Informe Trabajo Obligatorio

Bases de Datos II

Universidad Católica del Uruguay

25 de junio 2025

GRUPO 6

Luis Di Muro, Leandro Casaretto

Contenido

Introducción

Este informe documenta el proceso completo de diseño e implementación de un sistema de gestión de elecciones nacionales en Uruguay. El objetivo principal fue modelar conceptual y físicamente una base de datos robusta y normalizada que permitiera registrar, controlar y procesar los votos emitidos en cada circuito, así como administrar mesas, listas, papeletas, y candidatos.

El trabajo se desarrolló en distintas etapas, comenzando con el análisis de requisitos, la construcción del modelo entidad-relación (MER), su transformación al modelo lógico y físico, y la implementación del sistema en un entorno tecnológico. A lo largo del informe se detallan las decisiones tomadas en cada fase, los fundamentos teóricos aplicados y las perspectivas de desarrollo futuro.

Además, se incluye un análisis de alternativas posibles que fueron evaluadas durante la planificación, así como una reflexión final sobre el trabajo realizado.

Marco teórico

El desarrollo de sistemas basados en bases de datos requiere la aplicación de principios teóricos que guíen las decisiones de modelado, diseño lógico y físico, así como la posterior implementación. Este trabajo se apoyó en conceptos de la teoría de bases de datos provista en clase, arquitectura de software y principios propios del sistema electoral.

Modelo Entidad-Relación

Como bien sabemos, el modelo entidad-relación (MER), permite representar de forma conceptual la estructura de una base de datos mediante entidades, relaciones y atributos. Este enfoque facilita el análisis de requerimientos y la identificación de restricciones semánticas. En este proyecto se aplicó una metodología sistemática para construir el

MER, incluyendo refinamientos como generalizaciones, agregaciones y relaciones entre relaciones.

Normalización

La normalización es el proceso de estructurar una base de datos con el fin de reducir la redundancia y mejorar la integridad. Siendo las primeras tres formas normales:

- **1FN**: eliminación de atributos multivaluados o compuestos.
- **2FN**: eliminación de dependencias parciales en claves compuestas.
- **3FN**: eliminación de dependencias transitivas entre atributos no clave.

Estas formas garantizan un diseño lógico robusto, adaptable y sin anomalías de inserción, actualización o eliminación.

Arquitectura cliente-servidor

El sistema se diseñó bajo el paradigma cliente-servidor, una arquitectura distribuida que separa la lógica de presentación (cliente) de la lógica de negocio y acceso a datos (servidor). Esta separación promueve la escalabilidad, la reutilización y el mantenimiento del sistema.

APIs REST

REST (Representational State Transfer), es un estilo arquitectónico para el diseño de servicios web. Las APIs REST permiten la comunicación entre cliente y servidor mediante operaciones HTTP estandarizadas, manteniendo una interfaz uniforme y sin estado.

Patrón Modelo-Vista-Controlador (MVC)

El patrón MVC propone una división del sistema en tres componentes: el modelo (acceso a datos), la vista (interfaz de usuario) y el controlador (gestión de la lógica y flujo). En este trabajo se adoptó una variante desacoplada, donde la vista se implementa como una aplicación cliente separada (React), mientras que el modelo y controlador residen en el backend (Node.js).

Principios electorales

El sistema también se construyó respetando principios fundamentales del derecho electoral uruguayo, entre ellos:

- Secreto del voto: impide asociar la identidad del votante con su elección.
- Unicidad del voto: garantiza que cada persona vote solo una vez por elección.
- Cierre de mesa: una vez cerrada una mesa, no se pueden ingresar más votos ni visualizar resultados anticipadamente.

1

Modelo Conceptual

Metodología utilizada

Dado que el enunciado del problema nos presenta una visión general del sistema electoral y luego detalla distintos elementos específicos (votantes, elecciones, circuitos, listas, etc.), optamos por aplicar una metodología mixta.

Comenzamos con un enfoque descendente, identificando los conceptos principales (Persona, Elección, Circuito, Lista, etc.) para construir un esquema armazón. Luego, aplicamos primitivas descendentes y ascendentes para refinar las entidades, atributos y relaciones, incorporando complejidades como generalizaciones, agregaciones y relaciones N:N.

Análisis de requerimientos

A continuación, se presenta un análisis de los fragmentos que se consideraron más relevantes del enunciado, junto con los conceptos y relaciones identificados:

"En las elecciones participan votantes, ciudadanos mayores de 18 años que tramitaron su Credencial Cívica. De ellos se sabe además nombre completo y Cédula de Identidad."

- Entidad candidata: Persona
- Atributos: ci, cc, nombre, apellido

- Consideración: todos los actores del sistema (votantes, miembros de mesa, candidatos, agentes policiales) son personas

"Cada zona tiene establecimientos ..., cada establecimiento tiene varios circuitos. Cada circuito tiene una lista de credenciales, una mesa y está en un departamento."

- Entidades candidatas: Establecimiento, Circuito, Departamento
- Relaciones: Circuito → se ubica en → Establecimiento
- Relación derivada: Establecimiento → pertenece a → Departamento
- Atributos: zona, tipo, dirección, accesible (en Circuito)

"A los establecimientos se asignan agentes de la policía ... De ellos interesa saber nombre, CI, CC y comisaría."

- Entidad: AgentePolicial
- Relación: AgentePolicial → asignado a → Establecimiento
- Atributos adicionales: comisaría

"Cada circuito tiene una mesa, compuesta por tres ciudadanos: presidente, secretario y vocal."

- Entidades: MiembroMesa, Mesa
- Relación: MiembroMesa → asignado a → Mesa
- Atributos: organismo, rol (presidente, secretario, vocal)

"Del voto se sabe a qué lista(s) fue emitido, si fue válido, anulado o en blanco, circuito, observación, fecha y hora."

- Entidad: Voto

- Relaciones:
    - Voto → contabiliza para → Lista
    - Voto → contabiliza para → Papeleta
- Atributos: estado (válido/anulado/en blanco), es_observado
- Relacionado a: Circuito y Elección

"Una elección tiene listas y papeletas. Una lista tiene un número, pertenece a un partido y a un departamento."

- Entidades: Elección, Lista, Papeleta, Partido
- Relaciones:
    - Elección → usa → Lista / Papeleta
    - Lista → asociada a → Departamento
    - Lista → apoya a → Partido, Senado, FórmulaPresidencial

"Una fórmula presidencial tiene presidente y vicepresidente, ambos candidatos. Cada partido presenta una fórmula."

- Entidades: Candidato, FormulaPresidencial
- Relaciones:
    - FórmulaPresidencial → presidente → Candidato
    - FórmulaPresidencial → vicepresidente → Candidato

"Una lista puede tener muchos candidatos, y cada uno tiene un órgano (diputado, senador, edil) y un orden."

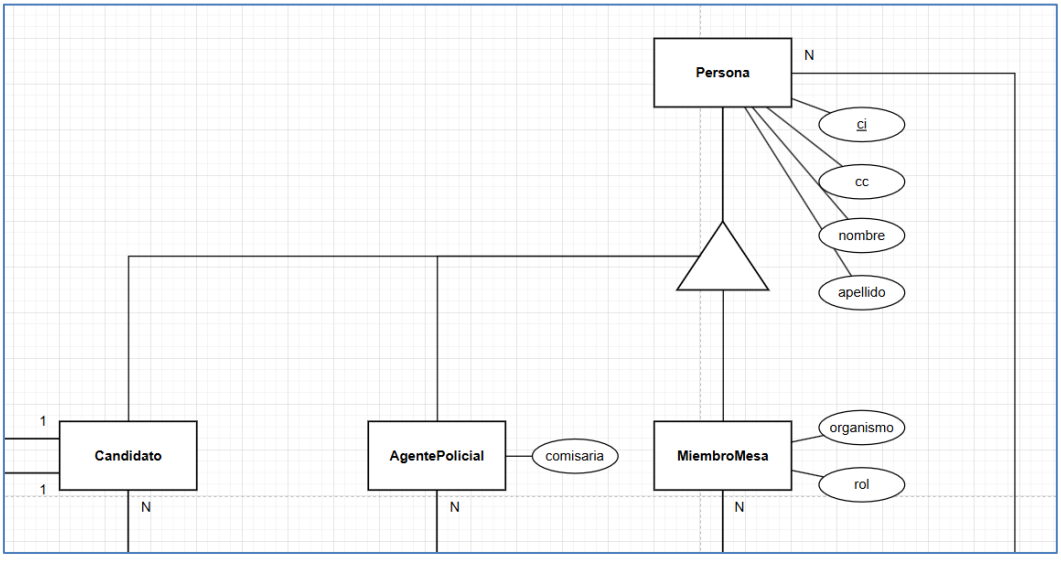
- Relación con atributos: Candidato → pertenece a → Lista
- Atributos: órgano, orden

Aplicación de primitivas

Una vez realizado el primer modelo inicial, se procedió a hacer un refinamiento mediante primitivas, tanto ascendentes como descendentes.

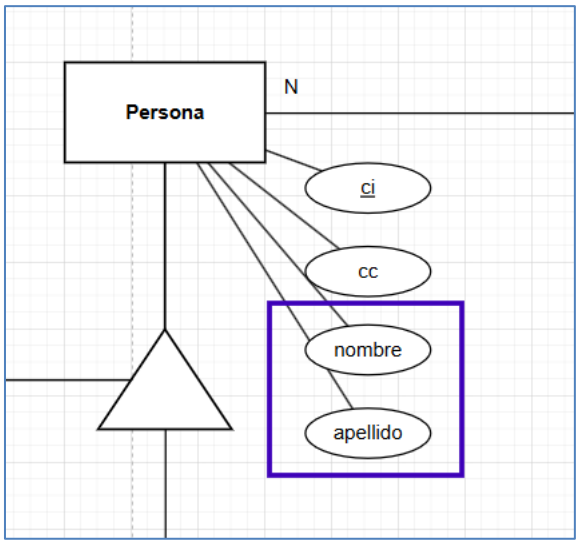
Primitiva descendente: Generalización

- Concepto inicial: Persona
- Aplicación: derivamos Candidato, MiembroMesa y AgentePolicial como subclases



Primitiva descendente: Desarrollo de atributos compuestos

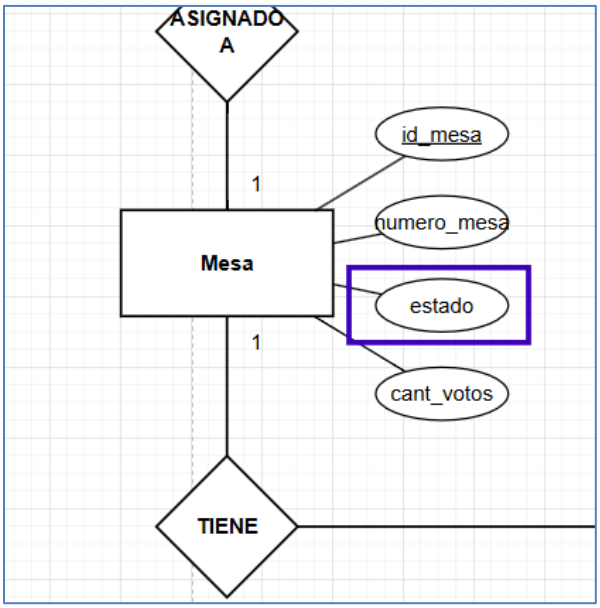
- Concepto inicial: nombre_completo
- Aplicación: se divide nombre y apellido en Persona

Primitiva 

descendente:

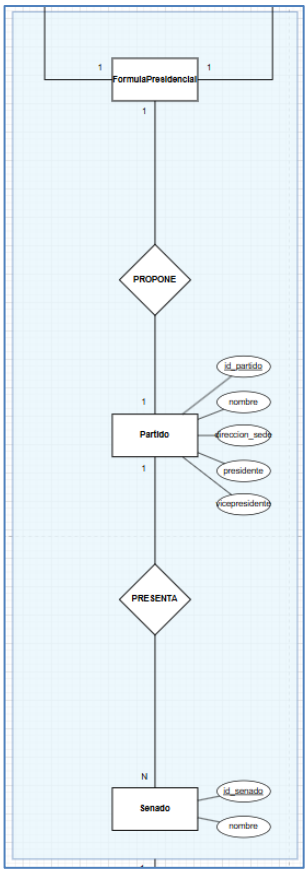
Desarrollo de atributos

- Concepto inicial: estado de la mesa (activa/cerrada)
- Aplicación: atributo "estado" en Mesa para controlar lógicamente la funcionalidad de cierre



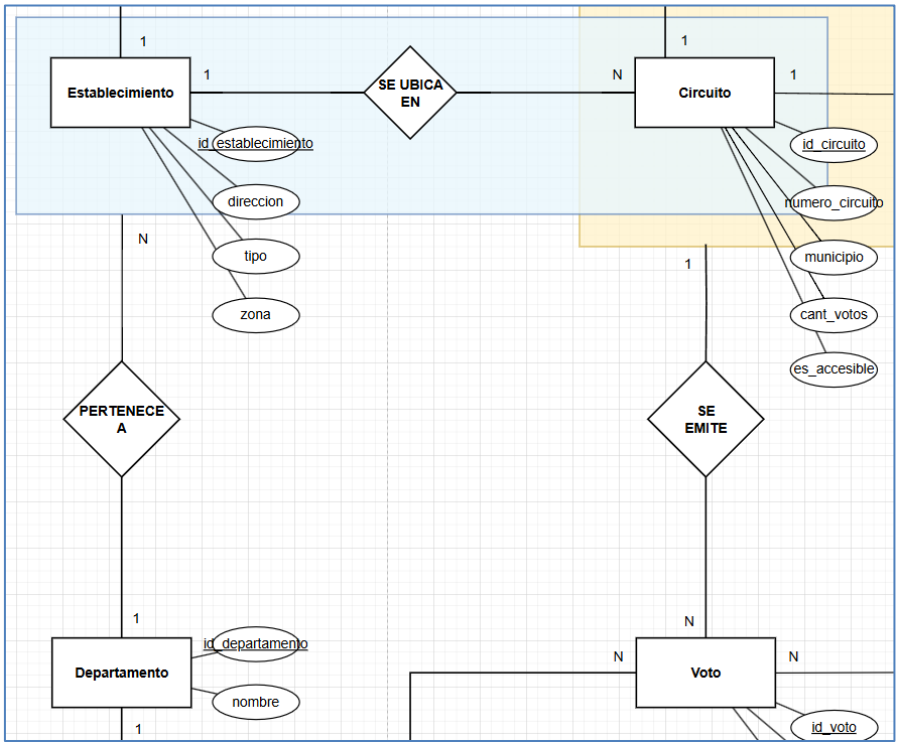
Primitiva ascendente: Agregación

- Concepto: Lista apoya una combinación de Partido, Senado y Fórmula
- Aplicación: agregación Lista_Apoya con cuatro FKs



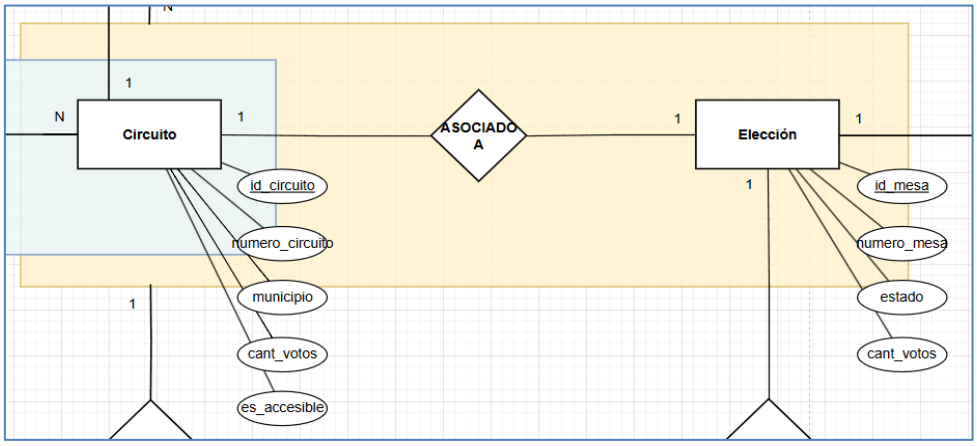
Primitiva ascendente: Agregación

- Concepto: Establecimiento + Circuito está en un Departamento
- Aplicación: agregación por contexto territorial



Primitiva ascendente: Agregación

- Concepto: Un voto ocurre en un circuito en el contexto de una elección
- Aplicación: agregación Circuito + Elección como contexto para emitir votos



Habiendo finalizado el refinamiento, las entidades, relaciones y atributos quedan conformados de la siguiente manera:

Entidades

Persona	Elección
Candidato	Papeleta
AgentePolicial	Voto
MiembroMesa	Lista
Mesa	Senado
Circuito	Partido
Establecimiento	FormulaPresidencial
Departamento	

Relaciones

R1: Persona **"VOTA EN"** Circuito x Elección -> (1,N)

Una persona vota en un circuito determinado para una elección concreta. La relación guarda la fecha y si fue observado.

R2: Circuito **"ASOCIADO A"** Elección -> (N,1)

Un circuito se asocia a una única elección. Una elección tiene muchos circuitos.

R3: MiembroMesa **"ASIGNADO A"** Mesa -> (N,1)

Cada miembro de mesa está asignado a una única mesa. Una mesa tiene varios miembros.

R4: Circuito **"TIENE"** Mesa -> (1,1)

Cada circuito tiene una y solo una mesa. La mesa corresponde exclusivamente a un circuito.

R5: Circuito **"SE UBICA EN"** Establecimiento -> (N,1)

Cada circuito se encuentra en un único establecimiento. Un establecimiento puede tener varios circuitos.

R6: AgentePolicial **"ASIGNADO A"** Establecimiento -> (N,1)

Un agente policial es asignado a un establecimiento. Un establecimiento puede tener varios agentes.

R7: Establecimiento + Circuito **"PERTENECE A"** Departamento -> (Agregación -> 1)

La agregación establecimiento-circuito se encuentra dentro de un único departamento.

R8: Voto **"SE EMITE EN"** Circuito x Elección -> (N,1)

Cada voto se emite en un circuito determinado y corresponde a una única elección.

R10: Elección **"USA"** Papeleta -> (1,N)

Una elección puede tener muchas papeletas asociadas (Ej. plebiscito seguridad social y allanamientos nocturnos en la última elección). Cada papeleta pertenece a una única elección.

R11: Elección "**USA**" Lista -> (1,N)

Una elección incluye varias listas. Cada lista pertenece a una única elección.

R12: Lista "**ASOCIADA A"** Departamento -> (N,1)

Una lista pertenece a un departamento. Un departamento tiene muchas listas.

R13: Voto "**CONTABILIZA PARA**" Lista -> (N,N)

Un voto puede contabilizarse para una lista, y una lista puede recibir muchos votos.

R14: Voto "**CONTABILIZA PARA**" Papeleta -> (N,N)

Un voto puede contener una papeleta, y una papeleta puede estar en muchos votos.

R15: Lista "**APOYA A**" Partido x Senado x FormulaPresidencial -> (Agregacion -> 1)

Una lista apoya una combinación de partido, senado y fórmula presidencial.

R16: Partido "**PRESENTA**" Senado -> (1,N)

Un partido puede presentar varias listas al senado. Cada senado pertenece a un único partido.

R17: Partido "**PROPONE**" FormulaPresidencial -> (1,1)

Un partido propone una única fórmula presidencial.

R18: FormulaPresidencial "**PRESIDENTE**" Candidato -> (1,1)

Una formula tiene un candidato a presidente. Ese candidato puede ser electo presidente por una única fórmula.

R19: FormulaPresidencial "**VICEPRESIDENTE**" Candidato -> (1,1)

Una formula tiene un vicepresidente. Ese candidato puede ser electo vicepresidente por una única fórmula.

R20: Candidato "**PERTENECE A**" Lista -> (N,N)

Un candidato puede pertenecer a varias listas y una lista puede tener muchos candidatos. La relación guarda el órgano y el orden en el que se encuentra ese candidato.

Atributos

Persona (ci, cc, nombre, apellido)

Candidato (generalización de Persona)

AgentePolicial (generalización de Persona) (comisaria)

MiembroMesa (generalización de Persona) (organismo, rol)

Mesa (id_mesa, numero_mesa, estado, cant_votos)

Circuito (id_circuito, numero_circuito, cant_votos, es_accesible, municipio)

Establecimiento (id_establecimiento, direccion, tipo, zona)

Departamento (id_departamento, nombre)

Elección (id_eleccion, fecha, tipo, descripcion)

Papeleta (id_papeleta, descripcion, color)

Voto (id_voto, estado, es_observado)

Lista (id_lista, numero_lista)

Senado (id_senado, nombre)

Partido (id_partido, nombre, presidente, vicepresidente, direccion_sede)

FormulaPresidencial (id_formula)

Validación del modelo

Para dar por finalizado el proceso de modelado conceptual, validamos el esquema teniendo en cuenta las siguientes cualidades:

Compleción: el modelo representa todos los conceptos principales del sistema electoral, incluyendo votantes, papeletas, listas, votos, circuitos, etc.

Corrección: se utilizan generalizaciones, relaciones N:M, atributos derivados, etc.

Minimalidad: no hay entidades ni relaciones redundantes, y las relaciones compuestas se justifican.

Expresividad: el modelo es natural y fácil de interpretar debido a su estructura.

Extensibilidad: el modelo está preparado para incorporar nuevos tipos de elecciones y mecanismos de votación sin tener que rediseñar la estructura central.

2

Modelo Lógico

Introducción

Como es sabido, el modelo lógico constituye la representación formal del modelo conceptual utilizando estructuras propias del modelo relacional. A partir del MER definido en la sección anterior, se derivaron las correspondientes tablas, claves primarias, claves foráneas y relaciones N:N, abordando también las generalizaciones y agregaciones.

Entidades simples

Persona (ci_persona, cc, nombre, apellido)

Mesa (id_mesa, numero_mesa, estado, cant_votos, id_circuito)
FK: id_circuito → Circuito.id_circuito

Circuito (id_circuito, numero_circuito, cant_votos, es_accesible, municipio, id_establecimiento, id_eleccion)

FK: id_establecimiento → Establecimiento.id_establecimiento
FK: id_eleccion → Eleccion.id_eleccion

Establecimiento (id_establecimiento, direccion, tipo, zona, id_departamento)

FK: id_departamento → Departamento.id_departamento

Departamento (id_departamento, nombre)

Elección (id_eleccion, fecha, tipo, descripcion)

Voto (id_voto, estado, es_observado, id_circuito, id_eleccion)

FK: id_circuito → Circuito.id_circuito
FK: id_eleccion → Eleccion.id_eleccion

Papeleta (id_papeleta, descripcion, color, id_eleccion)
FK: id_eleccion → Eleccion.id_eleccion

Lista (id_lista, numero_lista, id_eleccion, id_departamento)

FK: id_eleccion → Eleccion.id_eleccion
FK: id_departamento → Departamento.id_departamento

Senado (id_senado, nombre, id_partido)
FK: id_partido → Partido.id_partido

Partido (id_partido, nombre, presidente, vicepresidente, direccion_sede)

FormulaPresidencial (id_formula, ci_presidente, ci_vicepresidente, id_partido)

FK: ci_presidente → Candidato.ci
FK: ci_vicepresidente → Candidato.ci
FK: id_partido → Partido.id_partido

Generalizaciones

Candidato (ci)

PK y FK: ci → Persona.ci

MiembroMesa (ci, organismo, rol, id_mesa)

PK y FK: ci → Persona.ci
FK: id_mesa → Mesa.id_mesa

AgentePolicial (ci, comisaria, id_establecimiento)

PK y FK: ci → Persona.ci
FK: id_establecimiento → Establecimiento.id_establecimiento

Relaciones N:N, agregaciones y relaciones con atributos

Persona_Vota (ci, id_circuito, id_eleccion, fecha_hora, es_observado)

FK: ci → Persona.ci
FK: id_circuito → Circuito.id_circuito
FK: id_eleccion → Eleccion.id_eleccion

Voto_Lista (id_voto, id_lista)

FK: id_voto → Voto.id_voto
FK: id_lista → Lista.id_lista

Voto_Papeleta (id_voto, id_papeleta)

FK: id_voto → Voto.id_voto
FK: id_papeleta → Papeleta.id_papeleta

Candidato_Lista (ci, id_lista, organo, orden)

FK: ci → Candidato.ci
FK: id_lista → Lista.id_lista

Lista_Apoya (id_lista, id_partido, id_senado, id_formula)

FK: id_lista → Lista.id_lista
FK: id_partido → Partido.id_partido
FK: id_senado → Senado.id_senado
FK: id_formula → FormulaPresidencial.id_formula

Decisiones de modelado

- Se modelaron relaciones N:N mediante tablas intermedias con claves foráneas compuestas (ej. Voto_Lista, Candidato_Lista).
- Las generalizaciones se implementaron siguiendo el método de crear una tabla por subclase, compartiendo la PK como FK a Persona.
- Las agregaciones se resolvieron con claves compuestas (ej. Lista_Apoya), permitiendo mantener la estructura compleja del apoyo político.
- Se mantuvieron los identificadores naturales (como ci en Persona) como claves primarias.

Verificación de normalización

Para poder dar por finalizada esta etapa del modelado, nos pareció imprescindible comprobar que todas las tablas se encuentren en al menos la tercera forma normal (3FN):

1FN: Todos los atributos del modelo son atómicos y no repetitivos.

2FN: Todas las tablas que tienen claves compuestas (como Candidato_Lista) tienen sus atributos completamente dependientes de toda la clave.

3FN: No hay dependencias transitivas en ninguna tabla. Los atributos no clave dependen exclusivamente de la clave primaria.

A su vez:

- Las tablas como FormulaPresidencial evitan redundancia referenciando con claves foráneas a Persona.
- Las entidades como Establecimiento o Circuito no contienen atributos derivados ni redundantes.
- Las relaciones más complejas fueron normalizadas en tablas específicas y no se introdujo redundancia entre votos, listas y papeletas.

3

Modelo Físico

Sistema gestor de base de datos

La implementación del sistema se realizó utilizando el gestor de base de datos MySQL. Esta elección responde a los requerimientos establecidos en la letra del trabajo, y a su vez, por tratarse de un sistema que piensa ser ampliamente adoptado, con buenas herramientas de administración y una sintaxis que facilita la portabilidad y el despliegue en diferentes entornos, incluyendo Docker.

Características del modelo físico

A partir del modelo lógico previamente definido, se construyó un modelo físico que respeta la estructura relacional, pero considerando también aspectos de implementación como tipos de datos, integridad y rendimiento. En esta etapa se tradujeron las entidades y relaciones en sentencias SQL que crean las tablas correspondientes, definiendo claves primarias, claves foráneas, y restricciones según corresponda.

Se utilizaron tipos de datos simples y normalizados: INT para identificadores, VARCHAR para cadenas de texto de longitud acotada, BOOLEAN para atributos lógicos como es_observado o es_accesible, DATE para fechas y TEXT para descripciones extensas. Todas las claves primarias y foráneas fueron declaradas explícitamente para asegurar la buena referencialidad entre las distintas tablas.

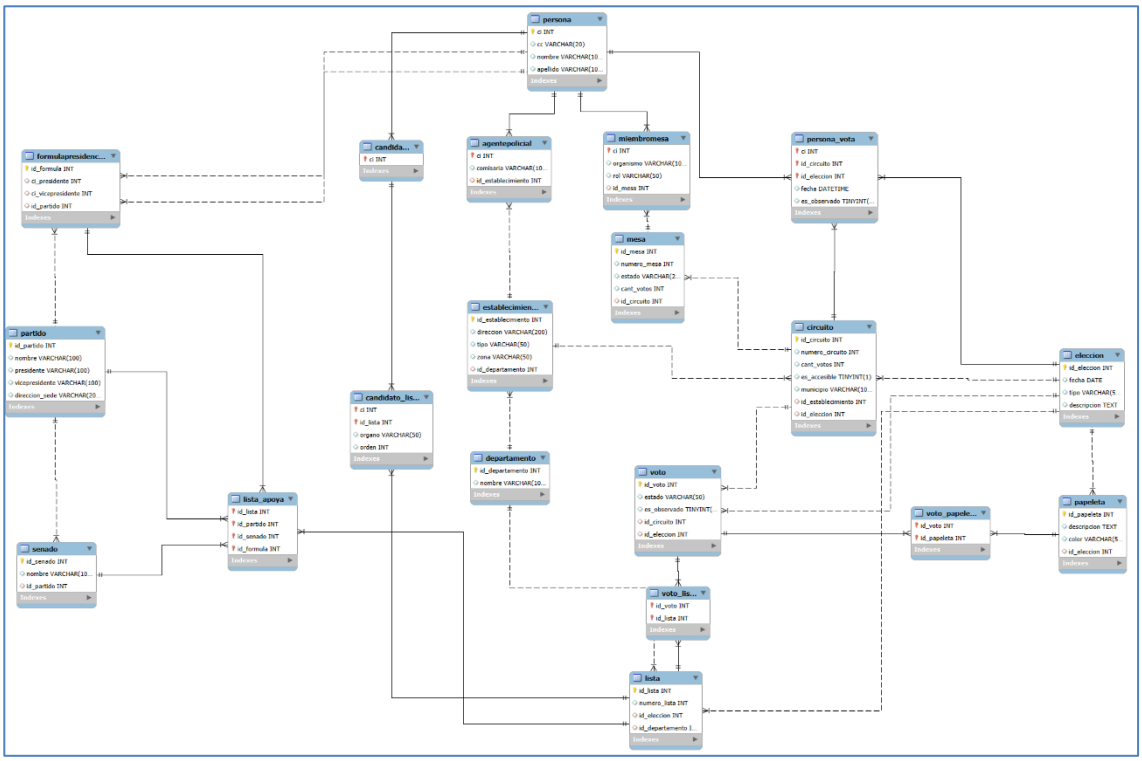
Las relaciones N:N, como se vió en el modelo lógico, fueron implementadas mediante tablas intermedias, como Voto_Lista o Candidato_Lista, que utilizan claves compuestas como identificador primario. Con esto se logra registrar información adicional relevante, como el órgano y orden del candidato en una lista. A su vez, las generalizaciones fueron resueltas usando una tabla por cada subclase (Candidato, MiembroMesa, AgentePolicial) que referencia la clave primaria de Persona.

La agregación más compleja del modelo, que vincula una lista con un partido, un senado y una fórmula presidencial (Lista_Apoya), se implementó como una relación con cuatro claves foráneas. Esta decisión permite mantener la lógica del modelo conceptual sin introducir redundancia ni violaciones en la normalización.

Script SQL y pruebas

Para la construcción de la base de datos se desarrolló un script SQL completo (ScriptSQL_Grupo6.sql.) Este script permite crear la base de datos desde cero, incluyendo todas las tablas, sus restricciones y relaciones. Fue validado mediante MySQL Workbench y también desde consola, garantizando que no existan errores ni conflictos.

Durante el proceso de prueba, se insertaron datos de ejemplo representativos para verificar el correcto funcionamiento de las restricciones. Se comprobó que el sistema rechaza inserciones inválidas (por ejemplo, claves foráneas inexistentes, duplicados en claves primarias) y acepta datos que respetan las reglas definidas. Esto valida tanto la consistencia del modelo como también la viabilidad de su uso.



4

Alternativas consideradas

Durante la etapa de modelado conceptual del sistema se discutieron varias formas de representar las principales entidades y relaciones que intervienen en una elección. Algunas de estas alternativas surgieron como intentos de simplificar o unificar estructuras, pero al analizar sus consecuencias a nivel de base de datos, se descartaron en favor de una solución más extensible y coherente con el sistema electoral de nuestro país.

Registro del voto dentro de la entidad Persona

Una de las primeras ideas fue registrar directamente en la entidad Persona el voto emitido, por ejemplo, mediante un atributo que indicara el ID de la lista o papeleta seleccionada. Esta solución buscaba simplificar el trazado del voto, evitando relaciones adicionales.

Sin embargo, esto también resultaba inviable por varias razones:

- Violaba el secreto del voto, al permitir vincular directamente a una persona con su elección.
- Impedía registrar votos en blanco, anulados u observados de forma clara.
- Generaba conflictos si una persona debía votar en distintos contextos (por ejemplo, como observado o como funcionario de mesa).

Esta solución fue descartada rápidamente por ir en contra de principios fundamentales del sistema electoral.

Unificación de listas y papeletas en una sola entidad

Otra alternativa considerada fue tratar a las listas y a las papeletas como instancias de una misma entidad general (Papeleta), y luego diferenciar mediante un atributo si se trataba de una opción partidaria o de plebiscito. Esta alternativa buscaba unificar la interfaz y simplificar algunas relaciones.

Sin embargo, esta opción presentaba múltiples problemas:

- Las listas están asociadas a candidatos, partidos y elecciones específicas, mientras que las papeletas solo corresponden a plebiscitos y su lógica es distinta (si/no).
- Esta unificación obligaba a manejar muchos atributos nulos según el tipo de papeleta, lo que afectaba la claridad del modelo.
- El registro de votos era más complejo, ya que listas y papeletas se computan y analizan de forma distinta en los resultados.

Por estas razones, se descartó la alternativa, y se optó por definir Lista y Papeleta como entidades separadas, cada una con su propia lógica y relaciones.

Uso de una única tabla para todos los votos

También se pensó utilizar una única entidad Voto para registrar tanto votos por lista como votos por papeleta, diferenciando luego con atributos adicionales. Aunque esto parecía más simple al principio, en la práctica traía varios inconvenientes:

- Mezclaba estructuras que tienen comportamientos distintos.
- Aumentaba la complejidad de las consultas para obtener resultados.
- Complicaba la validación de reglas específicas, como si una lista pertenece a un lema válido, o si una papeleta corresponde al circuito y elección.

Finalmente se decidió utilizar entidades separadas (Voto_Lista y Voto_Papeleta), cada una orientada a su tipo de elección, lo que resultó en un modelo más claro y extensible.

Solución implementada

La solución mostrada en secciones anteriores se basó en un enfoque modular y específico, en el que cada tipo de entidad refleja con su rol en el sistema electoral. Se mantuvieron Lista y Papeleta como entidades independientes, y se diseñaron estructuras separadas para registrar votos según el tipo de elección.

Además, se cuidó en todo momento preservar el voto secreto, evitando cualquier vínculo entre el votante y el contenido de su voto, y garantizando un diseño normalizado y extensible.

Esta elección permitió cumplir con los requisitos planteados en la consigna, y facilitó la implementación lógica posterior tanto en la base de datos como en el desarollo de la aplicación.

5

Sobre la aplicación

Idea general de la aplicación

La aplicación desarrollada acompaña el modelo de base de datos propuesto y tiene como objetivo principal digitalizar el proceso de votación. El sistema permite a los ciudadanos emitir su voto de forma sencilla, a las autoridades de mesa administrar el circuito y, una vez finalizada la jornada electoral, acceder a los resultados.

El desarrollo de la aplicación contempla tanto la interfaz de votación para los ciudadanos como las funcionalidades de control para los miembros de mesa, respetando los principios de voto secreto, unicidad del voto por persona, cierre definitivo del circuito, entre otras condiciones.

Tecnologías utilizadas

La aplicación se dividió en dos componentes principales: un backend desarrollado en Node.js y un frontend en React. Esta separación implica un modelo cliente-servidor clásico, donde la comunicación entre frontend y backend se realizó mediante una API REST.

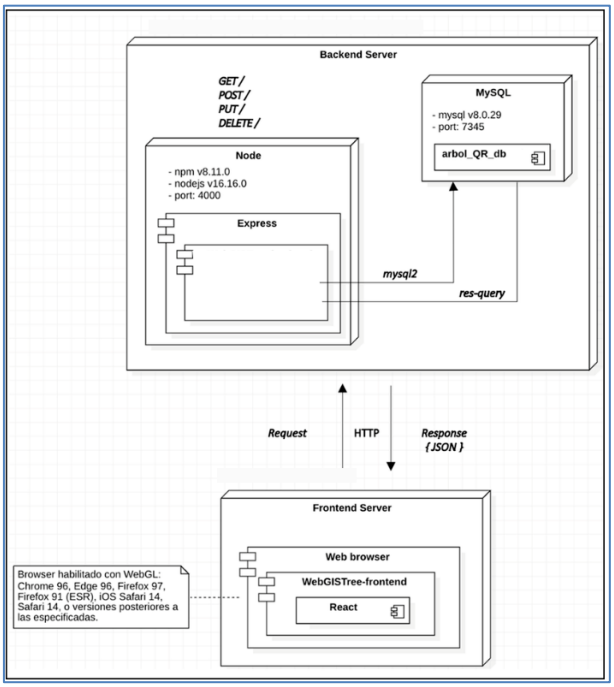
Arquitectura

En el backend, se aplicó una arquitectura inspirada en el patrón MVC, donde el Modelo y el Controlador se implementan en Node.js, mientras que la Vista es gestionada por el frontend. Esta separación permite una arquitectura desacoplada y modular.

- **models/:** encargada del acceso directo a la base de datos.
- **controllers/**: contiene los controladores que reciben las peticiones HTTP, extraen parámetros y responden en formato JSON.
- **services/**: implementa la lógica de negocio del sistema. Aquí se procesaron operaciones como "cerrar circuito", "registrar voto", "calcular resultados", entre otras.
- **middlewares/**: incluye funcionalidades como validaciones de entrada, manejo de errores, y eventualmente autenticación.
- **routes/**: define los endpoints de la API y los vincula con los controladores correspondientes.

Conexión con la base de datos

La conexión con la base de datos se realizó directamente desde el backend mediante el paquete mysql2. Dado que el uso de ORM no está permitido, se trabajó mediante consultas SQL escritas manualmente y organizadas dentro de los módulos del sistema (models).



Flujo de uso previsto para la aplicación

La aplicación estará desplegada en una computadora disponible en cada circuito de votación. El flujo previsto para cada votante es el siguiente:

1. **Pantalla de inicio:** el votante se encuentra con un formulario donde debe ingresar su número de credencial cívica y una contraseña personal previamente otorgada por la Corte Electoral. Esta información será validada por el sistema para verificar su identidad y asegurar que aún no haya emitido su voto. En la parte inferior derecha de esta pantalla se encontrará un botón destinado exclusivamente al presidente de mesa.

2. **Pantalla de votación:** Tras autenticarse correctamente, el votante debe indicar el número del circuito en el cual desea votar. Esta información se utiliza para filtrar las listas y papeletas habilitadas en ese contexto electoral.

   El sistema validará que el circuito ingresado coincida con el que se le asigno por padrón. En caso de que no coincida, se notificará al presidente de mesa, quien evaluará si corresponde permitirle votar como observado. Este procedimiento garantiza que el sistema contemple situaciones de desplazamiento o error, sin comprometer el proceso electoral.

3. **Confirmación del voto:** tras realizar la selección, el votante presiona un botón de "VOTAR". El sistema registra la decisión en la base de datos y muestra una pantalla de agradecimiento que confirma la emisión del voto.

4. **Reinicio automático:** luego de unos segundos, el sistema vuelve automáticamente a la pantalla inicial para que el siguiente votante pueda comenzar el proceso.

Este flujo se repite de forma cíclica para cada votante, asegurando confidencialidad y simplicidad a la hora de votar. La interfaz se construyó con React, pero no se tratará de una SPA en el sentido tradicional, ya que no hay navegación libre ni múltiples vistas internas. En cambio, se trata de una aplicación con flujo de interacción único, orientada específicamente a un proceso controlado de votación.

La comunicación entre la interfaz y el backend se realiza mediante peticiones HTTP (fetch o axios), con solicitudes que permiten registrar votos, consultar las listas y papeletas disponibles para el circuito y verificar el estado del votante.

Funciones del presidente de mesa y flujo de cierre

Además del flujo destinado a los votantes, la aplicación cuenta con funcionalidades específicas para el presidente de mesa. Desde la pantalla inicial, mediante un botón ubicado en la esquina inferior derecha, se puede acceder a una sección protegida un PIN previamente configurado.

Una vez autenticado, el presidente de mesa accederá a un panel que le permitirá:

1. **Cerrar el circuito**: esta acción cambia el estado de la mesa a "cerrado" en la base de datos. A partir de ese momento, el sistema ya no permitirá emitir votos en ese circuito.
2. **Consultar resultados**: disponible únicamente luego de cerrado el circuito. Se mostrarán los totales por lista, votos en blanco, votos anulados, resultados por papeleta (si aplica) y estadísticas locales.

Estas acciones estarán protegidas para evitar accesos indebidos. Además, el sistema validará que:

- No se puedan emitir votos luego del cierre.

- No se pueda volver a abrir una mesa cerrada.
- No se puedan consultar resultados anticipadamente.

Se garantiza que todos los ciudadanos, incluyendo los miembros de mesa y agentes de policía, puedan votar mientras la mesa esté abierta.

Restricciones lógicas y validaciones

El backend es el responasble de validar todas las reglas de negocio del sistema. Entre otras:

- Validar que una persona vote una única vez por elección.
- Verificar que la mesa esté habilitada antes de registrar votos.
- Validar que los votos sean coherentes con el circuito y la elección.
- Permitir o denegar votos observados según el caso.
- Validar que el cierre de circuito sólo pueda hacerse una vez cumplidas ciertas condiciones.

Estas reglas se implementan en la capa de servicios y son respaldadas por restricciones a nivel de base de datos cuando es posible.

Garantías del voto secreto

Uno de los principios fundamentales del sistema electoral de nuestro país es la preservación del voto secreto. El diseño de la base de datos y la lógica de aplicación contemplan este requerimiento, evitando cualquier asociación entre el contenido del voto y la identidad del votante.

En el modelo físico, los votos emitidos se registran en las tablas Voto_Lista y Voto_Papeleta, donde se almacena información sobre las listas seleccionadas y l as

papeletas correspondientes, junto con el circuito, la elección y la mesa. Sin embargo, no se guarda ningún identificador personal del votante en dichas tablas.

La información que identifica a las personas se utiliza únicamente en la etapa de verificación previa al voto, para asegurar que no hayan votado más de una vez en la misma elección y circuito. Una vez validado el derecho a votar, no se asocia de ninguna forma al contenido de su voto.

Además, el sistema impide ver los resultados hasta que la mesa haya sido cerrada, lo cual también contribuye a proteger el anonimato del voto a lo largo del día de la elección.

Este enfoque asegura el cumplimiento del voto secreto y genera confianza en el sistema que se propuso.

Perspectivas futuras

Si bien este proyecto está orientado a ser un MVP, en futuras etapas del proyecto, se podría incluir mecanismos más fuertes de autenticación y autorización, así como la integración con librerías de interfaz para mejorar la experiencia del usuario. También se podrá automatizar la generación de datos de prueba y realizar pruebas de carga para validar el rendimiento en condiciones reales.

En conclusión, consideramos que la aplicación se apoya en una arquitectura moderna, basada en tecnologías abiertas, y sigue patrones de diseño consolidados como MVC y REST. Esto asegura que el desarrollo posterior sea consistente con el modelo diseñado y ofrezca una base sólida y extensible.

Conclusión

El desarrollo de este trabajo permitió aplicar los conocimientos adquiridos en la materia, abordando todas las etapas del ciclo de vida de una base de datos: análisis de requisitos, modelado conceptual y lógico, normalización, y diseño físico.

Se logró construir un modelo de datos coherente, robusto y en tercera forma normal (3FN), adaptado a un caso realista como lo es el de una elección nacional. Se implementó una solución tecnológica moderna basada en una arquitectura cliente-servidor, cumpliendo con principios como el desacoplamiento de capas, la organización modular y el respeto al voto secreto.

Entre los principales desafíos se encontraron el tratamiento de la agregación entre elecciones y circuitos, la correcta implementación de votos observados y anulados, y el diseño del flujo de interacción para votantes y presidentes de mesa.

En síntesis, el trabajo permitió integrar teoría y práctica en un proyecto real, aplicando metodologías de modelado, diseño y desarrollo acordes a los estándares actuales del desarrollo.

Referencias

- Universidad Católica del Uruguay. *BD2 Normalización*
- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- Mozilla Developer Network (MDN). *MVC – Modelo-Vista-Controlador* https://developer.mozilla.org/en-US/docs/Glossary/MVC

- GeeksForGeeks. *Client Server Architecture* https://www.geeksforgeeks.org/system-design/client-server-model/
- Corte Electoral del Uruguay. *Sitio oficial*. https://www.corteelectoral.gub.uy/
- El Observador. (2024). *Elecciones 2024 Uruguay: ¿Cuál es la diferencia entre voto blanco y anulado y cómo afecta la elección del presidente?*
- Wikipedia. (2024). *Elecciones generales de Uruguay de 2024.* https://es.wikipedia.org/wiki/Elecciones_generales_de_Uruguay_de_2024
- Ceibal. (s.f.). *¿Plebiscito y referéndum son sinónimos? Recursos educativos abiertos.* https://rea.ceibal.edu.uy/elp/reformas-constitucionales/plebiscito_y_referndum_son_sinnimos.htmlWikipedia. (s.f.).