



DESARROLLO WEB - MANUAL TECNICO

Informe No. 4









11 DE SEPTIEMBRE DE 2025 GUATEMALA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA - PRÁCTICAS INICIALES PARA INGENIERÍA
EN CIENCIAS Y SISTEMAS
Ing. Herman Igor Veliz Linares

Índice




Integrantes	3
Tutores.....	3
Fotografía del Grupo con los tutores	4
Introducción	5
Registro de Usuario	5
• Registro Académico	¡Error! Marcador no definido.
• Nombres y Apellidos	¡Error! Marcador no definido.
• Contraseña.....	¡Error! Marcador no definido.
• Correo Electrónico	¡Error! Marcador no definido.
Funciones Principales	¡Error! Marcador no definido.
• Pantalla Principal.....	¡Error! Marcador no definido.
• Crear Publicación	¡Error! Marcador no definido.
• Comentarios en Publicaciones	¡Error! Marcador no definido.
• Ver Perfil	¡Error! Marcador no definido.
Conclusión	6

Integrantes

CARNÉ

 Diego J. Fernando Contreras Bantes	-	202200251
 Jasson Enrique Cabrera Estrada	-	202100047
 Bhrandon Omar Melendez Galvez	-	202307711
 Luis Alejandro Gutierrez Vasquez	-	202112281
 Abner Emanuel Palacios Morales	-	202002633
 Kevin Geovani Xum Quiej	-	202201297

Tutores

-  Alfredo Domínguez
-  Ángel Pérez
-  Mateo Diego

Fotografía del Grupo con los tutores



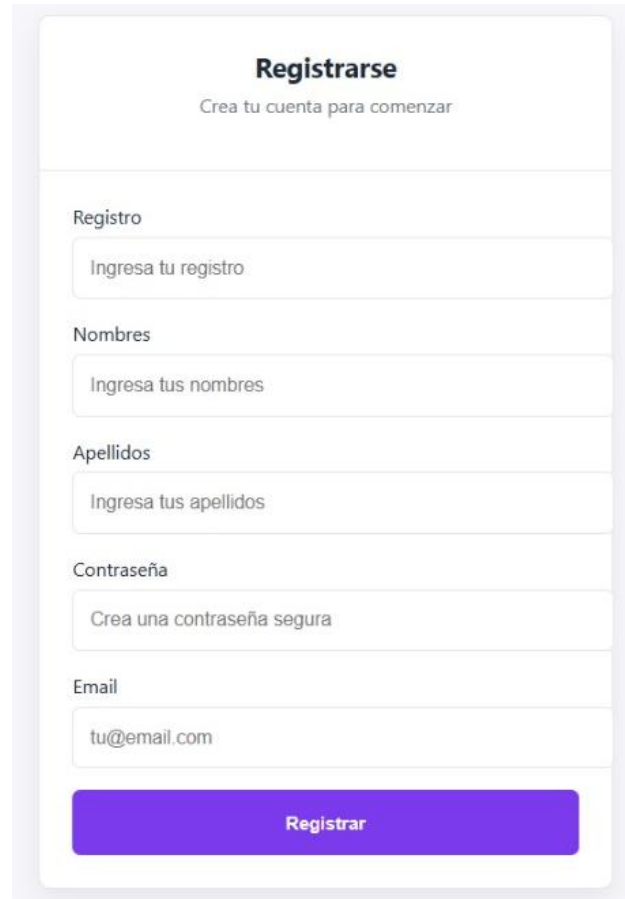
Introducción

El propósito de este manual técnico es proporcionar una visión detallada sobre el desarrollo y la estructura técnica de la aplicación web que se ha diseñado para los estudiantes de la Facultad de Ingeniería. Esta aplicación está orientada a facilitar el acceso a información clave sobre los cursos y catedráticos, así como a permitir la interacción entre los estudiantes a través de comentarios y publicaciones.

La arquitectura de la aplicación se ha desarrollado utilizando tecnologías modernas y eficientes como ReactJS para la interfaz de usuario y NodeJS para el backend, lo que garantiza una experiencia de usuario fluida y un servidor escalable. A través de este manual, los desarrolladores y técnicos involucrados podrán entender la estructura del proyecto, los detalles de la implementación de cada módulo y cómo interactúan los distintos componentes.

Arquitectura del Proyecto

La arquitectura de la aplicación se divide en tres componentes principales: el Frontend, el Backend y la Base de Datos. Cada una de estas partes juega un papel crucial en el funcionamiento de la aplicación, trabajando de manera interconectada para proporcionar una experiencia completa al usuario.



The image shows a registration form with a light gray background and rounded corners. At the top, the title 'Registrarse' is centered in bold, with the subtitle 'Crea tu cuenta para comenzar' below it. The form contains five input fields, each with a label above it: 'Registro' (placeholder: 'Ingresa tu registro'), 'Nombres' (placeholder: 'Ingresa tus nombres'), 'Apellidos' (placeholder: 'Ingresa tus apellidos'), 'Contraseña' (placeholder: 'Crea una contraseña segura'), and 'Email' (placeholder: 'tu@email.com'). A purple 'Registrar' button is positioned at the bottom of the form.

Registrarse
Crea tu cuenta para comenzar

Registro
Ingresa tu registro

Nombres
Ingresa tus nombres

Apellidos
Ingresa tus apellidos

Contraseña
Crea una contraseña segura

Email
tu@email.com

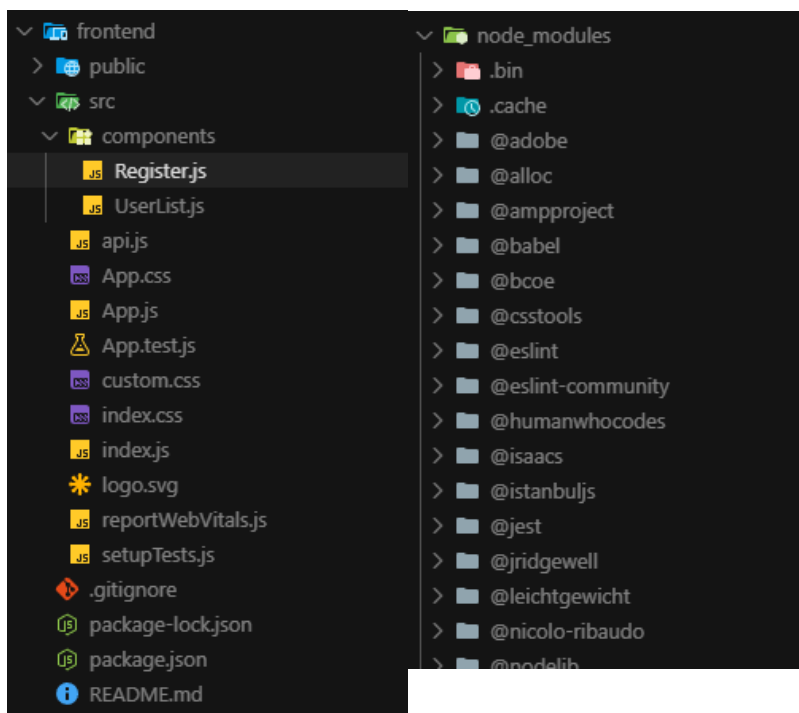
Registrar

Frontend (ReactJS)

La interfaz de usuario está construida utilizando ReactJS, una librería de JavaScript que permite construir interfaces dinámicas y reactivas. ReactJS facilita la creación de componentes reutilizables que pueden actualizarse eficientemente sin necesidad de recargar la página. Los principales componentes de la interfaz incluyen:

- ✚ **Formularios de Registro:** Para que los usuarios puedan registrarse con sus datos personales
- ✚ **Lista de Publicaciones:** Donde se muestran las publicaciones sobre catedráticos y cursos.
- ✚ **Visualización del Perfil:** Permite que los usuarios puedan ver y editar su información personal.

Además, se utilizan Hooks como `useState` y `useEffect` para gestionar el estado y realizar peticiones a la API, lo que asegura que la interfaz se mantenga reactiva y actualizada.



Backend (NodeJS y REST API)

El backend está desarrollado en NodeJS, un entorno de ejecución de JavaScript en el servidor, y utiliza una API REST para gestionar la comunicación entre el cliente y la base de datos. La API es responsable de recibir y procesar las solicitudes del frontend y devolver los datos necesarios.




Las principales funciones que maneja el servidor incluyen:

- ✚ **Registro de Usuarios:** El endpoint /register recibe los datos de los usuarios y los almacena en la base de datos. Además, valida que el correo electrónico o el número de registro académico no estén en uso, evitando duplicados.
- ✚ **Publicaciones:** A través del endpoint /posts, los usuarios pueden crear nuevas publicaciones relacionadas con catedráticos o cursos. También pueden eliminar o modificar sus publicaciones existentes.
- ✚ **Comentarios:** El endpoint /comments permite que los usuarios agreguen comentarios a las publicaciones existentes, facilitando la interacción y enriqueciendo la información proporcionada por otros usuarios.

```
Registerjs X
frontend > src > components > Registerjs > Register
1 import React, { useState } from 'react';
2 import api from '../api';
3 import '../custom.css';
4
5 const Register = ({ onSuccess }) => {
6   const [registro, setRegistro] = useState('');
7   const [nombres, setNombres] = useState('');
8   const [apellidos, setApellidos] = useState('');
9   const [contraseña, setContraseña] = useState('');
10  const [email, setEmail] = useState('');
11
12  const handleSubmit = async (e) => {
13    e.preventDefault();
14    try {
15      await api.post('/users', {
16        registro,
17        nombres,
18        apellidos,
19        contraseña,
20        email,
21      });
22      console.log('Usuario registrado');
23      if (onSuccess) onSuccess(); // Llama a la función de actualización
24    } catch (error) {
25      console.error('Error al registrar usuario:', error);
26    }
27  };
28
29  return (
30    <div className="page">
31      <div className="card">
32        <div className="card-header">
33          <h2 className="card-title">Registrarse</h2>
34          <p className="card-description">Crea tu cuenta para comenzar</p>
35        </div>
36        <div className="card-content">
37          <form onSubmit={handleSubmit} className="form">
38            <div>
39              <label htmlFor="registro" className="label">Registro</label>
40              <input
41                id="registro"
42                type="text"
43                className="input"
44                value={registro}
45                onChange={(e) => setRegistro(e.target.value)}
46                placeholder="Ingresa tu registro"
47              />
48            </div>
49          </form>
50        </div>
51      </div>
52    </div>
53  );
54 }
```


Base de Datos (MySQL)

La base de datos está implementada con MySQL, un sistema de gestión de bases de datos relacional que permite almacenar de manera estructurada toda la información relevante de la aplicación. Las entidades principales almacenadas son:

-  **Usuarios:** Información del registro académico, nombres, apellidos, etc.
-  **Publicaciones:** Datos sobre los comentarios y valoraciones de los catedráticos y cursos.
-  **Comentarios:** Los comentarios hechos por los usuarios en cada publicación.

Las relaciones entre estas entidades están bien definidas para garantizar la integridad de los datos y optimizar las consultas, lo que asegura un rendimiento adecuado y un acceso eficiente a la información.

```
UserList.js X
frontend > src > components > UserList.js > ...
1 import React, { useState, useEffect } from 'react';
2 import api from '../api';
3 import '../custom.css';
4
5 const UserList = ({ updateList }) => {
6   const [users, setUsers] = useState([]);
7
8   useEffect(() => {
9     const fetchUsers = async () => {
10       try {
11         const response = await api.get('/users');
12         setUsers(response.data);
13       } catch (error) {
14         console.error('Error al obtener usuarios:', error);
15       }
16     };
17
18     fetchUsers();
19   }, [updateList]); // Dependencia para actualizar la lista
20
21   return (
22     <div className="page">
23       <div className="card" style={{ maxWidth: 640 }}>
24         <div className="card-header">
25           <h2 className="card-title">Lista de Usuarios</h2>
26           <p className="card-description">Usuarios registrados recientemente</p>
27         </div>
28         <div className="card-content">
29           {users.length === 0 ? (
30             <div className="empty-state">
31               <div className="empty-state-icon">👤</div>
32               <p className="muted">No hay usuarios registrados.</p>
33             </div>
34           ) : (
35             <ul className="user-list">
36               {users.map(user => (
37                 <li key={user._id} className="user-item">
38                   <div className="user-name">{user.nombres} {user.apellidos}</div>
39                   <div className="user-details">
40                     {user.email} • Registro: {user.registro}
41                   </div>
42                 </li>
43               ))}
44             </ul>
45           )}
46         </div>
47       </div>
48     </div>
49   );
50 }
```

Funcionalidades del Servidor

El servidor maneja diversas funcionalidades clave que permiten a la aplicación ser dinámica y eficiente. A continuación, se describen las principales funcionalidades que ofrece el servidor.

```
App.js X
frontend > src > App.js > ...
1 import React, { useState } from 'react';
2 import Register from './components/Register';
3 import UserList from './components/UserList';
4 import './custom.css';
5
6 const App = () => {
7   const [updateList, setUpdateList] = useState(false);
8
9   const handleUpdate = () => {
10     setUpdateList(!updateList);
11   };
12
13   return (
14     <div className="app-container">
15       <Register onSuccess={handleUpdate} />
16       <UserList updateList={updateList} />
17     </div>
18   );
19 };
20
21 export default App;
22
```

Registro de Usuarios

El endpoint /register es responsable de recibir los datos de los usuarios y almacenarlos en la base de datos. Durante el proceso, el servidor verifica que el correo electrónico y el número de registro académico sean únicos, evitando duplicados en la base de datos.

Publicaciones

A través del endpoint /posts, los usuarios pueden crear, obtener y eliminar publicaciones. Estas publicaciones pueden ser sobre un catedrático o un curso, y pueden incluir comentarios adicionales que otros usuarios pueden agregar para enriquecer la discusión. Cada publicación incluye:

- ✚ Usuario que creó la publicación
- ✚ Curso o Catedrático asociado
- ✚ Mensaje de la publicación
- ✚ Fecha de creación

Registrarse

Crea tu cuenta para comenzar

Registro

Nombres

Apellidos

Contraseña

Email

Registrar

```
Problems  Output  Debug Console  Terminal  Ports
Compiled successfully!

You can now view frontend in the browser.

http://localhost:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Conclusión

Este manual técnico ha cubierto la arquitectura del sistema, las tecnologías utilizadas y las funcionalidades clave del servidor. El uso de ReactJS en el frontend, NodeJS en el backend y MySQL para la base de datos asegura que la aplicación sea rápida, escalable y fácil de mantener. Con estas bases sólidas, el proyecto es capaz de manejar eficientemente el registro de usuarios, la creación de publicaciones y la gestión de comentarios, proporcionando una plataforma robusta para la interacción estudiantil.