



Manual Técnico

Grupo No. 4
Luis Gustavo Morales Morales – 202203228
Hector Daniel Ortiz Osorio - 202203806

Introducción

El siguiente programa tiene la función de ser un sistema estudiantil en el cual se podrá realizar publicaciones sobre ciertos cursos o profesores, además de manejar cursos aprobados, en este manual encontrará el funcionamiento técnico del programa tocando asunto más del código para que pueda utilizarlo correctamente como usuario administrador del programa y la página.

Instalación

Para instalar el programa únicamente se requiere descargar las dos carpetas del siguiente repertorio

https://github.com/LuisMorales1008/Actividad_4_Grupo_4

Inicio del Programa

Debido a que ralentizaría la descarga del programa se tiene un archivo gitignore que ignora la carpeta node_module donde se encuentran los módulos necesarios así que se deberán instalar antes de iniciar el programa esto mediante los siguientes comandos

Frontend

```
> cd frontend
\frontend> npm install
```

Backend

```
4> cd backend
4\backend> npm install
```

Con esto ya se tendrá lo necesario para correr el programa con normalidad, se deberá de correr el backend y el frontend a la vez con los siguientes comandos en la respectiva carpeta

Frontend

```
\frontend> npm start run
```

Backend

```
\backend> node index.js
```

Arquitectura del Programa

Cliente-Servidor es uno de los estilos arquitectónicos distribuidos más conocidos, el cual está compuesto por dos componentes, el proveedor y el consumidor. El proveedor es un servidor que brinda una serie de servicios o recursos los cuales son consumido por el Cliente.

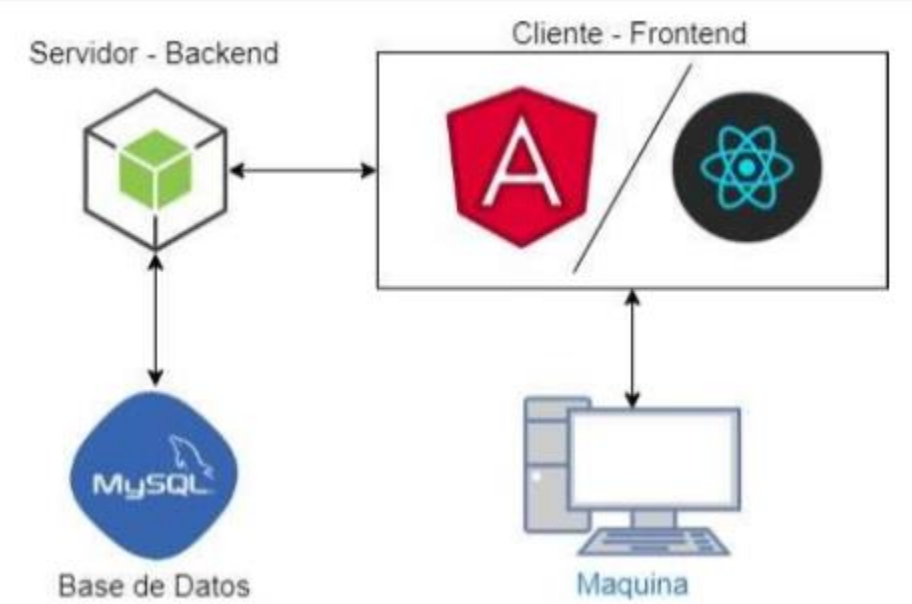
En una arquitectura Cliente-Servidor existe un servidor y múltiples clientes que se conectan al servidor para recuperar todos los recursos necesarios para funcionar, en este sentido, el cliente solo es una capa para representar los datos y se detonan acciones para modificar el estado del servidor, mientras que el servidor es el que hace todo el trabajo pesado.

En esta arquitectura, el servidor deberá exponer un mecanismo que permite a los clientes conectarse, que por lo general es TCP/IP, esta comunicación permitirá una comunicación continua y bidireccional, de tal forma que el cliente puede enviar y recibir datos del servidor y viceversa.

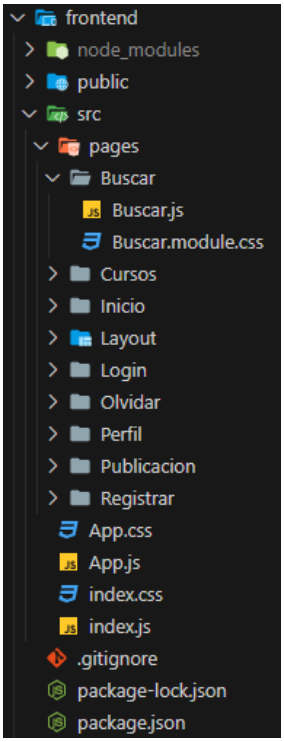
Tecnologías utilizadas:

- Cliente: ReactJS
- Servidor: Node.js y API REST
- Base de datos: MySQL

Diagrama de la arquitectura

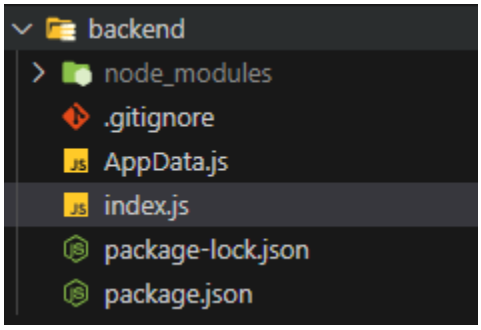


Estructura del Proyecto



El Frontend se divide en un amplio repertorio de carpetas dentro de la carpeta Pages donde se guardan todas las páginas del programa, cada carpeta cuenta con su archivo JS y su archivo CSS.

En el primero se encuentra la programación importante de la página, desde el HTML que estructura la página hasta las acciones que esta deberá realizar, mientras los archivos CSS poseen todas las configuraciones visuales de la página.



Mientras el Backend tiene una distribución a primera vista simple, pero es porque toda la programación se encuentra dentro del archivo index.js donde se realizan todas las funciones posteriormente descritas.

Componentes Principales

Login

```
  const [carnet, setCarnet] = useState('');
  const [contrasena, setContrasena] = useState('');
  const navigate = useNavigate();

  const iniciarSesion = async () => {
  if (!carnet || !contrasena) {
    alert('¡Campos incompletos! Por favor, complete todos los campos.');
    return;
  }

  const url = 'http://localhost:5000/iniciarSesion';

  try {
    const response = await axios.post(url, {
      carnet,
      contrasena
    });

    const { mensaje, usuario } = response.data;

    if (mensaje === 'Ingresó un Usuario') {
      navigate('/layout/inicio');
      alert('¡Bienvenido Estudiante!');
    } else {
      alert('Error: ' + mensaje);
    }
  } catch (error) {
    console.error('Ocurrió un error:', error);
    alert('Ocurrió un error al iniciar sesión. Por favor, inténtalo de nuevo.');
  }
};
```

La función inicial de la página, en esta el usuario introduce los datos solicitados, posteriormente el programa se encarga de leer los datos para para verificar que los datos sean correctos, en caso de serlos se redirección a la página de inicio.

Registrarse

```
const Registrar = () => {
  const navigate = useNavigate();
  const [carnet, setCarnet] = useState('');
  const [nombre, setNombre] = useState('');
  const [apellido, setApellido] = useState('');
  const [contrasena, setContrasena] = useState('');
  const [correo, setCorreo] = useState('');

  const guardarEstudiantes = async () => {
    if (contrasena.length < 4) { // Validar que la contraseña tenga al menos 4 caracteres
        alert('La contraseña debe tener al menos 4 caracteres');
        return;
    }

    const url = 'http://localhost:5000/crearEstudiante';

    try {
        const response = await axios.post(url, {
            carnet,
            nombre,
            apellido,
            contrasena,
            correo,
        });
        const mensaje = response.data.mensaje;

        if (mensaje.includes('Ya existe :(')) {
            alert(mensaje); // Esta alerta muestra si el usuario ya existe
        } else {
            console.log(mensaje);
            alert('Usuario agregado exitosamente');
            borrarCampos();
        }
    } catch (error) {
        console.error('Ocurrió un error:', error);
        alert('¡Ocurrió un error al guardar el usuario! Por favor, inténtalo de nuevo.');
    }
```

En este apartado al cual se accede mediante el menú de Login se podrá crear un nuevo usuario, en esta sección se le solicitan y se recolectan los datos para así mandarlos al backend y posteriormente a la base de datos para guardarlos de manera correcta

Olvidar Contraseña

```
const Olvidar = () => {
  const navigate = useNavigate();
  const [carnet, setCarnet] = useState('');
  const [correo, setCorreo] = useState('');
  const [validacionCorrecta, setValidacionCorrecta] = useState(false);
  const [nuevaContrasena, setNuevaContrasena] = useState('');
  const [confirmarContrasena, setConfirmarContrasena] = useState('');

  const validarDatos = async () => {
    // Verificar si los campos están vacíos
    if (!carnet || !correo) {
      alert('Por favor, completa todos los campos.');
      return;
    }

    try {
      const response = await axios.post('http://localhost:5000/validarDatos', { carnet, correo });
      if (response.data.valido) {
        setValidacionCorrecta(true);
      } else {
        alert('Los datos proporcionados son incorrectos. Por favor, inténtalo de nuevo.');
      }
    } catch (error) {
      console.error('Error al validar los datos:', error);
      alert('Ocurrió un error al validar los datos. Por favor, inténtalo de nuevo.');
    }
  };

  const restablecerContrasena = async () => {
    // Verificar si las contraseñas coinciden
    if (nuevaContrasena !== confirmarContrasena) {
      alert('Las contraseñas no coinciden. Por favor, inténtalo de nuevo.');
      return;
    }

    try {
      const response = await axios.post('http://localhost:5000/olvidoContrasena', {
        carnet,
        correo,
        nuevaContrasena
      });
```

En este apartado al igual que el anterior se accede mediante el menú Login, aquí se le pedirá al usuario su carnet y correo para verificar que sea el dueño de la cuenta y así permitir cambiar la contraseña y sobrescribirla en la base de datos.

Pantalla Inicial

```
const Inicio = () => {
  const [publicaciones, setPublicaciones] = useState([]);
  const [cursos, setCursos] = useState([]);
  const [catedraticos, setCatedraticos] = useState([]);
  const [filtroSeleccionado, setFiltroSeleccionado] = useState('');
  const [filtroValor, setFiltroValor] = useState('');
  const [nuevoComentario, setNuevoComentario] = useState('');
  const [datosUsuario, setDatosUsuario] = useState({
    carnet: '',
  });

  useEffect(() => {
    axios.get('http://localhost:5000/datosUsuario')
      .then(response => {
        if (response.data && response.data.datosUsuario) {
          const { correo, carnet, nombre, apellido } = response.data.datosUsuario;
          setDatosUsuario({ correo, carnet, nombre, apellido });
        } else {
          console.error('No se pudieron obtener los datos del usuario');
        }
      })
      .catch(error => {
        console.error('Error al obtener datos del usuario:', error);
      });
  }, []);

  useEffect(() => {
    axios.get('http://localhost:5000/publicaciones')
      .then(res => {
        setPublicaciones(res.data.publicaciones);
        setCursos(res.data.cursos);
        setCatedraticos(res.data.catedraticos);
      })
      .catch(err => console.error(err));
  }, []);
```

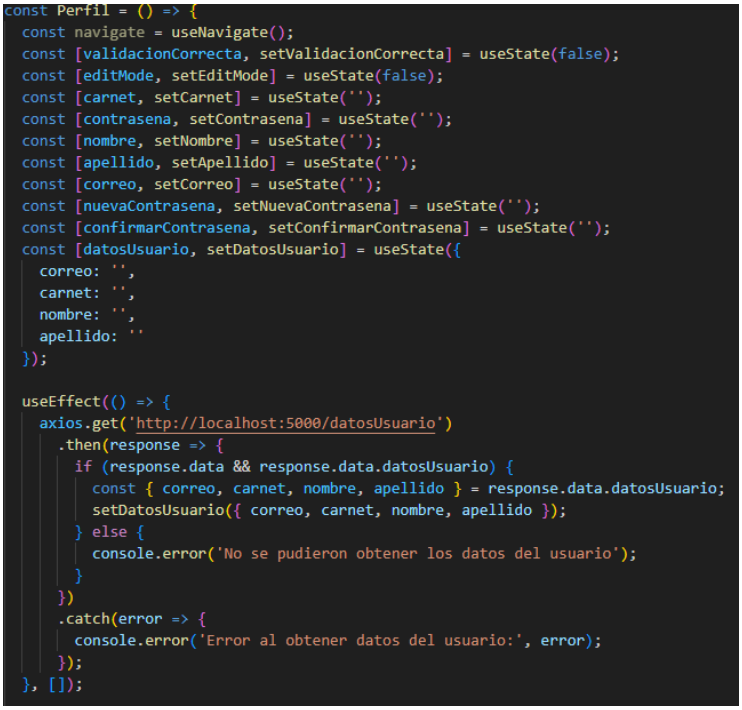
En esta sección se le presentara al usuario las publicaciones que se hayan realizado en el sistema, para esto se realiza un llamado a la base de datos para que entregue los datos solicitados como el texto de la publicación, además de una opción para filtrar las publicaciones según un dato a elección del usuario, momento donde el sistema se encarga de verificar que publicaciones cumplen con los datos dados y así presentarlos.

Crear Publicación

```
const Publicacion = () => {
  const navigate = useNavigate();
  const [tipo, setTipo] = useState('course'); // 'course' o 'professor'
  const [cursoOprofesor, setCursoOprofesor] = useState('');
  const [mensaje, setMensaje] = useState('');
  const [subir, setSubir] = useState(false);
  const [datosUsuario, setDatosUsuario] = useState({
    carnet: '',
  });

  useEffect(() => {
    axios.get('http://localhost:5000/datosUsuario')
      .then(response => {
        if (response.data && response.data.datosUsuario) {
          const { correo, carnet, nombre, apellido } = response.data.datosUsuario;
          setDatosUsuario({ correo, carnet, nombre, apellido });
        } else {
          console.error('No se pudieron obtener los datos del usuario');
        }
      })
      .catch(error => {
        console.error('Error al obtener datos del usuario:', error);
      });
  }, []);

  const handleSubmit = async (e) => {
    e.preventDefault();
    setSubir(true);

    const url = 'http://localhost:5000/crearPublicacion';
```

En este apartado como su nombre indica se pueden crear publicaciones, el sistema se encarga de guardar los datos esenciales como el texto de la publicación, si se trata de un profesor o un curso y también llama a la base de datos para detectar al usuario en uso y guardar su carnet en los datos de la publicación.

Ver Perfil



```
const Perfil = () => {
  const navigate = useNavigate();
  const [validacionCorrecta, setValidacionCorrecta] = useState(false);
  const [editMode, setEditMode] = useState(false);
  const [carnet, setCarnet] = useState('');
  const [contrasena, setContrasena] = useState('');
  const [nombre, setNombre] = useState('');
  const [apellido, setApellido] = useState('');
  const [correo, setCorreo] = useState('');
  const [nuevaContrasena, setNuevaContrasena] = useState('');
  const [confirmarContrasena, setConfirmarContrasena] = useState('');
  const [datosUsuario, setDatosUsuario] = useState({
    correo: '',
    carnet: '',
    nombre: '',
    apellido: ''
  });

  useEffect(() => {
    axios.get('http://localhost:5000/datosUsuario')
      .then(response => {
        if (response.data && response.data.datosUsuario) {
          const { correo, carnet, nombre, apellido } = response.data.datosUsuario;
          setDatosUsuario({ correo, carnet, nombre, apellido });
        } else {
          console.error('No se pudieron obtener los datos del usuario');
        }
      })
      .catch(error => {
        console.error('Error al obtener datos del usuario:', error);
      });
  }, []);
```

Una vez más el nombre dice todo lo necesario, en esta opción el usuario visualizaría los datos de su cuenta, lugar donde se realiza un llamado a la base de datos que almacena los datos del usuario, en el momento que el usuario realiza un inicio de sesión sus datos quedan guardados aparte para así solo tener que hacer el llamado a esos datos en específico.

Buscar

```
const Buscar = () => {
  const [cursosAprobados, setCursosAprobados] = useState([]);
  const [carnetBuscado, setCarnetBuscado] = useState('');
  const [usuarioEncontrado, setUsuarioEncontrado] = useState(null);
  const [error, setError] = useState('');
  const [mostrarCursos, setMostrarCursos] = useState(false);


  const buscarUsuario = async () => {
    try {
      const response = await axios.get(`http://localhost:5000/buscarUsuario?carnet=${carnetBuscado}`);
      if (response.data && response.data.datosUsuario) {
        setUsuarioEncontrado(response.data.datosUsuario);
        setError('');
        // Llamar a cargarCursosAprobados después de encontrar el usuario
        cargarCursosAprobados(carnetBuscado);
      } else {
        setUsuarioEncontrado(null);
        setError('No se encontró ningún usuario con el carnet proporcionado.');
      }
    } catch (error) {
      console.error('Error al buscar usuario:', error);
      setError('Ocurrió un error al buscar el usuario. Por favor, inténtalo de nuevo.');
    }
  };
```

En este apartado el usuario podrá buscar los datos o cursos aprobados de los demás usuarios de la página, para esto se solicita un numero de carnet, si este coincide con alguno en la base de datos el sistema se encarga de filtrar los datos necesarios de ese usuario para así presentarlos.

Cursos Aprobados

```
const Cursos = () => {
    const navigate = useNavigate();
    const [cursosAprobados, setCursosAprobados] = useState([]);
    const [todosCursos, setTodosCursos] = useState([]);
    const [selectedTab, setSelectedTab] = useState('aprobados');

    useEffect(() => {
        cargarCursosAprobados();
        cargarTodosCursos();
    }, []);

    const cargarCursosAprobados = async () => {
        try {
            const response = await axios.get('http://localhost:5000/cursos-aprobados');
            setCursosAprobados(response.data.cursosAprobados);
        } catch (error) {
            console.error('Error al cargar los cursos aprobados:', error);
        }
    };

    const cargarTodosCursos = async () => {
        try {
            const response = await axios.get('http://localhost:5000/cursos');
            setTodosCursos(response.data.cursos);
        } catch (error) {
            console.error('Error al cargar todos los cursos:', error);
        }
    };

    const handleTabChange = (tab) => {
        setSelectedTab(tab);
    };
```

En esta sección el usuario podrá ingresar los cursos que ha aprobado, para esto se le solicitaran una lista de datos, estos datos serán recogidos por el frontend y mandados a guardar a la base de dato para así tener un registro ordenado de los mismos y para ser llamados cuando sea necesario en otro apartado como la sección de Buscar.

Index.js

```
app.post('/cerrarSesion', (req, res) => {
    connection.end();
    res.json({ mensaje: 'Sesión cerrada exitosamente' });
});

// Ruta para crear un estudiante y guardarlo en la base de datos
app.post('/crearEstudiante', (req, res) => {
    const { nombre, apellido, contrasena, correo, carnet } = req.body;

    // Verifica si el correo ya esta registrado en la base de datos
    connection.query('SELECT * FROM usuarios WHERE correo = ?', [correo], (errorCorreo, resultadosCorr
        if (errorCorreo) {
            console.error('Error al verificar el correo en la base de datos:', errorCorreo);
            res.status(500).json({ mensaje: 'Error interno del servidor' });
            return;
        }

        if (resultadosCorreo.length > 0) {
            console.log('Error: El correo ya está registrado');
            res.status(400).json({ mensaje: 'El correo ya está registrado' });
            return;
        }

        // Verifica si el carnet ya esta registrado en la base de datos
        connection.query('SELECT * FROM usuarios WHERE carnet = ?', [carnet], (errorCarnet, resultados
            if (errorCarnet) {
                console.error('Error al verificar el carnet en la base de datos:', errorCarnet);
                res.status(500).json({ mensaje: 'Error interno del servidor' });
                return;
            }
```

Esto no es una función visible para un usuario, pero cabe aclarar que este documento es de los más importantes del proyecto esto debido a que se encarga de hacer la conexión a la base de datos de MySQL además de manejas de manera correcta los datos y hacer algunas verificaciones necesaria para que todo funcione.