UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingeniería

Desarrollo de aplicación web de tipo SPA para la optimización del registro, administración y análisis de reservas para el Hotel Sea World, mediante el uso de la arquitectura MEAN

Oscar Enrique Chiriboga Zambrano

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 20 de diciembre de 2022

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Desarrollo de aplicación web de tipo SPA para la optimización del registro, administración y análisis de reservas para el Hotel Sea World, mediante el uso de la arquitectura MEAN

Oscar Enrique Chiriboga Zambrano

Nombre del profesor, Título académico **Daniel Fellig Goldvechmiedt, MSc**

Quito, 20 de diciembre de 2022

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en la Ley Orgánica de Educación Superior del Ecuador.

Nombres y apellidos:	Oscar Enrique Chiriboga Zambrano
Código:	00205604
Cédula de identidad:	1720544632
Lugar y fecha:	Quito, 20 de diciembre de 2022

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses".

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

El Hotel Sea World es un negocio familiar ubicado en el balneario de San Jacinto, Manabí. Su visión es posicionarse como uno de los mejores hoteles a nivel nacional como internacional. Por este motivo, su principal enfoque es brindar una experiencia única a sus clientes mediante un servicio de calidad. Con ello en mente, los administradores consideran que la tecnología es un factor crucial para potenciar su crecimiento. Por lo cual han dirigido sus esfuerzos en invertir en herramientas para posicionarse en internet, así como implementos tecnológicos que optimicen su trabajo en el hotel.

Sin embargo, un problema que llevan desde que iniciaron el negocio es mantener un registro de las reservas y clientes, así como, tener una plataforma propia que facilite el proceso de realizar una reserva. Anteriormente, el hotel ya ha trabajado con dos plataformas para solventar estas dificultades. No obstante, estas plataformas han sido poco intuitivas y funcionales para los administradores. Por esta razón, el presente proyecto desarrolla una aplicación web de tipo SPA mediante una de las tecnologías más posicionadas en el mercado, el stack MEAN. Con ello, no sólo se plantea crear un sistema que simplifique todos los procesos involucrados con las reservas del hotel, sino, que también sea una herramienta que permite a los administradores conocer el rendimiento del negocio.

Palabras clave: Aplicación de una sola página, Plataforma como servicio, Aplicación web, API de REST, Stack MEAN, MongoDB, Express, Angular, Node.js, Sistema de administración de reservas.

ABSTRACT

Sea World Hotel is a family business located in the seaside resort of San Jacinto, Manabí. Its vision is to position itself as one of the best hotels nationally and internationally. For this reason, its focus is to provide a unique experience to its customers through quality service. The managers consider that technology is a crucial factor to enhance their growth. For this reason, they have focused their efforts on investing in tools to position themselves on the Internet, as well as technological tools to optimize their work at the hotel.

However, a problem they have been facing since they started the business is keeping a record of reservations and clients, as well as having their own platform to facilitate the process of making a reservation. Previously, the hotel has already worked with two platforms to solve these difficulties. However, these platforms have not been very intuitive and functional for the administrators. For this reason, this project develops a SPA-type web application using one of the most widely used technologies on the market, the MEAN stack. With this, it is not only proposed to create a system that simplifies all the processes involved with hotel reservations, but also a tool that allows managers to know the performance of the business.

Key words: Single-page application, Platform as Service, Web Application, REST API, MEAN Stack, MongoDB, Express, Angular, Node.js, Booking System.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Descripción del problema

El balneario de San Jacinto es un pueblo pesquero ubicado en el cantón Sucre, provincia de Manabí. San Jacinto cuenta con una variedad de paisajes, entre playas y manglares. Además, se distingue por su exquisita gastronomía, su gente y la tranquilidad que se percibe en su ambiente. Estas características hacen que el turismo sea una de las actividades principales de este paraíso natural.

Con el objetivo de satisfacer la demanda de turismo en esta zona, se funda el Hotel Sea World en el año 2012. El enfoque de este emprendimiento familiar es destacarse por su servicio de calidad y brindar una experiencia única a cada cliente. Para ello, el Hotel Sea World conoce la relevancia de la tecnología en la actualidad, por lo que busca estar a la vanguardia en temas de redes sociales y plataformas de búsqueda de alojamiento.

En este sentido, facilitar el proceso de reservas a los clientes como mantener información de estas es crucial para el hotel. Actualmente, el hotel cuenta con un sistema de administración de reservas no operativo, mientras que plataformas como Booking dificultan a los administradores la visualización de datos y estadísticas, debido a la complejidad de sus interfaces gráficas, además de que cobran altas tarifas por su servicio.

Por este motivo, el presente proyecto plantea el desarrollo de un sistema de administración de reservas para el Hotel Sea World que se adapte a sus necesidades, sea amigable para el usuario, ofrezca métricas sobre su rendimiento y sirva como repositorio de información, para de este modo mejorar la calidad del servicio del hotel y posicionarlo en el ámbito turístico.

Marco teórico

Desde que el Internet se hizo accesible al público en 1994, se ha transformado en una de las tecnologías más relevantes para la humanidad. Además de servir como medio para transmitir información, hoy en día también representa la plataforma por excelencia para una gran cantidad de aplicaciones que cada vez son más innovadoras y sofisticadas. Estas aplicaciones son las aplicaciones web, las cuales se caracterizan por ser invocadas mediante un navegador web, a través de Internet (Jazayeri, 2007).

Con el auge de las aplicaciones web dinámicas, surgieron nuevas tecnologías para facilitar su desarrollo. Tal es el caso de stack MEAN, el cual comprende un conjunto de herramientas modernas cuya característica en común es que se basan en el lenguaje de programación JavaScript (Poulter et al., 2015). Esta cualidad hace que el stack MEAN sea ampliamente utilizado, puesto que facilita el trabajo de los programadores tanto en el área del back-end como el área del front-end, al tener que trabajar con un único lenguaje (Holmes & Harber, 2019). Para entrar en detalle, las tecnologías que constituyen el stack MEAN son:

MongoDB (Base de datos): Es un sistema de base de datos orientado a documentos (Zhao et al., 2013). Este sistema de base de datos NoSQL de alto rendimiento se creó en torno al formato de datos JSON (JavaScript Object Notation) (Poulter et al., 2015).

Express (Marco de trabajo web): Express.js brinda un contenedor alrededor de la interfaz de bajo nivel de Node. De esta manera, da a los desarrolladores un medio conveniente para manejar operaciones de tipo HTTP (como GET y POST) y de enrutamiento (Poulter et al., 2015).

Angular (Marco de trabajo para Front-end): Angular provee un marco de trabajo del lado del cliente, para crear aplicaciones web llamadas aplicaciones de una página; Single Page Application (SPA) (Poulter et al., 2015). Las SPA son aplicaciones web que procesan una

solicitud de un usuario sin actualizar la página por completo (Oh et al., 2013). De esta manera, la entrega de plantillas, la lógica de la aplicación, el flujo de usuarios y el procesamiento de datos se pueden administrar en el navegador (Holmes & Harber, 2019).

Node (Plataforma para servidor web): Node.js es una plataforma de software cuya utilidad es crear un servidor web de alto rendimiento y construir aplicaciones web encima de este (Poulter et al., 2015). Contiene una biblioteca incorporada de servidor http. De esta manera, no es necesario ejecutar una aplicación de servidor web aparte (Holmes & Harber, 2019).

Respecto a la arquitectura de una aplicación que emplea el stack MEAN, es común utilizar una API de transferencia de estado representacional (REST) para invocar la funcionalidad de la aplicación. REST se refiere a un estilo de arquitectura sin mantener estado de sesión para diseñar aplicaciones en red (Zhou et al., 2014). Por lo tanto, una API de REST es una interfaz de programación de aplicaciones, la cual habilita la comunicación entre distintas aplicaciones. En el caso del stack MEAN, la API de REST se utiliza para crear una interfaz sin estado de sesión que habilite la comunicación entre la base de datos y la aplicación (Holmes & Harber, 2019).

Finalmente, el stack MEAN permite crear aplicaciones web modernas y de gran complejidad. Para que sean accesibles al público, estas aplicaciones deben correr en línea. Esta tarea se puede lograr de dos maneras en el caso de una aplicación desarrollada con Node.js: la primera opción es tener un servidor y configurarlo para Node, mientras que la segunda opción es emplear una plataforma como servicio (PaaS). Una PaaS, también conocida como plataforma de aplicaciones en la nube permite implementar y desplegar rápidamente aplicaciones basadas en web sin la complejidad ni los costos de comprar y manejar la infraestructura necesaria (Gass et al., 2014).

Objetivos

Objetivo General.

Construir un prototipo funcional de una aplicación web de tipo SPA para agilizar el proceso de reservas y obtención de analíticas del Hotel Sea World, mediante la implementación de las tecnologías que comprende el stack MEAN y el uso de la plataforma como servicio de Heroku.

Objetivos Específicos.

- Desarrollar una API de REST para crear una interfaz sin estado de sesión para la base de datos, que habilite una vía a la aplicación para trabajar con los datos.
- Crear un dashboard de analíticas mediante MongoDB Charts que permite al hotel conocer datos como su rendimiento e ingresos en un período dado.
- Implementar un sistema robusto de autenticación y de pagos a través de la arquitectura MEAN, para brindar confianza y seguridad a los clientes y administradores del hotel.

Metodología de trabajo

La metodología de trabajo empleada para el presente proyecto está fundamentada en las metodologías de desarrollo ágil de software. En este sentido, se dividió la aplicación en una serie de entregables, para manejar el proyecto bajo un ciclo de vida incremental. De este modo, cada uno de uno de los entregables propuestos ampliaba la funcionalidad de la aplicación, y permitía que se haga pruebas modulares y de integración lo antes posible. Respecto a los diversos entregables definidos, cabe mencionar que contaron con su propio ciclo de vida constituido por las siguientes fases:

- Planificación.
- Estudio de requisitos.
- Diseño.

- Programación
- Evaluación.

De las etapas mencionadas, se dio énfasis a las fases de planificación y estudio de requisitos. Con esto en mente, se consideró la norma ISO/IEC/IEEE 29148, la cual define disposiciones en relación con la ingeniería de requisitos para sistemas de software a lo largo del ciclo de vida (IEEE, 2018). Con fundamento en esta norma, se ejecutó el siguiente procedimiento para el análisis de requerimientos:

1. Establecer los requisitos del cliente: Se identificó los actores involucrados; en este caso los administradores del hotel, y sus requerimientos. Con ello, se definió las limitaciones de la solución, así como, la interacción entre el sistema propuesto y los usuarios. Tras obtener la lista de requerimientos, se los ordeno según su prioridad para tener noción de los requisitos esenciales, así como de los requisitos que pueden verse comprometidos debido a conflictos (Haik & Shahin, 2011).
2. Analizar los requerimientos: En base a los requisitos del cliente, se propuso las funcionalidades de los elementos que componen cada entregable de la aplicación.
3. Definición del plan: Se desarrolló el plan a seguir teniendo en cuenta los requisitos del sistema.

En función de las etapas de planificación y estudio de los requisitos, se ejecutó las fases de diseño y programación en los distintos entregables. Finalmente, la fase de evaluación fue hecha por los administradores del hotel, para garantizar de esta manera que el producto en desarrollo se ajuste a los requerimientos del hotel. Asimismo, se buscó el apoyo de consultores externos, que sirvieron como guía para la implementación de un buen diseño y arquitectura de software para la aplicación. Por lo tanto, el principal factor a evaluar fue la usabilidad y funcionalidad de la aplicación.

DESARROLLO DEL TEMA

Descripción de la aplicación

La aplicación propuesta para el presente proyecto consiste en dos secciones:

- Sistema de administración de reservas: Pertenece a los miembros del hotel. Conlleva la gestión de la información del negocio, como lo son clientes, habitaciones, reservas, tipos de reserva y visualización de estadísticas.
- Presentación de contenido a Clientes: Constituye la página actual del hotel. Está compuesta por cuatro vistas:
    - Inicio: Muestra información respecto a los servicios del hotel y datos de contacto.
    - Historia: Menciona la historia del hotel desde sus inicios.
    - Galería: Muestra fotos de las instalaciones del hotel y de su ubicación.
    - Reservar ahora: Permite a los clientes hacer una reserva en el hotel y realizar el proceso de pago.

A continuación, se describen en detalle los distintos actores y casos de usos que involucra la aplicación.

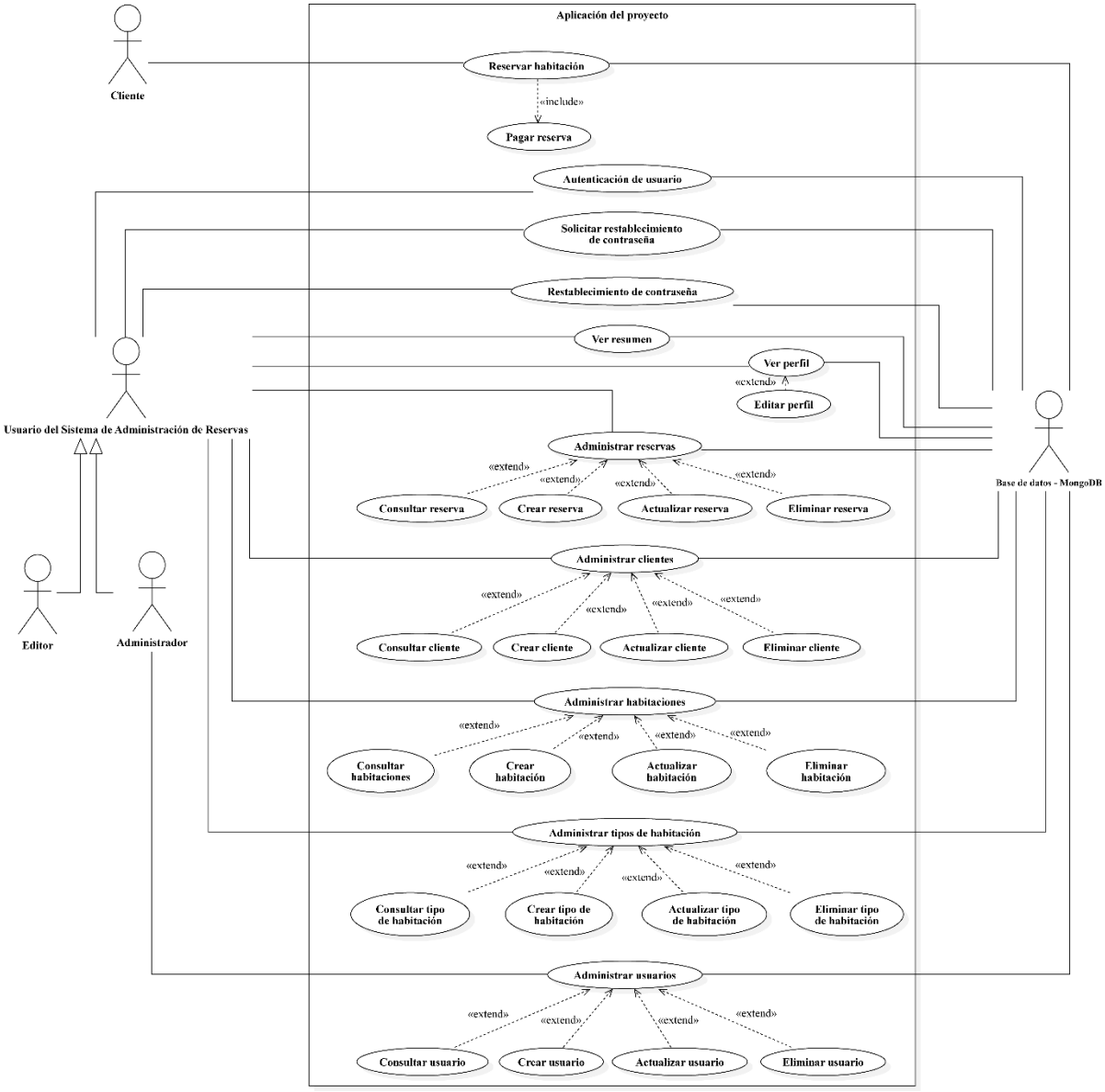

Figura 1. Diagrama de casos de uso para la aplicación del proyecto.

Actores.

- Usuario del sistema de administración de reservas: Actor que tiene acceso al sistema de administración de reservas. Puede realizar gestionar la creación, actualización, eliminación y lectura de reservas, clientes, habitaciones y tipos de habitación. Además, puede visualizar los gráficos de estadísticas del hotel.
- Administrador: Actor con acceso total a todas las funcionalidades del sistema de administración de reservas. Tiene acceso a la visualización y gestión de la creación, actualización y eliminación de usuarios.

- Editor: Solo tiene permisos para las funciones básicas para un usuario del sistema de administración de reservas.
- Cliente: Se refiere a los clientes del hotel. Este actor realiza los procesos de reservar habitaciones y realizar el pago de una reserva.
- Base de datos: Se refiere a la base de datos en MongoDB. Contiene información respecto a clientes, reservas, habitaciones, tipos de habitaciones y usuarios.

Casos de uso.

Casos de uso	Descripción
Autenticación de usuario	Este caso de uso hace referencia al inicio de sesión al sistema de administración de reservas. Por lo tanto, solo los administradores y editores del hotel tienen relación con este caso de su uso. Su flujo consiste en los siguientes pasos: 1. El usuario entra a la plataforma e ingresa su correo electrónico y contraseña en el formulario de inicio de sesión. 2. La plataforma verifica las credenciales ingresadas. 3. Si las credenciales son incorrectas o no se han completado todos los campos para iniciar sesión, se informa al usuario sobre el error. Caso contrario, se redirige al usuario a la vista de "Ver resumen".
Solicitar restablecimiento de contraseña	El objetivo de esta caso de uso es permitir a los usuarios solicitar el cambio de su contraseña. Consiste en: 1. El usuario da clic en la opción "¿Olvidaste tu contraseña?" que se encuentra en el formulario para iniciar sesión.

	2. Se despliega el formulario para ingresar el correo para el que se desea restablecer la contraseña. 3. El usuario ingresa su correo electrónico. 4. La plataforma verifica que el correo ingresado corresponda a uno de sus usuarios. 5. Si el correo electrónico es correcto, la plataforma genera el enlace para restablecimiento de contraseña al usuario y lo envía a su correo electrónico.
Restablecimiento de contraseña	Este proceso hace referencia al formulario para cambiar la contraseña de un usuario. Su flujo es: 1. El usuario entra al enlace de restablecimiento de contraseña que se encuentra en su correo electrónico. 2. Se despliega un formulario para establecer la nueva contraseña. 3. El usuario ingresa la nueva contraseña dos veces por seguridad. 4. La plataforma verifica las contraseñas. Si las contraseñas no coinciden, se notifica al usuario sobre el error. Caso contrario, se restablece la contraseña y se redirige al usuario al formulario de inicio de sesión.
Ver resumen	Se refiere a la vista principal del sistema de administración de reservas. Se llega a este caso de uso tras un inicio de sesión exitoso. 1. La plataforma muestra los gráficos estadísticos del hotel, los cuales son:

	• Tipo de reservas. • Reservas concretadas por provincia. • Total de reservas por semana. • Ingresos por semana. • Tipo de habitación más popular. • Ingresos totales. • Total de reservas. 2. La plataforma muestra en tablas la información de reservas, clientes y habitaciones. Se puede regresar a este caso de uso al dar clic en la opción "Resumen" en el menú de la aplicación.
Ver perfil	Un usuario entra a este caso de uso al dar clic en la opción "Mi Perfil" desde el menú de la aplicación. Aquí el usuario visualiza la información de su cuenta.
Editar perfil	Tras entrar a la opción "Mi Perfil", el usuario tiene la opción de modificar los datos de su cuenta. Para ello se realizan los siguientes pasos: 1. El usuario da clic en "Editar Perfil". 2. La plataforma habilita los campos para editar la cuenta. El campo para modificar el rol solo está disponible para los usuarios de tipo administrador 3. El usuario edita los campos de su cuenta que quiere modificar y da clic en guardar. 4. La plataforma verifica los datos ingresados. Si la información es correcta, se avisa al usuario sobre el

	cambio exitoso de sus datos y se cierra su sesión. Si la información es incorrecta, se notifica al usuario sobre el error.
Consultar reservas	El proceso de este caso de uso es: 1. El usuario da clic en la opción "Consultar Reservas" desde el menú de la aplicación. 2. La plataforma despliega la información de las reservas registradas en forma de tabla. 3. En esta tabla, el usuario puede buscar una reserva específica, ordenar las reservas según un campo, acceder al formulario de actualización de reserva y eliminar una reserva específica.
Crear reserva	Este caso de uso se basa en el siguiente procedimiento: 1. El usuario da clic en la opción "Registrar Reservas". 2. La plataforma despliega el formulario para la creación de reservas. 3. El usuario ingresa toda la información correspondiente a la reserva. 4. La plataforma verifica los datos ingresados por el usuario. • Si los datos no son correctos, se informa al usuario sobre el error. • Si la información es correcta, se registra la reserva, se notifica al usuario respecto a la creación exitosa y se restablece el formulario para ingresar una nueva reserva.

Actualizar reserva	Los pasos que involucran este caso de uso son: 1. El usuario da clic en el botón “Editar” de una reserva que se muestra en la tabla de reservas. 2. La plataforma despliega el formulario para actualizar dicha reserva. 3. El usuario modifica los campos que requiere y da clic en el botón “Guardar”. 4. La plataforma verifica la información ingresada. • Si los datos de la reserva son válidos, se actualiza la reserva y se da una notificación al usuario. Posteriormente, se redirige al usuario al caso de uso “Ver resumen”. • Si los datos no son válidos. Se informa al usuario sobre los campos incorrectos.
Eliminar reserva	El proceso de este caso de uso es: 1. El usuario da clic en el botón “Eliminar” de una reserva que se muestra en la tabla de reservas. 2. La plataforma pregunta al usuario si está seguro de que quiere eliminar la reserva. • Si el usuario da clic en “Aceptar”, se elimina la reserva y se notifica al usuario de la eliminación exitosa. • Si el usuario da clic en “Cancelar” no se elimina la reserva.

Consultar clientes	El proceso de este caso de uso es: 1. El usuario da clic en la opción “Consultar Clientes” desde el menú de la aplicación. 2. La plataforma despliega la información de los clientes registrados en forma de tabla. 3. En esta tabla, el usuario puede buscar un cliente específico, ordenar los clientes según un campo, acceder al formulario de actualización de cliente y eliminar un cliente específico.
Crear cliente	Este caso de uso se basa en el siguiente procedimiento: 1. El usuario da clic en la opción “Registrar Clientes”. 2. La plataforma despliega el formulario para la creación de clientes. 3. El usuario ingresa toda la información correspondiente al cliente. 4. La plataforma verifica los datos ingresados por el usuario. • Si los datos no son correctos, se informa al usuario sobre el error. • Si la información es correcta, se registra el cliente, se notifica al usuario respecto a la creación exitosa y se restablece el formulario para ingresar un nuevo cliente.
Actualizar cliente	Los pasos que involucran este caso de uso son: 1. El usuario da clic en el botón “Editar” de un cliente que se muestra en la tabla de clientes.

	2. La plataforma despliega el formulario para actualizar dicho cliente. 3. El usuario modifica los campos que requiere y da clic en el botón “Guardar”. 4. La plataforma verifica la información ingresada. • Si los datos del cliente son válidos, se actualiza el cliente y se da una notificación al usuario. Posteriormente, se redirige al usuario al caso de uso “Ver resumen”. • Si los datos no son válidos. Se informa al usuario sobre los campos incorrectos.
Eliminar cliente	El proceso de este caso de uso es: 1. El usuario da clic en el botón “Eliminar” de una reserva que se muestra en la tabla de clientes. 2. La plataforma pregunta al usuario si está seguro de que quiere eliminar el cliente. • Si el usuario da clic en “Aceptar”, se elimina el cliente y se notifica al usuario de la eliminación exitosa. • Si el usuario da clic en “Cancelar” no se elimina el cliente.
Consultar habitaciones	El proceso de este caso de uso es: 1. El usuario da clic en la opción “Consultar Habitación” desde el menú de la aplicación.

	2. La plataforma despliega la información de las habitaciones registradas en forma de tabla. 3. En esta tabla, el usuario puede buscar una habitación específica, ordenar las habitaciones según un campo, acceder al formulario de actualización de habitación y eliminar una habitación específica.
Crear habitación	Este caso de uso se basa en el siguiente procedimiento: 1. El usuario da clic en la opción "Registrar Habitaciones". 2. La plataforma despliega el formulario para la creación de habitaciones. 3. El usuario ingresa toda la información correspondiente a la habitación. 4. La plataforma verifica los datos ingresados por el usuario. • Si los datos no son correctos, se informa al usuario sobre el error. • Si la información es correcta, se registra la habitación, se notifica al usuario respecto a la creación exitosa y se restablece el formulario para ingresar una nueva habitación.
Actualizar habitación	Los pasos que involucran este caso de uso son: 1. El usuario da clic en el botón "Editar" de una habitación que se muestra en la tabla de habitaciones. 2. La plataforma despliega el formulario para actualizar dicha habitación.

	3. El usuario modifica los campos que requiere y da clic en el botón “Guardar”. 4. La plataforma verifica la información ingresada: • Si los datos de la habitación son válidos, se actualiza la habitación y se da una notificación al usuario. Posteriormente, se redirige al usuario al caso de uso “Ver resumen”. • Si los datos no son válidos. Se informa al usuario sobre los campos incorrectos.
Eliminar habitación	El proceso de este caso de uso es: 1. El usuario da clic en el botón “Eliminar” de una habitación que se muestra en la tabla de habitaciones. 2. La plataforma pregunta al usuario si está seguro de que quiere eliminar la habitación: • Si el usuario da clic en “Aceptar”, se elimina la habitación y se notifica al usuario de la eliminación exitosa. • Si el usuario da clic en “Cancelar” no se elimina la habitación.
Consultar tipos de habitación	El proceso de este caso de uso es: 1. El usuario da clic en la opción “Consultar Tipos de habitación” desde el menú de la aplicación. 2. La plataforma despliega la información de los tipos de habitación registrados en forma de tabla.

	En esta tabla, el usuario puede buscar un tipo de habitación específico, ordenar las clientes según un campo, acceder al formulario de actualización de cliente y eliminar un cliente específico.
Crear tipo de habitación	Este caso de uso se basa en el siguiente procedimiento: 1. El usuario da clic en la opción “Registrar Clientes”. 2. La plataforma despliega el formulario para la creación de tipos de habitación. 3. El usuario ingresa toda la información correspondiente al tipo de habitación. 4. La plataforma verifica los datos ingresados por el usuario. • Si los datos no son correctos, se informa al usuario sobre el error. • Si la información es correcta, se registra el tipo de habitación, se notifica al usuario respecto a la creación exitosa y se restablece el formulario para ingresar un nuevo tipo de habitación.
Actualizar tipo de habitación	Los pasos que involucran este caso de uso son: 1. El usuario da clic en el botón “Editar” de un tipo de habitación que se muestra en la tabla de tipos de habitación. 2. La plataforma despliega el formulario para actualizar dicho tipo de habitación. 3. El usuario modifica los campos que requiere y da clic en el botón “Guardar”.

	4. La plataforma verifica la información ingresada: • Si los datos del cliente son válidos, se actualiza el tipo de habitación y se da una notificación al usuario. Posteriormente, se redirige al usuario al caso de uso “Ver resumen”. Si los datos no son válidos. Se informa al usuario sobre los campos incorrectos.
Eliminar tipo de habitación	El proceso de este caso de uso es: 1. El usuario da clic en el botón “Eliminar” de un tipo de habitación que se muestra en la tabla de tipos de habitación. 2. La plataforma pregunta al usuario si está seguro de que quiere eliminar el tipo de habitación. • Si el usuario da clic en “Aceptar”, se elimina el cliente y se elimina el tipo de habitación y se notifica al usuario de la eliminación exitosa. • Si el usuario da clic en “Cancelar” no se elimina el tipo de habitación.
Consultar usuarios	El proceso de este caso de uso es: 1. El administrador da clic en la opción “Administrar Usuarios” desde el menú de la aplicación. 2. La plataforma despliega la información de los usuarios registrados en forma de tabla. 3. En esta tabla, el administrador puede buscar un usuario específico, ordenar los usuarios según un campo, acceder

	al formulario de actualización de usuario y eliminar un usuario específico.
Crear usuario	Este caso de uso se basa en el siguiente procedimiento: 1. El administrador da clic en la opción “Administrar Usuarios” desde el menú de la aplicación. 2. La plataforma despliega la información de los usuarios registrados en forma de tabla y muestra una opción para crear usuarios. 3. El administrador da clic en el botón “Crear usuario”. 4. La plataforma despliega el formulario para la creación de usuarios. 5. El administrador ingresa toda la información correspondiente al usuario. 6. La plataforma verifica los datos ingresados por el administrador. • Si los datos no son correctos, se informa al administrador sobre el error. • Si la información es correcta, se registra el usuario, se notifica al administrador respecto a la creación exitosa y se envía las credenciales al nuevo usuario por correo electrónico.
Actualizar usuario	Los pasos que involucran este caso de uso son: 1. El administrador da clic en el botón “Editar” de un usuario que se muestra en la tabla de usuarios.

	2. La plataforma despliega el formulario para actualizar dicho usuario. 3. El administrador modifica los campos que requiere y da clic en el botón “Guardar”. 4. La plataforma verifica la información ingresada: • Si los datos del usuario son válidos, se actualiza el usuario y se da una notificación al administrador de la actualización exitosa. • Si los datos no son válidos. Se informa al administrador sobre los campos incorrectos.
Eliminar usuario	El proceso de este caso de uso es: 1. El administrador da clic en el botón “Eliminar” de una reserva que se muestra en la tabla de usuarios. 2. La plataforma pregunta al administrador si está seguro de que quiere eliminar el usuario. • Si el administrador da clic en “Aceptar”, se elimina el usuario y se notifica al administrador de la eliminación exitosa. • Si el administrador da clic en “Cancelar” no se elimina el usuario.
Reservar habitación	Este caso de uso es empleado por los cliente del hotel. Consiste en: 1. El cliente da clic en “Reservar Ahora”. 2. La aplicación despliega la vista para realizar las reservas. 3. El cliente selecciona las fechas de entrada y salida.

	4. La aplicación muestra los tipos de habitación disponibles según las fechas dadas. 5. El cliente selecciona un tipo de habitación que le interese. 6. La aplicación muestra la información del tipo de habitación seleccionado y un formulario para agregarlo a la reserva. 7. Si el cliente desea reservar una habitación del tipo de habitación seleccionado, ingresa la cantidad de niños y adultos que asistirán y da clic en "Agregar a la reserva". 8. La aplicación guarda la información ingresada y redirige al cliente hacia la vista que muestra todos los tipos de habitación disponibles.
Pagar reserva	Para este caso de uso, se realizan los siguientes pasos: 1. Cuando el cliente seleccionó las habitaciones disponibles, da clic en "Pagar reserva". 2. La aplicación muestra el formulario para realizar el pago de la reserva. Este formulario contiene campos para ingresar los datos del cliente y muestra la información de la reserva y el total a pagar. 3. El cliente ingresa sus datos y da clic en "Pagar reserva". • Si los datos son inválidos, se notifica al cliente sobre los errores. • Si los datos del cliente son correctos, se continua con el flujo de pago.

	4. La aplicación empieza el proceso de pago y registra los datos del cliente y la reserva.

Tabla 1. Casos de uso de la aplicación.

Arquitectura de la aplicación

Arquitectura general de la aplicación.

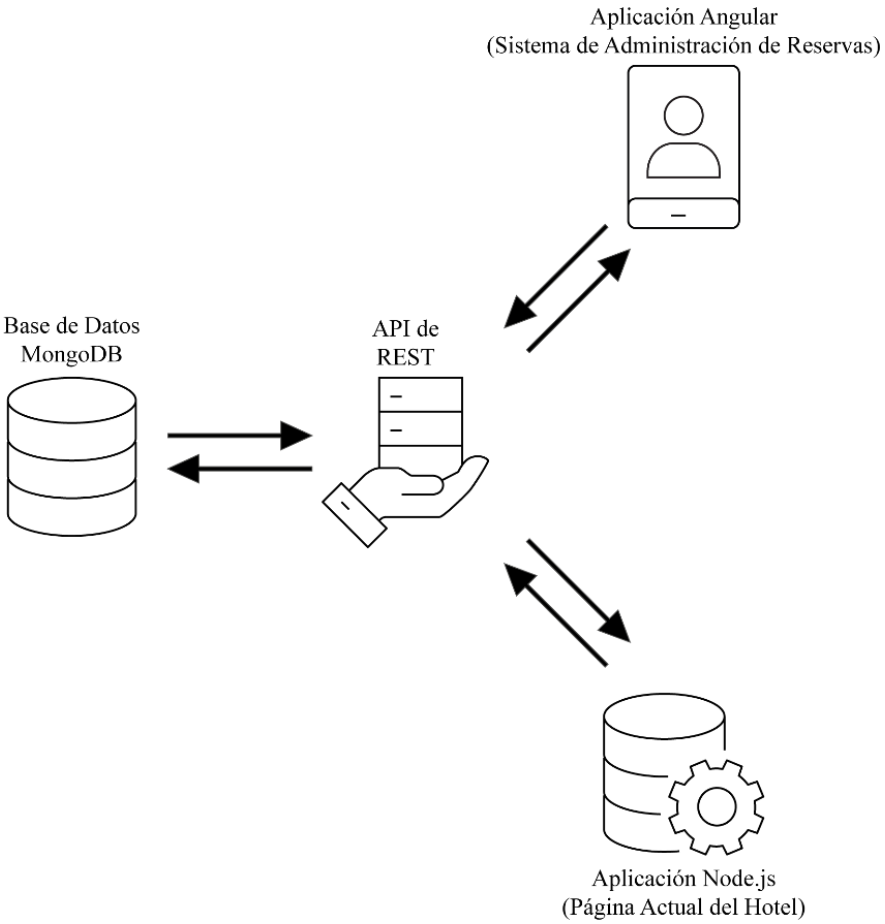

Figura 2. Arquitectura general de la aplicación.

La aplicación de manera global se construyó mediante una aproximación común para el diseño de aplicaciones MEAN. Este patrón consiste en tener una API de REST que alimente a una aplicación de página única o a una aplicación de Node.js, es decir, la API de REST actúa como un puente de comunicación entre la base de datos y las aplicación. A continuación, se detallan cada una de las partes que conforman la arquitectura del proyecto:

Base de datos.

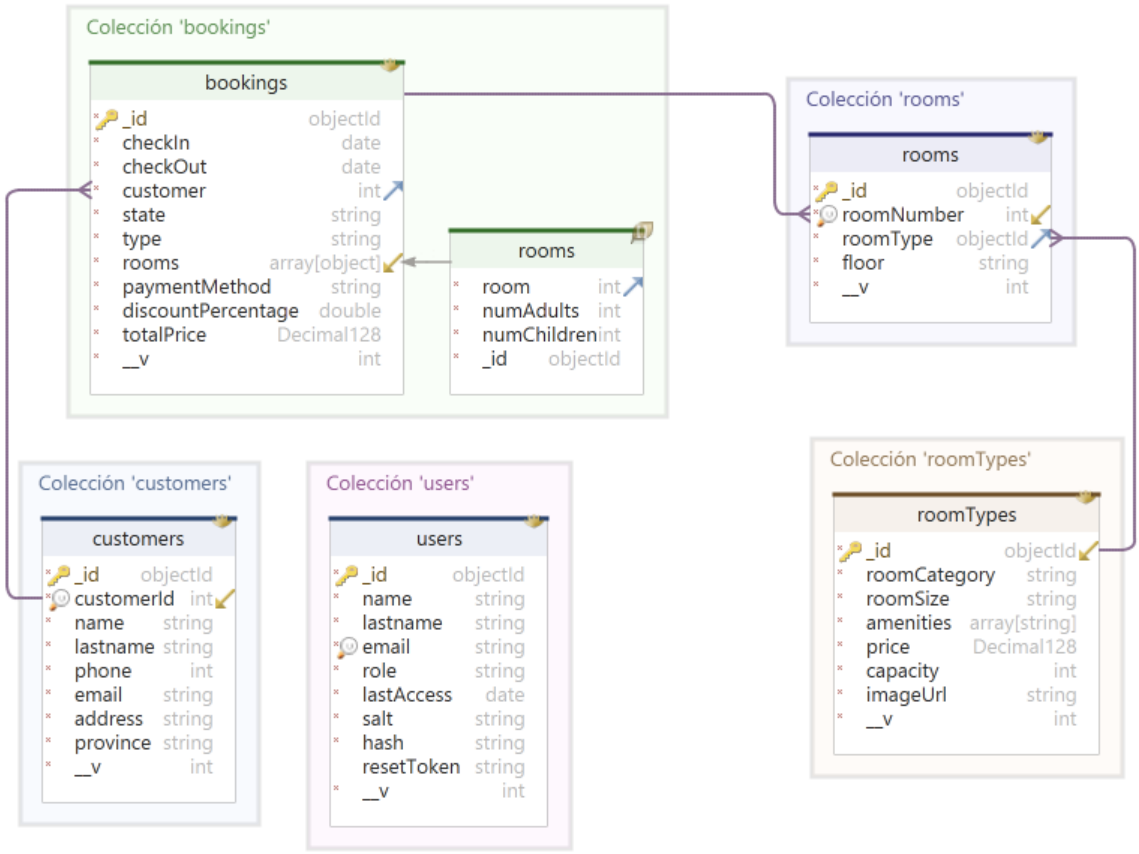

Figura 3. Diagrama entidad-relación de la base de datos.

La recomendación de MongoDB para el diseño de bases de datos no relacionales es que el modelo se ajuste a las necesidades de la aplicación. En este sentido, se consideró los siguientes requerimientos para la base de datos:

Almacenar la información.

Proporcionar un buen rendimiento de consulta.

Emplear una cantidad razonable de hardware (MongoDB Schema Design Best Practices | MongoDB, 2022).

En base a ello, se creó un total un total de cinco colecciones:

- Colección “users”: Representa a un usuario del sistema. Contiene información respecto al nombre, apellido, correo electrónico, último acceso y la contraseña del usuario.

- Colección “customers”: Contiene los datos sobre los clientes del hotel. Sus campos son la cédula, nombre, apellido, teléfono, correo electrónico, dirección y provincia del cliente.
- Colección “roomTypes”: Se emplea para almacenar información respecto a los tipos de habitación que tiene el hotel. Cada documento de esta colección posee una categoría de habitación, la cual puede ser “estándar”, “lujo” o “superior”, y un tamaño de habitación, que puede ser “matrimonial”, “triple” o “cuádruple”. Además, tiene campos respecto al precio del tipo de habitación, su capacidad máxima y las comodidades que tiene.
- Colección “rooms”: Se refiere a las habitaciones del hotel. Cada documento de esta colección tiene un número y tipo de habitación, y un campo referente al piso en el que se encuentra, que puede ser “planta baja”, “primer piso” y “segundo piso”.
- Colección “bookings”: Representa las reservas del hotel. Sus campos son, las fechas de entrada y salida, la cédula del cliente o pasaporte que realizó la reserva, el porcentaje de descuento y el precio total a pagar. El estado de la reserva, el cual es “en progreso”, “completada” o “cancelada”, mientras que para el tipo de reserva existen las opciones “en línea” y “presencial”. En el caso del método de pago; este puede ser “Efectivo” o “PayPhone”. Respecto al campo “habitaciones”, es un arreglo de objetos, donde cada objeto contiene datos respecto al número de habitación reservada y la cantidad de adultos y niños que la ocuparán.

Cabe recalcar que a pesar de que MongoDB brinda gran libertad en el modelado de bases de datos, también considera una serie de anti-patrones a evitar para prevenir problemas en relación con la eficiencia de las consultas y la integridad de los datos. Estos anti-patrones son:

- Arreglos masivos: Almacenar arreglos ilimitados en los documentos. Los arreglos masivos ocasionan problemas de memoria, ya que cada documento en MongoDB tiene un tamaño máximo de 16 MB. La única colección que contiene arreglos es la colección de reservas en su campo habitaciones. Este campo, como máximo puede tener un total de 24 elementos, lo cual ocurriría en el caso de que una reserva incluya todas las habitaciones del hotel.
- Número masivo de colecciones: Almacenar un gran número de colecciones, especialmente si no son necesarias. El esquema actual solo cuenta con cinco colecciones, las cuales son ampliamente utilizadas en toda la aplicación.
- Documentos masivos: Almacenar grandes cantidades de datos juntos en un documento si no se accede a esos datos frecuentemente juntos (*A Summary of Schema Design Anti-Patterns and How to Spot Them | MongoDB*, 2022). Para prevenir este conflicto, cada colección solo contiene información que debe ser accedida de manera conjunta.

API de REST.

REST se refiere a transferencia de estado representacional. Se trata de un estilo arquitectónica, el cual no posee estado de sesión (Holmes & Harber, 2019). Es decir, no mantiene idea sobre el historial actual del usuario.

Por otra parte, API es la abreviatura en inglés para interfaz de programa aplicación. Este tipo de interfaces se utilizan con el fin de permitir la comunicación entre aplicaciones. Para aplicaciones de tipo web, una API está conformada por un conjunto de URLs que responden con datos al ser llamadas de forma correcta y con la información correcta. En base a estos dos términos, una API de REST es una interfaz sin estado de sesión para la aplicación. En el stack MEAN, la API de REST se emplea como una interfaz para la base

de datos, lo cual permite que otras aplicaciones, como la aplicación de página única desarrollada con Angular, trabaje con los datos (Holmes & Harber, 2019).

La estructura de la API de REST del proyecto posee tres elementos que permiten el consumo de datos para el resto de las aplicaciones. En primer lugar, se encuentran los modelos u esquemas. Los modelos se desarrollaron con Mongoose, una capa ODM (Object Document Mapping) construida sobre el controlador de Node para MongoDB (Cherry et al., 2022). Su propósito es establecer la estructura de cada uno de los documentos dentro de una colección.

Por otro lado, los controladores de la API son los encargados de manejar las operaciones de creación, lectura, actualización y eliminación de documento de las distintas colecciones. Cada uno de los controladores devuelve una respuesta en formato JSON (JavaScript Object Notation). Finalmente, el ruteador tiene como propósito indicar en qué ruta y en qué operación se empleará un controlador. En este caso, la API de REST del proyecto maneja las operaciones CRUD sobre las colecciones de usuarios, reservas, tipos de habitación, habitaciones, clientes y reserva, así como, la autenticación de usuarios.

URI	Método	API
/login	POST	*Inicio de sesión:* Autenticación de usuarios
/forgotPassword	PUT	*Contraseña olvidada:* Genera enlace para restablecer contraseña para un usuario.
/resetPassword/:token	PUT	*Restablecimiento de contraseña:* Actualiza la contraseña de un usuario.
/bookings	POST	*Crear reserva:* Crea una reserva con las fechas de ingreso y salida, el cliente, tipo y estado de la reserva, porcentaje de descuento,

		total a pagar, método de pago y habitaciones ocupadas.
/bookings	GET	*Listar reservas:* Lista las reservas definidas.
/bookings/:bookingid	GET	*Mostrar reserva:* Muestra la información de una reserva específica.
/bookings/:bookingid	PUT	*Actualizar reserva:* Actualiza los valores de ciertos parámetros de la reserva.
/bookings/:bookingid	DELETE	*Eliminar reserva:* Elimina la reserva especificada.
/occupiedRooms/:checkIn/:checkOut	GET	*Listar habitaciones ocupadas:* Lista las habitaciones reservadas entre dos fechas dadas.
/customers	POST	*Crear cliente:* Crea un cliente con su cédula, nombre, apellido, teléfono, correo electrónico, dirección y provincia.
/customers	GET	*Listar clientes:* Lista los clientes definidos.
/customers/:customerid	GET	*Mostrar cliente:* Muestra la información de un cliente específico.
/customers/:customerid	PUT	*Actualizar cliente:* Actualiza los valores de ciertos parámetros de un cliente.
/customers/:customerid	DELETE	*Eliminar cliente:* Elimina el cliente especificado.
/rooms	POST	*Crear habitación:* Crea una habitación su número de habitación, tipo de habitación, y el piso en la que se encuentra.

/rooms	GET	*Listar habitaciones:* Lista las habitaciones definidas.
/completeRoom/:roomid	GET	*Mostrar habitación con tipo de habitación:* Muestra la información de una habitación específica con los detalles de su tipo de habitación.
/rooms/:roomid	GET	*Mostrar habitación:* Muestra la información de una habitación específica.
/rooms/:roomid	PUT	*Actualizar habitación:* Actualiza los valores de ciertos parámetros de una habitación.
/rooms/:roomid	DELETE	*Eliminar habitación:* Elimina la habitación especificada.
/roomTypes	POST	*Crear tipo de habitación:* Crea un tipo de habitación con su categoría, tamaño, comodidades, precio y capacidad máxima.
/roomTypes	GET	*Listar tipos de habitación:* Lista los tipos de habitación definidos.
/roomTypes/:roomTypeid	GET	*Mostrar tipo de habitación:* Muestra la información de un tipo de habitación específico.
/roomTypes/:roomTypeid	PUT	*Actualizar tipo de habitación:* Actualiza los valores de ciertos parámetros del tipo de habitación.
/roomTypes/:roomTypeid	DELETE	*Eliminar tipo de habitación:* Elimina el tipo de habitación especificado.
/users	POST	*Crear usuario:* Crea un usuario con su nombre, apellido y correo electrónico. La contraseña se

		almacena en los valores de hash y salt. Al crear un nuevo usuario, se envía su información para acceder al sistema a su correo electrónico.
/users	GET	*Listar usuarios:* Lista los usuarios definidos.
/users/:userid	GET	*Mostrar usuario:* Muestra la información de un usuario específico.
/users/:userid	PUT	*Actualizar usuario:* Actualiza los valores de ciertos parámetros de un usuario.
/users/:userid	DELETE	*Eliminar usuario:* Elimina el usuario especificado.
/customerBooking	POST	*Registrar cliente y reserva:* Crea una reserva y, crea o actualiza un cliente.

Tabla 2. URI y métodos de la API de REST.

API	**Parámetros de solicitud**		**Parámetros de respuesta**	
	Nombre	**Tipo**	**Nombre**	**Tipo**
Inicio de sesión	email	String	token	String
	password	String		
Olvidar de contraseña	email	String	_id	String
			name	String
			lastname	String
			email	String
			role	String
			hash	String
			salt	String
			resetToken	String
Restablecimiento de contraseña	token	String	name	String
			lastname	String
	password	String	email	String
			role	String
	repeatedPassword	String	hash	String
			salt	String

Crear reserva	_id	String	_id	String
	checkIn	Date	checkIn	Date
	checkOut	Date	checkOut	Date
	customer	Number	customer	Number
	state	String	state	String
	type	String	type	String
	rooms	array	rooms	array
	paymentMethod	String	paymentMethod	String
	discountPercentage	Number	discountPercentage	Number
	totalPrice	Number	totalPrice	Number
Listar reservas	none	none	bookingResult	array
Mostrar reserva	bookingId	String	id	String
			checkIn	Date
			checkOut	Date
			customer	Number
			state	String
			type	String
			rooms	array
			paymentMethod	String
			discountPercentage	Number
			totalPrice	Number
Actualizar reserva	_id	String	_id	String
	checkIn	Date	checkIn	Date
	checkOut	Date	checkOut	Date
	customer	int	customer	Number
	state	String	state	String
	type	String	type	String
	rooms	array	rooms	array
	paymentMethod	String	paymentMethod	String
	discountPercentage	Number	discountPercentage	Number
	totalPrice	Number	totalPrice	Number
Eliminar reserva	bookingId	String	none	none
Listar habitaciones ocupadas	checkIn	String	ocuppiedRooms	array
	checkOut	String		
Listar clientes	none	none	customerResult	array
Mostrar cliente	customerId	String	id	String
			customerId	Number
			name	String
			lastname	String
			phone	Number
			email	String
			address	String
			province	String
Actualizar cliente	_id	String	_id	String
	customerId	Number	customerId	Number
	name	String	name	String
	lastname	String	lastname	String
	phone	Number	phone	Number

	email	String	email	String
	address	String	address	String
	province	String	province	String
	customerId	Number	customerId	Number
Eliminar cliente	customerId	String	none	none
Crear habitación	_id	String	_id	String
	roomNumber	Number	roomNumber	Number
	roomType	ObjectId	roomType	ObjectId
	floor	String	floor	String
Listar habitaciones	none	none	roomResult	array
Mostrar habitación	roomId	String	_id	String
			roomNumber	Number
			roomType	ObjectId
			floor	String
Actualizar habitación	_id	String	_id	String
	roomNumber	Number	roomNumber	Number
	roomType	ObjectId	roomType	ObjectId
	floor	String	floor	String
Eliminar habitación	roomId	String	none	none
Crear tipo de habitación	_id	String	_id	String
	roomCategory	Number	roomNumber	Number
	roomSize	ObjectId	roomType	ObjectId
	amenities	String	floor	String
	price	Number	price	Number
	capacity	Number	capacity	Number
Listar tipos de habitación	none	none	roomTypeResult	array
Mostrar tipo de habitación	roomTypeId	String	_id	String
			roomNumber	Number
			roomType	ObjectId
			floor	String
			price	Number
			capacity	Number
Actualizar tipo de habitación	_id	String	_id	String
	roomNumber	Number	roomNumber	Number
	roomType	ObjectId	roomType	ObjectId
	floor	String	floor	String
	price	Number	price	Number
	capacity	Number	capacity	Number
Eliminar tipo de habitación	roomTypeId	String	none	none
Crear usuario	_id	String	_id	String
	name	String	name	String
	email	String	email	String
	role	String	role	String
	lastAccess	String	lastAccess	String
	password	String	hash	String

			salt	String
Listar usuarios	none	none	roomNumber	Number
Mostrar usuario	userId	String	_id	String
			name	String
			email	String
			role	String
			lastAccess	String
			hash	String
			salt	String
Actualizar tipo de habitación	_id	String	_id	String
	name	String	name	String
	email	String	email	String
	role	String	role	String
	lastAccess	String	lastAccess	String
	password	String	price	Number
			capacity	Number
Eliminar tipo de habitación	userId	String	none	none
Registrar cliente y reserva	checkIn	Date	checkIn	Date
	checkOut	Date	checkOut	Date
	customer	Number	customer	Number
	state	String	state	String
	type	String	type	String
	rooms	array	rooms	array
	paymentMethod	String	paymentMethod	String
	discountPercentage	Number	discountPercentage	Number
	totalPrice	Number	totalPrice	Number
	checkIn	Date	checkIn	Date
	checkOut	Date	checkOut	Date
	customer	Number	customer	Number
	state	String	state	String
	type	String	type	String
	rooms	array	rooms	array
	paymentMethod	String	paymentMethod	String
	discountPercentage	Number	discountPercentage	Number
	totalPrice	Number	totalPrice	Number

Tabla 3. Parámetros de solicitud y respuesta de la API.

Aplicación Node.js (Página actual del hotel).

En esta sección se incorporó la página actual del hotel a la aplicación. Además, se basa en el patrón de arquitectura MVC. La arquitectura Modelo-Vista-Controlador (MVC) se basa en dos principios fundamentales en el diseño de software: alta reutilización y bajo acoplamiento. Su arquitectura se compone del modelo, el cual se refiere al modelo de datos, la vista, que es la interfaz hombre-máquina, y el controlador, el cual es el

responsable de la comunicación entre la vista y el modelo (Zhao, 2022). En el caso de Node.js y Express.js; las vistas se desarrollaron con plantillas PUG. Además, se incluyó un ruteador, el cual indica qué controlador maneja cada ruta. Para el alcance del presente proyecto, esta sección no hace uso de un modelo de datos a excepción del componente Angular que incorpora para realizar reservar. Por lo tanto, los controladores se encargan únicamente de renderizar las vistas ante una solicitud.

Aplicación Angular (Sistema de Administración de Reservas).

Es la parte clave para los administradores del hotel. Contiene toda la funcionalidad referente a la creación, actualización, eliminación y lectura de reservas, habitaciones, tipos de habitaciones y clientes, así como el manejo y autenticación de usuarios.

Debido a que el sistema de administración de reservas requiere una alta interactividad con datos, se decidió construirlo con Angular. Debido a la naturaleza de este marco de trabajo, esta sección es una aplicación de una sola página, es decir, toda su lógica, entrega de plantillas, procesamiento de datos y flujo de usuarios se maneja en el navegador (Holmes & Harber, 2019).

Las aplicaciones Angular se desarrollan mediante dos conceptos importantes, los módulos y los componentes. Los componentes son considerados como el bloque de construcción principal de las aplicaciones de Angular (*Angular - Angular components overview* 2022), mientras que los módulos están constituidos por un conjunto de componentes que trabajan en conjunto.

Finalmente, otro bloque esencial de la aplicación son los servicios. Los servicios constituyen una clasificación amplia, la cual implica cualquier función, característica o valor que requiera una aplicación. En sí, un servicio de Angular se trata de una clase con un propósito específico. (*Introduction to services and dependency injection* 2022). Para

el sistema de administración de reservas se implementó un total de siete servicios. A continuación, se listan los servicios y componentes desarrollados:

Servicio	Descripción
AuthenticationService	Su objetivo es hacer uso de la API de autenticación en la aplicación. Sus funciones son facilitar el inicio y cierre de sesión, verificar si un usuario ha iniciado sesión y obtener su información, almacenar y eliminar el token de un usuario.
DataTableService	Se utiliza para crear tablas inteligentes en la aplicación. Para ello, incorpora el plugin DataTables. Esta tablas tienen funcionalidades como reordenar columnas, ordenar filas de forma alfabética, realizar búsquedas y ocultar columnas según el tamaño de la pantalla.
DateService	Se utiliza para dar el formato aa-mm-dd a una fecha.
NotificationService	Emplea el módulo SweetAlert. Su función es facilitar la creación de alertas dentro de la aplicación.
PaymentService	Este servicio utiliza la API de PayPhone para realizar pagos. Contiene métodos para hacer un pago y verificar el estado de una transacción.
Storage	Maneja el almacenamiento local del navegador.
SWDataService	Es un servicio de datos. Incorpora el módulo HttpClient para realizar solicitudes HTTP a la API de REST de la aplicación.

Tabla 4. Servicios implementados en la aplicación Angular.

Componente	Descripción
AuthenticacionLayoutComponent	Es un contenedor para los formularios de inicio de sesión, solicitar restablecimiento de contraseña y restablecer contraseña.
BookingManagerComponent	Formulario para la creación y actualización de reservas.
BookingReaderComponent	Tabla que muestra las reservas registradas. Permite eliminar o acceder al formulario de actualización de una reserva.
CustomerManagerComponent	Formulario para la creación y actualización de clientes.
CustomerReaderComponent	Tabla que muestra los clientes registrados. Permite eliminar o acceder al formulario de actualización de un cliente.
DashboardLayoutComponent	Contenedor para los distintos componentes del dashboard. Contiene
ForgotPasswordFormComponent	Formulario para solicitar restablecimiento de contraseña.
FrameworkComponent	Es el contenedor para todos los componentes de la aplicación. En su HTML tiene la etiqueta “router-outlet”, la cual le dice al enrutador dónde mostrar un componente.
LoginFormComponent	Formulario para inicio de sesión.
ProfileManagerComponent	Formulario para editar la información del usuario que inició sesión.

ResetPasswordFormComponent	Formulario para restablecer contraseña.
RoomManagerComponent	Formulario para la creación y la actualización de habitaciones.
RoomReaderComponent	Tabla que muestra las habitaciones registradas. Permite eliminar o acceder al formulario de actualización de una habitación.
RoomTypeManagerComponent	Formulario para la creación y actualización de tipos de habitación.
RoomTypeReaderComponent	Tabla que muestra los tipos de habitación registrados. Permite eliminar o acceder al formulario de actualización de un tipo de habitación.
SummaryComponent	Muestra un resumen de la información del hotel, es decir, muestra las tablas de reservas, habitaciones y clientes. Además, presenta los gráficos de estadísticas del negocio.
UserManagerComponent	Tabla que muestra los usuarios registrados. Permite eliminar o acceder al formulario de actualización de un usuario. Este componente solo está disponible para usuarios de tipo administrador.
AvailableRoomsComponent	Muestras los tipos de habitación disponibles dado una fecha de entrada y una fecha de salida.
PaymentComponent	Su función es permitir a un cliente realizar el pago de una reserva. Muestra la información de la

	reserva, como el total a pagar y contiene un formulario para que el cliente ingrese sus datos.
RoomInfoComponent	Muestra la información de un tipo de habitación. Además, incluye un formulario para agregar una habitación de dicho tipo a la reserva y establecer la cantidad de niños y adultos que asistirán.

Tabla 5. Componentes de la aplicación Angular.

Sistema de Autenticación de usuarios.

El sistema de autenticación de usuarios es una de las piezas clave para el sistema de administración de reservas. Permite el acceso a usuarios autorizados y control de perfiles de usuario. A continuación, se menciona los elementos principales del sistema:

Esquema Usuarios.

Define la estructura de la información de un usuario dentro del sistema. Implementa métodos para almacenar y verificar la contraseña del usuario. Para guardar la información de la contraseña de un usuario, se empleó una encriptación unidireccional mediante el algoritmo PBKDF2 y el uso de hashes y salts. El proceso de encriptación implementado consiste en:

1. Crear el salt, una cadena aleatoria de 16 bytes generada por la aplicación para cada usuario.
2. Combinar la contraseña del usuario con el salt.
3. Calcular el hash de la contraseña combinada con el salt.

Para verificar que una contraseña ingresada corresponda a un usuario dado se realiza un procedimiento similar. En esta situación, se combina la contraseña ingresada con el salt almacenado. Posteriormente se calcula el hash de esta combinación y se lo compara con

el hash almacenado. Si ambos valores de hash coinciden, entonces la contraseña ingresada es correcta.

Además, el esquema también cuenta con métodos para generar contraseñas aleatorias y tokens web de JSON. Por otra parte, este esquema también define los roles de cada usuario. Actualmente, la aplicación cuenta con dos perfiles, administrador y editor.

Passport.

Passport.js es un módulo de autenticación para Node.js. Su fortaleza principal radica en que se adapta a una serie de métodos de autenticación, llamados estrategias. Algunas de las estrategias que ofrece Passport.js son:

- OAuth.
- Facebook.
- Twitter.
- Nombre de usuario y contraseña local (Hanson).

Para el presente proyecto, la API de REST es la encargada de manejar Passport. Por consiguiente, la API cuenta con un archivo de configuración para Passport. Aquí se define la estrategia empleada para realizar la autenticación de usuarios en la aplicación, la cual requiere del correo electrónico y la contraseña de un usuario.

JSON Web Token.

Los tokens web JSON (JWT) son un estándar de la industria abierto (RFC 7519). Se utilizan principalmente para realizar autorización e intercambio de información. Establecen una manera compacta para la transmisión segura de información entre las partes como un objeto JSON. La información enviada puede ser verificada y considerarse como confiable, ya que está firmada digitalmente. Los token web JSON se pueden firmar con un par de claves, pública y privada, mediante ECDSA o RSA, o empleando un secreto mediante un algoritmo HMAC (Auth0). Su estructura consiste en tres partes:

- Encabezado (header): Contiene información del tipo de token, que es JWT, y el algoritmo para firmar y verificar la firma. Para la aplicación se utilizó el algoritmo HMAC SHA256, pero también se puede aplicar RSA.
- Carga útil (payload): Es el cuerpo del token. Contiene las reclamaciones, que son declaraciones sobre una entidad, como un usuario, e información adicional, como la fecha de expiración del token.
- Firma: Es un hash encriptado del encabezado y la carga útil. Para obtener este hash se emplea un secreto que solo es conocido por el servidor y nunca debe ser revelado al público. El propósito de la firma es verificar que el mensaje no fue modificado en el camino.

Dentro de la aplicación de una sola página, es decir, el sistema de administración de reservas es el token web de JSON el que facilita el manejo de sesiones de usuarios. Para almacenar este token se hace uso del almacenamiento local de navegador, debido a que este tipo de almacenamiento está diseñado para aplicaciones del lado del cliente. De este modo, el token permanece en el navegador. A continuación, se explica cómo se acceden a los distintos recursos con el token generado en la aplicación:

Acceso a rutas.

Para acceder a las rutas que requieren que un usuario haya iniciado una sesión en el sistema de administración de reservas, se emplea el servicio “AuthenticationService”. Este servicio cuenta con el método “isLoggedIn”. Esta función se encarga de verificar si se encuentra una sesión activa. Para ello, comprueba la existencia del token en el almacenamiento local y revisa si el token no ha expirado. Si una de estas condiciones no se cumple, no se permite el acceso a la ruta y se redirige al usuario a la vista de inicio de sesión.

Acceso a recursos de la API de REST.

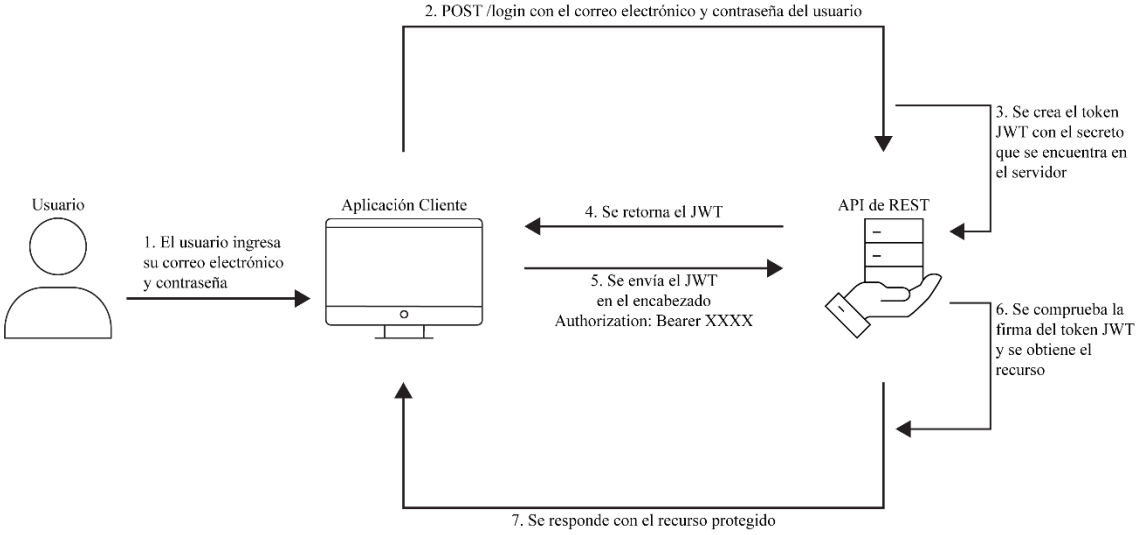

Figura 4. Flujo del JWT para acceder a los recursos de la API de REST.

Los recursos de la API de REST están protegidos ante accesos no autorizados. Para que un usuario acceda a estos recursos, se envía el token en el encabezado de autorización en cada solicitud HTTP a la API de REST.

Envío de correos electrónicos a usuarios y Nodemailer.

Un proceso trascendental al momento de crear un usuario o solicitar el restablecimiento de contraseña dentro de la aplicación, es notificar al usuario sobre su solicitud. Con esto en mente, se creó en Gmail la cuenta seaworldhotel.ec@gmail.com para la aplicación. Además, se solicitó una contraseña de aplicación, para conceder permiso al sistema de administración de reservas a la cuenta de Google creada.

Para manejar la cuenta de Gmail en la aplicación, se empleó Nodemailer, un módulo de Node.js que facilita el envío de correos electrónicos de forma sencilla. Nodemailer define el concepto de transportador, el cual es un objeto capaz de enviar correos electrónicos (Reinman). En este objeto se establecen los siguientes parámetros:

- Host: El nombre de host o dirección IP a conectarse. Para usar Gmail, este parámetro se configura con el valor “smtp.gmail.com”.
- Port: El puerto a conectarse. Se utiliza el puerto 465 para SMTP.

- Secure: Se establece como verdadero para emplear TLS al momento de conectarse al servidor.
- Auth: Define los datos de autenticación.
    - User: El nombre de usuario. En este caso se emplea el correo electrónico seaworldhotel.ec@gmail.com.
    - Pass: La contraseña del usuario. Se establece este parámetro con el valor de la contraseña de aplicación generada para la cuenta de Gmail.

La API de REST es la encargada de utilizar el transportador creado para el envío de correos electrónicos. La API emplea este transportador en dos procesos:

- Creación de usuarios: Al momento de crear un nuevo usuario, se envía sus credenciales de acceso a su correo electrónico.
- Solicitar restablecimiento de contraseña: Cuando un usuario solicita cambiar su contraseña, se envía a su correo electrónico el enlace para realizar este procedimiento.

Guardias.

Los guardias son una de las funcionalidades propias de Angular. Son middlewares, los cuales se ejecutan antes de que se cargue una ruta. Para el sistema de administración de reservas se desarrolló el guardia "HasRoleGuard". Este guardia implementa la interfaz "CanActivate", la cual define la lógica para decidir si una ruta puede ser activada. Si el guardia retorna verdadero, la navegación continua. Caso contrario, se cancela la navegación (*CanActivate* 2022).

El objetivo del guardia "HasRoleGuard" es controlar el acceso a las rutas de la aplicación según el rol del usuario, administrador o editor. Para ello, cada ruta contiene información respecto a los roles de usuario permitido. Con estos datos, el guardia "HasRoleGuard" obtiene los datos del usuario actual del token guardado en el almacenamiento local y

verifica si el rol del usuario actual se encuentra en los roles permitidos por la ruta. Si el rol sí está permitido, se da acceso a la ruta, de lo contrario, se redirige al usuario a la vista de "Ver resumen". Actualmente, todas las rutas del sistema de administración de reservas son accesibles ambos perfiles a excepción de la ruta para administrar usuarios. El acceso a esta ruta solo está permitido para usuarios de tipo administrador.

Sistema de pagos.

Implementación de la API de PayPhone.

Los pagos de las reservas en la aplicación se realizan mediante la API de PayPhone. Se escogió esta plataforma para el proyecto debido a su fácil implementación y porque permite pagos con:

- Tarjetas de regalo.
- Saldo PayPhone.
- Tarjetas de débito, MasterCard y Visa de cualquier país del mundo.
- Tarjetas de crédito, MasterCard y Visa de cualquier país del mundo.

Para utilizar esta API se realizó los siguientes pasos:

1. Se registró el hotel en PayPhone Business.
2. Se creó un perfil de desarrollador en PayPhone Developer.
3. Se creó una aplicación de tipo API en PayPhone Developer.
4. Con la aplicación configurada, se obtuvo un token de autenticación para emplear la aplicación creada en PayPhone Developer dentro del proyecto (*API PayPhone | Cobro por app* 2022).
5. Posteriormente se implementó las solicitudes a la API de PayPhone en el servicio "PaymentService". Las solicitudes a esta API se realizan a la url https://pay.payphonetodoesposible.com/api/sale.

Funcionamiento del sistema de pagos.

La lógica del sistema de pagos se encuentra implementada en el componente "PaymentComponent", el cual obtiene la información de la reserva desde el almacenamiento local. Este componente a su vez utiliza el servicio "PaymentService". El servicio "PaymentService" contiene los siguientes métodos:

- doPayment: Realiza un POST a la API de PayPhone para solicitar el pago. En el encabezado de autorización se envía el token de autorización de PayPhone, mientras que en el cuerpo de la solicitud se envían los siguientes datos:
    - Amount: Valor total a cobrar.
    - AmountWithoutTax: Valor a cobrar sin impuesto
    - clientTransactionId: Identificador único de la transacción. Para generar este identificador se combina el tiempo en el que solicitó realizar la transacción y la cédula del cliente.
    - phoneNumber: Número de teléfono del cliente.
    - countryCode: Código del país del cliente.
    - email: Correo electrónico del cliente.
    - documentId: Número de identificación del cliente (*API PayPhone | Cobro por app* 2022).

  Tras realizar la solicitud POST a la API de PayPhone, se obtiene un identificador de transacción "transactionId", el cual representa el estado de la transacción que puede ser: Pendiente, Aprobado o Cancelado.
- getTransactionState: Realiza un GET a la API de PayPhone para solicitar el pago. En el encabezado de autorización se envía el token de autorización de PayPhone. Además, se envía el identificador de la transacción. Con ello, este método permite conocer el estado de la transacción.

Cuando un cliente entra al formulario para pagar su reserva, la aplicación se encarga de:

1. Obtener la información de las habitaciones reservadas, la cantidad de niños y de adultos que se hospedarán en cada habitación y de las fechas de entrada y salida de la reserva del almacenamiento sesión. Este tipo de almacenamiento difiere del almacenamiento local, en el sentido de que se borra al cerrarse la pestaña de navegador en la que está abierta la aplicación.
2. Con los datos de las habitaciones seleccionadas, solicita a la API de REST los datos respecto a precios, para calcular el total a pagar en base a la cantidad de personas. Cabe recalcar que los niños pagan la mitad del precio de una habitación.
3. Muestra al cliente un resumen de su reserva y un formulario para que ingrese sus datos.

El flujo para realizar el pago de una reserva después de que el cliente revisa los datos de su reserva, ingresa sus datos y el cliente da clic en "Pagar reserva", consta de los siguientes pasos:

1. Se llama al método "doPayment" y se obtiene el identificador de transacción.
2. Se muestra al cliente una notificación para confirma o cancelar el pago.
3. Llega una notificación de PayPhone al celular del cliente para realizar el pago de la reserva.
4. El cliente realiza el pago en la aplicación de PayPhone.
5. El cliente da clic en "Pagar".
    - Si la transacción aún está pendiente, se informa al cliente sobre el estado de la transacción.
    - Si la transacción es cancelada desde PayPhone o desde la aplicación, se cancela el proceso de pago de reserva.
    - Si se aprueba la transacción, se registran los datos del cliente y la reserva en la base de datos. Si los datos del cliente ya están registrados en el

sistema, se actualiza su información, de lo contrario, se crea un nuevo documento en la base de datos. Posteriormente, se notifica al cliente sobre el registro exitoso de su reserva.

6. Se elimina la información de la reserva del almacenamiento de sesión.

CONCLUSIONES

Respecto a los resultados obtenidos, cabe mencionar que se cumplió con éxito los objetivos propuestos. Es decir, se desarrolló una aplicación capaz de satisfacer los requerimientos y que se acople a la imagen de marca del Hotel Sea World. Por lo tanto, esta aplicación a largo plazo será clave para cumplir la visión de este negocio familiar, la cual es posicionarse como uno de los mejores hoteles a nivel nacional como internacional mediante el uso de la tecnología.

A nivel de desarrollo se evidenció las ventajas del stack MEAN. Gracias a que todo este conjunto de tecnologías se fundamenta en el lenguaje de programación JavaScript, se facilitó la implementación del proyecto y consecuentemente este tuvo un mayor alcance. Realizar una aplicación de la misma naturaleza y en el mismo plazo de tiempo con otras tecnologías, como el stack LAMP (Linux, Apache, MySQL, PHP), representaría una mayor dificultad para un programador, ya que este requeriría conocer varios lenguajes para alcanzar los mismos resultados.

En base a la experiencia de usuario, el uso del stack MEAN también brinda beneficios. En específico, al incorporar el marco de trabajo Angular, es ideal para aplicaciones que son ricas en datos. Tal es el caso del sistema de administración de reservas implementado, el cual requiere mostrar varios datos como reservas, clientes y habitaciones. Por lo que al ser una aplicación de una sola página gracias a Angular, hace que el proceso de creación, eliminación, actualización y lectura de datos, y sobre todo, el flujo de navegación sea cómodo y rápido para los usuarios.

Finalmente, la metodología de trabajo empleada en el proyecto también fue crucial para la ejecución exitosa de sus distintos entregables. Este hecho se debe a que permitió tener una noción clara de los requerimientos de la aplicación, que a su vez, permitió planificar un ciclo de vida adecuado para cada uno de los entregables. De esta manera, fue posible que la aplicación desarrollada sea de calidad y se ajuste a las necesidades del hotel.

Trabajo a futuro.

La aplicación desarrollada para el proyecto cumple con las expectativas de los administradores del hotel. Por otra parte, ya se cuenta con una versión de producción de esta. Sin embargo, todavía no está lista para ser utilizada por el público. A partir de este momento, la aplicación entrará en un período de prueba, donde se revisará todas sus funcionalidades y se realizará cualquier cambio menor que soliciten los administradores. Se estima que este ciclo de prueba tenga una duración de cinco meses.

Por otra parte, a futuro se plantea agregar nuevas funciones para la aplicación. En el caso de la sección de presentación de contenido a clientes, se propone una optimización de los recursos que emplea. Esto se debe a que la página original del hotel fue construida en base a una plantilla, la cual utiliza varios archivos de JavaScript, lo que afecta el tiempo de carga de las páginas. Además, se planea crear un nuevo rol en la aplicación, que estaría enfocado para los clientes. De este modo, tendrán mayor control las reservas que realicen. Finalmente, se plantea agregar en el sistema de administración de reservas una sección para la gestión del posicionamiento en buscadores (Search Engine Optimization) de las páginas de presentación de contenido al cliente, así como, una opción para manejar tarifas de descuento en las reservas, como descuentos por festividades.

Recomendaciones.

Al momento de desarrollar productos de software para una organización o cliente, es de vital importancia considerar tanto los requerimientos necesarios como los recursos disponibles. Esta tarea es clave ya encamina al producto final a cumplir los estándares del cliente. Además, permite que el sistema de software desarrollado sea eficiente y de calidad. Por lo tanto, para lograr el fin de esta actividad es trascendental trabajar de forma constante con el cliente, para recibir retroalimentación continua de los resultados y realizar cambios a tiempo.

Respecto a los recursos disponibles, son de gran relevancia ya que brindan una mejor noción al programador sobre cómo enfrentar un problema para solucionarlo. Por ejemplo, en el caso del sistema de administración de reservas implementando, se consideró que todos los dispositivos con los que cuenta el hotel son laptops características más que suficientes para una aplicación realizada con Angular. Sin embargo, si la organización hubiese contado con dispositivos de menor gama, se hubiese considerado otro marco de trabajo para este sistema. Esto se debe a que en las aplicaciones desarrolladas con este marco de trabajo, parte del procesamiento de datos es realizado por el cliente, por lo que en un dispositivo sin características muy buenas, el rendimiento del sistema de administración de reservas se vería afectado.

REFERENCIAS BIBLIOGRÁFICAS

Angular - Angular components overview. Angular. (23 de febrero de 2022). Recuperado el 13 de noviembre, 2022 de https://angular.io/guide/component-overview

A Summary of Schema Design Anti-Patterns and How to Spot Them | MongoDB. (2022). Mongodb.com. Recuperado el 13 de noviembre, 2022 de https://www.mongodb.com/developer/products/mongodb/schema-design-anti-pattern-summary/#the-anti-patterns

API PayPhone: Cobro por app. Documentación PayPhone. (6 de diciembre de 2022). Recuperado el 12 de diciembre, 2022 de https://docs.payphone.app/doc/api/

Auth0. (s.f.). *Introduction to JSON Web Tokens*. JSON Web Token Introduction. Recuperado el 13 de noviembre, 2022 de https://jwt.io/introduction

CanActivate. Angular. (2022). Recuperado el 13 de noviembre, 2022 de https://angular.io/api/router/CanActivate

Cherry, B., Benats, P., Gobert, M., Meurice, L., Nagy, C., & Cleve, A. (2022). Static Analysis of Database Accesses in MongoDB Applications. *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. https://doi.org/10.1109/saner53432.2022.00111

Gass, O., Meth, H., & Maedche, A. (2014). PaaS Characteristics for Productive Software Development: An Evaluation Framework. *IEEE Internet Computing*, *18*(1), 56–64. https://doi.org/10.1109/mic.2014.12

Haik, Y., & Shahin, T. (2011). *Engineering Design Process* (2da ed.). Cengage Learning.

Hanson, J. (s.f.). Passport.js. Recuperado el 13 de noviembre, 2022 de https://www.passportjs.org/

Holmes, S., & Harber, C. (2019). *Getting MEAN with Mongo, Express, Angular, and Node*. Manning.

Introduction to services and dependency injection. Angular. (28 de febrero de 2022). Recuperado el 13 de noviembre, 2022 de https://angular.io/guide/architecture-services

IEEE. (30 de noviembre de 2018). ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering. https://doi.org/10.1109/ieeestd.2018.8559686

Jazayeri, M. (2007). Some Trends in Web Application DGevelopment. *Future of Software Engineering (FOSE '07)*. https://doi.org/10.1109/fose.2007.26

MongoDB Schema Design Best Practices | MongoDB. (2022). Mongodb.com. Recuperado el 13 de noviembre, 2022 de https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/

Oh, J., Ahn, W. H., Jeong, S., Lim, J., & Kim, T. (2013). Automated Transformation of Template-Based Web Applications into Single-Page Applications. *2013 IEEE 37th Annual Computer Software and Applications Conference*. https://doi.org/10.1109/compsac.2013.54

Poulter, A. J., Johnston, S. J., & Cox, S. J. (2015). Using the MEAN stack to implement a RESTful service for an Internet of Things application. *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. https://doi.org/10.1109/wf-iot.2015.7389066

Reinman, A. (s.f.). Nodemailer. Recuperado el 13 de noviembre, 2022 de https://nodemailer.com/

Zhao, G., Huang, W., Liang, S., & Tang, Y. (2013). Modeling MongoDB with Relational Model. *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*. https://doi.org/10.1109/eidwt.2013.25

Zhao, J. (2022). Application and Practice of MVC Architecture Pattern in On-the-job Internship Management System. *2022 International Conference on Networks, Communications and Information Technology (CNCIT)*. https://doi.org/10.1109/cncit56797.2022.00013

Zhou, W., Li, L., Luo, M., & Chou, W. (2014). REST API Design Patterns for SDN Northbound API. *2014 28tGh International Conference on Advanced Information Networking and Applications Workshops*. https://doi.org/10.1109/waina.2014.153

ANEXO A: Código de la aplicación

Todo el código referente a la aplicación se encuentra en el siguiente repositorio: https://github.com/ochiribogaz/Proyecto-Integrador-Sistema-de-Reservas-de-Sea-World-Hotel.git

ANEXO B: Diseño de la aplicación

El diseño de la aplicación está implementado en la herramienta de generación de prototipos de Figma: https://www.figma.com/file/TAP2UaCcGFcJbB8Xa0sJhy/Proyecto-Integrador-Oscar-Chiriboga?node-id=0%3A1&t=Ih5yn73ORO49tj81-1