



## Diseño Preliminar Cloud Provider Analytics

72.80 Big Data

BENGOLEA, IÑAKI (63515), ..... IBENGOLEA@ITBA.EDU.AR  
IJAS, CHRISTIAN (63555), ..... CIJJAS@ITBA.EDU.AR  
LOPEZ MENARDI, FELIX (62707), ..... FLOPEZMENARDI@ITBA.EDU.AR

13 de Octubre, 2025

## 1. Arquitectura de Alto Nivel

**Patrón elegido:** Lambda Architecture.

**Justificación:** Lambda es la elección adecuada porque necesitamos combinar dos objetivos: exactitud en el histórico (maestros, facturación) vía batch y baja latencia para métricas operativas (usage/costos incrementales) vía streaming. La separación de capas nos permite recalcular sobre el dataset completo para corregir datos tardíos o cambios de lógica y “canonizar” resultados en la Serving, mientras la capa speed entrega estimaciones inmediatas; esto se alinea naturalmente con su diseño de zonas Bronze/Silver/Gold, Spark (batch y Structured Streaming con watermark/dedup/checkpoint) y Serving en Cassandra. En cambio, Kappa fuerza todo a un único pipeline de eventos: las correcciones y cambios de lógica exigen replays completos, mayor gestión de estado, versionado y backfills complejos, elevando el riesgo operativo y la carga de desarrollo para garantizar que las cifras finales sean precisas y auditables. En este contexto académico, el pequeño trade-off de duplicación de lógica en Lambda se compensa con resiliencia, verificabilidad y una implementación más directa de los requisitos y componentes ya definidos.

**Alcance:** para la pre-entrega se implementará un alcance acotado que permitirá demostrar el patrón Lambda evitando complicaciones que puedan surgir al manejar un proyecto con volumen excesivo.

- **Streaming:** procesamiento de un *subset* (10–20 archivos .jsonl) desde `usage_events_stream/` con `schema` explícito, `withWatermark`, `dropDuplicates(event_id)` y `checkpointing`. El *backfill* completo queda documentado como trabajo futuro.
- **Batch completo:** lectura de 3–4 fuentes CSV representativas (por ejemplo, `customers_orgs.csv`, `users.csv`, `billing_monthly.csv`, `support_tickets.csv`) hacia *Bronze → Silver → Gold*.
- **Serving:** exposición reducida en AstraDB/Cassandra suficiente para ejecutar las 5 consultas requeridas.

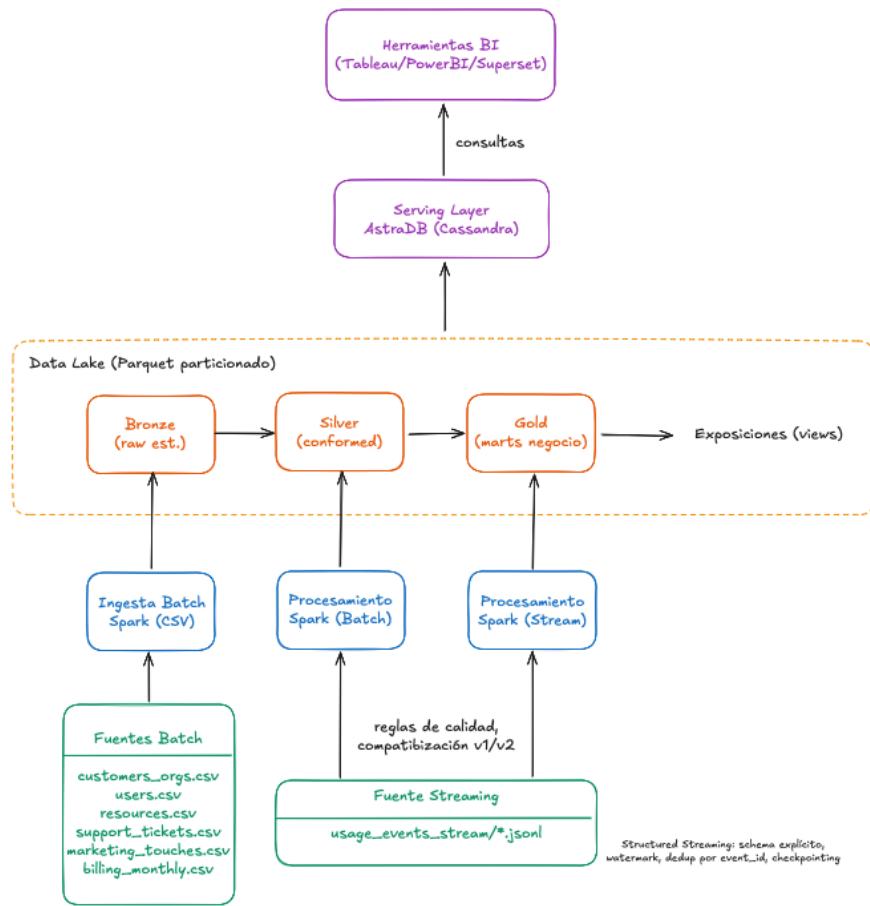


Figura 1: Diagrama detallado de la arquitectura

## 2. Mapeo de Requisitos a Componentes

Requisito (negocio/5V)	Componente / Decisión de Diseño	Justificación
<b>Near real-time</b> sobre <code>usage/cost_usd_increment</code>	<b>Spark Structured Streaming</b> con esquema explícito, <code>withWatermark</code> , dedup por <code>event_id</code> , <code>checkpointing</code> .	Cumple requisito de velocidad y maneja <i>late data</i> ; idempotencia para reprocesos.
<b>Batch</b> (CRM/NPS/facturación)	<b>Spark Batch → Parquet partitionado</b> (p. ej., <code>date=/service=</code> ).	Eficiente para volúmenes; estandariza tipos en Bronze.
Evolución de esquema (~45 días: <code>schema_version=2</code> , <code>carbon_kg</code> , <code>genai_tokens</code> )	<b>Silver</b> con control de versiones y compatibilización v1/v2.	Conformance y estabilidad aguas abajo.
Calidad de datos / anomalías	Reglas en Bronze/Silver, <i>quarantine</i> aparte; métodos: z-score/MAD/percentiles.	Aísla datos malos y habilita depuración.
Escalabilidad/Disponibilidad en serving	<b>AstraDB (Cassandra)</b> , modelado <b>query-first</b> con particiones adecuadas.	BD distribuida, escalable y tolerante a fallos.
Valor de negocio (FinOps/Soporte/Producto)	<b>Marts Gold:</b> <code>org_daily_usage_by_service</code> , <code>revenue_by_org_month</code> , <code>tickets_by_org_date</code> , <code>genai_tokens_by_org_date</code> .	Orientados a consultas mínimas de demo.

Cuadro 1: Decisiones de diseño y su justificación

### 3. Flujo de Datos (Data Pipeline)

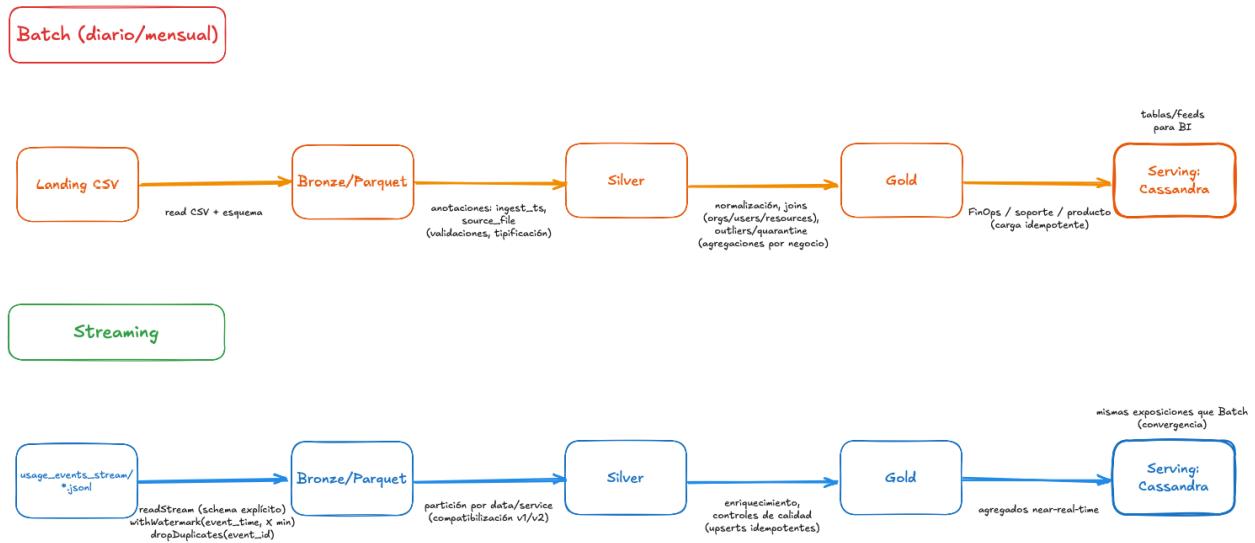


Figura 2: Diagrama detallado de flujo de datos

### 4. Asunciones y Riesgos Iniciales (con mitigaciones)

- **Volumen/Velocidad:** ~60 días de eventos; micro-lotes en carpeta stream. **Mitigación:** `maxFilesPerTrigger`, particionamiento por `date/service`.
- **Calidad:** nulos, tipos ambiguos (p. ej., `value string`), costos negativos y spikes. **Mitigación:** casteo con fallback, reglas  $[-0,01, +\infty]$ , outlier flags, quarantine.
- **Evolución de esquema:** `schema_version v1/v2` con columnas nuevas. **Mitigación:** rutas de compatibilización en Silver y backfill controlado.
- **Eventos tardíos:** uso de `withWatermark` y ventanas, re-procesos idempotentes. **Mitigación:** checkpointing y claves naturales/upserts.
- **Serving/latencia:** diseño query-first en Cassandra. **Mitigación:** elección de clave de partición y clustering según consulta.
- **Complejidad Lambda:** duplicación parcial de lógica. **Mitigación:** compartir librerías de transformaciones y contratos de esquema.
- **Volumen de streaming:** para la demo se procesará una muestra (10–20 archivos) a fin de demostrar watermark, deduplicación e idempotencia. El backfill total quedará documentado como trabajo futuro.

- **Columnas opcionales** (`genai_tokens`, etc...): si no están presentes en el período muestreado, las vistas *Gold* mostrarán 0 o valores nulos con nota metodológica correspondiente.

## 5. Estimación de Esfuerzo y Recursos (Rough)

### Equipo y roles:

- Data Engineer (1): ingesta Batch y Streaming, diseño de zonas Bronze/Silver/Gold, definición de esquemas, reglas de calidad y quarantine, compatibilización schema\_version, especificación de particionado y checkpointing, diseño de upserts (idempotencia) hacia Gold/Serving, carga a Cassandra
- Data Analyst/BI (1): definir métricas de negocio y consultas demo, validar granos y dimensiones de Gold, preparar mockups de paneles/queries sobre Serving, verificar consistencia (ejemplos numéricos) y narrar los insights esperados.
- Tech Lead (1): definir patrón (Lambda/Kappa), estándares de calidad, criterios de particionado y idempotencia, revisar decisiones de Serving y validar riesgos. Facilita bloqueos y asegura coherencia técnica.

La planificación se orienta a una entrega final funcional end-to-end, con trabajo en paralelo entre roles y un margen explícito para gestión del riesgo. Los esfuerzos se expresan en persona-día (pd) con rangos, y se acompaña de una calendarización comprimida pensada para un equipo de tres integrantes.

### Esfuerzo por hitos:

Hito	Esfuerzo base (pd)	Colchón 20 % (pd)
M0 Kickoff & contrato de datos	0.5–0.75	+0.10–0.15
M1 Ingesta Batch → Bronze	0.8–1.2	+0.16–0.24
M2 Silver (casts/joins, v1/v2, 3 reglas + quarantine)	0.8–1.2	+0.16–0.24
M3 Streaming (10–20 jsonl: schema+watermark+dedup+checkpoint)	0.8–1.2	+0.16–0.24
M4 Gold (5 vistas/tablas de negocio)	0.8–1.2	+0.16–0.24
M5 Serving Cassandra (keyspace + 5 feeds/tablas)	0.4–0.6	+0.08–0.12
M6 5 consultas CQL + capturas de evidencia	0.4–0.6	+0.08–0.12
M7 Evidencias (conteos, tiempos, decisiones)	0.4–0.6	+0.08–0.12
<b>Subtotal base (pd)</b>		<b>4.9–7.0</b>
<b>Subtotal con colchón (pd)</b>		<b>5.9–8.4</b>

**Lectura:** se presenta un rango base y un colchón del 20 % para absorber variabilidad (deriva de esquema, valores atípicos, micro-tuning de particiones).

**Totales aproximados:** 20 pd (DE ~15; BI ~2.5; TL parcial ~2.5 para decisiones y revisiones).