2. ARQUITECTURA E INTEGRACIÓN DE DATOS

Para armar un sistema sólido de recolección y análisis de datos en un casino online, diseñaría una arquitectura por capas que me ha funcionado de maravilla en proyectos similares con mucho tráfico.

Capa de Ingesta de Eventos Montaría un sistema doble para capturar eventos. Para el tracking en tiempo real, usaría Apache Kafka como columna vertebral - es súper confiable cuando necesitas procesar miles de eventos por segundo. Los eventos del frontend (clicks, tiempo jugando, apuestas) los agarraría con un SDK propio que manda los datos a topics de Kafka específicos por tipo de evento. Para el procesamiento batch, programaría jobs nocturnos que saquen logs históricos y transacciones del día anterior con Apache Spark.

La estructura de eventos la dejaría simple pero completa: user_id, session_id, timestamp, event_type, game_id, bet_amount, device_info, y un campo JSON flexible para metadata específica. Esta flexibilidad me ha salvado un montón de veces cuando el negocio pide trackear algo nuevo sin tocar todo el schema.

Integración con Sistemas Externos Para los pagos, armaría conectores específicos para cada proveedor. Con PayU, que es clave para LATAM, pondría webhooks para agarrar las transacciones en tiempo real y un job diario que reconcilie con su API REST por si algún webhook se cae. Con Stripe haría algo parecido, pero aprovecharía su sistema de eventos que está buenísimo para mantener sincronizado el estado de cada transacción.

Para Meta y Google Ads, la cosa cambia. Acá no necesito tiempo real, así que programaría extracciones cada 4 horas con sus APIs. Los datos de campañas, impresiones, clicks y conversiones los guardaría temporalmente en una zona de staging antes de transformarlos.

Con herramientas de engagement como Braze, haría una integración de ida y vuelta: mandaríamos eventos de comportamiento en tiempo real para disparar campañas, y sacaríamos diariamente los resultados de las campañas para ver qué tan bien funcionan.

Unificación en Data Warehouse Me iría por BigQuery porque procesa queries complejas sin que tengas que andar preocupándote por administrar clusters. Armaría el warehouse con tres capas bien claras:

- Raw: datos tal cual llegan de las fuentes
- Staging: datos limpios y normalizados

- Marts: tablas optimizadas para análisis específicos (player_analytics, game_performance, marketing_attribution)

El truco está en crear una tabla de usuarios unificada que conecte todas las identidades del jugador entre sistemas. Uso una mezcla de email hasheado, huella digital del dispositivo y cookies para hacer este match con 95% de precisión.

Orquestación del Pipeline Airflow sería mi orquestador principal - está super probado y lo manejo al dedillo. Armaría los DAGs por área de negocio: payments_pipeline, marketing_pipeline, player_behavior_pipeline. Cada uno con sus propias ventanas de procesamiento y dependencias.

dbt lo metería para todas las transformaciones dentro de BigQuery. Me encanta cómo documenta automáticamente de dónde viene cada dato y te deja testear la calidad. Crearía modelos incrementales para las tablas de hechos grandes y snapshots para dimensiones que cambian despacito como player_segments.

Para las integraciones más estándar, Fivetran me aceleraría todo. Lo usaría especialmente para Google Ads y Meta - ya tienen conectores que funcionan de una y me ahorro semanas de desarrollo.