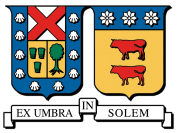


Universidad Técnica Federico Santa María
Sistemas Distribuidos
Segundo Semestre 2025

Laboratorio 2: Sistemas Distribuidos

CyberDay Distribuido

Profesor: Jorge Díaz
Ayudantes de Lab: Felipe Marchant V. & Javiera Cortés U.

Septiembre 2025

1 Objetivos del Laboratorio

- Comprender y aplicar **tolerancia a fallos** en sistemas distribuidos.
- Implementar un **sistema de almacenamiento replicado**.
- Diseñar un **broker centralizado** con validación de identidades y gestión de suscripciones.

2 Introducción

El comercio electrónico enfrenta uno de sus mayores desafíos durante eventos de alto tráfico como el **CyberDay**, donde miles de ofertas deben procesarse y distribuirse en tiempo real a millones de clientes.

En este laboratorio, se simulará un ecosistema con:

- **Pocos productores (tiendas)** que generan ofertas.
- Un **broker central** que valida, almacena y distribuye las ofertas.
- Un **sistema de almacenamiento distribuido** replicado en múltiples nodos.
- **Muchos consumidores** con preferencias que esperan recibir solo las ofertas relevantes.

El sistema debe ser **tolerante a fallos**: la caída de un nodo de base de datos o de un consumidor no debe detener la operación.

3 Tecnologías

- El lenguaje de programación a utilizar es **Go**.
- Para la comunicación síncrona se utilizará **gRPC**.
- Para la definición de mensajes síncronos se usará **ProtocolBuffers**
- Para distribuir el programa se utilizará **Docker**

4 Laboratorio

4.1 Contexto

En el mundo del comercio electrónico, los grandes eventos de descuentos como el **CyberDay** se han convertido en verdaderas batallas digitales. Miles de usuarios se conectan de manera simultánea para encontrar las mejores ofertas, mientras que las tiendas deben procesar, almacenar y distribuir enormes volúmenes de datos en tiempo real sin detenerse, incluso si alguno de sus sistemas falla.

4.2 Explicación

Para este laboratorio, deberán simular este escenario con tres gigantes ficticios del retail nacional:

- **Riploy**
- **Falabellox**
- **Parisio**

Cada uno de estos retailers es un **productor de ofertas**, enviando constantemente promociones de distintas categorías (electrónica, moda, hogar, deportes, belleza, infantil, entre otras) hacia un **broker central**.

El broker no solo debe validar la identidad de cada productor, sino también **filtrar, almacenar y redistribuir las ofertas** hacia un sistema de **base de datos distribuida** que funciona bajo el modelo de **replicación DynamoDB**.

Finalmente, en el otro extremo, existe una gran cantidad de **consumidores de datos**, cada uno con **preferencias personalizadas**: algunos buscan exclusivamente productos de electrónica, otros ropa con descuentos superiores al 40%, o incluso ofertas de una tienda específica. Cada consumidor espera recibir únicamente lo que realmente le interesa, lo que obliga al broker a manejar una **tabla de suscripciones** y filtrar la información en tiempo real.

Durante la simulación, deberán además considerar **fallos inevitables**:

- Un nodo de base de datos que deja de responder.
- Consumidores que se desconectan en plena transmisión.
- Replicaciones incompletas que luego deben sincronizarse.

El reto será demostrar que, incluso bajo estas condiciones, el sistema **se mantiene disponible, consistente bajo las reglas de DynamoDB (N, W, R), y entrega notificaciones relevantes a cada consumidor sin pérdida de datos**.

4.3 Productores de Ofertas (Riploy, Falabellox y Parisio)

Los **Productores de Ofertas** corresponden a las tres principales tiendas de retail ficticias del sistema: **Riploy, Falabellox y Parisio**. Su misión es generar ofertas en tiempo real durante el evento CyberDay y transmitirlas al Broker Central para su almacenamiento y distribución a los consumidores.

Cada productor simula una tienda distinta, con su propio catálogo y volúmenes de ofertas. Estos mensajes deben seguir una estructura uniforme y validada por el broker.

Funciones de los Productores:

- **Generación de Ofertas:** Cada productor crea ofertas en categorías variadas (ej. Electrónica, Hogar, Moda, Deportes, Belleza, Infantil, etc.).
- **Envío al Broker:** Las ofertas se envían al Broker Central mediante comunicación gRPC.
- **Identificación Segura:** Cada productor debe registrarse en el broker antes de operar, de lo contrario, sus mensajes serán rechazados.
- **Idempotencia:** Cada oferta debe incluir un identificador único (oferta_id) que permita al broker descartar mensajes duplicados.

Reglas de Generación de Ofertas: Cada productor debe contar con un catálogo base de productos (este será dado por los ayudantes). A partir de dicho catálogo, las ofertas se generarán de forma periódica en tiempo real, utilizando variaciones aleatorias en precio y stock. Las ofertas serán generadas por los mismos productores considerando un descuento aleatorio entre 10% a 50% de su precio base. La frecuencia de emisión debe ser continua y realista (por ejemplo, 1 oferta cada 1–2 segundos, con intervalos aleatorios). Todas las ofertas deben incluir un stock estrictamente mayor que cero y una fecha correspondiente al momento exacto de su generación (basado en `time.Now()` en Go). Ofertas con stock nulo o con fecha inválida no deben ser transmitidas al broker.

Estructura del Mensaje de Oferta (ejemplo):

Listing 1: Ejemplo de Oferta

```
{
  "oferta_id": "uuidv4",
  "tienda": "Riploy",
  "categoria": "Electronica",
  "producto": "Laptop ASOS",
  "precio": 499990,
  "stock": 30,
  "fecha": "2025-04-15"
}
```

4.4 Broker Central

El **Broker Central** es el núcleo del sistema. Su misión principal es recibir ofertas de los productores, validarlas, almacenarlas de manera distribuida y redistribuirlas a los consumidores relevantes según sus preferencias.

El broker actúa como **punto de control y consistencia**, asegurando que las reglas de replicación **N=3, W=2, R=2** se cumplan, y gestionando la relación entre productores, base de datos distribuida y consumidores.

Funciones del Broker Central:

- **Registro y Autenticación:** Mantiene un registro de productores, nodos DB y consumidores activos. Rechaza entidades no autorizadas.
- **Recepción de Ofertas:** Recibe todas las ofertas enviadas por los productores.
- **Almacenamiento Distribuido:** Envía cada oferta a los 3 nodos de base de datos (N=3), esperando confirmación de al menos 2 (W=2).
- **Recuperación de Datos:** Ante consultas de consumidores, coordina lecturas en los nodos DB, aceptando solo si al menos 2 coinciden (R=2).
- **Distribución a Consumidores:** Reenvía las ofertas únicamente a los consumidores que cumplen con los filtros de suscripción (categoría, tienda, precio máximo, etc.).
- **Generación de Reporte Final:** Crea un archivo **Reporte.txt** con estadísticas de ofertas procesadas, entregas realizadas y fallos ocurridos.

4.5 Nodos de Base de Datos Distribuida (DB1, DB2, DB3)

Los **Nodos** de **Base de Datos** simulan el modelo de replicación eventual de DynamoDB. Existen 3 nodos en total (**DB1, DB2, DB3**) y cada operación de escritura o lectura debe cumplir con los parámetros de consistencia **N=3, W=2, R=2**.

Funciones de los Nodos DB:

- **Almacenamiento Replicado:** Guardan cada oferta recibida desde el broker.
- **Confirmación de Escritura:** Devuelven ACK al broker si logran almacenar la oferta.
- **Replicación Eventual:** Si un nodo queda fuera de línea, debe resincronizarse con sus pares al volver a estar activo.
- **Respuesta a Lecturas:** Devuelven información histórica al broker. La lectura solo es válida si al menos 2 nodos coinciden en los datos.

Fallos Simulados:

- **Caída Temporal:** Un nodo puede dejar de responder, afectando la escritura/lectura.
- **Recuperación:** Al reintegrarse, debe recibir el backlog de datos pendientes desde los otros nodos.

4.6 Consumidores de Ofertas (3 grupos × 4 instancias)

Los **Consumidores de Ofertas** representan a los clientes conectados al CyberDay. Existen 3 entidades principales de consumidores (ej. **Electrónica, Moda, Hogar**), cada una compuesta por 4 instancias, totalizando 12 mini-consumidores.

Cada mini-consumidor posee **preferencias preestablecidas**, entregadas en un archivo de configuración externo. Estas preferencias determinan qué ofertas le interesan, por ejemplo:

- Categoría específica (Ej. Electrónica).
- Tienda específica (Ej. Falabellox).
- Precio máximo (Ej. menor a $50.000).
- Combinaciones (Ej. categoría Deportes con precio menor a $20.000).

Funciones de los Consumidores:

- **Registro Inicial:** Se registran en el broker al iniciar la ejecución.
- **Recepción de Ofertas:** Reciben las ofertas que coinciden con sus filtros, en tiempo real.
- **Almacenamiento Local:** Guardan las ofertas recibidas en un archivo CSV único por consumidor (ej: consumidor_electronica_3.csv).
- **Consultas de Histórico:** Pueden solicitar al broker el histórico de ofertas, lo que activa una lectura distribuida en los nodos DB (R=2).
- **Recuperación de Fallos:** Si un consumidor se desconecta, al reanudarse debe poder recuperar su histórico desde la base de datos.

Ejemplo de archivo de configuración consumidores.csv:
El archivo **consumidores.csv** tendrá el formato id_consumidor,categoria,tienda,precio_max, donde estos elementos se separan por una coma (,). En caso de haber más de un elemento por posición (ej: se quiere más de una categoría y/o tienda), estas serán separadas por un punto y coma (;). En caso de elegir cualquier valor para tienda (quiere recibir de cualquier tienda) o categoría (quiere recibir de cualquier categiría), esto tendrá un valor null, en caso de querer un valor específico, se especificarán los valores como en el ejemplo:

Listing 2: Ejemplo de Configuracion

```
{
id_consumidor,categoria,tienda,precio_max
C-E1,Electronica,null,null
C-E2,Electronica,Falabellox,null
C-E3,Electronica,Riploy;Falabellox,null
C-E4,Electronica,Parisio,100000
C-H3,Hogar;Electronica;Deportes,null,300000
C-H4,null,null,null
}
```

Interpretación de consumidores.csv

- Categoría fija, cualquier tienda, cualquier precio → C-E1 (solo Electrónica, sin importar tienda ni precio).
- Categoría fija + 1 tienda específica → C-E2 (Electrónica solo de Falabellox sin importar el precio).
- Categoría fija + varias tiendas → C-E3 (Electrónica solo de Riploy y Falabellox, sin importar el precio).
- Categoría fija + tienda + precio bajo → C-E4 (Electrónica solo de Parisio y $\leq$ \$100.000).
- Múltiples categorías + múltiples tiendas → C-H3 (Hogar o Electrónica o Deportes, de Riploy y Falabellox, $\leq$ \$300.000).
- Sin restricciones → C-H4 (recibe absolutamente todo).

4.7 Fases de la Ejecución

El sistema deberá desarrollarse y ejecutarse siguiendo una serie de fases claramente definidas. Cada una de estas fases deberá poder observarse durante la ejecución del laboratorio.

1. **Registro de Entidades**
   - Todos los productores, nodos de base de datos y consumidores deben registrarse en el broker antes de comenzar a operar.
   - El broker debe validar identidades y rechazar mensajes de entidades no registradas.
2. **Producción de Ofertas**
   - Riploy, Falabellox y Parisio generan y envían sus ofertas al broker mediante gRPC, de manera periódica (flujo continuo en tiempo real). Cada productor debe utilizar su propio catálogo base y variar dinámicamente precio y stock de las ofertas generadas.
   - Cada oferta debe contener un identificador único, categoría, tienda, producto, precio, stock y fecha.
3. **Escritura Distribuida (N=3, W=2)**
   - El broker reenvía cada oferta a los 3 nodos de base de datos.
   - La operación de escritura solo se considera exitosa si al menos 2 nodos (W=2) confirman el almacenamiento.
   - Los nodos deben implementar replicación eventual para resincronizarse tras fallas.
4. **Notificación a Consumidores**
   - El broker consulta la tabla de suscripciones basada en consumidores.csv.
   - Cada oferta se reenvía únicamente a los consumidores cuyas preferencias la acepten.
   - Los consumidores almacenan las ofertas en un archivo CSV propio.

5. **Simulación de Fallos**
    - Durante la ejecución, al menos un nodo de base de datos debe fallar temporalmente.
    - Además, uno o más consumidores pueden desconectarse en plena transmisión.
    - El sistema debe continuar funcionando y los nodos caídos deben poder recuperarse.
6. **Lectura Distribuida (R=2)**
    - Los consumidores pueden consultar histórico de ofertas al broker.
    - El broker coordina la lectura en al menos 2 nodos (R=2) y devuelve el resultado solo si hay consenso.
7. **Recuperación y Resincronización**
    - Los nodos de base de datos caídos deben resincronizarse al volver.
    - Los consumidores caídos deben recuperar histórico desde los nodos de lectura.
8. **Generación de Reporte Final**
    - Al concluir, el broker debe generar un archivo Reporte.txt con un resumen detallado de la ejecución.

4.8 Reporte Final

El archivo **Reporte.txt** debe ser generado por el broker al finalizar la ejecución. Este archivo debe incluir como mínimo las siguientes secciones:

1. **Resumen de Productores**
    - Cantidad total de ofertas enviadas por cada productor.
    - Cantidad de ofertas aceptadas por el broker.
2. **Estado de Nodos de Base de Datos**
    - Estado final de cada nodo (activo, caído).
    - Número de escrituras exitosas y fallidas.
3. **Notificaciones a Consumidores**
    - Cantidad de ofertas recibidas por cada consumidor según sus preferencias.
    - Confirmación de que cada consumidor generó su archivo CSV.
4. **Fallos y Recuperaciones**
    - Listado de fallos ocurridos (nodos caídos, consumidores desconectados).
    - Resultados de la resincronización.
5. **Conclusión**
    - Indicación de si el sistema logró mantenerse disponible y consistente bajo las reglas especificadas.

4.9 Categorías Disponibles

Las ofertas generadas por los productores deberán pertenecer exclusivamente a alguna de las siguientes categorías:

- Electrónica
- Moda
- Hogar
- Deportes
- Belleza
- Infantil
- Computación
- Electrodomésticos
- Herramientas
- Juguetes
- Automotriz
- Mascotas

El broker y los consumidores deberán validar que todas las ofertas pertenezcan a una de estas categorías. Ofertas con categorías no reconocidas deben ser descartadas.

5 Uso de Docker

Todos los procesos del laboratorio deben ejecutarse en contenedores Docker dentro de las máquinas virtuales. Todas las entidades deben estar aisladas en contenedores separados.

Ejemplo de distribución de las entidades en las máquinas virtuales:

- MV1: Riploy / BD1 / Consumidor2
- MV2: Falabellox / BD2 / Consumidor3
- MV3: Parisio / BD3
- MV4: Broker / Consumidor1

6 Restricciones

Las librerías de Golang permitidas son:

- `fmt`
- `log`
- `errors`
- `os`
- `io`
- `bufio`
- `time`
- `sync`
- `math/rand`
- `encoding/csv`
- `encoding/json`
- `net`
- `context`
- `flag`
- `strconv`
- `strings`
- `grpc`

Además de las necesarias para el funcionamiento de gRPC y Protocol Buffers. Todo uso de librerías externas que no se han mencionado en el enunciado debe ser consultado con los ayudantes.

7 Consideraciones

- **Comunicación:**
  - Todas las comunicaciones entre entidades deben realizarse estrictamente mediante **gRPC** y **Protocol Buffers**.
  - No está permitido el uso de colas de mensajes externas como RabbitMQ, Kafka, NATS, etc.
- **Broker Central:**
  - El broker es el único punto autorizado para interactuar con productores, nodos de base de datos y consumidores.
  - El broker debe validar la identidad de cada entidad antes de aceptar conexiones o mensajes.
  - Ofertas con identificadores duplicados deben ser descartadas para garantizar idempotencia.
- **Nodos de Base de Datos:**
  - El sistema debe respetar los parámetros de consistencia: **N=3, W=2, R=2**.
  - Si un nodo falla, las escrituras deben continuar funcionando mientras se cumpla $W = 2$.
  - Si un nodo se reincorpora, debe resincronizarse automáticamente con sus pares.
  - Lecturas históricas deben validar que al menos $R = 2$ nodos entreguen resultados coincidentes.
  - Lecturas históricas solo deben usarse cuando un consumidor se reintegre luego de haberse caido.
- **Consumidores:**
  - Cada consumidor debe recibir únicamente las ofertas que coincidan con las preferencias predefinidas en el archivo `consumidores.csv`.
  - Los filtros de categoría y tienda pueden aceptar múltiples valores o `null` (sin restricción).
  - Cada consumidor debe generar un archivo CSV único con las ofertas recibidas.
  - Los consumidores desconectados deben recuperar su histórico al reintegrarse.
- **Archivos de Configuración:**
  - El archivo `consumidores.csv` será entregado por los ayudantes. A la hora de evaluar se usará una configuración distinta así que no debe estar "hardcodeado".
  - Las ofertas generadas por los productores deben incluir exclusivamente las categorías listadas en el enunciado.
  - Ofertas con categorías no válidas o con campos faltantes deben ser rechazadas por el broker.
  - Los productores deben generar ofertas únicamente a partir de catálogos válidos, respetando las categorías listadas en el enunciado. La generación debe ser dinámica y continua, asegurando que toda oferta posea un stock mayor a 0 y fecha actualizada al momento de emisión.

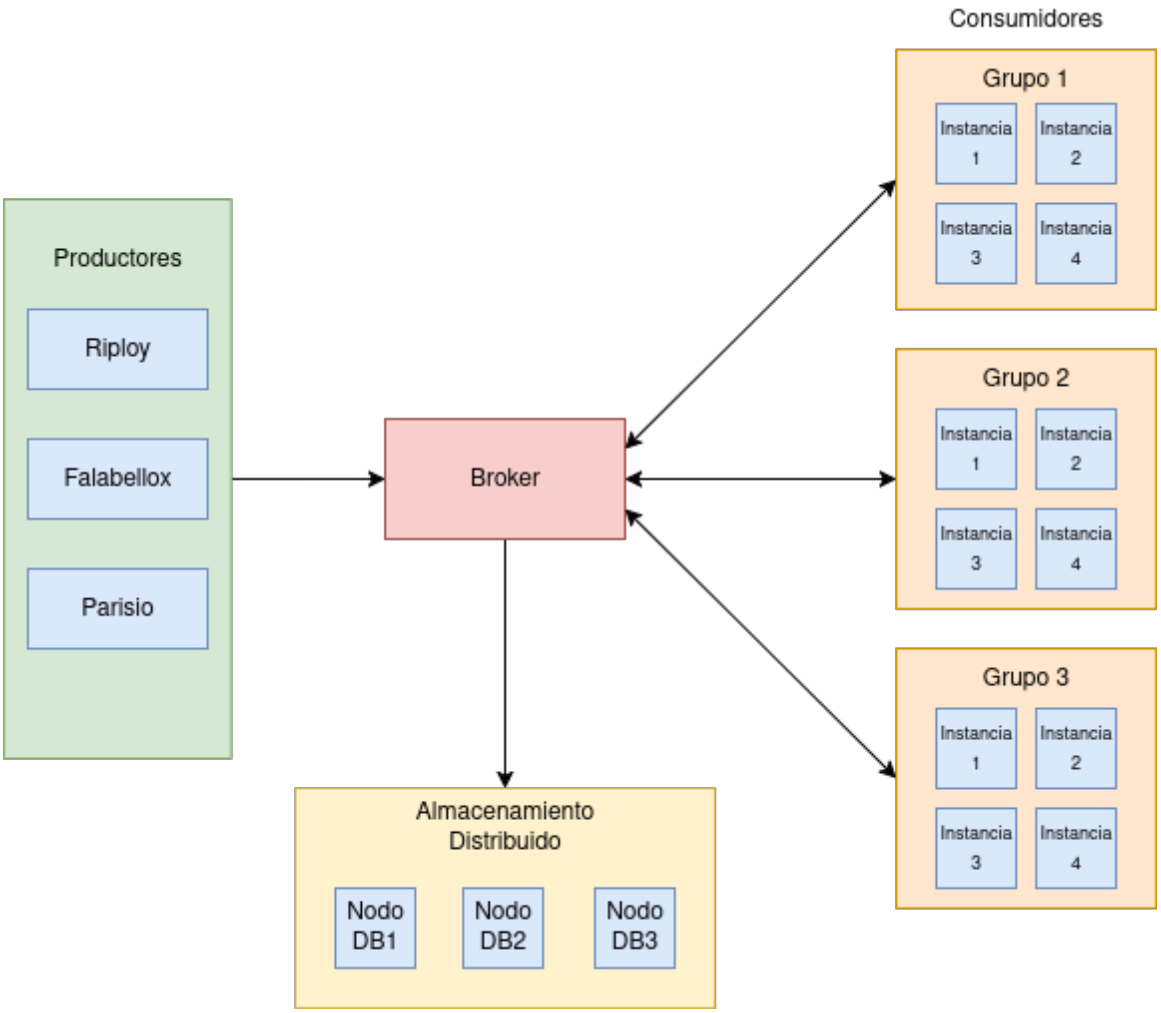
- **Contenedores Docker:**
  - Cada entidad debe ejecutarse en un contenedor Docker independiente.
  - Se debe incluir un `Makefile` con atajos como:
    - `make docker-VM1`
    - `make docker-VM2`
    - `make docker-VM3`
    - `make docker-VM4`
- **Logs y Reporte:**
  - Todas las entidades deben generar logs claros sobre su ejecución (conexiones, mensajes recibidos, fallos simulados).
  - El broker debe generar el archivo `Reporte.txt` al finalizar, con el detalle solicitado en el enunciado.
- **Consultas y Soporte:**
  - Se realizará una ayudantía para detallar el enunciado y resolver dudas generales.
  - Las consultas deben realizarse en el foro habilitado en Aula o al correo de los ayudantes:
    - `felipe.marchantv@usm.cl`
    - `javiera.cortes@usm.cl`
  - No se responderán consultas enviadas con menos de **48 horas antes de la fecha de entrega original**. En caso de extensión de plazo, para esta regla se considerará la fecha original de entrega.

8 Reglas de Entrega

- La tarea se entrega en grupos de 2 a 3 personas previamente asignados en aula.
- La fecha de entrega es el día **domingo 19 de octubre 23:59 hrs.**.
- La tarea se revisará en las máquinas virtuales, por lo que los archivos necesarios para la correcta ejecución de esta deben estar en ellas. Recuerde que el código debe estar identado, comentado, sin warnings y sin errores.
- Además de los códigos en las máquinas virtuales y github, deberán subir un archivo comprimido que contenga todos los códigos desarrollados en carpetas separadas por entidad en formato ***.zip*** con el nombre **GrupoXX-LabY.zip**. Donde XX es el número de su grupo e Y es el número del laboratorio. Ejemplo: *Grupo07-Lab2.zip*.
- **Debe** dejar un `MAKEFILE` o similar en cada máquina virtual asignada a su grupo para la ejecución de cada entidad.
- Debe dejar un **README** en la entrega asignada a su grupo con nombre y rol de cada integrante, además de la información necesaria para ejecutar los archivos.
- El uso de **Docker** es **obligatorio**. Teniendo nota 0 aquellas implementaciones que no hagan uso de Docker.
- Las faltas de README y/o copias serán evaluadas con nota 0.
- Todos los descuentos y aclaraciones necesarias están disponibles en el documento "**Reglas de Laboratorio**" disponible en el Aula virtual. Favor revisarlo con prontitud para evitar inconvenientes.

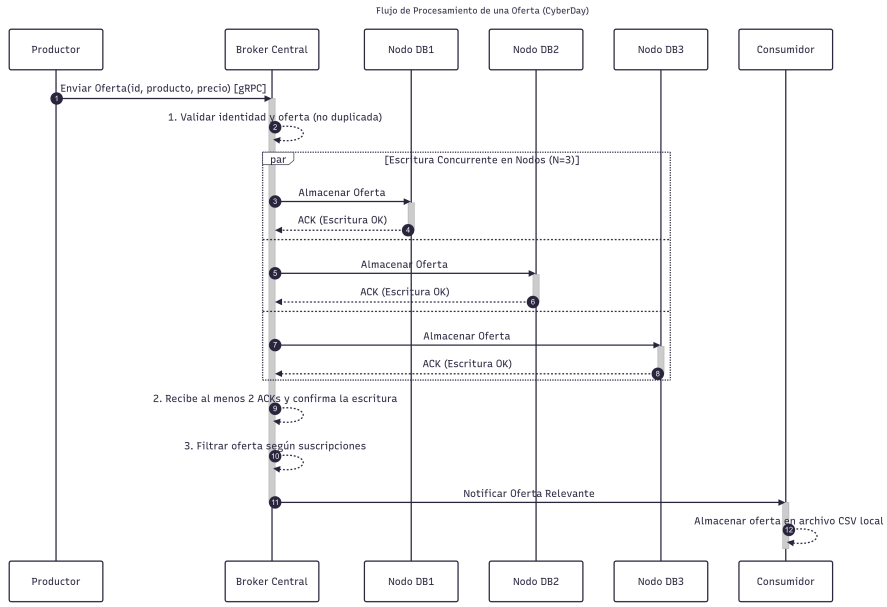
9 Anexos

9.1 Entidades



9.2 Diagramas de Secuencia

9.2.1 Procesamiento de una oferta



9.2.2 Recuperación de Histórico de un Consumidor