



UNIVERSIDAD DIEGO PORTALES

## Laboratorio 2: Votación

ESTRUCTURAS DE DATOS Y ALGORITMOS  
2025-1

Profesores:  
Valentina Aravena  
Cristián Llull  
Marcos Fantoval

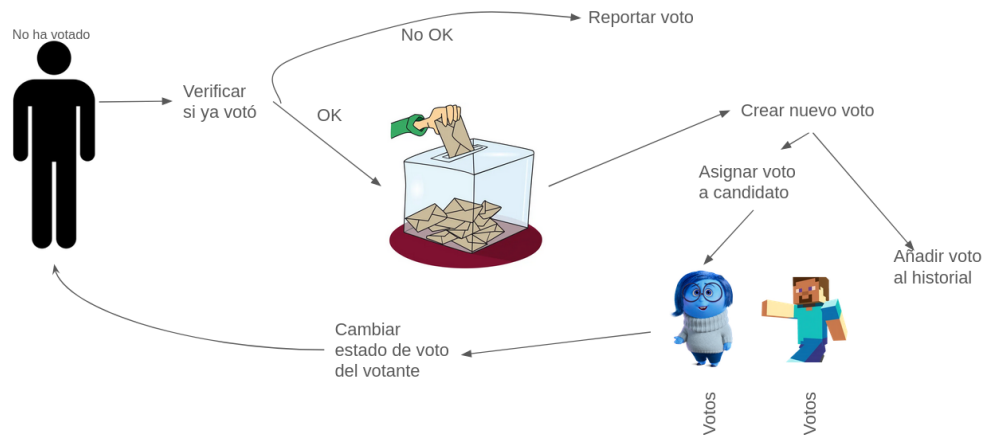


Figura 1: Digrama de votación.

## 1. Introducción

En la Facultad de Ingeniería y Ciencias de la Universidad Diego Portales se llevarán a cabo las elecciones para presidente del Centro de Alumnos de la Escuela de Informática y Telecomunicaciones. A usted se le ha solicitado crear un sistema de votaciones llamado “Electo”, que permitirá gestionar candidatos, votantes y resultados de elecciones. El sistema deberá utilizar **listas enlazadas**, **pilas** y **colas**, así como garantizar la integridad de los votos, evitar duplicaciones y permitir consultas eficientes sobre los resultados.

### 1.1. Flujo de la votación

La votación seguirá el siguiente flujo:

1. La persona va a votar.
2. El sistema verifica si la persona ya votó:
  - Si ya votó, reportar el voto.
  - Si no ha votado, ingresarlo a la urna.
3. Para ingresar el voto en la urna, se deben seguir una serie de pasos:
  - Crear nuevo voto.
  - Asignar el voto a la cola del candidato correspondiente.
  - Añadir voto al historial.
  - Cambiar estado de voto del votante.

### 1.2. Diagrama de clases UML

UML (*Unified Modeling Language*), es una notación estándar de representar objetos en la fase de diseño de un sistema orientado a objetos. La representación es una tabla de tres secciones, siendo el primero el nombre del objeto, el segundo los atributos y el tercero los métodos.

Perro
- Nombre: String - Ladrido: String - Nacimiento: Date - Dirección: Address
+ ladrar(): String + cambiarLadrido(String nuevoLadrido): void

Note la visibilidad de los atributos: ‘-’ es un atributo «*private*», mientras que ‘+’ es «*public*».

## 2. Implementación

Deberá implementar las siguientes clases:

Voto	Candidato	Votante
- id: int - votanteId: int - candidatoId: int - timeStamp: String	- id: int - nombre: String - partido: String - votosRecibidos: Queue<Voto>	- id: int - nombre: String - yaVoto: boolean
+ Voto(): Voto + getters... + setters...	+ Candidato(): Candidato + getters... + setters... + agregarVoto(Voto)	+ Votante(): Votante + getters... + setters... + marcarVotado(): void

UrnaElectoral
- listaCandidatos: LinkedList - historialVotos: Stack<Votos> - votosReportados: Queue<Votos> - idCounter: int
+ verificarVotante(Votante): boolean + registrarVoto(Votante, int): boolean + reportarVoto(Candidato, int): boolean + obtenerResultados(): String

### 2.1. Clase **Voto**

#### Atributos

- **id** (entero, único para cada voto).
- **votanteID** (ID del votante, entero, utilizado para evitar votos duplicados)
- **candidatoID** (ID del candidato seleccionado, entero)
- **timestamp** (Hora del voto en formato “hh:mm:s”).

#### Métodos

- Constructor, getters y setters.

## 2.2. Clase **Candidato**

### Atributos

- **id** (identificador único, entero)
- **nombre** (cadena de caracteres)
- **partido** (cadena de caracteres)
- **votosRecibidos** (cola de votos asociados, Voto)

### Métodos

- Constructor, getters y setters.
- **agregarVoto**(Voto v): Añade un voto a la cola de votos del candidato.

## 2.3. Clase **Votante**

### Atributos

- **id** (entero, único)
- **nombre** (cadena de caracteres)
- **yaVoto** (booleano)

### Método

- Constructor, getters y setters.
- **marcarComoVotado**(): Cambia **yaVoto** a **true**

## 2.4. Clase **UrnaElectoral**

### Atributos

- **listaCandidatos** (lista enlazada de Candidatos)
- **historialVotos** (pila para almacenar votos en orden cronológico inverso (la pila recibirá sólo Votos))
- **votosReportados** (cola para votos anulados o impugnados (La cola recibirá solo Votos))
- **idCounter** (contador de IDs para votos, entero)

### Métodos clave

- **verificarVotante**(Votante votante): Verifica si el votante ya ha votado.
- **registrarVoto**(Votante votante, int candidatoID): Utiliza el método **verificarVotante**, luego registra el voto (se tiene que crear un nuevo Voto para agregarlo al Candidato) en el Candidato correspondiente y lo añade al historial. Posteriormente tiene que cambiar el estado del Votante.
- **reportarVoto**(Candidato candidato, int idVoto): Mueve un voto de la cola correspondiente al candidato a la cola de votos reportados (Entrega un mensaje en caso de existir ya el voto en la cola por ejemplo, en caso de fraude).
- **obtenerResultados**(): Retorna un mapa con el conteo de votos por Candidato.

### 3. Análisis

#### Complejidad Temporal

Crea una tabla de notación Big-O para los métodos clave (`registrarVoto`, `obtenerResultados`, `reportarVoto`)

#### Gestión de Memoria

Calcula el espacio requerido para almacenar todos los votos si:

- Cada voto ocupa 64 bytes.
- El sistema admite hasta 10 millones de votantes.

#### Propuesta de Mejora

¿Cómo modificarías el sistema para soportar votaciones en múltiples facultades? <sup>1</sup>. Explica brevemente.

### 4. Informe

El informe debe incluir:

- Introducción al problema.
- Descripción de la implementación (clases auxiliares, pruebas realizadas)
- Análisis de complejidad y uso de memoria.
- Ventajas y desventajas de utilizar listas enlazadas en lugar de arreglos para este sistema.
- Conclusiones y posibles extensiones (Ej: Votación Electrónica Segura)

#### Objetivos del laboratorio

- Evaluar el dominio de listas, pilas y colas en un contexto realista.
- Practicar diseño de clases y gestión eficiente de memoria.
- Analizar ventajas y desventajas entre distintas estructuras de datos.

---

<sup>1</sup>¿Cómo podrías mantener el historial de votos en orden cronológico inverso?