CALCULADORA ELECTORAL LEY D'HONDT

MEMORIA TÉCNICA

PEDRO JUAN LÓPEZ MARTÍNEZ

ÍNDICE

1. Introducción

Esta memoria técnica describe el desarrollo completo del proyecto **"Calculadora** Electoral Ley D'Hondt", una aplicación web interactiva diseñada para simular el reparto de escaños utilizando el método oficial D'Hondt, ampliamente empleado en los sistemas electorales proporcionales. Su finalidad es ofrecer una herramienta pedagógica, funcional y accesible, orientada tanto al análisis político como al aprendizaje académico. El proyecto se presenta como Trabajo Final del módulo de Proyecto, dentro del ciclo formativo, integrando conocimientos adquiridos en programación, diseño web, bases de datos y desarrollo backend.

La aplicación permite introducir datos electorales personalizados, calcular la asignación de escaños en tiempo real y visualizar los resultados mediante tablas, gráficos y un pactómetro interactivo. Para lograrlo, se ha construido un backend basado en FastAPI, encargado de realizar todo el procesamiento lógico y matemático, y un frontend desarrollado con HTML, CSS, Bootstrap y JavaScript, que ofrece una interfaz clara, moderna y fácil de usar. Ambos componentes se comunican mediante una API REST propia.

Durante el desarrollo se han aplicado metodologías actuales de diseño web, principios de programación estructurada, separación de responsabilidades y validación robusta de datos tanto en el cliente como en el servidor. Además, el proyecto incluye un sistema básico de autenticación de usuarios y la posibilidad de guardar, cargar y gestionar simulaciones electorales en una base de datos MySQL. De este modo, la herramienta permite un uso continuado y personalizado, adaptado a distintos escenarios y necesidades.

El presente documento detalla el contexto de creación del proyecto, los objetivos perseguidos, las tecnologías y herramientas empleadas, las principales decisiones técnicas adoptadas, el diseño de la arquitectura, el proceso de implementación y una valoración final del resultado obtenido junto con posibles mejoras futuras. Se busca ofrecer una visión completa y rigurosa del trabajo realizado, facilitando su comprensión y evaluación técnica.

2. Objetivos del Proyecto

El desarrollo de la Calculadora Electoral Ley D'Hondt tiene como propósito construir una aplicación web completa, robusta y orientada al análisis electoral. Para ello, se han definido los siguientes objetivos fundamentales, que guían tanto la arquitectura técnica como la experiencia de usuario:

- Implementar un sistema fiable para calcular el reparto de escaños mediante el método D'Hondt. Se desarrolla un backend capaz de aplicar con precisión el algoritmo oficial utilizado en procesos electorales reales. La aplicación debe permitir realizar cálculos repetibles, coherentes y ajustados a la normativa matemática del método.

- Permitir la personalización completa de los parámetros electorales. El usuario puede ajustar elementos clave del proceso electoral, tales como:
  - Número total de escaños a repartir.
  - Votos en blanco.
  - Votos nulos.
  - Umbral electoral expresado en porcentaje.
  - Lista de partidos con sus votos y colores asignados.

  Esta flexibilidad convierte la herramienta en un entorno de simulación adaptable a diferentes sistemas municipales, autonómicos o estatales.

- Incorporar visualización interactiva mediante gráficos intuitivos. Uno de los objetivos es facilitar la comprensión del reparto de escaños mediante:
  - Un hemiciclo interactivo que representa la composición del órgano legislativo.
  - Un pactómetro funcional, capaz de mostrar mayorías, combinaciones posibles de partidos y representación visual proporcional.
- Integrar un sistema de gestión de usuarios y simulaciones. La aplicación debe permitir que cada usuario registrado pueda:
  - Guardar sus simulaciones personalizadas.
  - Cargarlas posteriormente.
  - Modificarlas y volver a guardarlas.
  - Gestionar múltiples escenarios electorales desde su cuenta.

- Garantizar validaciones sólidas en todos los niveles. Uno de los pilares del proyecto es asegurar que la información introducida por el usuario sea coherente y segura. Para ello:
  - El frontend realiza validaciones inmediatas para mejorar la experiencia de usuario,
  - El backend, mediante Pydantic y comprobaciones adicionales.

3. Planificación y Metodología

La planificación del proyecto se organizó en diferentes fases consecutivas pero interrelacionadas, siguiendo un enfoque incremental que permitió construir la aplicación de manera sólida y verificable desde sus primeras etapas. Cada fase tuvo unos objetivos concretos y unas tareas asociadas que facilitaron la evolución ordenada del desarrollo.

Análisis inicial

En esta primera fase se llevó a cabo un estudio detallado del método D'Hondt, prestando especial atención a su formulación matemática, reglas de asignación y requisitos para realizar simulaciones fieles a un proceso electoral real. Paralelamente, se definió el alcance del proyecto: funcionalidades imprescindibles, características opcionales y limitaciones técnicas. Se identificaron los perfiles de usuario y los escenarios de uso más relevantes, lo que permitió delimitar con claridad los objetivos funcionales y no funcionales.

Diseño

Durante el diseño se determinaron:

- La estructura general de la interfaz.
- El flujo de datos entre usuario, frontend y backend.
- La modularidad del sistema.
- Las tecnologías más adecuadas para cada capa.

Se eligió FastAPI como framework backend por su rapidez y su integración con Pydantic, y JavaScript en el frontend por su flexibilidad para construir interfaces dinámicas.

El diseño incluyó wireframes de la interfaz, planificación de endpoints API, definición de esquemas de datos y estudio de cómo representar visualmente el hemiciclo y el pactómetro.

Implementación

La implementación se dividió en dos líneas principales, desarrolladas de forma paralela:

- Backend: programación de la API REST, modelo de datos, validaciones, autenticación y operaciones con base de datos MySQL.
- Frontend: construcción de la interfaz, validaciones de formulario, comunicación con el backend, representación gráfica con Chart.js y gestión de estados de usuario.

Las dos partes se integraron progresivamente mediante peticiones AJAX, verificando en cada paso la coherencia del flujo de datos y el correcto funcionamiento del algoritmo D'Hondt.

Pruebas

Las pruebas incluyeron:

- Validación exhaustiva del algoritmo D'Hondt en distintos escenarios.
- Comprobación de los límites del sistema (votos cero, umbrales altos, partidos duplicados…).
- Pruebas manuales de usuario para asegurar la correcta experiencia de uso.
- Verificación del comportamiento del interfaz ante errores y entradas inválidas.
- Pruebas de persistencia en base de datos y carga/edición de simulaciones.

4. Tecnologías Empleadas

El proyecto se ha construido utilizando un conjunto de tecnologías modernas, estables y ampliamente utilizadas en entornos profesionales. Cada una ha sido seleccionada en función de su adecuación a los objetivos del proyecto, su curva de aprendizaje y su capacidad para integrarse con el resto de componentes.

Python + FastAPI

FastAPI se utilizó como framework principal para el backend debido a su rendimiento elevado, su sintaxis clara y su soporte nativo para tipado estático y validación de datos mediante Pydantic.
Proporciona una estructura ordenada para construir APIs REST y permite definir endpoints de forma intuitiva, facilitando la creación de operaciones como calcular escaños, guardar simulaciones, gestionarlas y autenticar usuarios.
Su rapidez y diseño asíncrono aseguran tiempos de respuesta muy reducidos.

JavaScript

Para el frontend se decidió emplear JavaScript por simplicidad y control total del flujo de datos. Contiene toda la lógica relacionada con:

- Validaciones del usuario.
- Comunicación con la API mediante Fetch.
- Actualización dinámica de la interfaz,
- Creación del pactómetro y el hemiciclo.

MySQL

La base de datos utilizada fue MySQL, un sistema de gestión de bases de datos relacional ampliamente extendido.
Su papel en el proyecto es almacenar:

- Usuarios registrados.
- Simulaciones guardadas.
- Fecha de creación de cada simulación y metadatos asociados.

Chart.js

Para la visualización gráfica del hemiciclo se utilizó Chart.js, una librería ligera y flexible.
A través de un gráfico semicircular, se simula de forma visual un parlamento. Esta librería permitió:

- Representar escaños por partido.
- Usar colores personalizados,

Bootstrap 5

Bootstrap proporciona la estructura visual del proyecto y permite una interfaz limpia, moderna y adaptable.
Se utilizó para:

- El diseño responsivo.
- El sistema de columnas.
- Botones personalizados.
- Formularios y tarjetas.
- Diseño del header y del login.

HTML + CSS

La base estructural del frontend se implementó en HTML5, componiendo una interfaz semántica y organizada.
El archivo CSS incorpora:

- Variables de color para la identidad visual.
- Estilos personalizados para botones, tablas y validaciones.
- Mejoras específicas para el pactómetro y el hemiciclo.

Pydantic

Pydantic se integra en FastAPI para validar y tipar datos de manera estricta. Permite garantizar que:

- Las peticiones del cliente cumplen la estructura esperada,
- No llegan al servidor valores malformados,
- Se evitan errores en la lógica interna del backend.

5. Diseño de la Interfaz

La interfaz de usuario ha sido diseñada siguiendo un enfoque funcional y orientado a la claridad, asegurando que cualquier persona, incluso sin conocimientos técnicos, pueda utilizar la aplicación de manera intuitiva. Para ello, la estructura visual del proyecto se divide en cuatro bloques principales, cada uno con responsabilidades bien definidas:

Cabecera (Header)

La cabecera integra la identidad visual del proyecto y el sistema de autenticación de usuarios.
Incorpora:

- El logotipo del proyecto.
- El título de la aplicación y una breve descripción.
- Un módulo compacto de inicio de sesión con campos para usuario y contraseña.
- Cambio dinámico del estado visual según el usuario esté conectado o no (login/logout).

Este enfoque permite acceder al sistema de usuario desde cualquier dispositivo sin ocupar espacio innecesario en pantallas pequeñas.

Zona de Configuración

Es el panel donde el usuario introduce todos los parámetros necesarios para la simulación.
Incluye campos para:

- Nombre de la simulación.
- Número total de escaños.
- Votos en blanco y nulos.
- Porcentaje de umbral electoral.
- Tabla editable de partidos con sus votos y colores personalizados.

La tabla permite añadir y eliminar partidos dinámicamente, y la validación del formulario guía al usuario para evitar errores en los datos.
Este bloque actúa como el punto de entrada lógico para todas las operaciones del sistema.

Zona de Resultados

Una vez realizada la simulación, los resultados se muestran en un área dedicada que combina información numérica y representación visual.
Está compuesta por:

- Tabla de reparto de escaños: lista de partidos con sus votos, escaños y color.
- Representación gráfica del parlamento: hemiciclo interactivo generado con Chart.js.

- Pactómetro: herramienta visual que permite seleccionar partidos y ver si alcanzan mayoría absoluta.
  El pactómetro calcula porcentajes, suma escaños y muestra un gradiente de colores proporcional al pacto.

Este bloque convierte los resultados en elementos visuales fácilmente interpretables, facilitando el análisis.

Bloque de Simulaciones

Este módulo permite al usuario:

- Ver sus simulaciones guardadas.
- Filtrarlas por nombre mediante un buscador.
- Cargarlas para revisarlas.
- Sobrescribirlas o eliminarlas.

El contenido se muestra en una tabla generada dinámicamente tras consultar la base de datos vía API.
Además, el bloque se desactiva visualmente cuando el usuario no está logueado, manteniendo coherencia y seguridad funcional.

Diseño Responsivo

Toda la interfaz está desarrollada con Bootstrap 5, aprovechando su sistema de rejilla y utilidades responsivas para adaptarse automáticamente a distintos tamaños de pantalla.
Se priorizó:

- Legibilidad en dispositivos móviles.
- Ordenamiento claro de secciones.
- Botones suficientemente grandes.
- Tablas con desplazamiento horizontal cuando es necesario.
- Reparto equilibrado entre secciones en pantallas grandes.

El resultado es una interfaz adecuada para su uso tanto en entornos educativos como en análisis electoral informal o profesional.

6. Backend: Arquitectura y Desarrollo

El backend del proyecto está construido sobre FastAPI, un framework moderno y de alto rendimiento diseñado para crear APIs RESTful de manera clara, eficiente y fácilmente escalable.
La arquitectura sigue un enfoque modular que favorece la mantenibilidad del código y permite una evolución fluida del proyecto en versiones futuras.

Estructura general del backend

El servidor se organiza en varios módulos independientes, cada uno con responsabilidades claramente definidas:

• main.py

Es el archivo principal del backend y actúa como punto de entrada de la API. En él se concentran:

- La declaración y configuración de la aplicación FastAPI.
- La definición de los endpoints:
  - Cálculo de escaños.
  - Registro y autenticación de usuarios.
  - Almacenamiento, carga, actualización y eliminación de simulaciones.
- La carga del middleware CORS para permitir la comunicación con el frontend.
- La validación de datos de entrada mediante modelos Pydantic.
- El control de errores y respuestas estándar.

Este archivo constituye el corazón operativo del backend.

• db.py

Gestiona la conexión con la base de datos MySQL usando un método dedicado (get_connection()), lo que permite:

- Separar la lógica de acceso a datos de la lógica de negocio.
- Reutilizar la función de conexión en todos los endpoints.
- Simplificar futuras migraciones a otros motores de base de datos (PostgreSQL, MariaDB, etc.).
- Facilitar el mantenimiento en caso de modificar credenciales o estructura de tablas.

Gracias a esta separación, el backend mantiene una estructura limpia y fácilmente comprensible.

• dhondt.py

Contiene la lógica del método D'Hondt en su forma más pura, sin dependencias de FastAPI ni de la base de datos.

Este aislamiento presenta diversas ventajas:

- El algoritmo es reutilizable y testeable de forma independiente.

- Se evitan efectos secundarios al usarlo desde diferentes partes del proyecto.
- Permite explicar de forma clara el funcionamiento del reparto de escaños dentro de la memoria.

Este módulo representa el nivel cero del cálculo, sobre el que el backend construye la lógica de validación y presentación de resultados.

API REST y funcionalidades implementadas

La API proporciona un conjunto completo de endpoints que cubren todas las necesidades del proyecto:

• Endpoint de cálculo

Permite al cliente enviar votos, umbral, partidos y parámetros electorales para obtener el reparto de escaños.

• Endpoints de autenticación

Incluyen:

- Registro con validación estricta de contraseña.
- Inicio de sesión con verificación de credenciales mediante hash SHA-256.

• CRUD completo de simulaciones

El backend permite:

- Crear simulaciones.
- Leerlas (listado y detalle).
- Actualizarlas evitando duplicados.
- Eliminar simulaciones asociadas al usuario.

Todos los datos se guardan en formato JSON dentro de MySQL, lo que simplifica la persistencia y hace posible almacenar cualquier estructura generada por el cálculo electoral.

Validación con Pydantic

Uno de los pilares del backend es la validación estricta mediante **Pydantic**, que garantiza:

- Tipado fuerte de los datos de entrada.
- Validación automática de campos obligatorios.
- Seguridad frente a datos incorrectos enviados desde el cliente.
- Generación automática de documentación interactiva en /docs.
- Trazabilidad clara de errores para el usuario.

Gracias a esta capa de validación, el backend mantiene su integridad incluso ante peticiones ilegítimas o manipuladas.

7. Frontend: Arquitectura y Desarrollo

El frontend del proyecto se ha desarrollado utilizando HTML5, CSS3, Bootstrap 5 y JavaScript.
La totalidad de la lógica dinámica se encuentra centralizada en el archivo app.js, que actúa como controlador principal del comportamiento de la interfaz, de la comunicación con el backend y del proceso de validación de datos.

La arquitectura del frontend se estructura de forma modular y funcional, dividiendo responsabilidades para garantizar orden, legibilidad y mantenibilidad. A continuación se detallan las áreas principales cubiertas por app.js:

Validación de formularios

El sistema de validación combina dos enfoques:

- Validación inmediata (UX)
  Se aplica mientras el usuario interactúa con los campos de entrada.
  - Marca en rojo los valores incorrectos.
  - Elimina el estado de error automáticamente cuando el usuario corrige el campo.
  - Refuerza la accesibilidad y evita errores antes de enviar datos al backend.
- Validación completa al ejecutar acciones clave
  Antes de calcular o guardar la simulación se ejecuta una validación estricta sobre:
  - Número de escaños.
  - Votos nulos y en blanco.
  - Umbral.
  - Estructura de los partidos.
  - Ausencia de duplicados en los nombres.
  - Integridad general de la simulación.

Este doble sistema asegura que los datos enviados al backend son coherentes, reduciendo errores y mejorando la experiencia de usuario.

Gestión de la tabla de partidos

La aplicación permite añadir y eliminar filas dinámicamente, lo que proporciona flexibilidad para simular diferentes configuraciones electorales.

Las funciones dedicadas realizan:

- Creación de nuevas filas preconfiguradas.
- Eliminación segura de la última fila garantizando que siempre exista al menos un partido.
- Lectura y validación completa de la tabla para generar un modelo de datos estructurado.
- Marcado individual de errores en caso de votación negativa, campo vacío o nombres duplicados.

Este módulo convierte la tabla en una entidad totalmente interactiva y libre de inconsistencias.

Representación gráfica del hemiciclo

El proyecto incluye un hemiciclo visual generado mediante Chart.js, que proporciona una representación clara y atractiva del reparto de escaños.

El frontend se encarga de:

- Transformar el resultado del algoritmo en una estructura gráfica.
- Asignar colores personalizados a cada partido.
- destruir y crear de nuevo el gráfico cuando se actualiza la simulación para evitar fugas de memoria.
- adaptar el tamaño al contenedor para mantener responsividad.

Este componente aporta un valor visual clave que hace la herramienta más intuitiva y didáctica.

Lógica del pactómetro

El pactómetro es una de las funcionalidades más interactivas del proyecto y permite explorar posibles combinaciones de gobierno.

El frontend gestiona:

- Generación dinámica de checkboxes para cada partido con escaños.
- Cálculo en tiempo real de la suma de fuerzas.
- Creación automática de un gradiente de colores proporcional al peso de cada partido seleccionado.
- Visualización de mayoría absoluta mediante una barra reactiva.
- Actualización simultánea de texto descriptivo: “Escaños del pacto” y “Mayoría absoluta”.

Sistema de usuarios y autenticación

Aunque la seguridad fuerte se delega al backend, el frontend implementa:

- Control de sesión a través de localStorage.
- Actualización dinámica de la interfaz según el estado de login.
- Restricción de botones y accesos cuando no hay sesión activa.
- Limpieza completa de la interfaz al cerrar sesión.
- Restablecimiento de los datos por defecto.

El manejo de estados permite que el usuario interactúe con la aplicación de forma personalizada y segura.

Comunicación con la API (Fetch)

El frontend se comunica con el backend a través de solicitudes HTTP realizadas con Fetch API, enviando y recibiendo datos en formato JSON.

App.js implementa:

- Envío de datos electorales para el cálculo de escaños.
- Registro de usuarios.
- Login mediante validación de credenciales.
- Guardado y sobrescritura de simulaciones.
- Listado, filtrado, carga y eliminación de simulaciones.

Cada llamada incluye manejo de errores, parsing de respuestas y comunicación clara con el usuario mediante mensajes específicos.

8. Base de Datos

Para el almacenamiento persistente de la información, el proyecto utiliza **MySQL**, un sistema de gestión de bases de datos relacional ampliamente utilizado en entornos profesionales y compatible con el servidor local XAMPP utilizado durante el desarrollo.
La elección de MySQL responde a su estabilidad, su facilidad de integración con Python mediante conectores estándar y su capacidad para manejar tanto datos estructurados como información serializada.

Estructura general de la base de datos

La base de datos está compuesta por dos tablas principales, diseñadas para mantener una separación clara entre la gestión de usuarios y el almacenamiento de simulaciones electorales:

Tabla usuarios

Campo	Tipo	Descripción
id	INT (PK, AI)	Identificador único del usuario
username	VARCHAR	Nombre de usuario, único en la aplicación
password_hash	VARCHAR	Contraseña almacenada mediante hash SHA-256

Esta tabla gestiona la autenticación básica del sistema. No se almacenan contraseñas en texto plano; en su lugar se utiliza un hash criptográfico, garantizando un nivel mínimo de seguridad.

Tabla simulaciones

Campo	Tipo	Descripción
id	INT (PK, AI)	Identificador único de la simulación
usuario_id	INT (FK)	Relación con la tabla usuarios
nombre	VARCHAR	Nombre asignado por el usuario a la simulación
datos_json	LONGTEXT / JSON	Estructura completa de la simulación almacenada como JSON
fecha	TIMESTAMP	Fecha automática de creación o modificación

Esta tabla permite almacenar una simulación completa sin necesidad de crear múltiples tablas auxiliares. Gracias al uso de JSON, el proyecto conserva:

- La lista de partidos.
- Los colores personalizados.
- Los votos blancos y nulos.
- El umbral aplicado.
- Los cálculos resultantes (escaños, votos válidos, umbral mínimo).
- El reparto final generado por el algoritmo.

Este enfoque facilita la evolución del proyecto: si en un futuro se ampliara el modelo (por ejemplo, añadiendo circunscripciones, más parámetros u otros métodos de reparto), la estructura JSON podría adaptarse fácilmente sin necesidad de alterar el esquema relacional.

Justificación del uso de JSON

El uso de datos_json como contenedor flexible permite:

- Almacenar de manera compacta toda la simulación, incluidos sus resultados.
- Evitar una proliferación innecesaria de tablas (partidos, resultados, umbrales, configuraciones, etc.).
- Reducir la complejidad del CRUD, ya que las operaciones se realizan sobre un único bloque estructurado.
- Facilitar la compatibilidad entre versiones, incluso cuando se añadan o eliminen campos.

Integridad y restricciones

Se aplican claves foráneas lógicas mediante la relación usuario_id → usuarios.id.

No se permite registrar nombres duplicados de simulación por usuario.

Todas las validaciones de integridad (votos válidos, nombres no vacíos, umbral correcto) se realizan también en el backend antes de guardar la simulación.

De esta forma, el sistema garantiza que los datos almacenados sean siempre consistentes.

Conexión y operaciones desde la API

El backend utiliza un módulo auxiliar db.py que encapsula la conexión a MySQL. Esto permite:

- Crear una capa de abstracción.
- Centralizar parámetros de conexión.
- Reutilizar la misma función get_connection() en todos los endpoints.

Las operaciones realizadas incluyen:

INSERT: guardar nuevas simulaciones.

SELECT: listar simulaciones o recuperar una concreta.

UPDATE: sobrescribir simulaciones existentes.

DELETE: eliminar simulaciones asociadas al usuario.

Todas las consultas están envueltas en bloques try/except para capturar errores y responder de forma clara al usuario.

9. Pruebas y Validación

La fase de pruebas tuvo un papel fundamental en el desarrollo de la aplicación, ya que permitió garantizar que el sistema funcionara correctamente tanto desde un punto de vista matemático como funcional y visual. Las pruebas realizadas abarcaron diferentes niveles: validación del algoritmo, comprobación del flujo completo de la aplicación, análisis de usabilidad y verificación de la robustez ante errores. A continuación se detallan las principales áreas evaluadas:

Validación del algoritmo D'Hondt

El núcleo del proyecto —el reparto de escaños mediante D'Hondt— fue comprobado utilizando:

- Escenarios reales, basados en resultados electorales municipales y autonómicos.
- Casos extremos, tales como:
  - Partidos con 0 votos.
  - Empates entre varias formaciones.
  - Reparto con un único partido válido.
  - Umbrales muy altos o muy bajos.
- Comparación manual con cálculos externos, lo que permitió confirmar que las divisiones sucesivas y la asignación de escaños coincidían exactamente con la metodología oficial.

Estas pruebas aseguraron que el algoritmo fuese matemáticamente correcto y consistente en todos los casos.

Pruebas de validación de entrada y control de errores

Se verificaron exhaustivamente los mecanismos de validación tanto en frontend como en backend:

- Detección de valores numéricos inválidos (negativos, vacíos, no numéricos).
- Comprobación de duplicados de partidos.
- Límites del umbral electoral.
- Al menos un partido con votos > 0 antes de permitir el cálculo.
- Rechazo de simulaciones con nombres vacíos o duplicados.

Además, se verificó que:

- El backend siempre actúa como “fuente de verdad”, devolviendo errores claros y coherentes.
- El frontend resalta visualmente los errores (campos en rojo) y evita continuar la ejecución cuando los datos son incorrectos.

Pruebas visuales del hemiciclo y del pactómetro

Dado que el proyecto incluye dos componentes gráficos interactivos, se realizaron pruebas específicas para garantizar que su comportamiento fuera adecuado:

Hemiciclo (Chart.js)

- Correcta distribución proporcional de escaños.
- Correspondencia de colores entre resultados, pactómetro y gráfico.
- Funcionamiento fluido al cambiar parámetros o cargar simulaciones.
- Redimensionamiento automático en pantallas pequeñas.
- Pactómetro
- Cálculo proporcional de la barra de colores según los partidos seleccionados.
- Identificación correcta del punto de mayoría absoluta.
- Transiciones visuales suaves.
- Comprobación de combinaciones complejas con muchos partidos seleccionados o deseleccionados.

Con estas pruebas se confirmó la coherencia entre datos, tablas y representación gráfica.

Pruebas del sistema de autenticación

El sistema de usuarios fue evaluado mediante pruebas funcionales de:

- Registro de nuevos usuarios con validaciones estrictas de contraseña.
- Prevención de duplicidad de nombres de usuario.
- Inicio y cierre correcto de sesión.

- Almacenamiento persistente mediante localStorage.
- Restricción de funciones protegidas (guardar, cargar, borrar simulaciones).
- Control visual automático del estado del usuario en la interfaz.

También se testearon casos erróneos:

- Contraseñas incorrectas.
- Intentos de cargar o borrar simulaciones pertenecientes a otro usuario.
- Errores de red simulados (API caída o inaccesible).

Pruebas de responsividad y usabilidad

Se evaluó la adaptación de la interfaz a diferentes dispositivos utilizando:

- Pruebas manuales en móvil y tablet.
- Navegadores principales: Chrome, Edge y Firefox.

Se verificó especialmente:

- Reorganización de columnas en pantallas estrechas.
- Tamaño adecuado de formularios en el header.
- Accesibilidad al pactómetro y hemiciclo sin desbordamientos.
- Legibilidad en distintos tamaños de pantalla.

Los resultados demostraron que la aplicación mantiene una experiencia de usuario coherente y usable en resoluciones diversas.

10. Conclusiones

El desarrollo de la aplicación **“Calculadora Electoral Ley D’Hondt”** ha permitido materializar una herramienta práctica, funcional y pedagógicamente valiosa para el análisis de sistemas electorales basados en el método D’Hondt. La aplicación no solo cumple con su propósito principal —facilitar el cálculo rápido y preciso del reparto de escaños—, sino que destaca por ofrecer una experiencia completa que abarca desde la introducción de datos electorales hasta la visualización gráfica y la gestión de simulaciones personalizadas.

Una de las principales bondades del proyecto es su **carácter didáctico**, al permitir al usuario comprender de forma intuitiva cómo influyen variables como el número de escaños, los votos válidos, el umbral mínimo o la distribución de apoyos entre partidos. Asimismo, el uso de representaciones visuales como el hemiciclo y el pactómetro aporta claridad y facilita el análisis comparativo entre escenarios, convirtiendo la app en una herramienta versátil tanto para estudiantes como para usuarios interesados en comprender mejor el funcionamiento de los sistemas electorales proporcionales.

Desde el punto de vista técnico, el proyecto se beneficia de una arquitectura modular que separa correctamente la lógica del backend, la interfaz visual y los estilos. Esta división mejora la mantenibilidad del código y permite evolucionar la aplicación sin necesidad de rediseños completos. El backend construido en FastAPI ofrece rapidez, claridad y validaciones robustas, mientras que el

frontend utiliza JavaScript para proporcionar una interacción dinámica, fluida y fácil de extender.

A pesar de su grado de madurez, la aplicación presenta **margen de mejora y crecimiento**, lo cual abre la puerta a versiones futuras. Entre las posibles ampliaciones destacables se encuentran:

- **Sistema de autenticación avanzado**, que permita iniciar sesión en diferentes dispositivos o sesiones persistentes.
- **Paginación y filtrado avanzado** en la gestión de simulaciones, ideal para usuarios con un alto volumen de escenarios guardados.
- **Soporte multielección**, permitiendo gestionar elecciones con circunscripciones múltiples o importar datos en bloque desde archivos CSV.
- **Modo comparativo**, en el que el usuario pueda ver dos simulaciones en paralelo y analizar cambios rápidamente.
- **Optimización de accesibilidad**, incorporando funcionalidades para usuarios con necesidades especiales.
- **Internacionalización**, permitiendo que la interfaz se muestre en distintos idiomas.

En conjunto, la aplicación logra un equilibrio sólido entre sencillez de uso, claridad visual, rigor en los cálculos y escalabilidad técnica. Se presenta, por tanto, como una base robusta que satisface los objetivos del proyecto a la vez que deja abiertas diversas líneas de evolución para futuras versiones más completas, ambiciosas y profesionales.