

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

SISTEMA DE APOYO AL PROCESO LEGISLATIVO DE CHILE

Francisco Javier Reyes Villalon
Byron Alexander Vásquez Aguilera
Nicolas Alberto Gonzalez Urrutia
Ricardo Benjamín Vergara Godoy

Noviembre 2024

Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

SISTEMA DE APOYO AL PROCESO LEGISLATIVO DE CHILE

Francisco Javier Reyes Villalon

Byron Alexander Vásquez Aguilera

Nicolas Alberto Gonzalez Urrutia

Ricardo Benjamín Vergara Godoy

Profesor Guía: Rafael Mellado Silva

Carrera: Ingeniería en Informática

Noviembre 2024

INDICE

Resumen	iii
Abstract	iii
Glosario	iv
Lista de Figuras	v
1. Introducción	1
2. Análisis del Problema y Estructura de Datos.....	2
2.1. Contexto del Problema	2
2.2. Requisitos del Sistema	2
2.3. Selección de Estructuras de Datos	2
2.3.1. Lista Enlazada para Ciudadanos y Parlamentarios.....	2
2.3.2. Árbol Binario de Búsqueda (ABB) para Proyectos de Ley.....	3
2.3.3. Otras Estructuras	3
2.4. Justificación de las Estructuras Seleccionadas	3
2.5. Diagrama Estructura de Datos.....	3
3. Diseño del Sistema	5
3.1 Componentes Principales.....	5
3.2 Diagrama de Estructuras de Datos.....	6
3.3 Flujo de Trabajo del Sistema.....	7
3.4 Diagrama de Flujo	7
4. Implementación Técnica	7
4.1 Estructuras de Datos	7
4.2 Funcionalidades Principales	8
4.3 Modularización y Organización del Código	8
5. Resultados y Conclusiones	8
5.1 Resultados Obtenidos	8
5.2 Menú Principal del Sistema	9

5.3 Conclusión	10
6. Bibliografía.....	12
Anexos.....	13
A Diagrama de Flujo del Proceso Legislativo	13
B Carta Gantt	14
C Currículums Grupo	20
D Código Fuente	24

Resumen

Este Proyecto esta desarrollado en un sistema en C, que simula el proceso legislativo de Chile, con la finalidad de gestionar y seguir el avance de los proyectos de ley desde su inicio que es la creación de el hasta el la votación y decisión final en el Congreso Nacional. Este programa en C permite al usuario la creación, modificación, y búsqueda de los proyectos de ley, utilizando las distintas estructuras de datos dinámicas y estáticas. El sistema emplea distintas estructuras de datos para representar las diferentes etapas del proceso legislativo como son listas enlazadas empleadas en los ciudadanos y parlamentarios, y un árbol binario de búsqueda (ABB) para almacenar los proyectos de ley.

Palabras clave: Proceso legislativo, Chile, proyectos de ley, árbol binario de búsqueda, listas enlazadas.

Abstract

This project is developed in a C system, which simulates the legislative process in Chile, in order to manage and follow the progress of the bills from the beginning, which is the creation of the bill, to the final vote and decision in the National Congress. This C program allows the user to create, modify, and search for bills, using different dynamic and static data structures. The system uses different data structures to represent the different stages of the legislative process such as linked lists used by citizens and parliamentarians, and a binary search tree (ABB) to store the bills.

Keywords: Legislative process, Chile, bills, binary search tree, linked lists.

Glosario

ABB: Árbol Binario de Búsqueda, Son aquellos árboles en los que todos sus nodos excepto los del último nivel, tienen dos hijos: el subárbol izquierdo y el subárbol derecho

Cámara de Origen: Cámara a la que ingresa el proyecto de ley para su tramitación.

Diagrama de Flujo: es la representación gráfica de un algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

Estructura de Datos: medios para manejar grandes cantidades de información de manera eficiente para usos tales como grandes bases de datos y servicios de indización de Internet.

Lista Enlazada Circular: Tipo de lista donde el último nodo se conecta de nuevo al primer nodo, formando un ciclo.

Lista Enlazada Doble: Estructura de datos donde cada nodo está vinculado al siguiente y al anterior nodo en la lista. Este tipo de lista facilita la navegación en ambas direcciones.

Nodo: Unidad de una estructura de datos, cada nodo contiene un valor o datos, y uno o más enlaces a otros nodos, dependiendo del tipo de estructura.

Modularización Técnica en programación que divide un sistema en módulos que cada uno cumple una función específica dentro de programa. Esto facilita el mantenimiento y comprensión del sistema.

Parlamentario: Término genérico para referirse a los miembros del Congreso, que incluye a diputados y senadores.

Puntero: Un puntero en programación es una variable que almacena la dirección de memoria de otra variable.

Quórum: Número mínimo de parlamentarios que deben estar presentes para que una votación sea válida.

Senador: Miembro de la Cámara del Senado, que, junto con la Cámara de Diputados, conforma el Congreso y participa en la aprobación de proyectos de ley.

Lista de Figuras

Figura 2.1 Diagrama Estructura de datos del Programa.....	4
Figura 3.2 Proceso de Eliminación de Parlamentario.....	5
Figura 3.3 Proceso de Agregar Proyecto de Ley.....	6
Figura 5.1 Menú Programa.....	10
Figura A.1 Diagrama de flujo.....	12
Figura B.1 Carta Gantt Grupal Página 1.....	13
Figura B.2 Carta Gantt Grupal Página 2.....	14
Figura B.3 Carta Gantt Nicolas Gonzalez Página 1.....	15
Figura B.4 Carta Gantt Nicolas Gonzalez Página 2.....	15
Figura B.5 Carta Gantt Ricardo Vergara Página 1.....	16
Figura B.6 Carta Gantt Ricardo Vergara Página 2.....	16
Figura B.7 Carta Gantt Byron Vásquez Página 1.....	17
Figura B.8 Carta Gantt Byron Vásquez Página 2.....	17
Figura A.9 Carta Gantt Francisco Reyes Página 1.....	18
Figura B.10 Carta Gantt Francisco Reyes Página 2.....	18
Figura C.1 Curriculum Ricardo Vergara Godoy.....	19
Figura C.2 Curriculum Nicolas Gonzalez Urrutia.....	20
Figura C.3 Curriculum Byron Vasquez Aguilera.....	21
Figura C.4 Curriculum Francisco Reyes Villalón.....	22

1. Introducción

En el sistema democrático chileno el proceso legislativo cumple un rol fundamental, dando la seguridad que las leyes se formulen y revisen cuidadosamente antes de su aprobación final. Este proceso implica varias etapas y la participación de distintas entidades como pueden ser los ciudadanos, parlamentarios y presidente. Sin embargo, dada la dificultad y cuidado de este proceso puede ser un reto comprender y analizar el como un proyecto de ley avanza desde su introducción hasta su aprobación o rechazo. Es por eso por lo que surge la necesidad de contar con herramientas que permitan simular este gran proceso legislativo de manera estructurada.

El presente proyecto tiene como objetivo desarrollar un sistema de software en C que simule el proceso legislativo de Chile, permitiendo gestionar y monitorear los proyectos de ley a lo largo de sus etapas. El sistema emplea diversas estructuras de datos, como listas enlazadas y árboles binarios de búsqueda, para representar los participantes y etapas del proceso legislativo, proporcionando funciones de creación, modificación, búsqueda y votación de proyectos de ley. Además, el sistema incluye funcionalidades adicionales que enriquecen el control sobre los proyectos, como la verificación de firmas ciudadanas y la gestión de parlamentarios.

Este informe está organizado de la siguiente manera: en la Sección 2 se detalla el análisis del problema y la estructura de datos seleccionada. La Sección 3 describe el diseño del sistema, incluyendo diagramas de flujo y ejemplos de implementación. La Sección 4 abarca la implementación técnica, explicando las funciones desarrolladas y la estructura modular del código. La Sección 5 presenta los resultados obtenidos y las conclusiones sobre la funcionalidad y utilidad del sistema, destacando posibles mejoras futuras. Finalmente, los anexos incluyen el código fuente, currículos de cada participante, carta Gantt con nuestro proceso en este proyecto y otros elementos complementarios que facilitan la comprensión del proyecto.

2. Análisis del Problema y Estructura de Datos

2.1. Contexto del Problema

El proceso legislativo de Chile es una parte importante de su sistema democrático y está estructurado para garantizar que los proyectos de ley sean discutidos y evaluados exhaustivamente antes de convertirse en ley. Este proceso involucra etapas como propuestas legislativas, votaciones en la Cámara y el Senado y revisión presidencial, y requiere la participación de diferentes entidades como ciudadanos, parlamentarios y comités. Sin embargo, dada la complejidad del proceso y el número de participantes, se necesita una herramienta que pueda representar y simular el proceso de forma ordenada y comprensible.

2.2. Requisitos del Sistema

Para modelar el proceso legislativo, el sistema debe cumplir con los siguientes requisitos funcionales:

- 1) **Gestión de Proyectos de Ley:** Permitir la creación, modificación, eliminación y búsqueda de proyectos de ley, cada uno con atributos como ID, nombre, descripción, nivel de urgencia, artículos y fecha de creación.
- 2) **Verificación de Firmas:** Verificar que un proyecto de ley propuesto por ciudadanos cuente con el número mínimo (3000) de firmas para ser presentado.
- 3) **Asignación de Proyectos:** Asignar cada proyecto de ley a una cámara de origen que puede ser o diputados o senadores.
- 4) **Gestión de Votación:** Almacenar y manejar las votaciones de parlamentarios en ambas cámaras, determinando si un proyecto de ley es aprobado o rechazado por alguna.
- 5) **Funciones Adicionales:** Ofrecer funciones como la identificación del proyecto de ley más votado y la eliminación de parlamentarios, para aumentar el control y la ayuda del sistema.

2.3. Selección de Estructuras de Datos

Para cumplir con los requisitos del sistema anteriormente mencionados, se ha seleccionado una combinación de estructuras de datos estáticas y dinámicas que permiten un control adecuado de cada etapa del proceso legislativo. A continuación, se explican las estructuras elegidas y su justificación.

2.3.1. Lista Enlazada para Ciudadanos y Parlamentarios

Dado que los ciudadanos y parlamentarios son entidades dinámicas que requieren inserción y eliminación frecuente, se determina utilizar listas enlazadas para representarlos:

En el caso de los ciudadanos se utilizará una lista doblemente enlazada que permite registrar a cada ciudadano y realizar un conteo rápido de firmas cuando un proyecto de ley se origina en la ciudadanía.

Una lista doblemente enlazada para parlamentarios que facilita la manipulación y eliminación de datos de parlamentarios, además de permitir el registro de sus votos de manera eficiente.

2.3.2. Árbol Binario de Búsqueda (ABB) para Proyectos de Ley

Se ha seleccionado un árbol binario de búsqueda (ABB) para organizar los proyectos de ley por su ID, lo que facilita búsquedas rápidas y garantiza un acceso eficiente a los proyectos dada sus características de búsqueda que otorga el árbol binario:

Organización por ID: Permite acceder, insertar o eliminar un proyecto en tiempo logarítmico, optimizando la gestión de proyectos de ley a medida que el sistema crece.

Almacenamiento jerárquico: Facilita la identificación del proyecto más votado y simplifica la búsqueda de proyectos en el flujo legislativo.

2.3.3. Otras Estructuras

Para otros elementos como artículos de ley y votos, se emplean estructuras más simples que permiten una gestión rápida:

Array de punteros para artículos en cada proyecto de ley, permitiendo un acceso indexado a los artículos.

Árbol binario para almacenar los votos de cada parlamentario, organizados por el ID del proyecto de ley.

2.4. Justificación de las Estructuras Seleccionadas

La combinación de listas enlazadas y árboles binarios permite que el sistema gestione de manera eficaz tanto los datos secuenciales como son las firmas de ciudadanos, y también como los datos jerárquicos como los proyectos de ley. Las listas enlazadas garantizan que los datos de los ciudadanos y parlamentarios sean fácilmente modificables, mientras que el árbol binario asegura un acceso rápido y ordenado a los proyectos de ley. De esta forma, el sistema cumple con los requerimientos de rendimiento y flexibilidad necesarios para simular adecuadamente el proceso legislativo chileno.

2.5. Diagrama Estructura de Datos

Este diagrama ilustra la estructura de datos utilizada en el programa, mostrando la organización y las relaciones entre las distintas estructuras y nodos. Cada nodo y estructura representan elementos clave del modelo de datos. El diagrama destaca cómo cada componente interactúa, y cómo las estructuras se vinculan a través de punteros, formando listas enlazadas y jerarquías necesarias para la lógica del sistema.

3. Diseño del Sistema

El diseño del sistema se estructura en torno a varios módulos que permiten gestionar y simular las diferentes etapas del proceso legislativo en Chile. Cada módulo se encarga de manejar uno de los elementos clave: ciudadanos, parlamentarios, proyectos de ley y votaciones. Este enfoque modular no solo mejora la organización y eficiencia del código, sino que facilita futuras modificaciones o expansiones en el sistema.

3.1 Componentes Principales

Para cumplir con los requisitos del sistema, se han implementado los siguientes componentes:

Gestión de Ciudadanos: Este módulo permite registrar ciudadanos y verificar sus firmas cuando un proyecto de ley es de origen ciudadano. Se utiliza una lista doblemente enlazada circular que facilita la inserción y el recorrido de los datos. La función `agregarNodoPersona` permite añadir nuevos ciudadanos a la lista, y `contarFirmasPersonas` verifica que se cumpla con el mínimo de firmas requerido. Este proceso se ilustra en el siguiente diagrama de flujo, que muestra el ingreso de ciudadanos y su verificación:

Gestión de Parlamentarios: Este módulo organiza los datos de los diputados y senadores en listas enlazadas dobles, lo que permite el manejo dinámico de estos registros. Además, cada parlamentario tiene un historial de votos registrado en un árbol binario. La función `crearParlamentario` inicializa un nuevo parlamentario, mientras que `eliminarParlamentario` permite gestionar su baja. A continuación, se muestra el flujo del proceso de eliminación de un parlamentario:

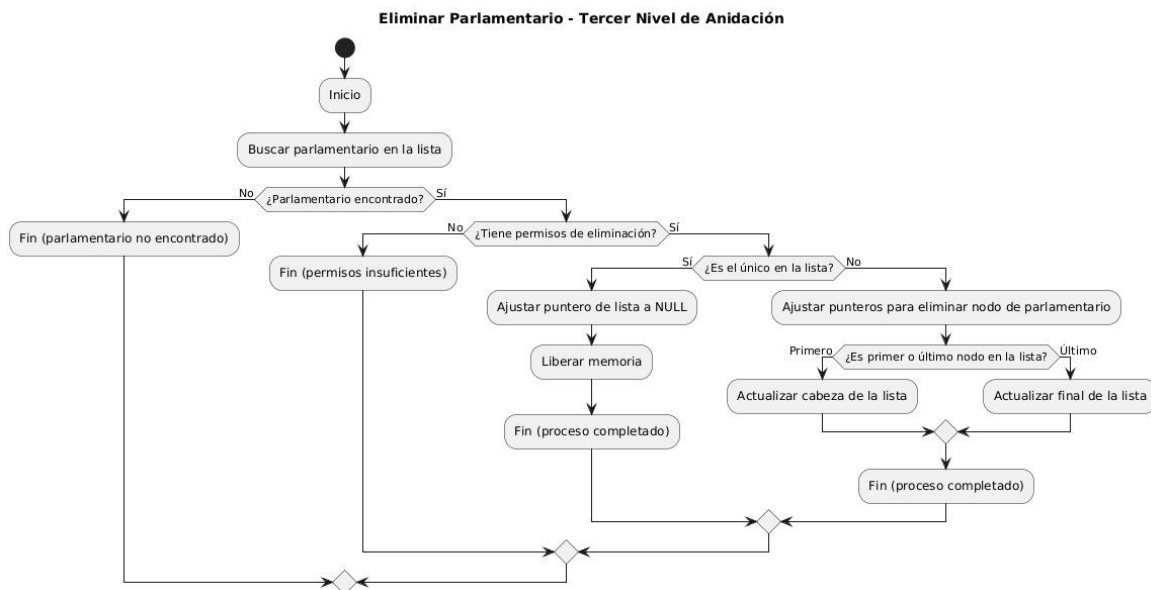


Figura 3.2 Proceso de Eliminación de Parlamentario.

Gestión de Proyectos de Ley: El núcleo del sistema es el manejo de proyectos de ley, los cuales se organizan en un Árbol Binario de Búsqueda (ABB) según su ID. Esto permite búsquedas rápidas y asegura un acceso ordenado a los proyectos. La función `agregarProyectoLey` inserta un proyecto en el ABB, y `buscarProyectoLey` facilita su recuperación. El siguiente diagrama muestra el proceso de inserción de un proyecto en el ABB:

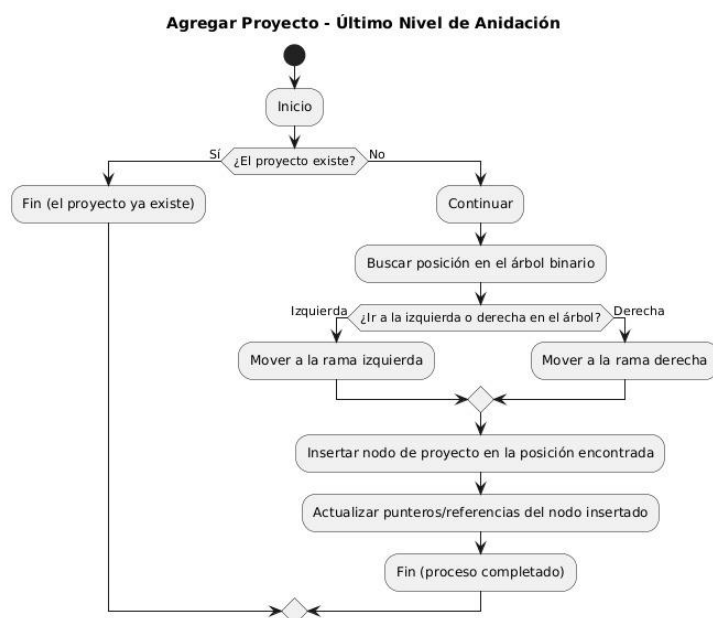


Figura 3.3 Proceso de Agregar Proyecto de Ley.

Votación y Asignación: Este módulo se encarga de la votación en ambas cámaras del Congreso y de la asignación de los proyectos a la cámara de origen (Diputados o Senadores) en función de su temática. La función `asignarCamaraOrigen` distribuye los proyectos a la cámara adecuada, y `votacionEnProceso` recopila y almacena los votos de cada parlamentario. Este flujo está representado en el siguiente diagrama de votación legislativa:

Funcionalidades Adicionales: Además de las operaciones principales, el sistema incluye funciones adicionales como `getProyectoMasVotado`, que permite identificar el proyecto con mayor apoyo, y `eliminarParlamentario`, que gestiona la baja de parlamentarios cuando sea necesario.

3.2 Diagrama de Estructuras de Datos

Las estructuras de datos seleccionadas y sus interacciones se representan en un diagrama general que muestra cómo los ciudadanos, parlamentarios y proyectos de ley están organizados en listas enlazadas y ABB. Esta visualización permite entender cómo se integran los diferentes elementos en el flujo del sistema:

3.3 Flujo de Trabajo del Sistema

El sistema sigue un flujo de trabajo que se basa en las etapas del proceso legislativo, con las siguientes fases:

Creación de Proyecto de Ley: Este proceso verifica las firmas si el proyecto proviene de ciudadanos y lo almacena en el ABB mediante `ingresarProyectoLey`. El flujo de trabajo desde la creación hasta la inserción en el ABB está representado en el siguiente diagrama:

Asignación a la Cámara de Origen: La asignación se realiza de acuerdo con el tema del proyecto, determinando si será discutido en la Cámara de Diputados o en el Senado. Esta asignación está representada en el flujo de trabajo general del sistema.

Proceso de Votación: El proyecto se somete a votación en ambas cámaras, siguiendo un proceso secuencial de votación en la cámara de origen y luego en la revisora, si es aprobado. Este proceso se ilustra en el diagrama de votación:

Funcionalidades Extra: El sistema también incluye funciones adicionales para mejorar el control y análisis legislativo.

3.4 Diagrama de Flujo

En el Anexo A del presente informe se adjunta un diagrama de flujo completo del proceso legislativo en el sistema, desde la entrada de datos de los participantes hasta la publicación final de la ley, pasando por cada etapa relevante en la validación y votación de un proyecto de ley.

4. Implementación Técnica

La implementación del sistema se realizó en lenguaje ANSI C, estructurando el código en módulos separados para cada componente, lo cual permite una gestión eficiente y un mantenimiento fácil. A continuación, se describe la implementación de las estructuras y funcionalidades principales.

4.1 Estructuras de Datos

Cada estructura de datos fue diseñada para cumplir con los requisitos funcionales del sistema:

Lista Enlazada Circular para Ciudadanos: La estructura `nodoPersona` facilita el recorrido y el conteo de firmas de ciudadanos. Se utiliza para validar proyectos de ley de origen ciudadano, con funciones como `contarFirmasPersonas` y `agregarNodoPersona`.

Lista Enlazada para Parlamentarios: La estructura `nodoParlamentario` organiza a los parlamentarios y permite registrar sus votos, almacenándolos en un árbol binario. Las funciones `agregarParlamentario` y `eliminarParlamentario` gestionan los registros de parlamentarios y facilitan su manipulación dinámica.

Árbol Binario de Búsqueda (ABB) para Proyectos de Ley: La estructura `nodoProyectoLey` permite organizar y acceder rápidamente a los proyectos por su ID. Las funciones `agregarProyectoLey` y `buscarProyectoLey` insertan y localizan proyectos en el ABB de manera eficiente.

4.2 Funcionalidades Principales

Las funcionalidades se implementaron de la siguiente manera:

Creación y Gestión de Proyectos de Ley: La función `crearProyectoLey` inicializa los datos de un proyecto, mientras que `ingresarProyectoLey` permite la creación de proyectos a través de un menú interactivo.

Verificación de Firmas Ciudadanas: `contarFirmasPersonas` recorre la lista de ciudadanos para verificar las firmas cuando un proyecto es de origen ciudadano.

Votación y Asignación: `votacionEnProceso` gestiona la votación en ambas cámaras, permitiendo registrar los votos y calcular el quórum necesario.

Funcionalidades Adicionales: `getProyectoMasVotado` y `eliminarParlamentario` mejoran el análisis y la administración del sistema.

4.3 Modularización y Organización del Código

Cada módulo del sistema se organizó de manera independiente para facilitar el desarrollo y la comprensión. A continuación, se presenta un desglose de cada módulo y su función en el sistema.

5. Resultados y Conclusiones

5.1 Resultados Obtenidos

El sistema implementado cumple con los objetivos establecidos y presenta los siguientes resultados:

1. Gestión de Proyectos de Ley: La estructura ABB permite una gestión eficiente de los proyectos, optimizando la inserción y búsqueda.
2. Verificación de Firmas: La lista circular facilita el conteo de firmas para validar los proyectos creados por ciudadanos.
3. Gestión de Votación: La votación en ambas cámaras se realiza de manera eficiente, registrando y evaluando los votos.
4. Funcionalidades Adicionales: La identificación del proyecto más votado y la eliminación dinámica de parlamentarios mejoran la funcionalidad del sistema.

5.2 Menú Principal del Sistema

El menú principal del sistema proporciona al usuario acceso directo a las diferentes funcionalidades para gestionar el proceso legislativo simulado de Chile. A continuación, se detallan cada una de las opciones disponibles en el menú, con una explicación de su propósito y función:

1) Menú Agregar Proyecto de Ley:

Permite al usuario crear un nuevo proyecto de ley. Al seleccionar esta opción, se abre un submenú en el que se solicita al usuario ingresar detalles específicos del proyecto, tales como su nombre, descripción, nivel de urgencia, y los artículos correspondientes. Esta opción también verifica las firmas requeridas si el proyecto es propuesto por ciudadanos.

2) Menú Parlamentario:

En este submenú, el usuario puede gestionar la información de los parlamentarios, incluyendo su registro y eliminación. Esta opción también permite registrar y consultar los votos realizados por cada parlamentario en los proyectos de ley asignados.

3) Menú Agregar Ciudadano:

Esta opción está destinada a la gestión de ciudadanos. Permite al usuario agregar ciudadanos al sistema y administrar sus firmas, lo cual es relevante para los proyectos de ley que se originan en la ciudadanía, ya que requieren un número mínimo de firmas para ser válidos.

4) Menú Clientes:

Aunque su funcionalidad específica no se detalla en el código, esta opción podría estar diseñada para gestionar perfiles de usuarios o clientes que interactúan con el sistema. Esto puede incluir el seguimiento de datos adicionales o información sobre usuarios que apoyan los proyectos de ley.

5) Menú Votación Proyecto de Ley:

Este submenú permite someter un proyecto de ley a votación en ambas cámaras del Congreso (Diputados y Senado). El sistema registra cada voto y verifica si se alcanza el quórum necesario para la aprobación o el rechazo del proyecto.

6) Introducir presidente:

En esta opción, el usuario puede registrar o actualizar la información del presidente dentro del sistema. Esta es una función importante, ya que el presidente juega un rol crítico en la revisión final de los proyectos de ley aprobados por el Congreso.

7) Salir:

Al seleccionar esta opción, el sistema cierra el menú principal y finaliza la ejecución del programa.

El menú principal es interactivo y solicita al usuario que ingrese el número correspondiente a la opción que desea utilizar. Cada opción redirige al usuario a un submenú o función específica, que se encarga de realizar las acciones detalladas anteriormente. El diseño de este menú facilita la navegación a través de las distintas funcionalidades del sistema, permitiendo un flujo de trabajo organizado y lógico en la simulación del proceso legislativo.

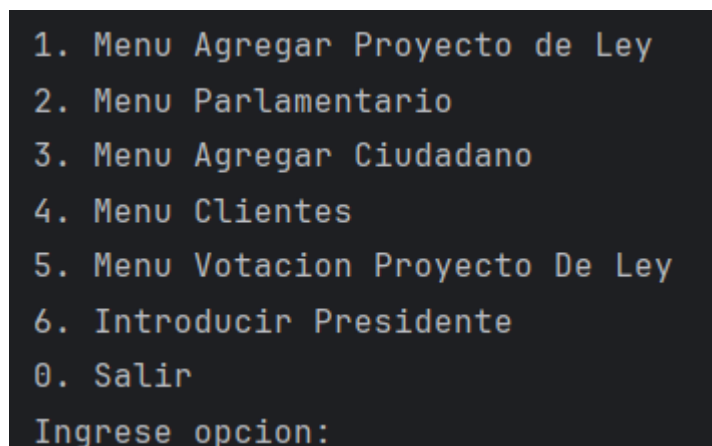


Figura 5.1 Menú Programa

5.3 Conclusión

Este informe documenta la implementación de un sistema para la gestión parlamentaria hecha con estructuras de datos en C. El programa permite representar de forma estructurada y eficiente, el proceso legislativo, cubriendo desde la creación de los proyectos de ley hasta su votación final en las cámaras del congreso, el sistema tiene un enfoque modular con el uso de estructuras de datos, como las listas enlazadas y árboles binarios, el sistema en C logra con eficiencia organizar y manipular una gran cantidad de datos de una manera eficiente, dando así una facilidad en el acceso como la modificación de los datos del proceso legislativo.

Uno de los logros significativos de este proyecto es la capacidad para gestionar datos dinámicos que tiene el sistema, como son los ciudadanos y parlamentarios, y organizar así eficientemente los proyectos de ley. El diseño modular del programa no solo permite que cada componente del sistema funcione de manera independiente, sino que también facilita el futuro mantenimiento y la posibilidad de futuras expansiones. Esta arquitectura proporciona una base sólida para implementar funciones adicionales, como consultas avanzadas y análisis estadísticos, que podrían enriquecer el sistema y ofrecer una mirada más detallada sobre el proceso legislativo. Estos elementos podrían ser especialmente útiles en investigaciones sobre el comportamiento parlamentario o en proyectos educativos que buscan enseñar el proceso legislativo de manera interactiva.

El proyecto también presentó varios desafíos, particularmente en la gestión de referencias y punteros en estructuras de datos complejas. La implementación de algoritmos de búsqueda y eliminación en el ABB y las listas enlazadas requirió un análisis cuidadoso y una planificación detallada. A través de este proceso, se fortalecieron las habilidades en programación y diseño de estructuras de datos, así como en la organización y gestión de proyectos de desarrollo de software.

En términos de aprendizaje, este proyecto no solo proporcionó una comprensión profunda de la implementación de estructuras de datos dinámicas en un contexto legislativo, sino que también destacó la importancia de un diseño modular y escalable en sistemas complejos. Además, el trabajo en equipo y la colaboración resultaron esenciales para lograr un sistema robusto, donde cada miembro contribuyó al desarrollo de componentes específicos, asegurando así un producto final coherente y funcional.

A futuro, este sistema podría ampliarse con una interfaz gráfica o herramientas de visualización de datos que lo hagan más accesible y comprensible para usuarios no técnicos. También se podrían añadir módulos de análisis legislativo que ofrezcan estadísticas y tendencias basadas en los datos acumulados, lo cual enriquecería el sistema como herramienta de análisis para investigadores y legisladores.

En conclusión, este proyecto sienta una base sólida para futuras investigaciones y aplicaciones en el ámbito de procesos legislativos y demuestra la versatilidad de las estructuras de datos avanzadas en la organización de datos complejos. Con estas características, el sistema puede convertirse en un recurso valioso tanto en el ámbito académico como en posibles aplicaciones prácticas dentro de entornos legislativos

6. Bibliografía

- [1] [BCN, 23] Biblioteca del Congreso Nacional de Chile, Guía de Formación Cívica. Disponible vía web en https://www.bcn.cl/formacioncivica/detalle_guia?h=10221.3/45763. Revisada por última vez el 5 de noviembre de 2024.

- [2] [MinSegPres, 15] Ministerio Secretaría General de la Presidencia de Chile, Cómo se hace una ley. Disponible vía web en https://www.minsegpres.gob.cl/wp-content/uploads/2015/01/como_se_hace_una_ley.pdf. Revisada por última vez el 5 de noviembre de 2024.

- [3] [Wikipedia, 24] Wikipedia, Procedimiento legislativo en Chile. Disponible vía web en https://es.wikipedia.org/wiki/Procedimiento_legislativo_en_Chile. Revisada por última vez el 5 de noviembre de 2024.

Anexos

A Diagrama de Flujo del Proceso Legislativo

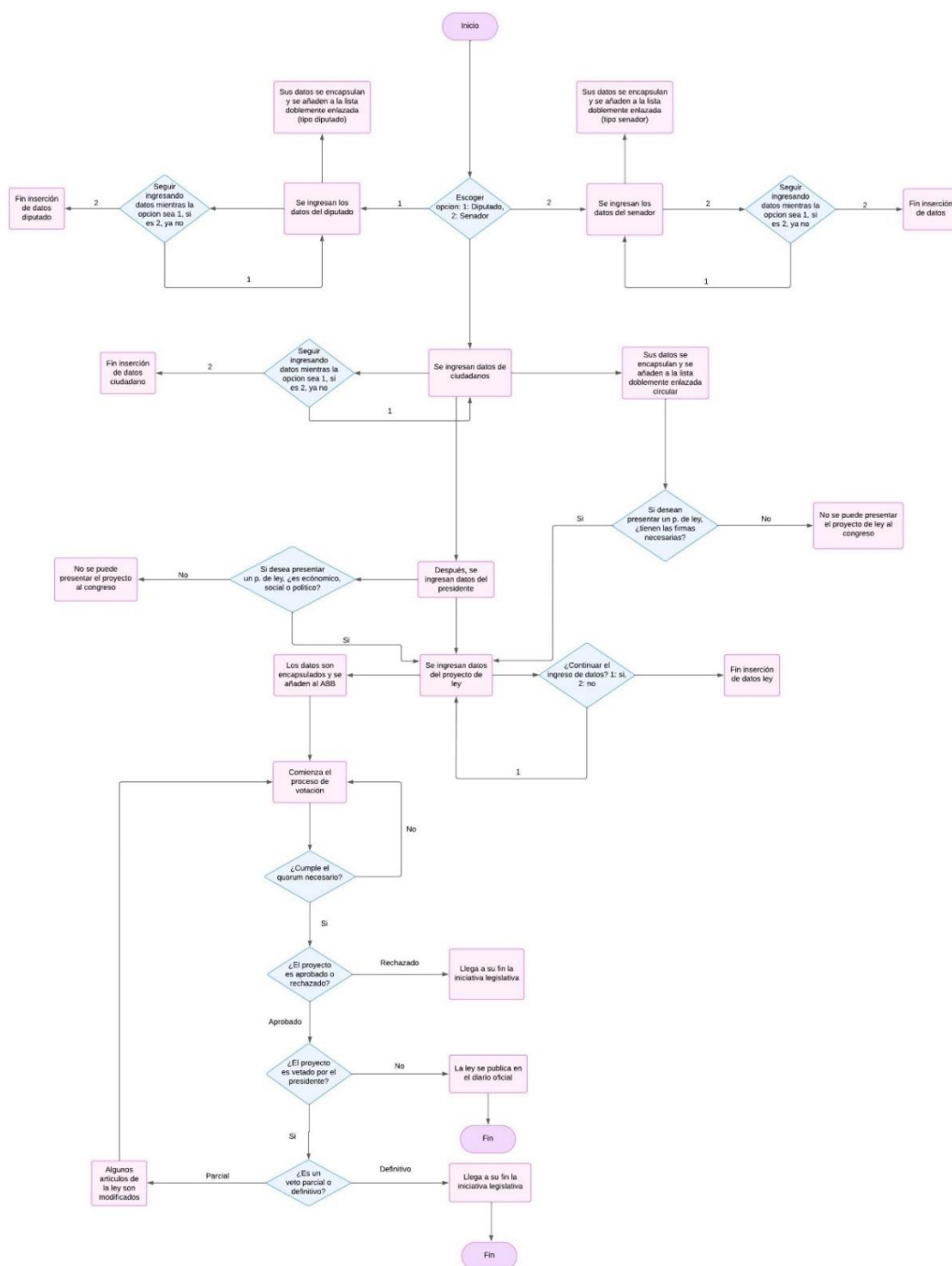


Figura A.1 Diagrama de flujo

B Carta Gantt

En este anexo se presentarán 2 figuras correspondientes a la Carta Gantt Grupal.

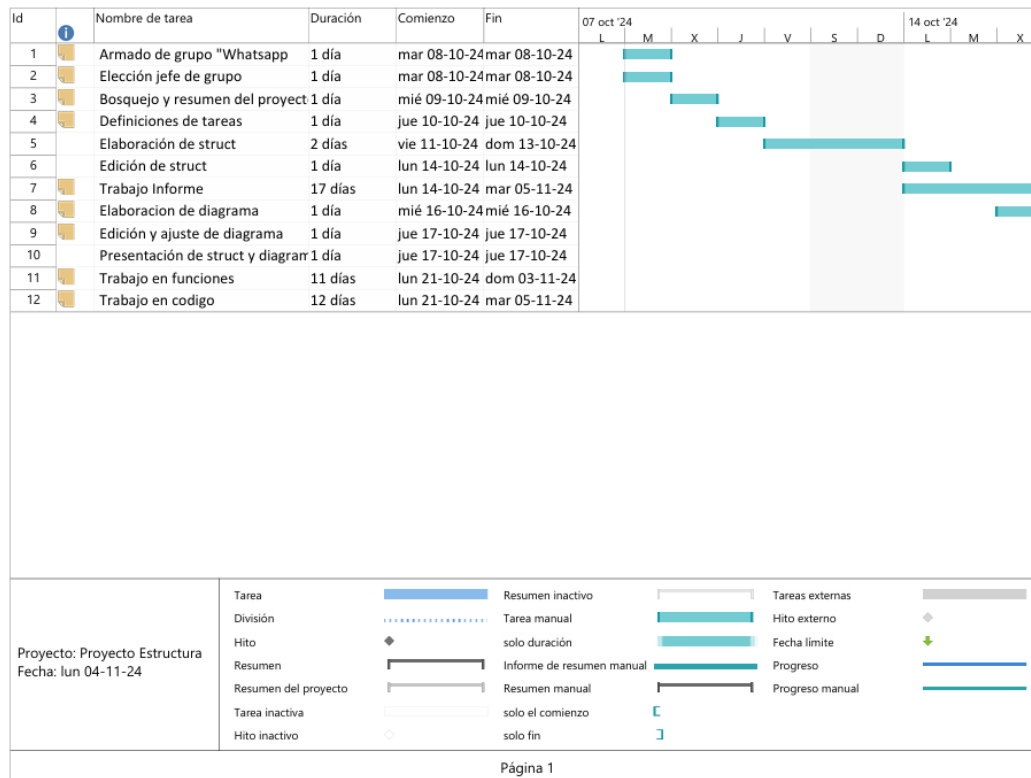


Figura B.1 Carta Gantt Grupal Página 1

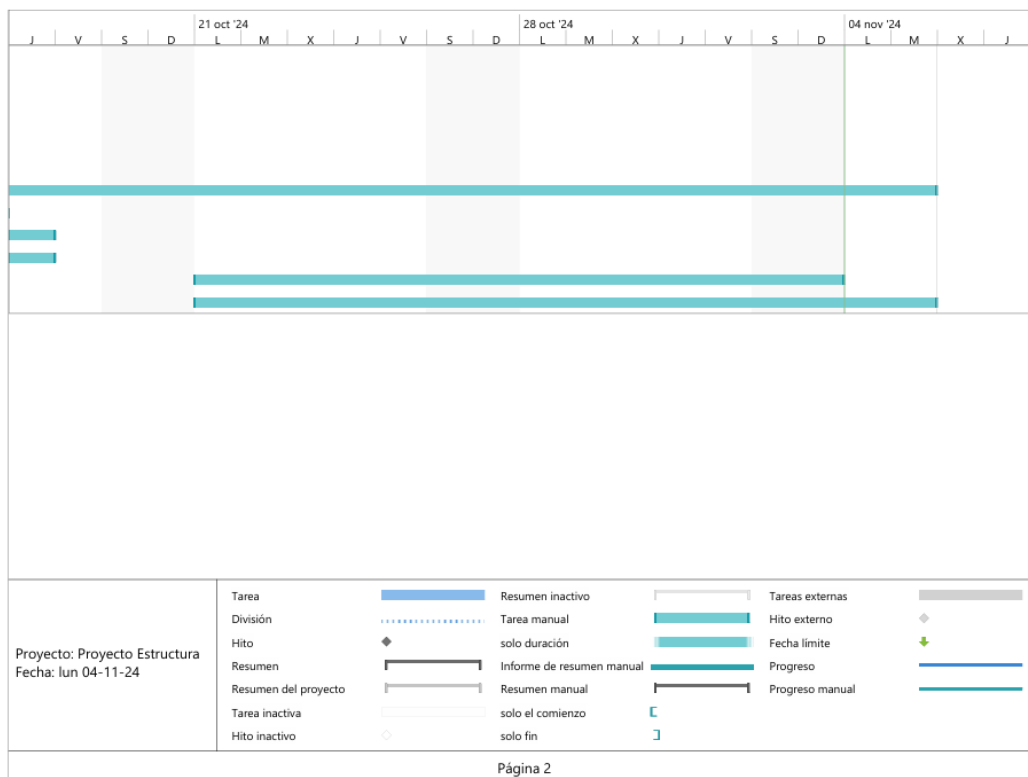


Figura B.2 Carta Gantt Grupal Página 2

En este anexo se presentarán 8 figuras correspondientes a la Carta Gantt de cada integrante del grupo.

Carta Gantt Nicolas Gonzalez

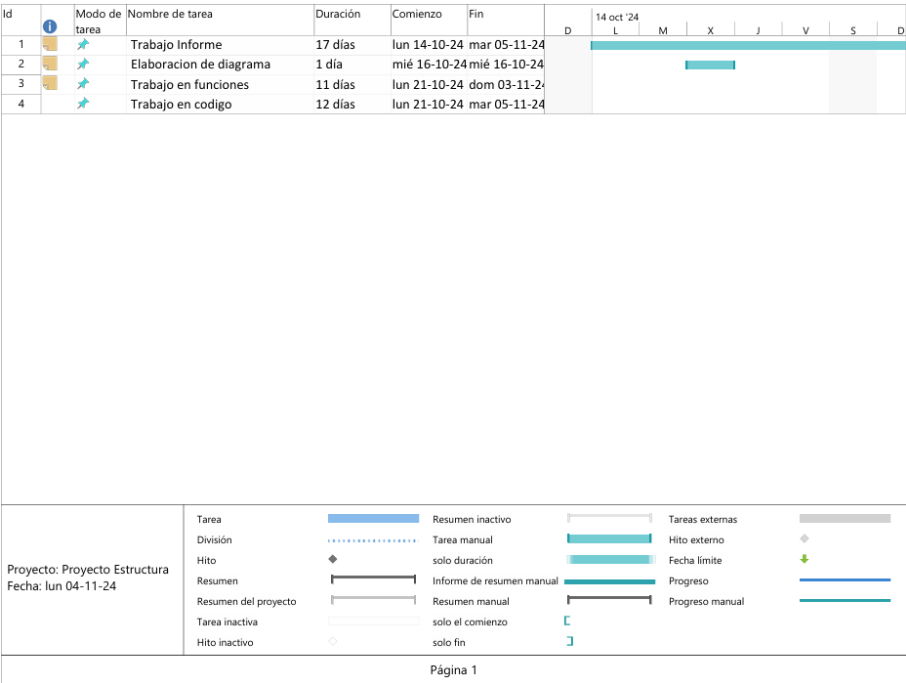


Figura B.3 Carta Gantt Nicolas Gonzalez Página 1

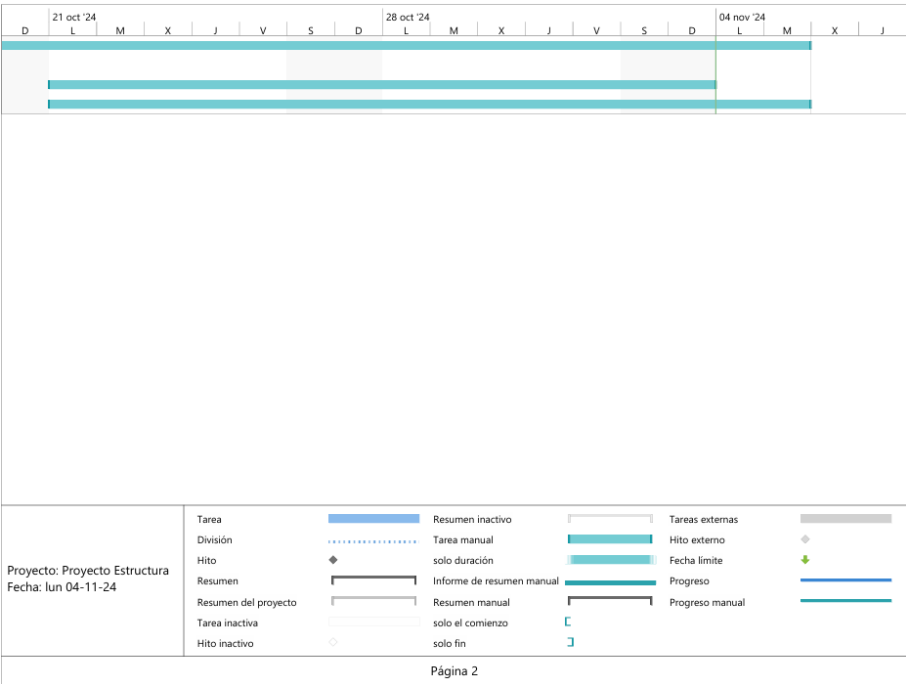


Figura B.4 Carta Gantt Nicolas Gonzalez Página 2

me Carta Gantt Ricardo Vergara

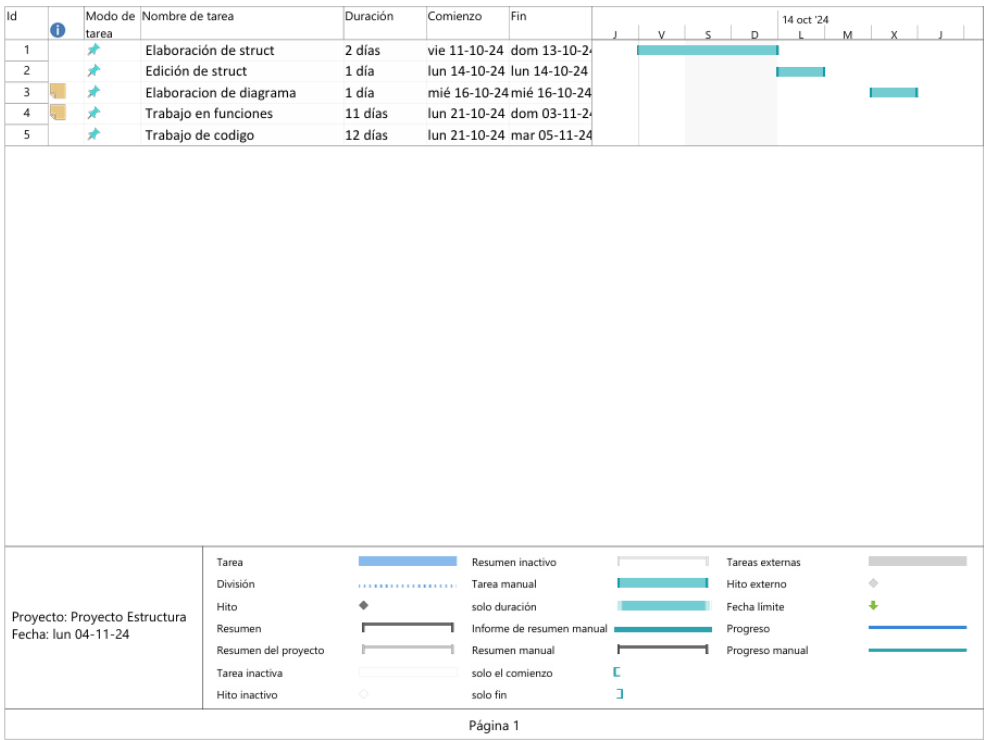


Figura B.5 Carta Gantt Ricardo Vergara Página 1

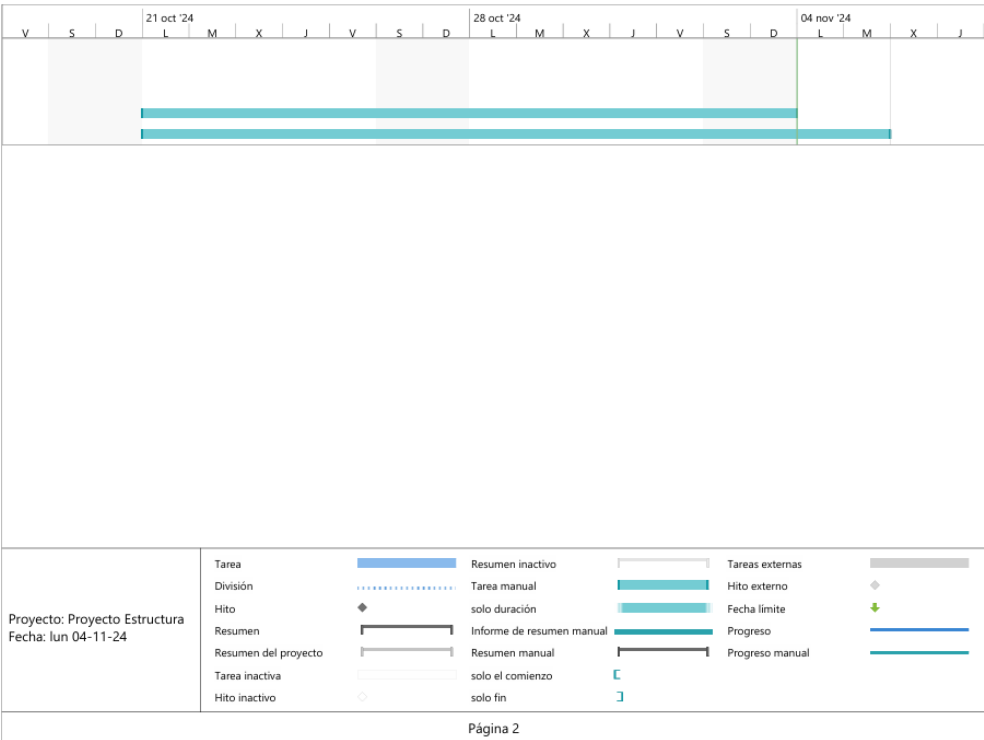


Figura B.6 Carta Gantt Ricardo Vergara Página 2

Carta Gantt Byron Vásquez

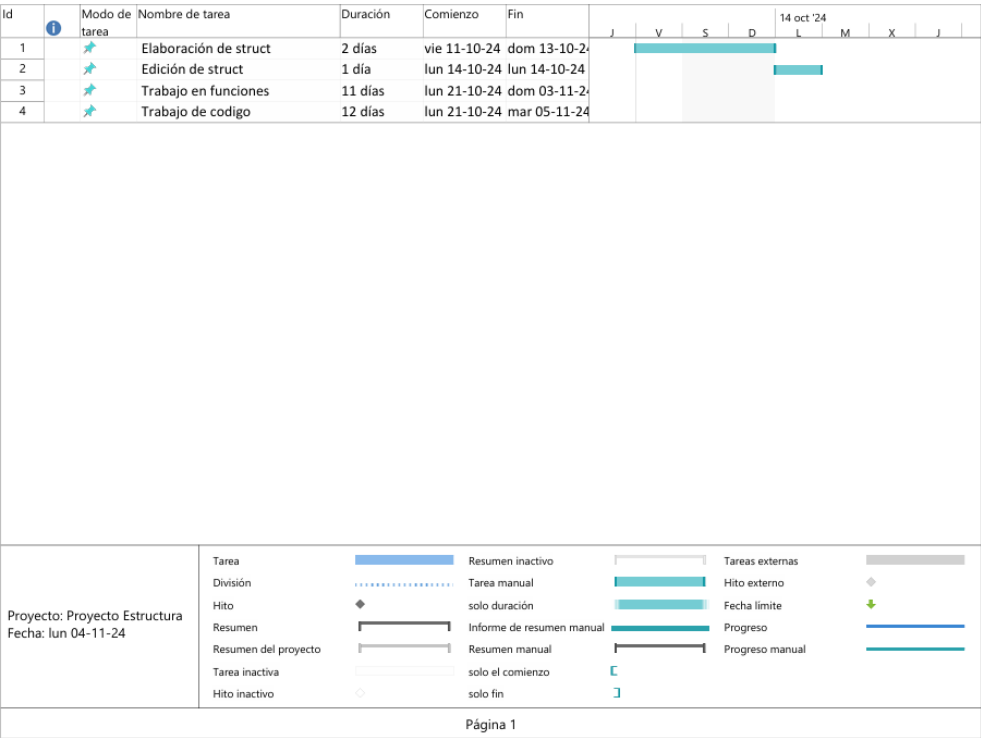


Figura B.7 Carta Gantt Byron Vásquez Página 1

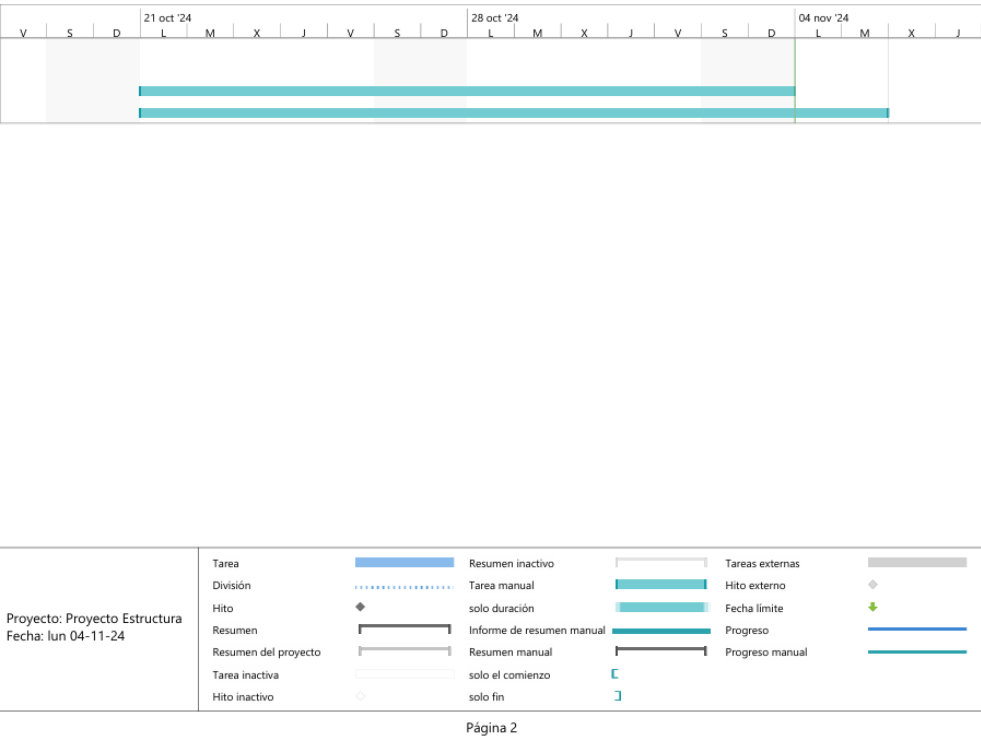


Figura B.8 Carta Gantt Byron Vásquez Página 2

Carta Gantt Francisco Reyes

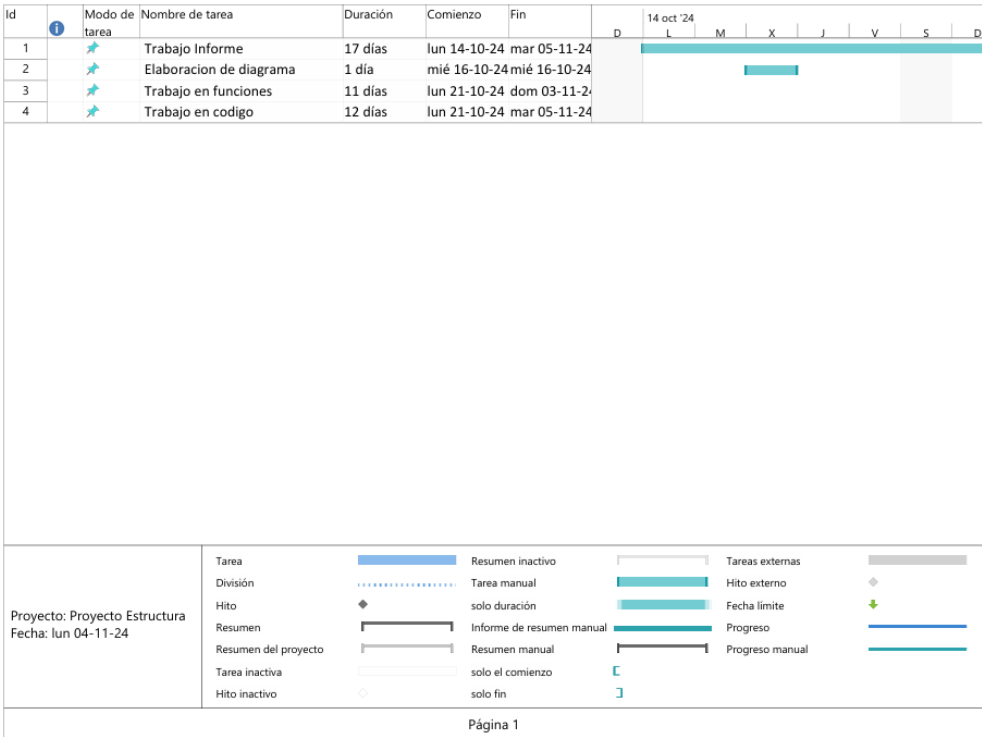


Figura A.9 Carta Gantt Francisco Reyes Página 1

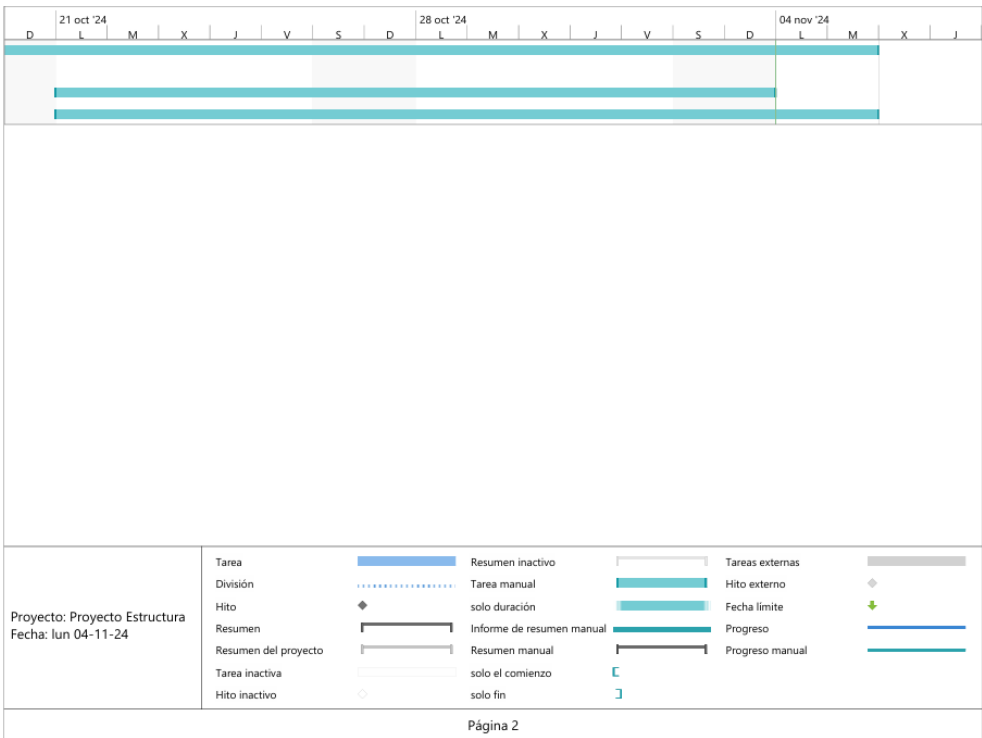


Figura B.10 Carta Gantt Francisco Reyes Página 2

RICARDO VERGARA GODOY

INGENIERO INFORMÁTICO

Profesional en la ingeniería informática, empeñado en brindar soluciones a la vida que requieran el uso de software. Capacitado para trabajar en equipo asumiendo roles de liderazgo e incentivando a la cooperación en consecución de una solución factible y resolver un problema determinado. Y asimismo, cumplir su propósito como licenciado en la ingeniería.



+56 9 9455 4321



richi345bg@gmail.com



La Calera

EDUCACIÓN

2023 - Actualmente

**Pontificia Universidad
Católica Valparaíso**

Ingeniería Informática

HABILIDADES

- Desarrollo de programas
- Manejo en sistemas operativos
- Manejo integral de Redes Sociales
- Idioma Inglés B1

Figura C.1 Curriculum Ricardo Vergara Godoy

NICOLAS GONZALEZ URRUTIA



+56 9 7242 5007



nico.gonzalez2203@gmail.com



San Antonio, Quinta Region

INGENIERO INFORMÁTICO

Profesional en ingeniería informática, dedicado a desarrollar soluciones prácticas y eficientes mediante el uso de software para mejorar diversos aspectos de la vida diaria y profesional. Poseo una sólida formación en trabajo colaborativo, lo que me permite asumir roles de liderazgo cuando es necesario, promoviendo un ambiente de cooperación entre los miembros del equipo. Mi enfoque está en guiar a los colaboradores hacia el logro de soluciones viables y eficaces para problemas específicos, contribuyendo así al cumplimiento de los objetivos generales del proyecto. Comprometido con el propósito y responsabilidad de mi carrera como ingeniero, busco siempre aplicar mis conocimientos de manera ética y eficaz para aportar valor a cada iniciativa.

EDUCACIÓN

2023 - Actualmente

**Pontificia Universidad
Católica Valparaíso**

Ingeniería Informática

HABILIDADES

- Desarrollo de programas
- Manejo en sistemas operativos
- Mantenimiento de hardware
- Idioma Inglés C1, Español nativo

Figura C.2 Curriculum Nicolas Gonzalez Urrutia

BYRON VASQUEZ AGUILERA



+56 9 66727894



byronvasquez430@gmail.com



valparaiso, Quinta Region

INGENIERO INFORMÁTICO

Ingeniero en informática con experiencia en el desarrollo de soluciones prácticas y efectivas mediante software, orientadas a mejorar tanto el ámbito personal como profesional. Cuento con una sólida base en trabajo en equipo, lo cual me permite asumir roles de liderazgo cuando es necesario y fomentar la colaboración entre los miembros del grupo. Mi enfoque se centra en guiar a los colaboradores hacia la implementación de soluciones viables y efectivas para problemas específicos, contribuyendo al logro de los objetivos generales del proyecto. Comprometido con la ética y la responsabilidad de mi profesión, busco aplicar mis conocimientos de manera eficaz para aportar valor a cada iniciativa en la que participo.

EDUCACIÓN

2023 - Actualmente

**Pontificia Universidad
Católica Valparaíso**

Ingeniería Informática


HABILIDADES


- Desarrollo de programas
- Manejo en sistemas operativos
- Manejo integral de Redes Sociales
- Mantenimiento de hardware
- Idioma Ingles B1, Español nativo y Frances A1

Figura C.3 Curriculum Byron Vasquez Aguilera

FRANCISCO REYES VILLALÓN

 +56 9 8470 6932

 fco.reyesx@gmail.com

 Ibsen #107 C° Delicias,
Valparaíso

INGENIERO INFORMÁTICO

Profesional en informática con experiencia en desarrollo de software y soluciones tecnológicas, buscando aportar habilidades en programación, análisis de sistemas y gestión de proyectos para optimizar procesos y mejorar la eficiencia tecnológica. Interesado en oportunidades que impulsen la innovación y el crecimiento en entornos colaborativos y dinámicos.

EDUCACIÓN

2023 - Actualmente

**Pontificia Universidad
Católica Valparaíso**

Ingeniería Informática

HABILIDADES

- Manejo de electrónica
- Manejo en sistemas operativos
- Manejo integral de Redes Sociales
- Idioma Inglés Básico

EXPERIENCIA LABORAL

Técnico Informático

Pontificia Universidad Católica Valparaíso

Unidad DSIC - Brindar soporte a funcionarios, alumnos y profesores en reparación y/o asesoría de dispositivos tecnológicos.

Figura C.4 Curriculum Francisco Reyes Villalón

D Código Fuente

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


#define minFirmas 3000

#define MaxCantidadArticulos 100


struct Estado {

    struct Ciudadanos *ciudadanos;

    struct CongresoNacional *congreso;

    struct Presidente *presidente;

};


struct Persona {

    char *nombreApellido;

    char *rut;

    int edad;

    int haFirmado;

};


struct Articulo {

    int num;

    char *descripcion;
```

```
};
```

```
struct nodoPersona { // Lista doblemente enlazada circular

    struct Persona *datosPersona;

    struct nodoPersona *sig, *ant;

};
```

```
struct Parlamentario {

    struct Persona *datos;

    struct nodoVoto *votos;

    int idParlamentario;

    char *periodo;

    char *partidoPolitico;

    char *especialidad;

};
```

```
struct nodoParlamentario { // Lista doblemente enlazada

    struct Parlamentario *datosParlamentario;

    struct nodoParlamentario *sig, *ant;

};
```

```
struct Presidente {

    struct Persona *datosPresidente;

    struct nodoProyectoLey *proyectoLey;
```



```
};
```

```
struct ProyectoLey {  
  
    struct CongresoNacional *congreso;  
  
    struct Comision *comision;  
  
    int id;  
  
    int urgencia;  
  
    char *tipo;  
  
    char *nombreProyecto;  
  
    char *descripcion;  
  
    char *camaraOrigen;  
  
    struct Articulo **articulos;  
  
    int votado;  
  
    char *fecha;  
  
    char *fechaEntradaVigencia;  
  
};
```

```
struct nodoProyectoLey { // Es un ABB  
  
    struct ProyectoLey *datosProyecto;  
  
    struct nodoProyectoLey *izq, *der;  
  
};
```

```
struct CongresoNacional {  
  
    struct nodoProyectoLey *ley;
```

```

    struct nodoParlamentario *diputados;

    struct nodoParlamentario *senadores;

    int numDiputados;

    int numSenadores;

};

struct Comision {

    struct nodoParlamentario *miembros;

    int idComision;

    char *tipo; // Si es de hacienda, educacion, etc

    char *tipoComision; // Si es mixta o no

};

struct Ciudadanos {

    struct nodoPersona *personas;

};

struct Voto {

    struct ProyectoLey *proyecto;

    int votado;

};

struct nodoVoto {

```

```

    struct Voto *datosVoto;

    struct nodoVoto *izq, *der;

};

struct Persona *crearPersona(char *nombre, char *Rut, int edad, int firma) {

    struct Persona *nuevaPersona;

    nuevaPersona = (struct Persona *) malloc(sizeof(struct Persona));

    nuevaPersona->nombreApellido = (char *)
    malloc(strlen(nombre)*sizeof(char));

    strcpy(nuevaPersona->nombreApellido, nombre);

    nuevaPersona->rut = (char *) malloc(strlen(Rut)*sizeof(char));

    strcpy(nuevaPersona->rut, Rut);

    nuevaPersona->edad = edad;

    nuevaPersona->haFirmado = firma;

    return nuevaPersona;
}

struct nodoPersona *crearNodoPersona(struct Persona *datosPersona) {

    struct nodoPersona *nuevoNodo;

    nuevoNodo = (struct nodoPersona *) malloc(sizeof(struct nodoPersona));

```

```

    nuevoNode->datosPersona = datosPersona;

    nuevoNode->ant = NULL;

    nuevoNode->sig = NULL;

    return nuevoNode;
}

int agregarNodePersona(struct nodePersona **head, struct Persona *nueva) {

    struct nodePersona *nuevoNode, *rec;

    if(nueva==NULL) {

        return 0;

    }

    nuevoNode = crearNodePersona(nueva);

    if (*head == NULL) {

        *head = nuevoNode;

        nuevoNode->sig = nuevoNode;

        nuevoNode->ant = nuevoNode;

    }else {

        rec = *head;

        do{

            rec=rec->sig;

        }while (rec->sig != *head);

```

```

        rec->sig = nuevoNodo;

        nuevoNodo->ant = rec;

        nuevoNodo->sig = *head;

        (*head)->ant = nuevoNodo;

    }

    return 1;

}

```

```

struct Parlamentario *crearParlamentario(struct Persona *datos, int id,
char *periodo ,char *partido ,char *especialidad){

    struct Parlamentario *nuevoParlamentario;

    nuevoParlamentario =(struct Parlamentario *) malloc(sizeof(struct
Parlamentario));

    nuevoParlamentario->datos = datos;

    nuevoParlamentario->idParlamentario = id;

    nuevoParlamentario->periodo =(char *)
malloc(strlen(periodo)*sizeof(char));

    strcpy(nuevoParlamentario->periodo,periodo);

    nuevoParlamentario->partidoPolitico = (char *)
malloc(strlen(partido)*sizeof(char));

    strcpy(nuevoParlamentario->partidoPolitico,partido);

```

```

    nuevoParlamentario->especialidad = (char *)
    malloc(strlen(especialidad)*sizeof(char));

    strcpy(nuevoParlamentario->especialidad,especialidad);

    nuevoParlamentario->votos = NULL;


    return nuevoParlamentario;

}


struct nodoParlamentario *crearNodoParlamentario(struct Parlamentario
*datosParlamentario) {

    struct nodoParlamentario *nuevoNodo;


    nuevoNodo = (struct nodoParlamentario *) malloc(sizeof(struct
nodoParlamentario));

    nuevoNodo->datosParlamentario = datosParlamentario;

    nuevoNodo->ant = NULL;

    nuevoNodo->sig = NULL;


    return nuevoNodo;

}


int agregarParlamentario(struct nodoParlamentario **head, struct
Parlamentario *nuevo,int *numParlamentario) {

    struct nodoParlamentario *nuevoNodo ;

    struct nodoParlamentario *rec ;

    if(nuevo == NULL) {

```

```

        return 0;

    }

    nuevoNodo= crearNodoParlamentario(nuevo);

    if (*head == NULL) {

        *head = nuevoNodo;

        (*numParlamentario)++;

        return 1;

    }

    rec = *head;

    while (rec->sig != NULL) {

        if (rec->datosParlamentario->idParlamentario == nuevo-
>idParlamentario || strcmp(rec->datosParlamentario->datos->rut, nuevo-
>datos->rut) == 0) {

            return 0;

        }

        rec = rec->sig;

    }

    rec->sig = nuevoNodo;

    nuevoNodo->ant = rec;

    (*numParlamentario)++;

    return 1;

}

```

```

struct ProyectoLey *crearProyectoLey(int id, char *tipo, int urgencia, char
*nombre, char *descripcion, struct Artículo **articulos,char *fechaIngreso)
{

    struct ProyectoLey *nuevoProyecto ;

    nuevoProyecto = (struct ProyectoLey *) malloc(sizeof(struct
ProyectoLey));

    nuevoProyecto->id = id;

    nuevoProyecto->tipo = (char *)malloc((strlen(tipo) + 1) *
sizeof(char));

    strcpy(nuevoProyecto->tipo, tipo);

    nuevoProyecto->nombreProyecto = (char *)malloc((strlen(nombre) + 1) *
sizeof(char));

    strcpy(nuevoProyecto->nombreProyecto, nombre);

    nuevoProyecto->urgencia = urgencia;

    nuevoProyecto->descripcion = (char *)malloc((strlen(descripcion) + 1) *
sizeof(char));

    strcpy(nuevoProyecto->descripcion, descripcion);

    nuevoProyecto->camaraOrigen = (char *)malloc(2*sizeof(char));

    strcpy(nuevoProyecto->camaraOrigen , "N"); // Se le asigna N como Forma
de Decir que no Tiene Camara de Origen Todavia

```



```

    nuevoProyecto->fecha = (char *)
malloc(strlen(fechaIngreso)*sizeof(char));

    strcpy(nuevoProyecto->fecha, fechaIngreso);


    nuevoProyecto->articulos = articulos;


    nuevoProyecto->comision = NULL;


    return nuevoProyecto;
}

```

```

struct Articulo *crearArticulo(int numero, char *descripcion) {

    struct Articulo *nuevoArticulo;


    nuevoArticulo =(struct Articulo *) malloc(sizeof(struct Articulo));

    nuevoArticulo->num = numero;

    nuevoArticulo->descripcion = (char *)
malloc(strlen(descripcion)*sizeof(char));

    strcpy(nuevoArticulo->descripcion, descripcion);


    return nuevoArticulo;

}

```

```

int buscarParlamentario(struct nodoParlamentario *head,char *rutPersona,int
idParlamentario) {

    struct nodoParlamentario *rec;

    if(head!=NULL) {

        rec = head;

        while(rec!=NULL) {

            if(rec->datosParlamentario->idParlamentario == idParlamentario
&& strcmp(rec->datosParlamentario->datos->rut,rutPersona)== 0)

                return 1;

            rec=rec->sig;

        }

    }

    return 0;

}

```

```

struct nodoProyectoLey *buscarProyectoLey(struct nodoProyectoLey
*arbol,struct nodoProyectoLey **ant,int idBuscado) {

```

```

int enc=0;

while(enc==0 && arbol!=NULL) {

    if(arbol->datosProyecto->id==idBuscado)

        enc++;

    else {

        *ant = arbol;

        if(idBuscado<arbol->datosProyecto->id)

            arbol= arbol->izq;

        else

            arbol= arbol->der;

    }

}

return arbol;

}

```

```

int agregarProyectoLey(struct nodoProyectoLey **head, struct ProyectoLey
*nuevoProyecto) {

    struct nodoProyectoLey *nuevo=NULL, *ant=NULL, *ptr=NULL;

    ptr = buscarProyectoLey(*head, &ant, nuevoProyecto->id);

    if(ptr==NULL) {

        nuevo =(struct nodoProyectoLey *) malloc(sizeof(struct
nodoProyectoLey));

        nuevo->datosProyecto = nuevoProyecto;

        if(ant==NULL) {

            *head = nuevo;

        }else {

            if(nuevoProyecto->id<ant->datosProyecto->id) {

                ant->izq = nuevo;

            }else{

                ant->der = nuevo;

            }

        }

        return 1;

    }

    return 0;

}

```

```

int verificarSiProyectoExiste(struct nodoProyectoLey *raiz, int idBuscado)
{
    struct nodoProyectoLey *actual;

    if(raiz != NULL) {
        actual = raiz;

        while (actual != NULL) {
            if (actual->datosProyecto->id == idBuscado){
                return 1;
            }
            else{
                if (idBuscado < actual->datosProyecto->id){
                    actual = actual->izq;
                }
                else{
                    actual = actual->der;
                }
            }
        }
    }

    return 0;
}

```

```

int agregarSectorProyectoLey(char *sector,int opcion) {

```

```

    char sectores[10][27] = {"Educacion", "Salud", "Reforma", "Impuesto",
    "Defensa",

        "Ministerio Publico", "Seguridad Social", "Presupuesto",

        "Poder Judicial", "Cooperacion Internacional"};

    int numSectores = 10;

    if(opcion >0 && opcion <=numSectores) {

        strcpy(sector, sectores[opcion - 1]);

        return 1;

    }

    return 0;

}

int contarFirmasPersonas(struct nodoPersona *head) {

    if(head == NULL)

        return 0;

    int contadorFirmas = 0;

    struct nodoPersona *rec;

    rec = head;

    do {

        printf("El Ciudadano '%s' Firmo?\n",rec->datosPersona->nombreApellido);

        do {

            printf("1:si --- 0: no");

```

```

        scanf("%d",&rec->datosPersona->haFirmado);

        }while(rec->datosPersona->haFirmado<0 ||rec->datosPersona-
>haFirmado<1);

        if(rec->datosPersona->haFirmado == 1)

            contadorFirmas++;

        rec = rec->sig;

    } while (rec!=head);

    return contadorFirmas;

}

int agregarProyectoPersona(struct nodoProyectoLey **proyectos,struct
ProyectoLey *nuevo, struct nodoPersona *head) {

    if(head == NULL)

        return 0;

    int totalFirmas;

    totalFirmas = contarFirmasPersonas(head);

    if(totalFirmas >= minFirmas) {

        if(agregarProyectoLey(proyectos, nuevo) == 1)

```

```

        return 1;

    return 0;

}

return 0;

}

void ingresarProyectoLey(struct Estado *es, struct nodoProyectoLey **head)
{

    int idProyectoLey, opcion = 1, salida, i;

    int year, dia, mes;

    int urgencia;

    char fechaIngreso[13]="";

    char sector[27], nombreProyecto[70], descripcionProyecto[200],
numero[6];

    struct Articulo *Articulos[MaxCantidadArticulos];

    char ArticuloDescripcion[500];

    struct ProyectoLey *nuevoProyecto;

    do {

        printf("Menu de Ingreso Proyecto de Ley\n");

        printf("Opcion 1 Salir al Menu Principal\n");

        printf("Opcion 2 Agregar nuevo Proyecto De Ley\n");

        scanf("%d", &opcion);

```



```

switch (opcion) {

    case 1: {

        printf("Saliendo Del Menu de Ingreso de Proyectos de
Ley\n\n");

        break;

    }

    case 2: {

        do {

            salida = 0;

            printf("Ingrese el ID del Nuevo Proyecto de Ley (digito
mayor a 0):\n");

            scanf("%d", &idProyectoLey);

            if (verificarSiProyectoExiste(*head, idProyectoLey) ==
1) {

                printf("El proyecto que está introduciendo ya
existe\n");

                printf("Si quiere salir al menú anterior, ingrese
1: ");

                scanf("%d", &salida);

                printf("\n");

            }


```

```

1);

}while(verificarSiProyectoExiste(*head, idProyectoLey) ==

if (salida == 1) {

    break;

}

printf("Ingrese el Nombre del Proyecto de Ley:\n");

scanf(" %[^\\n]", nombreProyecto);

do {

    printf("Ingrese de que Sector es el Proyecto de Ley

\\n");

    printf("Ingrese el numero Correspondiente\\n");

    printf("1:Educacion---2:Salud---3:Reforma---4:Impuesto-
--5:Defensa---06:Ministerio Publico \\n");

    printf("7:Seguridad Social---8:Presupuesto---9:Poder
Judicial---10:Coperacion Internacional\\n");

    scanf("%d",&salida);

    if(agregarSectorProyectoLey(sector,salida)==0) {

        salida = 0;

        printf("Ingrese un Numero Valido\\n");

    }

}while(salida==0);

```

```

printf("Ingrese la Descripcion del Proyecto de Ley:\n");

scanf(" %[^\\n]", descripcionProyecto);


printf("Ingrese el nivel de urgencia (1 - baja, 2 - media,
3 - alta):\n"); // Nueva solicitud para urgencia


do {

    scanf("%d", &urgencia);

} while(urgencia!= 1 && urgencia!= 2 && urgencia!=3);

salida=0;

printf("Ingrese los Articulos que tiene el Proyecto de
Ley:\n");

for (i = 0; i < MaxCantidadArticulos && salida == 0; i++) {

    printf("Articulo [%d]: ", i+1);

    scanf(" %[^\\n]", ArticuloDescripcion);

    Articulos[i] = crearArticulo(i+1,ArticuloDescripcion);

    printf("Si desea salir, ingrese 1 si no, ingrese 0: ");

    scanf("%d", &salida);

}


salida = 0;

do {

```

```

printf("Ingrese la Fecha de ingreso del Proyecto (year-
Mes-Dia)\n");

printf("year:");

scanf("%d", &year);

printf("Mes:");

scanf("%d", &mes);

printf("Dia:");

scanf("%d", &dia);


if (year > 0 && (mes > 0 && mes < 13) && (dia > 0 &&
dia <= 31)) {

    salida = 1;

    sprintf(numero, "%d-", year);

    strcat(fechaIngreso, numero);

    sprintf(numero, "%02d-", mes);

    strcat(fechaIngreso, numero);

    sprintf(numero, "%02d", dia);

    strcat(fechaIngreso, numero);

}

else{

    printf("La fecha ingresada no es valida\n");

}


} while (salida == 0);

```

```

        nuevoProyecto = crearProyectoLey(idProyectoLey, sector,
urgencia, nombreProyecto, descripcionProyecto, Articulos, fechaIngreso);

        printf("Ingrese de Quien Propuso el proyecto De ley\n");

        printf("1:El Presidente ---- 2:La Ciudadania --- 3:Los
Parlamentarios\n");

        do {

            scanf("%d",&salida);

            if(salida<1 || salida>3) {

                printf("Ingrese un numero Valido\n\n");

            }

        }while(salida<1 || salida>3);

        if(salida == 2) {

            printf("Comprobando si tiene Suficientes Firmas");

            if(agregarProyectoPersona(&es->congreso-
>ley,nuevoProyecto,es->ciudadanos->personas)) {

                printf("Cumple con las firmas\n");

            }else {

                printf("No Cumple con las firmas\n\n");

            }

        }else {

            if(salida==1) {

                agregarProyectoLey(&es->congreso-
>ley,nuevoProyecto);

                agregarProyectoLey(&es->presidente-
>proyectoLey,nuevoProyecto);

                printf("Se agrego el Proyecto De Ley\n\n");

```

```

        }else {

            agregarProyectoLey(&es->congreso-
>ley,nuevoProyecto);

            printf("Se agrego el Proyecto De Ley\n\n");

        }

    }

    break;

}

default:

    printf("Ingrese una opción válida\n\n");

}

} while (opcion != 1);

}

void asignarCamaraOrigen(struct ProyectoLey *proyecto) {

    if(strcmp(proyecto->tipo, "Presupuesto") == 0 || strcmp(proyecto->tipo,
"Impuesto")==0 || strcmp(proyecto->tipo, "Seguridad Social") == 0

        || strcmp(proyecto->tipo, "Salud")== 0 || strcmp(proyecto->tipo,
"Educacion")==0 ){

        strcpy(proyecto->camaraOrigen ,"D"); //Asignar la camara de
Diputados

    }else {

        strcpy(proyecto->camaraOrigen ,"S");//Asignar la camara de
Senadores

    }

```

```
}
```

```
struct nodoVoto *crearNodoVoto(struct Voto *datos) {
```

```
    struct nodoVoto *nuevo;
```

```
    nuevo = (struct nodoVoto *) malloc(sizeof(struct nodoVoto));
```

```
    nuevo->datosVoto = datos;
```

```
    nuevo->izq = nuevo->der = NULL;
```

```
    return nuevo;
```

```
}
```

```
struct Voto *crearVoto(struct ProyectoLey * proyecto,int decision) {
```

```
    struct Voto *nuevo;
```

```
    nuevo =(struct Voto *) malloc(sizeof(struct Voto ));
```

```
    nuevo->proyecto = proyecto;
```

```
    nuevo->votado = decision;
```

```
    return nuevo;
```

```
}
```

```
int buscarVoto(struct nodoVoto *votos,struct ProyectoLey *proyecto) {
```

```

while(votos!=NULL) {

    if(votos->datosVoto->proyecto->id == proyecto->id)

        return votos->datosVoto->votado; // Retorna la decision del
voto si lo encuentra

    else {

        if(proyecto->id < votos->datosVoto->proyecto->id)

            votos = votos->izq; // Sigue a la izquierda si el ID del
proyecto es menor

        else

            votos = votos->der; //Sigue a la derecha si el ID del
proyecto es mayor

    }

}

return -1; // No se encontro el voto

}

```

```

int votosCamara(struct ProyectoLey *proyecto,struct nodoParlamentario
*head) {

```

```

    struct nodoParlamentario *rec;

```

```

    int contarVotosPositivos = 0;

```



```

if(head!=NULL) {

    rec= head;

    while(rec!=NULL){

        if(buscarVoto(rec->datosParlamentario->votos,proyecto)==1) {

            contarVotosPositivos++;

        }

        rec=rec->sig;

    }

}

return contarVotosPositivos;

}

```

```

int agregarVotoParlamentario(struct Parlamentario *parlamentario,struct
ProyectoLey * proyecto,int decision) {

```

```

    struct nodoVoto *nuevo,*rec;

```

```

    struct nodoVoto *padre=NULL;

```

```

    nuevo = crearNodoVoto(crearVoto(proyecto,decision));

```

```

    if(parlamentario->votos==NULL)

```

```

        parlamentario->votos = nuevo;

```

```

else {

    rec = parlamentario->votos;

    while(rec!=NULL) {

        padre = rec;

        if(rec->datosVoto->proyecto == proyecto) {

            // ya existe un voto para este proyecto,actualiza la
decision
            rec->datosVoto->votado = decision;

            return 1;

        } else {

            if(proyecto->id < rec->datosVoto->proyecto->id)

                rec= rec->izq;

            else

                rec= rec->der;

        }

    }

    // insertar el nuevo voto en el lugar correcto

    if(padre!=NULL) {

```

```

        if(proyecto->id < padre->datosVoto->proyecto->id) {

            padre->izq = nuevo;

            return 1;

        }

        else {

            padre->der = nuevo;

            return 1;

        }

    }

}

return 0;

}

void seleccionarParlamentarioMixto(struct nodoParlamentario *head,struct
Comision *comision) {

    struct nodoParlamentario *ultimo;

    struct nodoParlamentario *nuevoNodo,*rec;

    int i;

    rec = head;

    for(i=0; i<6 && rec->datosParlamentario!=NULL ;i++) {

        nuevoNodo = crearNodoParlamentario(rec->datosParlamentario);

        if(comision->miembros==NULL) {

```

```

        comision->miembros = nuevoNodo;

    }else {

        ultimo = comision->miembros;

        while(ultimo->sig!=NULL) {

            ultimo = ultimo->sig;

        }

        ultimo->sig = nuevoNodo;

        nuevoNodo->ant = ultimo;

    }

    printf("EL nuevo miembro '%s' con ID '%d'\n",nuevoNodo->datosParlamentario->datos->nombreApellido,nuevoNodo->datosParlamentario->idParlamentario);

    printf("Se unio a la comision Mixta\n");

    rec->sig = rec;

}

}

```

```

void formarComisionMixta(struct ProyectoLey *proyecto,struct
CongresoNacional *congreso) {

    struct Comision *comisionMixta;

    comisionMixta = (struct Comision *) malloc(sizeof(struct Comision));

    comisionMixta->tipoComision = "Mixta";

    comisionMixta->tipo = proyecto->tipo; //Asignar el tipo Segun el
proyecto de ley

```

```

    comisionMixta->miembros = NULL;

    //Selección de parlamentarios

    seleccionarParlamentarioMixto(congreso->diputados,comisionMixta);

    seleccionarParlamentarioMixto(congreso->senadores,comisionMixta);

    proyecto->comision = comisionMixta;

    printf("Se ha formado una Comision Mixta para el proyecto
    '%s'.\n",proyecto->nombreProyecto);

}

void asignarComision(struct ProyectoLey *proyecto,struct nodoParlamentario
*head,struct Comision *comision) {

    struct nodoParlamentario *ultimo;

    struct nodoParlamentario *nuevoNodo,*rec;

    rec = head;

    while(rec!=NULL){

        if(rec->datosParlamentario->especialidad !=NULL && strcmp(rec-
>datosParlamentario->especialidad,proyecto->tipo) ==0) {

            nuevoNodo = crearNodoParlamentario(rec->datosParlamentario);

            if(comision->miembros == NULL) {

                comision->miembros = nuevoNodo;

            }else {

                ultimo= comision->miembros;

```

```

        while(ultimo->sig!=NULL) {

            ultimo = ultimo->sig;

        }

        ultimo->sig = nuevoNodo;

        nuevoNodo->ant = ultimo;

    }

    printf("EL nuevo miembro es '%s' con ID '%d'\n",nuevoNodo->datosParlamentario->datos->nombreApellido,nuevoNodo->datosParlamentario->idParlamentario);

}

rec= rec->sig;

}

}

```

```

void agregarComision(struct ProyectoLey *proyecto, struct nodoParlamentario *head) {

```

```

    struct Comision *Nueva;

```

```

    if(head!=NULL) {

```

```

        Nueva = (struct Comision *) malloc(sizeof (struct Comision));

```

```

        Nueva->miembros = NULL;

```

```

        Nueva->tipo = (char *) malloc(strlen(proyecto->tipo)*sizeof(char));

```

```

        strcpy(Nueva->tipo,proyecto->tipo);

```

```

        printf("El proyecto es derivado a una Comision permanente");

```

```

        asignarComision(proyecto, head, Nueva);

    }

}

void votacionEnProceso(struct ProyectoLey *proyecto, struct
nodoParlamentario *head, char *camara) {

    struct nodoParlamentario *rec;

    int decision;

    printf("Comienza la Votacion del Proyecto De Ley '%s' en la Camara de
'%s'\n", proyecto->nombreProyecto, camara);

    rec = head;

    while(rec!=NULL) {

        if(strcmp(camara, "Senadores")==0) {

            printf("Senador '%s' ingrese su decision acerca del Proyecto de
Ley '%s'\n", rec->datosParlamentario->datos->nombreApellido, proyecto->
nombreProyecto);

        }else {

            printf("Diputado '%s' ingrese su decision acerca del Proyecto
de Ley '%s'\n", rec->datosParlamentario->datos->nombreApellido, proyecto->
nombreProyecto);

        }

        do {

            printf("1:A favor --- 0:abstencion --- -1:en contra\n");

            scanf("%d", &decision);

        }while(decision<-1 || decision>1);

        if(agregarVotoParlamentario(rec->
datosParlamentario, proyecto, decision)==1) {

```

```

        printf("Su voto se ha guardado");

    }

    rec = rec->sig;

}

}

int votarArticulo(struct nodoParlamentario *head) {

    struct nodoParlamentario *rec;

    int votos=0,voto;

    rec = head;

    do {

        printf("Sr(a) '%s', Ingrese su Voto 1:Aprovado---2:Rechazo\n",rec-
>datosParlamentario->datos->nombreApellido);

        do {

            scanf("%d",&voto);

        }while(voto<1 || voto>2);

        if(voto==1) {

            votos++;

        }

        rec = rec->sig;

    }while(rec!=NULL);

    return votos;

}

```



```

int modificarArticulo(struct ProyectoLey *proyecto, struct Articulo
*articulo) {

    char nuevaDescripcion[500];

    int i;

    printf("Ingrese el Articulo modificado\n");

    scanf("%s", nuevaDescripcion);

    for(i=0; i<MaxCantidadArticulos && proyecto->articulos[i]!=NULL; i++) {

        if(proyecto->articulos[i]==articulo) {

            proyecto->articulos[i]->descripcion = (char *)
malloc(strlen(nuevaDescripcion)*sizeof(char));

            return 1;

        }

    }

    return 0;

}

```

```

void eliminarArticulo(struct ProyectoLey *proyecto, int numBuscado) {

    int encontrado =0, i, k;

    for(i=0; i<MaxCantidadArticulos && proyecto->articulos[i]!=NULL; i++) {

        if(proyecto->articulos[i]->num == numBuscado) {

            encontrado =1;

            for(k=i; k<MaxCantidadArticulos-1 && proyecto-
>articulos[k]!=NULL; k++) {

                proyecto->articulos[k] = proyecto->articulos[k+1];

```

```

        }

        printf("EL Articulo fue Elminado\n");

    }

}

if(encontrado==0) {

    printf("El Articulo No pudo ser Eliminado\n");

}

}

void votacionPorArticuloModificacion(struct ProyectoLey *proyecto, struct
nodoParlamentario *head, int NumParlamentario) {

    struct Articulo *Articulos[MaxCantidadArticulos];

    int cantVotos, eleccion;

    int i, k=0;

    printf("Comienza La Votacion Por Articulo\n");

    for(i=0; i<MaxCantidadArticulos && proyecto->articulos[i]!=NULL; i++) {

        printf("El '%d'Articulo: %s", i+1, proyecto->articulos[i]-
>descripcion);

        cantVotos = votarArticulo(head);

        if((( 2*NumParlamentario) / 3) <=cantVotos) {

            printf("El '%d'Articulo Fue Aprobado\n", i+1);

        }else {

            printf("El '%d'Articulo Fue Rechazado\n", i+1);

            Articulos[k]= proyecto->articulos[i];

            k++;

```

```

    }

}

for(i=0;i<MaxCantidadArticulos && Articulos[i]!=NULL;i++) {

    printf("El '%d'Articulo Fue Rechazado Que quieren hacer con
el?\n",Articulos[i]->num);

    printf("1:Modificarlo----2:Eliminarlo\n");

    do {

        scanf("%d",&eleccion);

    }while(eleccion<1 || eleccion>2);

    if(eleccion==1) {

        if(modificarArticulo(proyecto,Articulos[i])==1) {

            printf("El Articulo Se modifiko correctamente\n");

        }else {

            printf("El Articulo no pudo ser modificado\n");

        }

    }else {

        eliminarArticulo(proyecto,Articulos[i]->num);

    }

}

}

```

```

void votarProyecto(struct ProyectoLey *proyecto, struct CongresoNacional
*congreso) {

    char camara[10];

    // Determina si el proyecto se aprueba segun la camara de origen

    if(strcmp(proyecto->camaraOrigen, "S")==0) {

        strcpy(camara, "Senadores");

        agregarComision(proyecto, congreso->senadores);

        votacionEnProceso(proyecto, congreso->senadores, camara);

        if((( 2*congreso->numSenadores ) / 3) <=
votosCamara(proyecto, congreso->senadores)){ // Votos de la Camara de
Senadores

            printf("La idea de legislar fue aprobada en el Senado.\n");

            votacionPorArticuloModificacion(proyecto, congreso-
>senadores, congreso->numSenadores);

            // Votacion en Camara Revisora (Diputados)

            strcpy(camara, "Diputados");

            votacionEnProceso(proyecto, congreso->diputados, camara);

            if((( 2*congreso->numDiputados) / 3)
<=votosCamara(proyecto, congreso->diputados)) { // Votos de la Camara de
Diputados

                printf("El proyecto '%s' ha sido aprobado por ambas
camaras.\n", proyecto->nombreProyecto);

```

```

        proyecto->votado = 1; // Marca el proyecto como votado y
aprobado

        }else {

            printf("El proyecto '%s' no alcanzo el quorum en la Camara
de Diputado.\n",proyecto->nombreProyecto);

        }

    }else {

        printf("El proyecto '%s' no alcanzo el qiorum en el
Senado.\n",proyecto->nombreProyecto);

    }

}

    strcpy(camara,"Diputados");

    agregarComision(proyecto,congreso->diputados);

    votacionEnProceso(proyecto,congreso->diputados,camara);

    if(((2*congreso->numDiputados ) /3) <=
votosCamara(proyecto,congreso->diputados)) { // Votos de la Camara de
Diputados

        printf("La idea de legislar fue aprobada en la camara de
diputados");

        votacionPorArticuloModificacion(proyecto,congreso-
>senadores,congreso->numSenadores);

```

```

strcpy(camara, "Senadores");

votacionEnProceso(proyecto, congreso->senadores, camara);

if(((2*congreso->numSenadores)/3) <=
votosCamara(proyecto, congreso->senadores)){ // Votos de la Camara de
Senadores

printf("El proyecto '%s' ha sido aprobado por ambas
camaras.\n", proyecto->nombreProyecto);

proyecto->votado = 1; //Marca el proyecto como votado y
aprobado

}else {

printf("El proyecto '%s' no alcanzo el quorum en el
Senado.\n", proyecto->nombreProyecto);

}

}else {

printf("El proyecto '%s' no alcanzo el quorum en la camara de
Diputado.\n", proyecto->nombreProyecto);

}

}

}

```

```

//Funcion extra 1

```

```

void recorrerABB(struct nodoProyectoLey *nodo, struct nodoParlamentario
*head, struct nodoProyectoLey **masVotado, int *maxVotos) {

```

```

int votosActual;

if (nodo == NULL)

    return;

recorrerABB(nodo->izq, head, masVotado, maxVotos); // In-Orden

votosActual = votosCamara(nodo->datosProyecto, head);

if (votosActual > *maxVotos) {

    *maxVotos = votosActual;

    *masVotado = nodo;

}

recorrerABB(nodo->der, head, masVotado, maxVotos);

}

struct ProyectoLey *getProyectoMasVotado(struct nodoParlamentario *head,
struct nodoProyectoLey *raiz) {

    int maxVotos = -1;

    struct nodoProyectoLey *masVotado = NULL;

    struct nodoProyectoLey *rec;

    if (head == NULL || raiz == NULL)

        return NULL;

    printf("!\n");

    rec = raiz;

```

```

    recorrerABB(rec, head, &masVotado, &maxVotos);

    if (masVotado!=NULL) {

        return masVotado->datosProyecto;

    }

    return NULL;

}

//Fin funcion extra


//Funcion extra 2

int eliminarParlamentario(struct nodoParlamentario **head, int
idParlamentario) {

    struct nodoParlamentario *rec = *head;

    if (*head == NULL)

        return 0;

    while(rec!=NULL && rec->datosParlamentario->idParlamentario !=
idParlamentario)

        rec = rec->sig;

    if(rec == NULL)

```



```

        return 0;

    if(rec == *head) {

        *head = rec->sig;

        if(*head != NULL)

            (*head)->ant = NULL;

    }

    else {

        if (rec->ant != NULL)

            rec->ant->sig = rec->sig;

        if (rec->sig != NULL)

            rec->sig->ant = rec->ant;

    }

    return 1;

}

struct ProyectoLey * proyectoVotando(struct nodoProyectoLey *raiz,int
idBuscado){

    struct nodoProyectoLey *actual;

    actual = raiz;

    while (actual != NULL) {

```

```

    if (actual->datosProyecto->id == idBuscado){

        return actual->datosProyecto;

    }

    else{

        if (idBuscado < actual->datosProyecto->id){

            actual = actual->izq;

        }

        else{

            actual = actual->der;

        }

    }

}

return NULL;

}

```

```

void mostrarProyecto(struct ProyectoLey *proyecto) {

    int i;

    printf("'s'",proyecto->nombreProyecto);

    printf("tipo: 's'\n",proyecto->tipo);

    for(i=0;i<MaxCantidadArticulos && proyecto->articulos[i]!=NULL;i++) {

        printf("Articulo[%d]: 's' \n",i+1,proyecto->articulos[i]-
>descripcion);

    }

}

```

```
}
```

```
void recorrerArbolProyectos(struct nodoProyectoLey *raiz) {

    if (raiz != NULL) {

        // Recorrido en orden (in-order): Izquierda, Raíz, Derecha

        recorrerArbolProyectos(raiz->izq);

        mostrarProyecto(raiz->datosProyecto);

        recorrerArbolProyectos(raiz->der);

    }

    return;

}
```

```
void comienzoVotacionCongreso(struct Estado *es){

    int opcion,idProyectoVotar,salida=0;

    struct ProyectoLey *proyectoVotado,*proyectoMasVotado;

    int year,dia,mes;

    char fechaEnVigencia[13]="";

    char numero[6];

    do{

        printf("Menu de Comienzo de votaciones Congreso Nacional\n");

        printf("Opcion 1: Salir al menu Principal\n");

        printf("Opcion 2: Comenzar La votacion \n");

    }
```

```

printf("Opcion 3: Mostar Los Proyectos en el Congreso\n");

printf("Opcion 4: Mostar Proyecto de Ley con mas Votos
positivos\n");

scanf("%d",&opcion);

switch (opcion){

    case 1:{

        printf("Saliendo del Menu Comienzo de votaciones Congreso
Nacional\n");

        break;

    }

    case 2:{

        if(es->congreso->numDiputados!=155 && es->congreso-
>numSenadores!=50) {

            printf("No puede Comenzar una Votacion porque Alguna de
las Camara no esta llena\n\n");

        }else {

            do{

                printf("Ingrese el ID del Proyecto De Ley a
votar\n");

                scanf("%d",&idProyectoVotar);

                if(verificarSiProyectoExiste(es->congreso-
>ley,idProyectoVotar)==0) {

                    printf("No se encontro el Proyecto de Ley\n");

                    printf("Si quiere Salir al menu Anterior
ingrese 2 si quiere seguir ingresando un ID Ingrese 0 \n");

                    scanf("%d",&salida);

```

```

        }else{

            break;

        }

    }while(salida!=2);

    if(salida==2){

        break;

    }

    proyectoVotado = proyectoVotando(es->congreso-
>ley,idProyectoVotar);

    printf("Comienza La Votacion del Proyecto con el ID :
'%d'\n",idProyectoVotar);

    votarProyecto(proyectoVotado,es->congreso);

    if(proyectoVotado->votado==1){

        salida = 0;

        printf("El Proyecto de Ley es enviado al presidente
de la republica '%s'\n",es->presidente->datosPresidente->nombreApellido);

        printf("El proyecto '%s' Fue Aprobado o Vetado por
el Presidente?\n",proyectoVotado->nombreProyecto);

        printf("Ingrese el numero correspondiente\n");

        printf("1:Aprobado---2:Vetado---3:Veto Parcial
\n");

        do {

            scanf("%d",&salida);

        }while(salida<1 || salida>3);

        if(salida==1) {

            salida= 0;

```

```

do {

    printf("Ingrese la Fecha de entrada en
vigencia del Proyecto (year-Mes-Dia)\n");

    printf("year:");

    scanf("%d",&year);

    printf("\nMes:");

    scanf("%d",&mes);

    printf("\nDia:");

    scanf("%d",&dia);

    if(year>0 && (mes>0 && mes < 13) && (dia>0
&& dia <= 31)) {

        salida = 1;

        sprintf(numero,"%d-",year);

        strcat(fechaEnVigencia,numero );

        sprintf(numero,"%02d-",mes);

        strcat(fechaEnVigencia, numero);

        sprintf(numero,"%02d",dia);

        strcat(fechaEnVigencia, numero);

    }else {

        printf("La fecha Ingresada no es
valida\n");

    }

}while(salida==0);

proyectoVotado->fechaEntradaVigencia = (char *)
malloc(strlen(fechaEnVigencia)*sizeof(char));

```

```

        strcpy(proyectoVotado->fechaEntradaVigencia, fechaEnVigencia);

        printf("El proyecto fue enviado al diario
Oficial para su publicacion \n");

        printf("Diario Oficial\n\n");

        mostrarProyecto(proyectoVotado);

        printf("Fecha de entrada en Vigencia:
'%s'\n", proyectoVotado->fechaEntradaVigencia);

    }else {

        if(salida==2) {

            printf("El proyecto de ley '%s' fue Vetado
por el presidente de la republica", proyectoVotado->nombreProyecto);

            }else {

            }

        }

    }

    break;

}

case 3: {

    recorrerArbolProyectos(es->congreso->ley);

    break;

}

```

```

case 4:{

    printf("Camara Del Senado\n");

    proyectoMasVotado = getProyectoMasVotado(es->congreso-
>senadores,es->congreso->ley);

    if(proyectoMasVotado!=NULL) {

        printf("Nombre Proyecto de Ley:'%s'",proyectoMasVotado-
>nombreProyecto);

    }else {

        printf("No hay Proyectos\n");

    }

    printf("\nCamara Del Senado\n");

    proyectoMasVotado = getProyectoMasVotado(es->congreso-
>diputados,es->congreso->ley);

    if(proyectoMasVotado!=NULL) {

        printf("Nombre Proyecto de Ley:'%s'",proyectoMasVotado-
>nombreProyecto);

    }else {

        printf("No hay Proyectos\n");

    }

    printf("\n\n");

    break;

}

default: {

    printf("Ingrese una opcion valida");

    break;

}

```



```

        }

        }while(opcion!=1);
    }

int buscarPersonaCongreso(struct nodoParlamentario *head, char *RutBuscado)
{
    struct nodoParlamentario * rec;

    if(head==NULL) {

        return 0;

    }

    rec =head;

    while(rec!=NULL) {

        if(strcmp(rec->datosParlamentario->datos->rut,RutBuscado)==0) {

            return 1;

        }

        rec = rec->sig;

    }

    return 0;

}

struct Persona *datosPersona(struct Estado *es) {

    struct Persona *nueva;

    int edad,salida=0;

    char nombre[60],Rut[14];

    printf("Ingrese el nombre:");

```

```

scanf(" %[^\\n]", &nombre);

printf("Ingrese el Rut de la persona (Formato x.xxx.xxx-x):");

do {

    scanf("%s", &Rut);

    if (buscarPersonaCongreso(es->congreso->senadores, Rut) == 0 &&
        buscarPersonaCongreso(es->congreso->diputados, Rut) == 0) {

        salida = 1;

    } else {

        printf("Ese Rut ya Existe");

        printf("Ingrese 0 si quiere salir\\n");

        scanf("%s", &salida);

        if (salida == 1) {

            return NULL;

        }

    }

} while (salida != 1);

printf("Ingrese la Edad:");

scanf("%d", &edad);

nueva = crearPersona(nombre, Rut, edad, 0);

return nueva;

}

```

```

struct Parlamentario *DatosNuevoParlamentario(struct Estado *es, struct
nodoParlamentario *head, int ID) {

    int salida=0, IdParlamentario;

    char partido[50], especialidad[27], periodo[50];

    struct Persona *datos;

    struct Parlamentario *parlamentario;

    datos = datosPersona(es);

    if(datos!=NULL) {

        printf("Ingrese el ID:");

        do {

            scanf("%d",&IdParlamentario);

            if(buscarParlamentario(head, datos->rut, ID)==0) {

                salida=1;

            }else {

                printf("El ID Del Parlamentario Ya existe\n");

            }

        }while(salida!=1);

        printf("Ingrese el partido Politico: ");

        scanf(" %[^\\n]",partido);

        printf("Ingrese el numero Correspondiente a la Especialidad del
parlamentario\\n");

```

```

do{

    printf("1:Educacion---2:Salud---3:Reforma---4:Impuesto---
5:Defensa---06:Ministerio Publico \n");

    printf("7:Seguridad Social---8:Presupuesto---9:Poder Judicial--
-10:Coperacion Internacional\n");

    scanf("%d",&salida);

    if(agregarSectorProyectoLey(especialidad,salida)==0) {

        salida = 0;

        printf("Ingrese un Numero Valido\n");

    }

    }while(salida==0);

    printf("Ingrese el periodo: ");

    scanf(" %[^\\n]",periodo);

    parlamentario =
crearParlamentario(datos,IdParlamentario,periodo,partido,especialidad);

    return parlamentario;

}

printf("hubo un error\n");

return NULL;

}

```

```

void agregarCambioParlamentario(struct Estado *es,struct nodoParlamentario
*head,int idCambio){

```

```

    struct nodoParlamentario *rec;

```

```

struct Parlamentario *nuevoParlamentario;

rec=head;

while(rec!=NULL) {

    if(rec->datosParlamentario->idParlamentario == idCambio) {

        printf("Ingrese Los datos del Nuevo Parlamentario\n");

        nuevoParlamentario = DatosNuevoParlamentario(es,head,idCambio);

        rec->datosParlamentario = nuevoParlamentario;

        printf("Se agrego el Parlamentario\n");

    }

    rec = rec->sig;

}

}

```

```

void cambiarParlamentario(struct Estado *es,struct nodoParlamentario *head)
{

    int idCambio,salida=0;

    char rutBuscado[14];

    if(head==NULL) {

        printf("La Camara esta Vacía\n");

    }else {

```

```

        printf("Ingrese el ID del Parlamentario Que desea Cambiar y Tambien
el Rut (x.xxx.xxx-x)\n");

    do {

        printf("ID: ");

        scanf("%d",&idCambio);

        printf("Rut: ");

        scanf("%s",rutBuscado);

        if(buscarParlamentario(head,rutBuscado,idCambio)==1) {

            printf("Se encontro\n");

            agregarCambioParlamentario(es,head,idCambio);

            salida=1;

        }else {

            printf("No se encontro el ID \n");

            printf("Ingrese 0 si quiere seguir o 1 si quiere salir\n");

            scanf("%d",&salida);

        }

    }while(salida != 1);

}

}

void mostrarParlamentarios(struct CongresoNacional *congreso) {

    struct nodoParlamentario *rec;

    printf("Listado de Parlamentarios:\n");

```

```

// Mostrar Diputados

printf("\nCamara de Diputados:\n");

if(congreso->diputados==NULL) {

    printf("La camara de Diputados esta Vacía\n");

}else {

    rec = congreso->diputados;

    while (rec!= NULL) {

        printf("ID: %d, Nombre: %s\n", rec->datosParlamentario->idParlamentario, rec->datosParlamentario->datos->nombreApellido);

        rec = rec->sig;

    }

}

// Mostrar Senadores

printf("\nCamara del Senado:\n");

if(congreso->senadores==NULL) {

    printf("La camara del Senado esta Vacía\n");

}else {

    rec = congreso->senadores;

    while (rec!= NULL) {

        printf("ID: %d, Nombre: %s\n", rec->datosParlamentario->idParlamentario, rec->datosParlamentario->datos->nombreApellido);

        rec = rec->sig;

    }

}

```

```
}
```

```
void agregarDatosParlamentario(struct Estado *es){

    int opcion, salida = 0;

    int idFalso = -22;

    struct Parlamentario *nuevoParlamentario;

    do {

        printf("Menu de Comienzo de votaciones Congreso Nacional\n");

        printf("Opcion 1: Salir al menu Principal\n");

        printf("Opcion 2: Modificar Parlamentarios\n");

        printf("Opcion 3: Mostrar los Parlamentarios\n");

        scanf("%d", &opcion);

        switch (opcion) {

            case 1: {

                printf("Saliendo del Menu de Agregacion de  
Parlamentarios\n");

                break;

            }

            case 2: {

                printf("Ingreso a Modificar Parlamentario\n");

                do {

                    printf("Ingrese que camara quiere modificar\n");

                    printf("1:Diputado ---- 2:Senado\n");
```



```

do {

    scanf("%d", &salida);

    if (salida != 1 && salida != 2) {

        printf("Ingrese un Numero Valido\n");

    }

} while (salida != 1 && salida != 2);

if (salida == 1) {

    printf("Ingreso a la Camara de Diputados\n");

    printf("1:Modificar Diputados --- 2: Agregar
Diputados\n");

    do {

        scanf("%d", &salida);

        if (salida != 1 && salida != 2) {

            printf("Ingrese un Numero Valido\n");

        }

    } while (salida != 1 && salida != 2);

    if (salida == 1) {

        cambiarParlamentario(es, es->congreso-
>diputados);

    } else {

        if(es->congreso->numDiputados<155) {

```

```

        nuevoParlamentario =
DatosNuevoParlamentario(es, es->congreso->diputados, idFalso);

        agregarParlamentario(&(es->congreso-
>diputados), nuevoParlamentario, &(es->congreso->numDiputados));

    }else {

        printf("Esta Llena la Camara de
Diputados\n");

    }

}

} else {

    printf("Ingreso a la Camara del Senado\n");

    printf("1:Modificar Senado --- 2: Agregar
Senador\n");

    do {

        scanf("%d", &salida);

        if (salida != 1 && salida != 2) {

            printf("Ingrese un Numero Valido\n");

        }

    } while (salida != 1 && salida != 2);

    if (salida == 1) {

        cambiarParlamentario(es, es->congreso-
>senadores);

```

```

        } else {

            if(es->congreso->numSenadores<50) {

                nuevoParlamentario =
DatosNuevoParlamentario(es, es->congreso->senadores, idFalso);

                agregarParlamentario(&(es->congreso-
>senadores), nuevoParlamentario,&(es->congreso->numSenadores));

            }else {

                printf("Esta Llena la Camara del Senado");

            }

        }

    }

    printf("Ingrese 1 si quiere Salir al menu anterior o
Ingrese 0 si quiere seguir agregando Parlamentarios :\n");

    scanf("%d",&salida);

    }while(salida!=1);

    break;

}

case 3: {

    printf("Ingreso a Mostrar Parlamentario\n");

    mostrarParlamentarios(es->congreso);

    printf("\n\n");

    break;

```

```

    }

    default: {

        printf("Ingrese una opcion valida\n");

        break;

    }

}

} while (opcion != 1);

}

int eliminarCiudadano(struct nodoPersona **head, char *rut) {

    struct nodoPersona *rec,*ant;

    if (*head == NULL) {

        return 0;

    }

    rec= *head;

    do {

        if (strcmp(rec->datosPersona->rut, rut) == 0) {

            if (rec == *head && rec->sig == *head) {

                *head = NULL;

            } else {

                if (rec == *head) {

                    *head = rec->sig;

                }

            }

        }

    } while (rec != NULL);

}

```

```

        rec->ant->sig = rec->sig;

        rec->sig->ant = rec->ant;

    }

    return 1;

}

ant = rec;

rec = rec->sig;

} while (rec != *head);

return 0;

}

void mostrarCiudadanos(struct nodoPersona *head) {

    struct nodoPersona *rec ;

    if (head == NULL) {

        printf("No hay ciudadanos registrados.\n\n");

        return;

    }

    rec =head;

    do {

        printf("Nombre: %s, RUT: %s, Edad: %d\n", rec->datosPersona-
>nombreApellido, rec->datosPersona->rut, rec->datosPersona->edad);

        rec = rec->sig;

    } while (rec != head); // Asumiendo que es una lista circular

    printf("\n\n");

```

```
}
```

```
void agregarDatosCiudadanos(struct Estado *es) {

    struct Persona *nuevaPersona;

    int salida =0,opcion;

    char rutEliminar[14];

    do {

        printf("Menu de Ciudadanos:\n");

        printf("1. Agregar ciudadano\n");

        printf("2. Eliminar ciudadano\n");

        printf("3. Mostrar Ciudadanos\n");

        printf("4. Salir\n");

        printf("Seleccione una opcion:");

        scanf("%d", &opcion);

        switch (opcion) {

            case 1: {

                printf("Agregar un nuevo ciudadano:\n");

                // Crear una nueva persona solicitando datos al usuario

                nuevaPersona = datosPersona(es);

                if (nuevaPersona != NULL) {

                    // Agregar la nueva persona a la lista de ciudadanos
```

```

        if (agregarNodoPersona(&es->ciudadanos->personas,
nuevaPersona)==1) {

            printf("Ciudadano agregado exitosamente.\n");

        } else {

            printf("No se pudo agregar el ciudadano.\n");

        }

    }

    break;

}

case 2: {

    printf("Ingrese el RUT del ciudadano a eliminar (Formato
x.xxx.xxx-x):");

    scanf("%s", rutEliminar);

    if (eliminarCiudadano(&es->ciudadanos->personas,
rutEliminar)) {

        printf("Ciudadano eliminado exitosamente.\n\n");

    } else {

        printf("No se encontro el ciudadano con el RUT
proporcionado.\n");

    }

    break;

}

case 3: {

```

```

        printf("Mostrando todos los ciudadanos:\n");

        mostrarCiudadanos(es->ciudadanos->personas);

        break;
    }

    case 4: {

        salida = 1;

        printf("Saliendo del menu de ciudadanos.\n\n");

        break;
    }

    default: {

        printf("Opción no válida. Intente de nuevo.\n");

        break;
    }

}

} while (salida == 0);

}

int main(){

    int opcion;

    struct Estado *es;

    es= (struct Estado*)malloc(sizeof(struct Estado));

    es->congreso= (struct CongresoNacional *) malloc(sizeof(struct
CongresoNacional));

    es->congreso->ley = NULL;

```



```

es->congreso->senadores = es->congreso->diputados = NULL;

es->congreso->numSenadores = es->congreso->numDiputados = 0;

es->ciudadanos = (struct Ciudadanos *) malloc(sizeof(struct
Ciudadanos));

es->ciudadanos->personas = NULL;

es->presidente= (struct Presidente *) malloc(sizeof(struct
Presidente));

es->presidente->datosPresidente =NULL;

es->presidente->proyectoLey = NULL;

do {

    printf("1. Menu Agregar Proyecto de Ley\n");

    printf("2. Menu Parlamentario\n");

    printf("3. Menu Agregar Ciudadano\n");

    printf("4. Menu Clientes\n");

    printf("5. Menu Votacion Proyecto De Ley\n");

    printf("6. Introducir Presidente\n");

    printf("0. Salir\n");

    printf("Ingrese opcion: \n");

    scanf("%d", &opcion);

    switch (opcion) {

        case 0:{

            printf("Saliendo\n");

            break;

        }

    }

```

```

        case 1: {

            printf("\nSeleccionaste la opcion 1 Menu Agregar Proyecto
de Ley\n\n");

            ingresarProyectoLey(es, &es->congreso->ley);

            break;

        }

        case 2: {

            printf("\nSeleccionaste la opcion 2 Menu
Parlamentario\n\n");

            agregarDatosParlamentario(es);

            break;

        }

        case 3:{

            printf("\nSeleccionaste la opcion 3 Menu Agregar
Ciudadano\n\n");

            agregarDatosCiudadanos(es);

            break;

        }

        case 4: {

            printf("\nSeleccionaste la opcion 4 Menu Clientes\n\n");

            agregarDatosCiudadanos(es);

            break;

```

```

    }

    case 5: {

        printf("\nSeleccionaste la opcion 5 Menu Votacion Proyecto
De Ley\n\n");

        comienzoVotacionCongreso(es);

        break;

    }

    case 6: {

        printf("\nSelecciono Introducir Presidente\n\n");

        printf("Ingrese los datos\n");

        es->presidente->datosPresidente = datosPersona(es);

        printf("\n\n");

        break;

    }

    default: {

        printf("Ingrese opcion valida\n\n");

        break;

    }

}

}while(opcion!=0);

return 0;

}

```