Semillero de Proyectos - Proyecto No.5
Portafolio Digital

1.5 **Portafolio Digital**
Oliver Enriqque Suarez Mora
Karen Melissa Mancilla Jimenez
Samantha de la Mora López
Profesora: Diana Lourdes Avila Molina
Asignatura: Ingeniería de Software I
Universidad Autonoma de Guadalajara
Fecha - 15 de Mayo del 2025

Resumen

Este proyecto consistió en el desarrollo de una plataforma web **Portafolio Digital**, que permite a los desarrolladores gestionar y mostrar sus proyectos de software. La herramienta permite a cada usuario personalizar su perfil, agregar y clasificar proyectos, y conectar su cuenta con GitHub para importar automáticamente repositorios. Incluye funcionalidades de retroalimentación comunitaria, seguridad, y una experiencia de usuario responsiva. Además, se pretende promover la interacción entre usuarios para formar una comunidad activa de aprendizaje y retroalimentación. La plataforma fue desarrollada con herramientas actuales del ecosistema JavaScript y un enfoque centrado en la experiencia del usuario.

Objetivo general

Desarrollar un sistema web funcional, seguro y atractivo que permita a los usuarios crear, administrar , compartir y calificar un portafolio digital de sus proyectos de personales, mejorando su presencia profesional y facilitando la exploración de trabajos realizados por otros usuarios, mediante herramientas modernas y una interfaz intuitiva.

Justificación del sistema

Actualmente, muchos desarrolladores tienen sus proyectos dispersos en diferentes plataformas sin una manera ordenada de presentarlos. Un portafolio digital permite agrupar toda esa información en un solo lugar, lo cual es especialmente útil para estudiantes, freelancers

o profesionales que están construyendo su carrera. Este sistema no sólo ayuda a mejorar la imagen profesional, sino que también promueve la transparencia, la evaluación entre pares y la posibilidad de recibir retroalimentación de la comunidad. Además, puede ser usado como referencia en entrevistas de trabajo o postulaciones a becas y programas académicos.

Metodología aplicada

Se utilizó la metodología **Scrum**, dividiendo el trabajo en sprints de una semana con reuniones de planeación y revisión. Esta metodología permitió mantener un flujo constante de trabajo, definir entregables parciales y adaptarse a nuevas necesidades. El proyecto se organizó mediante tableros de tareas (como Trello), y se asignaron roles específicos como frontend, backend, diseño y pruebas.

Tecnologías y herramientas utilizadas

Frontend:

- **React 19:** Utilizado para construir la interfaz de usuario a partir de componentes reutilizables. Permitió manejar estados, eventos y renderizado dinámico, facilitando una experiencia fluida.
- **React Router DOM:** Se usó para la navegación entre páginas del sitio (como Home, Login, Registro, Perfil). Gracias a esto se construyó una SPA (Single Page Application).
- **React Icons:** Proporcionó los iconos usados en botones y menús, como los de GitHub, comentarios o cerrar sesión.
- **Vite:** Fue el empaquetador de módulos y servidor de desarrollo. Ofreció recarga en caliente y una construcción más rápida comparado con Webpack.
- **Bootstrap 5:** Se utilizó para estructurar los elementos visuales de la interfaz, aplicando un diseño responsivo para adaptarse a computadoras y móviles.
- **SweetAlert2:** Se usó para mostrar alertas estilizadas y confirmaciones (por ejemplo, al eliminar un proyecto o cerrar sesión).

Backend:

- **Node.js con Express:** Sirvió como base para la creación del servidor backend. Express permitió definir rutas como `/login`, `/registro`, `/proyectos`, donde se procesaban las peticiones del cliente.
- **PostgreSQL:** Fue la base de datos relacional donde se almacenó la información de usuarios, proyectos, comentarios y calificaciones. Se organizaron tablas con relaciones entre ellas.

- **JWT (jsonwebtoken):** Se utilizó para generar tokens de sesión al iniciar sesión. Estos tokens se enviaban al cliente y se almacenaban en cookies seguras para mantener al usuario autenticado.
- **bcrypt:** Antes de almacenar las contraseñas, se cifraban con bcrypt para proteger la información de los usuarios.
- **multer:** Librería usada para recibir imágenes subidas desde el frontend y guardarlas en el servidor (por ejemplo, las fotos de los proyectos o del perfil).
- **dotenv:** Permitió centralizar la configuración sensible (como las claves secretas, credenciales de correo o configuración de base de datos) en un archivo `.env` no compartido.
- **cors:** Se configuró para que el frontend pudiera comunicarse con el backend sin problemas de seguridad (CORS policy).
- **cookie-parser:** Se usó para leer y escribir cookies (por ejemplo, para guardar el token JWT del usuario).
- **sanitize-html:** Antes de guardar un comentario o descripción, se limpiaban los textos con esta librería para evitar inyecciones de código o ataques XSS.
- **nodemailer:** Se empleó para enviar correos automáticos a los usuarios, como mensajes de confirmación o restablecimiento de contraseña.
- **sharp y potrace:** Se usaron para procesar imágenes. Sharp optimizó el tamaño y calidad de las imágenes subidas, y potrace permitió generar versiones vectoriales si se requería.

Herramientas de apoyo:

- **Figma:** Con Figma se diseñaron los wireframes y mockups del sistema antes de programar. Esto ayudó a planificar la experiencia de usuario.
- **Postman:** Se utilizó para probar los endpoints del backend de forma independiente del frontend.
- **Git y GitHub:** Se versionó el proyecto por ramas, permitiendo trabajo colaborativo y control de cambios.

Resultados obtenidos

Se implementó un sistema funcional con las siguientes características:

- Registro/login con cifrado y validación.
- Perfiles con bio, imagen, redes sociales y habilidades.
- Creación de proyectos con categorías, etiquetas, imágenes y descripciones.

- Integración con GitHub para mostrar repositorios.
- Filtros y buscador avanzado.
- Visualización pública/privada y sistema de comentarios.
- Panel de administración para moderación.

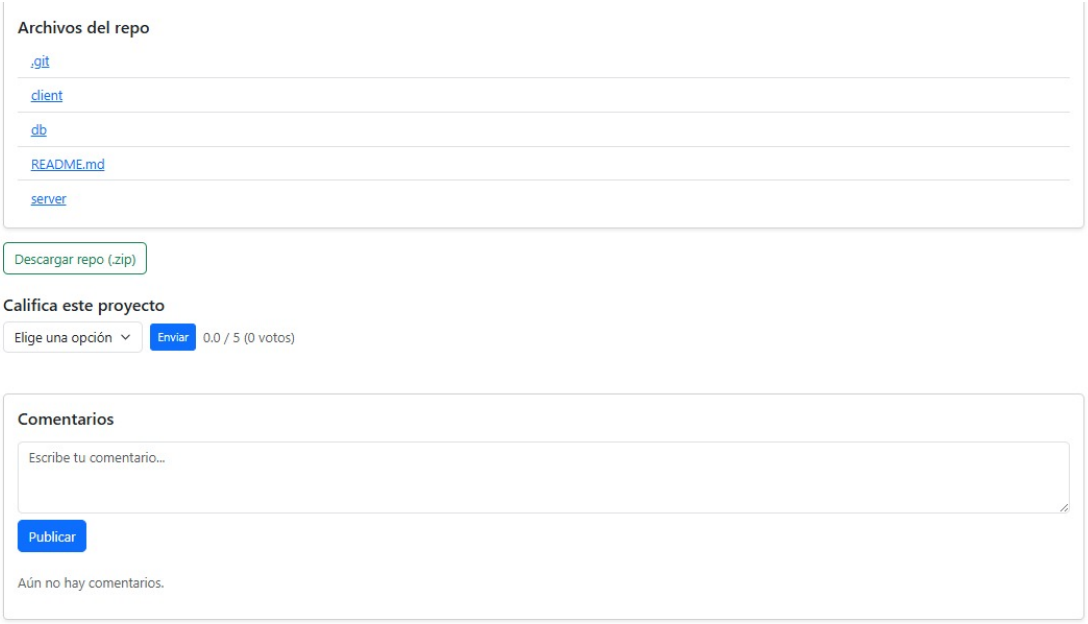

Figura 1: Sección de comentarios y calificación de proyectos

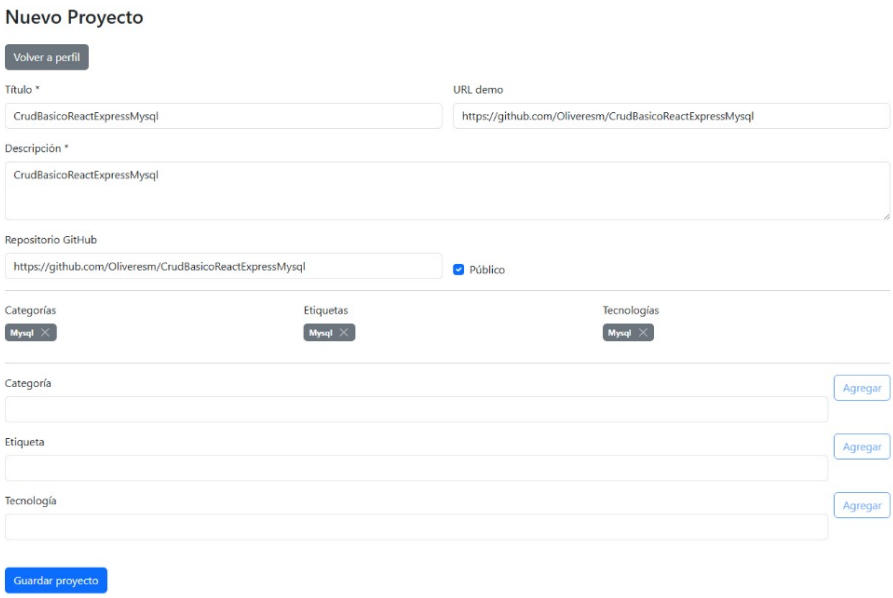

Figura 2: Formulario para agregar un nuevo proyecto al portafolio

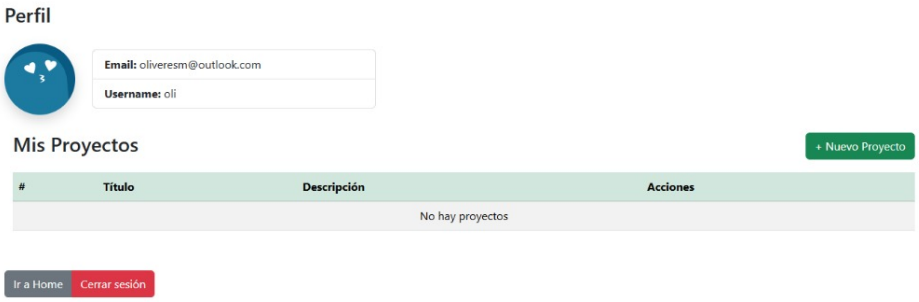

Figura 3: Vista del perfil del usuario con sus datos y proyectos asociados

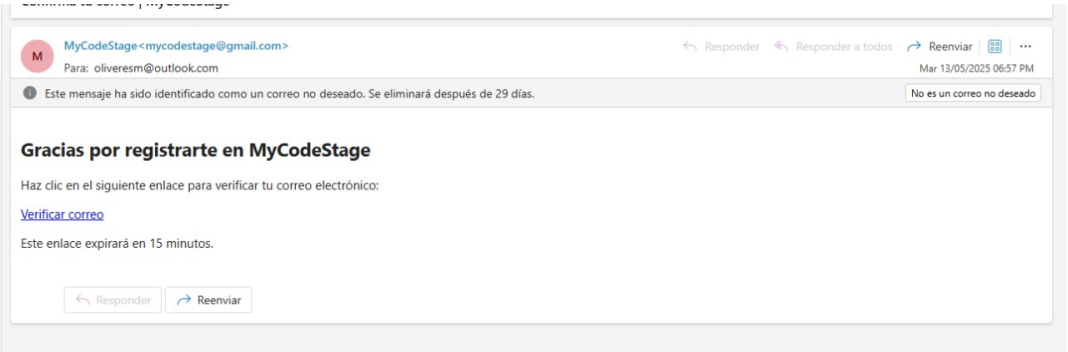

Figura 4: Correo de verificación automática enviado al usuario

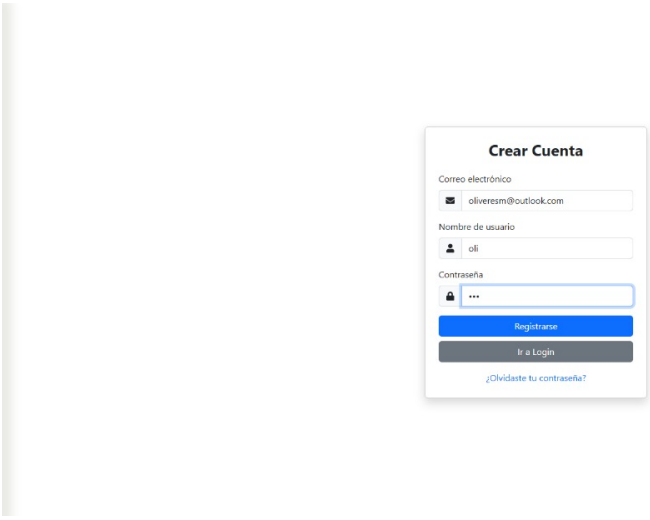

Figura 5: Interfaz de registro de usuario en el sistema

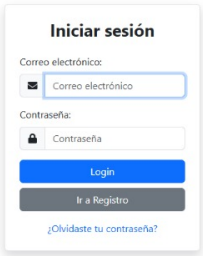

Figura 6: Formulario de inicio de sesión con validación

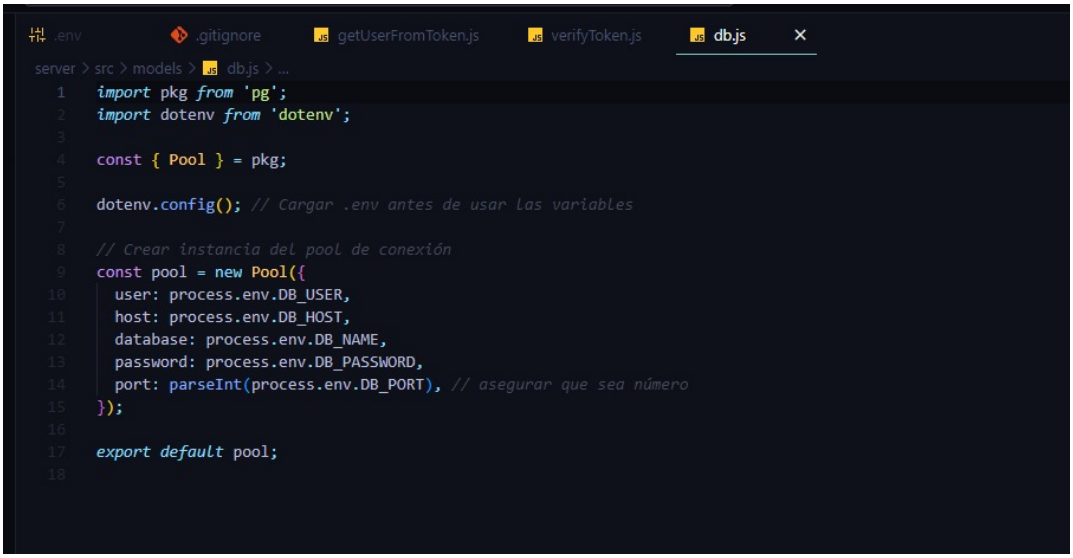

Figura 7: Patron usado: Singleton

server > src > middleware > verifyToken.js > ...

```
import jwt from 'jsonwebtoken';

const verifyToken = (req, res, next) => {
  const authHeader = req.headers.authorization;

  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    return res.status(401).json({ message: "Token no proporcionado o mal formado" });
  }

  const token = authHeader.split(" ")[1];

  jwt.verify(token, "Stack", (err, decoded) => {
    if (err) {
      return res.status(403).json({ message: "Token inválido o expirado" });
    }

    req.user = decoded;
    next();
  });
};

export default verifyToken;
```

Figura 8: Patron usado: Singleton 2

server > src > controllers > helpers > getUserFromToken.js > ...

```
// helpers/getUserFromToken.js
import jwt from "jsonwebtoken";
import pool from "../../models/db.js";

const getUserFromToken = async (req) => {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    throw { code: 401, message: "Token no proporcionado o mal formado" };
  }
  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, "Stack");
    const result = await pool.query("SELECT * FROM login WHERE email = $1", [
      decoded.email,
    ]);
    if (result.rows.length === 0) {
      throw { code: 404, message: "Usuario no encontrado" };
    }
    return {
      user: result.rows[0],
      decoded,
    };
  } catch (err) {
    if (err.code && err.message) throw err;
    throw { code: 403, message: "Token inválido o expirado" };
  }
};

export default getUserFromToken;
```

Figura 9: Patron usado Singleton 3

Conclusiones

El proyecto Portafolio Digital fue una oportunidad para integrar conocimientos técnicos y habilidades de trabajo en equipo. Se logró construir una aplicación web moderna, funcional y segura, con interfaz atractiva, integraciones útiles y funcionalidades avanzadas. Se cumplieron los objetivos y se dejó una base robusta para mejoras futuras. Además, el trabajo con herramientas reales como GitHub, Postman y React permitió simular un entorno laboral profesional.

Referencias

- Sommerville, I. (2011). *Ingeniería de software* (9ª ed.). Pearson Educación.
- GitHub Docs. (s.f.). *GitHub REST API*. `https://docs.github.com/rest`
- Express.js. (s.f.). `https://expressjs.com`
- React Documentation. (s.f.). `https://react.dev`
- PostgreSQL Docs. (s.f.). `https://www.postgresql.org`