Proyecto Final: "Plataforma **de Datos para el Sistema Nacional de Voto Electrónico** 'VotoSeguro **2026'"**

Visión: Construir la arquitectura de datos subyacente que garantice la integridad, transparencia, seguridad y auditabilidad del proceso electoral presidencial de 2026, desde la emisión del voto hasta la proclamación de resultados en tiempo real.

Fecha de Entrega: Jueves 30 de Julio de 2025.

---

1. Objetivo General

Diseñar e implementar una arquitectura de datos simulada en PostgreSQL que soporte el ciclo de vida completo de un proceso de votación electrónica a nivel nacional. El sistema debe ser capaz de registrar votos de manera segura, procesarlos de forma anónima, agrupaciones y sumarizaciónes para obtener resultados y proporcionar datos para auditorías exhaustivas, todo ello garantizando la confianza pública en el proceso.

2. Alcance del Proyecto

Dentro del Alcance (In-Scope):

- Registro de Votantes: Simulación de un padrón electoral nacional.
- Emisión del Voto: Arquitectura para recibir y almacenar de forma segura los votos emitidos.
- Procesamiento y Conteo: Lógica para validar, anonimizar y contar los votos.
- Publicación de Resultados: Modelado de datos para dashboards de resultados en tiempo real (nivel nacional, regional, por centro de votación).
- Auditoría: Diseño de tablas y vistas que permitan a un ente auditor externo verificar la integridad del proceso (ej: "hubo X votos emitidos y se contaron X votos").
- Simulación de Carga: El diseño debe considerar picos de carga durante la jornada electoral.
- Sensibilidad de encriptación de los datos.

Fuera del Alcance (Out-of-Scope):

- La aplicación cliente (Front-End): No construiremos la interfaz web o la aplicación móvil donde el ciudadano vota. Asumiremos que esta aplicación existe y nos envía los datos.
- Autenticación biométrica/criptografía avanzada: Aunque son cruciales, nos enfocaremos en la arquitectura de los datos que reciben la información, no en los métodos criptográficos de la emisión del voto en sí.
- Infraestructura física de red y hardware: Nos centraremos en la arquitectura lógica de la base de datos.

---

3. Requerimientos Funcionales (RF)

RF-01: Carga del Padrón Electoral: El sistema debe permitir la carga masiva de un padrón electoral con los datos de los ciudadanos habilitados para votar.

RF-02: Verificación de Votante: Antes de emitir un voto, el sistema debe poder verificar si un ciudadano está en el padrón y si no ha votado previamente. En este proceso se debe incluir la validación de los datos Biométricos (reconocimiento facial a partir de la foto del padrón electoral de la cedula de identidad + los datos de las huellas digitales).

RF-03: Emisión Segura del Voto: El sistema debe registrar cada voto recibido. Crucial: El voto debe ser almacenado de tal manera que sea imposible vincularlo directamente con el votante que lo emitió una vez que ha sido validado.

RF-04: Conteo de Votos: El sistema debe ser capaz de contar los votos para cada candidato a nivel de mesa de votación, centro de votación (Distrito / Centro Penal / Albergues de Ancianos y Consulados o Embajadas), Cantón (suman los datos de todos los centros de votación de los distritos), Provincia (suma todo los votos de los centros de votación de los Cantones) y nacional.

RF-05: Actualización de Resultados en Tiempo Real: El sistema debe proveer datos agregados (agrupaciones, sumarización y otros para la generación de estadísticas) que puedan ser consultados por un dashboard público para mostrar el progreso del conteo en tiempo real (transparencia electoral).

RF-06: Generación de Reportes de Auditoría: El sistema debe generar reportes que permitan verificar:

- Total de votantes en el padrón.
- Total de votantes que emitieron su voto.
- Total de votos recibidos.
- Total de votos contados (válidos, nulos, en blanco).
- Consistencia entre los totales (votos emitidos vs. votos contados).

---

4. Requerimientos No Funcionales (RNF)

RNF-01: Integridad de Datos: El sistema debe garantizar que un voto no pueda ser alterado, duplicado o eliminado después de ser emitido (Inmutabilidad).

RNF-02: Anonimato: Debe ser computacionalmente inviable rastrear un voto específico hasta el votante que lo emitió en la capa de datos de conteo.

RNF-03: Escalabilidad: La arquitectura debe soportar un alto volumen de escrituras concurrentes durante las horas pico de la votación (ej: primera hora de la mañana y última hora de la tarde).

RNF-04: Disponibilidad: El sistema debe tener una alta disponibilidad (simulada) durante toda la jornada electoral. Un fallo en la base de datos es inaceptable.

RNF-05: Seguridad: El acceso a los diferentes schemas (capas de datos) debe estar estrictamente controlado. Por ejemplo, el sistema que cuenta los votos no debe tener acceso a la tabla que vincula al votante con su estado de "ya votó".

RNF-06: Auditabilidad: Cada paso del proceso de datos (ingesta, validación, conteo) debe dejar un rastro (log de auditoria) que pueda ser auditado posteriormente.
RNF-07: Partidos Políticos, “MAE”, “Cebollitas”, “PERA”, “LA LIGA”, “PAS”

---

5. Entregables

1. Scripts SQL: Archivos .sql organizados para crear los schemas (bronze, silver, gold), las tablas, las vistas y las funciones de PostgreSQL.
2. Scripts de Datos (Opcional): Scripts en Python/Faker para generar datos de prueba realistas (simulación de votos). **[El profesor les ayudará con un generador de datos sintéticos para los con Python]**, o bien los estudiantes podrían implementar una función que genere los datos a insertar de los votó con un random por partido político.
3. Diagrama de Arquitectura de la solución tecnológica (Draw.io)
4. Diagrama de Arquitectura: Un diagrama claro mostrando los schemas bronze, silver, gold y el flujo de datos entre ellos para este caso de uso específico.
5. Modelo de Datos: Diagramas Entidad-Relación para cada schema, explicando el propósito de cada tabla.
6. Justificación de Diseño: Explicación detallada de las decisiones clave. Por ejemplo:
    - ¿Cómo se logra el anonimato del voto en el diseño de las tablas?
    - ¿Qué estrategia se usó para garantizar la integridad y evitar la duplicación de votos?
    - ¿Cómo está estructurado el schema
    - ¿Como se tiene la seguridad de los datos?
    - ¿Como se gestiono la clasificación, sensibilidad y encriptación de los datos?
    - ¿Como se resolvio el tema de la Auditoría de los datos para garantizar la transparencia de las votaciones?
    -
7. Plan de Auditoría: Descripción de cómo un auditor usaría las tablas y vistas creadas para verificar la elección.
8. Dashboard de Resultados: Un enlace a un dashboard público (Looker Studio, Power BI, etc.) o un video de demostración que muestre los resultados en tiempo real (simulados) y los reportes de auditoría. **[Opcional, pero premiado con puntos extras].**

---

6. Diccionario de Términos

- Padrón Electoral: Lista oficial de ciudadanos habilitados para votar. Contiene información de identificación (ID, nombre, centro de votación asignado).
- Voto Emitido (Capa Bronze): El registro crudo que llega desde la aplicación cliente. Podría contener un token temporal de la sesión del votante y el voto cifrado. Es una "caja negra" que aún no se ha abierto.

- Voto Validado (Capa Silver): Un voto que ha sido verificado como proveniente de un votante habilitado que no ha votado antes. En esta etapa, el vínculo con la identidad del votante se rompe.
- Voto Contabilizado (Capa Gold): Un voto anónimo que ha sido agregado a los totales de un candidato en las tablas dimensionales.
- Token de Voto: Un identificador único y de un solo uso generado cuando un votante se autentica. Se usa para enviar el voto y marcar al votante como "ya votó", pero se descarta antes de que el voto sea contado.
- Inmutabilidad: Principio que asegura que una vez que un registro (como un voto) es escrito, no puede ser modificado. Se puede simular con triggers o reglas en PostgreSQL.

Propuesta de Arquitectura Lógica en PostgreSQL

Para inspirar su diseño, aquí tienen un posible enfoque:

Schema Bronce

- bronze.electoral_roll_raw: Carga inicial del padrón en formato texto.
- bronze.incoming_votes: Tabla que recibe los votos. Columnas: vote_id (UUID), voter_token (JWT o similar), encrypted_vote_payload (JSONB), received_at (TIMESTAMP).

Schema Plata

- silver.voters: El padrón electoral limpio y tipado. Columnas: voter_id (PK), name, region_id, polling_station_id, has_voted (BOOLEAN, default FALSE).
- silver.valid_anonymous_votes: La "urna digital". Aquí es donde la magia ocurre. Columnas: vote_id (PK), candidate_id, polling_station_id, processed_at. Noten la ausencia de .
- Proceso Clave (a implementar con una función SQL o un script):
- Leer un voto de bronze.incoming_votes.
- Decodificar el voter_token para obtener el voter_id.
- Verificar en silver.voters si voter_id existe y has_voted es FALSE.
  Si es válido:
  a. Iniciar una transacción,
  b. Actualizar silver.voters SET has_voted = TRUE WHERE id = voter_id.
  c. Decodificar el encrypted_vote_payload para obtener el candidate_id.
  d. Insertar en silver.valid_anonymous_votes (candidate_id, polling_station_id).
  e. Confirmar la transacción (COMMIT).
  Si no es válido: Moverlo a una tabla de votos inválidos para auditoría.

Schema Oro

- gold.dim_candidates: Tabla de dimensión de candidatos.
- gold.dim_geography: Dimensión geográfica (región, ciudad, centro de votación).

- gold.fct_vote_count: Tabla de hechos agregada. Columnas: geography_key, candidate_key, total_votes. Esta tabla se actualizaría periódicamente (ej: cada minuto) para alimentar el dashboard.
- gold.audit_summary_view: Una vista materializada que muestra en tiempo real: total_voters, total_voted, total_valid_votes, total_invalid_votes.