Universidad Mariano Gálvez de Guatemala

Facultad de ingeniería en sistemas

8vo semestre

Sección: “B”

Ingeniero José Miguel Villatoro Hidalgo

MANUAL TECNICO

NOMBRE	CARNET
Francisco Alexander Chic Barrios	9490-22-2513
Herbert Daniel Jocol Morataya	9490-22-423

INDICE

INTRODUCCIÓN

El presente documento describe el Manual Técnico del proyecto "Votos Guate", una plataforma web desarrollada con el objetivo de gestionar procesos de votación en línea de forma segura, moderna y eficiente.

El sistema fue construido utilizando React para el desarrollo del frontend, haciendo uso de SASS/SCSS para la gestión avanzada de estilos y un diseño modular. La aplicación web fue desplegada en la nube a través del servicio Netlify, lo que permite su acceso desde cualquier dispositivo con conexión a internet.

En cuanto al backend, se implementó una API REST desarrollada en Node.js y Express, la cual maneja toda la lógica del servidor, la autenticación mediante JSON Web Tokens (JWT) y la comunicación con la base de datos. Dicha API está alojada en la plataforma Render, garantizando un entorno de ejecución estable y escalable.

La base de datos utilizada es PostgreSQL, también desplegada en Render, donde se almacenan los registros de usuarios, campañas, candidatos y votos emitidos. Esta integración entre frontend, backend y base de datos proporciona una arquitectura completa y funcional, que simula un entorno de desarrollo real utilizado en aplicaciones empresariales modernas.

Este manual tiene como finalidad documentar la estructura técnica del sistema, describiendo sus componentes, tecnologías, diagramas, y configuraciones necesarias para su comprensión, mantenimiento y futuras mejoras.

REQUERIMIENTOS TECNICOS

El proyecto **"Votos Guate"** fue desarrollado como una aplicación web moderna de votación en línea, cumpliendo con los requerimientos definidos por el curso de **Desarrollo Web** de la **Universidad Mariano Gálvez de Guatemala**, sede Naranjo.

A continuación, se detallan los requerimientos técnicos del sistema:

---

1. Requerimientos de Software

Frontend

- **Framework:** React.js
- **Lenguaje:** JavaScript (con compatibilidad TypeScript)
- **Gestor de paquetes:** npm
- **Preprocesador de estilos:** SASS/SCSS
- **Control de rutas:** React Router DOM
- **Diseño responsivo:** HTML5, CSS3, Flexbox y Grid
- **Despliegue:** Netlify
- **Dependencias principales:**
    - react-router-dom
    - axios (para peticiones HTTP al backend)
    - bootstrap o react-bootstrap (para interfaz adaptable)
    - jwt-decode (para manejar el token de sesión en el cliente)

Backend

- **Entorno de ejecución:** Node.js
- **Framework:** Express.js
- **Lenguaje:** JavaScript (ES6)
- **Base de datos:** PostgreSQL
- **ORM / Conexión:** pg (módulo de conexión nativo a PostgreSQL)
- **Autenticación:** JSON Web Token (JWT)
- **Configuración de variables de entorno:** dotenv
- **Despliegue:** Render (https://votosguate-api.onrender.com)

Base de Datos

- **Sistema Gestor:** PostgreSQL
- **Servidor:** Render PostgreSQL Cloud Database
- **Nombre de base de datos:** sistema_votacion_gt
- **Usuario:** sistema_votacion_gt_user
- **Puerto:** 5432

- URL de conexión externa:
- postgresql://sistema_votacion_gt_user:CxjVWPVM2p8jXq0MGs2YFPt7v1G1xURN@dpg-d3u1u8muk2gs73dhi6qg-a.oregon-postgres.render.com/sistema_votacion_gt

---

2. Requerimientos de Infraestructura y Servicios en la Nube

Componente	Servicio utilizado	Descripción
Frontend	Netlify	Aloja la aplicación React y gestiona el despliegue automático desde el repositorio.
Backend (API REST)	Render	Ejecuta el servidor Node.js/Express y gestiona la conexión con la base de datos.
Base de Datos	Render PostgreSQL	Servicio en la nube que almacena los datos de usuarios, campañas, candidatos y votos.
Autenticación	JWT (JSON Web Token)	Sistema de autenticación basado en tokens para validar las sesiones de usuario.

---

3. Requerimientos de Hardware (mínimos para desarrollo)

Elemento	Especificación mínima
Procesador	Intel Core i3 o superior
Memoria RAM	8 GB recomendados
Almacenamiento	500 MB disponibles para entorno de desarrollo
Conectividad	Conexión estable a internet para uso de servicios en la nube

---

4. Requerimientos de Seguridad

- Implementación de JWT para el control de sesiones seguras.
- Validación de tokens en cada petición al backend.
- Cierre automático de sesión al expirar el token o detectarse uno inválido.
- Cifrado de contraseñas antes de almacenarse en la base de datos.
- Validación de formularios en frontend y backend para evitar datos corruptos.

---

5. Enlaces del Proyecto

- **Sitio web funcional:** https://votos-guate.netlify.app
- **API Backend:** https://votosguate-api.onrender.com
- **Base de datos (Render PostgreSQL):** sistema_votacion_gt
- **Repositorio:** https://github.com/AlexanderChic/ProyectoDesarrolloWEB/tree/main

HERRAMIENTAS UTILIZADAS PARA EL DESARROLLO

Herramientas Utilizadas para el Desarrollo

Para la implementación del sistema se emplearon diversas herramientas de software que permitieron el desarrollo eficiente, la integración entre componentes y el despliegue en entornos en la nube. A continuación, se detallan las principales tecnologías utilizadas:

1. Lenguajes y Entornos de Programación

- **JavaScript (ES6+):** Lenguaje principal utilizado para el desarrollo tanto del frontend como del backend del sistema.
- **Node.js:** Entorno de ejecución que permite la ejecución de JavaScript en el servidor, utilizado para la creación de la API REST.
- **React.js:** Biblioteca de JavaScript empleada para el desarrollo de la interfaz de usuario, permitiendo una estructura modular y componentes reutilizables.

2. Frameworks y Librerías

- **Express.js:** Framework de Node.js utilizado para la creación y gestión de rutas, controladores y middleware en la API.
- **Axios:** Librería empleada en el frontend para la comunicación con el servidor mediante peticiones HTTP.
- **SASS/SCSS:** Preprocesador de CSS utilizado para mejorar la organización y mantenimiento de los estilos en la aplicación web.

3. Base de Datos

- **PostgreSQL:** Sistema de gestión de bases de datos relacional (RDBMS) utilizado para el almacenamiento de la información del sistema. Se caracteriza por su robustez, escalabilidad y soporte para consultas complejas.

4. Plataformas de Despliegue

- **Render:** Servicio en la nube utilizado para alojar la API y la base de datos PostgreSQL, permitiendo su acceso remoto y una administración centralizada.
- **Netlify:** Plataforma empleada para el despliegue del frontend desarrollado en React, facilitando la publicación continua y el manejo de dominios personalizados.

5. Entornos de Desarrollo y Herramientas Complementarias

- **Visual Studio Code:** Editor de código utilizado para la programación y depuración del proyecto.

- Postman: Herramienta utilizada para realizar pruebas de los endpoints de la API y verificar la correcta comunicación con el servidor.
- Git y GitHub: Utilizados para el control de versiones y la gestión colaborativa del código fuente.
- PgAdmin / DBeaver: Clientes gráficos utilizados para la administración y visualización de la base de datos PostgreSQL.

Gestión de Versiones y Ramas del Proyecto

Para el control de versiones del proyecto se utilizó Git como sistema de gestión de código fuente, junto con GitHub como plataforma de alojamiento del repositorio.
El proyecto está estructurado en tres ramas principales que permiten mantener un flujo de desarrollo ordenado y controlado:

- main (rama principal): Contiene la versión estable del proyecto lista para despliegue en producción.
- frontendproductivo: Rama destinada al desarrollo y actualización del código del lado del cliente (interfaz desarrollada en React).
- productivo: Rama utilizada para las pruebas finales y sincronización con el entorno desplegado en la nube. API

El flujo de trabajo se basa en la integración continua, donde los cambios realizados en las ramas de desarrollo se revisan y, una vez verificados, se integran en la rama principal para su posterior despliegue.

Esquema Conceptual de Componentes

El sistema se encuentra estructurado bajo una **arquitectura cliente-servidor**, compuesta principalmente por tres niveles: la interfaz de usuario (frontend), la API (backend) y la base de datos. Esta organización permite una separación clara entre las responsabilidades de cada módulo, facilitando el mantenimiento y la escalabilidad del sistema.

1. **Frontend (Cliente Web):**
   Desarrollado con **React.js**, es responsable de la interacción directa con el usuario. Desde esta capa se gestionan las vistas, formularios y la presentación de datos, permitiendo la comunicación con el backend mediante peticiones HTTP.
2. **Backend (API REST):**
   Implementado con **Node.js** y **Express.js**, se encarga de manejar las solicitudes provenientes del frontend, aplicar las reglas de negocio, procesar los datos y comunicarse con la base de datos PostgreSQL. Proporciona endpoints seguros para la gestión de usuarios, candidatos y votaciones.
3. **Base de Datos:**
   Utiliza **PostgreSQL** como sistema de gestión de base de datos relacional. Contiene las tablas y relaciones necesarias para el almacenamiento estructurado de la información del sistema, garantizando integridad y consistencia de los datos.

Descripción de Componentes

A continuación, se describen los principales componentes que conforman el sistema:

- **Componente de Autenticación:**
  Permite el inicio y cierre de sesión de los usuarios registrados. Se implementa un sistema de validación mediante credenciales, y la sesión se mantiene activa durante la interacción con la plataforma.
- **Componente de Administración:**
  Facilita la gestión de entidades principales como usuarios, candidatos y procesos de votación. Este componente está restringido únicamente a usuarios con privilegios de administrador.
- **Componente de Votación:**
  Es el módulo central del sistema. Permite al usuario emitir su voto de manera digital, validando que no se repita y registrándolo en la base de datos de forma segura.
- **Componente de Resultados:**
  Muestra los resultados de las votaciones en tiempo real o al finalizar el proceso. Los datos se obtienen desde la base de datos y se renderizan dinámicamente en la interfaz.
- **Componente de Conexión con la API:**
  Responsable de realizar las peticiones al servidor mediante la librería **Axios**, enviando y recibiendo información en formato JSON.

Descripción del uso de SASS/SCSS

Durante el desarrollo del frontend se empleó SASS/SCSS como preprocesador de estilos con el fin de optimizar el mantenimiento y la reutilización del código CSS.
El uso de SASS permitió una estructura más ordenada mediante la creación de variables, mixins y módulos de estilos reutilizables en distintos componentes del proyecto.

Entre los principales beneficios obtenidos se destacan:

- Definición de una paleta de colores global mediante variables.
- Modularización de los estilos por componente, mejorando la legibilidad y el control de los archivos.
- Utilización de anidamiento para una sintaxis más limpia y jerárquica.
- Facilita la consistencia visual en todas las páginas de la aplicación.

Esta metodología contribuyó a mantener un diseño coherente, escalable y fácil de modificar, lo que resulta fundamental en aplicaciones con múltiples vistas y componentes.

Diagrama Entidad-Relación (E-R)

El modelo entidad–relación del sistema de votaciones digitales desarrollado en **PostgreSQL** representa la estructura lógica y las relaciones entre los distintos elementos del sistema. Este modelo asegura la **integridad referencial** de los datos, evitando duplicidades y permitiendo una administración eficiente de la información.

A continuación, se describen las principales **entidades** y sus **relaciones**:

---

1. administradores

Esta tabla almacena la información de los usuarios con privilegios administrativos dentro del sistema.
Atributos principales:

- **id:** Identificador único del administrador (clave primaria).
- **nombre:** Nombre completo del administrador.
- **email:** Correo electrónico único, utilizado como credencial de acceso.
- **password:** Contraseña cifrada para autenticación.
- **fecha_creacion:** Fecha y hora de creación del registro.

Los administradores tienen acceso al panel de gestión, desde donde pueden administrar campañas, candidatos y monitorear la actividad del sistema.

---

2. ingenieros_colegiados

Representa a los usuarios registrados que pueden participar en los procesos de votación.
Atributos principales:

- **id:** Identificador único del ingeniero (clave primaria).
- **nombre, dpi, email, password:** Datos personales y de autenticación.
- **departamento_id, municipio_id:** Relaciones con las tablas **departamentos** y **municipios**.
- **ha_votado:** Indica si el usuario ya emitió su voto.
- **numero_colegiado:** Código único asignado a cada profesional.
- **rol:** Define si el usuario es “usuario” o “admin”.

Relaciones:

- Cada ingeniero **pertenece a un departamento** y a un **municipio**.
- Puede **emitir múltiples votos** a lo largo de distintas campañas.

3. campañas

Corresponde a los procesos electorales disponibles dentro del sistema.
Atributos principales:

- **id:** Identificador único de la campaña.
- **nombre, descripcion, titulo:** Información general de la campaña.
- **color, logo_url:** Datos visuales usados en la interfaz.
- **estado:** Puede ser *programada*, *activa* o *finalizada*.
- **fecha_inicio, fecha_fin:** Periodo durante el cual la campaña está disponible.

Relaciones:

- Una campaña puede **contener múltiples candidatos**.
- Cada voto está **asociado a una única campaña**.
- Se relaciona con la tabla **votos_por_campaña** para registrar cuántos votos emitió cada ingeniero.

---

4. candidatos

Representa a las personas postuladas dentro de cada campaña.
Atributos principales:

- **id:** Identificador único del candidato.
- **nombre, numero_colegiado, especialidad:** Datos personales.
- **cargo_id:** Hace referencia al cargo por el cual compite.
- **campaña_id:** Relación directa con la campaña a la que pertenece.
- **foto_url:** Enlace a la imagen del candidato.

Relaciones:

- Cada candidato **pertenece a una campaña**.
- Cada candidato **opta a un cargo específico** definido en la tabla **cargos_directiva**.
- Un candidato puede recibir **muchos votos**, cada uno proveniente de un ingeniero distinto.

---

5. cargos_directiva

Define los cargos disponibles dentro de la directiva o junta que se elige mediante las votaciones.

Atributos principales:

- **id:** Identificador único.
- **nombre:** Nombre del cargo (por ejemplo: Presidente, Secretario, Tesorero).
- **descripcion:** Detalle del rol del cargo.
- **orden:** Determina el orden jerárquico o de presentación.

Relaciones:

- Cada cargo puede estar **asociado a múltiples candidatos**.
- Cada voto está **vinculado a un cargo específico**, lo que garantiza que cada votante elija a un candidato por puesto.

---

6. departamentos y municipios

Estas tablas definen la estructura territorial del país para registrar la ubicación de los ingenieros colegiados.

departamentos

- **id, nombre:** Identificador y nombre del departamento.
- Se relaciona con **municipios** (uno a muchos).
- Se relaciona con **ingenieros_colegiados** (uno a muchos).

municipios

- **id, nombre, departamento_id:** Identificador, nombre del municipio y referencia al departamento al que pertenece.
- Cada municipio está **asociado a un departamento** mediante una clave foránea.

---

7. votos

Es la tabla central del sistema, ya que almacena cada voto emitido por los ingenieros colegiados.

Atributos principales:

- **id:** Identificador único del voto.
- **ingeniero_id:** Relación con el ingeniero que emitió el voto.
- **candidato_id:** Candidato seleccionado.
- **cargo_id:** Cargo votado.
- **departamento_id, municipio_id:** Lugar de origen del voto.
- **campaña_id:** Campaña electoral correspondiente.
- **fecha_voto:** Fecha y hora del voto.

Restricciones y relaciones:

- Cada ingeniero solo puede **votar una vez por cargo y campaña** (*votos_ingeniero_cargo_campaña_key*).
- Se implementan múltiples **claves foráneas** hacia las tablas relacionadas para mantener la integridad referencial.
- Un **trigger** actualiza los totales de votos por campaña después de cada inserción.

---

8. votos_por_campaña

Tabla auxiliar que lleva el conteo de los votos emitidos por cada ingeniero en cada campaña.
Atributos principales:

- **id:** Identificador único.
- **ingeniero_id, campaña_id:** Relación entre el votante y la campaña.
- **votos_emitidos:** Número de votos registrados.
- **ultima_actualizacion:** Fecha de la última modificación.

Relaciones:

- Cada registro vincula un ingeniero con una campaña específica.
- Ayuda a controlar la cantidad máxima de votos permitidos por votante.

---

9. categorias (opcional)

Tabla auxiliar que permite clasificar campañas, cargos o candidatos por tipo, área o especialidad.
Atributos principales:

- **id:** Identificador único.
- **nombre:** Nombre de la categoría.
- **descripcion:** Detalle del tipo de categoría.

Resumen de Relaciones Principales

Relación	Descripción
administradores – campañas	Un administrador puede gestionar varias campañas.
campañas – candidatos	Una campaña puede tener varios candidatos.
cargos_directiva – candidatos	Cada candidato pertenece a un cargo específico.
ingenieros_colegiados – votos	Un ingeniero puede emitir múltiples votos.
votos – campañas	Cada voto pertenece a una campaña específica.
votos – candidatos	Cada voto se emite por un candidato determinado.
departamentos – municipios	Un departamento contiene múltiples municipios.
municipios – ingenieros_colegiados	Un ingeniero se registra en un municipio.
ingenieros_colegiados – votos_por_campaña	Controla los votos emitidos por campaña.