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

Evaluación modular 3: Proceso Legislativo

RIGOBERTO IGNACIO ENRIQUE CANALES PUEBLA

FRANCISCO IGNACIO ESPINOZA VASQUEZ

SIMON ANDRES LEDEZMA SEPÚLVEDA

JOAQUIN ANTONIO MUÑOZ ALARCON

INFORME DEL CÓDIGO REALIZADO

PARA LA EVALUACIÓN #3

VALPARAÍSO, 2024

Índice

1. Introducción

El sistema de gestión legislativa en Chile es un proceso complejo y estructurado que abarca varias etapas críticas, desde la creación de propuestas hasta la promulgación o veto presidencial y el control de constitucionalidad. Para abordar la necesidad de simular este proceso, se ha desarrollado una aplicación en lenguaje C que gestiona de forma integral las distintas fases de tramitación legislativa. Este sistema tiene como objetivo representar fielmente los roles y funciones de los actores principales en el proceso, como el Congreso, el Presidente, el Tribunal Constitucional y los ciudadanos, permitiendo una gestión detallada y organizada de cada uno de los elementos involucrados en la elaboración de leyes.

La aplicación busca enfrentar los desafíos propios de este proceso, tales como la verificación y control de constitucionalidad de las leyes, el registro y almacenamiento de propuestas, y la gestión de votaciones en ambas cámaras. Además, contempla la creación y publicación de leyes en el boletín oficial, respetando las fechas de vigencia y las normativas constitucionales. Cada etapa del proceso legislativo involucra una serie de decisiones y votaciones de los diputados, senadores, y ministros del Tribunal Constitucional, cuyo resultado determina si una propuesta se convierte en ley o es rechazada.

Para lograr una representación eficiente de este sistema, se han implementado diversas estructuras de datos, tales como listas enlazadas, listas circulares, y árboles binarios de búsqueda. Estas estructuras permiten almacenar y gestionar la información de manera ordenada, facilitando el acceso y la manipulación de los datos para cada proceso. Así mismo, se implementaron funciones específicas que permiten al usuario interactuar con el sistema, realizar búsquedas de propuestas, registrar y visualizar los datos de los ciudadanos y sus cargos, y gestionar las diferentes etapas legislativas de manera coherente y controlada.

Este sistema no solo es una herramienta de simulación para el proceso legislativo en Chile, sino que también constituye una base sólida para el desarrollo de sistemas legislativos más avanzados, al incorporar técnicas de programación estructurada y algoritmos de gestión de datos que optimizan el rendimiento y la organización de la información.

2. Análisis

En un sistema de gobierno, la administración del proceso legislativo es compleja debido a la variedad de etapas y actores involucrados. La gestión de propuestas de ley, que puede incluir cientos de proyectos en distintas fases de evaluación, enfrenta problemas para coordinar adecuadamente las etapas de creación, debate, votación y revisión en ambas cámaras. Además, se requiere un proceso final de promulgación o veto presidencial y un control de constitucionalidad por parte de un tribunal constitucional antes de su implementación oficial. Para solucionar esta complejidad, se ha decidido implementar un sistema de gestión legislativa que consideran las siguientes estructuras de datos:

- **ProcesoLegislativo:** Contiene los elementos principales del sistema de proceso legislativo, incluyendo un puntero al presidente (struct presidente *presidente), el congreso (struct congreso *congreso), el árbol binario de búsqueda de propuestas (struct NodoPropuestas *propuesta), la lista doblemente enlazada de ciudadanos (struct NodoCiudadano *ciudadanos), y la lista simple de boletines de estado (struct NodoBoletin *BoletinEstado).

- **presidente:** Representa al presidente dentro del proceso legislativo, con un puntero a sus datos personales (struct Persona *persona), el año de mandato actual (int anioMandato), y el voto del presidente en decisiones legislativas (int voto)

- **nodoPropuestas:** Nodo de un árbol binario de búsqueda que representa las propuestas en trámite. Cada nodo contiene la información de una propuesta (struct Propuesta *datos) y los punteros a los nodos izquierdo (struct NodoPropuestas *izq) y derecho (struct NodoPropuestas *der).

- **propuesta:** Representa una propuesta legislativa específica con información como el identificador de la propuesta (int id), el tipo de propuesta (char *tipo), el tema de la propuesta (char *tema), la persona a cargo de la propuesta (struct Persona *PersonaAcargo), y el estado de la propuesta (char *estado).

- **congreso:** Contiene las listas de miembros del congreso, divididas en senadores (struct NodoSenador *senadores) y diputados (struct NodoDiputado *diputados), cada uno representado en listas circulares doblemente enlazadas.}

- **nodoDiputado:** Nodo de una lista circular doblemente enlazada que almacena la información de un diputado en el congreso. Contiene un puntero a los datos personales del diputado (struct Persona *headDiputados), y punteros al nodo anterior (struct nodoDiputado *ant) y al nodo siguiente (struct nodoDiputado *sig).

- **nodoSenador:** Nodo de una lista circular doblemente enlazada que almacena la información de un senador en el congreso. Contiene un puntero a los datos personales del senador (struct Persona *headSenadores), y punteros al nodo anterior (struct NodoSenador *ant) y al nodo siguiente (struct NodoSenador *sig).

- **nodoCiudadano:** Nodo de una lista doblemente enlazada que representa a un ciudadano en el sistema. Contiene un puntero a la información de la persona (struct Persona *datos), además de los punteros al nodo anterior (struct NodoCiudadano *ant) y al nodo siguiente (struct NodoCiudadano *sig).

- **persona:** Contiene la información básica de una persona en el sistema. Incluye el RUT (char *rut), nombre (char *nombre), edad (int edad), cargo (int cargo, donde 0 representa ciudadano, 1 diputado, 2 senador, 3 presidente, y 4 ministro), especialidad (char *especialidad), y el voto en decisiones legislativas (int voto).

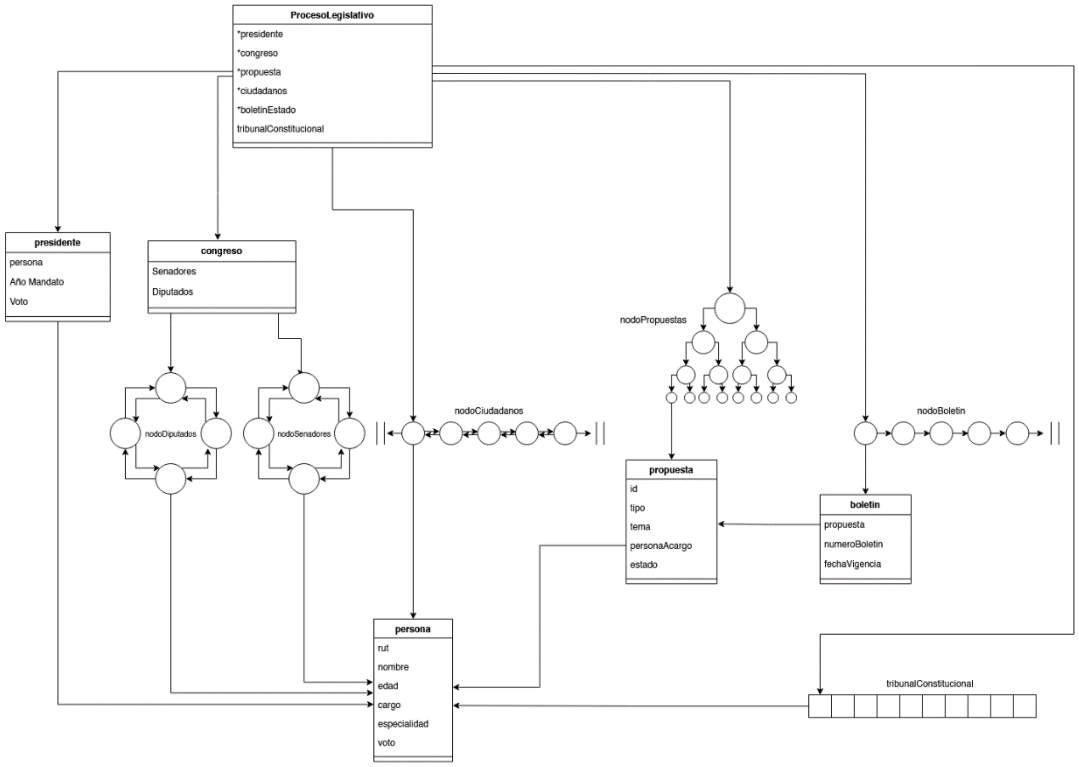
- **nodoBoletin:** Nodo de una lista simple que almacena los boletines de estado de propuestas legislativas. Cada nodo tiene un puntero a los datos del boletín (struct Boletin *head) y un puntero al siguiente nodo (struct NodoBoletin *sig).

- **boletin:** Representa un boletín legislativo específico, que incluye la propuesta asociada (struct Propuesta *propuesta), el número del boletín (int numeroBoletin), y la fecha en la que entra en vigencia (char *fechaVigencia).

3. Diseño

3.1 Diagrama de la estructura principal

Con base en las estructuras descritas previamente, el diseño del diagrama de la estructura general del proyecto a desarrollar se presenta en la figura 3.1 a continuación."



3.2 Funciones Básicas

Las funciones Básicas creadas para la solución de gestión del proceso legislativo en Chile son las siguientes:

3.2.1 Funciones para crear datos

crearPersona:

Esta función crea una nueva instancia de struct persona, asignando memoria dinámica para los campos rut, nombre, y especialidad, así como para la estructura completa. Copia los datos proporcionados a estos campos y establece el cargo como ciudadano (0) y el voto inicial.

crearCiudadano:

Esta función solicita al usuario ingresar los datos de un ciudadano, como RUT, nombre, edad y especialidad, valida que la edad sea mayor o igual a 18 años y llama a CrearPersona para almacenar estos datos en un nuevo objeto struct persona. Retorna el puntero a la estructura creada.

crearNodoCiudadano:

Esta función crea un nodo de tipo NodoCiudadano, asignando la persona proporcionada como su campo datos y estableciendo sus punteros ant y sig en NULL. Esto permite agregar al ciudadano en una lista doblemente enlazada.

crearNodoDiputado:

Esta función crea un nodo para almacenar un diputado, asignándole a headDiputados el puntero de struct persona correspondiente. Establece los punteros ant y sig en sí mismo, ya que es una lista circular doble.

crearNodoSenador:

Esta función crea un nodo circular doblemente enlazado para un senador. Asigna la persona correspondiente a headSenadores y apunta ant y sig al propio nodo.

crearPresidente:

La función CrearPresidente verifica que el ciudadano no tenga ya otro cargo. Si es elegible, solicita el año de mandato, asigna el cargo de presidente, y muestra un mensaje de éxito. Retorna el nuevo struct Presidente.

crearPropuesta:

Esta función solicita al usuario ingresar el ID, tipo y tema de la propuesta, luego llama a InsertarPropuesta para agregarla en el árbol binario de propuestas. Asigna el ciudadano a cargo y muestra un mensaje de éxito.

crearBoletin:

Esta función crea un nuevo boletín para una propuesta especificada y asigna una fecha de vigencia. Retorna el puntero al nuevo nodo del boletín.

3.2.2 Funciones para mostrar datos

mostrarCiudadanos:

La función recorre la lista de ciudadanos y muestra en consola los datos de cada uno, indicando su nombre, RUT, edad, especialidad y cargo (ciudadano, diputado, senador o presidente). Si la lista está vacía, indica que no hay ciudadanos en el sistema.

mostrarDiputados:

La función recorre la lista circular de diputados, mostrando los datos de cada uno, incluyendo su nombre, edad, especialidad y cargo. Si la lista está vacía, muestra un mensaje indicando que no hay diputados.

mostrarSenadores:

La función recorre la lista circular de senadores y muestra los datos de cada uno: nombre, edad, especialidad, y voto. Si no hay senadores, indica que la lista está vacía.

mostrarPresidente:

Esta función muestra los datos del presidente registrado, incluyendo su nombre, edad, especialidad, y año de mandato. Si no hay presidente, muestra un mensaje indicándolo.

mostrarPropuesta:

La función muestra los datos de una propuesta específica. Si la propuesta no existe, muestra un mensaje de error.

mostrarPropuestas:

Esta función recorre el árbol de propuestas en orden inorden, mostrando los datos de cada una.

mostrarTribunalConstitucional:

Esta función recorre el arreglo del tribunal y muestra los datos de los ministros registrados, como su nombre, edad, RUT, y especialidad. Si no hay ministros, muestra un mensaje indicando que está vacío.

mostrarPromedioEdadCiudadanos:

Esta función calcula y muestra el promedio de edad de los ciudadanos en la lista enlazada. Suma las edades y cuenta el total de ciudadanos para obtener el promedio. Luego, cuenta cuántos ciudadanos están por encima de esta edad promedio. Finalmente, muestra el promedio de edad, la cantidad total de ciudadanos y el porcentaje de aquellos cuya edad supera el promedio.

mostrarPorcentajeProyectosAprobados:

Esta función calcula y muestra el porcentaje de propuestas aprobadas en el árbol de propuestas. Cuenta el total de propuestas y las aprobadas usando funciones auxiliares. Si no hay propuestas, muestra un mensaje. En caso contrario, calcula el porcentaje de aprobadas, mostrándolo junto con las cantidades totales y aprobadas.

mostrarMenu:

Esta función imprime un menú de opciones para gestionar el sistema legislativo. Incluye opciones para crear, mostrar, agregar y eliminar ciudadanos, diputados, senadores, y el presidente. También permite gestionar propuestas, iniciar cámaras de origen y revisora, realizar comisiones mixtas, y procesar promulgaciones o vetos presidenciales. Además, ofrece opciones para manejar el boletín de leyes, agregar o eliminar ministros del tribunal constitucional, realizar control de constitucionalidad, y calcular estadísticas sobre ciudadanos y propuestas aprobadas. Finaliza con una opción para salir del programa.

3.2.3 Funciones para buscar datos

buscarCiudadanoPorRUT:

Esta función recorre la lista de ciudadanos buscando un nodo que coincida con el RUT especificado. Si encuentra el ciudadano, retorna el puntero a struct persona, de lo contrario, retorna NULL.

buscarDiputadoPorRUT:

Esta función recorre una lista circular de diputados y verifica si el RUT proporcionado corresponde a alguno de ellos. Si encuentra coincidencia, retorna 1; si no, retorna 0.

buscarSenadorPorRUT:

Esta función recorre la lista circular de senadores y verifica si alguno tiene el RUT especificado. Retorna 1 si lo encuentra y 0 si no hay coincidencias.

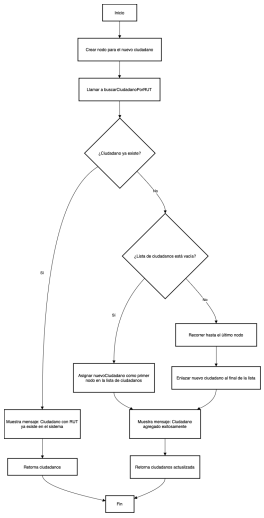
buscarPropuesta:

Esta función busca en el árbol binario de búsqueda de propuestas un nodo cuyo ID coincida con el especificado, retornando el puntero a struct propuesta o NULL si no encuentra coincidencias.

3.2.4 Funciones para agregar datos

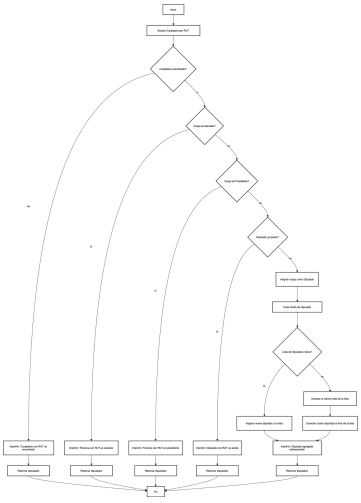
agregarCiudadano:

La función verifica que el ciudadano no esté ya registrado en la lista, y si no lo está, crea un nodo NodoCiudadano para agregarlo al final de la lista doblemente enlazada. Retorna la lista actualizada.



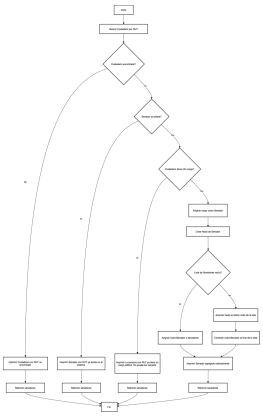
agregarDiputado:

Esta función verifica que el ciudadano no tenga otro cargo (senador o presidente) y que no esté ya registrado como diputado. Si pasa estas validaciones, se actualiza el cargo del ciudadano a diputado y se agrega su nodo a la lista circular de diputados.



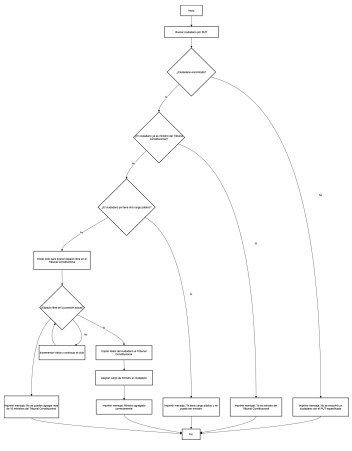
agregarSenador:

Esta función agrega un ciudadano como senador tras verificar que no tiene ya un cargo. Si cumple, cambia el cargo y agrega el nodo a la lista de senadores. Retorna la lista de senadores actualizada.



agregarMinistroTribunalConstitucional:

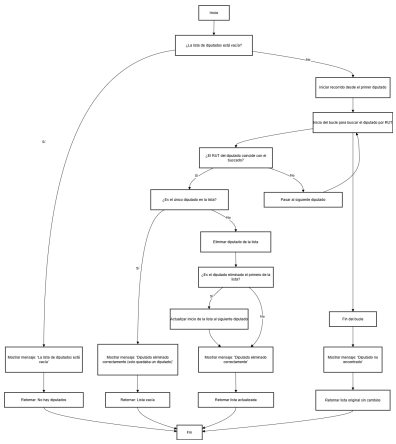
Esta función busca un ciudadano y, si cumple con los requisitos, lo añade como ministro en el tribunal constitucional. Verifica que el ciudadano no tenga ya otro cargo.



3.2.5 Funciones para eliminar datos

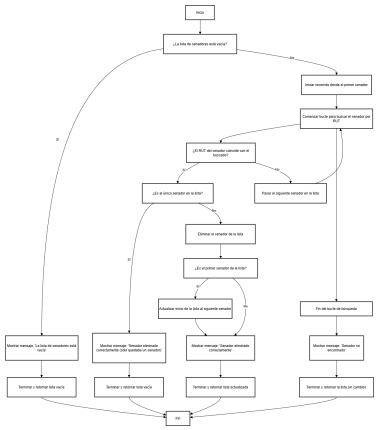
eliminarDiputado:

Esta función busca un diputado por RUT en la lista circular doblemente enlazada y lo elimina, reajustando los punteros. Si no hay más diputados en la lista, establece la lista como vacía.



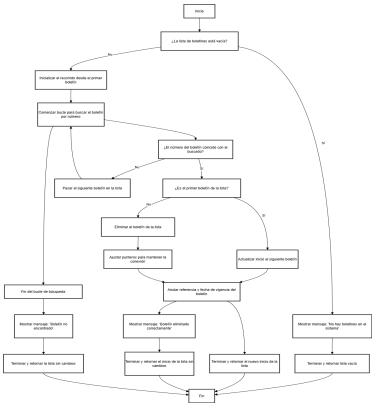
eliminarSenador:

Esta función es similar a EliminarDiputado, pero busca un senador en la lista circular y lo elimina, reajustando los punteros. Si no hay más senadores, la lista queda vacía.



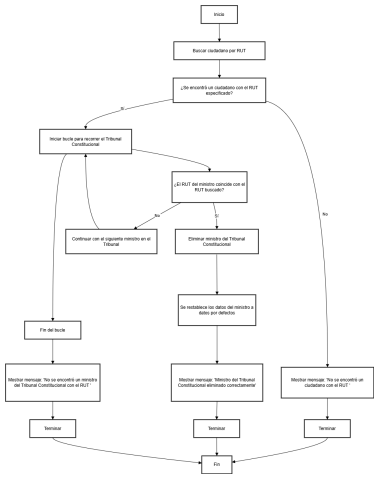
eliminarLeyDeBoletin:

Esta función busca un boletín por su número y lo elimina de la lista enlazada simple. Si el boletín no existe, muestra un mensaje de error.



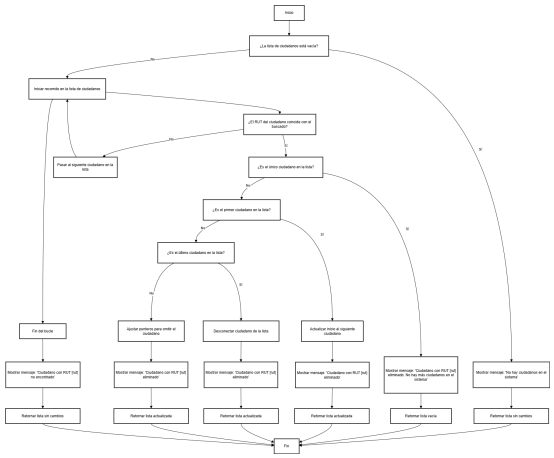
eliminarMinistroTribunalConstitucional:

Esta función elimina un ministro del tribunal constitucional por su RUT, liberando su espacio en el arreglo.



eliminarCiudadano:

La función elimina un ciudadano de la lista doblemente enlazada de ciudadanos según su RUT. Si el ciudadano es el único en la lista, la vacía; si está al inicio, ajusta el inicio de la lista; si está al final, desconecta el último nodo; y si está entre otros ciudadanos, ajusta los enlaces anterior y siguiente. Retorna la lista de ciudadanos actualizada, o muestra un mensaje si el ciudadano no se encuentra.



3.2.6 Funciones de Proceso Legislativo

camaraDeOrigen:

La función busca una propuesta en el árbol por su ID y, dependiendo de su tipo, inicia la votación en la cámara correspondiente (Diputados o Senado). Cuenta los votos y determina si la propuesta avanza a la siguiente etapa o es rechazada.

camaraRevisora:

La función simula la votación de una propuesta en la cámara revisora. Cuenta votos a favor y en contra, y permite que la cámara sugiera modificaciones. Dependiendo de los votos, decide si la propuesta avanza o es rechazada.

comisionMixta:

Esta función envía una propuesta a la Comisión Mixta en caso de discrepancias. Diputados y senadores votan, y la propuesta es aprobada si ambos cuerpos alcanzan un consenso.

publicarLeyEnBoletin:

Esta función publica una ley en el boletín oficial, permitiendo al usuario especificar la fecha de vigencia. Agrega el nuevo boletín al final de la lista.

promulgacionOVetoPresidencial:

La función permite al presidente decidir si promulga, veta totalmente, o veta parcialmente una propuesta. En el caso de veto parcial, el Congreso puede aceptar las modificaciones o rechazar el veto con una mayoría de dos tercios.

controlConstitucionalidad:

La función verifica si una propuesta está publicada en el boletín. Si es así, los ministros votan, y la función determina si la propuesta es constitucional o debe eliminarse del boletín.

3.2.7 Funciones Auxiliares

pause:

Esta función muestra un mensaje en la consola solicitando al usuario que presione Enter para continuar. Utiliza getchar() para esperar la entrada del usuario, pausando el flujo del programa.

cls:

Esta función simula la limpieza de pantalla en consola imprimiendo 100 líneas en blanco. Esto hace que el contenido previo quede oculto en la terminal.

limpiarBuffer:

La función vacía el buffer de entrada de la consola, descartando cualquier carácter pendiente, como un salto de línea, hasta que se detecta el final de línea o de archivo. Esto ayuda a prevenir problemas al cambiar entre scanf y fgets.

verificarTC:

Esta función recorre el tribunal constitucional y verifica si todas las posiciones están ocupadas. Retorna 1 si hay suficientes ministros y 0 si no los hay.

insertarPropuesta:

Esta función agrega una propuesta a un árbol binario de búsqueda. Si la raíz es NULL, crea un nodo para la nueva propuesta. En caso contrario, la inserta en el subárbol izquierdo o derecho, según su ID, y muestra un mensaje si el ID ya existe.

poblarTribunalConstitucional

Esta función inicializa un arreglo de 10 posiciones para el tribunal constitucional, asignando valores por defecto para indicar que las posiciones están vacías.

realizarVotacionMinistros

Esta función permite a cada ministro votar sobre una propuesta, contando votos a favor y en contra y determinando si la propuesta es aprobada o rechazada.

contarPropuestas

Esta función cuenta el número total de propuestas en un árbol binario de búsqueda, sumando recursivamente cada nodo para obtener el total de propuestas almacenadas en el árbol.

contarPropuestasAprobadas

Esta función recorre un árbol binario de búsqueda de propuestas y cuenta únicamente las propuestas aprobadas. Para cada nodo, verifica si la propuesta está en estado aprobado y, de ser así, suma 1 al total, continuando la cuenta en los subárboles izquierdo y derecho. La función devuelve el total de propuestas aprobadas en el árbol.

3.2.8 Función principal

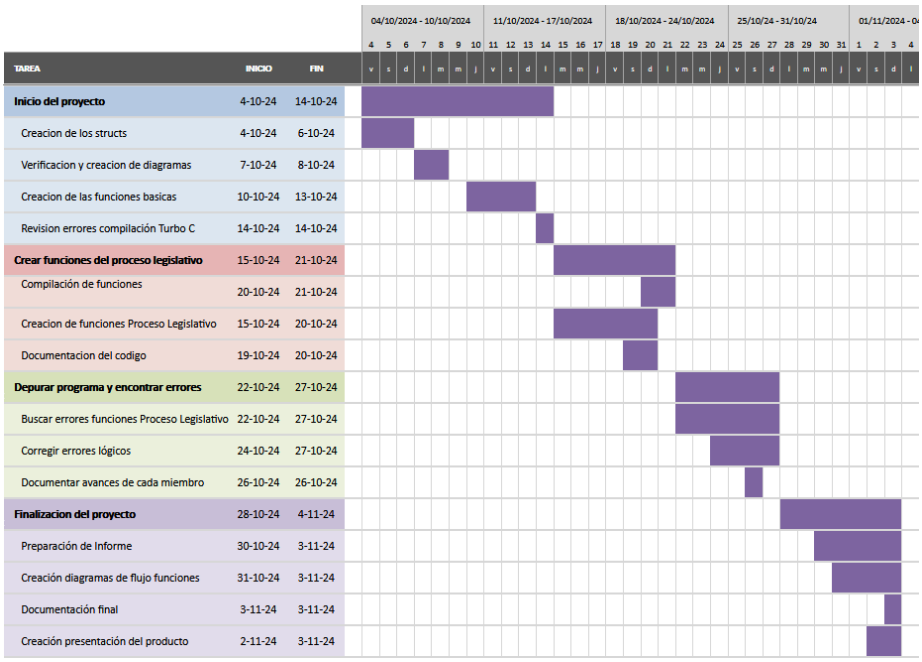
main:

Esta función principal muestra un menú interactivo para gestionar el sistema legislativo. Inicializa el sistema y crea una estructura que contiene el Congreso, propuestas, ciudadanos, el presidente, y el boletín de estado. Utiliza un bucle para presentar opciones de menú, como crear, mostrar, agregar y eliminar ciudadanos, diputados, senadores y el presidente, así como gestionar propuestas legislativas. Además, ofrece opciones para la promulgación, control de constitucionalidad, y estadísticas. Cada opción llama a funciones específicas y permite realizar el flujo completo de un proceso legislativo.

4. Planificación

A continuación, se muestra la distribución del tiempo asignado a las diferentes tareas realizadas durante el proyecto, iniciando con la planificación grupal (trabajo en conjunto) y continuando con las planificaciones individuales.

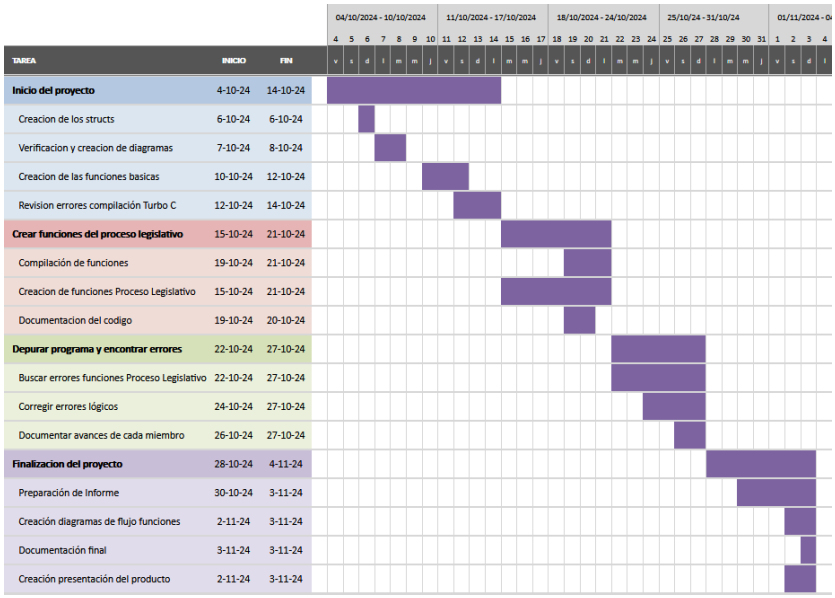
4.1 Planificación grupal



TAREA	INICIO	FIN
Inicio del proyecto	4-10-24	14-10-24
Creacion de los structs	4-10-24	6-10-24
Verificacion y creacion de diagramas	7-10-24	8-10-24
Creacion de las funciones basicas	10-10-24	13-10-24
Revision errores compilación Turbo C	14-10-24	14-10-24
Crear funciones del proceso legislativo	15-10-24	21-10-24
Compilación de funciones	20-10-24	21-10-24
Creacion de funciones Proceso Legislativo	15-10-24	20-10-24
Documentacion del codigo	19-10-24	20-10-24
Depurar programa y encontrar errores	22-10-24	27-10-24
Buscar errores funciones Proceso Legislativo	22-10-24	27-10-24
Corregir errores lógicos	24-10-24	27-10-24
Documentar avances de cada miembro	26-10-24	26-10-24
Finalizacion del proyecto	28-10-24	4-11-24
Preparación de Informe	30-10-24	3-11-24
Creación diagramas de flujo funciones	31-10-24	3-11-24
Documentación final	3-11-24	3-11-24
Creación presentación del producto	2-11-24	3-11-24

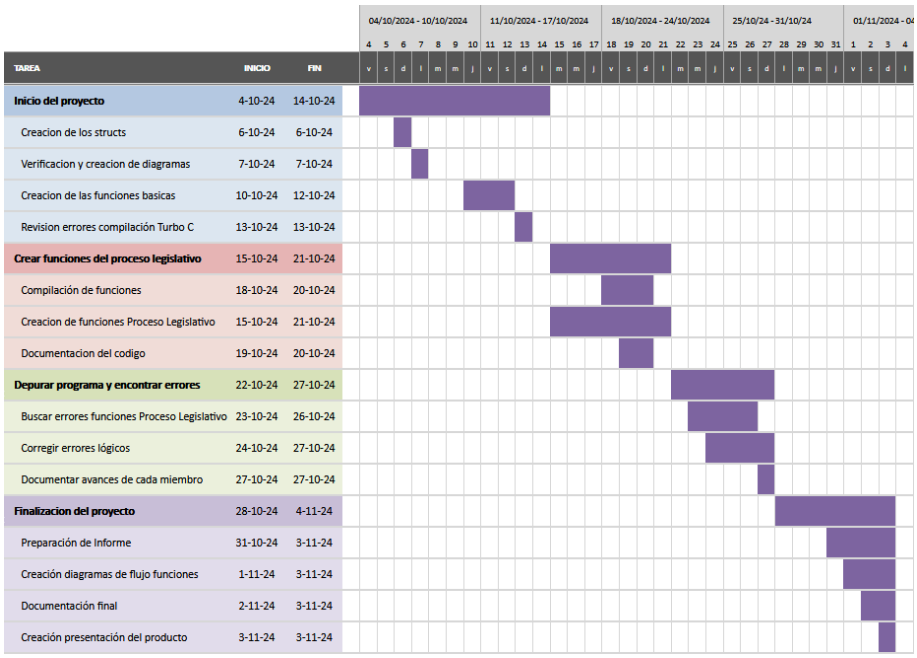
4.2 Planificación individual

4.2.1 Francisco Espinoza



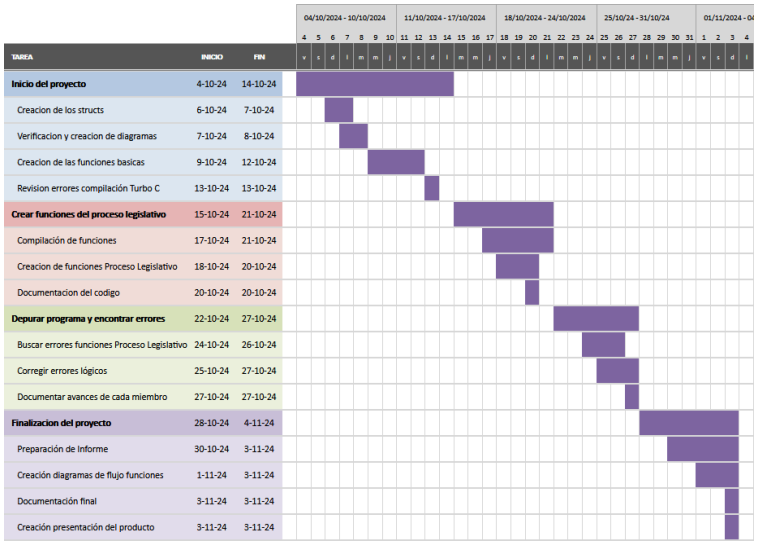
TAREA	INICIO	FIN
Inicio del proyecto	4-10-24	14-10-24
Creacion de los structs	6-10-24	6-10-24
Verificacion y creacion de diagramas	7-10-24	8-10-24
Creacion de las funciones basicas	10-10-24	12-10-24
Revision errores compilación Turbo C	12-10-24	14-10-24
Crear funciones del proceso legislativo	15-10-24	21-10-24
Compilación de funciones	19-10-24	21-10-24
Creacion de funciones Proceso Legislativo	15-10-24	21-10-24
Documentacion del codigo	19-10-24	20-10-24
Depurar programa y encontrar errores	22-10-24	27-10-24
Buscar errores funciones Proceso Legislativo	22-10-24	27-10-24
Corregir errores lógicos	24-10-24	27-10-24
Documentar avances de cada miembro	26-10-24	27-10-24
Finalizacion del proyecto	28-10-24	4-11-24
Preparación de Informe	30-10-24	3-11-24
Creación diagramas de flujo funciones	2-11-24	3-11-24
Documentación final	3-11-24	3-11-24
Creación presentación del producto	2-11-24	3-11-24

4.2.2 Joaquín Muñoz



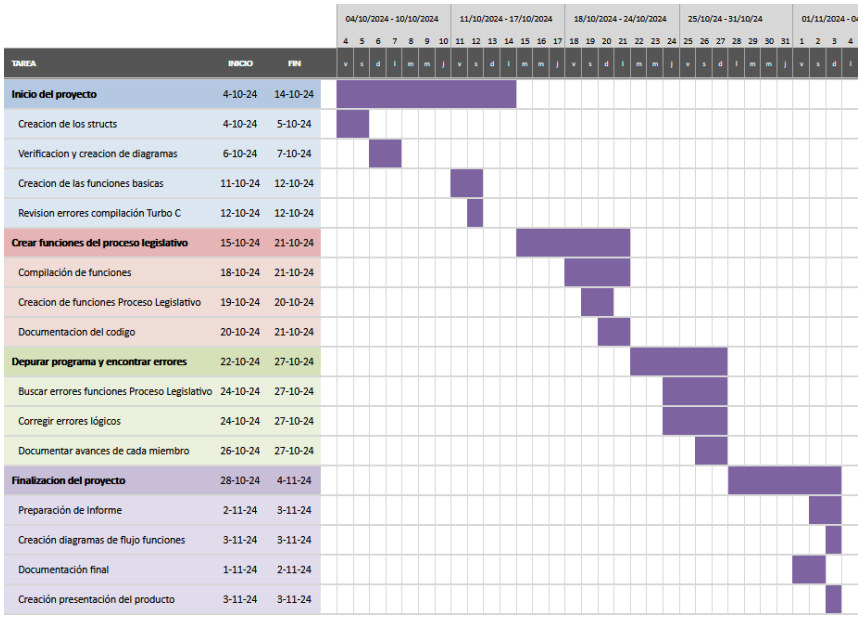
TAREA	INICIO	FIN
Inicio del proyecto	4-10-24	14-10-24
Creacion de los structs	6-10-24	6-10-24
Verificacion y creacion de diagramas	7-10-24	7-10-24
Creacion de las funciones basicas	10-10-24	12-10-24
Revision errores compilación Turbo C	13-10-24	13-10-24
Crear funciones del proceso legislativo	15-10-24	21-10-24
Compilación de funciones	18-10-24	20-10-24
Creacion de funciones Proceso Legislativo	15-10-24	21-10-24
Documentacion del codigo	19-10-24	20-10-24
Depurar programa y encontrar errores	22-10-24	27-10-24
Buscar errores funciones Proceso Legislativo	23-10-24	26-10-24
Corregir errores lógicos	24-10-24	27-10-24
Documentar avances de cada miembro	27-10-24	27-10-24
Finalizacion del proyecto	28-10-24	4-11-24
Preparación de Informe	31-10-24	3-11-24
Creación diagramas de flujo funciones	1-11-24	3-11-24
Documentación final	2-11-24	3-11-24
Creación presentación del producto	3-11-24	3-11-24

4.2.3 Rigoberto Canales



TAREA	INICIO	FIN
Inicio del proyecto	4-10-24	14-10-24
Creacion de los structs	6-10-24	7-10-24
Verificacion y creacion de diagramas	7-10-24	8-10-24
Creacion de las funciones basicas	9-10-24	12-10-24
Revision errores compilación Turbo C	13-10-24	13-10-24
Crear funciones del proceso legislativo	15-10-24	21-10-24
Compilación de funciones	17-10-24	21-10-24
Creacion de funciones Proceso Legislativo	18-10-24	20-10-24
Documentacion del codigo	20-10-24	20-10-24
Depurar programa y encontrar errores	22-10-24	27-10-24
Buscar errores funciones Proceso Legislativo	24-10-24	26-10-24
Corregir errores lógicos	25-10-24	27-10-24
Documentar avances de cada miembro	27-10-24	27-10-24
Finalizacion del proyecto	28-10-24	4-11-24
Preparación de Informe	30-10-24	3-11-24
Creación diagramas de flujo funciones	1-11-24	3-11-24
Documentación final	3-11-24	3-11-24
Creación presentación del producto	3-11-24	3-11-24

4.2.4 Simón Ledezma



TAREA	INICIO	FIN
Inicio del proyecto	4-10-24	14-10-24
Creacion de los structs	4-10-24	5-10-24
Verificacion y creacion de diagramas	6-10-24	7-10-24
Creacion de las funciones basicas	11-10-24	12-10-24
Revision errores compilación Turbo C	12-10-24	12-10-24
Crear funciones del proceso legislativo	15-10-24	21-10-24
Compilación de funciones	18-10-24	21-10-24
Creacion de funciones Proceso Legislativo	19-10-24	20-10-24
Documentacion del codigo	20-10-24	21-10-24
Depurar programa y encontrar errores	22-10-24	27-10-24
Buscar errores funciones Proceso Legislativo	24-10-24	27-10-24
Corregir errores lógicos	24-10-24	27-10-24
Documentar avances de cada miembro	26-10-24	27-10-24
Finalizacion del proyecto	28-10-24	4-11-24
Preparación de Informe	2-11-24	3-11-24
Creación diagramas de flujo funciones	3-11-24	3-11-24
Documentación final	1-11-24	2-11-24
Creación presentación del producto	3-11-24	3-11-24

5. Conclusión

El desarrollo de este proyecto de sistema de gestión legislativa en C ha sido una excelente oportunidad para familiarizarnos con la programación estructurada y profundizar en el manejo de estructuras de datos avanzadas como listas enlazadas, listas circulares y árboles binarios. La creación y gestión de los diferentes roles y fases del proceso legislativo nos hizo ver la importancia de cada detalle en el diseño de un sistema robusto y funcional. A lo largo del proyecto, pudimos observar cómo las distintas funcionalidades de la aplicación, como la creación de propuestas, la organización del Congreso y la gestión del Tribunal Constitucional, requieren no solo una lógica clara sino también una estructura de datos que permita un flujo ordenado y eficiente de la información.

Este proyecto nos ha permitido considerar múltiples escenarios legislativos que no solo han enriquecido la aplicación, sino también nos han hecho conscientes de la complejidad y los desafíos que conlleva una implementación de este tipo. Trabajar en equipo para resolver estos retos ha sido fundamental para el éxito del proyecto, ya que la colaboración entre todos ha permitido que cada fase sea bien implementada y optimizada.

Si bien el proyecto ha sido exitoso, identificamos áreas de mejora para futuros desarrollos, como la organización de prioridades en el diseño de funciones clave y el flujo de información entre las distintas etapas del sistema. Nos dimos cuenta de que definir una estructura sólida desde el principio habría facilitado el desarrollo de cada componente y nos habría permitido ahorrar tiempo al evitar modificaciones constantes en las estructuras de datos.

Este "mini proyecto" ha sido desafiante y nos ha obligado a reforzar conocimientos previos, aprender a diseñar menús interactivos y resolver problemas de lógica y compilación. Sin duda, esta experiencia nos ha proporcionado una valiosa perspectiva sobre los requerimientos de sistemas legislativos y nos ha preparado para afrontar proyectos más complejos en el futur

6. Anexo

6.1 Currículums de los integrantes

https://www.linkedin.com/in/francisco-ignacio-espinoza-vasquez-3155b8336/
https://www.linkedin.com/in/rigoberto-canales-puebla-479032337/
https://www.linkedin.com/in/joaquin-muñoz-8649b4336/
https://www.linkedin.com/in/simon-ledezma-ab4386336/

6.2 Enlace de GitHub al código fuente

https://github.com/FranciscoEsp01/estructura-de-datos2