

Proyecto: Base de Datos para un Sistema de Votación Electrónica

Grupo17: Daniela Sanabria Mosquera



CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de posgrado, aplicando todos los requisitos de las normas ISO implementados en sus sedes Neiva y Pácora"



Creación

Seguridad y Roles



Crea roles para acceso al sistema con distintas funciones.

CONNECT: pueden ingresar a la base de datos.

SELECT: pueden leer las tablas.

INSERT, UPDATE: solo el rol oficial puede emitir o modificar votos.

USAGE: permite el uso del esquema.

```
14 CREATE ROLE admin LOGIN PASSWORD 'admin123';
15 CREATE ROLE oficial LOGIN PASSWORD 'mesa123';
16 CREATE ROLE auditor LOGIN PASSWORD 'audit123';
17 CREATE ROLE observador LOGIN PASSWORD 'view123';
18
19
20 GRANT CONNECT ON DATABASE votacion TO oficial, auditor, observador;
21 GRANT SELECT ON ALL TABLES IN SCHEMA public TO observador, auditor;
22 GRANT INSERT, UPDATE ON VotosEmitidos TO oficial;
23 GRANT USAGE ON SCHEMA public TO oficial, auditor, observador;
24
```

Data Output Messages Notifications

RANT

Query returned successfully in 143 msec.

services:

▷ Run Service

myDB:

```
image: postgres:17.2
container_name: votacion_db
restart: always
ports:
  - 5434:5432
environment:
  - POSTGRES_USER=adminDaniela
  - POSTGRES_PASSWORD=candado/silla/mostaza
  - POSTGRES_DB=votacion
volumes:
  - ./postgres:/var/lib/postgresql/data
```

▷ Run Service

pdAdmin:

```
image: dpape/pgadmin4
container_name: pgadmin5
restart: always
depends_on:
  - myDB
ports:
  - 8089:80
environment:
  - PGADMIN_DEFAULT_EMAIL=admin@votacion.com
  - PGADMIN_DEFAULT_PASSWORD=candado/silla/mostaza
volumes:
  - ./pgadmin:/var/lib/pgadmin
  - ./pgadmin:/certs/server.cert
  - ./pgadmin:/certs/server.key
  - ./pgadmin:/pgadmin4/servers.json
```

PREPARACIÓN DEL ENTORNO

Creamos el docker-
compose.yml con las
características definidas.



Base de datos votacion

Tablas:

- TiposEleccion
- Elecciones
- PartidosPoliticos
- Candidatos
- Votantes
- PadronElectoral
- CentrosVotacion
- MesasElecttorales
- VotosEmitidos
- ResultadosMesa
- ResultadosCentro
- ResultadosGenerales
- Auditores
- UsuariosSistema
- LogsSistema

Creación



```
Query  Query History
90  CREATE TABLE Auditores (
91      id_auditor SERIAL PRIMARY KEY,
92      nombre_completo VARCHAR(150),
93      correo VARCHAR(100),
94      activo BOOLEAN DEFAULT TRUE
95  );
96
97  CREATE TABLE UsuariosSistema (
98      id_usuario SERIAL PRIMARY KEY,
99      usuario VARCHAR(50) UNIQUE NOT NULL,
100     rol VARCHAR(50) CHECK (rol IN ('admin', 'oficial', 'auditor', 'observador')),
101     hash_password TEXT NOT NULL,
102     activo BOOLEAN DEFAULT TRUE
103  );
104
105  CREATE TABLE LogsSistema (
106      id_log SERIAL PRIMARY KEY,
107      id_usuario INT REFERENCES UsuariosSistema(id_usuario),
108      fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
109      accion TEXT,
110      detalle TEXT
111  );
112
```


Creación



Activa la extensión pgcrypto,
necesaria para:

- Generar UUIDs con `gen_random_uuid()`.
- Cifrar contraseñas con `crypt()`.

```
104
105 ▼ CREATE TABLE LogsSistema (
106     id_log SERIAL PRIMARY KEY,
107     id_usuario INT REFERENCES UsuariosSistema(id_usuario),
108     fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
109     accion TEXT,
110     detalle TEXT
111 );
112
113
114 CREATE EXTENSION IF NOT EXISTS "pgcrypto";
115
```

Data Output Messages Notifications

CREATE EXTENSION

Query returned successfully in 551 msec.

```

39 ('123456789', 'Carlos Goto', '1980-04-20'),
40 ('87654321B', 'Lucía Ramírez', '1992-11-10'),
41 ('56781234C', 'Diego Torres', '1975-06-30'),
42 ('43218765D', 'María León', '2000-09-15'),
43 ('98761234E', 'Elena Vargas', '1985-03-01');
44
45
46 ✓ INSERT INTO PadronElectoral(id_votante, id_eleccion, habilitado)
47 VALUES (1, 1, true), (2, 1, true), (3, 1, false), (4, 1, true), (5, 1, true);
48
49
50 ✓ INSERT INTO CentrosVotacion(nombre, direccion)
51 VALUES ('Centro Cívico 1', 'Calle Falsa 123'), ('Centro Comunal 2', 'Av. Central 456');
52
53
54 ✓ INSERT INTO MesasElectORAles(numero_mesa, id_centro)
55 VALUES (1, 1), (2, 1), (1, 2);
56
57
58 ✓ INSERT INTO UsuariosSistema(usuario, rol, hash_password, activo)
59 VALUES
60 ('admin1', 'admin', crypt('admin123', gen_salt('bf')), true),
61 ('oficial1', 'oficial', crypt('mesa123', gen_salt('bf')), true),
62 ('auditor1', 'auditor', crypt('audit123', gen_salt('bf')), true),
63 ('observador1', 'observador', crypt('view123', gen_salt('bf')), true);
64
65

```

Data Output Messages Notifications

INSERT 0 4

Query returned successfully in 184 msec.

INSERCIONES

Realizamos las respectivas inserciones para cada una de las tablas con más de cinco datos.



Vistas



```
166 CREATE VIEW VotantesHabilitadosMesa AS
167 SELECT v.nombre_completo, m.numero_mesa, c.nombre AS centro
168 FROM PadronElectoral p
169 JOIN Votantes v ON p.id_votante = v.id_votante
170 JOIN MesasElecttorales m ON p.id_votante % 3 + 1 = m.id_mesa
171 JOIN CentrosVotacion c ON m.id_centro = c.id_centro
172 WHERE p.habilitado = TRUE;
173
174 SELECT * FROM VotantesHabilitadosMesa;
```

Data Output Messages Notifications



	nombre_completo character varying (150)	numero_mesa integer	centro character varying (150)
1	Carlos Soto	2	Centro Cívico 1
2	Lucía Ramírez	1	Centro Comunal 2
3	María León	2	Centro Cívico 1
4	Elena Vargas	1	Centro Comunal 2

```
177 CREATE VIEW CandidatosPorEleccion AS
178 SELECT c.nombre_completo, p.nombre AS partido, e.nombre AS eleccion
179 FROM Candidatos c
180 JOIN PartidosPolíticos p ON c.id_partido = p.id_partido
181 JOIN Elecciones e ON c.id_eleccion = e.id_eleccion;
182
183 SELECT * FROM CandidatosPorEleccion;
```

-- Resultados parciales por centro

```
186 CREATE VIEW ResultadosParcialesCentro AS
187 SELECT ce.nombre, ca.nombre_completo, SUM(rm.votos) AS total_votos
```

Data Output Messages Notifications



	nombre_completo character varying (150)	partido character varying (100)	eleccion character varying (200)
1	Juan Pérez	Partido Democrático	Elección Presidencial 2025

Muestra los votantes habilitados y su mesa asignada según un cálculo ($\% 3 + 1$).

Lista los candidatos con su partido y elección correspondiente.

Permiten visualización de auditoría y resultados generales por candidato.

Muestra cuántos votos tiene cada candidato por centro.

Vistas

```
CREATE VIEW AuditoriaAccesos AS
SELECT l.fecha, u.usuario, l.accion
FROM LogsSistema l
JOIN UsuariosSistema u ON l.id_usuario = u.id_usuario;

SELECT * FROM AuditoriaAccesos;
```

```
CREATE VIEW ResultadosGeneralesView AS
SELECT c.nombre_completo, SUM(rg.votos) AS total
FROM ResultadosGenerales rg
JOIN Candidatos c ON rg.id_candidato = c.id_candidato
GROUP BY c.nombre_completo;
```

Output Messages Notifications

fecha	usuario	accion
timestamp without time zone	character varying (50)	text
2025-05-21 03:00:55.354541	admin1	Inserción de Elección
2025-05-21 03:00:55.354541	oficial1	Registro de Voto
2025-05-21 03:00:55.354541	oficial1	Registro de Voto
2025-05-21 03:00:55.354541	auditor1	Auditoría

```
CREATE VIEW ResultadosParcialesCentro AS
SELECT ce.nombre, ca.nombre_completo, SUM(rm.votos) AS total_votos
FROM ResultadosMesa rm
JOIN MesasElecttorales me ON rm.id_mesa = me.id_mesa
JOIN CentrosVotacion ce ON me.id_centro = ce.id_centro
JOIN Candidatos ca ON rm.id_candidato = ca.id_candidato
GROUP BY ce.nombre, ca.nombre_completo;

SELECT * FROM ResultadosParcialesCentro;
```

```
CREATE VIEW AuditoriaAccesos AS
SELECT l.fecha, u.usuario, l.accion
```

Output Messages Notifications

nombre	nombre_completo	total_votos
character varying (150)	character varying (150)	bigint
Centro Comunal 2	Ana Gómez	2
Centro Cívico 1	Juan Pérez	3
Centro Cívico 1	Ana Gómez	3
Centro Comunal 2	Juan Pérez	2




```
79 WHERE id_votante = votante_id AND id_eleccion = eleccion_id AND habilitado = TRUE
80 );
81 END;
82 $$ LANGUAGE plpgsql;
83
84
85 CREATE PROCEDURE RegistrarVoto(votante_id INT, eleccion_id INT, candidato_id INT, mesa_id INT)
86 LANGUAGE plpgsql AS $$
87 BEGIN
88     IF NOT VerificarHabilitacion(votante_id, eleccion_id) THEN
89         RAISE EXCEPTION 'Votante no habilitado';
90     END IF;
91
92     INSERT INTO VotosEmitidos(id_eleccion, id_candidato, id_mesa)
93     VALUES (eleccion_id, candidato_id, mesa_id);
94
95
96     UPDATE ResultadosMesa
97     SET votos = votos + 1
98     WHERE id_mesa = mesa_id AND id_candidato = candidato_id;
99
100
101     UPDATE PadronElectoral SET habilitado = FALSE
102     WHERE id_votante = votante_id AND id_eleccion = eleccion_id;
103 END;
104 $$;
```

Data Output Messages Notifications

CREATE PROCEDURE

Query returned successfully in 108 msec.



Procedimiento

Este procedimiento:

- Verifica si el votante puede votar.
- Registra el voto.
- Incrementa el conteo en ResultadosMesa.
- Marca al votante como no habilitado (evita que vote otra vez).

TRIGGER

- Se ejecutará automáticamente cuando haya un cambio en el padrón electoral.
- Usa row_to_json(NEW) para registrar los datos nuevos.
- current_setting('app.user')::INT: obtiene el ID del usuario activo (si se configura).
- Se ejecuta después de cada actualización del padrón electoral. Genera una entrada en LogsSistema.
- Guarda un registro cada vez que se emite un voto.
- Se ejecuta automáticamente al registrar un voto. Guarda un log del evento.

```
CREATE OR REPLACE FUNCTION LogPadronChange() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO LogsSistema(id_usuario, accion, detalle)
    VALUES (current_setting('app.user')::INT, 'MODIFICACION_PADRON', row_to_json(NEW)::TEXT);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_log_padron_mod
AFTER UPDATE ON PadronElectoral
FOR EACH ROW EXECUTE FUNCTION LogPadronChange();

CREATE OR REPLACE FUNCTION LogVotoEmitido() RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO LogsSistema(id_usuario, accion, detalle)
    VALUES (current_setting('app.user')::INT, 'REGISTRO_VOTO', row_to_json(NEW)::TEXT);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tr_log_voto
AFTER INSERT ON VotosEmitidos
FOR EACH ROW EXECUTE FUNCTION LogVotoEmitido();
```

```
CREATE OR REPLACE FUNCTION ActualizarResultadosDespuesVoto() RETURNS TRIGGER AS $$  
BEGIN
```

```
INSERT INTO ResultadosMesa(id_mesa, id_candidato, votos)  
VALUES (NEW.id_mesa, NEW.id_candidato, 1)  
ON CONFLICT (id_mesa, id_candidato) DO UPDATE  
SET votos = ResultadosMesa.votos + 1;
```

```
UPDATE ResultadosCentro rc  
SET votos = rc.votos + 1  
FROM MesasElectorales me  
WHERE rc.id_centro = me.id_centro  
AND rc.id_candidato = NEW.id_candidato  
AND me.id_mesa = NEW.id_mesa;
```

```
INSERT INTO ResultadosCentro(id_centro, id_candidato, votos)  
SELECT me.id_centro, NEW.id_candidato, 1  
FROM MesasElectorales me  
WHERE me.id_mesa = NEW.id_mesa  
AND NOT EXISTS (  
SELECT 1 FROM ResultadosCentro rc  
WHERE rc.id_centro = me.id_centro  
AND rc.id_candidato = NEW.id_candidato);
```

Output Messages Notifications

FUNCTION

```
DROP TRIGGER IF EXISTS tr_actualizar_resultados ON VotosEmitidos;  
CREATE TRIGGER tr_actualizar_resultados  
AFTER INSERT ON VotosEmitidos  
FOR EACH ROW  
EXECUTE FUNCTION ActualizarResultadosDespuesVoto();
```

ta Output Messages Notifications

EATE TRIGGER

TRIGGER



Actualiza automáticamente las tablas de resultados (mesa, centro, generales) cuando se registra un voto.

Usa ON CONFLICT - DO UPDATE para evitar errores si ya existen registros.

Tiene relación con el anterior por cada vez que se inserta un nuevo voto, se actualizan automáticamente los resultados.

**QUIERE SER
SEGURIDAD**

**NO ENTIENDE
EN BASES**

**HE SOBRESCRITO
LA BASE DE DATOS
UNA VERSIÓN ANTERIOR**

GRACIAS POR SU ATENCION.

APLAUDAN Y NO HAGAN PREGUNTAS

