Diseño del Experimento: Validación de Fiabilidad y Unicidad en el Sistema de Votación

1. Objetivo

El objetivo de este experimento es verificar que el sistema de votación desarrollado para la Registraduría cumple con los requisitos de **confiabilidad** (todos los votos válidos se registran sin pérdidas) y **unicidad** (ningún voto se cuenta más de una vez), en el contexto de un sistema distribuido basado en arquitectura cliente-servidor.

2. Metodología

La validación se basará en una combinación de pruebas funcionales automatizadas, observación de comportamiento del sistema, análisis de logs y verificación de consistencia entre base de datos y archivos de respaldo (CSV).

2.1. Casos de prueba diseñados

ID Caso	Descripción	Resultado esperado
TC-01	Voto válido emitido desde estación autorizada	Voto registrado en la base de datos y parcial_votes.csv
TC-02	Voto duplicado por el mismo documento	Rechazo del voto, registro en security_events
TC-03	Documento no registrado en sistema	Rechazo del voto, log en security_events con tipo DOCUMENTO_NO_REGISTRADO
TC-04	Voto desde estación válida pero votante no habilitado	Rechazo del voto, log de seguridad
TC-05	Generación de resumen de votos (reporte)	Archivo resume.csv con total de votos por candidato

3. Métricas a evaluar

- **% de votos registrados exitosamente** = (votos válidos aceptados / total de votos emitidos) × 100
- **Cantidad de votos duplicados detectados correctamente**
- **Integridad del resumen**: Validación cruzada entre resume.csv y la tabla votes
- **Consistencia en archivo parcial**: Cada línea de partial_votes.csv debe tener su entrada correspondiente en la base de datos.

4. Herramientas de apoyo

- **Base de Datos:** PostgreSQL 15
- **Framework de comunicación distribuida:** ICE ZeroC 3.7.9
- **Lenguaje de programación:** Java 17
- **Pruebas funcionales:** Cliente de prueba TestClient.java que ejecuta escenarios simulados
- **Monitoreo de resultados:** Archivos partial_votes.csv, resume.csv, tabla votes, y tabla security_events

5. Procedimiento experimental

1. Se inicia el servidor central (ServerApp) y se asegura su estado activo.
2. Se ejecuta TestClient.java, simulando múltiples escenarios de votación válidos e inválidos.
3. Se valida el estado de la base de datos con queries manuales.
4. Se revisan los archivos .csv generados.
5. Se detiene el servidor con Ctrl+C, lo cual dispara automáticamente la generación final del reporte.
6. Se compara el contenido del archivo resume.csv con la sumatoria en la tabla votes.

6. Validación de unicidad y confiabilidad

- La unicidad se garantiza mediante:
  - Validación previa al voto (has_voted = true)
  - Restricción única en la tabla votes para el campo document
- La confiabilidad se asegura mediante:
  - Confirmación transaccional (commit) en la base de datos
  - Registro redundante en archivos CSV
  - Mecanismo de reconstrucción del estado desde partial_votes.csv en caso de reinicio

7. Consideraciones sobre el Entorno de Validación

El sistema fue desarrollado y validado en su totalidad sobre un entorno **Windows 11**, utilizando:

- Java 17
- PostgreSQL 15
- ICE ZeroC 3.7.9 (instalación manual)
- Terminales de comandos para servidor y cliente local

Si bien lo ideal es realizar pruebas adicionales en entornos **Linux** para simular escenarios reales de despliegue, esto no fue posible en esta fase por restricciones de tiempo y disponibilidad.

A pesar de ello, se ha diseñado el sistema respetando principios de portabilidad:

- Se evitaron rutas absolutas
- Se empleó SQL estándar
- La lógica se encapsuló en módulos desacoplados del sistema operativo

Recomendación: En fases posteriores, se debe validar el sistema en Linux con herramientas como tc, netem, o incluso entornos Docker, para pruebas de red degradada, concurrencia y disponibilidad.