

Katie Lai Yin Chiu Vasquez-20211920057

Wendy Elizabeth Orellana Benitez -20211920040

Mavet Abigail Cruz Méndez -20211920123

Erick David Blanco Santos - 20221900148

|  |
| --- |
| Sistema de reservaciones online  2018 |
| Universidad Nacional Autónoma de Honduras - Wikipedia, la enciclopedia libre |
| 30 abril |

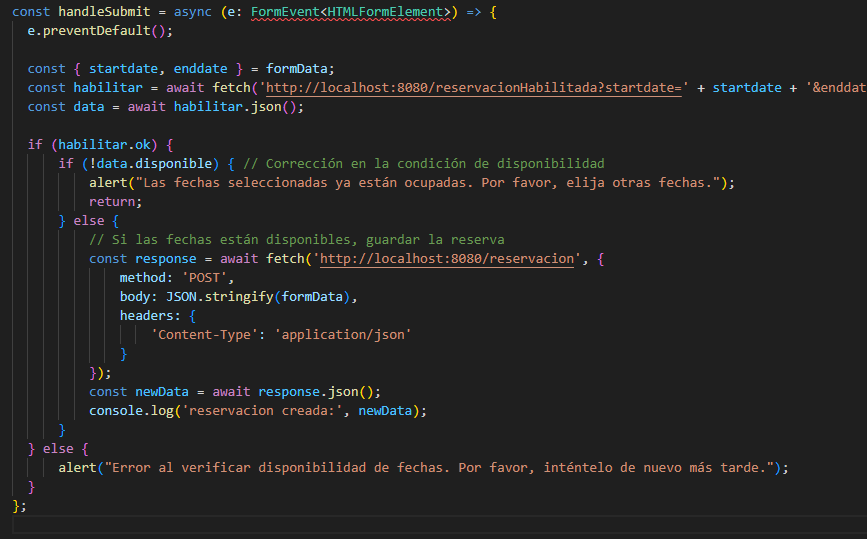
# Introduccion

La creación de un sistema de reservaciones de hotel efectivo implica una combinación técnica y una experiencia de usuario intuitiva. Mediante la utilización de Go, un lenguaje de programación eficiente y escalable; React, una potente biblioteca para la construcción de interfaces de usuario dinámicas y atractivas; y MongoDB, una base de datos NoSQL flexible y altamente escalable, se establece un sólido cimiento para la implementación de un sistema que cumpla con las demandas de la industria hotelera moderna.

A lo largo de este documento, exploraremos en detalle cada etapa del proceso de desarrollo de nuestro sistema de reservaciones de hotel. Desde el diseño inicial del sistema y la implementación de sus características clave, hasta la integración de tecnologías y la gestión de datos, cada aspecto será meticulosamente documentado para proporcionar una guía completa y detallada para el desarrollo del proyecto.

|  |
| --- |
| Herramientas |
| Backend  * Lenguaje: Go * Framework: Echo * Base de datos: Mongo DB   Funcionalidad  Configuración de un servidor HTTP con Echo.  Definición de rutas para manejar peticiones HTTP.  Conexión a la base de datos MongoDB.  Implementación de operaciones CRUD (Crear, Leer, Actualizar, Borrar) para interactuar con la base de datos. Frontend  * React con Vitejs   Funcionalidad  Configuración de un proyecto React con Vite.js para el desarrollo rápido.  Creación de componentes para la interfaz de usuario.  Integración con el backend mediante peticiones HTTP para consumir y mostrar datos.  Implementación de funcionalidades interactivas utilizando React Hooks y componentes. |
| Base de datos:  * Mongo DB   Funcionalidad  Creación de una base de datos MongoDB para almacenar los datos de la aplicación.  Definición de colecciones y documentos para estructurar los datos de manera adecuada.  Implementación de operaciones CRUD para interactuar con la base de datos desde el backend. |
| Documentación  Para realizar este aplicativo, el primer paso que tomamos fue instalar nuestras herramientas principales, GO que es nuestro lenguaje de programación y Visual Studio Code que es el entorno en el trabajamos con las 3 herramientas.  Luego, alojamos nuestro proyecto en una carpeta, en este caso llamada “ReOnline”, una vez estamos en la carpeta, creamos otra carpeta(“ReservacionWeb”) y un archivo “.go”. Proximamente en el directorio “ReservacionWeb” aplicamos el comando “go mod init” para crear un modulo en el cual se guardaran todas las dependencias de nuestro proyecto.    Instalamos “Echo” para empezar a crear nuestra aplicación web y utilizar nuestra API.    Creamos el archivo go.work, el cual contendrá la versión de go que usamos y la carpeta donde esta nuestro archivo .go      Con el siguiente código se inicia “Echo”.    Dentro del directorio “ReOnline”, con el comando “npm create vite”, creamos nuestro proyecto de React para conectarlo y trabajar junto con go.    E instalamos el resto de sus dependencias con el comando “npm i”.    Próximamente, instalamos MongoDB en nuestra máquina, instalamos la extensión de MongoDB en Visual Studio e instalamos las dependencias dentro de la carpeta donde tenemos el archivo .go.        Este es el total de librerías o dependencias que usamos en nuestro proyecto    Para iniciar React utilizamos el comando “npm run build” dentro de la carpeta con el nombre de nuestro proyecto de React, en este caso la llamamos “Client”. El código de la imagen es parte del archivo APP.tsx que es nuestro frontend conecta con el archivo main.go que es nuestro backend, para hacer la función inicial de nuestro proyecto qué es crear un registro dentro de nuestra BD      Algunos errores que se nos presentaron fueron los siguientes |
| El error 405(Method no allowed) ocurre por varias situaciones:   1. El método HTTP incorrecto: Asegúrate de que estás utilizando el método HTTP correcto para acceder al recurso. Por ejemplo, si estás intentando enviar datos a través de un formulario, asegúrate de que el formulario esté configurado para enviar datos utilizando el método POST. 2. Permisos incorrectos: Es posible que el servidor no permita el método solicitado para el recurso específico debido a restricciones de configuración o permisos incorrectos en el servidor. 3. Configuración del servidor: La configuración del servidor web puede no permitir el método solicitado para la URL específica. Esto podría estar configurado en el archivo de configuración del servidor. 4. Problemas de red: A veces, los problemas de red pueden causar que se produzcan errores 405. Asegúrate de que la conexión a Internet esté funcionando correctamente.   Para resolver este error, hay que verificar la configuración del servidor y asegurar de que estés utilizando el método HTTP correcto para acceder al recurso deseado  Error 404 (Not found)  Este error ocurrio ya que no habiamos iniciado el servidor de echo eso se logra ejecutando el comando “go run server.go” en tu terminal. Esto iniciará el servidor en http://localhost:8080. Puedes acceder al servidor desde tu navegador o realizar solicitudes HTTP  Errores en funcion reservacioHabilitada:    reservacionHabilitada se encarga de verificar si hay reservaciones activas en la base de datos que coincidan con un rango de fechas específico   * Se conecta a la base de datos MongoDB utilizando el cliente proporcionado. * Construye un filtro de consulta usando la estructura bson.M de MongoDB. Este filtro busca documentos que cumplan con al menos una de dos condiciones: * La fecha de inicio (start\_date) del documento es anterior a la fecha de finalización del rango proporcionado (enddate), y la fecha de finalización (end\_date) del documento es posterior a la fecha de inicio del rango proporcionado (startdate). * La fecha de inicio del documento está dentro del rango proporcionado (startdate y enddate). * Utiliza el método CountDocuments para contar los documentos en la colección reservacion que cumplen con el filtro. * Si ocurre algún error durante la consulta, la función devuelve false y el error asociado. * Si no hay documentos que coincidan con el filtro, la función devuelve true, lo que indica que las fechas proporcionadas están disponibles para una nueva reserva. Si hay documentos que coinciden con el filtro, devuelve false, indicando que las fechas ya están reservadas.   Esta funcion nos retornaba siempre el valor true por lo que la siguiente condicion no hacia el llamado a una alerta que indicaba que las fechas estaban ocupadas:  C:\Users\user\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\C0C7C76D30BD3DCAEFC96F40275BDC0A\Imagen de WhatsApp 2024-04-29 a las 21.06.10_3eb4e5b2.jpg  El problema que estaba experimentando el sistema se debia a que el fetch que llama la funcion mediante la direccion a localhost:8080 donde se aloja el backend no estaba enviando correctamente los parametros que la funcion necesitaba comparar, por lo tanto agregamos las siguientes lineas de codigo para verificar que se enviara las fechas en el formato correcto  console.log('${enddate}');  console.log('${startdate}');  y luego obtuvimos un nuevo fecth de esta manera  const habilitar = await fetch('http://localhost:8080/reservacionHabilitada?startdate=' + startdate + '&enddate=' + enddate);  y de esta manera verificamos y concluimos que se enviaran correctamente los parametros que la funcion necesitaba y al encontrar fechas que coincidian esta retornaba un valor bool False  C:\Users\user\AppData\Local\Packages\5319275A.WhatsAppDesktop_cv1g1gvanyjgm\TempState\B1A59B315FC9A3002CE38BBE070EC3F5\Imagen de WhatsApp 2024-04-29 a las 22.29.24_4ad9afef.jpg  Este código proporciona un servidor web que permite gestionar reservaciones de alojamientos. Utiliza el framework “Echo” para manejar las solicitudes HTTP y una base de datos MongoDB para almacenar la información de las reservas y verificar la disponibilidad de fechas.  Importaciones:     * Se importan las bibliotecas y paquetes necesarios para el funcionamiento del servidor web y la interacción con la base de datos. Esto incluye paquetes estándar de Go para operaciones básicas, así como paquetes externos como Echo para la gestión de solicitudes HTTP y el controlador oficial de MongoDB para Go para interactuar con la base de datos.     Estructura del Código     * “reservación”   Define la estructura de datos para representar una reserva de alojamiento. Cada campo tiene una etiqueta “json” que especifica cómo será cuando se envíe/reciba como JSON.  Función “main”     * Función principal del programa donde se configura el servidor “Echo”y se establecen las rutas de las APIS. Se conecta a la base de datos MongoDB. * Se crea un cliente de MongoDB utilizando la URI "mongodb://localhost:27017/BDR", donde "BDR" es el nombre de la base de datos.   Manejo de Solicitudes HTTP   * Solicitud POST /”reservacion” * Crea una nueva reserva. El cuerpo de la solicitud debe contener los datos de la reserva en formato JSON. Los datos se insertan en la colección "reservacion" de la base de datos MongoDB. * Solicitud GET /”reservacionHabilitada”      * Verifica la disponibilidad de fechas para una reserva. Se espera que la solicitud incluya los parámetros de consulta "startdate" y "enddate", que representan las fechas de inicio y fin de la reserva. La función reservacionHabilitada consulta la base de datos para determinar si hay reservas superpuestas para las fechas especificadas.   Función “reservacionHabilitada”     * Esta función se utiliza para verificar si hay disponibilidad de fechas para una reserva. Realiza una consulta a la base de datos MongoDB para contar el número de reservas que se superponen con las fechas dadas. Si el recuento es cero, indica que las fechas están disponibles; de lo contrario, indica que ya hay reservas para esas fechas.   Este código es un componente de React que representa un formulario para realizar reservaciones de hotel.  Importaciones:     * Aquí importamos los módulos necesarios de React. useState se usa para crear un estado en el componente funcional App, ChangeEvent se utiliza para manejar eventos de cambio en los campos del formulario, y FormEvent se utiliza para manejar eventos de envío del formulario.   Estado del formulario:     * Aquí utilizamos useState para inicializar el estado del formulario. formData es un objeto que contiene los valores de los diferentes campos del formulario.   Función handleChange:     * Esta función se encarga de manejar los cambios en los campos del formulario. Extrae el nombre (name) y el valor (value) del campo que ha cambiado. Si el campo es uno de los que deben ser números enteros (Cantidad, ninos, adultos), convierte el valor a entero. Luego, actualiza el estado del formulario con los nuevos valores utilizando setFormData. |

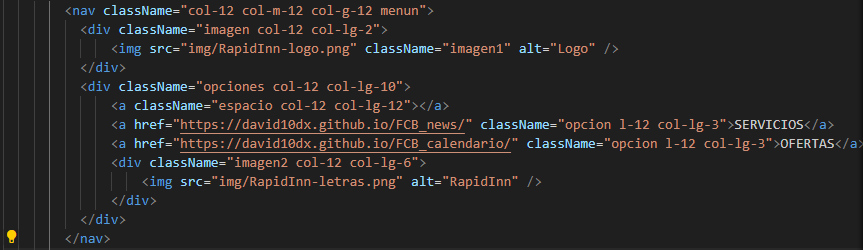
Función handleSubmit:



* Esta función maneja el envío del formulario. Primero, evita el comportamiento predeterminado del envío del formulario utilizando e.preventDefault(). Luego, utiliza fetch para enviar una solicitud al servidor para verificar la disponibilidad de las fechas seleccionadas. Si las fechas están disponibles, envía otra solicitud POST al servidor para guardar la reserva. Si hay algún error, muestra un mensaje de alerta.

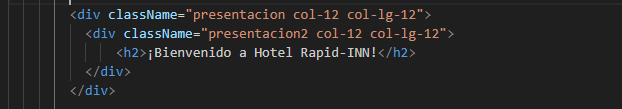
Esta parte del código es la parte de retorno del componente App, que contiene toda la interfaz de usuario definida con JSX.

Navegación:



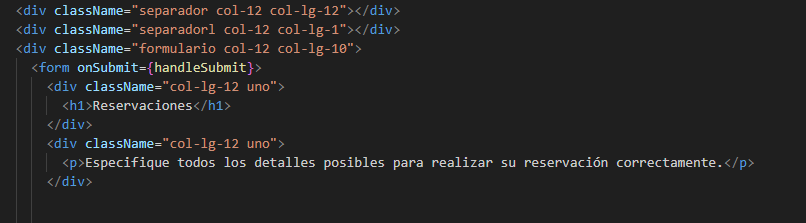
* Esta sección representa la barra de navegación. Contiene un logotipo (imagen), enlaces a los servicios (SERVICIOS) y ofertas (OFERTAS), y un texto adicional (imagen2).

Presentación:



* Esta sección muestra un mensaje de bienvenida al hotel Rapid-INN.

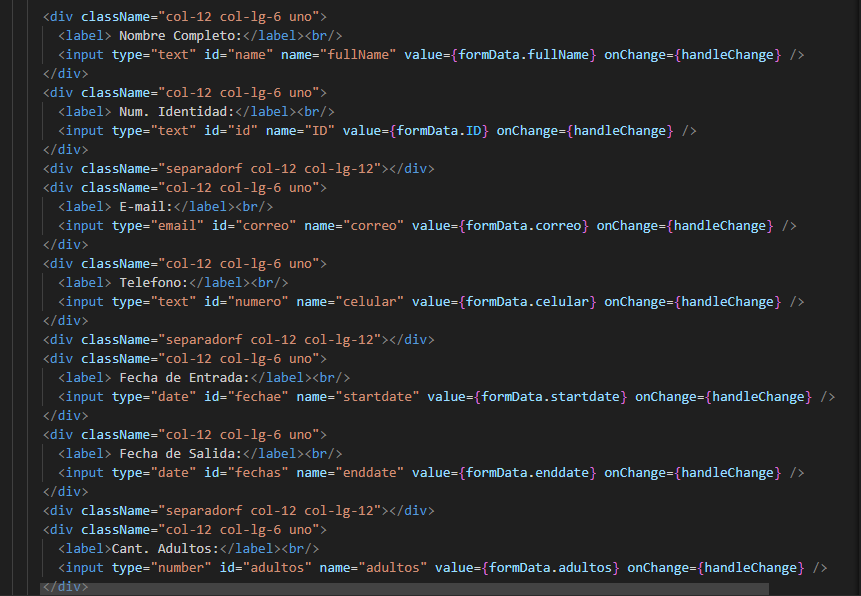
Formulario de Reservaciones:

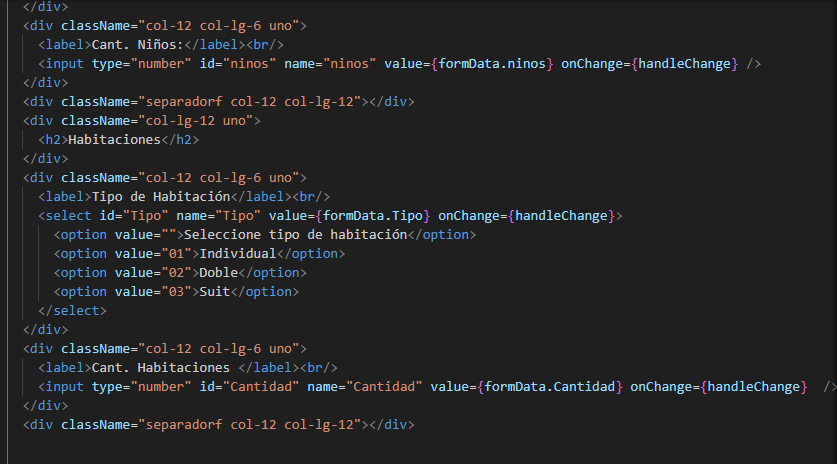


* Este es el contenedor del formulario de reservaciones. El formulario está contenido dentro de un elemento <form> y la función handleSubmit se ejecutará cuando se envíe el formulario.

Campos del Formulario:

* Hay varios campos de entrada dentro del formulario, como nombre completo, número de identidad, correo electrónico, teléfono, fechas de entrada y salida, cantidad de adultos y niños, tipo de habitación y cantidad de habitaciones. Cada campo está representado por un conjunto de elementos <div> que definen el diseño y etiquetas (<label>) para los nombres de los campos.





Botón de Guardar:



* Este es el botón de guardar, que se encuentra al final del formulario. Al hacer clic en este botón, se enviará el formulario y se ejecutará la función handleSubmit.

Separadores:

● Hay varios elementos <div> con clases específicas (separador, separadorl, separadorf, separadorb) que se utilizan para crear espacios y separadores visuales en la interfaz de usuario.

En resumen, esta parte del código representa la interfaz de usuario de la aplicación de reservaciones de hotel, que incluye una barra de navegación, un mensaje de bienvenida, un formulario para ingresar los detalles de la reservación y un botón para guardar la reservación.