UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESTRUCTURA DE DATOS
CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA
TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO

MANUAL TÉCNICO
USOCIAL

ENNER ESAÍ MENDIZABAL CASTRO
CARNÉ: 202302220
SECCIÓN: B

GUATEMALA, 27 DE ABRIL DEL 2,024

ÍNDICE

INTRODUCCIÓN

Este manual se creó con la finalidad de dar a conocer las funciones, método y procedimientos que se llevaron para la creación y diseño de la aplicación web USocial, para que de esta forma se puedan comprender e reimplementar de ser necesario.

OBJETIVOS

1. GENERAL

1.1. Especificar de manera clara las soluciones que se utilizaron para la implementación del programa web USocial.

2. ESPECÍFICOS

2.1. Presentar una tabla de endpoints que muestre la manera en que se realizaron las conexiones del backend con el frontend.

2.2. Dar a conocer las especificaciones de software y hardware que se requieren para el funcionamiento correcto de la aplicación web.

ALCANCES DEL SISTEMA

Este manual tiene el propósito de proporcionar el conocimiento necesario para comprender la estructura que posee la aplicación web USocial junto a la lógica que se llevó a cabo para la implementación de cada una de las soluciones de la red social USocial. Por todo esto, está dirigido a los desarrolladores que desean aprender más sobre API's y busquen ejemplos de soluciones a este tipo de implmentaciones a ser desarrolladas en sus proyectos personales o profesionales.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**
  - Tarjeta de red
  - Procesador de 1GHz
  - 2 GB de memoria RAM
  - Ratón y Teclado.
  - 2GB de almacenamiento libre
- **REQUISITOS DE SOFTWARE**
  - Sistema operativo: Windows, macOS o Linux
  - Navegador web (de preferencia basado en chromium)
  - Editor de código (de preferencia VSCode)
  - NodeJS y sus dependencias
  - Acceso a una red privada de preferencia.

DESCRIPCIÓN DE LA SOLUCIÓN

- **Creación del diseño del Frontend:** Para la creación del frontend se utilizó REACT, pero este requiere de JavaScript, CSS y HTML para estructurar la página web, así que para resolver este problema y facilitar el proceso de la creación del programa, se usó bootstrap, el cual prestó velocidad y agilidad al trabajo de diseño en gran medida.
- **Validación de la contraseña:** Para esta aplicación web, era necesario que se validara que la contraseña tuviera al menos 1 letra mayúscula, 1 letra minúscula, 1 número, 1 carácter especial y que tuviera por lo menos 7 caracteres. Para solucionar esto, lo que se llevaron a cabo fueron variables que contaban cada uno de los tipos de caracteres de la contraseña, y, si estos poseían más de uno entonces, se podría usar la contraseña.
- **Relleno obligatorio de los campos:** Para los campos de texto, algunos eran opcionales, pero otros no, por lo que para que se rellenaran los campos de texto obligatoriamente se usó una función de bootstrap que se llama "requered", la cual solicitaba que se ingresara algún dato dentro del campo, sino no se podría pulsar el botón, apareciendo un mensaje sobre el campo de texto que dictamina que se ingrese algo.
- **Manejo de Usuarios y publicaciones:** Para el manejo de los usuarios se usó como "servidor" un arreglo dentro del backend, el cual permitía que se fueran ingresando objetos de tipo Json que representaban los usuarios. De la misma manera se trabajó con las publicaciones, únicamente variando la estructura del Json, dado a que son objetos distintos.
- **Manejo del administrador:** Así como hay usuarios, también debía haber un administrador, el cual tendría opciones especiales al ingresar a su modulo. Para distinguir entre el usuario administrador y el usuario común y corriente, se usó únicamente un condicional, el cual, al reconocer que el carné era el del administrador, puesto a que este carnet y contraseña ya

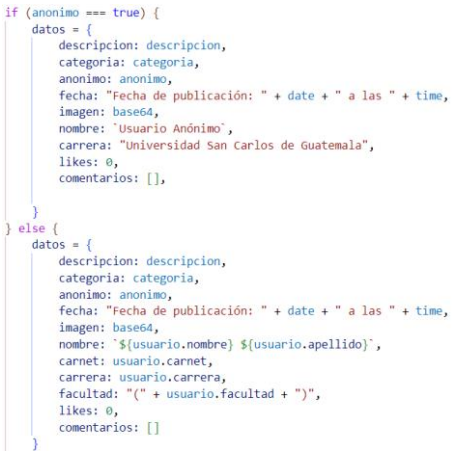
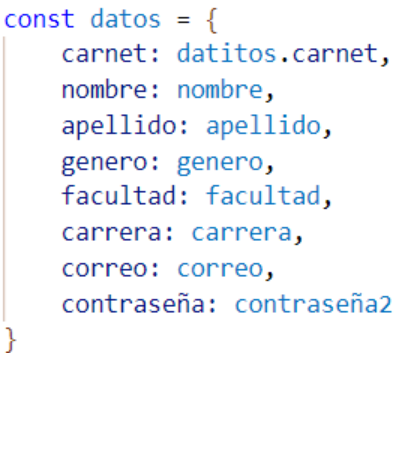
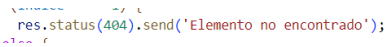
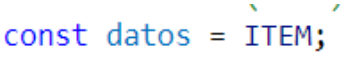
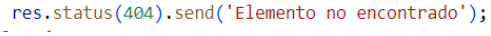
estaban previamente establecidos, redirigiría el usuario a la página a la que debería de ir.

- **Tabla de usuarios y publicaciones:** Para la realización de estas tablas, las cuales permitirían la visualización de toda la información de cada una de las publicaciones y usuarios, así como su eliminación, simplemente se llamó una función del backend la cual entregaría todo el arreglo que se usaba como "servidor". Posteriormente, se usó toda la información se este para crear la tabla y acceder a los datos por medio de un índice que proporcionaría al programa el conocimiento de la ubicación de la posición de la información de cada uno de los objetos de tipo Json.
- **Eliminación de usuario y publicación:** Para la realización de una eliminación de una publicación o de un usuario se creó, dentro del backend, una función de tipo "delete", la cual permitiría que, con el índice que obtendría el programa para identificar la posición de cada una de las publicaciones y usuarios, pueda eliminarlo.
- **Código único de publicación:** Para las publicaciones, era necesario crear un ID para cada una, dado a que se precisaba poder hallar cada una de estas. Por tal motivo se recurrió a la utilización de una variable dentro del backend, la cual aumentaría de uno en uno y se colocaría a cada una de las nuevas publicaciones, de esta manera sería capaz de evitar que dos publicaciones tengan el mismo ID.
- **Pestaña Inicio:** La pestaña inicio es la pestaña en la que se muestran todas las publicaciones que se han creado en la red social, para que se pudiera mostrar en orden de la más reciente a la más antigua simplemente se usó una función del backend que entregaría el arreglo con todos los objetos de tipo de Json que guardan la información de las publicaciones, para que de este modo se obtenga la información de estos, y se pueda presentar de forma reversa para que se muestren en orden de los más nuevos a los más antiguos.

- **Creación de las publicaciones:** Para la creación de las publicaciones se usó una cookie la cual guarda la información del usuario al momento del registro, para que de esta forma la publicación tenga la información del usuario que la creó si es que no se escoge la opción de anónimo, la cual se creó con un condicional que al reconocer que esta seleccionada esa opción, simplemente no coloca la información personal de quien lo público. Ahora para la colocación de la imagen, se insertó un buscador de archivos con el cual seleccionaría la imagen, la cual a posterior se guardaría en base 64 dentro del Json. Por último, para guardar la fecha de la creación de la publicación, se usó la función date() y se le dio el formato deseado.
- **Pestaña de tendencias:** En esta pestaña de muestran las 10 publicaciones con más me gustas dentro de toda la aplicación, y para esto se realizó, dentro del backend, una función, la cual tomaría el arreglo que guardaría las publicaciones, las ordenaría y entregaría al frontend únicamente las 10 publicaciones con más me gustas para que se muestren.
- **Pestaña para editar perfil de usuario:** Para editar los datos del usuario se creó una pestaña, muy similar a la pantalla de registro, la cual solicitaría los datos del usuario para que sean actualizados, todos menos el carné dado a que ese se usa como identificador. Luego se guarda dentro del arreglo dentro del backend con la posición que se obtendría por medio del identificador que es el carné.

LÓGICA DEL PROGRAMA

<table>
<tr><th>Método</th><th>Dirección y tipo de método</th><th>Body</th><th>Respuesta</th></tr>
<tr><td>Validar Inicio de Sesión</td><td>/inicio</td><td rowspan="2"><pre>const datos = {
  carnet: parseInt(carnet, 10),
  contraseña: contraseña
}</pre></td><td rowspan="2"><pre>if (!usuario) {
    const respuesta = {
        success: false,
        user: null
    }
    res.status(404).send(respuesta);
} else {
    const respuesta = {
        success: true,
        user: usuario
    }
    res.json(respuesta)</pre></td></tr>
<tr><td>Recibe el Json que se envía en el inicio de sesión y verifica que el carnet y la contraseña coincidan con alguno de los usuarios para luego enviar el Json con la información completa del usuario</td><td>POST</td></tr>
<tr><td>Crear un Usuario</td><td>/registro</td><td rowspan="2"><pre>const datos = {
  carnet: parseInt(carnet, 10),
  nombre: nombre,
  apellido: apellido,
  genero: genero,
  facultad: facultad,
  carrera: carrera,
  correo: correo,
  contraseña: contraseña2
}</pre></td><td rowspan="2"><pre>const respuesta = {
    success: false
}</pre></td></tr>
<tr><td>Recibe los datos para la creación del un nuevo usuario y los sube al arreglo en donde se guarda la información de los usuarios.</td><td>POST</td></tr>
<tr><td>Crear un Post nuevo</td><td>/home</td><td></td><td><pre>res.status(201).send({response: 'Publicación guardada correctamente'})</pre></td></tr>
</table>

<table>
<tr><td>Recibe la información de que se desea colocar en la publicación y los datos de las cookies del usuario para luego asignarlos a un nuevo objeto de tipo Json para el arreglo</td><td>POST</td><td> 
</td><td></td></tr>
<tr><td>Editar Usuario</td><td>/editar/${carnet}</td><td rowspan="2"> 
</td><td> 
</td></tr>
<tr><td>Recibe los datos que se ingresaron para actualizar y el carnet de las cookies para buscar al usuario y modificar los datos en sus posiciones respectivas</td><td>POST</td><td>
</td></tr>
<tr><td>Dar me gusta a las publicaciones</td><td>/like/${id}</td><td rowspan="2"> 
</td><td> 
</td></tr>
<tr><td>Recibe el ID de la publicación que recibió el me gusta y le suma +1 al contador de me gustas que tiene</td><td>POST</td><td> 
</td></tr>
</table>