

FreeBridge - Documentación Completa de Arquitectura

Versión: 2.0

Fecha: Noviembre 2025

Plataforma: React 19 + Flask 3.1 + MySQL

Tabla de Contenidos

1. [Visión General](#)
 2. [Arquitectura Frontend](#)
 3. [Componentes del Sistema](#)
 4. [Comunicación Backend-Frontend](#)
 5. [Sistema de Autenticación](#)
 6. [Sistema de Archivos y Uploads](#)
 7. [Sistema de Notificaciones y Email](#)
 8. [Gestión de Perfiles \(Avatares y Logos\)](#)
 9. [Guía de Desarrollo](#)
 10. [Debugging y Testing](#)
 11. [Anexos](#)
-

1. Visión General

1.1 ¿Qué es FreeBridge?

FreeBridge es una plataforma que conecta **freelancers** con **empresas**, permitiendo:

-  **Empresas:** Publicar vacantes y gestionar postulaciones
-  **Freelancers:** Buscar y postularse a oportunidades laborales

1.2 Stack Tecnológico

```
Frontend: React 19.1.1 + Vite 7.1.7 + React Router DOM 7.9.5 + Axios 1.13.1
Backend: Flask 3.1.2 + Flask-SQLAlchemy 3.1.1 + PyJWT 2.8.0 + Flask-Mail 0.9.1
Base de Datos: MySQL 8+ (con PyMySQL 1.1.0)
Email: Mailtrap (sandbox SMTP)
Estilos: CSS Modules
Avatares: DiceBear 9.2.4
```

1.3 Arquitectura de Capas

COMPONENTS (Componentes)	← UI reutilizable
CONTEXT + HOOKS (Lógica)	← Estado y comportamiento
API (Comunicación)	← Backend requests
UTILS (Herramientas)	← Funciones puras

2. Arquitectura Frontend

2.1 Estructura de Carpetas

```

client-react/
├── public/                      # Archivos estáticos
└── src/
    ├── App.jsx                  # Componente raíz
    ├── main.jsx                 # Punto de entrada
    └── index.css                # Estilos globales

    ├── api/                     # 🌐 Comunicación HTTP
    │   ├── axiosConfig.js       # Configuración base Axios
    │   ├── authApi.js           # Endpoints autenticación
    │   ├── vacancyApi.js        # Endpoints vacantes
    │   ├── companyApi.js         # Endpoints empresas
    │   ├── profileApi.js         # Endpoints perfil
    │   └── cityApi.js            # Endpoints ciudades

    ├── components/              # 💡 Componentes reutilizables
    │   ├── Navbar.jsx
    │   ├── Footer.jsx
    │   ├── Modal.jsx
    │   ├── loginForm.jsx
    │   ├── registerForm.jsx
    │   ├── vacancyCard.jsx
    │   ├── vacancyList.jsx
    │   └── vacancyForm.jsx

    ├── pages/                   # 📄 Páginas completas
    │   ├── Home.jsx
    │   ├── Login.jsx
    │   ├── Register.jsx
    │   ├── Vacancies.jsx
    │   ├── Profile.jsx
    │   └── CompanyDashboard.jsx

    └── context/                 # 🏃 Estado global
        └── AuthContext.jsx

```

```
hooks/          # ⚖ Lógica reutilizable
  └── useSessionTimeout.js

router/         # ⚡ Navegación
  └── appRouter.jsx

styles/         # 🎨 CSS Modules
  ├── Navbar.module.css
  ├── Footer.module.css
  └── ...

utils/          # 🛠 Utilidades
  └── sessionManager.js

.gitignore
package.json
vite.config.js
ARQUITECTURA_AUTH.md
GUIA_ESTILOS.md
```

2.2 Flujo de Datos



3. Componentes del Sistema

3.1 Sistema de Autenticación

📁 utils/sessionManager.js

Propósito: Gestión centralizada de localStorage

```
// Funciones principales
export const getToken = () => localStorage.getItem("token");

export const setSessionData = (token, userId, role, name) => {
  localStorage.setItem("token", token);
  localStorage.setItem("userId", userId);
  localStorage.setItem("userRole", role);
  localStorage.setItem("userName", name);
  localStorage.setItem("lastActivity", Date.now());
};

export const clearSession = () => {
  localStorage.clear();
};

export const getLastActivity = () => {
  return parseInt(localStorage.getItem("lastActivity")) || Date.now();
};

export const updateLastActivity = () => {
  localStorage.setItem("lastActivity", Date.now());
};
```

Ventajas:

- Evita código duplicado
 - Funciones puras (sin dependencias React)
 - Fácil de testear
-

📁 context/AuthContext.jsx

Propósito: Estado global de autenticación

```
import { createContext, useContext, useState, useEffect } from "react";
import {
  getToken,
  setSessionData,
  clearSession,
} from "../utils/sessionManager";

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [isAuthenticated, setIsAuthenticated] = useState(false);
  const [userRole, setUserRole] = useState(null);
  const [userId, setUserId] = useState(null);

  // Verificar sesión al cargar
  useEffect(() => {
```

```

const token = getToken();
if (token) {
  setIsAuthenticated(true);
  setUserRole(localStorage.getItem("userRole"));
  setUserId(localStorage.getItem("userId"));
}
[], []);

const login = (token, userId, role, name) => {
  setSessionData(token, userId, role, name);
  setIsAuthenticated(true);
  setUserRole(role);
  setUserId(userId);
};

const logout = () => {
  clearSession();
  setIsAuthenticated(false);
  setUserRole(null);
  setUserId(null);
};

return (
  <AuthContext.Provider
    value={{
      isAuthenticated,
      userRole,
      userId,
      login,
      logout,
    }}
  >
  {children}
  </AuthContext.Provider>
);
};

export const useAuth = () => useContext(AuthContext);

```

Uso en componentes:

```

import { useAuth } from "../context/AuthContext";

const MyComponent = () => {
  const { isAuthenticated, userRole, login, logout } = useAuth();

  if (!isAuthenticated) {
    return <p>Debes iniciar sesión</p>;
  }

  return <p>Bienvenido, tu rol es: {userRole}</p>;
};

```

hooks/useSessionTimeout.js

Propósito: Cerrar sesión automáticamente por inactividad

```
import { useEffect } from "react";
import { useAuth } from "../context/AuthContext";
import { getLastActivity, updateLastActivity } from "../utils/sessionManager";

const useSessionTimeout = (timeout = 30 * 60 * 1000) => {
    // 30 min
    const { logout } = useAuth();

    useEffect(() => {
        // Actualizar actividad con eventos del usuario
        const events = ["mousedown", "keydown", "scroll", "touchstart"];

        const handleActivity = () => {
            updateLastActivity();
        };

        events.forEach((event) => {
            document.addEventListener(event, handleActivity);
        });

        // Verificar cada minuto si expiró
        const interval = setInterval(() => {
            const lastActivity = getLastActivity();
            const now = Date.now();

            if (now - lastActivity > timeout) {
                logout();
                window.location.href = "/login?timeout=true";
            }
        }, 60000); // Cada 60 segundos

        return () => {
            events.forEach((event) => {
                document.removeEventListener(event, handleActivity);
            });
            clearInterval(interval);
        };
    }, [logout, timeout]);
};

export default useSessionTimeout;
```

Uso:

```
import useSessionTimeout from "../hooks/useSessionTimeout";

const App = () => {
  useSessionTimeout(30 * 60 * 1000); // 30 minutos

  return <Router>...</Router>;
};
```

3.2 Navegación y Rutas

📁 router/appRouter.jsx

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Home from "../pages/Home";
import Login from "../pages/Login";
import Register from "../pages/Register";
import Vacancies from "../pages/Vacancies";
import CompanyDashboard from "../pages/CompanyDashboard";
import Profile from "../pages/Profile";

const AppRouter = () => {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/vacantes" element={<Vacancies />} />
        <Route path="/company-dashboard" element={<CompanyDashboard />} />
        <Route path="/profile" element={<Profile />} />
      </Routes>
    </BrowserRouter>
  );
};

export default AppRouter;
```

📁 components/Navbar.jsx

Propósito: Barra de navegación adaptativa según estado de sesión

```
import { Link, useNavigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import styles from "../styles/Navbar.module.css";

const Navbar = ({ isCompact = false }) => {
```

```

const { isAuthenticated, userRole, logout } = useAuth();
const navigate = useNavigate();

const handleLogout = () => {
    logout();
    navigate("/");
};

return (
    <nav className={isCompact ? styles.navbarCompact : styles.navbar}>
        <Link to="/" className={styles.logo}>
            FreeBridge
        </Link>

        <div className={styles.navLinks}>
            <Link to="/vacantes">Vacantes</Link>

            {!isAuthenticated ? (
                <>
                    <Link to="/login">Iniciar Sesión</Link>
                    <Link to="/register" className={styles.btnRegister}>
                        Registrarse
                    </Link>
                </>
            ) : (
                <>
                    {userRole === "Empresa" && (
                        <Link to="/company-dashboard">Dashboard</Link>
                    )}
                    <Link to="/profile">Perfil</Link>
                    <button onClick={handleLogout} className={styles.btnLogout}>
                        Cerrar Sesión
                    </button>
                </>
            )}
            </div>
    </nav>
);
};

export default Navbar;

```

3.3 Páginas Principales

📁 pages/Home.jsx

Propósito: Landing page con hero section y categorías

```

import Navbar from "../components/Navbar";
import Footer from "../components/Footer";

```

```

import styles from "../styles/Home.module.css";

const Home = () => {
  return (
    <>
      <Navbar />

      <section className={styles.hero}>
        <h1>Conecta tu talento con oportunidades</h1>
        <p>La plataforma que une freelancers con empresas</p>
        <Link to="/vacantes" className={styles.btnCTA}>
          Ver Vacantes
        </Link>
      </section>

      <section className={styles.categories}>
        <h2>Categorías Populares</h2>
        <div className={styles.categoryGrid}>
          {/* Tarjetas de categorías */}
        </div>
      </section>

      <Footer />
    </>
  );
};

}

```

📁 pages/Login.jsx

Propósito: Página de inicio de sesión

```

import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useAuth } from "../context/AuthContext";
import { loginUser } from "../api/authApi";
import LoginForm from "../components/loginForm";
import Navbar from "../components/Navbar";

const Login = () => {
  const [error, setError] = useState("");
  const { login } = useAuth();
  const navigate = useNavigate();

  const handleLogin = async (credentials) => {
    try {
      const data = await loginUser(credentials);
      login(data.token, data.id_usuario, data.rol, data.nombre);

      // Redirigir según rol
      if (data.rol === "Empresa") {

```

```

        navigate("/company-dashboard");
    } else {
        navigate("/vacantes");
    }
} catch (err) {
    setError(err.message || "Error de autenticación");
}
};

return (
<>
<Navbar isCompact />
<div className="container">
{error && <div className="alert-error">{error}</div>}
<LoginForm onSubmit={handleLogin} />
</div>
</>
);
};

```

📁 pages/CompanyDashboard.jsx

Propósito: Panel de control para empresas

```

import { useState, useEffect } from "react";
import { useAuth } from "../context/AuthContext";
import { getCompanyProfile, saveCompanyProfile } from "../api/companyApi";
import VacancyForm from "../components/vacancyForm";
import Navbar from "../components/Navbar";

const CompanyDashboard = () => {
    const [hasProfile, setHasProfile] = useState(false);
    const [companyData, setCompanyData] = useState(null);
    const { userId } = useAuth();

    useEffect(() => {
        const loadProfile = async () => {
            try {
                const profile = await getCompanyProfile(userId);
                setCompanyData(profile);
                setHasProfile(true);
            } catch {
                setHasProfile(false);
            }
        };
        loadProfile();
    }, [userId]);

    if (!hasProfile) {
        return (

```

```

        <>
          <Navbar isCompact />
          <div>
            <h2>Completa tu perfil de empresa</h2>
            {/* Formulario de perfil */}
          </div>
        </>
      );
}

return (
  <>
  <Navbar isCompact />
  <div>
    <h1>Dashboard de {companyData.nombre_empresa}</h1>

    <section>
      <h2>Publicar Nueva Vacante</h2>
      <VacancyForm embedded={true} />
    </section>

    <section>
      <h2>Postulaciones Recibidas</h2>
      {/* Lista de postulaciones */}
    </section>
  </div>
</>
);
};

```

3.4 Componentes Reutilizables

📁 [components/vacancyCard.jsx](#)

Propósito: Tarjeta individual de vacante

```

import styles from "../styles/vacancyCard.module.css";

const VacancyCard = ({ vacancy }) => {
  const { titulo, descripcion, salario, ciudad, empresa } = vacancy;

  return (
    <div className={styles.card}>
      <h3>{titulo}</h3>
      <p className={styles.company}>{empresa}</p>
      <p className={styles.description}>{descripcion}</p>
      <div className={styles.footer}>
        <span className={styles.location}>📍 {ciudad}</span>
        <span className={styles.salary}>฿ ${salario}</span>
      </div>
    </div>
  );
};

export default VacancyCard;

```

```
        <button className={styles.btnClose}>Postularse</button>
    </div>
);
};
```

components/Modal.jsx

Propósito: Modal genérico reutilizable

```
import { useEffect } from "react";
import styles from "../styles/Modal.module.css";

const Modal = ({ isOpen, onClose, title, children }) => {
  useEffect(() => {
    if (isOpen) {
      document.body.style.overflow = "hidden";
    } else {
      document.body.style.overflow = "unset";
    }
  });

  return () => {
    document.body.style.overflow = "unset";
  };
}, [isOpen]);

if (!isOpen) return null;

return (
  <div className={styles.overlay} onClick={onClose}>
    <div className={styles.modal} onClick={(e) => e.stopPropagation()}>
      <div className={styles.header}>
        <h2>{title}</h2>
        <button onClick={onClose} className={styles.btnClose}>
          ×
        </button>
      </div>
      <div className={styles.content}>{children}</div>
    </div>
  </div>
);
};
```

Uso:

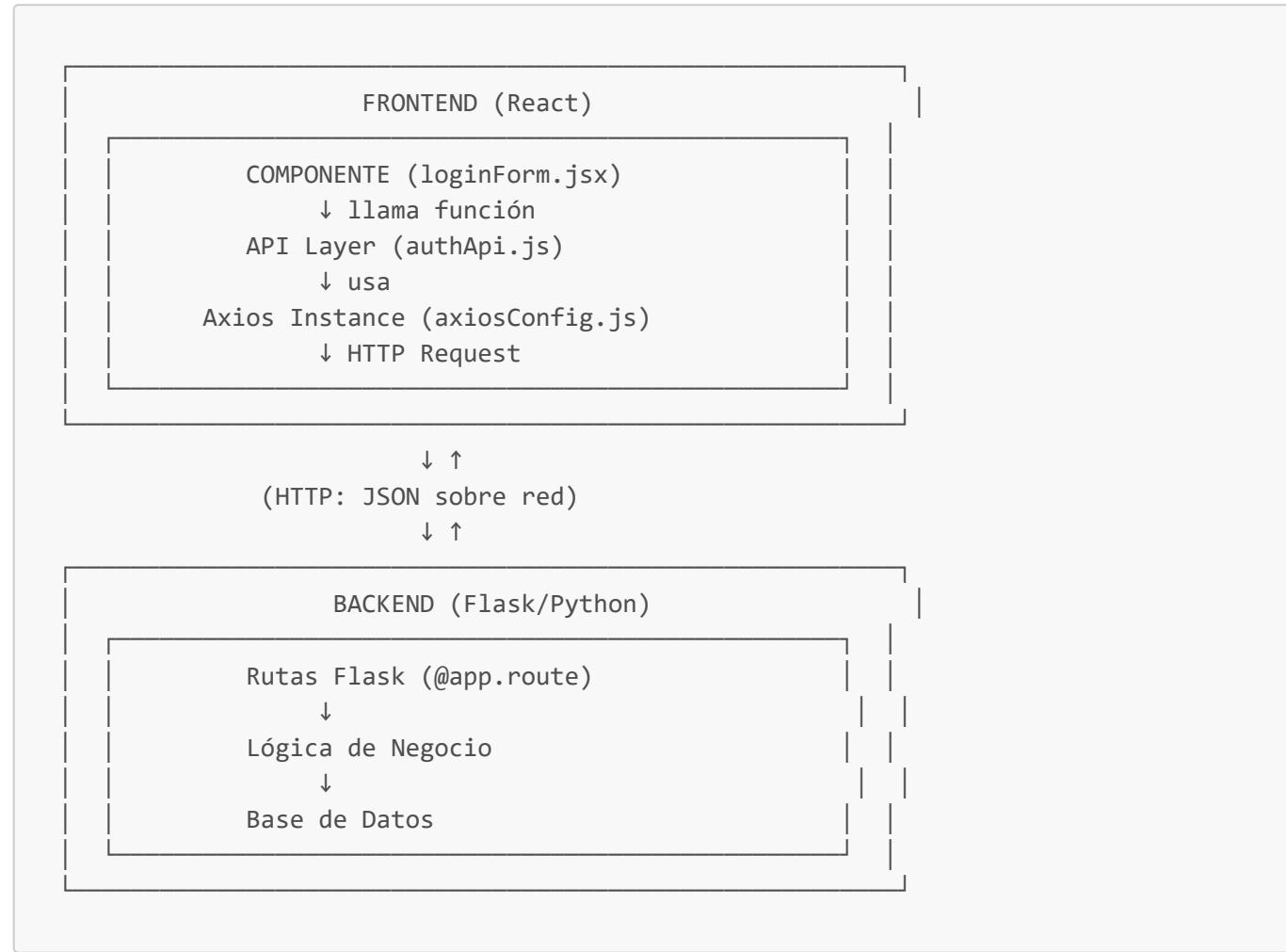
```
const [showTerms, setShowTerms] = useState(false);

<Modal
  isOpen={showTerms}
```

```
onClose={() => setShowTerms(false)}
title="Términos y Condiciones"
>
<TermsAndConditions />
</Modal>;
```

4. Comunicación Backend-Frontend

4.1 Arquitectura de Comunicación



4.2 Configuración Base - axiosConfig.js

```
import axios from "axios";

// Crear instancia personalizada
const api = axios.create({
  baseURL: "http://localhost:5000",
  headers: {
    "Content-Type": "application/json",
  },
});
```

```
// INTERCEPTOR: Se ejecuta antes de cada request
api.interceptors.request.use(
  (config) => {
    const token = localStorage.getItem("token");
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
  },
  (error) => Promise.reject(error)
);

// INTERCEPTOR: Se ejecuta después de cada response
api.interceptors.response.use(
  (response) => response,
  (error) => {
    if (error.response?.status === 401) {
      // Token expirado - redirigir a login
      localStorage.clear();
      window.location.href = "/login?expired=true";
    }
    return Promise.reject(error);
  }
);

export default api;
```

¿Por qué usar interceptors?

- DRY:** No repetir headers en cada request
 - Seguridad:** Token se agrega automáticamente
 - Manejo global de errores:** 401 → logout automático
-

4.3 Funciones API por Módulo

📁 api/authApi.js

```
import api from "./axiosConfig";

export const registerUser = async (userData) => {
  try {
    const response = await api.post("/api/registro", userData);
    // userData = { email, password, nombre, apellido, rol }
    return response.data;
  } catch (error) {
    throw error.response?.data || error.message;
  }
};

export const loginUser = async (credentials) => {
```

```
try {
  const response = await api.post("/api/login", credentials);
  // credentials = { email, password }
  return response.data;
  // Retorna: { token, id_usuario, rol, nombre, apellido }
} catch (error) {
  throw error.response?.data || error.message;
}
};
```

📁 api/vacancyApi.js

```
import api from "./axiosConfig";

export const getVacantes = async () => {
  try {
    const response = await api.get("/vacantes/");
    return response.data;
  } catch (error) {
    throw error.response?.data || error.message;
  }
};

export const crearVacante = async (vacancyData) => {
  try {
    const response = await api.post("/vacantes/crear", vacancyData);
    return response.data;
  } catch (error) {
    if (error.response?.status === 401) {
      throw new Error("Sesión expirada. Inicia sesión nuevamente.");
    }
    throw error.response?.data || error.message;
  }
};

export const getVacantePorId = async (id) => {
  try {
    const response = await api.get(`/vacantes/${id}`);
    return response.data;
  } catch (error) {
    throw error.response?.data || error.message;
  }
};
```

📁 api/companyApi.js

```

import api from "./axiosConfig";

export const saveCompanyProfile = async (companyData) => {
  try {
    const response = await api.post("/api/empresa/perfil", companyData);
    return response.data;
  } catch (error) {
    throw error.response?.data || error.message;
  }
};

export const getCompanyProfile = async (userId) => {
  try {
    const response = await api.get(`/api/empresa/perfil/${userId}`);
    return response.data;
  } catch (error) {
    throw error.response?.data || error.message;
  }
};

```

4.4 Flujo Completo de un Request

Ejemplo: Login de Usuario

1. USUARIO
Ingresa email y password en formulario
↓
2. COMPONENTE (loginForm.jsx)
handleSubmit() → llama loginUser(credentials)
↓
3. API (authApi.js)
export const loginUser = async (credentials) => {
 return api.post('/api/login', credentials);
}
↓
4. INTERCEPTOR (axiosConfig.js)
Agrega headers (si existe token previo)
↓
5. HTTP REQUEST
POST http://localhost:5000/api/login
Headers: { Content-Type: application/json }
Body: { email: "user@example.com", password: "123" }
↓

(VIAJA POR RED)

6. BACKEND FLASK
@app.route('/api/login', methods=['POST'])
def login():

```

data = request.get_json()
# Validar credenciales en DB
# Generar JWT token
return jsonify({
    "token": "eyJhbGc...",
    "id_usuario": 5,
    "rol": "Empresa",
    "nombre": "Juan"
}), 200

```

(VIAJA POR RED)

7. FRONTEND RECIBE RESPUESTA

```

const data = await loginUser(credentials);
// data = { token: "eyJ...", id_usuario: 5, ... }
↓

```

8. COMPONENTE PROCESA

```

login(data.token, data.id_usuario, data.rol, ...)
↓

```

9. AUTH CONTEXT

```

setSessionData() → Guarda en localStorage
setIsAuthenticated(true)
↓

```

10. NAVEGACIÓN

```

navigate('/company-dashboard')

```

4.5 Anatomía de un HTTP Request

```

await api.post('/api/login', { email: 'user@example.com', password: '123' })
          |           |
          |           └ BODY (request payload)
          |           |
          |           └ RUTA (se concatena con baseURL)
          |
          └ MÉTODO HTTP (GET, POST, PUT, DELETE)

```

Request HTTP resultante:

```

POST http://localhost:5000/api/login
Headers:
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
Body:
{
  "email": "user@example.com",
  "password": "123"
}

```

Response del Backend:

```

HTTP/1.1 200 OK
Content-Type: application/json

{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "id_usuario": 5,
  "rol": "Empresa",
  "nombre": "Juan",
  "apellido": "Pérez"
}

```

4.6 Tipos de Requests HTTP

Método	Propósito	Ejemplo FreeBridge	Archivo API
GET	Obtener datos	getVacantes()	vacancyApi.js
POST	Crear recurso	crearVacante()	vacancyApi.js
PUT	Actualizar completo	updateProfile()	profileApi.js
PATCH	Actualizar parcial	(no implementado)	-
DELETE	Eliminar recurso	(no implementado)	-

Ejemplos:

```

// GET con parámetros de query
const response = await api.get("/vacantes?ciudad=Bogotá&salario_min=2000");

// GET con parámetros de ruta
const response = await api.get(`/vacantes/${vacanteId}`);

// POST con datos
const response = await api.post("/vacantes/crear", {
  titulo: "Desarrollador React",
  descripcion: "...",
  salario: 3000,
});

// PUT (actualización completa)
const response = await api.put(`/usuario/${userId}`, {
  nombre: "Juan",
  apellido: "Pérez",
  email: "juan@example.com",
});

```

4.7 Manejo de Errores

En el Frontend

```
try {
  const data = await loginUser(credentials);
  // Éxito
  console.log("Login exitoso", data);
} catch (error) {
  // Error

  if (error.response) {
    // Backend respondió con error (4xx, 5xx)
    console.error("Error del servidor:", error.response.data);
    setError(error.response.data.message || "Error desconocido");
  } else if (error.request) {
    // Request enviado pero sin respuesta (backend caído)
    console.error("Sin respuesta del servidor:", error.request);
    setError("No se pudo conectar al servidor");
  } else {
    // Error al configurar request
    console.error("Error:", error.message);
    setError("Error inesperado");
  }
}
```

En el Backend (Flask)

```
@app.route('/api/login', methods=['POST'])
def login():
    try:
        data = request.get_json()

        # Validaciones
        if not data.get('email') or not data.get('password'):
            return jsonify({"error": "Campos incompletos"}), 400

        # Lógica de autenticación...

        return jsonify({
            "token": token,
            "id_usuario": user_id,
            "rol": rol
        }), 200

    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

FreeBridge utiliza **JSON Web Tokens (JWT)** para manejar la autenticación y autorización de usuarios de forma stateless y segura.

4.8.1 ¿Qué es un JWT y para qué sirve?

Un **JWT (JSON Web Token)** es un estándar abierto (RFC 7519) que define una forma compacta y autónoma de transmitir información de forma segura entre partes como un objeto JSON.

Ventajas en FreeBridge:

- **Stateless**: El servidor no necesita almacenar sesiones
- **Portable**: Se puede usar entre diferentes servicios
- **Seguro**: Firmado criptográficamente para evitar manipulación
- **Eficiente**: Reduce consultas a la BD en cada request
- **Escalable**: Ideal para arquitecturas distribuidas

Usos en FreeBridge:

1. Autenticar usuarios después del login
 2. Identificar al usuario en cada request sin consultar la BD
 3. Autorizar acceso a recursos según el rol (FreeLancer/Empresa)
 4. Controlar expiración de sesiones automáticamente
-

4.8.2 Estructura del Token JWT en FreeBridge

Ubicación de creación: `server-flask/routes/routes_auth/login.py` (líneas 39-47)

```
import jwt
from datetime import datetime, timedelta
from flask import current_app

# Generación del token después de validar credenciales
token = jwt.encode(
    {
        "user_id": usuario.id_usu,          # ID único del usuario en BD
        "rol": usuario.rol,                # "FreeLancer" o "Empresa"
        "exp": datetime.utcnow() + timedelta(hours=24) # Expiración: 24 horas
    },
    current_app.config["SECRET_KEY"],      # Clave secreta (definida en config.py)
    algorithm="HS256"                    # Algoritmo HMAC con SHA-256
)
```

Componentes del Payload:

Campo	Tipo	Descripción	Ejemplo
<code>user_id</code>	String	ID único del usuario en la tabla <code>usuarios</code>	"usr_5f8a2b3c"

Campo	Tipo	Descripción	Ejemplo
rol	String	Rol del usuario para autorización	"FreeLancer" o "Empresa"
exp	Timestamp	Fecha/hora de expiración (Unix timestamp)	1732060800 (24h después)

Configuración:

- Algoritmo:** HS256 (HMAC + SHA-256)
- Clave Secreta:** Variable SECRET_KEY en utils/config.py
- Tiempo de Vida:** 24 horas desde la creación
- Biblioteca:** PyJWT 2.8.0

4.8.3 Anatomía de un JWT Real

Un JWT consta de 3 partes separadas por puntos (.):

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyX2lkIjoidXNyXzVmOGEyYjNjIiwicm9sIjoiRnJlZXuhbmNlcisiImV4cCI6MTczMjA2MDgwMH0.Sf1KxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

1. Header (Encabezado)

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2. Payload (Datos)

```
{
  "user_id": "usr_5f8a2b3c",
  "rol": "FreeLancer",
  "exp": 1732060800
}
```

3. Signature (Firma)

```
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload),
  SECRET_KEY
)
```

4.8.4 Flujo Completo del Token en FreeBridge

1. LOGIN EXITOSO

Frontend → Backend: { correo, contraseña }

Backend valida en DB → Usuario encontrado ✓

Backend genera JWT:

```
{
  "user_id": "usr_abc123",
  "rol": "FreeLancer",
  "exp": 1732060800
}
```

Backend → Frontend:

```
{
  "token": "eyJhbGc...",
  "usuario": { "id", "nombre", "correo", "rol" }
}
```

2. FRONTEND ALMACENA TOKEN

loginForm.jsx recibe respuesta

llama login() del AuthContext

sessionManager.js guarda en localStorage:

- localStorage.setItem("token", "eyJhbGc...")
- localStorage.setItem("userId", "usr_abc123")
- localStorage.setItem("userRole", "FreeLancer")

3. REQUESTS POSTERIORES (Automático)

Usuario intenta crear vacante

axiosConfig interceptor detecta el request

Lee token: const token = getToken()

Agrega header automáticamente:

```
headers: { Authorization: "Bearer eyJhbGc..." }
```

Request enviado al backend ✓

4. BACKEND VALIDA TOKEN (Decorador @token_required)

utils/auth.py → token_required()

Extrae header: token = request.headers.get('Authorization')

Quita "Bearer ": token = token[7:]

Decodifica: data = jwt.decode(token, SECRET_KEY, ["HS256"])

Extrae user_id: current_user = Usuario.query.get(user_id)

Valida expiración: if exp < now → 401 Expired

Token válido ✓ → Ejecuta función protegida

5. CASOS DE ERROR

- Token expirado (>24h) → 401 "Token expirado"

- Token manipulado → 401 "Token inválido"

- Sin token → 401 "Token faltante"

- Usuario eliminado → 401 "Usuario no encontrado"

6. LOGOUT

Usuario hace clic en "Cerrar Sesión"

AuthContext.logout() → sessionManager.clearSession()

```
localStorage limpiado → Token eliminado  
Usuario redirigido a /login
```

4.8.5 Ejemplos Reales de Uso en FreeBridge

Ejemplo 1: Login y Generación de Token

Backend: routes/routes_auth/login.py

```
@login_bp.route("/api/login", methods=["POST"])
def login():
    data = request.get_json()
    correo = data.get("correo")
    contraseña = data.get("contraseña")

    # Buscar usuario
    usuario = Usuario.query.filter_by(correo=correo).first()

    if not usuario or not check_password_hash(usuario.contraseña, contraseña):
        return jsonify({"error": "Credenciales inválidas"}), 401

    # ☑ GENERAR TOKEN JWT
    token = jwt.encode(
        {
            "user_id": usuario.id_usu,
            "rol": usuario.rol,
            "exp": datetime.utcnow() + timedelta(hours=24)
        },
        current_app.config["SECRET_KEY"],
        algorithm="HS256"
    )

    # Retornar token y datos del usuario
    return jsonify({
        "mensaje": "Inicio de sesión exitoso",
        "token": token,
        "usuario": {
            "id": usuario.id_usu,
            "nombre": usuario.nombre,
            "correo": usuario.correo,
            "rol": usuario.rol
        }
    }), 200
```

Frontend: components/authComponents/LoginForm.jsx

```

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

  try {
    // Llamar API de login
    const res = await loginUser({ email, password });

    // ☑ GUARDAR TOKEN Y DATOS
    if (res.usuario) {
      login({
        token: res.token, // Token JWT
        userRole: res.usuario.rol, // Extraído del token
        userName: res.usuario.nombre,
        userId: res.usuario.id,
      });

      // Redirigir según rol
      if (res.usuario.rol === "Empresa") {
        navigate("/company-dashboard");
      } else {
        navigate("/freelance-dashboard");
      }
    }
  } catch (error) {
    setError(error.response?.data?.error || "Error al iniciar sesión");
  }
};

```

Ejemplo 2: Request Protegido con Token

Frontend: Crear Vacante

```

// api/vacancyApi.js
export const crearVacante = async (vacanteData) => {
  // ☑ El interceptor de axiosConfig agrega el token automáticamente
  const res = await api.post("/api/crear-vacantes", vacanteData);
  return res.data;
};

// axiosConfig.js - Interceptor
api.interceptors.request.use(
  (config) => {
    const token = getToken(); // Lee de localStorage
    if (token) {
      // ☑ AGREGA TOKEN A CADA REQUEST
      config.headers.Authorization = `Bearer ${token}`;
    }
    return config;
},

```

```
(error) => Promise.reject(error)
);
```

Backend: Validación de Token

```
# routes/routes_vacancy/crear_vacante.py
from utils.auth import token_required

@crear_vacante_bp.route("/api/crear-vacantes", methods=["POST"])
@token_required # ☑ DECORADOR QUE VALIDA EL TOKEN
def crear_vacante(current_user):
    # current_user es el objeto Usuario extraído del token
    data = request.get_json()

    # Verificar que el usuario es empresa
    empresa = Empresa.query.filter_by(id_usu=current_user.id_usu).first()
    if not empresa:
        return jsonify({"error": "Solo empresas pueden crear vacantes"}), 403

    # Crear vacante
    nueva_vacante = Vacante(
        nombre=data.get("nombre"),
        desc_vac=data.get("desc_vac"),
        salario=data.get("salario"),
        id_emp=empresa.id_emp, # ☑ Empresa extraída del token
        id_ciud=data.get("id_ciud"),
        estado_vac="abierta"
    )

    db.session.add(nueva_vacante)
    db.session.commit()

    return jsonify({"success": True, "vacante": nueva_vacante.to_dict()}), 201
```

Decorador: utils/auth.py

```
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = request.headers.get("Authorization")

        if not token:
            return jsonify({"error": "Token faltante"}), 401

        try:
            # ☑ EXTRAER TOKEN (quitar "Bearer ")
            if token.startswith("Bearer "):
                token = token[7:]

```

```

# ☑ DECODIFICAR Y VALIDAR TOKEN
data = jwt.decode(
    token,
    current_app.config["SECRET_KEY"],
    algorithms=["HS256"]
)

# ☑ OBTENER USUARIO DESDE EL PAYLOAD
current_user = Usuario.query.get(data["user_id"])

if not current_user:
    return jsonify({"error": "Usuario no encontrado"}), 401

except jwt.ExpiredSignatureError:
    return jsonify({"error": "Token expirado"}), 401
except jwt.InvalidTokenError:
    return jsonify({"error": "Token inválido"}), 401

# ☑ EJECUTAR FUNCIÓN PROTEGIDA CON EL USUARIO
return f(current_user, *args, **kwargs)

return decorated

```

Ejemplo 3: Verificar Rol desde Token

Postulación a Vacante (Solo FreeLancers)

```

# routes/routes_post/postulacion.py
@postulacion_bp.route("/api/postular/<id_vac>", methods=["POST"])
@token_required
def postular_vacante(current_user, id_vac):
    # ☑ VERIFICAR ROL DESDE EL TOKEN
    if current_user.rol != "FreeLancer":
        return jsonify({
            "error": "Solo freelancers pueden postularse"
        }), 403

    # Buscar freelancer
    freelancer = Freelancer.query.filter_by(id_usu=current_user.id_usu).first()

    if not freelancer:
        return jsonify({"error": "Perfil de freelancer no encontrado"}), 404

    # Crear postulación
    postulacion = Postulacion(
        id_vac=id_vac,
        id_free=freelancer.id_free,  # ☑ ID extraído del token
        estado_post="pendiente"
    )

```

```
db.session.add(postulacion)
db.session.commit()

return jsonify({"success": True, "message": "Postulación exitosa"}), 201
```

Ejemplo 4: Manejo de Token Expirado

Frontend: Interceptor de Respuesta

```
// axiosConfig.js
api.interceptors.response.use(
  (response) => response,
  (error) => {
    // ☑ DETECTAR TOKEN EXPIRADO
    if (error.response?.status === 401) {
      const errorMessage = error.response?.data?.error;

      if (
        errorMessage === "Token expirado" ||
        errorMessage === "Token inválido"
      ) {
        // Limpiar sesión
        clearSession();

        // Redirigir a login
        window.location.href = "/login";

        return Promise.reject(
          new Error("Sesión expirada. Por favor, inicia sesión nuevamente.")
        );
      }
    }

    return Promise.reject(error);
  }
);
```

4.8.6 Seguridad y Buenas Prácticas

Implementado en FreeBridge:

- Token firmado con SECRET_KEY para prevenir manipulación
- Expiración de 24 horas para limitar ventana de ataque
- Validación de usuario en cada request protegido
- Token almacenado solo en localStorage (no en cookies)
- Interceptor centralizado para agregar token automáticamente

⚠ Consideraciones para Producción:

- Cambiar `SECRET_KEY` a valor seguro (256 bits mínimo)
 - Implementar HTTPS obligatorio
 - Considerar Refresh Tokens para renovación sin re-login
 - Agregar blacklist para tokens revocados
 - Implementar rate limiting en endpoints de autenticación
 - Usar cookies HttpOnly + SameSite para mayor seguridad (alternativa a localStorage)
-

4.8.7 Debugging de Tokens

Ver contenido del token (sin validar):

```
// Frontend - Decodificar token (solo para debug, no validación)
const token = localStorage.getItem("token");
const payload = JSON.parse(atob(token.split(".")[1]));
console.log("Payload del token:", payload);
// Output: { user_id: "usr_123", rol: "FreeLancer", exp: 1732060800 }
```

Verificar expiración:

```
const payload = JSON.parse(atob(token.split(".")[1]));
const expDate = new Date(payload.exp * 1000);
console.log("Token expira:", expDate.toLocaleString());
// Output: "Token expira: 20/11/2025, 10:30:00"
```

5. Sistema de Autenticación

5.1 Flujo Completo de Autenticación

1. Usuario llena formulario en `loginForm.jsx`
↓
2. Componente llama `loginUser(credentials)`
↓
3. `authApi.js` hace POST a `/api/login`
↓
4. Backend valida credenciales en DB
↓
5. Backend genera JWT token
↓
6. Backend retorna `{ token, id_usuario, rol, nombre }`
↓
7. `loginForm` llama `login()` del `AuthContext`
↓

8. AuthContext llama setSessionData() de utils
↓
9. sessionManager guarda en localStorage
↓
10. Estado global se actualiza (isAuthenticated=true)
↓
11. Navbar re-renderiza (muestra "Perfil/Logout")
↓
12. useSessionTimeout inicia monitoreo de inactividad
↓
13. Usuario redirigido a dashboard según rol

5.2 Registro de Usuario

1. Usuario selecciona rol (Freelancer/Empresa)
↓
2. Llena formulario de registro
↓
3. Acepta términos y condiciones
↓
4. registerForm llama registerUser(userData)
↓
5. authApi.js hace POST a /api/registro
↓
6. Backend crea usuario en DB
↓
7. Backend retorna { message, id_usuario }
↓
8. Frontend muestra SuccessModal
↓
9. Usuario redirigido a /login

5.3 Protección de Rutas

```
// Ejemplo de componente protegido
import { useAuth } from "../context/AuthContext";
import { useNavigate } from "react-router-dom";
import { useEffect } from "react";

const CompanyDashboard = () => {
  const { isAuthenticated, userRole } = useAuth();
  const navigate = useNavigate();

  useEffect(() => {
    if (!isAuthenticated) {
      navigate("/login");
    }
  }, [isAuthenticated]);
}

CompanyDashboard.propTypes = {
  history: PropTypes.object,
};

export default CompanyDashboard;
```

```

    } else if (userRole !== "Empresa") {
      navigate("/");
    }
  }, [isAuthenticated, userRole, navigate]);

  if (!isAuthenticated || userRole !== "Empresa") {
    return null; // o un spinner
  }

  return <div>{/* Contenido del dashboard */}</div>;
};


```

6. Sistema de Archivos y Uploads

6.1 Arquitectura de Uploads

FreeBridge implementa un sistema completo de gestión de archivos para:

- **Hojas de vida (CVs)**: PDFs de freelancers
- **Avatares**: Imágenes de perfil de freelancers
- **Logos**: Imágenes de empresas

Backend Storage:

```

server-flask/
└── uploads/
    ├── hojas_vida/      # CVs de freelancers
    ├── avatares/        # Fotos de perfil
    └── logos/           # Logos de empresas

```

6.2 Endpoints de Archivos

Servir Archivos (GET)

```

// Backend: routes/archivos.py
GET /api/uploads/hojas_vida/<filename>
GET /api/uploads/avatares/<filename>
GET /api/uploads/logos/<filename>

```

Ejemplo:

```

// Frontend: Mostrar avatar
<img
  src={`http://localhost:5000/api/uploads/avatares/${avatar}`}
  alt="Avatar"
/>

```

Subir Archivos (POST/PUT)

Freelancer - Subir hoja de vida y avatar:

```
// routes/routes_perfil/perfil_freelancer.py
POST /api/freelancer/perfil
PUT /api/freelancer/perfil/<id_free>

// Frontend: FormData
const formData = new FormData();
formData.append('profesion', 'Desarrollador');
formData.append('experiencia', 'Texto... ');
formData.append('id_ciud', ciudadId);
formData.append('id_usu', userId);
formData.append('hoja_vida', pdfFile);           // PDF
formData.append('avatar', imageFile);            // Imagen
```

Empresa - Subir logo:

```
// routes/routes_empresa/perfil_empresa.py
POST /api/empresa/perfil
PUT /api/empresa/perfil/<id_emp>

const formData = new FormData();
formData.append('NIT', '123456');
formData.append('tamaño', 'Mediana');
formData.append('desc_emp', 'Descripción... ');
formData.append('logo', logoFile);                // Imagen
```

6.3 Validaciones de Archivos

Backend (Flask)

```
# perfil_freelancer.py
ALLOWED_EXTENSIONS = {'pdf'}
ALLOWED_IMAGE_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif', 'webp'}

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def allowed_image(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_IMAGE_EXTENSIONS
```

Nombres únicos:

```
from werkzeug.utils import secure_filename
import uuid

filename = secure_filename(file.filename)
unique_filename = f"{user_id}_{uuid.uuid4().hex[:8]}_{filename}"
filepath = os.path.join(UPLOAD_FOLDER, unique_filename)
file.save(filepath)
```

Frontend (React)

```
// Validar antes de enviar
const validateFile = (file, type) => {
  const maxSize = type === "pdf" ? 5 * 1024 * 1024 : 2 * 1024 * 1024; // 5MB / 2MB

  if (file.size > maxSize) {
    throw new Error(`Archivo muy grande (máx ${maxSize / 1024 / 1024}MB)`);
  }

  const allowedTypes =
    type === "pdf"
    ? ["application/pdf"]
    : ["image/png", "image/jpeg", "image/jpg", "image/gif", "image/webp"];

  if (!allowedTypes.includes(file.type)) {
    throw new Error("Tipo de archivo no permitido");
  }
};
```

6.4 Eliminación de Archivos**Al actualizar perfil:**

```
# Eliminar archivo anterior si existe
if freelancer.hoja_vida and os.path.exists(freelancer.hoja_vida):
    try:
        os.remove(freelancer.hoja_vida)
    except:
        pass # Continuar si falla
```

Al eliminar cuenta:

```
# routes/routes_auth/eliminar_cuenta.py
@eliminar_cuenta_bp.route('/api/usuario/eliminar', methods=['DELETE'])
```

```
@token_required
def eliminar_cuenta(current_user):
    # Eliminar archivos físicos
    if freelancer.hoja_vida and os.path.exists(freelancer.hoja_vida):
        os.remove(freelancer.hoja_vida)

    if freelancer.avatar and os.path.exists(freelancer.avatar):
        os.remove(freelancer.avatar)

    # Eliminar registros de BD
    db.session.delete(freelancer)
    db.session.delete(current_user)
    db.session.commit()
```

6.5 Frontend - Componentes de Upload

```
// EditFreelancerProfile.jsx
const [selectedCV, setSelectedCV] = useState(null);
const [selectedAvatar, setSelectedAvatar] = useState(null);

const handleCVChange = (e) => {
  const file = e.target.files[0];
  if (file) {
    try {
      validateFile(file, "pdf");
      setSelectedCV(file);
    } catch (error) {
      alert(error.message);
    }
  }
};

const handleAvatarChange = (e) => {
  const file = e.target.files[0];
  if (file) {
    try {
      validateFile(file, "image");
      setSelectedAvatar(file);
      // Preview
      const reader = new FileReader();
      reader.onload = (e) => setAvatarPreview(e.target.result);
      reader.readAsDataURL(file);
    } catch (error) {
      alert(error.message);
    }
  }
};

const handleSubmit = async () => {
  const formData = new FormData();
  formData.append("profesion", profesion);
```

```

    formData.append("experiencia", experiencia);
    if (selectedCV) formData.append("hoja_vida", selectedCV);
    if (selectedAvatar) formData.append("avatar", selectedAvatar);

    await updateFreelancerProfile(freelancerId, formData);
};


```

7. Sistema de Notificaciones y Email

7.1 Configuración Flask-Mail

Backend: `utils/config.py`

```

from flask_mail import Mail

class Config:
    # Configuración Mailtrap (desarrollo)
    MAIL_SERVER = os.environ.get('MAIL_SERVER', 'sandbox.smtp.mailtrap.io')
    MAIL_PORT = int(os.environ.get('MAIL_PORT', 2525))
    MAIL_USERNAME = os.environ.get('MAIL_USERNAME', '')
    MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD', '')
    MAIL_USE_TLS = os.environ.get('MAIL_USE_TLS', 'True') == 'True'
    MAIL_USE_SSL = os.environ.get('MAIL_USE_SSL', 'False') == 'True'
    MAIL_DEFAULT_SENDER = os.environ.get('MAIL_DEFAULT_SENDER',
                                         'noreply@freebridge.com')


```

Backend: `app.py`

```

from flask_mail import Mail

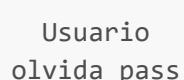
app = Flask(__name__)
app.config.from_object(Config)
mail = Mail(app)

# Inyectar mail a blueprints que lo necesitan
password_reset_bp.mail = mail
actualizar_estado_bp.mail = mail


```

7.2 Recuperación de Contraseña

Flujo Completo



Usuario
olvida pass



Código Backend

```

# routes/routes_auth/password_reset.py
from flask_mail import Message
from models.modelo_password_reset import PasswordResetToken

@password_reset_bp.route('/api/auth/forgot-password', methods=['POST'])
def forgot_password():
    data = request.get_json()
    email = data.get('email')

    usuario = Usuario.query.filter_by(correo=email).first()
    if not usuario:
        # No revelar si el email existe (seguridad)
        return jsonify({'message': 'Si el correo existe, recibirás un enlace'}), 200

    # Generar token
    reset_token = str(uuid.uuid4())
    reset_id = str(uuid.uuid4())[:10]

    password_reset = PasswordResetToken(
        id_reset=reset_id,
        email=email,
        token=reset_token,
        expires_in_hours=1 # Válido por 1 hora
  
```

```

    )
db.session.add(password_reset)
db.session.commit()

# Construir link
reset_link = f"http://localhost:5173/reset-password?token={reset_token}"

# Enviar email
msg = Message(
    subject='Recuperación de Contraseña - FreeBridge',
    recipients=[email],
    html=f"""
<h2>Recuperación de Contraseña</h2>
<p>Haz clic en el botón para restablecer tu contraseña:</p>
<a href="{reset_link}">Restablecer Contraseña</a>
<p>Este enlace expirará en 1 hora.</p>
"""
)
password_reset_bp.mail.send(msg)

return jsonify({'message': 'Email enviado'}), 200

```

Modelo de Token:

```

# models/modelo_password_reset.py
from datetime import datetime, timedelta

class PasswordResetToken(db.Model):
    __tablename__ = 'password_reset_tokens'

    id_reset = db.Column(db.String(36), primary_key=True)
    email = db.Column(db.String(100), nullable=False)
    token = db.Column(db.String(255), unique=True, nullable=False)
    expires_at = db.Column(db.DateTime, nullable=False)
    used = db.Column(db.Boolean, default=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    def __init__(self, id_reset, email, token, expires_in_hours=1):
        self.id_reset = id_reset
        self.email = email
        self.token = token
        self.expires_at = datetime.utcnow() + timedelta(hours=expires_in_hours)
        self.used = False

    def is_valid(self):
        """Verifica si el token es válido (no usado y no expirado)"""
        return not self.used and datetime.utcnow() < self.expires_at

    def mark_as_used(self):
        """Marca el token como usado"""
        self.used = True
        db.session.commit()

```

Código Frontend

```
// pages/ForgotPassword.jsx
import { useState } from "react";
import { forgotPassword } from "../api/authApi";

const ForgotPassword = () => {
  const [email, setEmail] = useState("");
  const [message, setMessage] = useState("");

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await forgotPassword({ email });
      setMessage("Si el correo existe, recibirás un enlace de recuperación");
    } catch (error) {
      setMessage("Error al procesar la solicitud");
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <input
        type="email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
        placeholder="Tu correo electrónico"
        required
      />
      <button type="submit">Recuperar Contraseña</button>
      {message && <p>{message}</p>}
    </form>
  );
};


```

```
// pages/ResetPassword.jsx
import { useState, useEffect } from "react";
import { useSearchParams, useNavigate } from "react-router-dom";
import { resetPassword } from "../api/authApi";

const ResetPassword = () => {
  const [searchParams] = useSearchParams();
  const [password, setPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const navigate = useNavigate();
  const token = searchParams.get("token");

  const handleSubmit = async (e) => {


```

```

e.preventDefault();

if (password !== confirmPassword) {
  alert("Las contraseñas no coinciden");
  return;
}

try {
  await resetPassword({ token, password });
  alert("Contraseña actualizada exitosamente");
  navigate("/login");
} catch (error) {
  alert("Token inválido o expirado");
}
};

return (
  <form onSubmit={handleSubmit}>
    <input
      type="password"
      value={password}
      onChange={(e) => setPassword(e.target.value)}
      placeholder="Nueva contraseña"
      minLength={6}
      required
    />
    <input
      type="password"
      value={confirmPassword}
      onChange={(e) => setConfirmPassword(e.target.value)}
      placeholder="Confirmar contraseña"
      required
    />
    <button type="submit">Restablecer Contraseña</button>
  </form>
);
};

```

7.3 Notificaciones de Postulaciones

Para Freelancers (Estado de postulación)

Backend: Envío automático al aceptar/rechazar

```

# routes/routes_post/actualizar_estado.py
@actualizar_estado_bp.route('/api/postulacion/estado/<id_post>', methods=['PUT'])
@token_required
def actualizar_estado_postulacion(current_user, id_post):
  data = request.get_json()
  nuevo_estado = data.get('estado') # 'aceptada' | 'rechazada'

```

```

postulacion = Postulacion.query.filter_by(id_post=id_post).first()
freelancer = Freelancer.query.filter_by(id_free=postulacion.id_free).first()
vacante = Vacante.query.filter_by(id_vac=postulacion.id_vac).first()

# Actualizar estado
postulacion.estado_post = nuevo_estado
db.session.commit()

# Enviar email al freelancer
enviar_email_notificacion(freelancer, vacante, nuevo_estado)

return jsonify({'success': True}), 200

def enviar_email_notificacion(freelancer, vacante, estado):
    usuario = freelancer.usuario

    if estado == 'aceptada':
        asunto = f'¡Felicitaciones! Tu postulación fue aceptada - {vacante.nombr_vacante}'
        html_body = f"""
        <h1>¡Felicitaciones {usuario.nombre}!</h1>
        <p>Tu postulación ha sido <strong>aceptada</strong>.</p>
        <h3>Detalles de la Vacante:</h3>
        <p><strong>Puesto:</strong> {vacante.nombr_vacante}</p>
        <p><strong>Descripción:</strong> {vacante.descripcion}</p>
        <p>La empresa se pondrá en contacto contigo pronto.</p>
        """
    else:
        asunto = f'Actualización sobre tu postulación - {vacante.nombr_vacante}'
        html_body = f"""
        <h1>Actualización de Postulación</h1>
        <p>Gracias por tu interés en {vacante.nombr_vacante}.</p>
        <p>En esta ocasión decidimos continuar con otros candidatos.</p>
        <p>Te animamos a seguir explorando otras oportunidades.</p>
        """

    msg = Message(subject=asunto, recipients=[usuario.correo], html=html_body)
    actualizar_estado_bp.mail.send(msg)

```

Para Empresas (Nuevas postulaciones)

```

# routes/routes_post/notificaciones_empresa.py
@notificaciones_empresa_bp.route('/api/empresa/notificaciones/nuevas-postulaciones', methods=['GET'])
@token_required
def nuevas_postulaciones(current_user):
    empresa = Empresa.query.filter_by(id_usu=current_user.id_usu).first()

    # Obtener postulaciones pendientes
    vacantes = Vacante.query.filter_by(id_emp=empresa.id_emp).all()
    vacantes_ids = [v.id_vac for v in vacantes]

```

```

postulaciones_pendientes = Postulacion.query.filter(
    Postulacion.id_vac.in_(vacantes_ids),
    Postulacion.estado_post == 'pendiente'
).order_by(Postulacion.fecha_post.desc()).all()

notificaciones = []
for p in postulaciones_pendientes:
    freelancer = Freelancer.query.filter_by(id_free=p.id_free).first()
    vacante = Vacante.query.filter_by(id_vac=p.id_vac).first()

    notificaciones.append({
        'id': p.id_post,
        'fecha': p.fecha_post.isoformat(),
        'vacante': {'id': vacante.id_vac, 'nombre': vacante.nomb_vacante},
        'freelancer': {
            'nombre': freelancer.usuario.nombre,
            'email': freelancer.usuario.correo
        }
    })

return jsonify({
    'success': True,
    'notificaciones': notificaciones,
    'total': len(notificaciones)
}), 200

```

Frontend: Polling de notificaciones

```

// hooks/useNotifications.js
import { useEffect, useState } from "react";
import {
    getNuevasPostulacionesEmpresa,
    getCambiosRecientes,
} from "../api/postApi";
import { useAuth } from "../context/AuthContext";

export const useNotifications = (pollInterval = 60000) => {
    // 1 minuto
    const [notifications, setNotifications] = useState([]);
    const { userRole, isAuthenticated } = useAuth();

    useEffect(() => {
        if (!isAuthenticated) return;

        const fetchNotifications = async () => {
            try {
                if (userRole === "Empresa") {
                    const data = await getNuevasPostulacionesEmpresa();
                    setNotifications(data.notificaciones || []);
                } else if (userRole === "FreeLancer") {
                    const data = await getCambiosRecientes();

```

```
        setNotifications(data.cambios || []);
    }
} catch (error) {
    console.error("Error al obtener notificaciones:", error);
}
};

fetchNotifications();
const interval = setInterval(fetchNotifications, pollInterval);

return () => clearInterval(interval);
}, [userRole, isAuthenticated, pollInterval]);

return { notifications, count: notifications.length };
};
```

8. Gestión de Perfiles (Avatares y Logos)

8.1 Sistema de Avatares con DiceBear

FreeBridge usa **DiceBear** para generar avatares por defecto cuando el usuario no sube una imagen personalizada.

Instalación:

```
npm install @dicebear/core @dicebear/collection
```

Uso en Frontend:

```
// utils/avatarGenerator.js
import { createAvatar } from "@dicebear/core";
import { avataars, initials } from "@dicebear/collection";

export const generateAvatar = (seed, style = "avataars") => {
    const avatar = createAvatar(avataars, {
        seed: seed, // Usa nombre o email como seed
        size: 128,
    });

    return avatar.toDataUri(); // Retorna data:image/svg+xml;base64,...
};

export const generateInitialsAvatar = (name) => {
    const avatar = createAvatar(initials, {
        seed: name,
        backgroundColor: ["16a085", "16685a", "3498db", "e74c3c", "f39c12"],
    });
}
```

```
    return avatar.toDataUri();
};
```

Componente de Avatar:

```
// components/profileComponents/FreelancerAvatar.jsx
import { useState, useEffect } from "react";
import { generateAvatar } from "../../utils/avatarGenerator";
import styles from "../../styles/modules_profile/Avatar.module.css";

const FreelancerAvatar = ({
  avatar,
  userName,
  editable = false,
  onAvatarChange,
}) => {
  const [avatarSrc, setAvatarSrc] = useState("");

  useEffect(() => {
    if (avatar && avatar.startsWith("http")) {
      // Avatar subido (URL)
      setAvatarSrc(`http://localhost:5000/api/uploads/avatares/${avatar}`);
    } else if (avatar && avatar.startsWith("data:")) {
      // Avatar DiceBear (data URI)
      setAvatarSrc(avatar);
    } else {
      // Generar avatar por defecto
      setAvatarSrc(generateAvatar(userName || "default"));
    }
  }, [avatar, userName]);

  const handleFileChange = (e) => {
    const file = e.target.files[0];
    if (file && onAvatarChange) {
      // Preview local
      const reader = new FileReader();
      reader.onload = (e) => setAvatarSrc(e.target.result);
      reader.readAsDataURL(file);

      // Notificar cambio al padre
      onAvatarChange(file);
    }
  };
}

return (
  <div className={styles.avatarContainer}>
    <img src={avatarSrc} alt="Avatar" className={styles.avatar} />

    {editable && (
      <label className={styles.uploadLabel}>
        <input
          type="file"
    
```

```

        accept="image/*"
        onChange={handleFileChange}
        style={{ display: "none" }}
      />
      <span>📷 Cambiar foto</span>
    </label>
  )
);
};

export default FreelancerAvatar;

```

8.2 Selector de Avatares Predefinidos

```

// components/profileComponents/AvatarSelector.jsx
import { useState } from "react";
import { generateAvatar } from "../../utils/avatarGenerator";
import styles from "../../styles/modules_profile/AvatarSelector.module.css";

const AvatarSelector = ({ onSelect }) => {
  const avatarStyles = ["avataaars", "bottts", "personas", "lorelei"];
  const seeds = ["felix", "aneka", "john", "mary", "alex"];

  const [selectedAvatar, setSelectedAvatar] = useState(null);

  const handleSelect = (style, seed) => {
    const avatarUri = generateAvatar(seed, style);
    setSelectedAvatar(avatarUri);
    onSelect(avatarUri); // Enviar al componente padre
  };

  return (
    <div className={styles.grid}>
      {seeds.map((seed) => (
        <div
          key={seed}
          className={styles.avatarOption}
          onClick={() => handleSelect("avataaars", seed)}
        >
          <img src={generateAvatar(seed, "avataaars")} alt={seed} />
        </div>
      ))}
    </div>
  );
};

export default AvatarSelector;

```

8.3 Logos de Empresas

Componente de Logo:

```
// components/profileComponents/CompanyLogo.jsx
const CompanyLogo = ({ logo, companyName, editable = false, onLogoChange }) => {
  const [logoSrc, setLogoSrc] = useState("");

  useEffect(() => {
    if (logo) {
      setLogoSrc(`http://localhost:5000/api/uploads/logos/${logo}`);
    } else {
      // Logo placeholder con iniciales
      setLogoSrc(generateInitialsAvatar(companyName || "Empresa"));
    }
  }, [logo, companyName]);

  const handleFileChange = (e) => {
    const file = e.target.files[0];
    if (file && onLogoChange) {
      const reader = new FileReader();
      reader.onload = (e) => setLogoSrc(e.target.result);
      reader.readAsDataURL(file);
      onLogoChange(file);
    }
  };
}

return (
  <div className={styles.logoContainer}>
    <img src={logoSrc} alt="Logo empresa" className={styles.logo} />

    {editable && (
      <label className={styles.uploadLabel}>
        <input
          type="file"
          accept="image/*"
          onChange={handleFileChange}
          style={{ display: "none" }}
        />
        <span>  Cambiar logo</span>
      </label>
    )}
  </div>
);
};
```

8.4 Perfiles Completos

Freelancer Profile

```
// components/profileComponents/FreelancerProfileForm.jsx
import { useState } from "react";
```

```
import { saveFreelancerProfile } from "../../api/freelancerApi";
import FreelancerAvatar from "./FreelancerAvatar";
import AvatarSelector from "./AvatarSelector";

const FreelancerProfileForm = ({ userId, userName, onSuccess }) => {
  const [profesion, setProfesion] = useState("");
  const [experiencia, setExperiencia] = useState("");
  const [ciudadId, setCiudadId] = useState("");
  const [hojaVida, setHojaVida] = useState(null);
  const [avatar, setAvatar] = useState(null);
  const [avatarDefault, setAvatarDefault] = useState(null);
  const [showAvatarSelector, setShowAvatarSelector] = useState(false);

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

    const formData = new FormData();
    formData.append("id_usu", userId);
    formData.append("profesion", profesion);
    formData.append("experiencia", experiencia);
    formData.append("id_ciud", ciudadId);

    if (hojaVida) {
      formData.append("hoja_vida", hojaVida);
    }

    if (avatar) {
      formData.append("avatar", avatar); // Archivo subido
    } else if (avatarDefault) {
      formData.append("avatar_default", avatarDefault); // Data URI DiceBear
    }

    try {
      await saveFreelancerProfile(formData);
      onSuccess();
    } catch (error) {
      alert("Error al guardar perfil: " + error.message);
    }
  };
};

return (
  <form onSubmit={handleSubmit}>
    <h2>Completa tu Perfil de Freelancer</h2>

    {/* Avatar */}
    <div>
      <FreelancerAvatar
        avatar={avatar || avatarDefault}
        userName={userName}
        editable
        onAvatarChange={setAvatar}
      />
      <button
        type="button"
      >
    
```

```
    onClick={() => setShowAvatarSelector(!showAvatarSelector)}
  >
  Elegir avatar prediseñado
</button>

{showAvatarSelector && (
  <AvatarSelector
    onSelect={(uri) => {
      setAvatarDefault(uri);
      setShowAvatarSelector(false);
    }}
  />
)
</div>

{/* Campos del formulario */}
<input
  type="text"
  value={profesion}
  onChange={(e) => setProfesion(e.target.value)}
  placeholder="Profesión (ej: Desarrollador Web)"
  required
/>

<textarea
  value={experiencia}
  onChange={(e) => setExperiencia(e.target.value)}
  placeholder="Experiencia (ej: 3 años en React...)"
  rows={5}
/>

<select
  value={ciudadId}
  onChange={(e) => setCiudadId(e.target.value)}
  required
>
  <option value="">Selecciona tu ciudad</option>
  {/* Opciones de ciudades */}
</select>

{/* Hoja de vida */}
<div>
  <label>Subir Hoja de Vida (PDF)</label>
  <input
    type="file"
    accept=".pdf"
    onChange={(e) => setHojaVida(e.target.files[0])}
  />
</div>

  <button type="submit">Guardar Perfil</button>
</form>
);

};
```

Company Profile

```
// components/profileComponents/CompanyProfileForm.jsx
import { useState } from "react";
import { saveCompanyProfile } from "../../api/companyApi";
import CompanyLogo from "./CompanyLogo";

const CompanyProfileForm = ({ userId, companyName, onSuccess }) => {
  const [nit, setNit] = useState("");
  const [tamaño, setTamaño] = useState("");
  const [descripcion, setDescripcion] = useState("");
  const [ciudadId, setCiudadId] = useState("");
  const [logo, setLogo] = useState(null);

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

    const formData = new FormData();
    formData.append("id_usu", userId);
    formData.append("NIT", nit);
    formData.append("tamaño", tamaño);
    formData.append("desc_emp", descripcion);
    formData.append("id_ciud", ciudadId);

    if (logo) {
      formData.append("logo", logo);
    }

    try {
      await saveCompanyProfile(formData);
      onSuccess();
    } catch (error) {
      alert("Error al guardar perfil: " + error.message);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <h2>Completa tu Perfil de Empresa</h2>

      <CompanyLogo
        logo={logo}
        companyName={companyName}
        editable
        onLogoChange={setLogo}
      />

      <input
        type="text"
        value={nit}
      >
    
```

```
        onChange={(e) => setNit(e.target.value)}
        placeholder="NIT"
        required
    />

    <select
        value={tamaño}
        onChange={(e) => setTamaño(e.target.value)}
        required
    >
        <option value="">Tamaño de empresa</option>
        <option value="Pequeña">Pequeña (1-50 empleados)</option>
        <option value="Mediana">Mediana (51-200 empleados)</option>
        <option value="Grande">Grande (200+ empleados)</option>
    </select>

    <textarea
        value={descripcion}
        onChange={(e) => setDescripcion(e.target.value)}
        placeholder="Descripción de la empresa"
        rows={5}
        required
    />

    <select
        value={ciudadId}
        onChange={(e) => setCiudadId(e.target.value)}
        required
    >
        <option value="">Selecciona ciudad</option>
        {/* Opciones */}
    </select>

    <button type="submit">Guardar Perfil</button>
</form>
);
};
```

9. Guía de Desarrollo

Nueva Feature Completa

- Diseñar endpoint en backend
- Crear función en archivo API
- Crear/modificar componente
- Crear estilos CSS Module
- Agregar ruta (si es página)
- Testear en navegador
- Verificar en Network Tab
- Manejar errores

- Agregar loading states
 - Documentar código
-

8.6 Recursos Adicionales

Documentación Oficial:

- [React Docs](#)
- [React Router](#)
- [Axios](#)
- [Vite](#)

Archivos de Referencia Interna:

- [ARQUITECTURA_AUTH.md](#) - Detalles de autenticación
 - [GUIA_ESTILOS.md](#) - Convenciones de CSS
-

10. Debugging y Testing

10.1 Herramientas de Debugging

Browser DevTools - Network Tab

```
// Ver requests en tiempo real
1. Abrir DevTools (F12)
2. Ir a Network tab
3. Filtrar por "XHR" o "Fetch"
4. Inspeccionar request/response:
   - Headers (ver Authorization token)
   - Payload (datos enviados)
   - Preview (respuesta formateada)
   - Response (raw JSON)
```

React DevTools

1. Instalar extensión React DevTools
2. Ver árbol de componentes
3. Inspeccionar props y state
4. Profiler para performance

Backend Logging

```
# En cualquier endpoint
import traceback
```

```

try:
    # código
except Exception as e:
    print(f"X Error: {str(e)}")
    traceback.print_exc()
    return jsonify({'error': str(e)}), 500

```

10.2 Testing de Endpoints

Usando Postman/Thunder Client

Colección de pruebas:

```
{
  "name": "FreeBridge API Tests",
  "requests": [
    {
      "name": "Health Check",
      "method": "GET",
      "url": "http://localhost:5000/api/health"
    },
    {
      "name": "Login",
      "method": "POST",
      "url": "http://localhost:5000/api/login",
      "body": {
        "correo": "test@example.com",
        "contraseña": "123456"
      }
    },
    {
      "name": "Get Vacantes",
      "method": "GET",
      "url": "http://localhost:5000/api/vacantes"
    }
  ]
}
```

10.3 Errores Comunes y Soluciones

Error	Causa	Solución
401 Unauthorized	Token faltante o inválido	Verificar que el token se envía en header <code>Authorization: Bearer <token></code>
CORS Error	Backend no permite el origen	Verificar configuración CORS en <code>app.py</code>
404 Not Found	Endpoint incorrecto	Verificar URL en Network tab y comparar con backend

Error	Causa	Solución
500 Internal Server Error	Error en backend	Revisar logs de consola del servidor Flask
Network Error	Backend no está corriendo	Verificar que Flask está en http://localhost:5000
File too large	Archivo excede límite	Verificar <code>MAX_CONTENT_LENGTH</code> en Flask config

10.4 Checklist de Implementación

Nueva Feature Completa

- Diseñar endpoint en backend
- Crear modelo de datos si es necesario
- Crear función en archivo API (frontend)
- Crear/modificar componente
- Crear estilos CSS Module
- Agregar ruta si es página
- Implementar manejo de errores
- Agregar loading states
- Testear en navegador
- Verificar en Network Tab
- Probar con datos reales
- Documentar código

11. Anexos

11.1 Dependencias Completas

Backend ([requirements.txt](#))

```
# Core Framework
Flask==3.1.2
flask-cors==6.0.1

# Database
Flask-SQLAlchemy==3.1.1
PyMySQL==1.1.0

# Authentication & Security
PyJWT==2.8.0
werkzeug==3.0.1

# Email
Flask-Mail==0.9.1

# Environment variables
```

```
python-dotenv==1.0.0

# Optional (comentadas)
# Flask-Migrate==4.0.5      # Migraciones de BD
# Flask-Limiter==3.5.0      # Rate limiting
# gunicorn==21.2.0          # Servidor producción
```

Frontend (package.json)

```
{
  "dependencies": {
    "@dicebear/collection": "^9.2.4",
    "@dicebear/core": "^9.2.4",
    "axios": "^1.13.1",
    "react": "^19.1.1",
    "react-dom": "^19.1.1",
    "react-icons": "^5.5.0",
    "react-router-dom": "^7.9.5"
  },
  "devDependencies": {
    "@eslint/js": "^9.36.0",
    "@types/react": "^19.1.16",
    "@types/react-dom": "^19.1.9",
    "@vitejs/plugin-react": "5.0.4",
    "eslint": "^9.36.0",
    "eslint-plugin-react-hooks": "^5.2.0",
    "eslint-plugin-react-refresh": "^0.4.22",
    "globals": "^16.4.0",
    "vite": "^7.1.7"
  }
}
```

11.2 Variables de Entorno (.env)

```
# Base de Datos MySQL
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=tu_password
DB_NAME=freebridge
DB_PORT=3306

# Flask
FLASK_ENV=development
DEBUG=True

# JWT
SECRET_KEY=genera_una_clave_secreta_fuerte_aqui_cambiar_en_produccion
JWT_EXPIRATION_HOURS=24
```

```
# Email (Mailtrap)
MAIL_SERVER=sandbox.smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=tu_username_mailtrap
MAIL_PASSWORD=tu_password_mailtrap
MAIL_USE_TLS=True
MAIL_USE_SSL=False
MAIL_DEFAULT_SENDER=noreply@freebridge.com

# CORS (opcional)
CORS_ORIGINS=http://localhost:5173,http://localhost:5200
```

11.3 Scripts de Utilidad

Backend - Crear Usuario de Prueba

```
# scripts/create_test_user.py
from app import app, db
from models.modelo_usuarios import Usuario
from werkzeug.security import generate_password_hash
import uuid

with app.app_context():
    # Freelancer
    freelancer = Usuario(
        id_usu=str(uuid.uuid4())[:10],
        nombre='Juan Test',
        correo='freelancer@test.com',
        contraseña=generate_password_hash('123456'),
        rol='FreeLancer',
        estado='Activo'
    )

    # Empresa
    empresa = Usuario(
        id_usu=str(uuid.uuid4())[:10],
        nombre='TechCorp',
        correo='empresa@test.com',
        contraseña=generate_password_hash('123456'),
        rol='Empresa',
        estado='Activo'
    )

    db.session.add(freelancer)
    db.session.add(empresa)
    db.session.commit()

    print('✅ Usuarios de prueba creados')
```

Frontend - Hook de Detección de Online/Offline

```
// hooks/useOnlineStatus.js
import { useState, useEffect } from "react";

export const useOnlineStatus = () => {
  const [isOnline, setIsOnline] = useState(navigator.onLine);

  useEffect(() => {
    const handleOnline = () => setIsOnline(true);
    const handleOffline = () => setIsOnline(false);

    window.addEventListener("online", handleOnline);
    window.addEventListener("offline", handleOffline);

    return () => {
      window.removeEventListener("online", handleOnline);
      window.removeEventListener("offline", handleOffline);
    };
  }, []);

  return isOnline;
};


```

11.4 Paleta de Colores

```
:root {
  /* Colores Primarios */
  --primary: #16a085; /* Cyan principal */
  --primary-dark: #16685a; /* Cyan oscuro */
  --primary-light: #b8f2e6; /* Cyan claro */

  /* Colores Secundarios */
  --secondary: #f39c12; /* Naranja */
  --success: #27ae60; /* Verde */
  --danger: #e74c3c; /* Rojo */
  --warning: #f39c12; /* Amarillo */
  --info: #3498db; /* Azul */

  /* Neutros */
  --white: #ffffff;
  --gray-100: #f8f9fa;
  --gray-200: #e9ecef;
  --gray-300: #dee2e6;
  --gray-400: #ced4da;
  --gray-500: #adb5bd;
  --gray-600: #6c757d;
  --gray-700: #495057;
  --gray-800: #343a40;
  --gray-900: #212529;
  --black: #000000;
```

```
/* Sombras */
--shadow-sm: 0 1px 2px 0 rgba(0, 0, 0, 0.05);
--shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.1);
--shadow-md: 0 4px 6px -1px rgba(0, 0, 0, 0.1);
--shadow-lg: 0 10px 15px -3px rgba(0, 0, 0, 0.1);

/* Border Radius */
--radius-sm: 4px;
--radius: 8px;
--radius-lg: 12px;
--radius-full: 9999px;
}
```

11.5 Comandos Útiles

Backend

```
# Activar entorno virtual
.\venv\Scripts\Activate

# Instalar/actualizar dependencias
pip install -r requirements.txt

# Ejecutar servidor
python index.py

# Resetear base de datos
$env:RESET_DB="1"; python index.py

# Ver dependencias instaladas
pip list

# Generar requirements actualizado
pip freeze > requirements.txt
```

Frontend

```
# Instalar dependencias
npm install

# Desarrollo
npm run dev

# Build producción
npm run build

# Preview build
npm run preview
```

```
# Linting
npm run lint

# Limpiar cache
Remove-Item -Recurse -Force node_modules
Remove-Item package-lock.json
npm install
```

11.6 Recursos Adicionales

Documentación Oficial:

- [React 19 Docs](#)
- [React Router v7](#)
- [Axios](#)
- [Vite](#)
- [Flask 3](#)
- [Flask-SQLAlchemy](#)
- [Flask-Mail](#)
- [PyJWT](#)
- [DiceBear](#)

Archivos de Referencia Interna:

- [ARQUITECTURA_AUTH.md](#) - Detalles de autenticación
- [GUIA_ESTILOS.md](#) - Convenciones de CSS
- [SETUP.md](#) - Guía de instalación completa
- [DEPENDENCIAS_UPDATE.md](#) - Resumen de dependencias

11.7 Estructura de Base de Datos Actualizada

```
-- Tabla USUARIO
CREATE TABLE USUARIO (
    id_usu VARCHAR(36) PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    contraseña VARCHAR(255) NOT NULL,
    rol ENUM('Empresa', 'FreeLancer') NOT NULL,
    estado ENUM('Activo', 'Inactivo', 'Bloqueado', 'Eliminado') DEFAULT 'Activo'
);

-- Tabla CIUDAD
CREATE TABLE CIUDAD (
    id_ciud VARCHAR(36) PRIMARY KEY,
    nombr_ciud VARCHAR(30) NOT NULL
);

-- Tabla FREELANCER
CREATE TABLE FREELANCER (
```

```
id_free VARCHAR(36) PRIMARY KEY,  
id_usu VARCHAR(36) NOT NULL,  
id_ciud VARCHAR(36) NOT NULL,  
profesion VARCHAR(50) NOT NULL,  
experiencia TEXT,  
hoja_vida VARCHAR(255),  
avatar VARCHAR(255),  
FOREIGN KEY (id_usu) REFERENCES USUARIO(id_usu),  
FOREIGN KEY (id_ciud) REFERENCES CIUDAD(id_ciud)  
);  
  
-- Tabla EMPRESA  
CREATE TABLE EMPRESA (  
    id_emp VARCHAR(36) PRIMARY KEY,  
    id_usu VARCHAR(36) NOT NULL,  
    id_ciud VARCHAR(36) NOT NULL,  
    NIT VARCHAR(20) UNIQUE NOT NULL,  
    tamano ENUM('Pequeña', 'Mediana', 'Grande') NOT NULL,  
    desc_emp VARCHAR(250) NOT NULL,  
    logo VARCHAR(255),  
    FOREIGN KEY (id_usu) REFERENCES USUARIO(id_usu),  
    FOREIGN KEY (id_ciud) REFERENCES CIUDAD(id_ciud)  
);  
  
-- Tabla VACANTE  
CREATE TABLE VACANTE (  
    id_vac VARCHAR(36) PRIMARY KEY,  
    id_emp VARCHAR(36) NOT NULL,  
    nombr_vacante VARCHAR(50) NOT NULL,  
    descripcion TEXT NOT NULL,  
    requisitos TEXT NOT NULL,  
    salario DECIMAL(10,2),  
    fecha_publicacion DATETIME DEFAULT CURRENT_TIMESTAMP,  
    estado_vac VARCHAR(20) DEFAULT 'abierta',  
    FOREIGN KEY (id_emp) REFERENCES EMPRESA(id_emp)  
);  
  
-- Tabla POSTULACION  
CREATE TABLE POSTULACION (  
    id_post VARCHAR(36) PRIMARY KEY,  
    id_free VARCHAR(36) NOT NULL,  
    id_vac VARCHAR(36) NOT NULL,  
    fecha_post DATETIME DEFAULT CURRENT_TIMESTAMP,  
    estado_post VARCHAR(20) DEFAULT 'pendiente',  
    FOREIGN KEY (id_free) REFERENCES FREELANCER(id_free),  
    FOREIGN KEY (id_vac) REFERENCES VACANTE(id_vac)  
);  
  
-- Tabla password_reset_tokens  
CREATE TABLE password_reset_tokens (  
    id_reset VARCHAR(36) PRIMARY KEY,  
    email VARCHAR(100) NOT NULL,  
    token VARCHAR(255) UNIQUE NOT NULL,  
    expires_at DATETIME NOT NULL,
```

```
    used BOOLEAN DEFAULT FALSE,  
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

🎓 Conclusión

Esta documentación completa cubre la arquitectura, implementación y funcionalidades de **FreeBridge v2.0**, incluyendo:

- Sistema de autenticación con JWT
- Gestión de archivos (CVs, avatares, logos)
- Sistema de notificaciones por email (Mailtrap)
- Recuperación de contraseña
- Generación de avatares con DiceBear
- CRUD completo de vacantes y postulaciones
- Dashboards para empresas y freelancers
- Timeout de sesión por inactividad

Notas importantes:

- Para producción, cambiar **SECRET_KEY** y credenciales
- Implementar HTTPS
- Considerar cookies HttpOnly en vez de localStorage
- Agregar rate limiting (Flask-Limiter)
- Implementar refresh tokens para JWT
- Configurar servicio de email real (SendGrid, AWS SES)
- Usar almacenamiento en la nube para archivos (S3, Cloudinary)

Equipo FreeBridge

Última actualización: Noviembre 2025

Versión: 2.0

Este documento debe actualizarse cada vez que se agreguen nuevas features o se modifique la arquitectura.