UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESTRUCTURA DE DATOS

CATEDRÁTICO: ING. HERMAN IGOR VELIZ LINARES

TUTOR ACADÉMICO: ESTEBAN HUMBERTO VALDEZ ENNATI

MANUAL TÉCNICO

Natalia Suceli Garrido Montenegro

CARNÉ: 202201385

SECCIÓN: D

GUATEMALA,10 de marzo DEL 2,024

ÍNDICE

ÍNDICE **1**

INTRODUCCIÓN **1**

OBJETIVOS **1**

1. GENERAL 1

2. ESPECÍFICOS 1

ALCANCES DEL SISTEMA **1**

ESPECIFICACIÓN TÉCNICA **1**

● REQUISITOS DE HARDWARE 1

● REQUISITOS DE SOFTWARE 1

DESCRIPCIÓN DE LA SOLUCIÓN **2**

LÓGICA DEL PROGRAMA **2**

❖ NOMBRE DE LA CLASE

Captura de las librerías usadas 2

➢ Librerías 2

➢ Variables Globales de la clase _(El nombre de su clase actual) 3

➢ Función Main 3

➢ Métodos y Funciones utilizadas 3

INTRODUCCIÓN

El backend de USocial es la parte del sistema encargada de gestionar la lógica de negocio, el almacenamiento de datos y la interacción con el frontend a través de APIs. USocial es una plataforma que permite a estudiantes y catedráticos de la Universidad de San Carlos de Guatemala compartir opiniones, reportes y anuncios, por lo que el backend debe ser robusto, seguro y escalable para atender a un gran número de usuarios.

Estructura y Tecnologías

El backend de USocial se desarrolla utilizando Node.js, un entorno de ejecución de JavaScript del lado del servidor, y Express.js, un framework web para construir APIs RESTful de manera rápida y eficiente. Node.js proporciona la capacidad de manejar múltiples solicitudes de manera asíncrona, lo que hace que el backend sea altamente escalable.

OBJETIVOS

1. GENERAL

1.1. familiarizar al estudiante con lenguaje java script

1.2. Aplicar los conocimientos adquiridos en el curso de Introducción a la Programación y Computación 1.

1.3. Utilizar conceptos de programacion web

2.ESPECÍFICOS

2.1. Utilizar node como herramienta de trabajo

2.2. reconocer la base de las conecciones entre datos y rutas en un programa javascript

2.3.

ALCANCES DEL SISTEMA

Gestión de Publicaciones: Permite a los usuarios crear, modificar y eliminar publicaciones en el sistema.
Interacción entre Usuarios: Gestiona comentarios, hilos de conversación y votos en las publicaciones, permitiendo a los usuarios interactuar y participar activamente.
Autenticación y Autorización: Implementa mecanismos para autenticar a los usuarios y verificar sus permisos para acceder a ciertas partes de la plataforma. Se puede usar JWT (JSON Web Tokens) para autenticar las solicitudes del frontend al backend.
Seguridad y Control de Acceso: Proporciona medidas de seguridad para proteger datos sensibles y evitar accesos no autorizados. Esto incluye el uso de HTTPS para las comunicaciones y la validación de datos para prevenir ataques como la inyección de SQL.
Escalabilidad y Rendimiento: Diseña el sistema para soportar un gran número de usuarios concurrentes sin sacrificar el rendimiento, utilizando técnicas de escalabilidad como clustering y caché.
El backend expone una serie de endpoints a través de una API RESTful, que permite al frontend interactuar con el sistema para obtener y enviar datos. Los endpoints están diseñados para ser intuitivos y seguros, permitiendo que el frontend realice acciones como obtener publicaciones, agregar comentarios, votar por contenido y administrar cuentas de usuario.

ESPECIFICACIÓN TÉCNICA

- REQUISITOS DE HARDWARE
  - - una computadora con una buena memoria
  - tener una Core i5 minimo
  - una fuente de poder
- REQUISITOS DE SOFTWARE
  - Node.js:
  - Necesitas tener Node.js instalado en tu sistema para ejecutar código JavaScript del lado del servidor. Node.js incluye npm (Node Package Manager), que te permite instalar y gestionar las dependencias del proyecto. Asegúrate de tener una versión reciente de Node.js para garantizar compatibilidad y seguridad.
  - Express.js:
  - Express.js es el framework web utilizado para construir el backend. Aunque es parte de las dependencias del proyecto, debes asegurarte de tener la configuración correcta para Express.

DESCRIPCIÓN DE LA SOLUCIÓN

- primero se aprendio conceptos basicos para el uso de node js. asi como la instalacion de posibles paquetes de apoyo como lo es npm node. npm bootstrap npm chart entre otros datos

- primero se hizo la aparte logica basica de es decir el backend de el registro de usuarios su actualizacion y su abstracción
- luego se empezo a implementar el are basica en el fronted viendo como era la comunicacion entre rutas y como se debe levantar varios localhost distintos tanto como para el backend como para el fornted.
- se planto la diferencia de modulos entre usuario y administrador por medio de modulos y diferentes sidebar
- una vez entendiendo el funcionamiento de la parte de usuario se hizo una parte mas tecnica que es la de los post ya que la creacion de un post puede tener varias opciones diferentes tanto como opcionales como requeridas

registrar usuario	usa el metodo POST	`router.post('/Registro', Registro)`

Recibe un JSON con los datos que el usuario ingresa desde el frontend, verifica si el usuario esta en uso y también le validación si su contraseña es válida, si todo está bien entonces registra al usuario agregando un nuevo objeto de tipo Usuario y lo agrega dentro del arreglo de Usuarios, de lo contrario retorna mensajes de error para el ingreso correcto de datos.	{
	"carnet":555,
	"nombre":"natalia",
	"apellido":"garrido",
	"genero":"feme",
	"correo":"correo",
	"facultad":"ingenieria",
	"carrera":"sistemas",
	"password":"123"
	
	}
		al aceptarse los datos los guarda como correctos y en casa que el json sea incorrecto manda un mensaje de error

user	usa el metodo GET	`router.get('/users', GetUsers)`
responde con un json pasando toda la lista de usuarios que a registrado		{ "mensaje": "Exito", "usuarios": [{ "carnet": "12523", "nombre": "natalia", "apellido": "Garrido", "genero": "Masculino", "correo": "natalia@gmail.com", "facultad": "Facultad 2", "carrera": "sistemas", "password": "Natalia/8" },

login	usa el metodo post	`router.post('/Login', Login)`

aunque puede ser raro que se registre como post en este metodo e simporntatne el ingreso de dato spor lo cual al ingresar la contrase;a y el carnet estos se comparan y se verifica si existe el usuario en caso de que no lo advierte y tambien sabe si es un usuario normal o un administrador	{
	“carnet”:”123”,
	“password”:”Nombre/1”
	}	

uupdate	usa el metodo PUT	`router.put('/update', UpdateUser )`

se usa para poder editar el registro elaborado de los usuarios donde se manda a llamar al carnet y se puede editar su datos	{
	"carnet":555,
	"nombre":"natalia",
	"apellido":"actualizado",
	"genero":"feme",
	"correo":"correo",
	"facultad":"ingenieria",
	"carrera":"sistemas",
	"password":"123"
	
	}	

delete user	usa el metodo delete	`router.delete('/delete_user', DeleteUser)`
en este se llama al listado de usuarios a buscar una coincidencia con el carnet pasada en caso de que se encuentre se hace el delete con el metodo splice		responde con un json de si el usuario fue eliminado de manera correcta

crear post	usa el metodo post	`router.post('/publicar', Publicar)`

aca se lee un arhivho tipo sjon y se guarda en la lista de post que tendira solo cierto datos de a la hora de registrarlos en el fronted como lo es la catoegira texto imagen los demas se van modificando con el tiempo	{
	{
	"carnet":12,
	"category":"full",
	"text":"aaaa",
	"image":"tuu",
	"dateTime":"15",
	"likes":15,
	"comments":"aaaa"
	}	

mostrar post	usa el metodo get	`router.get('/posts', GetPosts)`
aca se lee la lista que ya se guardo y se envia a manera de que se presente en forma de lista todos los posts hechos con sus modificaciones se hace un map para encontrar el post que queremos que coincida con los numeros de carnet		

mostrar post segun likes	usa el metodo get	`router.get('/TopPosts', GetPostLikes)`
esta es mas usada para las graficas ya que necesitamos saber cuales son las que tienes mas likes y se hizo una funcion adecuada a ella donde se llama al listado de post y se copara para ver cual es mayor		

delete post	metodo DELETE	`router.delete('/delete_post', DeletePost)`
es para borrar un post donde necesitamos el id del post que deseamos comparar para buscarlo en el listado de post haciendo uso de splice		solo regresa un mensaje tipo json de confirmacion de eliminacion

traer usuarios con mas posts	usa el metodo get	`router.get('/TopUsers', GetTopUsers)`
aca se manda a traer la lista de post y se hace una comparacion entre la lista usando `forEach(` para hacer un contador de post por carnet y guardamos cuales son los usuarios con mas posts para asi buscarlos tora vez en la lista		trae a los 10 usuarios con mayor cantidad de posts

metodo para reporte de categoria	usa el metodo get	`router.get('/reportecategoria', reporteCategoria)`
se vuelve a llamar a la lista de posts y se hace un contador por categoria		regresa un json con la respuesta de cual es al cantidad de post segun la categoria

Comentar	usa el metodo post	`router.post('/comentar', Comentar)`

aca recibe datos para guardarlos en la lista de comentarios.	{
	“carnet”:”123”,
	“comentario”:”holi”
	}	

like/dislike	usa el metodo post	`router.post('/like', Like)` `router.post('/dislike', Dislike)`
aunnque son dos endpoints trabajan de manera igual solo que uno aumenta el numero de likes y otro los resta. se puede tomar como un update. pero para no confundirse al operar en el fronted se establece dos post		solo retorna un cambio en el numero de likes que por defecto empieza en cero