde el archivo controlador.

En la Figura 2.47, en la línea 1 se importa desde la librería *Mongoose* un objeto *Schema* que define los tipos de datos que se van a usar y el objeto *Model* que permite a *Mongoose* crear y leer documentos de la base de datos. Desde la línea 4 a la línea a 6 se va a definir los atributos y el tipo de valor a registrarse.

```
1  const {Schema,model}=require('mongoose');
2  const categoriaSchema=new Schema
3  ({
4      nombreCategoria: String,
5      descripcionCategoria: String,
6      imagen:String,
7  });
8  module.exports=model('categoria',categoriaSchema);
```

Figura 2.47. Modelo categoría.

2.4.2. CONTROLADORES

Gestión de categoría

A continuación, se muestran los controladores creados para: visualizar lista de categorías, crear categoría, editar categoría, borrar categoría y visualizar categoría a detalle.

Visualizar lista de categorías

Para obtener la lista de categorías, se crea la función *getCategorias* que usa el método propio de *Mongoose find* mostrado en la Figura 2.48 en la línea 15, el cual permite obtener todos los documentos de la colección Categoría.

```
14  categoriaCtrl.getCategorias = async (req, res) => {
15    const categorias = await CategoriaModel.find();
16
17    res.json(categorias);
18  };
```

Figura 2.48. Visualizar lista de categorías.

Crear categoría

Para crear una categoría, se crea la función *createCategorias* que usa el método propio de *Mongoose* mostrado en la Figura 2.49 en la línea 40, el cual permite insertar el documento en la colección Categoría, este documento es generado previamente con el modelo *CategoriaModel*.

```
19  categoriaCtrl.createCategorias = async (req, res) => {
20    const { nombreCategoria, descripcionCategoria } = req.body;
21    var file = "";
22    var url = "";
23
24    if (req.files.imagen != undefined || req.files.imagen != null) {
25      file = req.files.imagen;
26      const valor = await cloudinary.uploader.upload(
27        file.tempFilePath,
28        (err, result1) => {
29          return result1;
30        }
31      );
32      url = String(valor.url);
33    }
34
35    const newCategoriaModel = new CategoriaModel({
36      nombreCategoria: nombreCategoria,
37      descripcionCategoria: descripcionCategoria,
38      imagen: url,
39    });
40    await newCategoriaModel.save();
41    console.log(
42      newCategoriaModel.nombreCategoria,
43      newCategoriaModel.descripcionCategoria
44    );
```

Figura 2.49. Crear categoría.

Para el almacenamiento de imágenes dentro de la colección Categoría se hace uso de *Cloudinary*[9] el cual permite almacenar imágenes en la nube y proporciona una url única por imagen. Como se puede observar en la Figura 2.50, inicialmente se establece los parámetros de conexión. Luego, para cargar la imagen en la nube se hace uso del

[9] Plataforma que permite a los usuarios administrar imágenes y videos para sitios web y aplicaciones [24].

método *upload* propio de *Cloudinary,* el cual retorna la url de la imagen que será almacenada dentro del atributo imagen.

```
9  cloudinary.config({
10   cloud_name: "dgypr0grx",
11   api_key: "752778558282185",
12   api_secret: "I5jxirxlrGRVhKdTDizgbZAvRaI",
13 });
```

Figura 2.50. Configuración de Cloudinary.

Editar categoría

Para editar una categoría se crea la función *updateCategoria* que usa el método propio de *Mongoose findbyidandupdate* mostrado en la Figura 2.51 en la línea 80, el cual permite modificar el documento en función del id dentro de la colección Categoría.

```
63 categoriaCtrl.updateCategoria = async (req, res) => {
64   const { nombreCategoria, descripcionCategoria } = req.body;
65
66   var file = "";
67
68   if (req.files) {
69     if (req.files.imagen != undefined || req.files.imagen != null) {
70       file2 = req.files.imagen;
71
72       const valor = await cloudinary.uploader.upload(
73         file2.tempFilePath,
74         (err, result) => {
75           return result;
76         }
77       );
78       url = String(valor.url);
79     }
80     await CategoriaModel.findByIdAndUpdate(req.params.id, {
81       nombreCategoria,
82       descripcionCategoria,
83       imagen: url,
84     });
85     res.json("actualizado");
86   } else {
87     await CategoriaModel.findByIdAndUpdate(req.params.id, {
88       nombreCategoria,
89       descripcionCategoria,
90     });
91     res.json("actualizado");
92   }
93 };
```

Figura 2.51. Editar categoría.

Eliminar categoría

Para eliminar una categoría se crea la función *delateCategoria* que usa el método propio de *Mongoose findbyidanddelete* mostrado en la Figura 2.52 en la línea 55, el cual permite eliminar el documento en función del id dentro de la colección Categoría.

```
53  categoriaCtrl.delateCategoria = async (req, res) => {
54
55    await CategoriaModel.findByIdAndDelete(req.params.id);
56    res.json({ message: "Categoria eliminada" });
57  };
```

Figura 2.52. Eliminar categoría.

Visualizar categoría a detalle

Para visualizar una categoría se crea la función *getCategoria* que usa el método propio de *Mongoose findbyId* mostrado en la Figura 2.53 en la línea 51, el cual permite obtener el documento en función del id dentro de la colección Categoría.

```
50  categoriaCtrl.getCategoria = async (req, res) => {
51    const newCategoriaModel = await CategoriaModel.findById(req.params.id);
52    res.json(newCategoriaModel);
53  };
```

Figura 2.53. Visualizar categoría a detalle.

2.4.3. RUTAS

Para definir las rutas cada módulo cuenta con su propio archivo, el cual será importado dentro de *index.js* donde se configurarán rutas específicas. A continuación, se va a detallar el archivo de ruta para el módulo categoría.

En la Figura 2.54 se importan las funciones definidas en el controlador y se relaciona cada función a su correspondiente método *http*; por ejemplo, *post* para crear una nueva categoría, *put* para actualizarla, *delete* para eliminar y *get* para obtenerla. Además, cada método estará asociado a la ruta principal ('/'), en caso de requerir un parámetro de entrada a través de la url la ruta se modificará mediante *'/:nombre_parametro'*.

```
1  const {Router}=require('express');
2  const router =Router();
3
4  const {getCategorias,createCategorias,getCategoria,delateCategoria,updateCategoria}
5  = require('../controllers/categoria.controller')      You, now • Uncommitted changes
6  router.route('/')
7  .get(getCategorias)
8  .post(createCategorias)
9  router.route('/:id')
10 .get(getCategoria)
11 .delete(delateCategoria)
12 .put(updateCategoria)
13 module.exports=router;
```

Figura 2.54. Rutas de categoría.

Para definir una ruta especifica dependiendo del módulo, se configura la ruta y el archivo al que va a hacer referencia, como se puede observar en la Figura 2.55 en la línea 22 se establece la ruta *'/api/categorias'* asociada al archivo creado para categorías '/routes/categorias'

```
22  app.use("/api/categorias", require("./routes/categorias"));
23  app.use("/api/encargados", require("./routes/encargados"));
24  app.use("/api/actividades", require("./routes/actividades"));
```

Figura 2.55. Archivo index.js.

2.4.4. FRONTEND DE GESTIÓN DE CATEGORÍAS

En esta sección se describe la implementación de la aplicación web que interactuará con el usuario final con la tecnología *React JS* y el IDE Visual Studio Code.

Dentro de la aplicación web se definen rutas mediante la librería *Reactrouterdom*. El segmento de Figura 2.56 muestra un ejemplo de las rutas configuradas para la aplicación web. Se configuraron rutas estáticas (Figura 2.57) y rutas dinámicas (Figura 2.58), estas últimas son necesarias para enviar información entre formularios.

```
<Route
  exact
  path="/categorias"
  element={
    <div>
      <nav className="navbar navbar-dark bg-dark">
        <NavBar></NavBar>
      </nav>
      <br></br>

      <Container fluid>
        <h1 className="text-center">
          <strong>Lista de categorías</strong>
        </h1>
        <Container>
          <Categoria></Categoria>
        </Container>

        <br></br>
        <br></br>

        <ListCategoria></ListCategoria>
      </Container>
    </div>
  }
/>
```

Figura 2.56. Ruta categoría.

```
<Routes>
  <Route path="/login"
  element=
  {  <Login>

    </Login>
  }
  />
```

Figura 2.57. Ejemplo ruta estática.

```
<Route
  path="/cambiarcontrasena/:id"
  element={
    <div>
      <CambiarContrasena></CambiarContrasena>
    </div>
  }
/>
```

Figura 2.58. Ejemplo ruta dinámica.

Gestión de categorías

La gestión de categorías permite al administrador agregar, editar, eliminar, visualizar la información detallada de una categoría y la lista de ellas. A continuación, se detallarán cada una de las acciones que el administrador puede ejecutar.

Ver lista de categorías

Para extraer la lista de categorías se envía una petición desde el *fronted* hacia el servidor mediante la función *getData* en la cual se utiliza la librería *axios* que ejecuta la petición.

En la Figura 2.59 en la línea 24 se observa la ruta del servidor, desde la cual se obtendrá la información de categorías. En la línea 26 se muestra el uso del método get y en la línea 27 se almacena el resultado de esta petición.

```
24  const URL = "http://localhost:4000/api/categorias";
25  const getData = async () => {
26    const response = await axios.get(URL);
27
28    return response;
29  };
```

Figura 2.59. Ver lista de Categorías.

Para aprovechar la funcionalidad de actualización y renderización de *Reactjs* en función de los cambios generados durante la interacción sobre la interfaz de usuario, se hace uso de la variable de estado *useState* para actualizar la lista de categorías obtenidas desde el servidor.

En la Figura 2.60, en la línea 65 se observa que se almacena la lista de categorías dentro de la variable de estado *list*, esto genera que *React* dispare una nueva renderización y actualización de la interfaz de usuario en la cual genera una vista de la lista de categorías.

```
62  useEffect(() => {
63    getData().then((response) => {
64      const listaDeCategorias = response.data;
65      setList(listaDeCategorias);
66    });
67  }, []);
```

Figura 2.60. Almacenamiento de lista categorías.

Reactjs permite mostrar listas de datos de manera sencilla a través del método *map* el cual permite generar un array de elementos de tipo *html* con la información de cada objeto de la lista de categorías como se indica en la Figura 2.61, en la línea 119, lo cual facilita diseñar la interfaz gráfica de cada elemento.

```
116                 <tbody>
117                   {list.map((elemento, pos) => (
118                     <tr key={pos}>
119                       <td>{elemento.nombreCategoria}</td>
120                       <td className="d-flex justify-content-center .justify-content-lg-start gap-1">
121                         <button
122                           className="btn btn-success"
123                           onClick={() => seleccionarCategoria(elemento, "Editar")}
124                         >
```

Figura 2.61. List map.

Crear categoría

Para crear una categoría se hace uso de una plantilla de formulario desde la librería *Formik*, mediante la cual se puede validar el tipo de datos que ingresa el usuario, en caso de ser incorrectos se generara una alerta. Para ello primero, mediante la función *validationSchema* (Figura 2.62), se verifica el ingreso de los atributos solicitados y en caso de ser una imagen se verifican los formatos definidos en la línea 11.

```
11  const SUPPORTED_FORMATS = [
12    "image/jpg",
13    "image/jpeg",
14    "image/gif",
15    "image/png",
16  ];
17  const validationSchema = Yup.object().shape({
18    nombreCategoria: Yup.string()
19      .min(2, "Nombre mínimo 2 carácteres")
20      .max(1000, "*Nombre no puede tener mas de 1000 carácteres")
21      .required("*Nombre es requerido"),
22    descripcionCategoria: Yup.string()
23      .min(10, "Descripción mínima de 10 carácteres")
24      .max(1000, "**Nombre no puede tener mas de 1000 carácteres")
25      .required("*Descripción es requerida"),
26    imagen: Yup.mixed().required("Esta imagen es requerida")
27    .test(
28      "FILE_FORMAT",
29      "Formato incorrecto.",
30      (value) => !value || (value && SUPPORTED_FORMATS.includes(value.type))
31    ),
32  });
```

Figura 2.62. Función ValidationSchema.

En la Figura 2.63 se observa como llegan los datos del formulario a través del método *onSubmit,* y con *formdata* se genera el cuerpo del objeto que se va a enviar en el método *post* definido en la línea 54. Una vez ingresada la nueva categoría, en la interfaz gráfica se redirige hacia la lista de categorías en donde se visualizará el nuevo elemento.

```
48  const onSubmit = async (values) => {
49    var formdata = new FormData();
50    formdata.append("nombreCategoria", values.nombreCategoria);
51    formdata.append("descripcionCategoria", values.descripcionCategoria);
52    formdata.append("imagen", values.imagen);
53    const URL = "http://localhost:4000/api/categorias/";
54    await axios.post(URL, formdata);
55    setmodalAgregar(false);
56    window.location.href = "/categorias";
57  };
```

Figura 2.63. Función onSubmit crear.

Visualizar categoría

Para visualizar una categoría se crea una función obtener, la cual tiene como parámetro de entrada el id. En la línea 38 de la Figura 2.64 se observa una petición *get* hacia el servidor con la ruta y el id; la respuesta es la categoría correspondiente.

```
37  const obtener = async (id) => {
38    const res = await axios.get(`http://localhost:4000/api/categorias/${id}`);
39    setImagen11(res.data.imagen);
40  };      You, 6 months ago • hasta aqui1 …
```

Figura 2.64. Obtener categoría.

Editar categoría

Inicialmente, se muestra en el formulario la información de la categoría obtenida a través del método *get,* la cual se podrá editar. En la Figura 2.65 se observa como llegan los datos del formulario y el id a través del método *onSubmit. Además,* con *formdata* se genera el cuerpo del objeto que se va a enviar en el método *put* definido en la línea 102. Una vez actualizada la categoría, en la interfaz gráfica se redirige hacia la lista de categorías en donde se visualizará la actualización.

```
89  const onSubmit = async (id, values) => {
90    if (id == "62b13e5f86a98429f943c8d5") {      You, 5 seconds ago • Uncommitted
91      setmodalEditar(false);
92      setmodalAdvertencia(true);
93    } else {
94      var formdata = new FormData();
95      formdata.append("nombreCategoria", values.nombreCategoria);
96      formdata.append("descripcionCategoria", values.descripcionCategoria);
97      if (values.imagen != null || values.imagen != undefined) {
98        formdata.append("imagen", values.imagen);
99      } else {
100       formdata.append("imagen", imagen11);
101     }
102     await axios.put(`http://localhost:4000/api/categorias/${id}`, formdata);
103     setmodalEditar(false);
104     window.location.href = "/categorias";
105   }
106 };
```

Figura 2.65. Función onSubmit editar categoría.

Eliminar categoría

Para eliminar una categoría se crea una función eliminar, la cual tiene como parámetro de entrada el id. En la línea 82 de la Figura 2.66 se observa una petición delete hacia el servidor con la ruta y el id. En la interfaz gráfica se redirige hacia la lista de categorías en donde se visualizará la actualización.

```
80  const eliminar = async (id) => {
81
82      await axios.delete(`http://localhost:4000/api/categorias/${id}`);
83      window.location.href = "/categorias";
84          You, 1 second ago • Uncommitted changes
85  }
```

Figura 2.66. Función eliminar categoría.

En la Figura 2.67 se muestra la actualizacion del tablero Kanban.

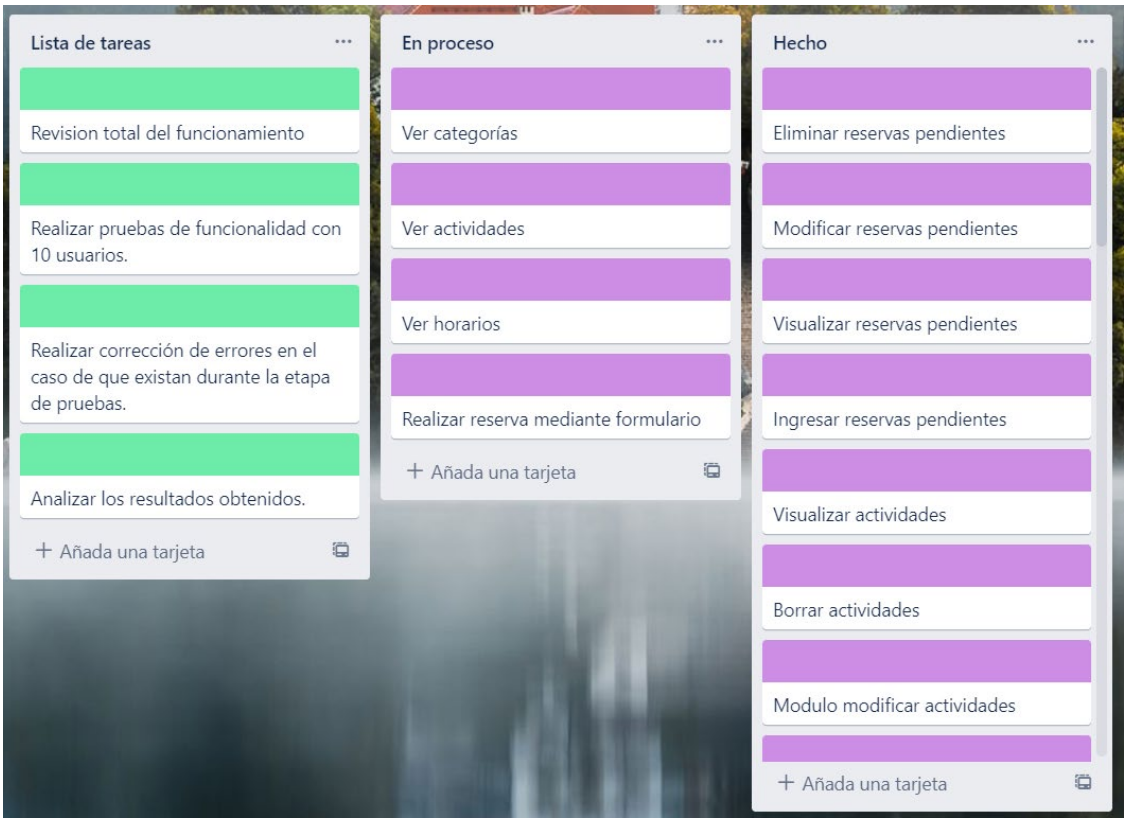

Figura 2.67. Actualización del tablero Kanban, gestión de módulo turista.

2.5. MÓDULO TURISTA

Este módulo permite al usuario turista generar una reserva a través del sistema web, funcionalidad fundamental del sistema construido. Para esto, primero va a tener que escoger una categoría, una actividad, un horario y finalmente va a tener que llenar un formulario con sus datos. A continuación, se muestra cada uno de estos pasos.

Ver categorías

Las categorías, creadas previamente por el administrador, se van a mostrar en un carrusel de imágenes generados mediante el código que se presenta en la Figura 2.68. Cuando el turista escoja una categoría se va a llamar a la función *verActividad* (Figura 2.69), que tiene como parámetro de entrada el *id* de la categoría escogida, esta función va a redireccionar a una nueva página con las actividades disponibles de la categoría seleccionada.

```
<Container className="containerCarrusel"
>
  <img className="imagenesCarrusel" src={elemento.imagen}></img>
  <button
    className="btn btn-success"
    type="button"
    onClick={() => verActividad(elemento._id)}
  >
    {elemento.nombreCategoria}
  </button>
</Container>
```

Figura 2.68. Contenedor del carrusel de imágenes.

```
85  const verActividad = async (id) => {
86    window.location.href = `/veractividad/${id}`;
87  };
```

Figura 2.69. Función verActividad.

Ver actividades

Al turista se le van a mostrar las actividades referentes a la categoría escogida, esto se logra con la función *getData* que envía una petición *get* desde el *fronted* hacia el servidor con la librería *axios,* para extraer la lista de actividades asociadas a la categoría. Esto se muestra en la Figura 2.70 línea 26.

```
25  const getData = async () => {
26    const response = await axios.get(
27      `http://localhost:4000/api/actividades/categorias/${categoriaId}`
28    );
29    return response;
30  };
```

Figura 2.70. Función getData

Al escoger una actividad se invoca a la función ver actividad perfil la cual recibe como parámetro de entrada el id de la actividad escogida por el turista (Figura 2.71)

```
74
75  const verActividadPerfil = async (id) => {
76    window.location.href = `/perfilActividad/${id};
77  };
```

Figura 2.71. Función verActividadPerfil.

Perfil de la actividad

Una vez se encuentre el turista en el perfil de la actividad, se evalúa con el id de la categoría si la actividad escogida pertenece a la categoría comida o no.

Si la actividad pertenece a la categoría comida, se va a activar el botón “menú de platos y reservar” como se muestra en la línea 248 de la Figura 2.72, caso contrario se mostrará el botón “reservar”, en la línea 260, que redireccionará a los horarios disponibles evaluando si existen o no con la función *VerificarHorarios* (Figura 2.73).

```
247  {bottonActivo ? (      You, 1 second ago • Uncommitted changes
248    <button
249      className="btn btn-success btn-lg btn-block"
250      type="button"
251      onClick={() => verplatos()}
252    >
253      Menú de platos y Reservar
254    </button>
255  ) : (
256    <>
257      <br></br>
258      <button
259        className="btn btn-success btn-secondary btn-lg btn-block d-flex justify-content-center"
260        onClick={() => VerificarHorarios()}
261      >
262        Reservar
263      </button>
264    </>
265  )}
```

Figura 2.72. Botones activos.

```
129  const VerificarHorarios = async () => {
130    getDataHorarios().then((responseHorarios) => {
131      const newListHorarios = responseHorarios.data;
132      if (newListHorarios.length > 0) {
133        var count1 = 0;
134        newListHorarios.forEach((element) => {
135          if (element.cantDisponible > 0) {
136            count1 = count1 + 1;
137          }
138        });
139        if (count1 == 0) {
140          setmodalHorarios(true);
141        } else {
142          window.location.href = `/verhorario/${perfilId}`;
143        }
144      } else {
145        setmodalHorarios(true);
146      }
147    });
148  };
```

Figura 2.73. Función VerificarHorarios.

Ver horarios

En la Figura 2.74 se muestra la función reservar que se va a activar cuando el turista escoja un horario y va a redireccionar a la página que contiene el formulario para realizar la reserva.

```
104  const reservar = async (id) => {
105    window.location.href = `/reservar/${id}`;
106  };
```

Figura 2.74. Función reservar.

Reserva

Finalmente, el turista llena el formulario y se guarda la reserva mediante el método *post* que se muestra en la Figura 2.75.

```
402  const URL = "http://localhost:4000/api/reservas";
403  await axios.post(URL, user);
```

Figura 2.75. Método post guardar reserva.

En caso de que la reserva se encuentre en estado pendiente, se va a llamar a la función *enviarWhatsapp* mostrada en la Figura 2.76, que abre el link de Whatsapp con la información de la reserva, para enviar un mensaje al encargado del establecimiento.

```
117  const enviarWhatsapp = (values) => {
118    window.open(
119      `https://api.whatsapp.com/send?phone=${number}&text=Hola soy ${values.nombreTuristaSolicitud}
120      ${values.apellidoTuristaSolicitud}, quiero hacer una reservación para
121       ${values.cantidadReservaSolicitud} personas, en la actividad
122       ${nombreActividad}, en el horario de las ${horarioInicio} horas del día
123       ${fechaInicio} hasta las ${horarioFin} horas del día ${fechaFin}.  `      You, now • Uncommitte
124    );
125  };
```

Figura 2.76. Función enviarWhatsapp.

3. RESULTADOS Y DISCUSIÓN

En la Figura 3.1 se muestra la actualización del tablero Kanban

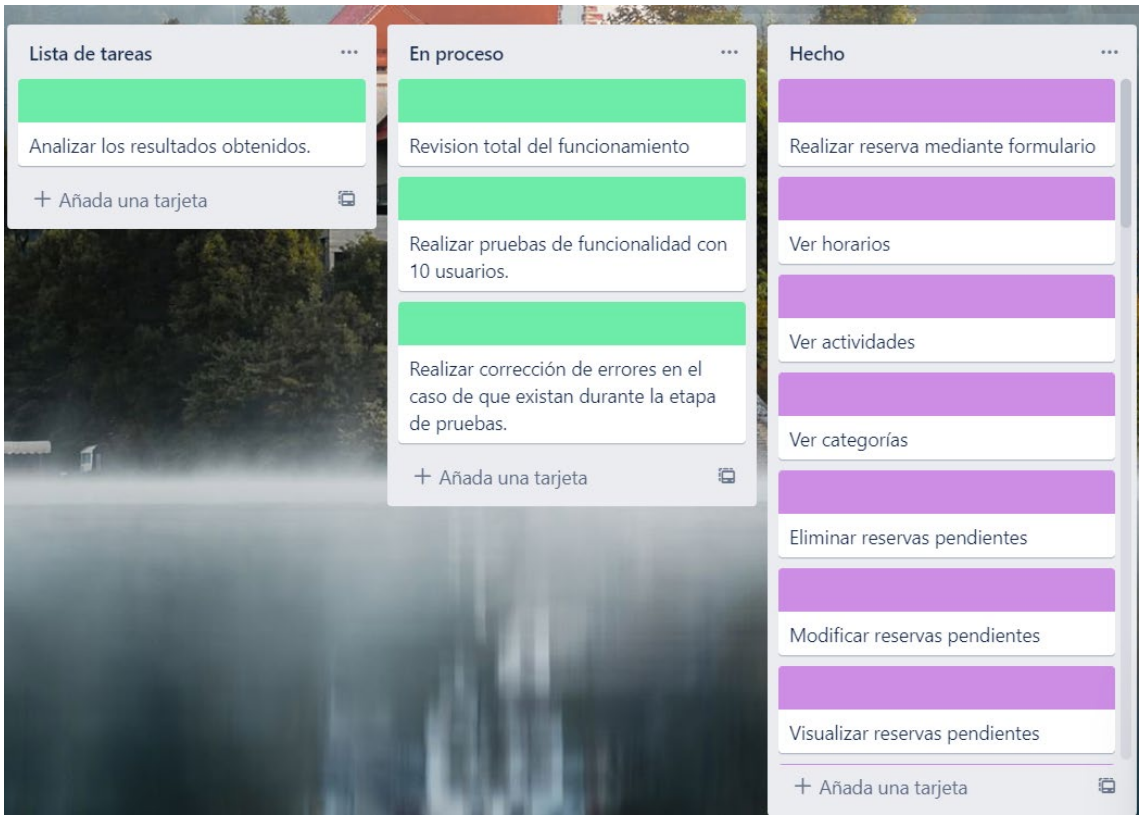

Figura 3.1. Actualización del tablero Kanban, pruebas de funcionalidad.

En esta sección se presentan las pruebas realizadas al sistema web. Inicialmente, se presenta las pruebas de funcionalidad de los servicios usados por los distintos módulos del sistema; por ejemplo, el servicio de correo. A continuación, se muestran las pruebas para verificar el cumplimiento de los requerimientos funcionales y no funcionales del sistema construido, definidos en el Capítulo 2. Después, se muestran los resultados de la encuesta de satisfacción realizada a 10 usuarios. Por último, se detallan los errores encontrados en las pruebas realizadas y su respectiva corrección.

3.1. FUNCIONALIDAD DE LOS SERVICIOS USADOS POR EL SISTEMA CONSTRUIDO

Las pruebas de esta sección se realizan con la herramienta Postman.

3.1.1. PRUEBA DE FUNCIONALIDAD DEL SERVICIO DE BASE DE DATOS

En esta prueba se verificó las funcionalidades de lectura, escritura, actualización y eliminación de elementos en la base de datos.

La Figura 3.2 muestra un ejemplo de la petición POST para el ingreso de un encargado a la base de datos, y en la Figura 3.3 muestra el ingreso exitoso de la categoría a la base de datos.

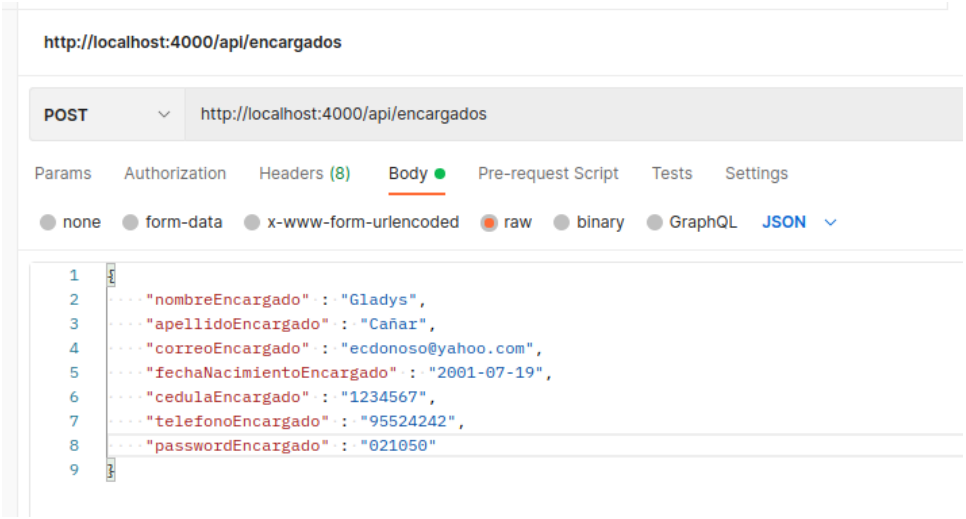

Figura 3.2. Prueba en Postman del método post.

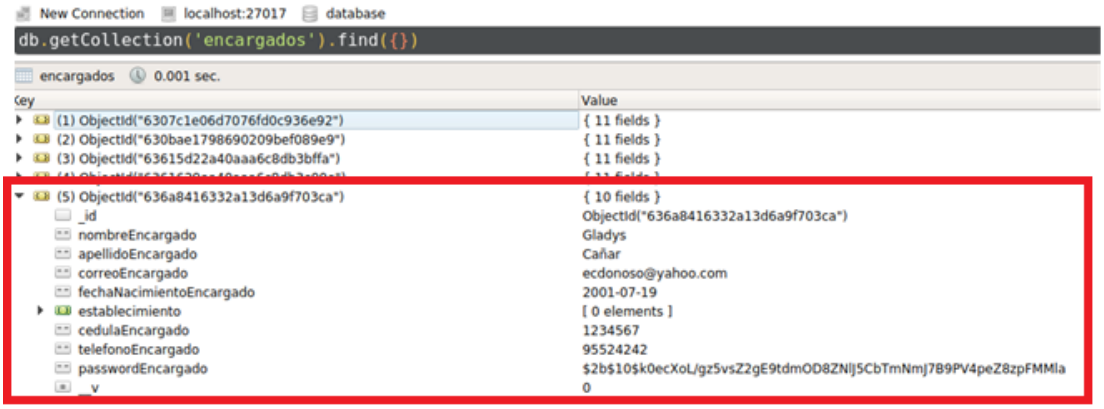

Figura 3.3. Categoría en la base de datos.

3.1.2. PRUEBA DE FUNCIONALIDAD DEL SERVICIO DE CORREO

En esta prueba se verificó el envío de correo cuando se crea un usuario o cuando se generan reservas. La Figura 3.4 muestra la petición POST utilizada para el envío de correos hacia el usuario.

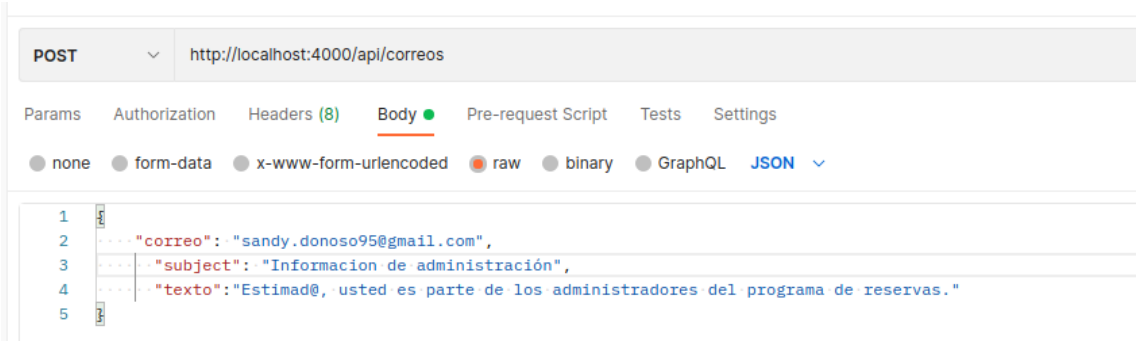

Figura 3.4. Prueba en Postman del envío de correos.

En la Figura 3.5 y Figura 3.6 se muestran los correos recibidos por el usuario en la creación de usuarios y la generación de reservas.

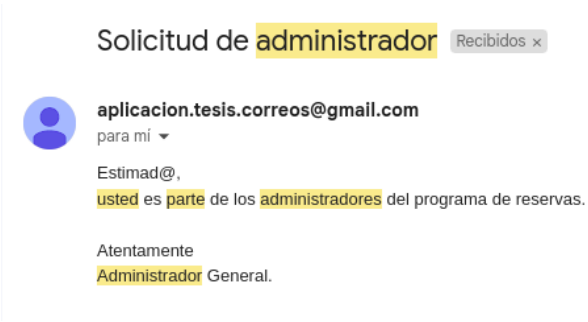

Figura 3.5. Ejemplo de correo cuando se crea un usuario.

Figura 3.6. Ejemplo de correo cuando se genera una reserva.

3.1.3. PRUEBA DE FUNCIONALIDAD DEL SERVICIO DE IMÁGENES

En esta prueba (Figura 3.7) se verificó el almacenamiento de imágenes en el servidor *Cloudinary* (Figura 3.8), el cual proporciona la *url* pública de acceso a la imagen que será almacenada como un atributo del documento dentro de la base de datos (Figura 3.9).

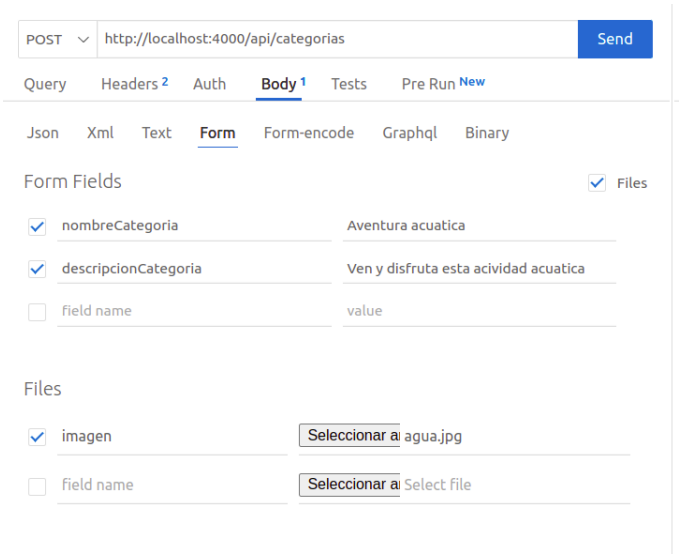

Figura 3.7. Prueba en Postman de la creación de una categoría con imagen incluida.

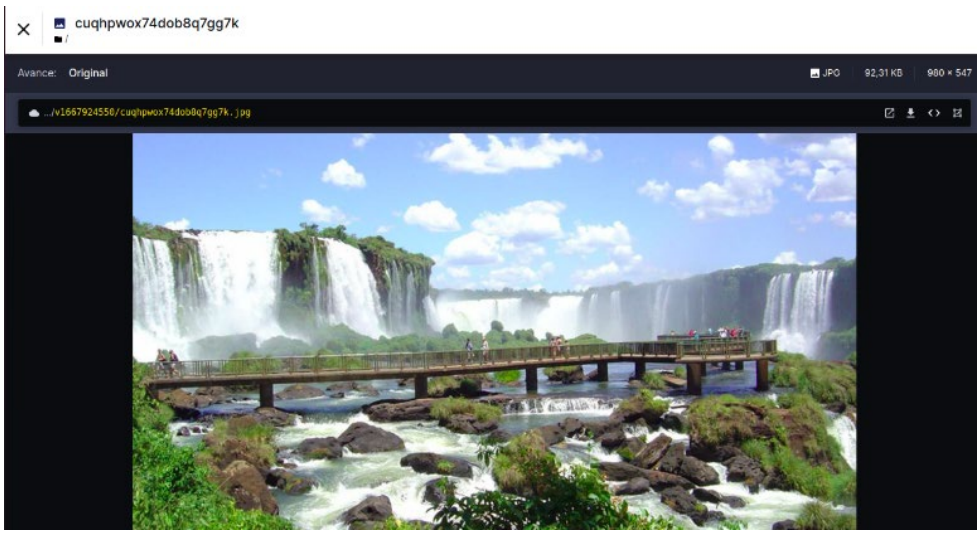

Figura 3.8. Imagen subida en Cloudinary.

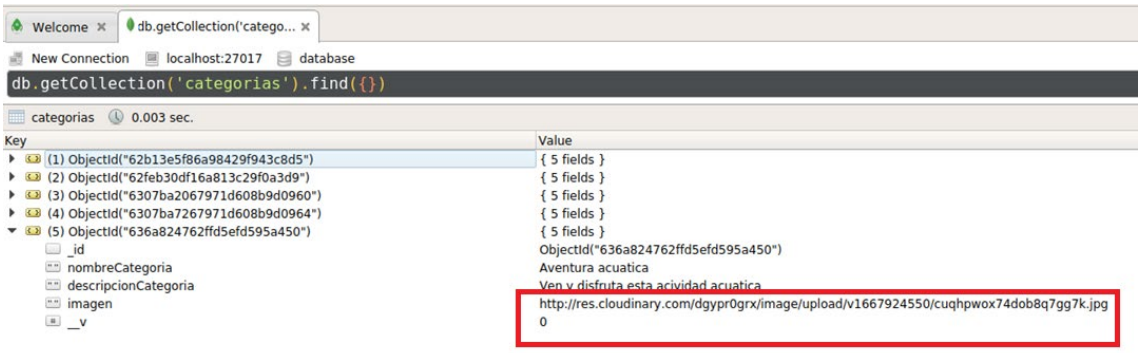

Figura 3.9. Url de la imagen almacenada en la base de datos.

3.2. PRUEBAS DE FUNCIONALIDAD DEL SISTEMA WEB

En esta sección se describen las pruebas de funcionalidad de los módulos de administración, encargado de establecimiento y turista.

Inicio de sesión administrador (R19) y encargado (R21)

Inicialmente, se realizó la prueba de autenticación para el administrador principal, administrador y encargado de establecimiento.

En la Figura 3.10 se muestra la interfaz gráfica mediante la cual los usuarios ingresaron su correo, contraseña y seleccionaron el perfil al que pertenecen, para proceder con la autenticación. Además, los usuarios pudieron cambiar la contraseña en la opción de recuperar contraseña (Figura 3.11).

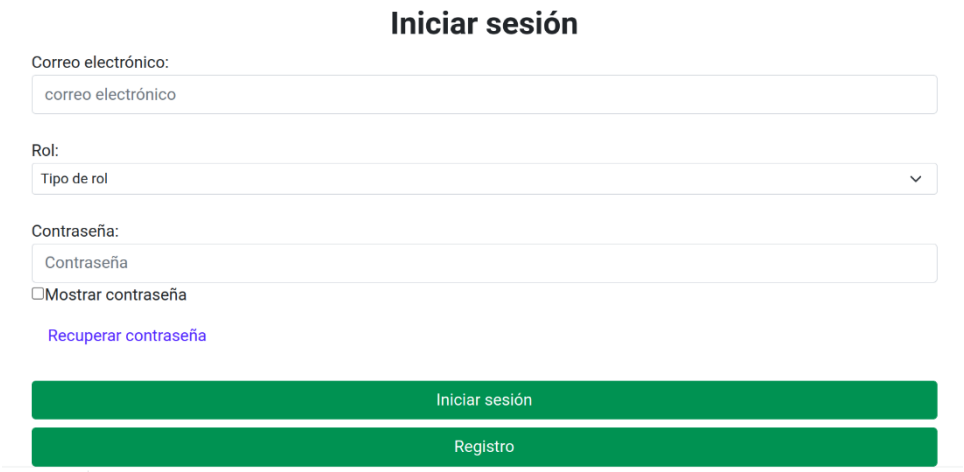

Figura 3.10. Inicio de sesión.

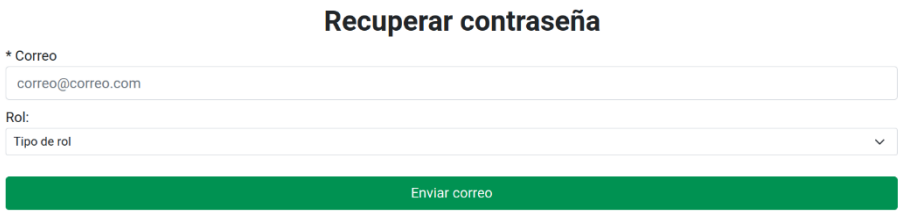

Figura 3.11. Recuperar contraseña.

En la Figura 3.12 se observa el mensaje mostrado a los usuarios en caso de generarse un error en la autenticación.

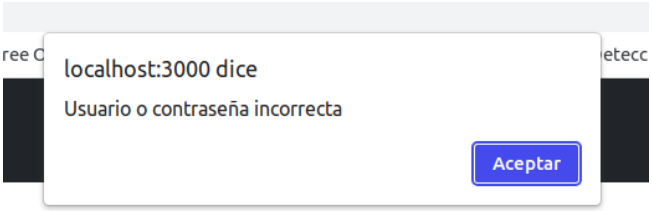

Figura 3.12. Mensaje de error.

3.2.1. MÓDULO DEL ADMINISTRADOR

En la Figura 3.13 se muestra el menú con las opciones de gestión que tuvo el administrador principal una vez que inició sesión.

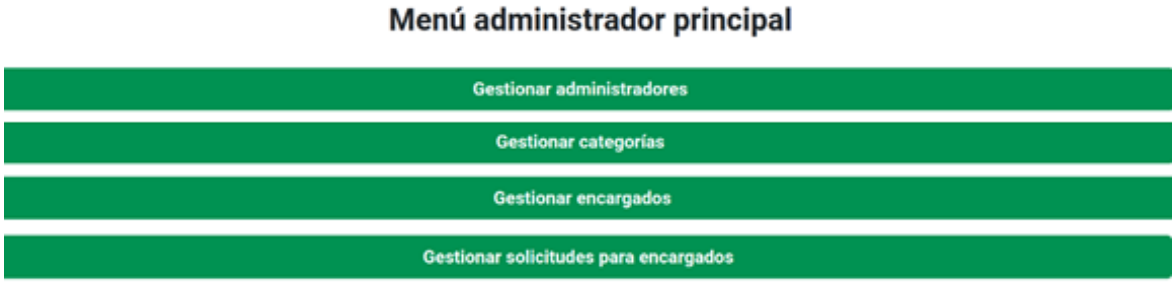

Figura 3.13. Menú del administrador principal.

3.2.1.1. Gestión de administradores

Durante esta prueba se verificó que el administrador principal pueda visualizar la lista de administradores, y crear, editar, borrar y ver a detalle la información de cada administrador.

Visualizar lista de administradores (R1)

La Figura 3.14 muestra la lista de administradores desde la cual se puede acceder a las diferentes acciones CRUD que se realizarán.

Figura 3.14. Lista de administradores.

Crear Administrador (R2)

El administrador principal accedió a la opción crear nuevo administrador donde se mostró el formulario donde se ingresa la información correspondiente al usuario.

En el formulario de la Figura 3.15 se muestran los mensajes de alerta en caso de que los campos obligatorios se encuentren vacíos. Estos mensajes de error se van a aplicar para el resto de los formularios del sistema web.

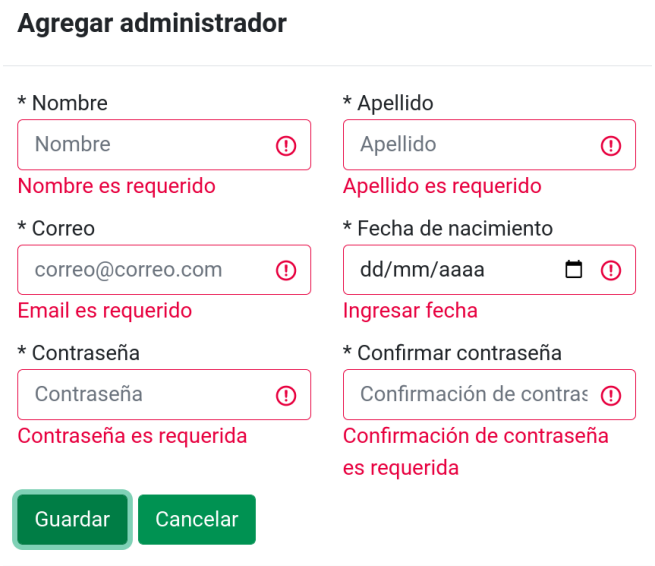

Figura 3.15. Formulario para agregar administrador.

Modificar administrador (R3)

La Figura 3.16 muestra el formulario al que accedió el administrador principal para editar un administrador.

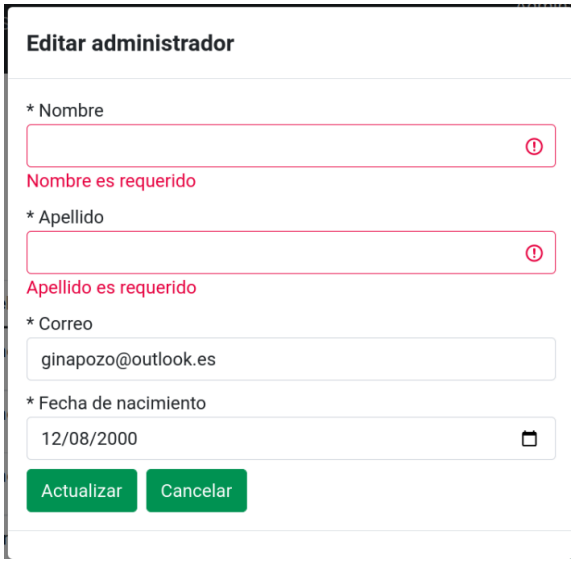

Figura 3.16. Formulario para editar administrador.

Borrar administrador (R4)

Para eliminar a un administrador el sistema web, se solicitó la confirmación al administrador principal mediante el modal mostrado en la Figura 3.17.

Figura 3.17. Modal para eliminar administrador.

Visualizar administrador a detalle (R5)

La opción visualizar a detalle mostró al administrador principal un formulario con información no editable correspondiente al administrador seleccionado como se puede observar en la Figura 3.18.

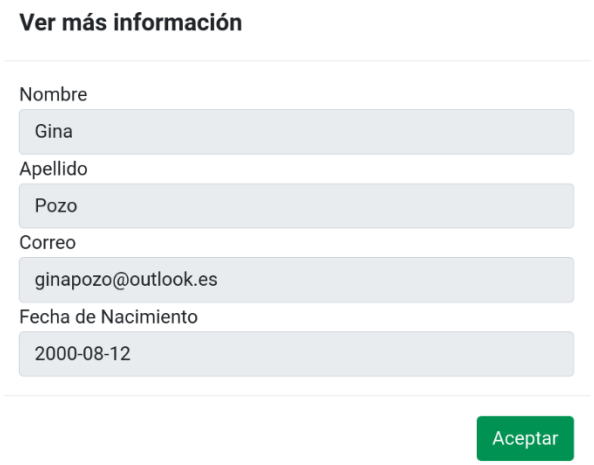

Figura 3.18. Formulario ver más información de administrador.

3.2.1.2. Gestión de categorías (Funciones CRUD[10]) (R6, R7, R8, R9, R10)

Similar al caso anterior, se verificó que los administradores puedan visualizar la lista de categorías (Figura 3.19), crear (Figura 3.20), borrar (Figura 3.21), editar (Figura 3.22), y ver a detalle la información (Figura 3.23) de cada categoría.

Figura 3.19. Lista de categorías.

[10] Crear, leer, actualizar, y eliminar.

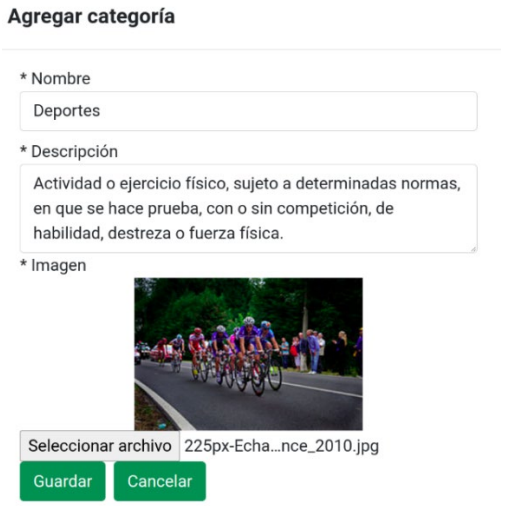

Figura 3.20. Formulario agregar categoría.

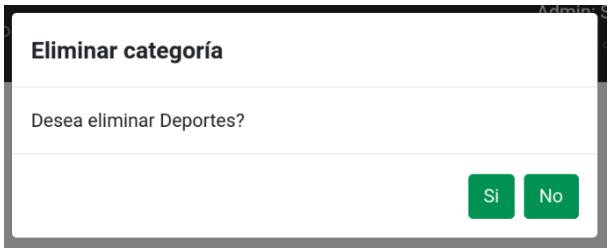

Figura 3.21. Aviso eliminar categoría.

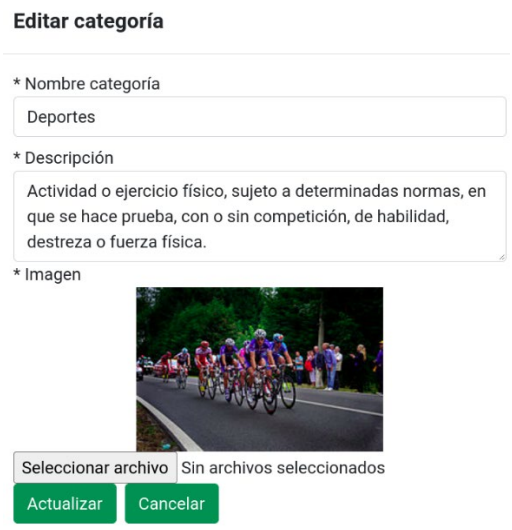

Figura 3.22. Formulario editar categoría.

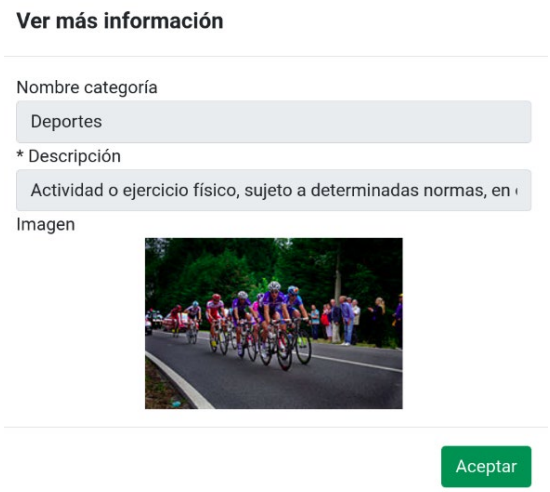

Figura 3.23. Ver más información de categoría.

3.2.1.3. Gestión de encargados (Funciones CRUD) (R11, R12, R13, R14, R15)

Los administradores en esta prueba pueden visualizar la lista de encargados (Figura 3.24), crear (Figura 3.25), borrar (Figura 3.26), editar (Figura 3.27), y ver a detalle la información de cada encargado (Figura 3.28).

Figura 3.24. Lista de encargados.

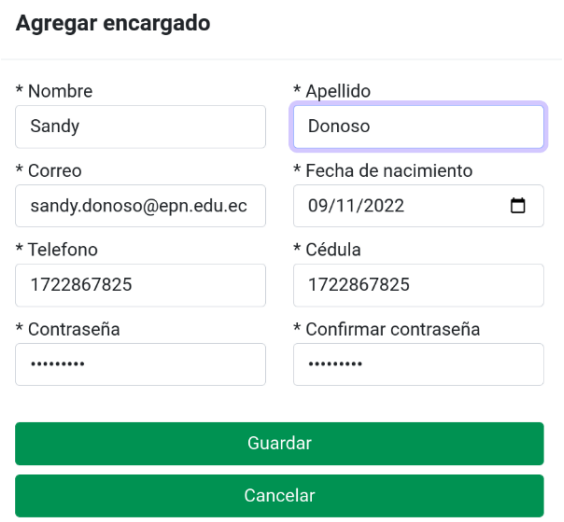

Figura 3.25. Formulario agregar encargado.

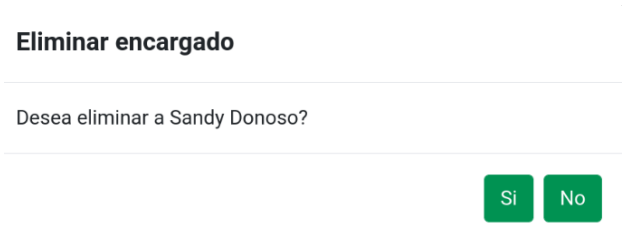

Figura 3.26. Aviso para eliminar encargado.

Figura 3.27. Formulario editar encargado.

Figura 3.28. Formulario ver más información del encargado.

3.2.1.4. Gestión de solicitudes para encargados

Registro de encargado (R20)

Además de la gestión de encargados a través de la interfaz de administración, un encargado puede auto-registrarse en el sistema. Para esto, desde la interfaz de inicio de sesión, un usuario no autenticado puede registrarse como encargado en el sistema web. Se usó el formulario mostrado en la Figura 3.29, desde el cual se envió la solicitud de registro.

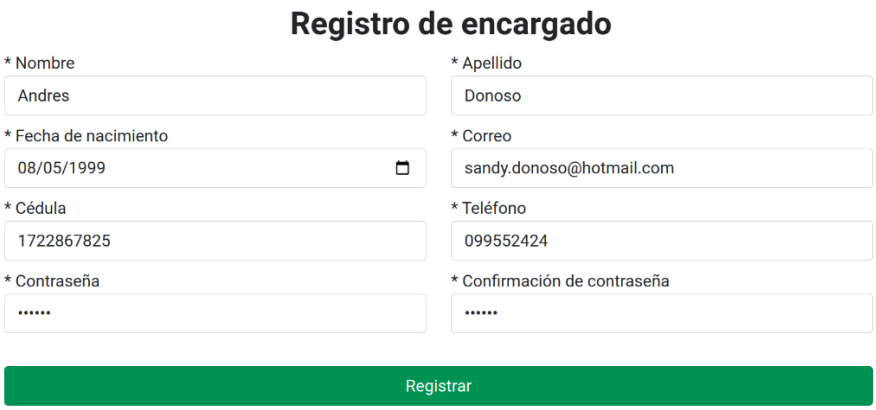

Figura 3.29. Formulario para registro de encargado.

Una vez que se envió la solicitud se despliega el mensaje mostrado en la Figura 3.30.

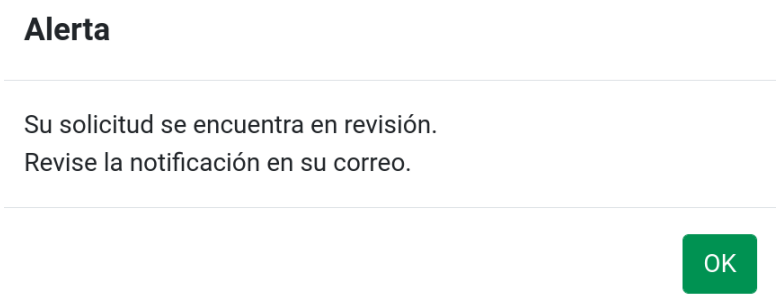

Figura 3.30. Mensaje de alerta del registro pendiente.

Visualizar lista de encargados de establecimientos pendientes (R16)

Para completar el proceso de registro de un nuevo encargado, los administradores tuvieron que revisar la solicitud, para ello se visualizó la lista de solicitudes pendientes como se muestra en la Figura 3.31.

Figura 3.31. Lista de solicitudes pendientes de encargados.

Aceptar nuevo encargado de establecimientos (R17)

En la Figura 3.32 se puede visualizar la información de la solicitud del encargado. Cuando el administrador acepta la solicitud del encargado se muestra al nuevo encargado en la lista de encargados como se puede ver en la Figura 3.33.

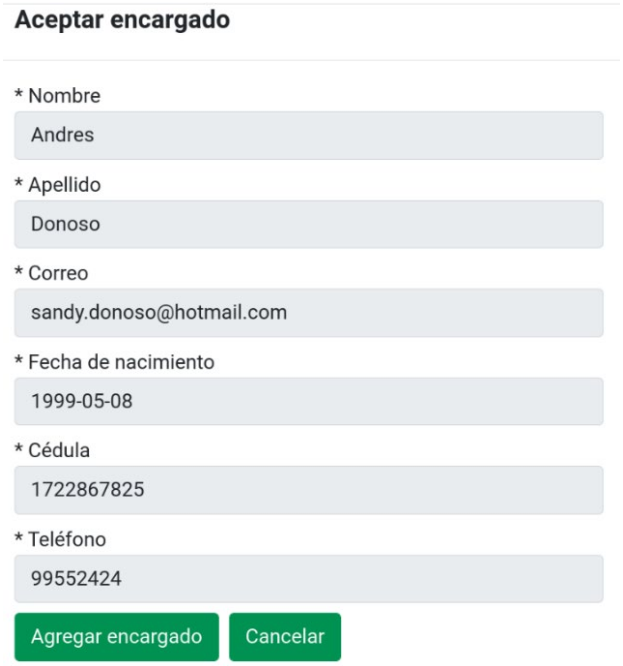

Figura 3.32. Formulario con información del posible encargado.

Figura 3.33. Lista de encargados actualizada.

Rechazar solicitud de encargado de establecimiento (R18)

En caso de que el administrador rechace la solicitud se abre un modal de confirmación (Figura 3.34).

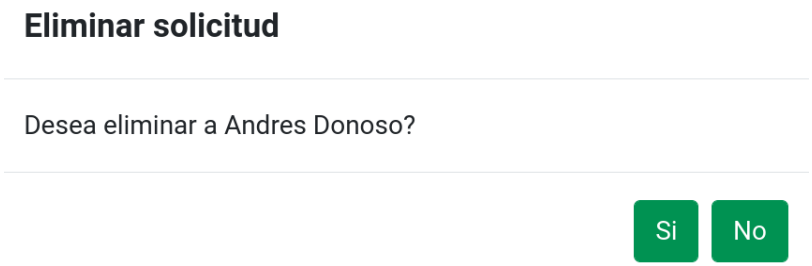

Figura 3.34. Aviso para eliminar la solicitud de encargado de establecimiento.

3.2.2. MÓDULO DEL ENCARGADO DE ESTABLECIMIENTO

El encargado tiene las opciones para gestión de establecimientos, actividades, horarios y reservas como se muestra en la Figura 3.35.

Figura 3.35. Menú del encargado.

3.2.2.1. Gestión de establecimientos (R22, R23, R24, R25, R26)

Los encargados pudieron visualizar la lista de establecimientos (Figura 3.36), crear (Figura 3.37), editar (Figura 3.38), borrar (Figura 3.39) y ver a detalle la información de establecimiento (Figura 3.40).

Figura 3.36. Lista de establecimientos.

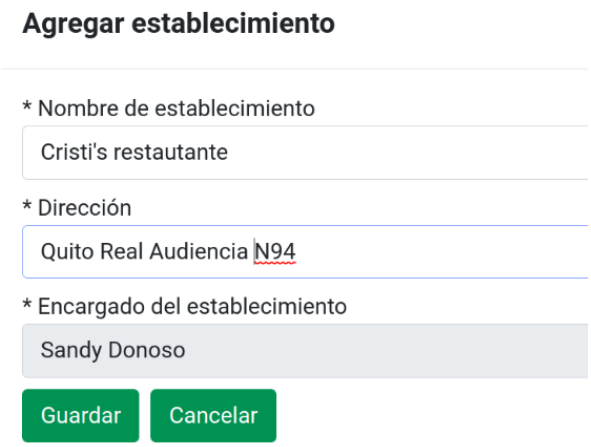

Figura 3.37. Formulario para agregar establecimiento.

Figura 3.38. Aviso para eliminar establecimiento.

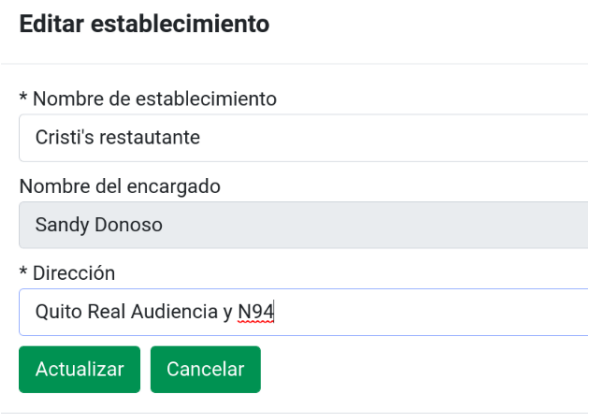

Figura 3.39. Formulario para editar establecimiento.

Figura 3.40. Ver más información del establecimiento.

3.2.2.2. Gestión de actividades

Visualizar lista de actividades (R27)

Los encargados durante esta prueba pudieron ver lista de actividades (Figura 3.41), crear, editar, eliminar y visualizar actividades correctamente.

Figura 3.41. Visualizar lista de actividades.

Crear actividad (R28)

Para la creación de una actividad, el encargado tuvo que seleccionar el establecimiento a la cual estará ligada la actividad como se observa en la Figura 3.42.

Figura 3.42. Lista de actividades por establecimiento.

A continuación, se muestra el formulario de actividad (Figura 3.43) en el cual se ingresó la información necesaria para la creación de actividad. Este formulario consta de la lista de categorías creadas previamente por el administrador, de las cuales el encargado pudo seleccionar una de ellas para asociarla con la actividad.

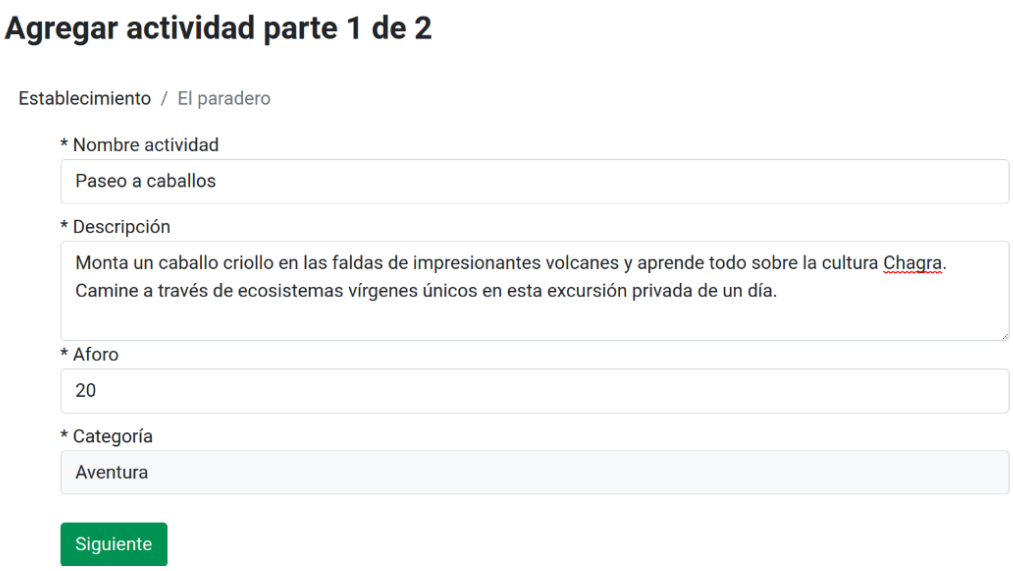

Figura 3.43. Ingreso de datos para la creación de la actividad.

Finalmente, en la Figura 3.44 se muestra la interfaz en la que el encargado pudo subir de 2 a 5 imágenes relacionadas con la actividad.

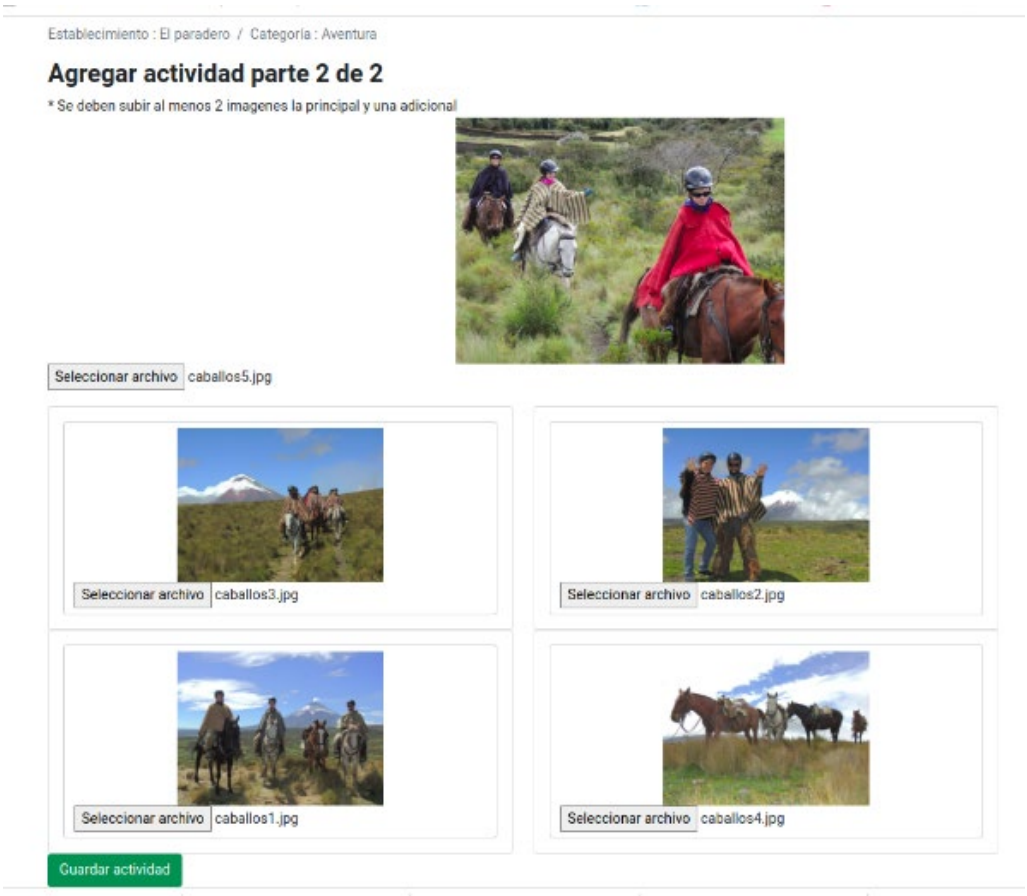

Figura 3.44. Ingreso de imágenes para la creación de la actividad.

Una vez que la actividad fue creada, se mostró un mensaje informando el ingreso exitoso de la actividad. En caso de que se trate de la categoría comida, el mensaje contendrá una pregunta, cuestionando si desea agregar platos en ese momento (Figura 3.45). Caso contrario, pregunta si desea agregar horarios (Figura 3.46).

Figura 3.45. Opción para agregar información de platos.

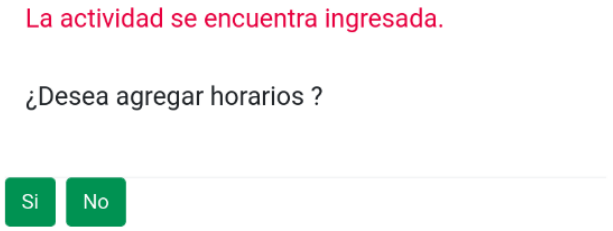

Figura 3.46. Opción para agregar información de horarios.

Modificar actividad (R29)

El encargado pudo acceder al formulario editar actividades como se muestra en la Figura 3.47. Desde este formulario se pudo acceder a la opción editar imágenes, la cual abre el modal mostrado en la Figura 3.48. Además, permitió acceder a la interfaz de gestión de horarios.

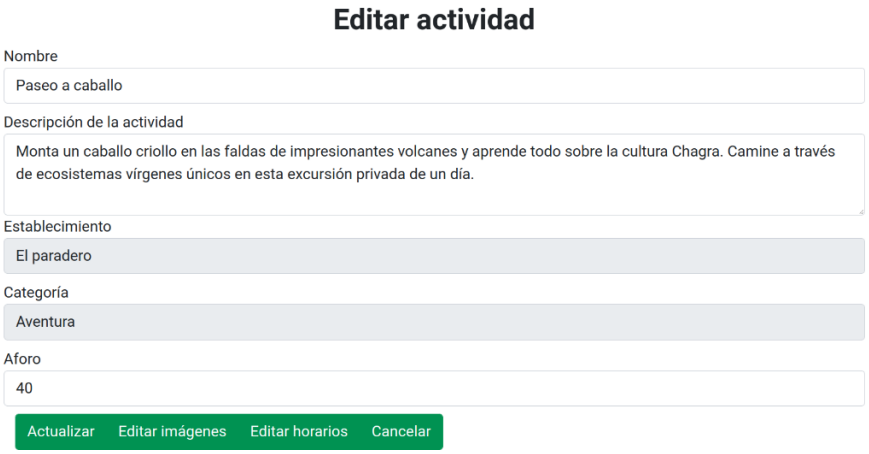

Figura 3.47. Formulario editar actividad.

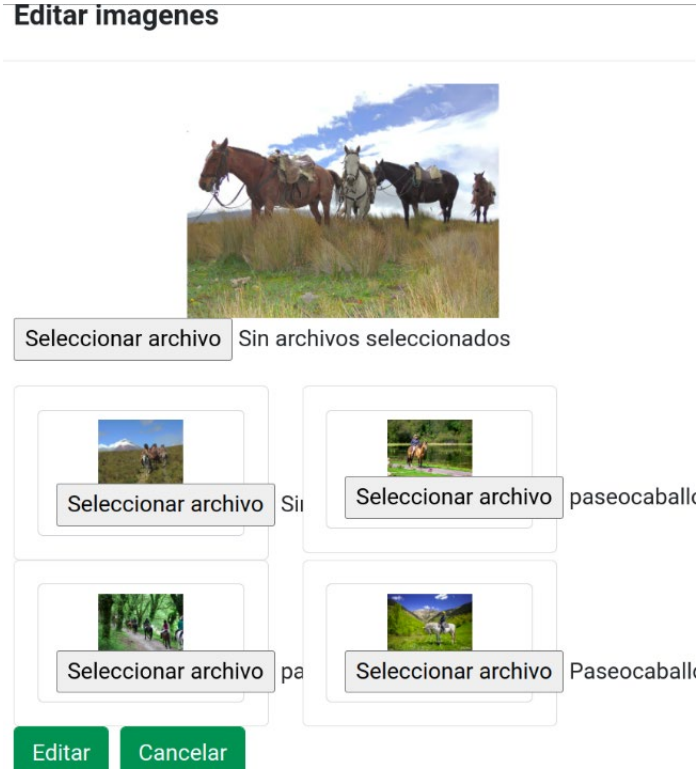

Figura 3.48. Formulario editar imágenes.

En caso de que la actividad pertenezca a la categoría comida, se mostró una opción llamada comida (Figura 3.49) que va a redireccionar a la interfaz de gestión de platos.

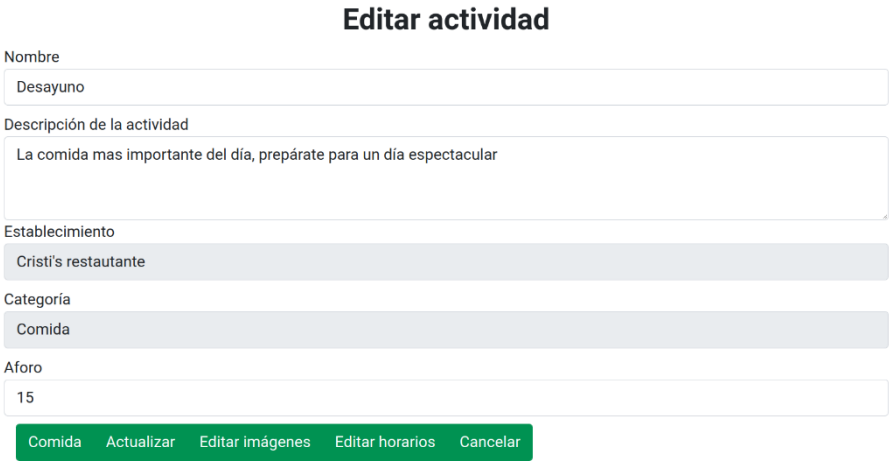

Figura 3.49. Formulario editar actividad de categoría comida.

Borrar actividad (R30)

La opción eliminar, mostró al encargado un mensaje que solicita la confirmación para eliminar la actividad, como se observa en la Figura 3.50.

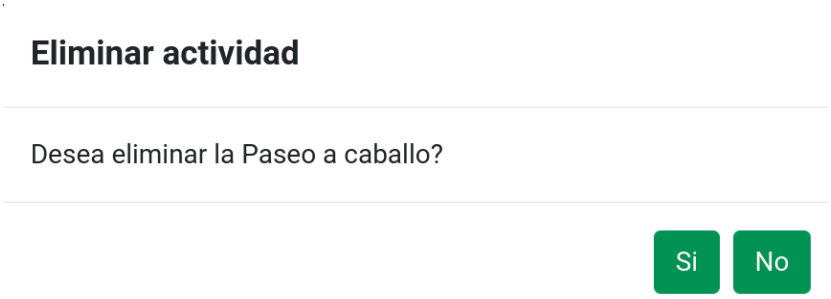

Figura 3.50. Alerta eliminar actividad.

Ver información a detalle de actividad (R31)

La opción visualizar a detalle, mostró al encargado un formulario con información correspondiente a la actividad como se puede observar en la Figura 3.51. Adicionalmente, para revisar las imágenes se accede al botón de imágenes que despliega un nuevo formulario con las imágenes disponibles (Figura 3.52).

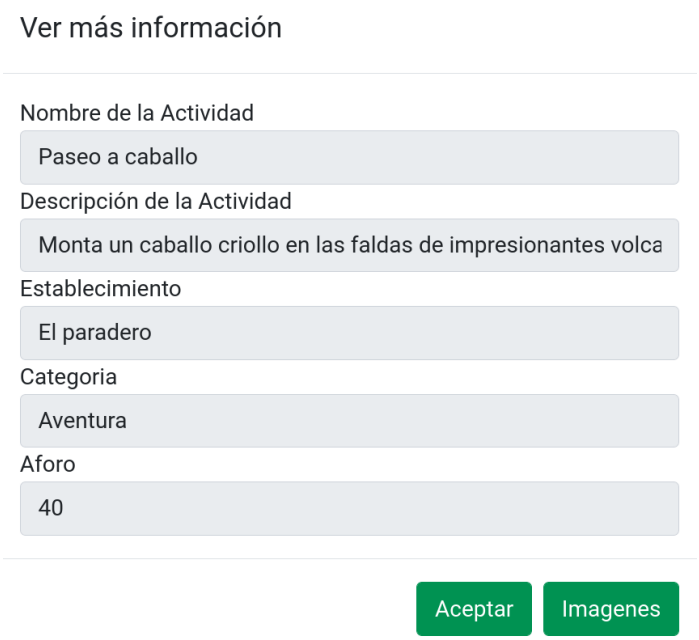

Figura 3.51. Formulario ver más información de actividad.

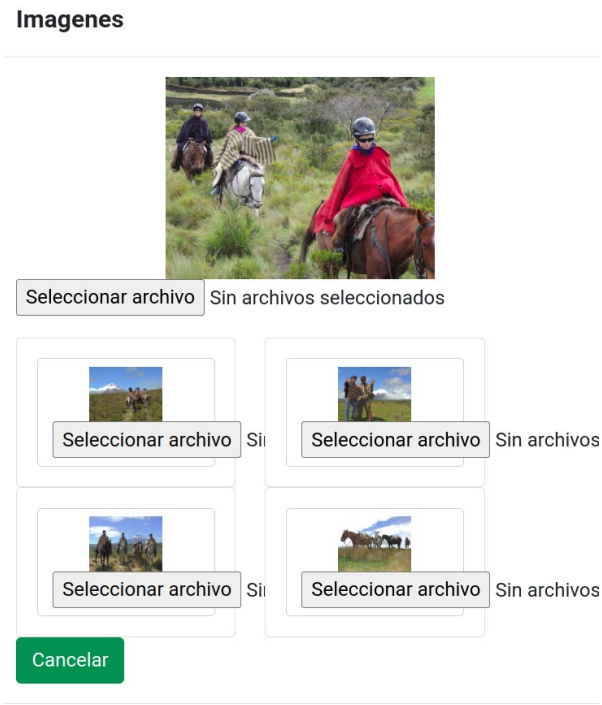

Figura 3.52. Formulario ver imágenes.

Gestión de horarios (R32, R33, R34, R35)

Los encargados pudieron visualizar la lista de horarios (Figura 3.53), crear (Figura 3.54), borrar (

Figura 3.55) y ver a detalle la información de horario (Figura 3.56). Para ver la lista de horarios creados primero se escoge un establecimiento y una actividad.

Figura 3.53. Lista de horarios.

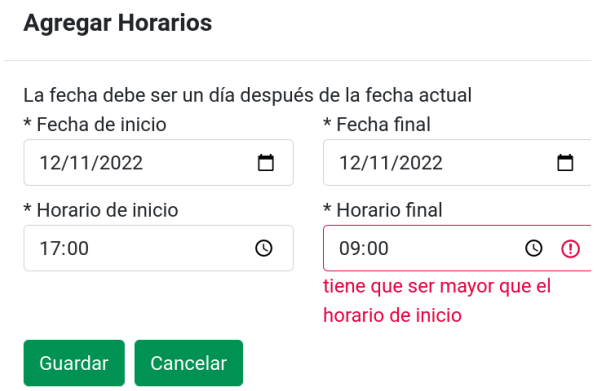

Figura 3.54. Formulario crear horario.

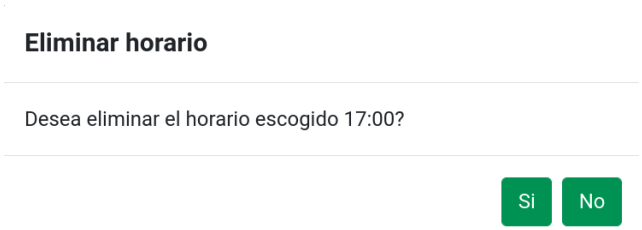

Figura 3.55. Alerta eliminar horario.

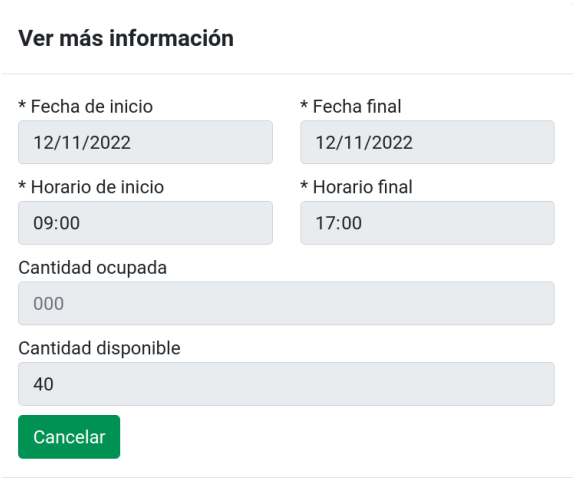

Figura 3.56. Formulario ver más información de horario.

Gestión de platos (R36, R37, R38, R39, R40)

Los encargados pudieron visualizar la lista de platos (Figura 3.57), y crear (Figura 3.58), borrar (Figura 3.59), editar (Figura 3.60) y ver a detalle la información de plato (Figura 3.61).

Figura 3.57. Lista de platos.

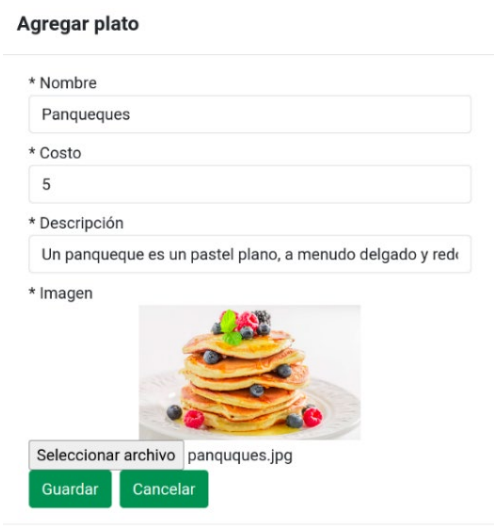

Figura 3.58. Formulario para agregar plato.

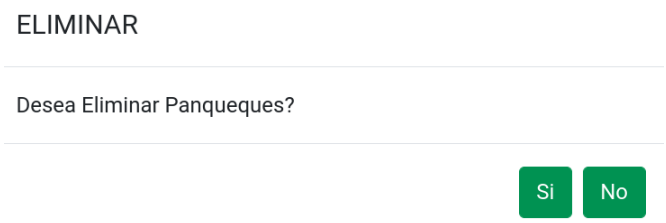

Figura 3.59. Alerta para eliminar plato.

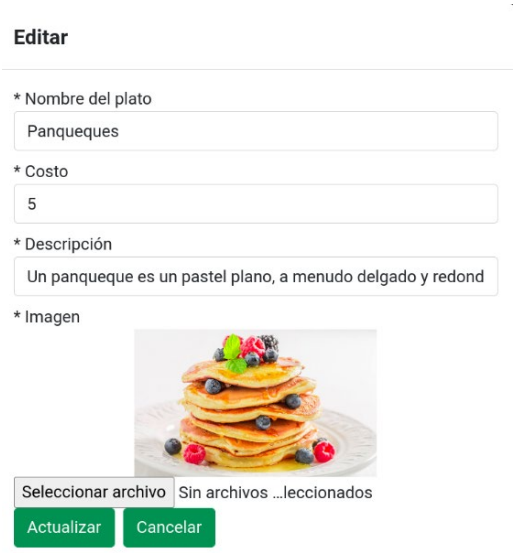

Figura 3.60. Formulario para editar plato.

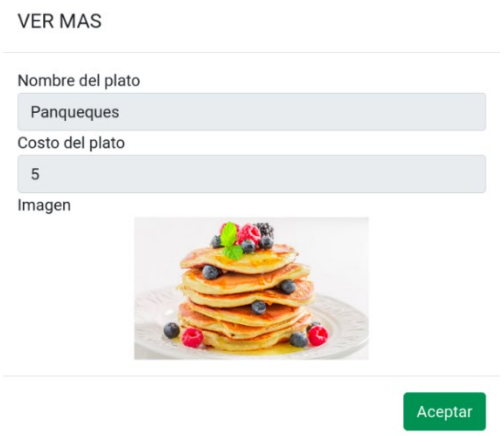

Figura 3.61. Ver más información de plato.

Gestionar reservas confirmadas (R41, R42, R43, R44) y reservas pendientes (R45, R46, R47, R48, R49)

Se comprobó que cuando un usuario turista genera una reserva, el encargado del establecimiento pudo gestionar las reservas confirmadas (Figura 3.62) y pendientes (Figura 3.63).

Figura 3.62. Lista de reservas pendientes.

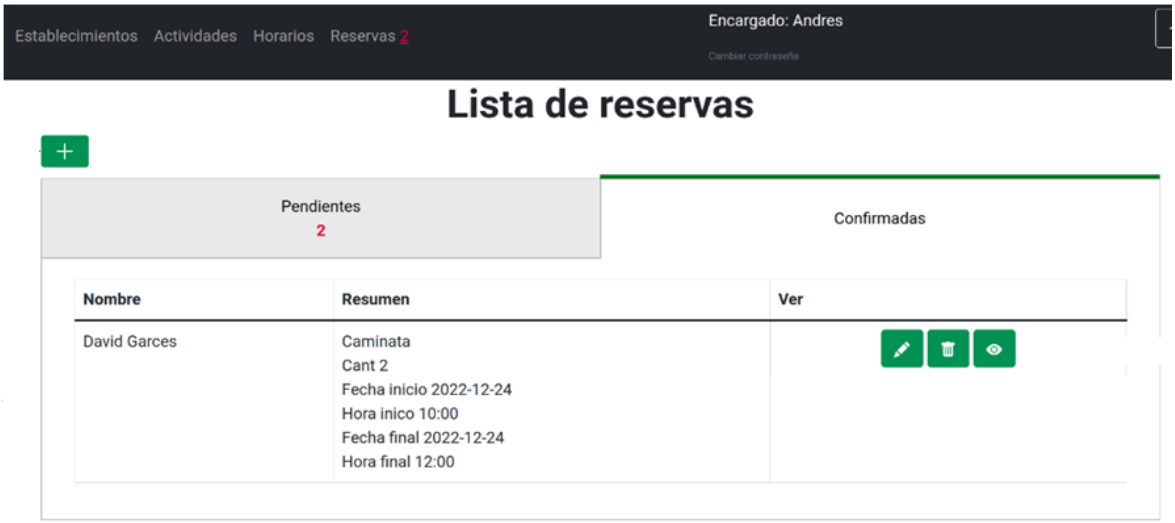

Figura 3.63. Lista de reservas confirmadas.

3.2.3. MÓDULO TURISTA

Visualizar categorías y actividades (R50, R51)

El usuario turista pudo seleccionar una categoría de las registradas previamente por el administrador (Figura 3.64) y se mostraron las actividades disponibles de dicha categoría (Figura 3.65).

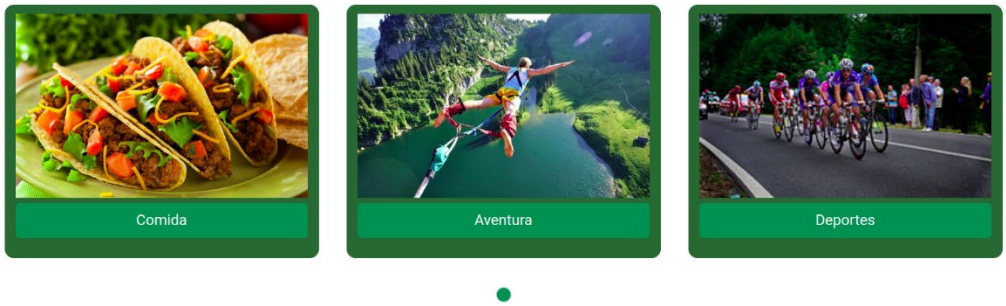

Figura 3.64. Página con las categorías disponibles.

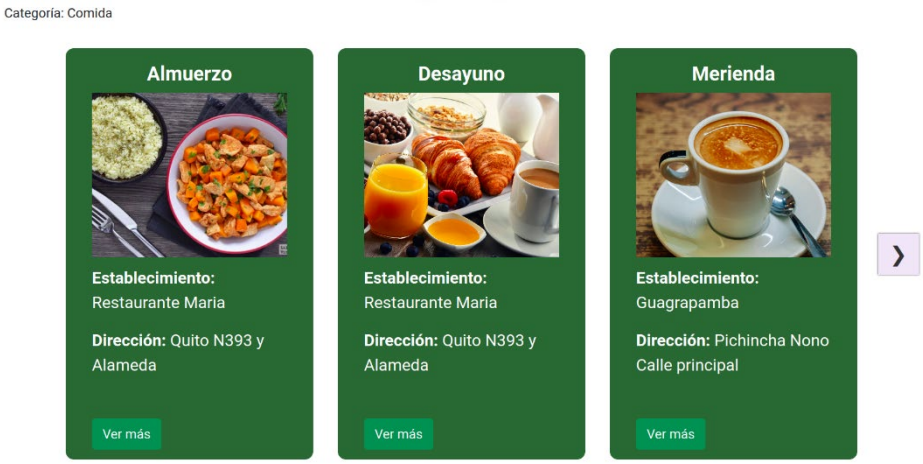

Figura 3.65. Página con las actividades disponibles.

Visualizar perfil de cada actividad (R52)

A continuación, el usuario escogió una de las actividades registradas para ver información de la misma. Al escoger una actividad de categoría comida se mostró un botón para acceder al menú de platos (Figura 3.66), donde se mostró un formulario para seleccionar platos para la reserva (Figura 3.67). El botón de reserva de dicho formulario redireccionó a la interfaz donde se muestran los horarios disponibles de la actividad (Figura 3.69).

Figura 3.66. Página web del perfil de la actividad seleccionada, tipo de categoría comida.

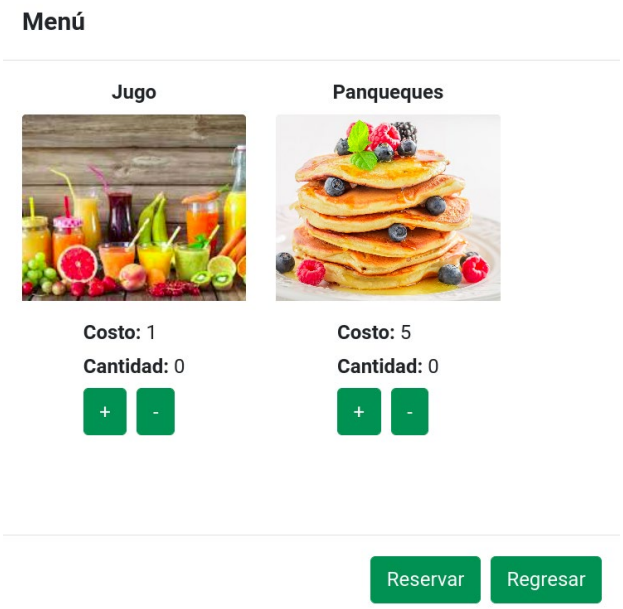

Figura 3.67. Formulario con platos disponibles para reserva.

En caso de que la categoría escogida sea diferente a comida, el perfil de la actividad seleccionada muestra un botón reservar (Figura 3.68), que redirecciona directamente a la página de horarios disponibles (Figura 3.69).

Figura 3.68. Página del perfil de la actividad seleccionada, tipo de categoría diferente a comida.

Visualizar horarios disponibles R53

Figura 3.69. Página con horarios para reservar.

Realizar reserva mediante formulario (R54)

Una vez el turista escogió un horario, se mostró un formulario (Figura 3.70) donde va a colocar sus datos para la reserva.

Figura 3.70. Formulario para realizar reserva.

Colocar reserva en estado pendiente o confirmada (R55)

Si la reserva cumple con los parámetros para ser confirmada automáticamente se confirma con una alerta (Figura 3.71) informando que la actividad fue agendada y enviando un correo electrónico (Figura 3.72).

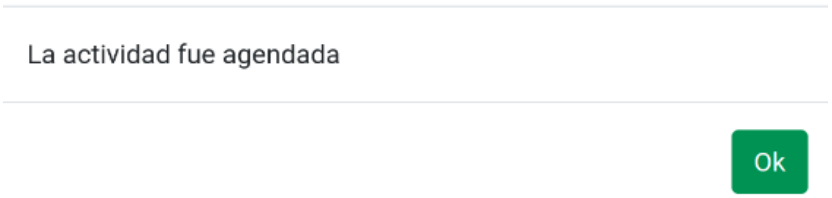

Figura 3.71. Alerta actividad agendada.

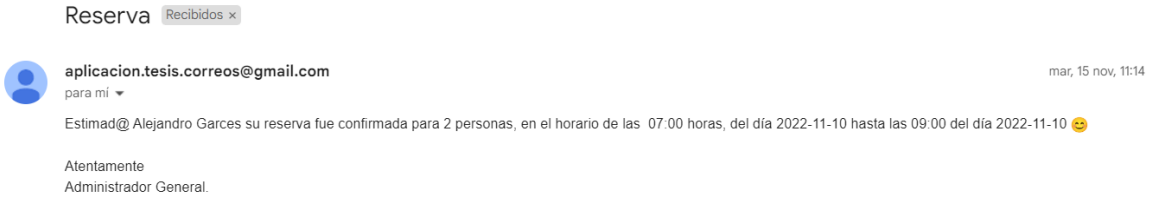

Figura 3.72. Correo de confirmación de reserva.

Caso contrario, si la reserva no cumple los parámetros para ser confirmada automáticamente, se convierte en reserva pendiente. Aparece un mensaje de alerta informando que se encuentra pendiente de aprobación como se muestra en la Figura 3.73, adicional se envía un correo electrónico informando que la reserva se encuentra pendiente (Figura 3.74), se muestra al cliente una página con la información de la reserva, para iniciar un diálogo a través de WhatsApp (Figura 3.75).

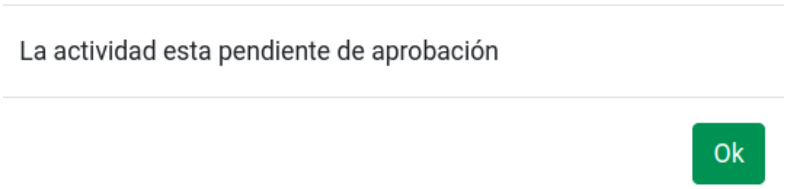

Figura 3.73. Mensaje de alerta actividad pendiente de aprobación.

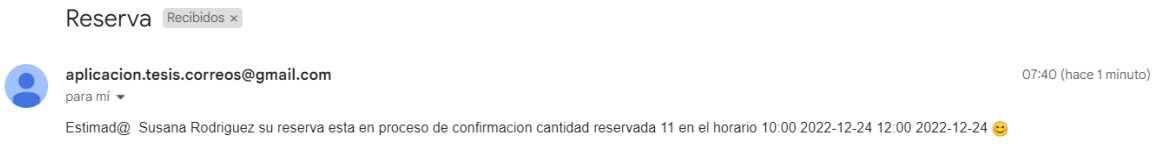

Figura 3.74. Correo de reserva pendiente de aprobación.

Establecer términos específicos a través de Whatsapp (R56, RNF1)

Figura 3.75. Mensaje de WhatsApp.

3.3. ANÁLISIS DE RESULTADOS

Después de haber comprobado el correcto funcionamiento de los módulos del sistema, se procedió a realizar las pruebas de usuario por cada uno de ellos. Para esto se recurrió a 10 usuarios: 5 usuarios con perfil turista, 3 usuarios con perfil encargado de establecimiento y 2 usuarios con perfil administrador. Posteriormente, se realizó una encuesta a estos 10 usuarios para determinar sus apreciaciones.

Todas las encuestas fueron preguntas de sí y no, asociadas a los requisitos con los cuales se construyó el sistema, y se esperó idealmente que todas aquellas que se refieren a funcionalidad conseguida tengan un 100% en la opción sí, y la última pregunta que se refiere a errores encontrados se esperó que tenga un 0% en sí. Existieron errores reportados, que posteriormente fueron solventados, y que se muestran en la Tabla 4.4.

En resumen, antes de las correcciones finales, los resultados muestran que casi la totalidad de funcionalidades del sistema pudieron ser llevadas a cabo por los usuarios; así, 21 de 22 en el módulo de administración, 31 de 32 en el módulo del encargado y 7 de 8 en el módulo del turista. Luego de haber corregido los 6 errores puntuales finales, se estima que el sistema cumple con lo definido en los requerimientos.

3.3.1. PRUEBAS DEL MÓDULO DE ADMINISTRACIÓN

En la Tabla *3.1* se muestran los resultados de la encuesta de satisfacción de los usuarios administradores, obtenidos después de realizar las pruebas respectivas. La encuesta fue realizada a 2 personas, lo que explica el alto porcentaje de error en la pregunta 22.

Tabla 3.1. Pruebas del módulo de administración.

#	Pregunta	Resultado % Si	Resultado % No
1	¿Usted consiguió iniciar sesión como administrador principal?	100%	0%
2	¿Usted consiguió cambiar de contraseña como administrador principal?	100%	0%

3	¿Usted consiguió visualizar lista de administradores?	100%	0%
4	¿Usted consiguió crear un nuevo administrador?	100%	0%
5	¿Usted consiguió modificar a un administrador?	100%	0%
6	¿Usted consiguió borrar a un administrador?	100%	0%
7	¿Usted consiguió visualizar a un administrador a detalle?	100%	0%
8	¿Usted consiguió visualizar la lista de categorías?	100%	0%
9	¿Usted consiguió ingresar nueva categoría?	100%	0%
10	¿Usted consiguió modificar una categoría?	100%	0%
11	¿Usted consiguió borrar una categoría?	100%	0%
12	¿Usted consiguió visualizar una categoría a detalle?	100%	0%
13	¿Usted consiguió visualizar la lista de encargados de establecimientos?	100%	0%
14	¿Usted consiguió crear un nuevo encargado de establecimientos?	100%	0%
15	¿Usted consiguió modificar a un encargado de establecimiento?	100%	0%
16	¿Usted consiguió borrar a un encargado de establecimiento?	100%	0%
17	¿Usted consiguió visualizar un encargado de establecimiento a detalle?	100%	0%
18	¿Usted consiguió visualizar lista de encargados de establecimientos pendientes?	100%	0%

19	¿Usted consiguió aceptar la solicitud de encargado de establecimiento?	100%	0%
20	¿Usted consiguió eliminar una solicitud de encargado de establecimiento?	100%	0%
21	¿Usted consiguió iniciar de sesión como administrador?	100%	0%
22	¿Se presentó algún problema con el módulo de administración? De ser así, detalle el error.	50%	50%

3.3.2. PRUEBAS DEL MÓDULO DEL ENCARGADO DE ESTABLECIMIENTO

En la Tabla *3.2* se muestran los resultados de la encuesta de satisfacción de los usuarios encargados de establecimiento, obtenidos después de realizar las pruebas respectivas. La encuesta fue realizada a 3 personas.

Tabla 3.2. Pruebas del módulo del encargado de establecimiento

#	Pregunta	Resultado % si	Resultado % no
1	¿Usted consiguió registrarse en el sistema como encargado del establecimiento?	100%	0%
2	¿Usted consiguió iniciar sesión como encargado?	100%	0%
3	¿Usted consiguió resetear su contraseña?	66.66%	33.33%
4	¿El sistema le permite visualizar la lista de establecimientos?	100%	0%
5	¿Usted consiguió ingresar un nuevo establecimiento?	100%	0%
6	¿Usted consiguió modificar el establecimiento?	100%	0%
7	¿Usted consiguió borrar el establecimiento?	100%	0%
8	¿Usted consiguió visualizar el establecimiento a detalle?	100%	0%

9	¿El sistema le permite visualizar la lista de actividades?	100%	0%
10	¿Usted consiguió ingresar una nueva actividad?	100%	0%
11	¿Usted consiguió modificar la actividad?	100%	0%
12	¿Usted consiguió borrar la actividad?	100%	0%
13	¿Usted consiguió visualizar la actividad a detalle?	100%	0%
14	¿El sistema le permite visualizar la lista de horarios?	100%	0%
15	¿Usted consiguió ingresar un nuevo horario?	100%	0%
16	¿Usted consiguió borrar el horario?	100%	0%
17	¿Usted consiguió visualizar el horario a detalle?	100%	0%
18	¿El sistema le permite visualizar la lista de platos?	100%	0%
19	¿Usted consiguió ingresar un nuevo plato?	100%	0%
20	¿Usted consiguió modificar el plato?	100%	0%
21	¿Usted consiguió borrar el plato?	100%	0%
22	¿Usted consiguió visualizar el plato a detalle?	100%	0%
23	¿El sistema le permite visualizar la lista de reservas confirmadas?	100%	0%
24	¿Usted consiguió agregar una reserva confirmada?	100%	0%
25	¿Usted consiguió eliminar la reserva confirmada?	100%	0%
26	¿Usted consiguió visualizar la reserva confirmada a detalle?	100%	0%
27	¿El sistema le permite visualizar la lista de reservas pendientes?	100%	0%
28	¿El sistema le permite aceptar reservas pendientes?	100%	0%
29	¿Usted consiguió modificar la reserva pendiente?	100%	0%
30	¿Usted consiguió borrar la reserva pendiente?	100%	0%

31	¿Usted consiguió visualizar la reserva pendiente a detalle?	100%	0%
32	¿Se presentó algún problema con el módulo del encargado de establecimiento? De ser así, detalle el error	33.33%	66.66%

3.3.3. PRUEBAS DEL MÓDULO DEL TURISTA

En la Tabla *3.3* se muestran los resultados de la encuesta de satisfacción de los usuarios de perfil turista, obtenidos después de realizar las pruebas respectivas. Esta prueba fue realizada a 6 personas.

Tabla 3.3. Pruebas del módulo del turista.

#	Pregunta	Resultado % si	Resultado % no
1	¿Usted consiguió visualizar categorías?	100%	0%
2	¿Usted consiguió visualizar actividades por categorías?	100%	0%
3	¿Usted consiguió visualizar el perfil de cada actividad?	100%	0%
4	¿Usted consiguió visualizar los horarios disponibles de la actividad escogida?	100%	0%
5	¿Usted consiguió realizar una reserva mediante formulario?	100%	0%
6	¿El sistema guardo la reserva como confirmada o pendiente?	100%	0%
7	En caso de que su reserva se haya quedado en estado pendiente, ¿se abrió el link de Whatsapp para establecer términos específicos con el encargado?	100%	0%
8	¿Tuvo algún problema con el módulo de turista? De ser así, detalle el error.	60%	40%

3.3.4. REPORTE DE ERRORES

Como se puede observar en la Tabla *3.1*, Tabla *3.2* y Tabla *3.3*, la mayoría de los requerimientos del usuario fueron cumplidos satisfactoriamente. Durante las pruebas se reportaron algunos errores, los cuales se detallan en la Tabla *3.4*; así también se describe la solución de cada uno de ellos.

Tabla 3.4. Reporte de errores.

Error identificado	**Solución**	**Pregunta de encuesta**
Cuando el usuario administrador principal crea un nuevo usuario administrador y se detecta la existencia de un usuario con el mismo correo registrado no se genera ninguna alerta de error ni se crea el usuario.	Enviar código de error desde el *backend* al detectar que el usuario ya existe y capturarlo desde el *fronted* para mostrar una alerta de error.	Tabla 3.1 pregunta 21
Cuando el usuario intenta recuperar contraseña el sistema no reconoce a que perfil pertenece.	En la base de datos se actualizan todos los registros correspondientes al usuario, permitiendo que este posea una sola contraseña independiente del perfil.	Tabla 3.2 pregunta 3
Cuando el usuario turista genera una reserva en estado pendiente, no se actualiza el aforo disponible en el horario de la reserva.	Se disminuye el aforo temporalmente durante 2 días hasta que la reserva sea confirmada, si esta se confirma se actualiza de manera permanente el valor del aforo caso contrario se aumenta el valor del aforo	Tabla 3.3 Pregunta 8
Cuando el usuario turista ve los horarios disponibles	Se configura el *backend* para que los horarios con aforo 0	Tabla 3.3 Pregunta 8

también se muestran los horarios que tienen aforo 0.	no se envíen en la lista de horarios hacia el *fronted*, por lo tanto, no se muestran en el sistema web	
Cuando el usuario turista genera una reserva cuya categoría es comida, no se muestran los platos escogidos en el módulo del encargado en la parte de ver reservas a detalle.	Desde el *backend* se actualizó el tipo de dato de object a list en el registro que va a contener el array de los platos de la reserva, por lo que la información se almaceno exitosamente y al solicitarla desde el *fronted* se muestra la lista en el sistema web	Tabla 3.3 Pregunta 8
Cuando el usuario turista revisó las actividades por categoría no se mostraba la dirección de establecimiento.	Se eliminó el segmento de código que sobrescribe el atributo dirección como vacío permitiendo que este atributo se muestre correctamente en el sistema web	Tabla 3.3 Pregunta 8

Comentarios adicionales del usuario

Como observación se pidió agregar el costo de cada actividad. Se procedió a hacer los cambios en la base de datos, en el *backend* y en *fronted* para agrega este atributo.

En la Figura 3.76 se muestra la actualización del tablero Kanban.

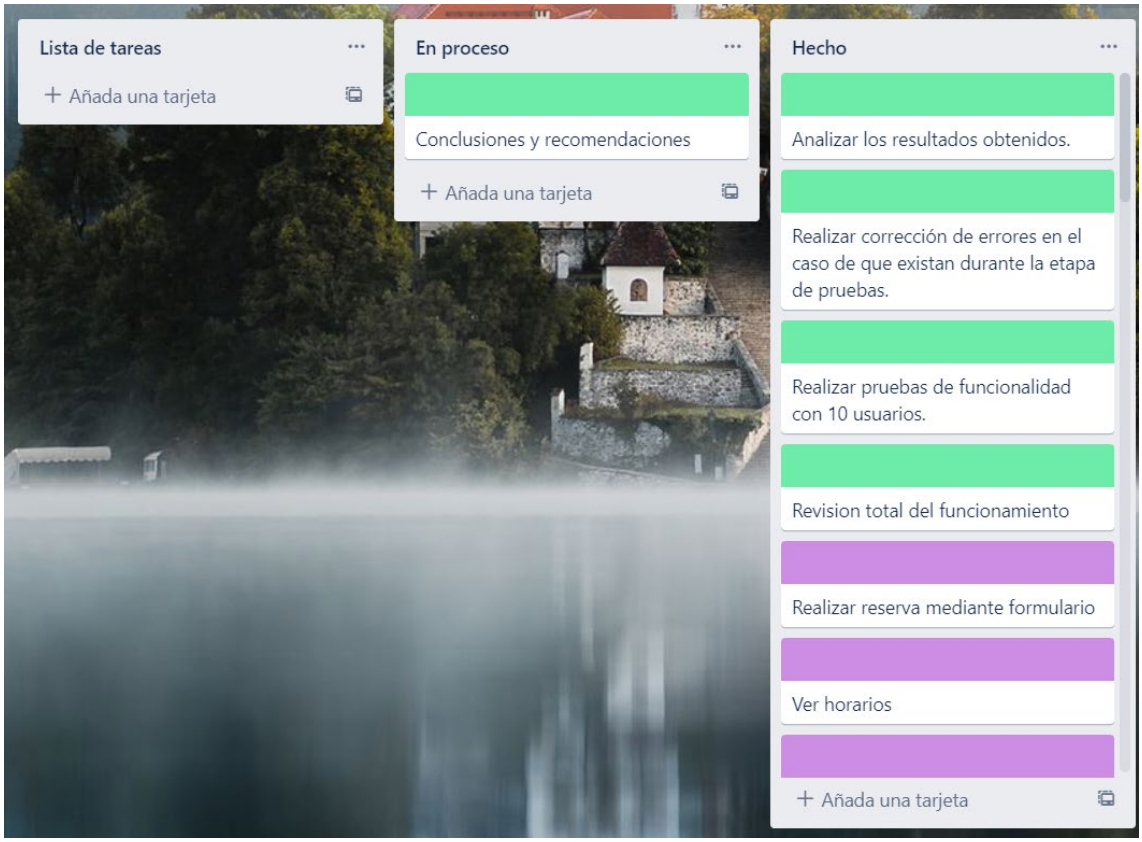

Figura 3.76. Actualización de tablero, conclusiones y recomendaciones.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Se ha desarrollado un sistema web para brindar visibilidad a las actividades turísticas que se llevan a cabo en la parroquia de Nono. Este sistema web cuenta con tres módulos. El módulo de administración permite gestionar administradores, encargados, y categorías de actividades. El módulo del encargado de establecimiento permite la gestión de establecimientos, sus actividades, conjuntamente con sus horarios y reservas. Finalmente, el módulo del turista que permite visualizar las actividades según diferentes categorías y generar una reserva en el horario de interés.

- Se analizaron los conceptos fundamentales para el desarrollo del proyecto propuesto. Los conceptos fundamentales incluyen el stack MERN (*MongoDB*, *Express*JS, *React*, *NodeJS*) como conjunto de tecnologías para el desarrollo del sistema web.

- Se diseñaron los módulos del sistema especificados en el alcance planteado en el plan del presente proyecto. Para esto, inicialmente se realizaron entrevistas a 2 dueños de 2 establecimientos de la parroquia de Nono, en función de las cuales se establecieron las historias de usuario, y los requerimientos funcionales y no funcionales. Luego del análisis respectivo, se realizaron los diagramas de casos de uso, de secuencia y de clases, que sirvieron para diseñar el sistema desde diferentes perspectivas.

- Se implementaron los módulos del sistema según el diseño realizado, siguiendo la arquitectura MVC (*Modelo, Vista, Controlador*). Para la capa Modelo se usó MongoDB. En la capa Controlador se usaron las tecnologías *NodeJS* y *ExpressJS* que facilitaron la generación de *APIs* para la conexión con MongoDB. Finalmente, para la capa Vista se utilizó *React* para generar las interfaces que interactúan con el usuario final.

- Se llevaron a cabo un conjunto de pruebas de usuario y se analizaron sus resultados. las pruebas se realizaron con 10 usuarios: 5 turistas, 3 usuarios encargados de establecimientos y 2 usuarios administradores, quienes usaron el sistema web y pudieron identificar errores o plantearon sugerencias sobre el mismo. Se solucionaron los problemas reportados por los usuarios. Por todo esto, el sistema cumple con el alcance y requisitos.

- El sistema construido puede crecer o escalar de manera dináminca en cuanto a la información que muestra: categorías, actividades, establecimientos, encargados y administradores, lo cual brinda escalabilidad. Esto se debe a las facilidades de gestión implementadas.

- El sistema permite la automatización del proceso de reservas a los turistas, proceso principal en este tipo de sistemas; y, además, brinda soporte manual para aquellos casos que lo requieran. En el segundo caso, apegándose al esquema actual de funcionamiento que tienen los establecimientos mediante una comunicación directa a través de WhatsApp para establecer los términos de la reserva.

- Finalmente, es necesario resaltar los beneficios de las tecnologías usadas en el desarrollo de este trabajo. Por ejemplo, en el caso de *Cloudinary, que* permite un almacenamiento en la nube de las imágenes, su uso reduce el tamaño de la base de datos al tener únicamente información en texto plano almacenada. De igual modo, una de las ventajas de usar *ReactJS* fue que permitió dividir y reutilizar los

componentes del sistema web, lo cual facilitó la creación e integración de nuevos componentes.

4.2. RECOMENDACIONES

- Se podría implementar otros módulos como facturación o inventario.
- Se puede implementar filtros de búsqueda en el módulo de turista donde pueda directamente buscar el perfil de una actividad.
- Alternativamente a lo ya implementado se puede ocupar otro servicio de autenticación como Firebase que admite la autenticación mediante contraseñas, números de teléfono, Google, Facebook y Twitter para agilizar el manejo de usuarios.
- Se puede generar un módulo de reportes con herramientas visuales como Power Bi para que el encargado revise las actividades más populares, fechas más populares, etc.
- Se recomienda realizar un análisis de disponibilidad en trabajos futuros, previo a una implementación del sistema.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] MongoDB Inc., “What is the MERN Stack?,” 2021. [Online]. Available: https://www.mongodb.com/. [Accessed: 28-Ago-2021].

[2] Prezi Inc., “Método Kanban,” 2022. [Online]. Available: https://prezi.com/p/ooz-cdy9isru/que-es-el-metodo-kanban/?fallback=1. [Accessed: 8-Ago-2022].

[3] Kanbanize, “Que es kanban,” 2022. [Online]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban. [Accessed: 25-Ago-2022].

[4] IBM, “Three tier architecture,” 2022. [Online]. Available: https://www.ibm.com/mx-es/cloud/learn/three-tier-architecture. [Accessed: 25-Ago-2022].

[5] MongoDB Inc., “Data Model Design,” 2022. [Online]. Available: https://www.mongodb.com/docs/manual/core/data-model-design/. [Accessed: 22-Ago-2022].

[6] Mongoosejs, “Mongoose,”. [Online]. Available: https://mongoosejs.com/. [Accessed: 10-Ago-2022].

[7] DigitalOcean LLC, “Cómo crear un servidor web en Node.js con el módulo HTTP,” 2022. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-create-a-web-server-in-node-js-with-the-http-module-es. [Accessed: 19-Sep-2022].

[8] Fundación OpenJS, “Nodejs,” 2022. [Online]. Available: https://nodejs.dev/en/learn/. Accessed: 10-Ago-2022.

[9] Meta Platforms Inc., “Reactjs,” 2022. [Online]. Available: https://es.reactjs.org/. [Accessed: 10-Ago-2022].

[10] React-Bootstrap, “React bootstrap,”. [Online]. Available: https://react-bootstrap.github.io/. [Accessed: 23-Dic-2022].

[11] Formium, “Formik,” 2020. [Online]. Available: https://formik.org/. [Accessed: 23-Dic-2022].

[12] Npm Inc, “Yup,” [Online]. Available: https://www.npmjs.com/package/yup. [Accessed: 23-Dic- 2022].

[13] J. J. Sarjeant, “Axios,” 2020. [Online]. Available: https://axios-http.com/. [Accessed: 23-Dic-2022].

[14] Microsoft, “Visual Studio,” 2022. [Online]. Available: https://code.visualstudio.com/docs. Accessed: 15-Ago-2022.

[15] Postman Inc., “Postman,” 2022. [Online]. Available: https://www.postman.com/. [Accessed: 10-Ago-2022].

[16] Github, “Github,” 2022. [Online]. Available: https://docs.github.com/es. [Accessed: 10-Ago-2022].

[17] Gobierno Autónomo Descentralizado de Nono, "Plan de desarrollo y ordenamiento territorial NONO 2015-2019", Quito, 2015.

[18] El Comercio, "Gobierno anuncia ‘nueva normalidad’ en Ecuador desde el 4 de mayo del 2020; ¿finaliza la cuarentena por el covid-19?", El Comercio, 24 mayo 2020.

[19] F. Becerra, J.M Vega, M.N Orellana, "Impacto financiero del covid-19 en el turismo ecuatoriano: Estrategias empresariales para la reactivación," mayo 2021. [En línea]. Available: https://www.593dp.com. [Último acceso: 14 julio 2021].

[20] IBM Corporation, “Definición de casos de uso,” 2016. [Online]. Available: https://www.ibm.com/docs/es/elm/6.0.3?topic=requirements-defining-use-cases. Accessed: 26-Sep-2022.

[21] Lucid Software Inc., “¿Para qué necesitas crear un diagrama UML?,” [Online]. Available: https://www.lucidchart.com/pages/es/diagrama-de-secuencia. [Accessed: 8-Ago-2022].

[22] Lucid Software Inc., “Uml Class Diagram,” [Online]. Available: https://www.lucidchart.com/pages/uml-class-diagram. [Accessed: 21-Dic-2022].

[23] Mozilla Foundation, “Control de acceso HTTP (CORS),” 2022. [Online]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/CORS. [Accessed:19-Oct-2022].

[24] Google Developers, “Documentación para desarrolladores de apps,” 2022. [Online]. Available: https://developer.android.com/. Accessed: 19-Oct-2022.

[25] Npm Inc., “Express-fileupload,” 2022. [Online]. Available: https://www.npmjs.com/package/express-fileupload. [Accessed: 19-Oct-2022].

[26] Microsoft , “¿Qué es middleware?,” 2022. [Online]. Available: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-middleware/. Accessed: 19/10/2022.

[27] Atlassian, “Historias de usuario con ejemplos y plantilla,” 2022. [Online]. Available: https://www.atlassian.com/es/agile/project-management/user-stories#:~:text=Una%20historia%20de%20usuario%20es%20la%20unidad%20de%20trabajo%20m%C3%A1s,del%20usuario%20final%20o%20cliente. [Accessed: 21-Dic-2022].

[28] Cloudinary, “Cloudinary,” [Online]. Available: https://cloudinary.com/features. [Accessed: 22-Dic-2022].

ANEXOS

ANEXO A. Historias de usuarios.

ANEXO B. Manual de usuario.

ANEXO A - Historias de usuario

Tabla A.0.1. Historia de usuario 2 - Gestionar categorías.

HISTORIA DE USUARIO	H2
Nombre: Gestionar categorías	
Usuario: Administrador y Administrador principal	
Programador responsable: Sandy Donoso	
Descripción: como usuario administrador quiero gestionar las categorías para darle una clasificación a las actividades creadas por los dueños del negocio. Cada una va a contar al menos con un nombre y una descripción.	
Criterios de aceptación. El administrador puede realizar las siguientes acciones: • Visualizar lista de categorías. • Ingresar categoría. • Modificar categoría. • Borrar categoría. • Visualizar categoría a detalle.	

Tabla A.0.2. Historia de usuario 3 - Gestionar encargados de establecimientos.

HISTORIA DE USUARIO	H3
Nombre: Gestionar encargados de establecimientos	
Usuario: Administrador y Administrador principal	
Programador responsable: Sandy Donoso	
Descripción: como usuario administrador deseo gestionar a los encargados de establecimientos (quienes actúan en representación del establecimiento) para controlar o restringir el acceso de usuarios a la plataforma.	

Criterios de aceptación. El administrador puede realizar las siguientes acciones: • Visualizar lista de encargados de establecimientos. • Crear encargado de establecimiento. • Modificar encargado de establecimiento. • Borrar encargado de establecimiento. • Visualizar encargado de establecimiento a detalle.

Tabla A.0.3. Historia de usuario 7 - Inicio de sesión en el sistema encargado.

HISTORIA DE USUARIO	H7
Nombre: Inicio de sesión en el sistema encargado	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero ingresar al sistema para poder usar las opciones de gestión según mi perfil.	
Criterios de aceptación: • El sistema debe pedir el correo electrónico y la contraseña para realizar el inicio de sesión del encargado. • El sistema tiene que validar que el correo electrónico y la contraseña no estén vacíos y coincidan con lo registrado. De no ser así debe mostrarse un mensaje de error que indique “correo electrónico o contraseña incorrectos”. • El sistema debe validar que se ingrese como perfil encargado.	

Tabla A.0.4. Historia de usuario 8 - Gestionar establecimientos.

HISTORIA DE USUARIO	H8
Nombre: Gestionar establecimientos	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	

Descripción: como usuario encargado de establecimiento quiero administrar los establecimientos y gestionar la información de cada uno de ellos. Cada establecimiento al menos contará con un nombre, descripción, ubicación, redes sociales y actividades.
Criterios de aceptación. El administrador pude realizar las siguientes acciones: • Visualizar lista de establecimientos. • Crear establecimiento. • Modificar establecimiento. • Borrar establecimiento. • Visualizar establecimiento a detalle.

Tabla A.0.5. Historia de usuario 10 - Gestionar horarios.

HISTORIA DE USUARIO	H10
Nombre: Gestionar horarios	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero gestionar los horarios disponibles para cada actividad. Los horarios cuentan con al menos una fecha inicio, fecha fin, horario inicio, horario fin, aforo, actividad.	
Criterios de aceptación: • Visualizar lista de horarios. • Crear horario. • Borrar horario. • Visualizar horario a detalle.	

Tabla A.0.6. Historia de usuario 11 - Gestionar platos.

HISTORIA DE USUARIO	H11
Nombre: Gestionar platos	
Usuario: Encargado de establecimiento	

Programador responsable: Sandy Donoso
Descripción: como usuario encargado de establecimiento quiero gestionar platos para cuando la actividad se alimentación. Los platos cuentan con al menos un nombre, descripción y costo.
Criterios de aceptación: • Visualizar lista de platos. • Crear plato. • Modificar plato. • Borrar plato. • Visualizar plato a detalle.

Tabla A.0.7. Historia de usuario 9 - Gestionar actividades.

HISTORIA DE USUARIO	H9
Nombre: Gestionar actividades	
Usuario: Encargado de establecimiento	
Programador responsable: Sandy Donoso	
Descripción: como usuario encargado de establecimiento quiero gestionar las actividades que voy a ofrecer. Las actividades pertenecen a una categoría y cuentan con al menos un nombre, descripción, horarios para reserva y fotografías.	
Criterios de aceptación: • Visualizar lista de actividades. • Crear actividad. • Modificar actividad. • Borrar actividad. • Visualizar actividad a detalle.	

ANEXO B – Manual de usuario

El manual de usuario se presenta de forma digital en la carpeta compartida accesible en (https://epnecuador-my.sharepoint.com/:f:/g/personal/sandy_donoso_epn_edu_ec/ErOANeRLFxdIoc-4uaER8OsB6pLAQGfeH_ZcmVo492rOtg?e=8Bxpkh) y en el CD que se dejará en el Laboratorio de informática.

ORDEN DE EMPASTADO