UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

INTRODUCCIÓN A LA PROGRAMACIÓN Y COMPUTACIÓN 1

CATEDRÁTICO: ING. WILLIAM ESTUARDO ESCOBAR ARGUETA

TUTOR ACADÉMICO: JOSUÉ RODOLFO MORALES CASTILLO

MANUAL TÉCNICO

DANIEL ALEJANDRO PORTILLO GARCIA

CARNÉ: 202307534

SECCIÓN: B

GUATEMALA, 27 DE ABRIL DEL 2,024

ÍNDICE

INTRODUCCIÓN

El fin de este manual es, exponer la lógica general de todos los procesos del programa USocial, que hace en aspectos generales el Código para que funcione, describir procesos, funciones, variables, objetos, clases, procedimientos y entre más aspectos de los sistemas que se utilizaron dentro del programa USocial.

OBJETIVOS

1. GENERAL

1.1. Exponer las funciones del Código del sistema “USocial” de forma general.

2. ESPECÍFICOS

2.1. Objetivo 1: Explicar el funcionamiento de funciones utilizadas dentro de código

2.2. Objetivo 2: Indicar qué es lo que se hará en este manual relacionado a su código

2.3. Objetivo 3: Indicar el uso de librerías externas y el porqué de su uso.

2.4. Objetivo 5: Explicar el orden de la lógica que sigue el programa según las necesidades del usuario.

2.5. Objetivo 6: mostrar los requisitos del programa, lógica y flujo.

ALCANCES DEL SISTEMA

Este manual explica el funcionamiento detallado en sus aspectos generales, sobre cada método, función, condicional y demás procesos sobre el funcionamiento del sistema “USocial”, en el cual se darán características de la lógica utilizando consultas de javaScrip, uso de html, css en React junto a paquetes y funciones del mismo. Para exponer cómo es posible poner en práctica conocimientos de programación en la lógica en el uso de problemas de manejo de objetos y consultas web por medio de metodos mode, react y manejo de datos entre interfaces BACEND Y FRINTEND para adaptar líneas de Código a la lógica de una problemática que se solicite, en este caso para la creación de un programa de

manejos de datos de una red social orientado en la universidad, administrando usuarios, publicaciones, reportes de tablas, mejo de usuarios, entre otras funciones.

ESPECIFICACIÓN TÉCNICA

- **REQUISITOS DE HARDWARE**
    - Para continuar un futuro desarrollo del programa los requisitos utilizados para el desarrollo del programa son los siguientes:

    - Computadora Windows, procesador, mínimo corei5, 7th generación, con memoria RAM de 4 gb como mínimo.

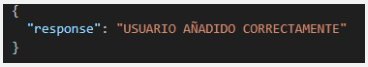
- **REQUISITOS DE SOFTWARE**
    - Requisitos de software necesario y obligatorios son los siguientes:

- Sistema operativo Windows 10/11.
- Visual Studio code.
- Node js v20.12.2(LTS).
- Recusros y librerias de React.

DESCRIPCIÓN DE LA SOLUCIÓN

- El enunciado del problema solicita la creación de un sistema que nos permita simular una red social ambientada en la universidad en la cual existen 2 tipos de usuarios, los cuales tendrá acceso a creación y edición de distintas datos según el tipo de su usuario, para ello se opto por el uso de manejo de datos de servidor/usuario llamado backend y frontend, los cuales manejan los datos enviados por los usuario para ser procesados de forma mas eficiente en segundo plano, para un manejo de datos dinámico, tanto en el registro de usuarios, y creación de datos como publicaciones, usuarios, cargas de datos etc.

LÓGICA DEL PROGRAMA

Método	Dirección y tipo de método	Body	Respuesta
Mostrar usuarios Por medio de una petición get obtenemos la matriz/arreglo donde contendremos nuestros usuarios y se mostraran o listaran en formato json	/usuarios GET		
Mostrar Publicacione Al igual que en usuario listaremos o mostraremos la lista que contiene los datos de nuestras publicaciones en formato jason y procesar sus datos en el frontend	/getPublicaciones GET		
Búsqueda por Carnets Por medio del campo de carnet de nuestro arreglo usamos funciones de busqueda para filtrar el carnet que queremos mostrar por medio de get, en caso de encontrarlos lo listara y si no se notificara al usuario, esto servirá principalmente para identificar las publicaciones e inicios de sesión de distintos usuarios	/usuarios/:carnet GET		
Añadir Usuario Se llamará al body o cuerpo del json para añadir datos en los campos del arreglo de usuarios y se insertaran datos conforme estos campos, esto para obtener y enviar datos entre backend y frontend y añadirlos en formato json.	/usuarios POST		
Cargar Usuario Su principal funcion es comunicarse con el frontend para al momento de listar tablas por la carga de archivo json se añadan de forma automática a los array de usuario	/cargauser POST		

Cargar Post
 	
Al igual que usuarios se encarga de comunicar el frontend para al momento de listar tablas por la carga de archivo json se añadan de forma automática a los array de publicaciones, sin contar que en el proceso verificara la existencia de una publicación repetida, que si es encontrada omitirá la inserción
	/cargarpost
	
	POST
		{
		"id": 2,
		"descripcion": "Hola a todos",
		"categoria": "Variedad",
		"anonimo": "false"
		}
			{
			"response": "PUBLICACION AÑADIDO CORRECTAMENTE"
			}

Añadir Publicación
 	
Al igual que usuarios se encarga de comunicar datos entre backend y frontend para añadirlos en formato json al array, en este caso almacenara una imagen en base 64.
	/crearPublicacion
	
	POST
		{
		"id": 2,
		"descripcion": "Hola a todos",
		"imagen": "",
		"nombre": "Daniel Portillo",
		"carfacu": "Sistemas(Ingenieria)",
		"categoria": "Academico",
		"like": 1,
		"fecha": "Fecha: 2024/04/27 Hora: 19:03"
		}
			{
			"response": "PUBLICACION AÑADIDO CORRECTAMENTE"
			}

Validar Login
 	
Valida credenciales enviadas desde el frontend y comprueba su existencia por medio de busqueda dentro del array de usuarios, esto en base al id identificador y su contraseña, que dependiendo de la respuesta se redirecciona al frontend para validar su inicio de sesión.
	/login
	
	POST
		{
		"carnet": 202307534,
		"constraseña": "D123f*"
		}
			"encontrado": false,
			"user": null
			
			{
			"encontrado": true,
			"user": "Daniel Portillo"
			}

Actualizar Usuario
 	
Por medio de búsqueda de identificador se realizara la actualización de campos por medio del array que contiene los usuarios, buscando su índice y recorriendo la matriz para cambiar los campos, esto en respuesta de los campos y tipo de inicio de sesión que se manden desde el frontend.
	/usuarios/:carnet
	
	PUT
		{
		"nombre": "Gabriel",
		"apellido": "Mendez",
		"genero": "Hombre",
		"correo": "G@gmaail.com",
		"facutad": "Ingenieria",
		"carrera": "Sistemas",
		"contraseña": "Mario234-"
		}
			{
			"response": "NO SE ENCONTRO!!!"
			}
			
			{
			"response": "ACTUALIZADO CORRECTAMENTE"
			}

Actualizar Publicación
 	
método con función principal de validar o cambiar estados de publicaciones, se uso principalmente como prueba en los estados de likes.
	/getPubliacaciones/:id
	
	PUT
		{
		"descripcion": "Me voy",
		"imagen": "",
		"nombre": "Usuario anonio",
		"carfacu": "Universidad de san carlos de guatemala",
		"categoria": "Academico",
		"like": 3,
		"fecha": "Fecha: 2024/04/27 Hora: 19:03"
		}
			{
			"response": "LIKE ACTUALIZADO CORRECTAMENTE"
			}
			
			{
			"response": "NO SE ENCONTRO!!!"
			}

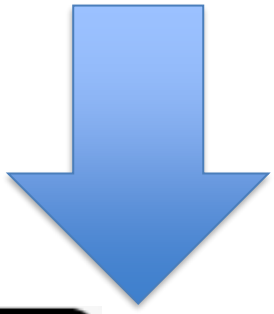
Eliminar Usuario
 	
Por medio de la búsqueda del indice de la matriz se elimina el registro completo del usuario, esto por medio de su campo carnet
	/usuario/:carnet
	
	DELETE
			{
			"response": "Eliminado CORRECTAMENTE"
			}
			
			{
			"response": "NO SE ENCONTRO!!!"
			}

Eliminar Publicación
 	
Misma funcionalidad que usuarios en método delete, en donde por el id o índice de su publicación se elimina de forma dinámica cada registro de la array.
	/getpubliacaciones/:id
	
	DELETE
			{
			"response": "Eliminado CORRECTAMENTE"
			}
			
			{
			"response": "NO SE ENCONTRO!!!"
			}

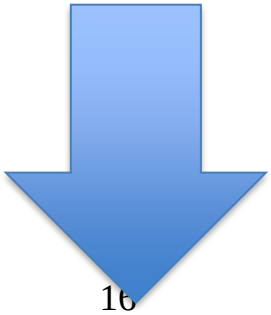
Flujo de la Aplicación

El flujo se basa principalmente en el Backend y frontend, donde el usuario por medio del frontend para interactuar con la aplicación sea un inicio de sesión botones, tablas dinámicas entre otras, para enviar datos y procesarlos en segundo plano por el backend.

Usuario



Frontend



Backend

```
});

//-------------ELIMINAR USUARIO-------------------
servidor.delete("/usuarios/:carnet", (req, res) => { // eliminar element
    const carnet = parseInt(req.params.carnet);
    // const actualizarEstudiante = req.body;
    console.log(carnet)
    const indice = dataUsuarios.findIndex(usuario => {//encontrar el ind
        console.log(usuario.carnet)
        if(usuario.carnet === carnet){
            console.log("Elemento encontrado")
            return usuario
        }
    });

    if (indice === -1) {
        res.status(404).send({ response: 'NO SE ENCONTRO!!!' })
    } else {
        dataUsuarios.splice(indice, 1);
        actualizarDatosArchivo();
        res.status(201).send({ response: "Eliminado CORRECTAMENTE" });
    }


});

//-------------ELIMINAR PUBLICACION-------------------
servidor.delete("/getPublicaciones/:id", (req, res) => { // eliminar ele
    const id = parseInt(req.params.id);
    // const actualizarEstudiante = req.body;
    console.log(id)
    const indice = dataPublicaciones.findIndex(post => {//encontrar el i
        console.log(post.id)
        if(post.id === id){
            console.log("Elemento encontrado")
            return post
        }
    });

    if (indice === -1) {
```