Diseño de experimento

A continuación, se describe de manera detallada el Diseño del Experimento para validar que, dentro de la arquitectura Nodo de votación → Lugar de votación → (por reliable message)→ Servicio de votación → Base de datos, se cumplan estrictamente los requisitos de confiabilidad (100% de votos registrados) y unicidad (ningún voto contado más de una vez). Este diseño se basa en la aclaración de que la comunicación interna dentro del Lugar de votación y los nodos individuales es confiable, pero la comunicación externa (Lugar de votación al Servicio de votación) puede no serlo.

Objetivos del experimento

Validar Confiabilidad (Tolerancia a Fallos en la Conexión Externa)

- Garantizar que, incluso si la conexión entre el “Lugar de votación” y el “Servicio de votación” falla intermitentemente, todos los votos generados en los Nodos de votación terminen persistidos en la Base de datos del Servicio.

Verificar Unicidad (Idempotencia y Detección de Duplicados), (Esto queda para otra entrega)

- Asegurar que, sin importar la cantidad de reenvíos (por reintentos o duplicaciones accidentales), el “Servicio de votación” persista cada voto exactamente una sola vez en la tabla de votos.

Confirmar Atributos de Calidad Específicos

- Consistencia: los resultados en la Base de datos (conteos) reflejen fielmente los votos únicos sin pérdidas ni duplicaciones.

Explicación de la arquitectura

Nodo de votación (CLI local)

- Ubicado en cada máquina de votación (por ejemplo, un terminal de la sala de cómputo).
- Provee una interfaz de línea de comandos para que el operador o ciudadano ingrese su número de documento y seleccione el candidato.
- Construye internamente un objeto de voto que incluye:
    - voterId (String)
    - candidateId (String)

Se comunica de forma directa con el “Lugar de votación” para enviar el voto.

La comunicación entre el Nodo y el Lugar se considera 100 % confiable.

Lugar de votación (Servidor ICE local)

Cada estación (mesa) ejecuta un proceso que expone la interfaz ICE VotingSite con el método sendVote(Vote vote).

Funciones principales:

- Validación local del documento (por esta entrega no).
- Control de doble votación local (por esta entrega no):
    - Mantiene un conjunto en memoria (cedulasVotadasLocal) con los documentos que ya votaron en esa estación.
    - Si el documento ya figura en ese conjunto, rechaza sin llamar al servicio central.
- Lógica de envío confiable (Reliable Messaging) hacia el Servicio de votación:
    - Cada vez que llega sendVote(vote), el VotingSiteController encola ese Vote en un BatchingController.
    - El BatchingController agrupa votos en un List<Vote> cuando se cumple cualquiera de estos dos límites:
        - Han transcurrido 500 ms desde que llegó el primer voto del lote.
        - Se acumulan 1000 votos en la cola.
    - Se envía el lote por reliable message, se reintentará el envío cada 10 segs.

Servicio de votación (Servidor central)

Expone la interfaz ICE RMDestination.receiveMessage()

Recepción de cada lote para recibir los votos:

- Envía el ACK con el uuid
- Luego verifica los duplicados mediante registerVotesFromSite(voteBatch).

Persistencia en Base de datos:

Usa un esquema relacional (H2) con una tabla Votos que incluye columnas obligatorias:

- voter_id (VARCHAR)
- candidate_id (VARCHAR)
- La primary key es la combinación de las dos anteriores

Para la entrega final: Al cerrar la jornada electoral, lee de la tabla Votos la cantidad total de votos por candidato en todo el sistema y produce el archivo resume.csv.

Base de datos (H2)

Tabla única: Votes.

Tabla única: votes, con columnas:

- voter_id VARCHAR(255)
- candidate_id VARCHAR(255)
- Clave primaria compuesta (voter_id, candidate_id).

Al término de la jornada, esta tabla es la única fuente de la verdad para:

- Validación de unicidad: no puede haber dos filas con el mismo (voter_id, candidate_id).
- Generación de conteos para el CSV final.

Para resumir, el flujo del voto es el siguiente:

1. **Ingreso del voto en el Nodo de votación:** El ciudadano introduce su voterId y selecciona candidateId en la CLI del Nodo.
2. **Envío del voto al Lugar de votación:** El Nodo crea new Vote(voterId, candidateId) y llama VotingSitePrx.sendVote(vote).
3. **Agrupación en batches:** El "Lugar de votación" recibe cada Vote y lo encola en un BatchingController. Cuando se alcanzan 1000 votos o transcurren 500 ms, se forma un List<Vote> y se invoca RMSourcePrx.sendMessage(message) para enviar ese lote a ReliableMessaging.
4. **Reliable Messaging (RMJob):** El lote se almacena en messagesPending. Cada 10 s, RMJob reenvía todos los lotes pendientes a RMDestination.receiveMessage(...). Si no recibe ACK, vuelve a intentar en el siguiente ciclo de 10 s, sin límite de intentos.
5. **Procesamiento en el Servicio de votación:** VotingServiceImpl.receiveMessage(rmessage, prx) envía prx.ack(uuid), luego verifica duplicados en H2 y persiste cada voto único en la tabla votes(voter_id, candidate_id).
6. **Verificación final y CSV:** La tabla H2 debe contener exactamente todos los votos únicos (voter_id, candidate_id) sin duplicados.

Metodología de prueba

Entorno:

- Tres instancias de lugar de votación (una por cada mesa: 1, 2 y 3). Cada una expone un servicio ICE local.
- Tres instancias de Nodo de votación (CLI), cada una asociada a su respectiva mesa y a uno de los lugares.
- Una instancia de Servicio de votación (ICE central) conectada a la Base de datos PostgreSQL.
- Una instancia de base de datos H2 que contiene únicamente la tabla Votos.

No se probará la confiabilidad de la comunicación entre el nodo de votación y el sitio de votación. Caso contrario a la comunicación entre el sitio de votación y el servicio de votación. Se puede simular interrumpiendo el proceso del servicio de votación y volviéndolo a reanudar.

Generación de datos:

Se crean 1000 votos de prueba, repartidos así:

- Mesa 1: 333 votos
- Mesa 2: 333 votos
- Mesa 3: 334 votos

Cada voto contiene:

- documento (cadena única generada automáticamente, ejemplo: "mesa1_voter123").
- candidate_id (entero entre 1 y 3)
- mesa_id (1, 2 o 3)

Para pruebas de carga ("modo fuego"), en el CLI del Nodo se elige la opción 2, que disparará K votos consecutivos con:
voterId = <nodoId> + "_voter" + i
candidateId = <nodoId> + "_candidate" + i.

Logs:

Nodo de votación (CLI):

- [INFO] [VOTE SENDED]: <voterId> <candidateId> <mesaId>

Lugar de votación:

- [INFO] Voting Site is running
- [INFO] Processing batch of <n> votes
- [INFO] [RELIABLE MESSAGING] Starting Reliable Server
- [INFO] [RELIABLE MESSAGING] Message sent: <uuid_del_lote>
- [INFO] [RELIABLE MESSAGING] Cicle <número_de_ciclo> completed. Pending messages: <N>
- [ERROR] [RELIABLE MESSAGING] <detalle_error> - msg: <uuid_del_lote>

Servicio de votación:

- [INFO] Voting Service is running
- [ERROR] Error initializing communicator: <detalle_error>
- [INFO] Received batch with size: <tamaño_del_lote_de_votos>
- [INFO] Vote Batch registered successfully
- [WARNING] Duplicated vote detected for voter: <voterId> and candidate: <candidateId>
- [ERROR] [H2] Failed saving vote: <detalle_error>

(para la entrega final) Al cierre de jornada se ejecutarán los siguientes métodos :

- START_CLOSE_ELECTION
- WRITE_RESUME_CSV
- WRITE_PARTIAL_CSV: mesaId
- END_CLOSE_ELECTION

Consultas SQL para verificar:

- Conteo total de votos persistidos:

SELECT COUNT(*) AS total_votos FROM Votes;

- Conteo de votos únicos por documento y mesa

SELECT COUNT(DISTINCT voter_id, candidate_id) AS votos_unicos, COUNT(*) AS total_votos

FROM Votes;

- Verificación de duplicados (no debe devolver filas)

SELECT voter_id, candidate_id, COUNT(*) AS ocurrencias

FROM Votes

GROUP BY voter_id, candidate_id

HAVING COUNT(*) > 1;

Escenarios de prueba

Escenario 1, Flujo ideal:

Objetivo: Verificar que, cuando el Servicio está disponible en todo momento, los 1000 votos se persisten correctamente sin duplicados ni reintentos.

Configuración:

- Base de datos funcionando con la tabla Votes.
- El Servicio de votación corre sin interrupción.
- Los tres Lugares (Mesa 1, Mesa 2, Mesa 3) están activos y en línea.
- Cada Nodo de votación contiene un lote de votos:
    - Mesa 1: 333 votos
    - Mesa 2: 333 votos
    - Mesa 3: 334 votos

Pasos a ejecutar:

1. Iniciar base de datos
2. Iniciar los lugares de votación

3. Iniciar simultáneamente los 3 nodos y sus respectivos votos
4. Esperar 20 segundos para detener los procesos
5. Corroborar estado de la base de datos

Estado final:

- [INFO] [VOTE SENDED]: <voterId> <candidateId> <mesaId> (1000 veces)
- No se muestra ningún error.

Lugares (sumados los 3):

- 3 líneas de [INFO] Voting Site is running (una por mesa).
- 3 líneas de [INFO] Processing batch of 333/333/334 votes.
- 1 línea de inicio de servidor fiable: [INFO] [RELIABLE MESSAGING] Starting Reliable Server.
- 3 líneas de [INFO] [RELIABLE MESSAGING] Message sent: <uuid>.
- 3 líneas de ciclo sin pendientes: [INFO] [RELIABLE MESSAGING] Cicle <n> completed. Pending messages: 0.
- 0 líneas de reintento ni de error.

Servicio de votación:

- 1 línea de arranque: [INFO] Voting Service is running.
- 3 líneas de recepción de lote: [INFO] Received batch with size: 333/333/334.
- 3 líneas de confirmación de guardado: [INFO] Vote Batch registered successfully.
- 0 advertencias de duplicado.
- 0 errores de base de datos.

Base de datos:

- Filas totales en Votos = 1000
- Filas únicas (document, mesa_id) = 1000
- Duplicados = 0 filas

Escenario 2, Fallo temporal del servicio:

Objetivo: Confirmar que, si el Servicio de votación se interrumpe temporalmente, los Lugares reintentan y, al restablecerse el Servicio, no se pierde ningún voto.

Configuración:

- Mismas condiciones que en el Escenario 1 (Servicio y Lugares activos).
- Cuando el Servicio haya procesado aproximadamente **500 votos**, se detendrá el proceso del Servicio durante **5 segundos**.

- Durante esos 5 segundos:
    - Los Nodos siguen enviando votos al Lugar.
    - Cada Lugar intenta enviar al Servicio y registra reintentos.
- Pasados los 5 segundos, se reinicia el proceso del Servicio.

Pasos a ejecutar:

1. Iniciar Base de datos y Servicio de votación.
2. Arrancar los tres Lugares.
3. Iniciar los tres Nodos para que envíen sus 1000 votos (50 ms entre cada uno).
4. Monitorear el log del Servicio; cuando PERSISTE_VOTO alcance **100**, detener el proceso del Servicio.
5. Durante los siguientes **5 segundos**, los Lugares registran reintentos (TIMEOUT_SERVICIO y REINTENTO_RM).
6. A los 5 segundos, reiniciar el Servicio.
7. Esperar 20 segundos adicionales para que los Lugares completen los envíos pendientes.
8. Detener los tres Lugares, los tres Nodos y finalmente el Servicio.
9. Corroborar el estado de la Base de datos.

Estado final esperado:

- **Lugares (cada uno):**
    - 3 × [INFO] Voting Site is running
    - 3 × [INFO] Processing batch of 333/333/334 votes
    - 3 × [INFO] [RELIABLE MESSAGING] Starting Reliable Server
    - 3 × [INFO] [RELIABLE MESSAGING] Message sent: <uuid> (envío inicial)
    - ≥ 3 líneas de error de reintento durante la caída: [ERROR] [RELIABLE MESSAGING] … - msg: <uuid>

      (tantas como ciclos ocurran en 5 s)
    - Tras la reanudación, de nuevo [INFO] [RELIABLE MESSAGING] Message sent: <uuid> para cada lote re-emitido.
    - Ciclos posteriores terminan con Pending messages: 0.
- **Servicio de votación:**

  - 2 arranques (antes y después de la caída): [INFO] Voting Service is running
  - 3 × recepción de lote: [INFO] Received batch with size: 333/333/334
  - 3 × confirmación: [INFO] Vote Batch registered successfully
  - Posibles advertencias si algún voto ya estaba en la BD: [WARNING] Duplicated vote detected for voter: …
- **Base de datos:**
  - Filas totales = 1000
  - Filas únicas = 1000
  - Duplicados = 0 filas

Escenario 3, Duplicado de votos:

Objetivo: Comprobar que, si un nodo envía 20 votos duplicados (mismo voter_id y candidate_id) dentro del mismo lote, el Servicio solo persiste cada par (voter_id,candidate_id) una vez.

Configuración:

- Solo participa la Mesa 1.
- Nodo 1 lee un archivo con 120 votos organizado como sigue:
  1. Primeros 50 votos con documentos únicos.
  2. Reenvío de 20 de esos 50 (duplicados).
  3. Envío de los 50 votos restantes
- El Servicio permanece activo sin interrupciones.

Pasos a ejecutar:

1. Iniciar la base de datos y el Servicio de votación.
2. Arrancar Lugar 1.
3. Ejecutar Nodo 1 (versión que reenvía los 20 duplicados).
4. Esperar ≈ 5 s después de que el nodo termine los 120 envíos.
5. Detener Lugar 1 y el Servicio.
6. Consultar la base.

Estado final esperado:

- **Nodo 1:**

    - [INFO] [VOTE SENDED]: … (120 veces)
- **Lugar 1:**
    - [INFO] Voting Site is running
    - [INFO] Processing batch of 120 votes
    - [INFO] [RELIABLE MESSAGING] Starting Reliable Server
    - [INFO] [RELIABLE MESSAGING] Message sent: <uuid>
    - [INFO] [RELIABLE MESSAGING] Cicle 1 completed. Pending messages: 0
- **Servicio de votación:**
    - [INFO] Voting Service is running
    - [INFO] Received batch with size: 120
    - [INFO] Vote Batch registered successfully
    - [ERROR] [H2] Failed saving vote: <PK violation> (20 veces, uno por cada duplicado)
- **Base de datos:**
    - Filas totales = 100
    - Filas únicas = 100
    - Duplicados = 0 filas

Escenario 4, Caída y recuperación del servicio:

Objetivo: Verificar que, si el Servicio de votación se detiene mientras los Lugares siguen recibiendo votos, los lotes pendientes se re-envían y todos los votos terminan persistidos exactamente una vez cuando el Servicio vuelve a estar en línea.

Configuración:

- Tres mesas (Lugar 1, Lugar 2, Lugar 3) planean enviar **1000 votos** (333 + 333 + 334).
- El servicio estará activo al inicio. Se detendrá después de persistir el primer lote y permanecerá apagado 10 s.
- Pasados los 10 segundos, se reinicia el Servicio en el mismo puerto y con la misma Base de datos.
- Se reenvían los lotes pendientes
- Se revisa la base de datos

Pasos a ejecutar:

1. Iniciar Base de datos y Servicio de votación.
2. Arrancar los tres Lugares.
3. Iniciar los tres Nodos para que envíen 1000 votos con intervalo de **50 ms**.
4. Monitorear BD; cuando PERSISTE_VOTO alcance **200**, detener el proceso del Servicio.
5. Durante los siguientes **10 segundos**, los Lugares registran TIMEOUT_SERVICIO y REINTENTO_RM.
6. Tras 10 segundos, reiniciar el Servicio.
7. Los Lugares retoman los envíos pendientes y reciben ACK_SERVICIO hasta completar 1000.
8. Esperar **5 segundos** tras el último ACK.
9. Detener los tres Lugares, los tres Nodos y finalmente el Servicio.
10. Corroborar el estado de la Base de datos.

Estado final esperado:

- **Lugares (cada mesa):**
  - [INFO] Voting Site is (3 líneas)
  - [INFO] Processing batch of 333 / 333 / 334 votes (3 líneas)
  - [INFO] [RELIABLE MESSAGING] Starting Reliable Server (3 líneas)
  - [INFO] [RELIABLE MESSAGING] Message sent: <uuid> (3 líneas – envío inicial)
  - Mensajes con el servicio caído:

    [ERROR] [RELIABLE MESSAGING] … - msg: <uuid>
  - Con el servicio recuperado:

    [INFO] [RELIABLE MESSAGING] Message sent: <uuid>

    [INFO] [RELIABLE MESSAGING] Cicle <n> completed. Pending messages: 0
- **Servicio de votación:**
  - [INFO] Voting Service is running
  - [INFO] Received batch with size: 333
  - [INFO] Vote Batch registered successfully

    - Se detiene el servicio 10 segundos
    - [INFO] Received batch with size: 333
    - [INFO] Vote Batch registered successfully
    - [INFO] Received batch with size: 334
    - [INFO] Vote Batch registered successfully
- **Base de datos:**
    - Filas totales = 1 000
    - Filas únicas = 1 000
    - Duplicados = 0 filas