PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Proceso electoral presidencial chileno
Escuela de Ingeniería Informática

Mateo cueto
Henry Huerta
Rodrigo Escudero

Asignatura: **Estructura de datos**
Fecha de entrega: **26 de noviembre de 2025**

Índice

Índice de figuras

Introducción

1. Contexto del problema

La administración del proceso electoral presidencial en Chile exige un sistema capaz de gestionar información de forma dinámica, segura y verificable. La complejidad inherente a este proceso, que involucra la inscripción de candidatos, la habilitación de votantes, la operación de mesas electorales y el escrutinio final de los votos, requiere una estructura ordenada que permita manejar datos de manera coherente y oportuna. Cada elección presenta características particulares según su contexto político, el número de participantes, el territorio involucrado y las condiciones institucionales que rigen su ejecución. En este escenario, el sistema desarrollado ofrece una herramienta que organiza y estructura el flujo de información dentro de una elección presidencial, permitiendo simular sus principales etapas de forma controlada.

Durante el desarrollo del proceso electoral, resulta indispensable mantener un registro claro de los candidatos, votantes y mesas que participan en la elección, así como del resultado final obtenido tras el escrutinio. Para ello, la interacción sistemática entre los distintos componentes del sistema debe permitir registrar datos, modificarlos cuando sea necesario y consolidarlos en reportes verificables que representen fielmente el estado de la elección. Dado que cada estructura almacena información con un propósito específico, su correcta implementación y relación con las demás resulta crucial para garantizar la precisión de los cálculos realizados y la integridad del proceso electoral simulado.

La representación computacional del proceso electoral requiere un diseño estructural que permita diferenciar las funciones del Servicio Electoral (Servel), encargado del registro de candidatos y de la creación de elecciones, de aquellas propias del Tribunal Calificador de Elecciones (Tricel), que actúa en la fase final del proceso validando los resultados y determinando la existencia de segunda vuelta. De igual forma, es necesario modelar la interacción con las mesas de votación, que funcionan como unidades operativas donde se registran votantes y se contabilizan los votos emitidos.

En términos generales, el sistema implementado busca resolver la necesidad de organizar la información electoral mediante estructuras de datos adecuadas para cada entidad involucrada, con el fin de permitir un flujo coherente entre la creación de la elección, la participación de los votantes y la verificación final de los resultados. Para ello se utilizan estructuras dinámicas y estáticas que permiten administrar candidatos, votantes, mesas y resultados de manera eficiente, permitiendo al usuario simular las principales operaciones que se llevan a cabo durante una elección real.

1.1. Objetivo general

Construir un sistema que permita gestionar candidatos, votantes, mesas y resultados de una elección presidencial, proporcionando mecanismos para registrar, consultar y procesar información electoral de manera ordenada y verificable.

1.2. Objetivos específicos

- Diseñar estructuras de datos que permitan almacenar y organizar candidatos, votantes, mesas y resultados de manera eficiente dentro del sistema.
- Implementar mecanismos que aseguren el registro, consulta y actualización correcta de la información electoral durante cada etapa del proceso.
- Desarrollar funciones que permitan simular el flujo real de una elección presidencial, integrando las acciones del Servel, las Mesas de sufragio y el Tricel.
- Garantizar que los datos registrados en una etapa del proceso se reflejen coherentemente en las siguientes, permitiendo que el sistema determine resultados válidos y verificados.

2. Descripción del problema

El proceso electoral presidencial en Chile requiere administrar información compleja y diversa, correspondiente a candidatos, votantes, mesas de sufragio y resultados oficiales. La correcta gestión de estos datos es fundamental para asegurar la transparencia del proceso, su trazabilidad y la validez de las decisiones que se toman en cada etapa. Sin embargo, cuando dicha información no está digitalizada o se administra sin un sistema estructurado, su manejo puede volverse lento, impreciso e incluso propenso a errores que comprometen la integridad del proceso electoral.

El registro manual o disperso de los datos dificulta realizar consultas oportunas sobre la participación electoral, la verificación de candidatos habilitados, la validez de los votos emitidos y la detección de inconsistencias entre mesas. Esta situación no solo afecta la eficiencia operativa, sino también la capacidad de los organismos electorales para garantizar un proceso ordenado y verificable para todos los actores involucrados. Los problemas derivados de una mala gestión de la información pueden observarse de distintas formas.

2.1. Problemas derivados

- La falta de respaldo digital o estructurado de los datos aumenta el riesgo de pérdida, duplicación o corrupción de información relevante del proceso electoral.
- La dispersión de los registros dificulta el seguimiento de candidatos, la verificación de sus requisitos legales y la correcta incorporación a cada elección.
- La ausencia de mecanismos uniformes para registrar votantes genera inconsistencias en las mesas, afectando el conteo final y el cálculo de participación.
- La falta de un sistema que consolide automáticamente los votos emitidos en distintas mesas retrasa la generación de resultados preliminares y oficiales.
- No contar con reglas automatizadas para determinar si existe ganador en primera vuelta o si procede una segunda vuelta puede generar errores en la validación final.
- La administración manual de la información obliga a destinar tiempo a tareas operativas básicas (ordenar, clasificar, validar), reduciendo la capacidad de análisis y fiscalización.

2.2. Necesidades de gestión y seguridad del proceso electoral

El proceso electoral requiere un sistema que permita organizar de forma clara los datos relacionados con cada una de sus etapas. Cada entidad —candidatos, votantes, mesas y resultados— debe mantener una identidad propia dentro de la estructura general del proceso, siguiendo una jerarquía coherente y un flujo que evite contradicciones entre la información.

Asimismo, la correcta administración del proceso demanda mecanismos que aseguren:

- La verificación automática de requisitos de los candidatos antes de ser incorporados a una elección.
- El registro fiable de votantes habilitados y su relación con una mesa específica.
- La capacidad de recorrer y consolidar datos provenientes de múltiples mesas de votación.
- La generación de resultados claros, verificables y basados en cálculos consistentes.
- La protección de la integridad de los datos para evitar manipulaciones no autorizadas.

Estas necesidades justifican la importancia de contar con un sistema estructurado que permita almacenar, mantener y consultar la información electoral de forma ordenada, brindando soporte a todas las etapas del proceso y evitando las inconsistencias propias de un manejo manual o fragmentado de los datos.

3. Análisis

Para comprender adecuadamente el funcionamiento del sistema electoral desarrollado, es necesario analizar tanto las entidades fundamentales del proceso —candidatos, votantes, mesas y resultados— como las funciones que permiten manipularlas dentro de la implementación en C. Este análisis integra la visión conceptual del proceso electoral con la lógica interna del programa, de modo que las estructuras y funciones puedan entenderse como partes de un mismo flujo coherente.

El propósito de esta sección es describir los subconjuntos de datos que componen el sistema, su relación dentro del proceso electoral y las funciones encargadas de operar sobre ellos. Esta visión mixta permite comprender no solo qué representa cada componente, sino también cómo se gestiona y transforma la información dentro del sistema.

3.1. Candidatos

Los candidatos representan a los participantes que compiten en la elección presidencial. Cada uno posee características estrictamente reguladas por el Servel, tales como edad mínima, nacionalidad y, en el caso de candidaturas independientes, la cantidad de firmas requeridas. Esta información se almacena en un *pool* estático de candidatos, donde cada estructura contiene datos personalizados y un indicador que señala si cumple o no los requisitos legales.

Funciones asociadas

La función `Servel_inicializarCandidato` crea dinámicamente un nuevo candidato y su estructura `Persona`. Posteriormente, `Servel_cargarDatosCandidato` solicita y valida la información ingresada por el usuario, asegurando que los datos sean correctos antes de ser almacenados.

La validación formal se realiza mediante `validarDatosCanditado`, que comprueba edad, nacionalidad y firmas de apoyo según el tipo de candidatura. Esta función garantiza que solo candidatos habilitados sean incorporados a elecciones.

3.2. Votantes

Los votantes representan a las personas habilitadas para sufragar. Su información incluye datos personales, estado de habilitación y si ya han emitido su voto. Cada votante pertenece a una mesa específica, lo que permite organizar el flujo de participación.

En el sistema, los votantes se almacenan mediante listas doblemente enlazadas, lo cual permite recorrerlos en ambos sentidos y realizar inserciones eficientes.

Funciones asociadas

La función `RegistrarVotanteEnMesa` implementa un procedimiento completo para añadir un nuevo votante. Incluye:

1. Verificación del ID de elección.
2. Búsqueda de la mesa en el árbol binario de búsqueda.
3. Creación dinámica del votante y su `Persona`.
4. Validación de los datos ingresados.
5. Inserción en la lista doblemente enlazada manteniendo punteros `ant` y `sig`.

Para consultas, `ListarVotantesDeMesa` recorre la lista desde su primer nodo hasta el último, mostrando la información relevante para control ciudadano y para la operación del Tricel.

3.3. Mesas electorales

Las mesas corresponden a las unidades operativas donde se registra la participación electoral. Cada mesa almacena votantes y el conteo de votos válidos, blancos y nulos. Internamente, las mesas están organizadas mediante un árbol binario de búsqueda (ABB), lo que permite buscar mesas por su identificador de manera eficiente.

Funciones asociadas

La función `buscarLaMesaConId` sigue estrictamente las reglas de un ABB:

- Si el ID buscado es menor, se recorre el subárbol izquierdo.
- Si es mayor, se recorre el subárbol derecho.
- Si coincide, se retorna el nodo.

Este diseño reduce el tiempo de acceso y permite realizar operaciones rápidas incluso con un número elevado de mesas.

3.4. Resultados

Los resultados electorales representan la última etapa del proceso. Contienen el total de votos emitidos, participación, votos por candidato y la determinación de ganador o

segunda vuelta. El Tricel consolida esta información recorriendo todas las mesas de la elección.

Los resultados se almacenan en una lista circular enlazada, lo que permite mantener un registro permanente sin necesidad de reiniciar el recorrido.

Funciones asociadas

Todas las funciones de conteo emplean un patrón recursivo *in-order* del ABB:

- `ContarMesas`
- `ContarVotosEmitidos`
- `ContarVotosBlancos`
- `ContarVotosNulos`
- `ContarXcandidato`

La función principal `recopilarResultados` realiza:

1. Conteo de mesas y votos.
2. Cálculo de participación total.
3. Cálculo de porcentajes por candidato.

Luego, `ValidarSegundaVuelta` determina si existe ganador con mayoría absoluta o si deben competir los dos más votados en una segunda vuelta.

Si una nueva elección es necesaria, `paraSegundaVuelta` crea dinámicamente un nuevo proceso electoral con solo dos candidatos.

Finalmente, `agregarAtricel` inserta los resultados en la lista circular, permitiendo conservar el historial completo. La función `proclamarUnGanador` recorre dicha lista hasta el último nodo para determinar y mostrar el ganador final.

3.5. Implicancias para la estructuración del sistema

La integración entre las entidades del proceso electoral y las funciones que operan sobre ellas permite establecer un diseño robusto y representativo del proceso real. Las estructuras de datos seleccionadas —arreglos estáticos, listas simples, árboles binarios, listas dobles y listas circulares— proporcionan mecanismos eficientes de búsqueda, inserción, recorrido y almacenamiento.

Comprender estas relaciones es esencial para avanzar hacia un diseño que no solo sea funcional desde el punto de vista computacional, sino también coherente con la lógica institucional del proceso electoral chileno.

4. Diseño del sistema

A continuación se presenta el diseño estructural del sistema electoral, centrado en la definición de las estructuras de datos que permiten modelar cada uno de los componentes involucrados en el proceso. Al igual que en el capítulo anterior, se organiza el análisis siguiendo niveles de anidación, de manera que sea posible observar cómo estructuras de alto nivel contienen elementos de niveles inferiores, y cómo estos se relacionan entre sí mediante funciones específicas y flujos de datos.

Este enfoque facilita comprender el funcionamiento interno del sistema, así como la jerarquía entre Servel, Mesas, Votantes y los Resultados administrados por el Tricel.

4.1. Estructura de nivel cero: Sistema Electoral

El nivel cero corresponde a la estructura raíz de todo el sistema: `SistemaElectoral`. Esta estructura representa el equivalente al Ministerio Público en el caso judicial del ejemplo anterior, ya que desde ella se accede a todos los componentes principales del proceso electoral.

- Contiene un puntero a `Servel`, encargado de administrar candidatos y elecciones.
- Contiene un puntero a `Tricel`, encargado de consolidar resultados.
- Actúa como contenedor general de todos los datos y relaciones del sistema.

Descripción gráfica equivalente: Un recuadro principal llamado `SistemaElectoral` del cual salen dos flechas: una hacia `Servel` y otra hacia `Tricel`, cada uno representado como módulos independientes.

4.2. Estructura de primer nivel: Servel

En este nivel encontramos las entidades administradas directamente por el Servel. La estructura `Servel` posee tres elementos fundamentales:

- Un arreglo estático que constituye el *pool* de candidatos válidos.
- Una lista simple enlazada de elecciones creadas en el sistema.
- Un contador del total de votantes registrados en el sistema.

Los candidatos permanecen en memoria durante toda la ejecución, mientras que las elecciones pueden ser agregadas o eliminadas dinámicamente.

Campos relevantes

- `Candidato candidatos[MAX_CANDIDATOS];`
- `NodoEleccion *headElecciones;`

- `int totalVotantesRegistrados;`

Descripción gráfica equivalente: Un bloque `Servel` que contiene: (1) un arreglo horizontal de candidatos, (2) una lista simple vertical de elecciones, (3) un contador numérico.

4.3. Estructura de segundo nivel: Elecciones y Tricel

Cada nodo de la lista simple del Servel representa una elección. Las elecciones constituyen estructuras compuestas que contienen:

- Identificador único de elección.
- Número de ronda (primera o segunda).
- Estado electoral (`ABIERTA`, `CERRADA`, `PROCLAMADA`).
- Un árbol binario de búsqueda (ABB) de mesas.
- Un puntero a un nodo de la lista circular de resultados.

El Tricel se encuentra también en este nivel, ya que opera sobre los resultados asociados a las elecciones. Su estructura principal es una lista circular enlazada que almacena uno o varios resultados históricos.

Descripción gráfica equivalente: Una figura donde `Eleccion` cuenta con: un bloque de campos simples, una flecha a un árbol binario (mesas), una flecha a un nodo circular (resultados).

4.4. Estructura de tercer nivel: Mesas, Votantes y Resultados

Este nivel desglosa las estructuras contenidas dentro de cada elección. Aquí se encuentran las unidades operativas del proceso: mesas de votación, votantes asociados a cada mesa, y los datos numéricos consolidados del resultado electoral.

Árbol binario de Mesas

Cada elección contiene un ABB de mesas, organizado por identificador de mesa. Cada nodo almacena:

- ID de la mesa.
- Comuna y dirección.
- Lista doblemente enlazada de votantes.
- Contadores de votos: válidos, blancos y nulos.

Descripción gráfica equivalente: Un árbol binario donde cada nodo tiene: campos simples, punteros `left`/`right`, y una flecha hacia una lista doble.

Lista doble de votantes

Cada mesa posee una lista doblemente enlazada que almacena todos los votantes registrados:

- Datos personales del votante.
- Estado de habilitación.
- Indicador de si votó o no.

Descripción gráfica equivalente: Una lista doble clásica: nodos conectados con flechas bidireccionales.

Lista circular de resultados

Los resultados electorales se almacenan en una lista circular enlazada. Cada nodo contiene:

- Votos totales, válidos, blancos y nulos.
- Porcentajes para cada candidato.
- Identificación del ganador.

Descripción gráfica equivalente: Una secuencia circular de nodos conectados en un ciclo cerrado.

4.5. Estructuras adicionales relevantes

El diseño incorpora además:

- Arreglo compacto almacenando candidatos presentes en cada elección.
- Contadores acumulativos usados por Tricel durante el cálculo final.
- Variables auxiliares para registrar participaciones, totales y validaciones legales.

4.6. Diagramas de flujo

Los diagramas de flujo son esenciales para comprender la lógica de las funciones implementadas en las estructuras anteriores. A continuación se describen de forma textual las versiones adaptadas para este sistema electoral.

4.6.1. Agregar votante a una mesa

Flujo equivalente:

1. Se ingresa ID de elección.
2. Se busca elección en la lista simple.
3. Si no existe, se finaliza.
4. Se ingresa ID de mesa.
5. Se busca mesa en el ABB.
6. Si no existe, se finaliza.
7. Se crea votante dinámicamente.
8. Se validan campos ingresados.
9. Se inserta al final de la lista doble (ajuste de `ant/sig`).

4.6.2. Insertar mesa en ABB

1. Inicia en la raíz del ABB.
2. Si es NULL, se inserta la nueva mesa.
3. Si el ID es menor, se recurre al subárbol izquierdo.
4. Si el ID es mayor, se recurre al subárbol derecho.
5. Se inserta el nodo en la posición correcta.

4.6.3. Generar resultados y verificar segunda vuelta

1. Recorrer ABB para contar votos válidos, nulos y blancos.
2. Sumar participación total.
3. Calcular porcentaje por candidato.
4. Verificar si algún candidato supera el 50 %.
5. Si no supera, seleccionar los dos más votados.
6. Crear elección de segunda vuelta.

5. Diseño del sistema

A continuación se presenta el diseño estructural del sistema electoral, centrado en la definición de las estructuras de datos que permiten modelar cada uno de los componentes involucrados en el proceso. Al igual que en el capítulo anterior, se organiza el análisis siguiendo niveles de anidación, de manera que sea posible observar cómo estructuras de alto nivel contienen elementos de niveles inferiores, y cómo estos se relacionan entre sí mediante funciones específicas y flujos de datos.

Este enfoque facilita comprender el funcionamiento interno del sistema, así como la jerarquía entre Servel, Mesas, Votantes y los Resultados administrados por el Tricel.

5.1. Estructura de nivel cero: Sistema Electoral

El nivel cero corresponde a la estructura raíz de todo el sistema: `SistemaElectoral`. Esta estructura representa el equivalente al Ministerio Público en el caso judicial del ejemplo anterior, ya que desde ella se accede a todos los componentes principales del proceso electoral.

- Contiene un puntero a `Servel`, encargado de administrar candidatos y elecciones.
- Contiene un puntero a `Tricel`, encargado de consolidar resultados.
- Actúa como contenedor general de todos los datos y relaciones del sistema.

Descripción gráfica equivalente: Un recuadro principal llamado `SistemaElectoral` del cual salen dos flechas: una hacia `Servel` y otra hacia `Tricel`, cada uno representado como módulos independientes.

5.2. Estructura de primer nivel: Servel

En este nivel encontramos las entidades administradas directamente por el Servel. La estructura `Servel` posee tres elementos fundamentales:

- Un arreglo estático que constituye el *pool* de candidatos válidos.
- Una lista simple enlazada de elecciones creadas en el sistema.
- Un contador del total de votantes registrados en el sistema.

Los candidatos permanecen en memoria durante toda la ejecución, mientras que las elecciones pueden ser agregadas o eliminadas dinámicamente.

Campos relevantes

- `Candidato candidatos[MAX_CANDIDATOS];`
- `NodoEleccion *headElecciones;`

- `int totalVotantesRegistrados;`

Descripción gráfica equivalente: Un bloque `Servel` que contiene: (1) un arreglo horizontal de candidatos, (2) una lista simple vertical de elecciones, (3) un contador numérico.

5.3. Estructura de segundo nivel: Elecciones y Tricel

Cada nodo de la lista simple del Servel representa una elección. Las elecciones constituyen estructuras compuestas que contienen:

- Identificador único de elección.
- Número de ronda (primera o segunda).
- Estado electoral (`ABIERTA`, `CERRADA`, `PROCLAMADA`).
- Un árbol binario de búsqueda (ABB) de mesas.
- Un puntero a un nodo de la lista circular de resultados.

El Tricel se encuentra también en este nivel, ya que opera sobre los resultados asociados a las elecciones. Su estructura principal es una lista circular enlazada que almacena uno o varios resultados históricos.

Descripción gráfica equivalente: Una figura donde `Eleccion` cuenta con: un bloque de campos simples, una flecha a un árbol binario (mesas), una flecha a un nodo circular (resultados).

5.4. Estructura de tercer nivel: Mesas, Votantes y Resultados

Este nivel desglosa las estructuras contenidas dentro de cada elección. Aquí se encuentran las unidades operativas del proceso: mesas de votación, votantes asociados a cada mesa, y los datos numéricos consolidados del resultado electoral.

Árbol binario de Mesas

Cada elección contiene un ABB de mesas, organizado por identificador de mesa. Cada nodo almacena:

- ID de la mesa.
- Comuna y dirección.
- Lista doblemente enlazada de votantes.
- Contadores de votos: válidos, blancos y nulos.

Descripción gráfica equivalente: Un árbol binario donde cada nodo tiene: campos simples, punteros `left`/`right`, y una flecha hacia una lista doble.

Lista doble de votantes

Cada mesa posee una lista doblemente enlazada que almacena todos los votantes registrados:

- Datos personales del votante.
- Estado de habilitación.
- Indicador de si votó o no.

Descripción gráfica equivalente: Una lista doble clásica: nodos conectados con flechas bidireccionales.

Lista circular de resultados

Los resultados electorales se almacenan en una lista circular enlazada. Cada nodo contiene:

- Votos totales, válidos, blancos y nulos.
- Porcentajes para cada candidato.
- Identificación del ganador.

Descripción gráfica equivalente: Una secuencia circular de nodos conectados en un ciclo cerrado.

5.5. Estructuras adicionales relevantes

El diseño incorpora además:

- Arreglo compacto almacenando candidatos presentes en cada elección.
- Contadores acumulativos usados por Tricel durante el cálculo final.
- Variables auxiliares para registrar participaciones, totales y validaciones legales.

6. Diagramas de flujo

6.1. Agregar y validar un candidato

El procedimiento de registro de un candidato corresponde a uno de los flujos más importantes del módulo Servel, ya que determina qué postulantes cumplen con los requisitos legales para participar en la elección. Este proceso involucra lectura de datos, validación estricta, asignación dinámica de memoria y registro en el *pool* estático de candidatos.

Flujo equivalente:

1. Verificar si aún existe espacio disponible en el arreglo estático de candidatos.
2. Intentar reservar memoria dinámica para la estructura `Candidato` y su `Persona`.
3. Si la reserva falla, finalizar el proceso mostrando un mensaje de error.
4. Solicitar al usuario los datos base: edad, nacionalidad, RUT, nombre, partido y tipo de candidatura.
5. Si el candidato es independiente, leer cantidad de firmas de apoyo; de lo contrario, asignar 0.
6. Validar que los datos cumplas con los requisitos mínimos (edad, nacionalidad, RUT correcto, etc.).
7. Si algún dato es inválido, descartar la creación y finalizar.
8. Aplicar las reglas electorales específicas, verificando requisitos según tipo de candidatura.
9. Si las reglas no se cumplen, marcar candidato como inválido y finalizar.
10. Si cumple los requisitos, marcar candidato como válido.
11. Guardar el candidato en el *pool* estático del Servel.
12. Incrementar el contador de candidatos válidamente registrados.

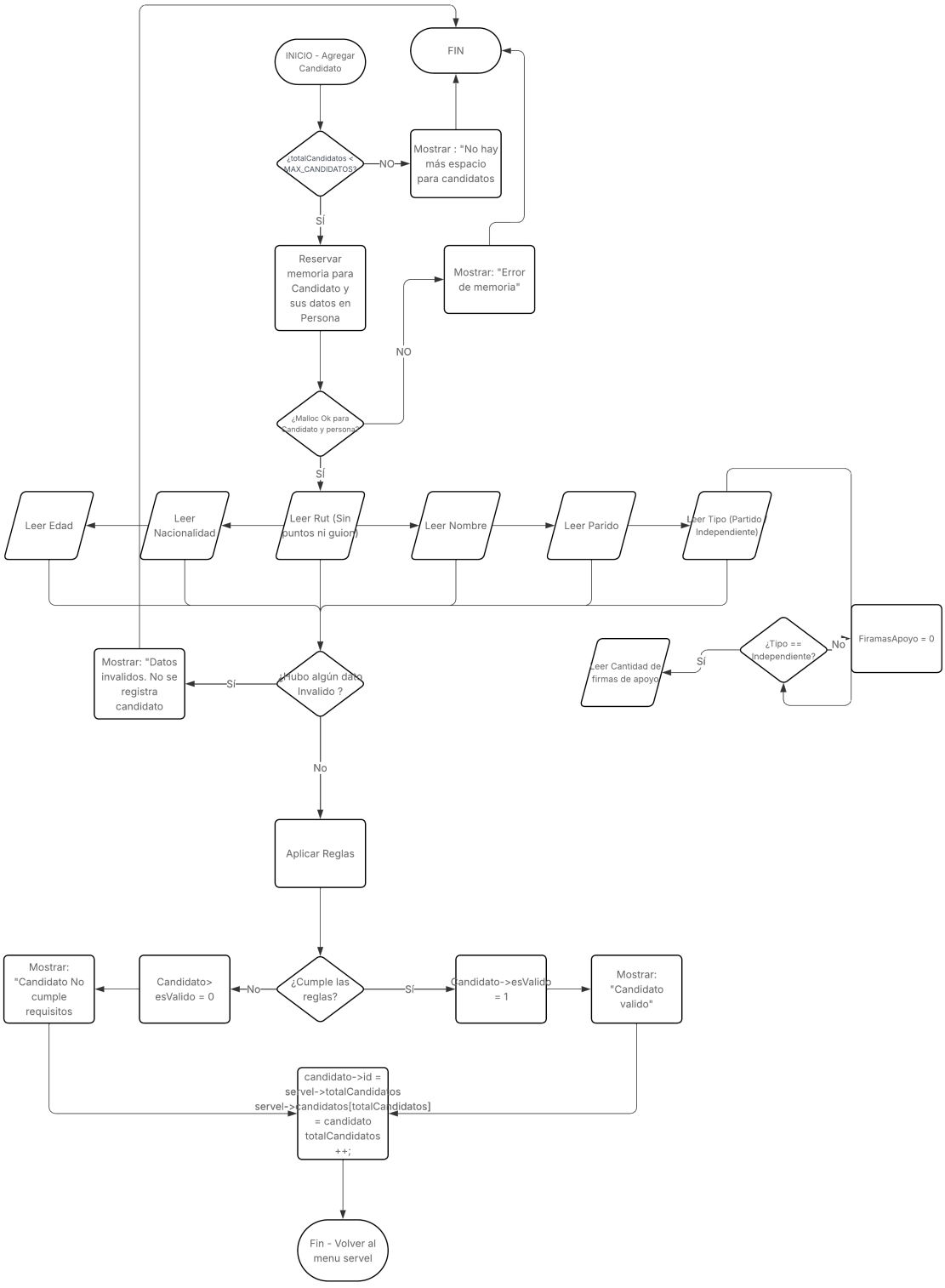

Figura 6.1: Diagrama de flujo — Registro y validación completa de candidatos

6.2. Buscar resultado de una elección e insertarlo en el Tricel

El Tricel almacena los resultados electorales en una lista circular enlazada. Este procedimiento describe el flujo ejecutado cuando el usuario selecciona una elección por su ID con el fin de generar, procesar e insertar su resultado dentro del historial gestionado por el Tricel.

Flujo equivalente:

1. Solicitar al usuario el ID de la elección cuyo resultado se desea generar.
2. Recorrer la lista simple de elecciones del Servel buscando coincidencia con el ID ingresado.
3. Si no existe ninguna elección con ese ID, finalizar el procedimiento.
4. Si la elección existe, llamar a la función `recopilarResultados`, la cual:
   - Cuenta votos válidos, blancos y nulos desde el ABB de mesas.
   - Calcula porcentajes de participación.
   - Determina los porcentajes por candidato.
5. Con el resultado generado, llamar a `agregarAtricel` para insertarlo en la lista circular.
6. Si la lista circular estaba vacía, enlazar el primer nodo consigo mismo.
7. Si ya contiene resultados, recorrer hasta el último nodo e insertar el nuevo resultado.
8. Finalizar informando al usuario que el resultado fue correctamente generado.

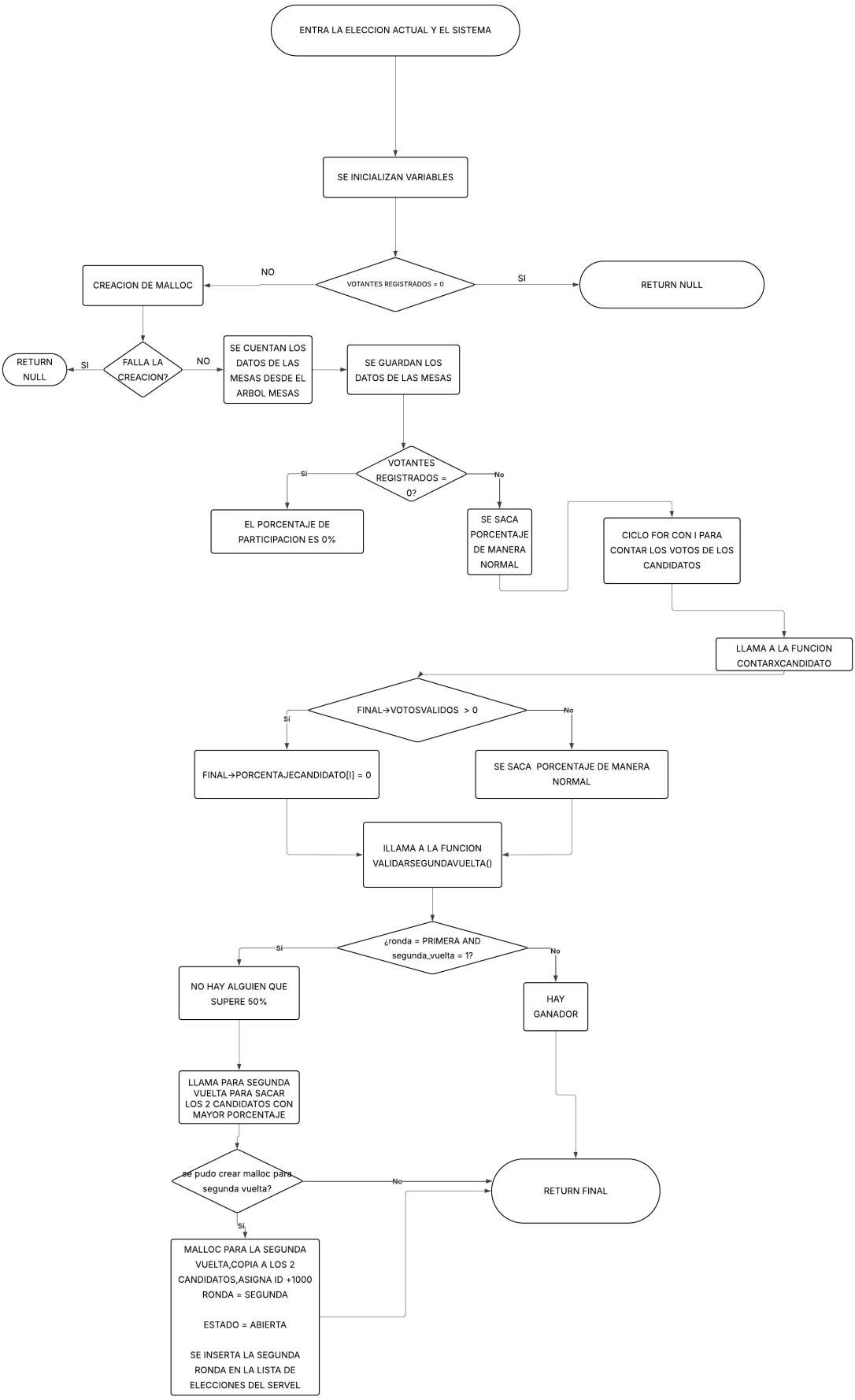

Figura 6.2: Diagrama de flujo — Búsqueda de una elección e inserción del resultado en el Tricel

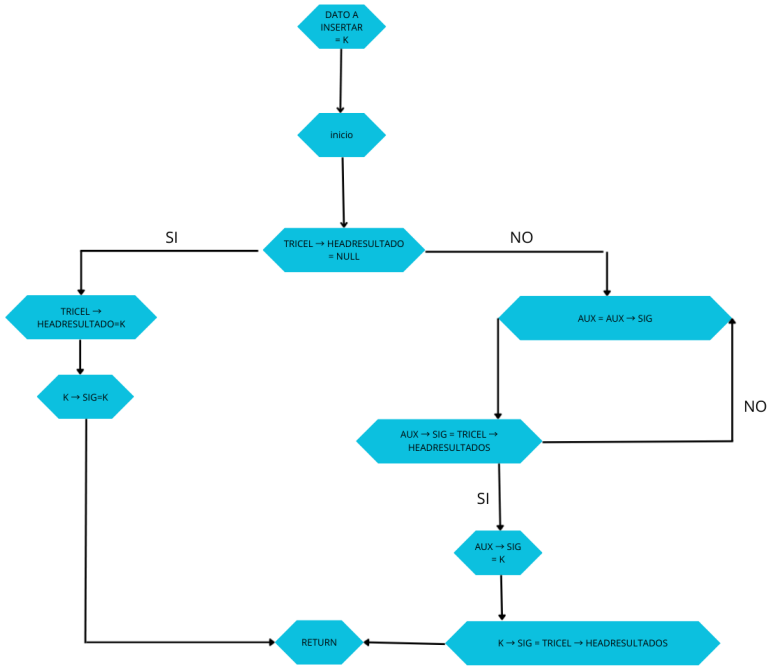

Figura 6.3: Diagrama de flujo — Proceso de recopilación interna de resultados

6.3. Diagrama funcionamiento proyecto

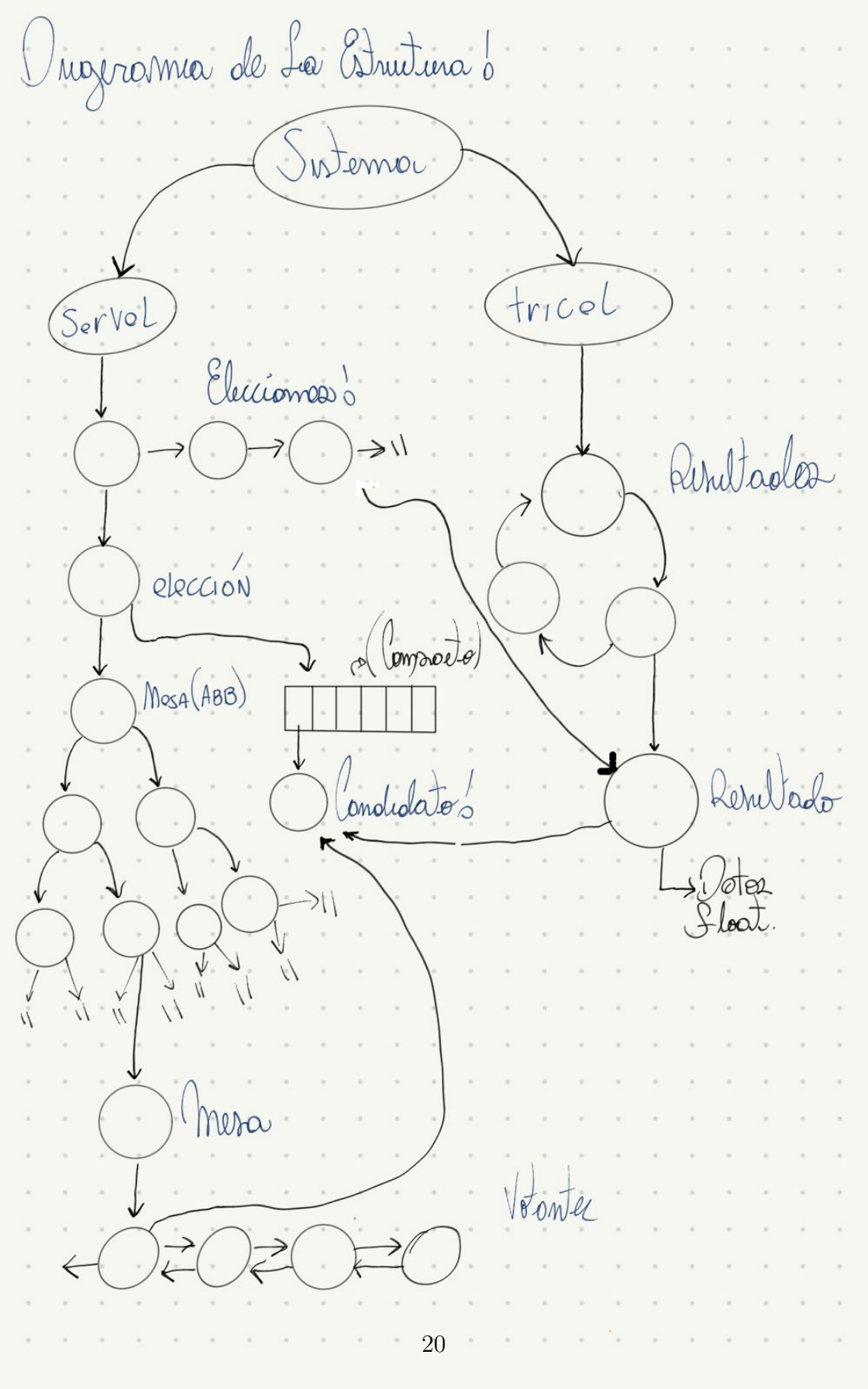

Figura 6.4: Diagrama de flujo — funcionamiento de todo el proyecto

7. Carta Gantt

7.1. Carta gantt proyecto

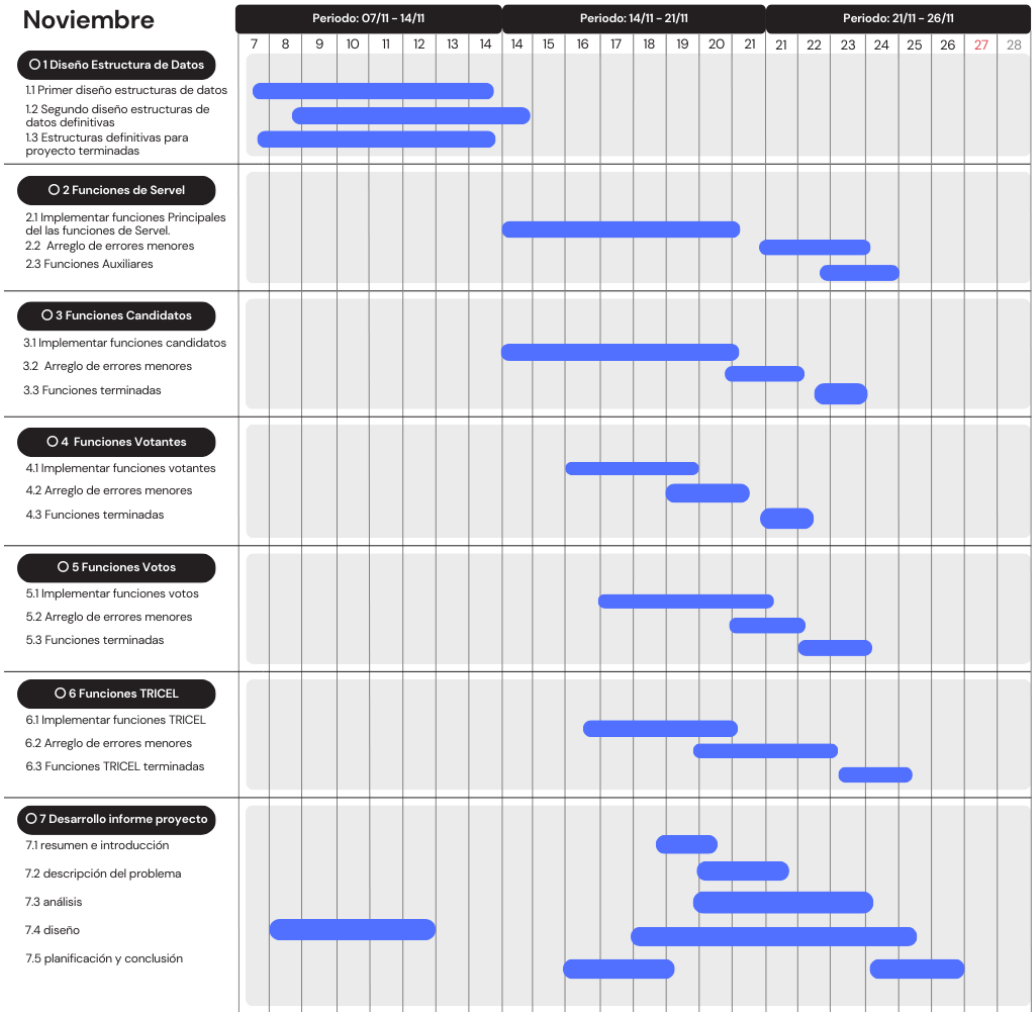

Figura 7.1: Carta Gantt grupal del proyecto

7.2. Carta gantt individual

Figura 7.2: Carta Gantt – Henry

Figura 7.3: Carta Gantt – Mateo

Figura 7.4: Carta Gantt – Rodrigo

Conclusión

El sistema electoral desarrollado permitió modelar de manera organizada las etapas esenciales de una elección presidencial, integrando el registro de candidatos, la administración de mesas, el manejo de votantes y la generación de resultados. El uso de estructuras de datos como listas, árboles binarios y arreglos posibilitó una implementación coherente y eficiente, capaz de mantener relaciones claras entre los componentes del proceso.

La separación por módulos —Servel, Mesas y Tricel— facilitó un diseño ordenado, reflejando las funciones reales de cada organismo y permitiendo un flujo de información consistente desde la creación de la elección hasta la proclamación del ganador.

En conjunto, el sistema cumple con el objetivo de ofrecer una simulación funcional y estructurada del proceso electoral, estableciendo una base sólida que puede ampliarse con nuevas características o mejoras futuras.

Anexos

Repositorio del proyecto

Para consultar el código fuente completo del sistema electoral desarrollado, se puede acceder al repositorio oficial en GitHub mediante el siguiente enlace:

https://github.com/YungRodri/proyecto-proceso-electoral