Documentación Completa del Proyecto

Sistema de Hoja de Vida en Formato Único

Versión: 1.0.0
Fecha: 2024
Autor: Sistema de Hojas de Vida

Tabla de Contenidos

---

1. Descripción General

1.1 Propósito

El Sistema de Hoja de Vida en Formato Único es una aplicación web full-stack diseñada para permitir a los usuarios crear, gestionar y descargar sus hojas de vida en un formato estandarizado. La aplicación facilita el proceso de registro de información personal, formación académica, experiencia laboral e idiomas, permitiendo generar un documento PDF profesional.

1.2 Características Principales

- ☑ **Autenticación de usuarios**: Sistema de registro e inicio de sesión con JWT
- ☑ **Gestión de datos personales**: Formulario completo para información personal
- ☑ **Formación académica**: Registro de estudios básicos y superiores
- ☑ **Experiencia laboral**: Gestión de múltiples experiencias laborales
- ☑ **Idiomas**: Registro de competencias lingüísticas
- ☑ **Generación de PDF**: Exportación de la hoja de vida completa en formato PDF
- ☑ **Interfaz responsive**: Diseño adaptable para dispositivos móviles y desktop
- ☑ **Recuperación de contraseña**: Sistema de recuperación mediante email

1.3 Tecnologías Utilizadas

Frontend

- **Vue.js 3.3.4**: Framework JavaScript progresivo
- **Vue Router 4.5.1**: Enrutamiento del lado del cliente
- **Pinia 2.3.1**: Gestión de estado
- **Axios 1.11.0**: Cliente HTTP para peticiones API
- **Vite 5.4.19**: Herramienta de construcción
- **html2pdf.js 0.12.0**: Generación de PDFs desde HTML
- **SweetAlert2 11.22.5**: Alertas y notificaciones

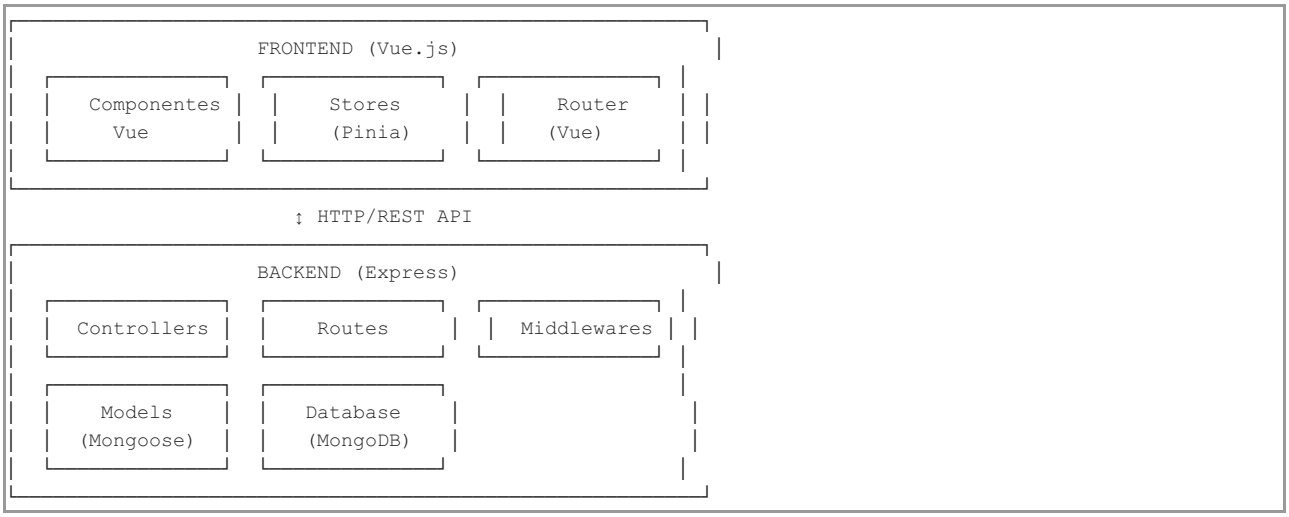
Backend

- **Node.js**: Entorno de ejecución JavaScript
- **Express 4.19.2**: Framework web
- **MongoDB**: Base de datos NoSQL
- **Mongoose 8.4.1**: ODM para MongoDB
- **JWT (jsonwebtoken 9.0.2)**: Autenticación basada en tokens
- **bcryptjs 2.4.3**: Encriptación de contraseñas
- **Nodemailer 7.0.10**: Envío de emails
- **PDFKit 0.15.0**: Generación de PDFs en el servidor

2. Arquitectura del Sistema

2.1 Arquitectura General

El sistema sigue una arquitectura **MVC (Modelo-Vista-Controlador)** con separación clara entre frontend y backend:



2.2 Flujo de Datos

1. **Usuario interactúa** con la interfaz Vue.js
2. **Componente Vue** realiza petición HTTP mediante Axios
3. **Middleware de autenticación** valida el token JWT
4. **Controller** procesa la lógica de negocio
5. **Model** interactúa con MongoDB
6. **Respuesta** se envía de vuelta al frontend
7. **Store (Pinia)** actualiza el estado global
8. **Componente** se re-renderiza con los nuevos datos

2.3 Patrón de Datos Embebido

El sistema utiliza un **modelo embebido** donde toda la información del usuario (datos personales, formación, experiencias, idiomas) se almacena en un único documento `UsuarioEmbebido` en MongoDB. Esto optimiza las consultas y reduce la complejidad de las relaciones.

---

3. Instalación y Configuración

3.1 Requisitos Previos

- **Node.js** >= 18.0.0
- **npm** o **yarn**
- **MongoDB** (local o MongoDB Atlas)
- **Git**

3.2 Instalación del Proyecto

Paso 1: Clonar el repositorio

```
git clone <url-del-repositorio>
cd formatounicoenlinea
```

Paso 2: Instalar dependencias del backend

```
npm install
```

Paso 3: Instalar dependencias del frontend

```
cd frontend
npm install
cd ..
```

3.3 Configuración de Variables de Entorno

Backend (.env)

Crear archivo `.env` en la raíz del proyecto:

```
# MongoDB
MONGO_URI=mongodb://localhost:27017/baseDeDatosHV
# O para MongoDB Atlas:
# MONGO_URI=mongodb+srv://usuario:password@cluster.mongodb.net/baseDeDatosHV

# JWT
JWT_SECRET=tu_secreto_jwt_muy_seguro_aqui_minimo_32_caracteres

# Servidor
PORT=4000
NODE_ENV=development

# Email (para recuperación de contraseña)
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=tu_email@gmail.com
EMAIL_PASS=tu_contraseña_de_aplicacion
```

Frontend

El frontend se conecta automáticamente al backend. Si el backend está en un puerto diferente, editar `frontend/src/api/axios.js`:

```
const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL || 'http://localhost:4000/api',
  // ...
});
```

3.4 Ejecución en Desarrollo

Backend

```
npm run dev
# O
nodemon backend/app.js
```

Frontend

```
cd frontend
npm run dev
```

El frontend estará disponible en `http://localhost:5173` (Vite)
El backend estará disponible en `http://localhost:4000`

3.5 Construcción para Producción

Frontend

```
cd frontend
npm run build
```

Esto generará los archivos estáticos en `frontend/dist/`

Backend

El backend ya está listo para producción. Solo asegúrate de tener las variables de entorno configuradas.

---

4. Estructura del Proyecto

4.1 Estructura de Directorios

```
formatounicoenlinea/
├── backend/
│   ├── app.js                # Configuración principal de Express
│   ├── index.js              # Punto de entrada del servidor
│   ├── config/
│   │   └── db.js             # Configuración de MongoDB
│   ├── controllers/          # Lógica de negocio
│   │   ├── authController.js
│   │   ├── datosPersonalesControllers.js
│   │   ├── experienciaControllers.js
│   │   ├── experienciaTotControllers.js
│   │   ├── formacionAcademicaControllers.js
│   │   ├── hojaVidaController.js
│   │   ├── idiomaController.js
│   │   ├── firmaServidorControllers.js
│   │   ├── pdfControllers.js
│   │   └── recoveryController.js
│   ├── middlewares/          # Middlewares de Express
│   │   ├── auth.js
│   │   ├── verificarJWT.js
│   │   ├── requerirAdmin.js
│   │   ├── validateFormacion.js
│   │   └── idiomasValidations.js
│   ├── models/               # Modelos de Mongoose
│   │   ├── UsuarioEmbebido.js
│   │   ├── Usuario.js
│   │   ├── DatosPersonales.js
│   │   ├── Experiencia.js
│   │   ├── ExperienciaTot.js
│   │   ├── FormacionAcademica.js
│   │   ├── Idioma.js
│   │   └── FirmaServidor.js
│   ├── routes/                # Rutas de la API
│   │   ├── login.js
│   │   ├── usuarios.js
│   │   ├── datosPersonales.js
│   │   ├── formacionAcademica.js
│   │   ├── experiencia.js
│   │   ├── experienciaTot.js
│   │   ├── idiomas.js
│   │   ├── firmaServidor.js
│   │   ├── hojaVidaRoutes.js
│   │   ├── pdf.js
│   │   └── recovery.js
│   └── scripts/              # Scripts de utilidad
│       ├── migrarAEmbebido.js
│       └── limpiarColeccionesAntiguas.js
│
├── frontend/
│   ├── index.html
│   ├── vite.config.js
│   ├── package.json
│   ├── public/
│   │   └── assets/           # Recursos estáticos
│   └── src/
│       ├── main.js           # Punto de entrada
│       ├── App.vue           # Componente raíz
│       ├── style.css         # Estilos globales
│       ├── api/              # Configuración de API
│       │   ├── axios.js
│       │   ├── datosAPI.js
│       │   └── pdfAPI.js
│       ├── components/       # Componentes Vue
│       │   ├── HeaderComponent.vue
│       │   ├── Header2Component.vue
│       │   ├── FooterComponent.vue
│       │   ├── DatosPerComponent.vue
```

```
│       │   ├── FormacionAcadComponent.vue
│       │   ├── ExperienciaComponent.vue
│       │   ├── Experiencia2Component.vue
│       │   ├── ExperienciaTotComponent.vue
│       │   ├── IdiomasComponent.vue
│       │   ├── FirmaServidorComponent.vue
│       │   ├── RecursosHumComponent.vue
│       │   └── MenuComponents.vue
│       ├── views/             # Vistas/páginas
│       │   ├── Login.vue
│       │   ├── Home.vue
│       │   ├── Hoja1.vue
│       │   ├── Hoja2.vue
│       │   ├── Hoja2Extra.vue
│       │   ├── Hoja3.vue
│       │   ├── VistaCompleta.vue
│       │   └── RecuperarPassword.vue
│       ├── Layouts/           # Layouts de página
│       │   ├── LayoutPublico.vue
│       │   └── LayoutPrivado.vue
│       ├── router/            # Configuración de rutas
│       │   └── index.js
│       ├── stores/             # Stores de Pinia
│       │   ├── hojaVida.js
│       │   ├── datos.js
│       │   ├── experienciaStore.js
│       │   └── usuarios.js
│       ├── composables/        # Composables de Vue
│       │   └── useFormatoOficialHV.js
│       ├── utils/             # Utilidades
│       │   ├── showMessage.js
│       │   └── experienciaUtils.js
│       └── helpers/           # Helpers
│           └── axiosInstance.js
│
├── package.json               # Dependencias raíz
├── .env.example              # Ejemplo de variables de entorno
├── Procfile                  # Configuración para Heroku
└── README.md                 # Documentación básica
```

4.2 Descripción de Carpetas Principales

Backend

- **controllers/**: Contiene la lógica de negocio para cada entidad
- **models/**: Define los esquemas de Mongoose para MongoDB
- **routes/**: Define las rutas HTTP y conecta con los controllers
- **middlewares/**: Funciones intermedias (autenticación, validación)
- **config/**: Configuraciones (base de datos, etc.)

Frontend

- **components/**: Componentes Vue reutilizables
- **views/**: Páginas completas de la aplicación
- **stores/**: Estado global gestionado con Pinia
- **router/**: Configuración de rutas del frontend
- **api/**: Cliente HTTP y funciones de API
- **composables/**: Funciones composables de Vue 3
- **utils/**: Funciones de utilidad

---

5. Documentación de API

5.1 Autenticación

POST /api/login

Inicia sesión de un usuario.

Request Body:

```
{
  "email": "usuario@example.com",
  "password": "contraseña123"
}
```

Response (200 OK):

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "usuario": {
    "uid": "507f1f77bcf86cd799439011",
    "nombre": "Juan Pérez",
    "email": "usuario@example.com",
    "roles": ["usuario"]
  }
}
```

Errores:

- `400`: Campos faltantes
- `404`: Usuario no encontrado
- `401`: Contraseña incorrecta

POST /api/usuarios

Registra un nuevo usuario.

Request Body:

```
{
  "nombre": "Juan Pérez",
  "email": "usuario@example.com",
  "password": "contraseña123",
  "roles": ["usuario"]
}
```

Response (201 Created):

```
{
  "mensaje": "Usuario registrado exitosamente."
}
```

Errores:

- `400`: Campos faltantes
- `409`: Email ya registrado

5.2 Datos Personales

GET /api/datos-personales

Obtiene los datos personales del usuario autenticado.

Headers:

```
Authorization: Bearer <token>
```

Response (200 OK):

```
{
  "apellido1": "Pérez",
  "apellido2": "García",
  "nombres": "Juan",
  "tipoDocumento": "C.C",
  "numDocumento": "1234567890",
  "sexo": "M",
  "nacionalidad": "Colombiana",
  "pais": "Colombia",
  "fechaNacimiento": {
    "dia": "15",
    "mes": "03",
    "anio": "1990",
    "pais": "Colombia",
    "depto": "Cundinamarca",
    "municipio": "Bogotá"
  },
  "direccionCorrespondencia": {
    "pais": "Colombia",
    "depto": "Cundinamarca",
    "municipio": "Bogotá",
    "direccion": "Calle 123 #45-67",
    "telefono": "3001234567",
    "email": "usuario@example.com"
  }
}
```


POST /api/datos-personales

Crea los datos personales del usuario.

Request Body:


```
{
  "apellido1": "Pérez",
  "apellido2": "García",
  "nombres": "Juan",
  "tipoDocumento": "C.C",
  "numDocumento": "1234567890",
  "sexo": "M",
  "nacionalidad": "Colombiana",
  "pais": "Colombia",
  "fechaNacimiento": {
    "dia": "15",
    "mes": "03",
    "anio": "1990"
  },
  "direccionCorrespondencia": {
    "pais": "Colombia",
    "depto": "Cundinamarca",
    "municipio": "Bogotá",
    "direccion": "Calle 123 #45-67",
    "telefono": "3001234567",
    "email": "usuario@example.com"
  }
}
```


PUT /api/datos-personales

Actualiza los datos personales del usuario.

Request Body: (Igual que POST)

5.3 Formación Académica

GET /api/formacion-academica

Obtiene la formación académica del usuario.

Response:

```
{
  "gradoBasica": 11,
  "tituloBachiller": "Bachiller Académico",
  "mesGrado": "11",
  "anioGrado": "2008",
  "formacionSuperior": [
    {
      "modalidad": "Técnico",
      "semestres": "6",
      "graduado": "SI",
      "titulo": "Técnico en Sistemas",
      "mesTermino": "06",
      "anioTermino": "2010",
      "tarjeta": "12345"
    }
  ]
}
```


POST /api/formacion-academica

Crea o actualiza la formación académica.

Request Body:


```
{
  "gradoBasica": 11,
  "tituloBachiller": "Bachiller Académico",
  "mesGrado": "11",
  "anioGrado": "2008",
  "formacionSuperior": [
    {
      "modalidad": "Técnico",
      "semestres": "6",
      "graduado": "SI",
      "titulo": "Técnico en Sistemas",
      "mesTermino": "06",
      "anioTermino": "2010",
      "tarjeta": "12345"
    }
  ]
}
```


5.4 Experiencia Laboral

GET /api/experiencia

Obtiene todas las experiencias laborales del usuario.

Response:


```
[
  {
    "_id": "507f1f77bcf86cd799439011",
    "empresa": "Empresa XYZ",
    "tipoEntidad": "Privada",
    "pais": "Colombia",
    "departamento": "Cundinamarca",
    "municipio": "Bogotá",
    "correoEntidad": "contacto@empresa.com",
    "telefonos": "6012345678",
    "fechaIngreso": "2020-01-15T00:00:00.000Z",
    "fechaRetiro": "2022-12-31T00:00:00.000Z",
    "cargo": "Desarrollador",
    "dependencia": "TI",
    "direccion": "Calle 100 #50-30"
  }
]
```


POST /api/experiencia

Crea una nueva experiencia laboral.

Request Body:

```
{
  "empresa": "Empresa XYZ",
  "tipoEntidad": "Privada",
  "pais": "Colombia",
  "departamento": "Cundinamarca",
  "municipio": "Bogotá",
  "correoEntidad": "contacto@empresa.com",
  "telefonos": "6012345678",
  "fechaIngreso": "2020-01-15",
  "fechaRetiro": "2022-12-31",
  "cargo": "Desarrollador",
  "dependencia": "TI",
  "direccion": "Calle 100 #50-30"
}
```

Validaciones:

- `fechaIngreso` y `fechaRetiro` deben ser fechas válidas
- `fechaIngreso` no puede ser mayor que `fechaRetiro`
- `tipoEntidad` debe ser "Publica" o "Privada"

DELETE /api/experiencia/:id

Elimina una experiencia laboral.

5.5 Idiomas

GET /api/idiomas

Obtiene los idiomas del usuario.

Response:

```
{
  "idiomas": [
    {
      "nombre": "Inglés",
      "habla": "B",
      "lee": "B",
      "escribe": "R"
    },
    {
      "nombre": "Francés",
      "habla": "R",
      "lee": "R",
      "escribe": ""
    }
  ]
}
```

Niveles: R (Regular), B (Bueno), MB (Muy Bueno)

POST /api/idiomas

Crea o actualiza los idiomas del usuario.

Request Body:

```
{
  "idiomas": [
    {
      "nombre": "Inglés",
      "habla": "B",
      "lee": "B",
      "escribe": "R"
    }
  ]
}
```

Validaciones:

- Máximo 10 idiomas por usuario
- `habla`, `lee`, `escribe` deben ser "R", "B", "MB" o ""

5.6 Hoja de Vida Completa

GET /api/hoja-vida

Obtiene toda la información de la hoja de vida del usuario.

Response:

```
{
  "datosPersonales": { /* ... */ },
  "formacionAcademica": { /* ... */ },
  "experiencias": [ /* ... */ ],
  "idiomas": [ /* ... */ ],
  "firmaServidor": { /* ... */ }
}
```

5.7 Generación de PDF

GET /api/pdf/generar

Genera un PDF de la hoja de vida completa.

Response: Archivo PDF binario

Headers de respuesta:

```
Content-Type: application/pdf
Content-Disposition: attachment; filename="hoja-de-vida.pdf"
```

5.8 Recuperación de Contraseña

POST /api/recovery/solicitar

Solicita un código de recuperación de contraseña.

Request Body:

```
{
  "email": "usuario@example.com"
}
```

Response:

```
{
  "mensaje": "Código de recuperación enviado al correo electrónico"
}
```

POST /api/recovery/verificar

Verifica el código de recuperación.

Request Body:

```
{
  "email": "usuario@example.com",
  "codigo": "123456"
}
```

Response:

```
{
  "token": "token_temporal_para_cambiar_contraseña"
}
```

POST /api/recovery/cambiar

Cambia la contraseña con el token temporal.

Request Body:

```
{
  "token": "token_temporal",
  "nuevaPassword": "nueva_contraseña123"
}
```

5.9 Autenticación

Todas las rutas (excepto `/api/login`, `/api/usuarios` y `/api/recovery/*`) requieren autenticación mediante JWT.

Header requerido:

```
Authorization: Bearer <token>
```

El token expira en **2 horas**.

---

6. Componentes Vue

6.1 Componentes de Layout

LayoutPrivado.vue

Layout principal para páginas autenticadas. Incluye menú lateral responsive.

Props: Ninguna

Slots:

- Default: Contenido de la página

Características:

- Menú lateral colapsable en móviles
- Header con botón de menú hamburguesa
- Backdrop para móviles

LayoutPublico.vue

Layout para páginas públicas (login, recuperación).

Props: Ninguna

Slots:

- Default: Contenido de la página

6.2 Componentes de Formulario

DatosPerComponent.vue

Formulario completo de datos personales.

Props: Ninguna

Emits:

- `guardado`: Cuando se guardan los datos exitosamente

Métodos:

- `enviarFormulario()`: Envía los datos al backend
- `actualizarDatos()`: Actualiza datos existentes

Campos:

- Apellidos, nombres
- Tipo y número de documento
- Sexo, nacionalidad
- Fecha de nacimiento
- Dirección de correspondencia

FormacionAcadComponent.vue

Formulario de formación académica.

Props: Ninguna

Métodos:

- `guardarFormacion()`: Guarda la formación académica

Campos:

- Grado de básica (1-11)
- Título de bachiller
- Hasta 3 formaciones superiores

ExperienciaComponent.vue

Lista y edición de experiencias laborales.

Props:

- `experiencias`: Array de experiencias

Métodos:

- `eliminarExperiencia(id)`: Elimina una experiencia
- `editarExperiencia(experiencia)`: Edita una experiencia

Experiencia2Component.vue

Formulario para crear nueva experiencia.

Props:

- `experiencia`: Objeto experiencia (opcional, para edición)

Emits:

- `saved`: Cuando se guarda exitosamente

Métodos:

- `guardarExperiencia()`: Guarda la experiencia
- `resetFormulario()`: Limpia el formulario

Validaciones:

- Fechas válidas
- Fecha de ingreso <= fecha de retiro

IdiomasComponent.vue

Gestión de idiomas.

Props: Ninguna

Métodos:

- `agregarIdioma()`: Agrega un nuevo idioma

- `eliminarIdioma(index)`: Elimina un idioma
- `guardarIdiomas()`: Guarda todos los idiomas

Límites:

- Máximo 10 idiomas

ExperienciaTotComponent.vue

Resumen de tiempo total de experiencia.

Props:

- `experienciaPublica`: String con tiempo público
- `experienciaPrivada`: String con tiempo privado

Características:

- Muestra cálculo automático de tiempos

FirmaServidorComponent.vue

Gestión de firma del servidor y declaraciones.

Props: Ninguna

Métodos:

- `capturarFirma()`: Captura la firma desde canvas
- `guardarFirma()`: Guarda la firma y declaraciones

Campos:

- Canvas para firma
- Declaración de inhabilidad
- Ciudad y fecha de diligenciamiento

6.3 Componentes de Presentación

HeaderComponent.vue

Encabezado de la hoja de vida (Hoja 1).

Props: Ninguna

Header2Component.vue

Encabezado para Hoja 2 y Hoja 3.

Props: Ninguna

FooterComponent.vue

Pie de página de la hoja de vida.

Props: Ninguna

MenuComponents.vue

Menú lateral de navegación.

Props: Ninguna

Emits:

- `close`: Cuando se cierra el menú (móviles)

Métodos:

- `cerrarSesion()`: Cierra la sesión del usuario
- `handleNavigate()`: Cierra el menú al navegar

RecursosHumComponent.vue

Componente para descarga de PDF.

Props: Ninguna

Métodos:

- `descargarPDF()`: Descarga el PDF de la hoja de vida

6.4 Vistas

Login.vue

Página de inicio de sesión y registro.

Funcionalidades:

- Login de usuarios existentes
- Registro de nuevos usuarios
- Toggle entre modos
- Contador de usuarios registrados
- Panel de información de contacto

Hoja1.vue

Vista principal de datos personales y formación.

Componentes utilizados:

- HeaderComponent
- DatosPerComponent
- FormacionAcadComponent
- IdiomasComponent

Hoja2.vue

Vista de experiencias laborales.

Componentes utilizados:

- Header2Component
- ExperienciaComponent

Hoja2Extra.vue

Vista para registrar nueva experiencia.

Componentes utilizados:

- Header2Component
- Experiencia2Component
- FooterComponent

Hoja3.vue

Vista de resumen de experiencia y firma.

Componentes utilizados:

- Header2Component
- ExperienciaTotComponent
- FirmaServidorComponent
- RecursosHumComponent
- FooterComponent

VistaCompleta.vue

Vista previa completa de la hoja de vida con opción de generar PDF.

Funcionalidades:

- Vista previa de todas las hojas
- Generación de PDF
- Exportación con html2pdf.js

RecuperarPassword.vue

Página de recuperación de contraseña.

Funcionalidades:

- Solicitud de código
- Verificación de código
- Cambio de contraseña

---

7. Base de Datos

7.1 Modelo de Datos Embebido

El sistema utiliza un modelo **embebido** donde toda la información del usuario se almacena en un único documento `UsuarioEmbebido`.

7.2 Esquema UsuarioEmbebido

```
{
  // Autenticación
  nombre: String (required),
  email: String (required, unique, lowercase),
  password: String (required, hasheado),
  roles: [String] (default: ["usuario"]),

  // Datos personales
  apellido1: String,
  apellido2: String,
  nombres: String,
  tipoDocumento: String,
  numDocumento: String,
  sexo: String,
  nacionalidad: String,
  pais: String,
  libretaMilitar: String,
  numeroLibreta: String,
  dm: String,

  fechaNacimiento: {
    dia: String,
    mes: String,
    anio: String,
    pais: String,
    depto: String,
    municipio: String
  },

  direccionCorrespondencia: {
    pais: String,
    depto: String,
    municipio: String,
    direccion: String,
    telefono: String,
    email: String
  },

  // Formación académica
  gradoBasica: Number (min: 1, max: 11),
  tituloBachiller: String,
  mesGrado: String,
  anioGrado: String,
  formacionSuperior: [{
    modalidad: String,
    semestres: String,
    graduado: String (enum: ["SI", "NO", ""]),
    titulo: String,
    mesTermino: String,
```

```
    anioTermino: String,
    tarjeta: String
  }],

  // Experiencia laboral
  experiencias: [{
    empresa: String,
    tipoEntidad: String (enum: ["Publica", "Privada"]),
    pais: String,
    departamento: String,
    municipio: String,
    correoEntidad: String,
    telefonos: String,
    fechaIngreso: Date,
    fechaRetiro: Date,
    cargo: String,
    dependencia: String,
    direccion: String
  }],

  // Resumen de experiencia
  experienciaPublica: String,
  experienciaPrivada: String,

  // Idiomas
  idiomas: [{
    nombre: String,
    habla: String (enum: ['R', 'B', 'MB', '']),
    lee: String (enum: ['R', 'B', 'MB', '']),
    escribe: String (enum: ['R', 'B', 'MB', ''])
  }],

  // Firma y declaraciones
  firmaServidor: String (base64),
  firmaBase64: String,
  declaracionInhabilidad: String (enum: ['SI', 'NO', '']),
  ciudadDiligenciamiento: String,
  fechaDiligenciamiento: Date,

  // Metadatos
  fechaCreacion: Date (default: Date.now),
  ultimoAcceso: Date,
  createdAt: Date,
  updatedAt: Date
}
```

7.3 Índices

- `email`: Índice único
- `numDocumento`: Índice para búsquedas

7.4 Modelos Legacy (No utilizados actualmente)

Los siguientes modelos existen pero no se utilizan en el sistema actual (se migró a modelo embebido):

- `Usuario.js`
- `DatosPersonales.js`
- `FormacionAcademica.js`
- `Experiencia.js`
- `Idioma.js`
- `FirmaServidor.js`
- `ExperienciaTot.js`

Estos modelos se mantienen por compatibilidad o para futuras migraciones.

7.5 Relaciones

Con el modelo embebido, **no hay relaciones** entre colecciones. Toda la información está en un solo documento, lo que optimiza las consultas.

---

8. Guías de Desarrollo

8.1 Configuración del Entorno

Variables de Entorno Requeridas

Backend:

- `MONGO_URI`: URI de conexión a MongoDB
- `JWT_SECRET`: Secreto para firmar tokens JWT
- `PORT`: Puerto del servidor (opcional, default: 4000)
- `NODE_ENV`: Entorno (development/production)

Frontend:

- `VITE_API_URL`: URL del backend (opcional, default: http://localhost:4000/api)

8.2 Scripts Disponibles

Backend (package.json raíz)

```
{
  "start": "node backend/app.js",
  "dev": "nodemon backend/app.js",
  "heroku-postbuild": "cd frontend && npm install --production=false && npm run build"
}
```

Frontend (frontend/package.json)

```
{
  "dev": "vite",
  "build": "vite build",
  "serve": "vite preview"
}
```

8.3 Convenciones de Código

Nomenclatura

- **Componentes Vue**: PascalCase (ej: `DatosPerComponent.vue`)
- **Archivos JavaScript**: camelCase (ej: `authController.js`)
- **Variables y funciones**: camelCase (ej: `guardarExperiencia`)
- **Constantes**: UPPER_SNAKE_CASE (ej: `JWT_SECRET`)
- **Clases**: PascalCase (ej: `UsuarioEmbebido`)

Estructura de Componentes Vue

```
<template>
  <!-- HTML aquí -->
</template>

<script setup>
// Imports
// Props
// Emits
// Reactive state
// Computed
// Methods
// Lifecycle hooks
</script>

<style scoped>
/* Estilos aquí */
</style>
```

Estructura de Controllers

```
export const nombreFuncion = async (req, res) => {
  try {
    // Lógica aquí
    res.status(200).json({ mensaje: "Éxito", data: resultado });
  } catch (error) {
    console.error("Error:", error);
    res.status(500).json({ mensaje: "Error", error: error.message });
  }
};
```

8.4 Manejo de Errores

Backend

- Usar try-catch en todas las funciones async
- Retornar códigos HTTP apropiados
- Incluir mensajes descriptivos
- Loggear errores en consola

Frontend

- Usar `showError()` de `utils/showMessage.js` para errores
- Usar `showSuccess()` para mensajes de éxito
- Manejar errores de red con mensajes amigables

8.5 Testing

Actualmente no hay tests automatizados. Se recomienda:

- **Backend**: Usar Jest o Mocha para tests unitarios
- **Frontend**: Usar Vitest para tests de componentes Vue
- **E2E**: Usar Cypress o Playwright

8.6 Git Workflow

1. Crear branch desde `main`: `git checkout -b feature/nombre-feature`
2. Hacer commits descriptivos
3. Push: `git push origin feature/nombre-feature`
4. Crear Pull Request
5. Revisar y mergear

---

9. Despliegue

9.1 Despliegue en Heroku

Paso 1: Preparar el proyecto

Asegúrate de tener:

- `Procfile` en la raíz
- Variables de entorno configuradas
- Build del frontend generado

Paso 2: Instalar Heroku CLI

```
npm install -g heroku
heroku login
```

Paso 3: Crear aplicación

```
heroku create nombre-aplicacion
```

Paso 4: Configurar variables de entorno

```
heroku config:set MONGO_URI=tu_uri_mongodb
heroku config:set JWT_SECRET=tu_secreto
heroku config:set NODE_ENV=production
```

Paso 5: Desplegar

```
git push heroku main
```

9.2 Despliegue en Railway

Paso 1: Conectar repositorio

1. Ir a railway.app (https://railway.app)
2. Crear nuevo proyecto
3. Conectar repositorio de GitHub

Paso 2: Configurar variables

En la pestaña "Variables":

```
MONGO_URI=mongodb+srv://...
JWT_SECRET=...
NODE_ENV=production
```

Paso 3: Desplegar

Railway detectará automáticamente Node.js y desplegará.

9.3 Despliegue en Vercel/Netlify (Solo Frontend)

Si deseas desplegar el frontend por separado:

Vercel

```
cd frontend
npm run build
vercel deploy
```

Netlify

```
cd frontend
npm run build
netlify deploy --prod --dir=dist
```

Nota: Necesitarás configurar la variable `VITE_API_URL` con la URL de tu backend.

9.4 Despliegue Manual

Backend

```
# En el servidor
git clone <repo>
cd formatounicoenlinea
npm install
# Configurar .env
npm start
```

Frontend

```
cd frontend
npm install
npm run build
# Servir archivos de frontend/dist con nginx o similar
```

9.5 Configuración de Nginx (Opcional)

```
server {
    listen 80;
    server_name tu-dominio.com;

    # Frontend
    location / {
        root /ruta/a/frontend/dist;
        try_files $uri $uri/ /index.html;
    }

    # Backend API
    location /api {
        proxy_pass http://localhost:4000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

---

10. Casos de Uso

10.1 Registro de Nuevo Usuario

Actor: Usuario no registrado

Flujo:

1. Usuario accede a la página de login
2. Hace clic en "Crear cuenta gratis"
3. Completa formulario: nombre, email, contraseña
4. Sistema valida que el email no exista
5. Sistema crea cuenta y redirige a login
6. Usuario inicia sesión

Resultado: Usuario registrado y autenticado

10.2 Inicio de Sesión

Actor: Usuario registrado

Flujo:

1. Usuario ingresa email y contraseña
2. Sistema valida credenciales
3. Sistema genera token JWT
4. Token se almacena en localStorage
5. Usuario es redirigido al panel principal

Resultado: Usuario autenticado con acceso al sistema

10.3 Completar Hoja de Vida

Actor: Usuario autenticado

Flujo:

1. Usuario accede a "Datos Personales"
2. Completa formulario de datos personales
3. Guarda información
4. Accede a "Experiencia Laboral"
5. Agrega experiencias laborales
6. Completa formación académica
7. Agrega idiomas
8. Completa resumen de experiencia
9. Agrega firma y declaraciones

Resultado: Hoja de vida completa y guardada

10.4 Generar PDF

Actor: Usuario autenticado

Flujo:

1. Usuario completa toda su hoja de vida
2. Accede a "Generar PDF"
3. Revisa vista previa
4. Hace clic en "Descargar PDF"
5. Sistema genera PDF
6. PDF se descarga automáticamente

Resultado: PDF de hoja de vida descargado

10.5 Recuperar Contraseña

Actor: Usuario que olvidó su contraseña

Flujo:

1. Usuario hace clic en "¿Olvidaste tu contraseña?"
2. Ingresa su email
3. Sistema envía código de recuperación al email
4. Usuario ingresa código recibido
5. Sistema valida código
6. Usuario ingresa nueva contraseña
7. Sistema actualiza contraseña
8. Usuario puede iniciar sesión

Resultado: Contraseña recuperada y actualizada

10.6 Agregar Nueva Experiencia

Actor: Usuario autenticado

Flujo:

1. Usuario accede a "Registrar Experiencia"
2. Completa formulario de experiencia
3. Sistema valida fechas
4. Usuario guarda experiencia
5. Formulario se limpia automáticamente
6. Usuario puede agregar otra experiencia

Resultado: Nueva experiencia agregada, formulario listo para otra

10.7 Editar Información Personal

Actor: Usuario autenticado

Flujo:

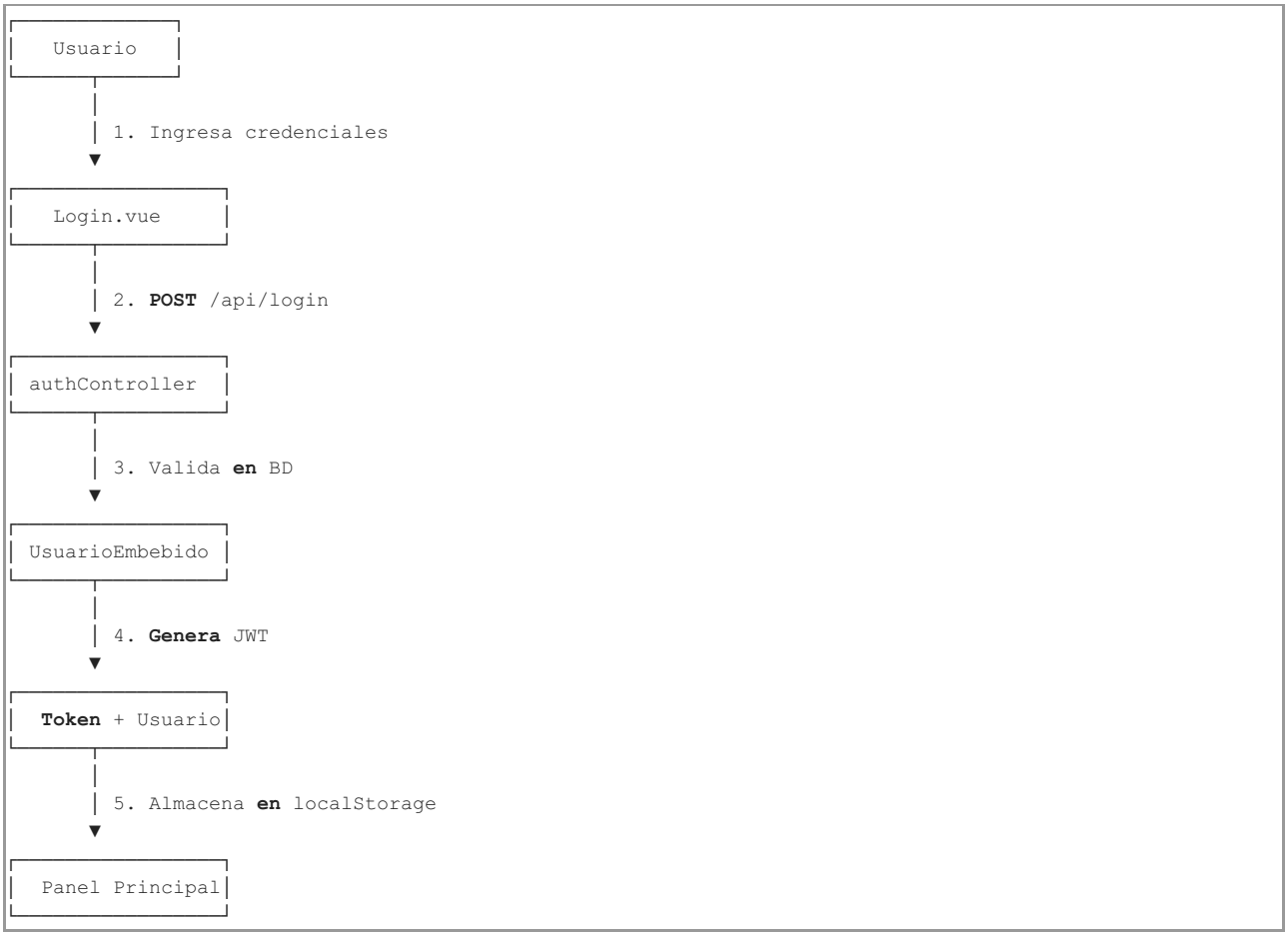
1. Usuario accede a sección con información existente
2. Modifica campos deseados
3. Hace clic en "Actualizar"
4. Sistema valida y guarda cambios
5. Muestra mensaje de éxito

Resultado: Información actualizada

---

11. Diagramas

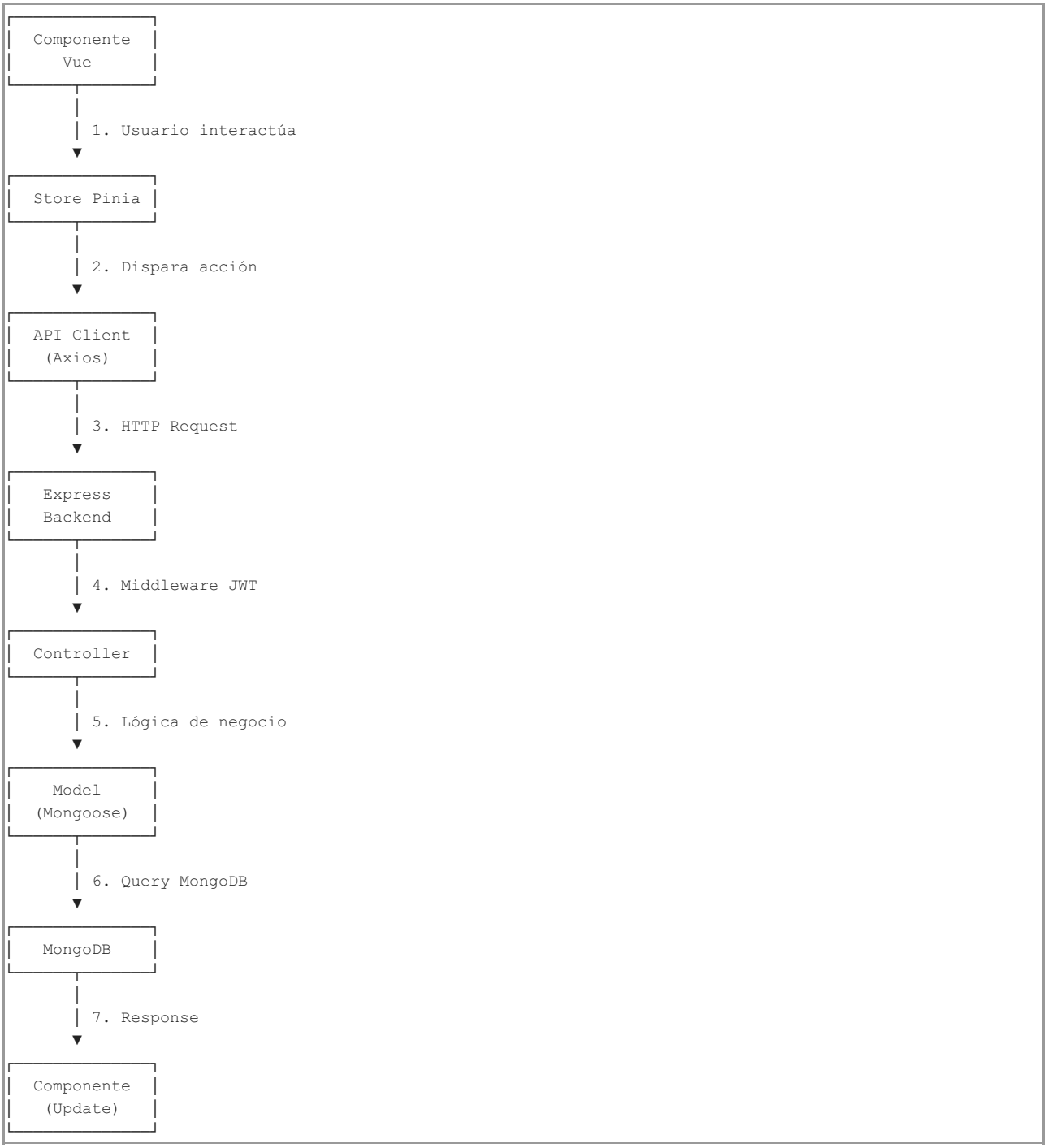
11.1 Diagrama de Flujo de Autenticación



11.2 Diagrama de Arquitectura de Componentes

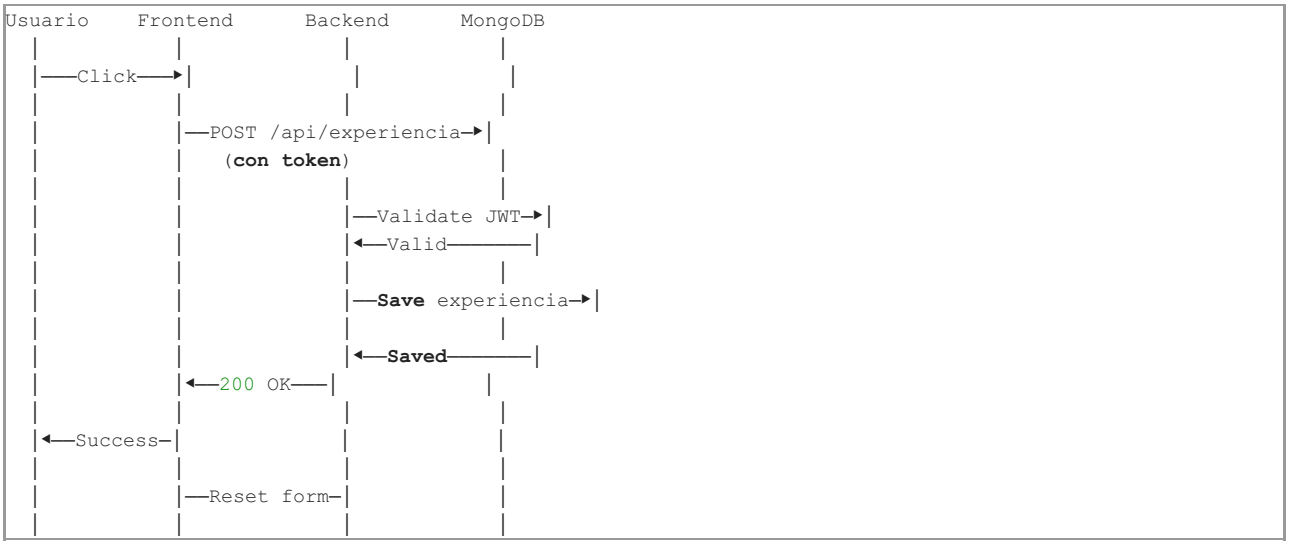
```
App.vue
|
├── RouterView
    |
    ├── LayoutPublico
    |   └── Login.vue
    |   └── RecuperarPassword.vue
    |
    └── LayoutPrivado
        ├── MenuComponents
        └── RouterView
            ├── Hoja1.vue
            |   ├── HeaderComponent
            |   ├── DatosPerComponent
            |   ├── FormacionAcadComponent
            |   └── IdiomasComponent
            |
            ├── Hoja2.vue
            |   ├── Header2Component
            |   └── ExperienciaComponent
            |
            ├── Hoja2Extra.vue
            |   ├── Header2Component
            |   ├── Experiencia2Component
            |   └── FooterComponent
            |
            ├── Hoja3.vue
            |   ├── Header2Component
            |   ├── ExperienciaTotComponent
            |   ├── FirmaServidorComponent
            |   ├── RecursosHumComponent
            |   └── FooterComponent
            |
            └── VistaCompleta.vue
                ├── Hoja1
                ├── Hoja2
                └── Hoja3
```

11.3 Diagrama de Flujo de Datos



11.4 Diagrama de Base de Datos

11.5 Diagrama de Secuencia - Guardar Experiencia



---

12. Consideraciones Adicionales

12.1 Seguridad

- **Contraseñas**: Hasheadas con bcrypt (10 rounds)
- **JWT**: Tokens con expiración de 2 horas
- **CORS**: Configurado para permitir solo orígenes autorizados
- **Validación**: Validación de datos en frontend y backend
- **Sanitización**: Los datos se validan antes de guardar

12.2 Performance

- **Modelo embebido**: Reduce consultas a la base de datos
- **Índices**: Email y numDocumento indexados
- **Lazy loading**: Componentes cargados bajo demanda
- **Code splitting**: Vite divide el código automáticamente

12.3 Escalabilidad

- **MongoDB Atlas**: Soporta escalado horizontal
- **Stateless API**: Fácil de escalar horizontalmente
- **CDN**: Frontend puede servirse desde CDN

12.4 Mantenimiento

- **Logs**: Errores logueados en consola
- **Versionado**: Control de versiones con Git
- **Documentación**: Este documento se actualiza con cambios

---

13. Contacto y Soporte

Email: randysimancamercado@gmail.com
WhatsApp: +57 314 519 3285

---

14. Licencia

Fin de la Documentación