



UNIVERSIDAD TÉCNICA
FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTROTECNIA E INFORMÁTICA
INGENIERÍA EN INFORMÁTICA

Diseño de la Base de Datos y Gestión de la Información en Plataforma SaaS de Gestión de Flota Vehicular

Benjamín Andrés Villouta Lobos

bvilloutavl@gmail.com

Lais San Martín Navarro
Profesora Guía

Rodrigo Pinochet Ponce
Profesor Correferente

Resumen: La gestión de flotas vehiculares requiere información consistente para coordinar solicitudes, vehículos y conductores, mantener trazabilidad y apoyar decisiones de gestión. Esta tesina presenta el diseño e implementación de la arquitectura de datos de FlotUp, una plataforma SaaS multi-organización originada a partir de necesidades operacionales de la Municipalidad de Viña del Mar y concebida como un producto adaptable a instituciones que administran flotas internas propias. El trabajo integra una base transaccional PostgreSQL/PostGIS, un modelo organizacional flexible basado en unidades jerárquicas, una base analítica separada con hechos y dimensiones, un proceso ETL trazable y una capa analítica orientada a indicadores. Además, define una estrategia de respaldo y recuperación para la base operacional, el Data Warehouse y los archivos del sistema. La validación funcional considera integridad, aislamiento entre organizaciones, calidad del ETL y consistencia de indicadores; los objetivos de continuidad quedan sujetos al despliegue y a pruebas periódicas de restauración. El resultado es una arquitectura que sostiene la operación actual de FlotUp y entrega una base verificable para analítica, auditoría y evolución del producto.

Palabras clave: arquitectura de datos, Data Warehouse, gestión de flota, modelado dimensional, multi-tenancy, ETL, respaldos.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Diseño de la base de datos y gestión de la información en plataforma SaaS de gestión de flota vehicular

Nombre del candidato(a): Benjamín Andrés Villouta Lobos

Carrera / Grado: Ingeniería en Informática

Campus: Sede Viña del Mar

Departamento: Electrotecnia e Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Lais San Martín Navarro, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por (marcar una opción):

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 16-09-2026 Firma: 

Estudiante o Candidato(a):

Fecha: 16-09-2026 Firma: 

Índice

Contenido

1	Introducción	5
1.1	Contexto y antecedentes	5
1.2	Definición del problema	6
1.3	Breve descripción de la propuesta de solución	6
1.4	Justificación del proyecto	7
1.5	Objetivos	7
1.6	Metodología y plan de trabajo	8
2	Marco teórico	9
2.1	Arquitectura de datos en sistemas SaaS	9
2.2	Modelos transaccionales OLTP	9
2.3	Modelos analíticos OLAP y Data Warehouse	9
2.4	Modelado dimensional	10
2.5	Multi-tenancy y seguridad de datos	10
2.6	Gobernanza, calidad y continuidad	10
2.7	Niveles conceptual, lógico y físico	10
3	Marco tecnológico y justificación	11
3.1	PostgreSQL y PostGIS	11
3.2	Django y GeoDjango	11
3.3	Separación entre base operacional y analítica	11
3.4	Redis y Celery	11
3.5	Visualización analítica	12
3.6	Respaldo y recuperación	12
4	Diseño de la arquitectura de datos	12
4.1	Diseño OLTP: cuentas y organización	14
4.2	Diseño OLTP: gestión de flota	17
4.3	Diseño OLTP: solicitudes y viajes	19
4.4	Diseño OLAP y Data Warehouse	22
4.5	Integración y procesamiento ETL	23
4.6	Integridad, rendimiento y evolución del esquema	24
4.7	Indicadores analíticos	24
4.8	Seguridad, retención y continuidad	25
5	Validación y resultados	25

5.1 Plan de validación	26
5.2 Resultados del modelo transaccional	26
5.3 Resultados del modelo analítico y ETL	26
5.4 Resultados de indicadores analíticos	26
5.5 Definición y estado de validación de la continuidad	28
5.6 Limitaciones de la validación	28
6 Conclusiones y trabajo futuro	29
6.1 Conclusiones principales	29
6.2 Aprendizajes	29
6.3 Trabajo futuro	29
7 Referencias	30
Anexo A. Diccionario resumido y metadatos del modelo OLTP	31
Anexo B. Diccionario resumido y metadatos del Data Warehouse	34
Anexo C. Matriz de trazabilidad	38

1 Introducción

La flota vehicular interna constituye un recurso estratégico para instituciones públicas y privadas que disponen de vehículos propios para coordinar actividades en terreno, movilizar equipos y utilizar responsablemente los recursos disponibles. En este tipo de operación intervienen funcionarios que requieren movilización, administradores que gestionan la disponibilidad de la flota y conductores encargados de ejecutar los viajes.

FlotUp surge a partir de una necesidad observada en la Ilustre Municipalidad de Viña del Mar, donde la solicitud y asignación de vehículos dependía de llamadas, registros manuales y conocimiento distribuido entre distintas personas. Esta forma de coordinación dificultaba conocer la disponibilidad real, reconstruir las decisiones adoptadas y disponer de información histórica confiable. Actualmente, la Municipalidad constituye el caso de referencia y la única organización con la que se ha trabajado.

La solución se concibe como una plataforma SaaS multi-organización para clientes que administran una flota interna propia. El presente trabajo se concentra en la arquitectura de datos que sustenta la plataforma: una base transaccional para la operación diaria, una base analítica para el análisis histórico, un proceso ETL trazable y una estrategia de respaldo y recuperación. El propósito es que la información generada pueda mantenerse íntegra, aislada por organización, disponible para auditoría y preparada para la obtención de indicadores analíticos.

El documento desarrolla el contexto y los antecedentes del proyecto, define el problema y presenta brevemente la solución propuesta. Luego expone la justificación y los objetivos del trabajo, los fundamentos teóricos y tecnológicos, los modelos OLTP y OLAP, la integración y continuidad de la información, la validación realizada, las conclusiones y los anexos técnicos.

1.1 Contexto y antecedentes

El proyecto FlotUp se origina en un desafío de coordinación vehicular planteado por la Ilustre Municipalidad de Viña del Mar. La institución dispone de vehículos y conductores para responder a necesidades de traslado de sus funcionarios, pero la gestión de solicitudes se apoyaba en comunicaciones telefónicas, anotaciones y conocimiento operativo disperso. La información necesaria para coordinar un viaje no se encontraba centralizada ni seguía una estructura común.

En el proceso participan tres actores principales. El funcionario informa una necesidad de traslado, el administrador revisa la solicitud y coordina los recursos disponibles, y el conductor ejecuta el viaje. Cada decisión vincula datos sobre el solicitante, su unidad organizacional, el vehículo, el conductor, los lugares involucrados, los horarios y los estados del servicio. Por ello, el problema no consiste únicamente en digitalizar un formulario, sino en representar de manera consistente el ciclo operacional completo.

Actualmente, la Municipalidad de Viña del Mar es el único caso real de referencia del proyecto. Sin embargo, el mismo problema puede presentarse en organizaciones que operan vehículos institucionales propios, como universidades, redes de centros de salud familiar u otras entidades públicas y privadas con necesidades recurrentes de traslado interno. Estos ejemplos corresponden a ámbitos potenciales de aplicación y no a clientes

actuales de FlotUp. Esta proyección explica que la plataforma haya sido diseñada con aislamiento de datos y configuración independiente para cada organización.

1.2 Definición del problema

La coordinación manual de los traslados produce información fragmentada y una alta dependencia de comunicaciones verbales. Sin una fuente central resulta difícil responder quién solicitó un vehículo, qué recurso fue asignado, quién lo condujo, cuándo se ejecutó el servicio, qué ocurrió durante el viaje y cuál fue su resultado. Esta falta de trazabilidad limita el seguimiento diario y la reconstrucción posterior de las decisiones operacionales.

Desde la perspectiva de los datos, la plataforma debe resolver simultáneamente varias exigencias. Debe mantener separada la información de cada organización, representar estructuras internas que pueden adoptar diferentes niveles jerárquicos, reutilizar ubicaciones, controlar relaciones temporales como la vinculación entre conductor y vehículo, y conservar la integridad entre solicitudes, viajes, paradas y eventos. Un modelo rígido o sin reglas explícitas trasladaría las inconsistencias del proceso manual a la solución digital.

También existe una necesidad analítica. La operación cotidiana genera información sobre demanda, asignaciones, duración, cumplimiento, uso de vehículos y eventos del viaje, pero una base orientada únicamente a transacciones no resulta adecuada para consolidar y consultar ese historial. Además, el valor administrativo y de auditoría de los datos exige considerar respaldo, retención y recuperación. El problema abordado por esta tesina es, por tanto, diseñar una arquitectura que sostenga la operación de FlotUp y transforme sus registros en información trazable, analizable y recuperable.

1.3 Breve descripción de la propuesta de solución

FlotUp digitaliza el ciclo de los traslados mediante una plataforma web para funcionarios y administradores, complementada por una aplicación móvil para conductores. Los funcionarios registran solicitudes; los administradores gestionan la flota, la estructura organizacional y las políticas de asignación; y los conductores se vinculan a un vehículo, reciben viajes, validan su inicio y registran hitos operacionales.

En el modelo de datos, cada cliente se representa mediante Organization, que actúa como tenant y delimita la pertenencia de los registros. OrgUnitType y OrgUnit permiten configurar estructuras internas de distinta profundidad, mientras que Location centraliza lugares reutilizables asociados con unidades organizacionales o perfiles. Esta organización general se anticipa aquí para contextualizar la solución; sus reglas y relaciones se desarrollan con mayor detalle en las secciones 4.1.3 y 4.1.4.

La arquitectura separa dos responsabilidades. La base OLTP registra la operación diaria normalizada y aplica reglas de integridad y aislamiento entre organizaciones. La base OLAP conserva hechos y dimensiones orientados al análisis histórico. Un proceso ETL transfiere y transforma la información de forma trazable, mientras que los indicadores analíticos y la estrategia de respaldo y recuperación completan el tratamiento del ciclo de vida de los datos.

FlotUp fue desarrollado como un proyecto de equipo. El aporte específico de esta tesina consiste en diseñar e implementar esa arquitectura de datos, incluyendo el modelo

transaccional, la representación flexible de organizaciones y ubicaciones, el Data Warehouse dimensional, el proceso ETL, los indicadores analíticos y los criterios de continuidad de la información.

1.4 Justificación del proyecto

La justificación de este proyecto se encuentra en la necesidad de transformar una operación basada en registros manuales y conocimiento disperso en una fuente de información confiable para la gestión. Digitalizar una solicitud de viaje no es suficiente si los datos generados no poseen estructura, reglas de pertenencia, trazabilidad y mecanismos que aseguren su continuidad en el tiempo. Sin una arquitectura de datos explícita, la plataforma podría registrar operaciones, pero no necesariamente entregar información consistente para auditar, analizar o mejorar la gestión de la flota.

En FlotUp, esta necesidad se vuelve especialmente relevante por su carácter SaaS multi-organización. Cada cliente administra una flota interna propia, con usuarios, unidades organizacionales, vehículos, conductores, solicitudes y viajes que deben mantenerse separados de los demás clientes. Por ello, el modelo transaccional debe resguardar la integridad de los datos, evitar duplicidades, representar adecuadamente las relaciones operativas y proteger el aislamiento entre organizaciones.

La separación entre la base transaccional OLTP y la base analítica OLAP se justifica porque la operación diaria y el análisis histórico responden a necesidades distintas. La aplicación requiere registrar solicitudes, asignaciones, viajes y eventos mediante transacciones breves y consistentes; en cambio, la analítica requiere consolidar información histórica para calcular indicadores de demanda, ejecución, utilización y desempeño. Mantener ambas responsabilidades desacopladas permite que la operación no dependa de consultas analíticas costosas y que la capa de análisis evolucione según sus propios requerimientos.

Finalmente, la información gestionada tiene valor operacional, administrativo y de auditoría. Por esta razón, el diseño de la arquitectura de datos debe considerar no solo tablas y relaciones, sino también procesos ETL, criterios de calidad de datos, respaldo, retención y recuperación ante fallas. De esta forma, el proyecto contribuye a que la plataforma no solo registre viajes, sino que conserve información confiable y utilizable para la toma de decisiones.

1.5 Objetivos

1.5.1 Objetivo general de la solución

Desarrollar una plataforma SaaS multi-organización para la gestión digital de flotas vehiculares internas, capaz de centralizar solicitudes, asignaciones, ejecución de viajes y seguimiento operativo, manteniendo información trazable para fortalecer el control operacional, la transparencia y la toma de decisiones.

1.5.2 Objetivo general del trabajo

Diseñar e implementar la arquitectura de datos de FlotUp, integrando un modelo transaccional para la operación, un modelo dimensional separado para analítica, un proceso ETL, indicadores analíticos y lineamientos de respaldo y recuperación orientados a la continuidad de la información.

1.5.3 Objetivos específicos del trabajo

1. Analizar los requerimientos de información de la gestión de flota, considerando actores, procesos operativos y necesidades de trazabilidad.
2. Diseñar e implementar el modelo OLTP con criterios de integridad, flexibilidad organizacional y aislamiento entre organizaciones.
3. Diseñar e implementar el modelo OLAP para representar demanda, viajes, eventos operacionales y estado diario de la flota.
4. Implementar un proceso ETL incremental, ejecutable por organización y con registro de sincronizaciones.
5. Definir indicadores analíticos de demanda, ejecución, utilización y desempeño a partir del modelo dimensional.
6. Definir la estrategia de respaldo, retención, recuperación y validación de continuidad para los datos críticos de la plataforma.

1.6 Metodología y plan de trabajo

El trabajo se desarrolló de forma iterativa dentro del flujo ágil utilizado en FlotUp. Para levantar y validar la información se revisaron el repositorio del proyecto, las migraciones de base de datos, las pruebas automatizadas, la documentación en Confluence y el seguimiento de tareas en Jira.

El plan de trabajo se concretó en las siguientes etapas:

1. Levantamiento del problema, actores involucrados, reglas de negocio y necesidades de información para la gestión de la flota.
2. Modelado conceptual de procesos y entidades, y posterior traducción a modelos lógicos relacionales y dimensionales.
3. Implementación física del modelo en Django y PostgreSQL/PostGIS, considerando una base analítica separada para el Data Warehouse.
4. Construcción y verificación del proceso ETL, incorporando controles de calidad de datos e indicadores analíticos.
5. Definición de criterios de aislamiento de datos, retención, respaldo, recuperación y del protocolo de pruebas de continuidad.
6. Validación mediante pruebas automatizadas, comandos de calidad, revisión del aislamiento entre organizaciones y evidencia de funcionamiento de los procesos principales.

2 Marco teórico

El diseño de la arquitectura de datos se sustenta en conceptos propios de los sistemas SaaS, el procesamiento transaccional, los sistemas analíticos, el modelado dimensional, el multi-tenancy, la calidad de datos y la continuidad operacional. Este marco teórico presenta dichos conceptos para establecer una base común desde la cual justificar las decisiones de diseño desarrolladas en los capítulos posteriores.

2.1 Arquitectura de datos en sistemas SaaS

El modelo SaaS, Software as a Service, corresponde a una forma de entrega de software en la que una aplicación se ofrece como servicio a través de una infraestructura compartida. En este tipo de arquitectura, una misma solución puede atender a múltiples organizaciones, reutilizando componentes comunes sin mezclar la información de cada cliente. En FlotUp, la organización constituye la frontera principal de pertenencia de los datos, por lo que todo acceso operacional o analítico debe resolverse dentro de ese alcance.

2.2 Modelos transaccionales OLTP

Los sistemas OLTP, Online Transaction Processing, están orientados al registro de operaciones frecuentes, concurrentes y de corta duración^[1]. En este tipo de sistemas se prioriza la integridad, la consistencia y la respuesta eficiente ante acciones como crear solicitudes, cambiar estados, vincular conductores y vehículos o registrar eventos operacionales.

En una base de datos relacional, estas operaciones se representan mediante tablas, claves primarias, claves foráneas, restricciones e índices. La normalización permite reducir redundancia y evitar anomalías de inserción, actualización o eliminación. Estos principios permiten que la base operacional mantenga relaciones consistentes entre organizaciones, usuarios, vehículos, solicitudes, viajes y eventos.

2.3 Modelos analíticos OLAP y Data Warehouse

Los sistemas OLAP, Online Analytical Processing, están orientados al análisis de información histórica mediante consultas agregadas, comparaciones y exploración de tendencias. A diferencia de los sistemas OLTP, su objetivo principal no es registrar transacciones individuales, sino facilitar preguntas de gestión sobre periodos, categorías, actores y resultados.

Un Data Warehouse reorganiza información proveniente de la operación para análisis. Esta separación permite que la base transaccional mantenga su foco en la consistencia de la operación diaria, mientras la base analítica se optimiza para indicadores, agregaciones y consultas históricas. Esta distinción permite analizar demanda, ejecución de viajes, eventos y estado de flota sin sobrecargar el modelo operacional.

2.4 Modelado dimensional

El modelado dimensional organiza la información analítica separando las medidas cuantitativas en tablas de hechos y el contexto descriptivo en tablas de dimensiones^[2]. Una tabla de hechos almacena eventos medibles, como solicitudes, viajes o estados diarios. Una dimensión describe el contexto de análisis, como tiempo, organización, vehículo, conductor, estado o motivo.

La definición de la granularidad de cada hecho es central. Una fila puede representar una solicitud, un viaje, un evento operacional o el estado diario de un vehículo. Mantener estos granos separados evita sumar conceptos no equivalentes y permite construir indicadores con significado consistente^[2].

2.5 Multi-tenancy y seguridad de datos

El multi-tenancy corresponde a una arquitectura en la que una misma aplicación atiende a múltiples clientes o tenants. En este contexto, el aislamiento de datos es una condición crítica: cada organización debe acceder únicamente a su propia información^[9]. Por ello, las consultas deben filtrarse y las mutaciones deben validarse según la organización autenticada.

La seguridad debe fallar de forma cerrada: si no es posible resolver la organización o el rol del usuario, el sistema no debe entregar datos globales. Esta regla se aplica a las plataformas web y móvil, las API, los WebSockets, las consultas analíticas y los procesos ETL.

2.6 Gobernanza, calidad y continuidad

La gobernanza de datos comprende definiciones, responsables, reglas de calidad, trazabilidad y documentación asociada al uso de la información^[10]. En una plataforma operacional, estos elementos permiten que los datos mantengan significado consistente y puedan ser auditados o interpretados correctamente.

La continuidad de datos se relaciona con la capacidad de conservar, respaldar y recuperar información crítica ante fallas. Para ello se definen criterios de retención, objetivos de recuperación y pruebas de restauración. Estos conceptos son relevantes porque la información de solicitudes, viajes, eventos y analítica tiene valor operacional, administrativo y de auditoría.

2.7 Niveles conceptual, lógico y físico

El modelo conceptual identifica procesos y conceptos del dominio sin depender de una tecnología específica: organización, unidad organizacional, usuario, ubicación, vehículo, solicitud, viaje y evento. El modelo lógico define entidades, atributos, cardinalidades, hechos, dimensiones y reglas de integridad. El modelo físico concreta estas decisiones en modelos Django, tablas PostgreSQL/PostGIS, claves, índices, migraciones y enrutamiento de bases.

Esta separación permite justificar cada entidad por su función dentro del dominio y, posteriormente, explicar cómo se implementa en la plataforma. También permite distinguir entre el significado conceptual del dato y su representación física en la base de datos.

3 Marco tecnológico y justificación

3.1 PostgreSQL y PostGIS

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, orientado a integridad, consistencia transaccional y extensibilidad^[1]. En FlotUp se utiliza como base para el modelo operacional y para el Data Warehouse, debido a su soporte de claves, restricciones, transacciones, consultas SQL y herramientas de respaldo.

PostGIS es una extensión espacial para PostgreSQL que incorpora tipos de datos geográficos y operaciones sobre ubicaciones. Su uso permite representar orígenes, destinos, ubicaciones favoritas, posiciones de vehículos y puntos de operación dentro de la misma base relacional. Esta decisión es coherente con una plataforma de gestión de flota, donde la información operacional y territorial forman parte del mismo dominio de datos^{[1][4]}.

3.2 Django y GeoDjango

Django es un framework web de alto nivel basado en Python que permite estructurar aplicaciones mediante modelos, vistas, rutas, formularios, migraciones y pruebas^[5]. Django organiza el dominio en aplicaciones como cuentas, flota, viajes y analítica, manteniendo una separación clara entre responsabilidades del sistema.

El ORM de Django permite representar entidades del dominio como modelos versionados, los cuales se traducen en tablas, relaciones y restricciones dentro de la base de datos. GeoDjango extiende esta integración para trabajar con tipos espaciales de PostGIS desde la aplicación, lo que facilita el manejo de ubicaciones y coordenadas sin abandonar el modelo relacional^[5].

3.3 Separación entre base operacional y analítica

La implementación utiliza conexiones diferenciadas para la base operacional y la base analítica del Data Warehouse. Esta separación permite que el modelo OLTP mantenga su foco en registrar la operación diaria, mientras el modelo OLAP se orienta a consultas históricas e indicadores.

En Django, esta decisión se implementa mediante un router de base de datos que dirige los modelos analíticos a la conexión correspondiente. Con ello se hace explícito el límite entre operación y analítica, se reduce el acoplamiento entre cargas de trabajo y se facilita la aplicación de políticas de respaldo y recuperación diferenciadas.

3.4 Redis y Celery

Redis es una base de datos en memoria utilizada frecuentemente como intermediario de mensajería y soporte para comunicación en tiempo real^[7]. En FlotUp cumple un rol de apoyo para operaciones que requieren intercambio rápido de mensajes y coordinación entre procesos.

Celery es una herramienta de ejecución de tareas asíncronas en Python. Su uso permite ejecutar procesos de sincronización o carga de datos fuera del ciclo normal de una petición web. En el contexto de esta tesina, resulta relevante porque el proceso ETL

puede ejecutarse de forma desacoplada, programable y sin depender de mantener abierta una petición HTTP^{[6][7]}.

3.5 Visualización analítica

La visualización analítica corresponde a la capa que permite presentar indicadores calculados a partir del Data Warehouse. En FlotUp se implementa mediante bibliotecas web para construir gráficos de series, distribuciones, comparaciones y vistas territoriales de puntos de operación.

Estas bibliotecas no forman parte del diseño físico de las bases de datos, pero justifican la necesidad de contar con información analítica estructurada, agregable y filtrada por organización. Su función es consumir datos ya preparados por el modelo analítico, sin modificar la estructura OLTP ni reemplazar el proceso ETL.

3.6 Respaldo y recuperación

pgBackRest es una herramienta especializada para respaldos y restauración de bases PostgreSQL^[8]. Permite administrar respaldos físicos, repositorios, políticas de retención y procesos de restauración. Su uso es pertinente para FlotUp porque la plataforma depende de información operacional y analítica que debe conservarse ante fallas.

La estrategia de continuidad considera respaldos físicos de PostgreSQL y archivado WAL, Write-Ahead Logging. Este mecanismo permite registrar cambios de la base de datos y habilita escenarios de recuperación a un punto en el tiempo, siempre que exista un respaldo base y los archivos WAL necesarios. En el contexto de esta tesina, esta tecnología respalda la definición de lineamientos de recuperación para la base operacional y el Data Warehouse^{[3][8]}.

4 Diseño de la arquitectura de datos

Este capítulo presenta el diseño central de la arquitectura de datos de FlotUp. La solución se organiza en una base transaccional, una base analítica, un proceso de integración y una estrategia de continuidad. La base operacional conserva el estado vigente de la plataforma y aplica las reglas necesarias para ejecutar solicitudes y viajes. El Data Warehouse reorganiza una selección de esos datos para consultar periodos, comparar resultados y calcular indicadores sin trasladar esa carga a la operación diaria.

El diseño parte de dos decisiones transversales. La primera es utilizar Organization como límite de pertenencia, seguridad y análisis: cada unidad organizacional, perfil, vehículo, solicitud, viaje y proceso operacional debe poder resolverse dentro de una organización. La segunda es conservar identificadores de origen en el Data Warehouse, lo que permite relacionar cada fila analítica con el registro operacional que la produjo, auditar su procedencia y repetir una sincronización sin crear duplicados.

Antes de describir cada modelo en detalle, la Tabla 4-1 resume la arquitectura de datos por capas funcionales. Esta vista permite distinguir qué parte del diseño sostiene la

operación diaria, qué parte permite el análisis histórico y qué componentes aseguran la integración y continuidad de la información. Para mantener la legibilidad de los diagramas, los atributos con sus tipos y tamaños se documentan en los Anexos A y B.

Tabla 4-1. Mapa general de la arquitectura de datos.

Fuente: elaboración propia.

Capa	Tablas o componentes principales	Función dentro de la arquitectura
Organización y acceso	Organization, OrgUnitType, OrgUnit, Profile, DriverProfile, Location, OrgUnitLocation, ProfileLocation	Define los clientes del SaaS, su estructura interna, usuarios, roles y ubicaciones de operación.
Gestión de flota	VehicleType, Vehicle, VehicleBinding, FleetBlock	Administra los activos de flota, su disponibilidad, tipificación y vínculo temporal entre conductor y vehículo.
Solicitudes y viajes	TravelRequest, Stop, Trip, TripEvent	Registra la demanda de traslado, la ejecución efectiva del viaje, sus paradas y eventos operacionales.
Data Warehouse	DimTiempo, DimOrganizacion, DimVehiculo, DimConductor, DimUnidadOrganizacional, FactSolicitudes, FactViajes, FactEventosViaje, FactEstadoFlotaDiario	Reorganiza información operacional en dimensiones y hechos para análisis histórico e indicadores.
Integración y continuidad	populate_dwh, DWHSyncLog, respaldos PostgreSQL, WAL, pgBackRest	Sincroniza datos entre OLTP y OLAP, registra ejecuciones y define mecanismos de respaldo y recuperación.

La separación por capas evita presentar el modelo como un conjunto aislado de tablas. Cada grupo cumple una responsabilidad específica dentro del sistema: representar la organización del cliente, administrar la flota, registrar el ciclo de solicitudes y viajes, consolidar información analítica y proteger la continuidad de los datos. A partir de esta estructura se desarrollan las principales decisiones de diseño del modelo OLTP y OLAP.

4.1 Diseño OLTP: cuentas y organización

4.1.1 Del modelo conceptual al modelo físico

Conceptualmente, una organización contiene una estructura jerárquica de unidades internas, usuarios y lugares de operación. Esta estructura es relevante porque las solicitudes, los perfiles, los vehículos y los indicadores deben poder analizarse según la unidad organizacional a la que pertenecen. En el modelo lógico, cada unidad tiene un tipo configurable y puede depender de otra unidad. Las ubicaciones son entidades reutilizables vinculadas mediante relaciones intermedias. En el modelo físico, estas decisiones se implementan con Organization, OrgUnitType, OrgUnit, Location, OrgUnitLocation, ProfileLocation y Profile.

La Tabla 4-2 sintetiza las entidades principales del bloque de organización y acceso. Su propósito es mostrar la función de cada tabla antes de profundizar en las reglas de integridad y relaciones.

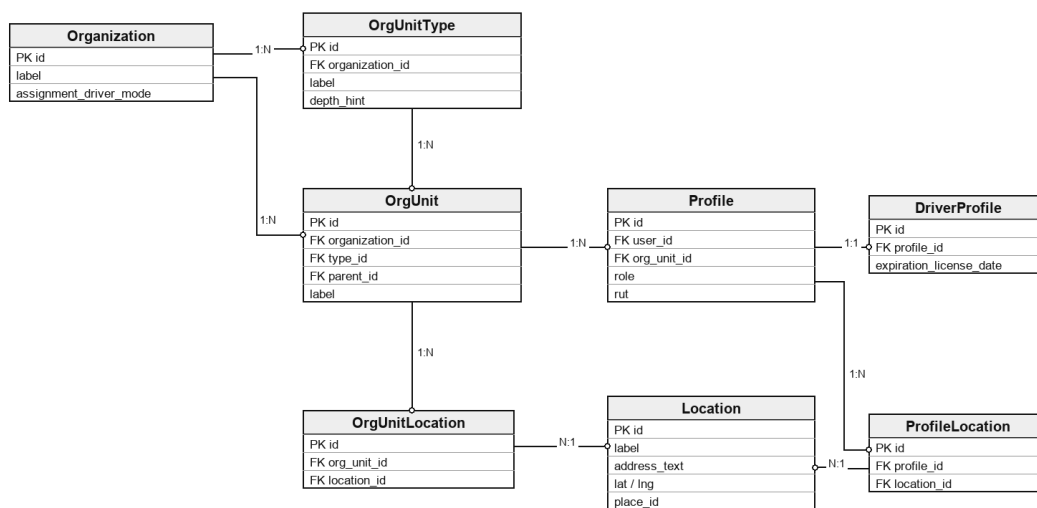


Figura 4-1. Modelo simplificado de organización y acceso. Fuente: elaboración propia.

Tabla 4-2. Principales tablas del modelo de organización y acceso.

Fuente: elaboración propia.

Tabla	Qué representa	Decisión de diseño
Organization	Cliente o entidad que utiliza FlotUp.	Actúa como frontera de datos, permisos y análisis.
OrgUnitType	Tipo configurable de unidad interna.	Permite que cada organización defina su propia estructura sin tablas rígidas.
OrgUnit	Nodo concreto dentro de la estructura organizacional.	Usa relación jerárquica recursiva para representar organigramas variables.
Location	Lugar reutilizable con dirección, coordenadas y referencia geográfica.	Se mantiene independiente de la organización para evitar duplicación y permitir reutilización.
OrgUnitLocation	Relación entre una unidad organizacional y una ubicación.	Expresa lugares de operación de una unidad sin duplicar ubicaciones.
ProfileLocation	Ubicaciones favoritas asociadas a un perfil.	Separa preferencias personales de la estructura institucional.
Profile	Extensión del usuario con rol y unidad interna.	Permite resolver pertenencia, permisos y alcance de consulta desde la unidad organizacional.
DriverProfile	Información operacional específica de un conductor.	Permite especializar al conductor sin duplicar datos comunes del usuario.

La traducción entre niveles evita que la estructura interna de un cliente quede codificada como una jerarquía universal. Organization representa el tenant; OrgUnitType define el vocabulario que utiliza ese tenant; y OrgUnit representa cada nodo concreto. De esta forma, conceptos como sede, dirección, gerencia, área o programa se representan como configuraciones del mismo modelo, sin requerir una tabla específica para cada nivel de la estructura organizacional.

Tabla 4-3. Relaciones principales del modelo de organización y acceso.

Fuente: elaboración propia.

Relación	Cardinalidad	Decisión de diseño
Organization - OrgUnitType	1:N	Cada tenant define sus propios tipos de unidad
Organization - OrgUnit	1:N	Toda unidad pertenece a una única organización
OrgUnit - OrgUnit padre	0..1:N	La autorrelación construye una jerarquía de profundidad variable
OrgUnit - Location	N:M	OrgUnitLocation permite reutilizar lugares y evitar duplicación
Profile - Location	N:M	ProfileLocation separa favoritos personales de la estructura
OrgUnit - Profile	1:N	La unidad del perfil determina pertenencia y alcance organizacional

4.1.2 Auditoría y trazabilidad

Los modelos que requieren auditoría incorporan campos temporales y de autoría. Esto permite conocer cuándo se creó o modificó un registro y, cuando el contexto de autenticación lo permite, identificar quién realizó la acción. Esta información es relevante porque las decisiones sobre solicitudes, asignaciones, viajes y disponibilidad pueden requerir revisión posterior.

4.1.3 Organización como tenant

Organization representa al cliente dentro del SaaS. La pertenencia de vehículos, unidades, perfiles y solicitudes se resuelve directa o indirectamente desde esta entidad. Las consultas se acotan por organización y, si no es posible determinarla, se deniega el acceso.

Además de actuar como tenant, Organization conserva el modo de asignación de conductor y la política aplicable a los viajes planificados. El modo puede ser manual o automático; en este último caso, la asignación puede ejecutarse inmediatamente, pocos minutos antes del viaje o mediante una preasignación con revalidación. El número de minutos de anticipación se configura por organización, permitiendo adaptar el flujo sin modificar el esquema de datos.

La pertenencia de una solicitud puede resolverse siguiendo la ruta Profile → OrgUnit → Organization. No obstante, TravelRequest mantiene también una clave foránea directa a Organization, derivada del perfil solicitante al crear el registro. Esta referencia facilita la generación del folio operacional, los filtros multi-tenant y la auditoría, mientras la aplicación valida que coincida con la organización obtenida desde el perfil. Las demás relaciones que cruzan entidades comprueban que los objetos involucrados pertenezcan al mismo tenant.

4.1.4 Tipos y unidades organizacionales

OrgUnitType define las categorías que cada organización utiliza para describir su estructura interna. OrgUnit representa los nodos concretos de esa estructura y utiliza una relación recursiva parent para construir jerarquías de profundidad variable. El atributo depth_hint funciona como una guía de ordenamiento e interpretación de niveles sin imponer un organigrama universal, lo que permite adaptar el modelo a municipalidades, empresas u otras instituciones con estructuras distintas.

El modelo impone unicidad del nombre del tipo y de la unidad dentro de cada organización. También valida que el tipo y la unidad padre pertenezcan al mismo tenant. Antes de guardar se recorre la cadena de ancestros para impedir que una unidad sea su propio padre o quede debajo de uno de sus descendientes. Con ello se evitan ciclos que harían imposible recorrer o representar el organigrama.

4.1.5 Ubicaciones y relaciones

Location concentra el nombre, dirección textual, coordenadas y el identificador de un proveedor geográfico. OrgUnitLocation asocia ubicaciones con unidades organizacionales y permite que una misma unidad tenga uno o más lugares vinculados. ProfileLocation conserva las ubicaciones favoritas de los perfiles. Las tablas intermedias evitan duplicar lugares y mantienen separadas pertenencia organizacional y preferencia personal.

Location no contiene una clave foránea directa a Organization porque un mismo punto puede reutilizarse en distintos contextos. En consecuencia, una ubicación solo adquiere pertenencia operacional cuando se asocia a una unidad o perfil dentro del contexto de una organización validada. OrgUnitLocation impide repetir la misma combinación de unidad y ubicación. ProfileLocation aplica la misma regla para un perfil y limita sus favoritos a cinco, evitando listas sin control y simplificando su uso en la solicitud de viajes.

4.1.6 Perfiles, roles y conductores

Profile vincula el usuario de autenticación con su rol y unidad organizacional. DriverProfile conserva información específica del conductor. La pertenencia a la organización se obtiene mediante Profile.org_unit.organization, relación utilizada por filtros de seguridad, ETL y analítica.

4.2 Diseño OLTP: gestión de flota

La Tabla 4-4 resume las tablas que sostienen la gestión de flota y la disponibilidad operacional.

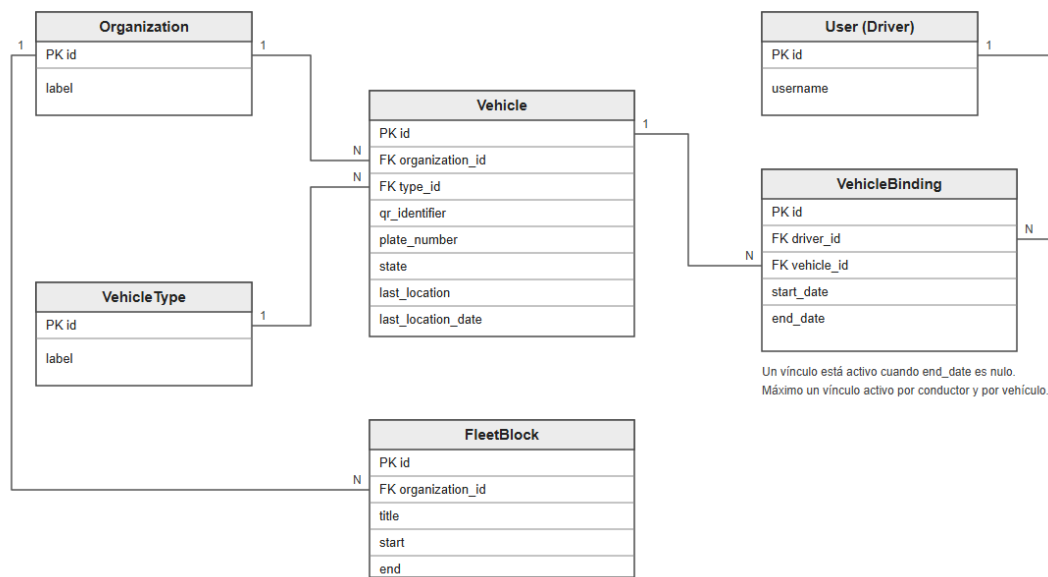


Figura 4-2. Modelo simplificado de gestión de flota. Fuente: elaboración propia.

Tabla 4-4. Principales tablas del modelo de gestión de flota.

Fuente: elaboración propia.

Tabla	Qué representa	Decisión de diseño
VehicleType	Categoría o tipo de vehículo.	Evita repetir atributos descriptivos y estandariza categorías de activos.
Vehicle	Activo físico de la flota interna.	Pertenece a una organización y su disponibilidad depende de estado, vínculo, bloqueo y viaje vigente.
VehicleBinding	Relación temporal entre conductor y vehículo.	Conserva historial y protege que conductor o vehículo tengan solo una vinculación activa.
FleetBlock	Ventana temporal de bloqueo operacional.	Permite restringir solicitudes o asignaciones durante contingencias o indisponibilidad global.

4.2.1 Vehículos y tipificación

VehicleType estandariza las categorías de vehículos. Vehicle registra patente, código interno, capacidad, estado, kilometraje, documentación y pertenencia organizacional. La ubicación reciente y los datos espaciales apoyan el seguimiento operacional.

La separación entre tipo y vehículo evita repetir atributos descriptivos y permite aplicar validaciones comunes a una categoría. Vehicle conserva los atributos que cambian por activo, mientras la organización determina quién puede consultarlo o asignarlo. La disponibilidad no se deduce únicamente de un campo de estado: también depende de vinculaciones activas, bloqueos y viajes vigentes.

4.2.2 Vinculación conductor-vehículo

VehicleBinding representa la relación temporal entre el conductor y el vehículo que opera, generada operacionalmente cuando el conductor se vincula al vehículo mediante su identificador QR. Esta entidad actúa como fuente de verdad para determinar qué conductor está usando qué vehículo en un momento dado. Las reglas impiden más de una vinculación activa por conductor o vehículo, validan que ambos pertenezcan a la misma organización y permiten auditar inicio y término de la asociación.

Una vinculación se considera activa cuando end_date es nulo. Dos restricciones únicas condicionales, una por conductor y otra por vehículo, protegen esta regla directamente en la base de datos. Los índices sobre conductor, vehículo y fecha de término facilitan las consultas más frecuentes: conocer el vehículo actual de un conductor y determinar si un activo está disponible.

4.2.3 Bloqueos de flota

FleetBlock representa ventanas de indisponibilidad global o excepcional. Su propósito es impedir solicitudes o asignaciones inviables dentro de un rango temporal, protegiendo la consistencia entre planificación y capacidad real.

4.3 Diseño OLTP: solicitudes y viajes

La Tabla 4-5 resume las entidades que conectan la demanda registrada por el funcionario con la ejecución concreta del servicio y sus eventos operacionales.

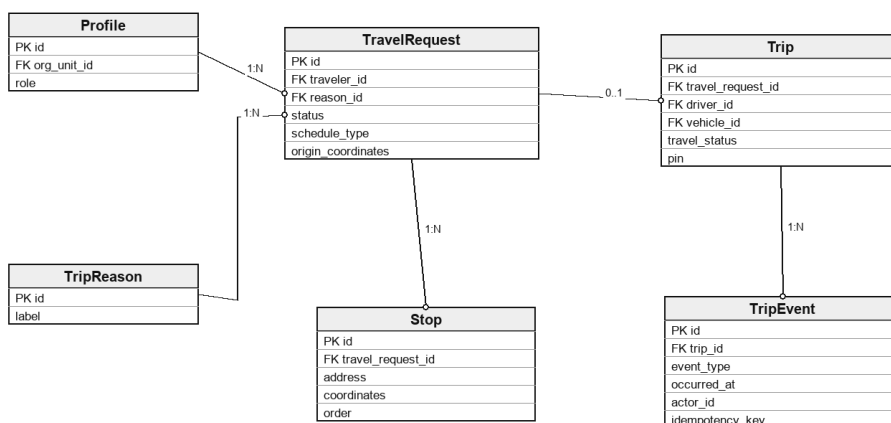


Figura 4-3. Modelo simplificado de solicitudes, viajes y eventos. Fuente: elaboración propia.

Tabla 4-5. Principales tablas del modelo de solicitudes, viajes y eventos.

Fuente: elaboración propia.

Tabla	Qué representa	Decisión de diseño
TravelRequest	Necesidad de traslado registrada por un funcionario.	Permite analizar demanda aunque no exista un viaje ejecutado.
Stop	Parada intermedia asociada a una solicitud.	El orden es parte del dato porque define el recorrido esperado.
Trip	Ejecución concreta de una solicitud aprobada y asignada.	Separa intención de servicio y ejecución efectiva; concentra conductor, vehículo y horarios reales.
TripEvent	Hito ocurrido durante el ciclo del viaje.	Permite reconstruir el ciclo operacional sin depender solo del estado final.

4.3.1 Solicitud de viaje

TravelRequest representa la necesidad de traslado y conserva solicitante, organización, folio operacional, origen, destino, motivo, pasajeros, modalidad inmediata o planificada, horario y estado. Para una solicitud planificada se registra la fecha esperada de salida o llegada, según la modalidad seleccionada. El esquema no impone un máximo fijo de anticipación, por lo que puede registrarse una fecha futura mientras se informe el campo temporal correspondiente y la política operacional de la organización lo permita. La solicitud existe aunque no llegue a materializarse en un viaje, lo que permite analizar demanda no atendida.

Este diseño separa el compromiso administrativo de la ejecución física. Una solicitud puede permanecer pendiente, aprobarse, rechazarse o cancelarse sin que necesariamente exista un Trip. Por eso, contar solicitudes no equivale a contar viajes. La distinción también permite medir demanda insatisfecha y tiempos desde la creación hasta la asignación.

4.3.2 Paradas y ubicaciones operacionales

Stop modela paradas intermedias y su orden. Los orígenes, destinos y paradas combinan descripciones legibles con coordenadas cuando están disponibles, manteniendo utilidad operacional y analítica.

La relación entre TravelRequest y Stop es de uno a muchos. El orden forma parte del dato porque dos solicitudes con los mismos puntos pueden representar recorridos diferentes. Conservar texto y coordenadas permite mostrar la dirección al usuario y, al mismo tiempo, realizar cálculos geográficos cuando la precisión disponible lo permite.

4.3.3 Viaje y validación de inicio

Trip representa la ejecución concreta, asociada a solicitud, conductor y vehículo. Conserva estados, horarios reales y un PIN de cuatro dígitos utilizado para validar el inicio entre conductor y solicitante. La separación entre solicitud y viaje distingue intención de servicio y ejecución efectiva.

La relación operacional esperada es una solicitud con cero o un viaje. Al crear el viaje se fijan conductor y vehículo, mientras los horarios reales se completan durante la ejecución. La duración y la desviación horaria no se almacenan como decisiones independientes en el OLTP: pueden derivarse de las marcas temporales y se materializan como medidas en el DWH para acelerar el análisis.

4.3.4 Eventos operacionales

Los eventos registran hitos operacionales confirmados por el modelo, como asignación, llegada al origen, inicio con PIN, finalización y cancelación. Su granularidad permite reconstruir el ciclo del servicio y calcular tiempos o fricciones sin inferirlos únicamente a partir del estado final. Situaciones como reasignaciones o incidentes pueden apoyarse en metadatos y futuras extensiones del catálogo de eventos, por lo que no se presentan como eventos independientes del modelo actual.

TripEvent mantiene una relación uno a muchos con Trip y ordena los registros por occurred_at. Los índices por viaje, tipo y fecha apoyan la reconstrucción cronológica. Cuando una integración entrega idempotency_key, una restricción evita registrar dos

veces el mismo tipo de evento para el mismo viaje, sin impedir que el historial contenga eventos repetibles legítimos.

4.4 Diseño OLAP y Data Warehouse

4.4.1 Granularidad y dimensiones

El DWH contiene dimensiones de tiempo, organización, vehículo, conductor, unidad organizacional, motivo, tipo de solicitud, estados, eventos y usuario. La dimensión de unidad organizacional recibe los registros de OrgUnit y conserva su identificador de origen, organización y etiqueta. Esto permite segmentar los indicadores según la estructura interna de cada cliente sin imponer niveles organizacionales rígidos.

Las dimensiones utilizan como clave el identificador del registro operacional cuando este existe. Este enfoque simplifica la trazabilidad y el upsert del ETL. Los atributos descriptivos se actualizan con el valor vigente; por lo tanto, el diseño actual se comporta como una dimensión de tipo 1 y no conserva versiones históricas de cada cambio de nombre. La historia principal se preserva en las fechas y medidas de los hechos y en la secuencia de eventos.

4.4.2 Tablas de hechos

El grano define qué representa exactamente una fila dentro de una tabla de hechos. Esta definición evita duplicar conteos entre demanda, ejecución y eventos. FactSolicitudes tiene una fila por solicitud y permite estudiar demanda y resultados. FactViajes tiene una fila por viaje terminado o cancelado y contiene medidas de ejecución. FactEventosViaje tiene una fila por evento operacional. FactEstadoFlotaDiario tiene una fila por vehículo y día para analizar disponibilidad y utilización.

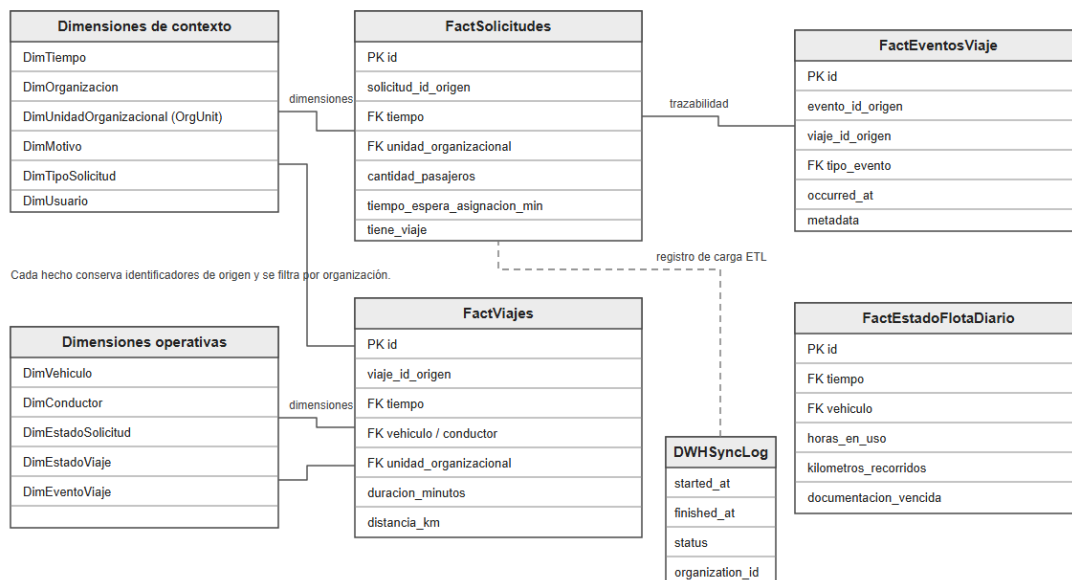


Figura 4-4. Esquema dimensional simplificado del Data Warehouse. Fuente: elaboración propia.

Tabla 4-6. Granularidad y principales atributos de las tablas de hechos.

Fuente: elaboración propia.

Tabla de hechos	Grano	Medidas o atributos analíticos principales
FactSolicitudes	Una solicitud	Pasajeros, espera de asignación, estado, tiene viaje y resultado
FactViajes	Un viaje finalizado o cancelado	Duración, distancia, pasajeros, esperas, estado y desvío
FactEventosViaje	Un evento de viaje	Fecha y hora, tipo, actor y metadatos del evento
FactEstadoFlotaDiario	Un vehículo por día	Horas en uso, kilómetros y documentación vencida

Las claves foráneas hacia tiempo, vehículo, conductor, unidad, motivo y estados permiten combinar hechos con contexto sin repetir descripciones en cada fila. Los identificadores de origen son únicos en FactSolicitudes, FactViajes y FactEventosViaje; en el estado diario, la combinación tiempo-vehículo es única. Estas restricciones hacen que una carga repetida actualice el mismo hecho en vez de duplicarlo.

4.4.3 Crecimiento y capacidad

El crecimiento depende del número de solicitudes, viajes, eventos por viaje y vehículos activos por día. Puede estimarse como: filas diarias = solicitudes + viajes cerrados + eventos + vehículos observados. FactEventosViaje y FactEstadoFlotaDiario son las estructuras con mayor crecimiento acumulativo. Como aún no existe una serie productiva representativa del pilotaje, no se presenta una tasa empírica; el volumen debe medirse durante la operación y utilizarse para ajustar índices, particionamiento y retención.

4.5 Integración y procesamiento ETL

El comando `populate_dwh` extrae datos desde la base operacional default, transforma claves y atributos, y realiza inserciones o actualizaciones en la conexión DWH. Admite carga completa, carga incremental y acotamiento por identificador de organización. `DWHSyncLog` registra estado, fechas, alcance y errores de cada ejecución.

La extracción consulta primero catálogos y entidades operacionales dentro del alcance solicitado. La transformación resuelve la dimensión de tiempo, normaliza estados, calcula duraciones y esperas, y traduce referencias OLTP a claves dimensionales. La carga utiliza actualizaciones o inserciones según el identificador de origen. Las dimensiones se sincronizan antes que los hechos para asegurar que las claves foráneas estén disponibles.

La tarea `analytics.run_dwh_sync_task` ejecuta el comando mediante Celery. La aplicación permite iniciar la sincronización de forma asíncrona; una frecuencia periódica puede configurarse en el entorno de despliegue según el nivel de actualización

requerido. La programación no se fija en el modelo para evitar asumir la misma ventana operacional para todos los clientes.

La fuente primaria del DWH es la base transaccional generada por FlotUp. La carga inicial utiliza sincronización completa; las siguientes ejecuciones usan marcas temporales para acotar registros insertados o modificados. La eliminación física de registros no se propaga automáticamente mediante este criterio incremental, por lo que requiere una política de conciliación o una carga completa controlada. Los datos de demostración se generan mediante comandos controlados y se mantienen separados por organización. Al cierre de esta tesina no se dispone todavía de una serie productiva municipal suficiente para validar tendencias estadísticas.

4.6 Integridad, rendimiento y evolución del esquema

La integridad se distribuye entre la base y la aplicación. Las claves foráneas, restricciones únicas e índices protegen invariantes que no deben depender de una pantalla específica. Las validaciones de modelo comprueban reglas que requieren recorrer relaciones, como la prevención de ciclos o la pertenencia al mismo tenant. Los servicios agregan las reglas asociadas al estado de la operación.

Los índices se concentran en las rutas de consulta frecuentes: identificadores públicos, vinculaciones activas, ventanas de bloqueo, fechas y tipos de eventos. En el DWH, los hechos se consultan principalmente por tiempo y dimensiones. El crecimiento real del pilotaje permitirá determinar si es necesario agregar índices compuestos o particionar hechos de alto volumen, evitando optimizaciones prematuras.

Las migraciones de Django versionan el modelo físico y permiten evolucionarlo de forma controlada. Cada cambio de esquema debe mantener compatibilidad con los datos existentes, validar el aislamiento por organización y considerar el efecto tanto en la base default como en la conexión DWH. Esta disciplina evita que la base analítica quede desalineada respecto de la fuente operacional.

4.7 Indicadores analíticos

Los indicadores definidos se organizan en demanda, ejecución, uso de flota, desempeño de conductores, unidades organizacionales y comportamiento temporal. Entre los indicadores directamente sustentados por el modelo se encuentran total de solicitudes, proporción inmediata/planificada, tasa de ejecución, viajes finalizados y cancelados, duración y tiempos de espera. Métricas como kilómetros, utilización detallada de la flota o análisis de eventos específicos dependen de la completitud de datos geográficos, eventos y registros operacionales durante el pilotaje, por lo que se presentan como capacidades analíticas condicionadas por la calidad de los datos disponibles.

Los indicadores se calculan desde hechos con granularidad explícita y dimensiones acotadas por organización. Las consultas de demanda utilizan FactSolicitudes; las de ejecución utilizan FactViajes; las de trazabilidad utilizan FactEventosViaje; y las de disponibilidad utilizan FactEstadoFlotaDiario. Esta separación evita duplicar conteos o interpretar una solicitud como un viaje ejecutado.

4.8 Seguridad, retención y continuidad

4.8.1 Seguridad de la información

La autenticación web utiliza sesión Django para administradores y funcionarios; la aplicación móvil de conductores utiliza JWT. La autorización combina rol y pertenencia organizacional. Los identificadores recibidos desde formularios, API o WebSockets se validan en el backend y los grupos de tiempo real se separan por organización cuando transportan datos de la organización.

4.8.2 Política de respaldo y retención

La estrategia de continuidad considera proteger la base operacional mediante respaldo físico completo diario y archivado continuo de WAL con pgBackRest, con copia local y externa cifrada^{[3][8]}. La política de retención propuesta contempla cinco copias diarias, cuatro semanales y seis mensuales. El DWH se incorpora a respaldos diarios y, además, puede reconstruirse desde la base operacional y el proceso ETL. Los archivos de medios se incluyen en respaldos diarios por contener evidencias y documentos asociados a la operación.

4.8.3 Objetivos de recuperación

Tabla 4-7. Objetivos de recuperación por componente.

Fuente: elaboración propia.

Componente	RPO objetivo	RTO objetivo	Criterio
Base operacional	Igual o inferior a 1 hora	4 horas	Respaldo base y WAL/PITR
Data Warehouse	24 horas	8 horas	Restauración o reconstrucción ETL
Archivos de medios	24 horas	24 horas	Copia diaria y verificación

Los valores indicados corresponden a objetivos de diseño y no a resultados medidos. El RPO de la base operacional supone archivado continuo de WAL y deberá confirmarse según la frecuencia efectiva de envío al repositorio. Para que los objetivos RPO/RTO sean verificables, la estrategia debe incluir pruebas periódicas de restauración en un entorno aislado y registrar sus resultados. La recuperación parcial por organización se mantiene como una línea de profundización, porque un respaldo físico PostgreSQL restaura el clúster y no una organización aislada de forma nativa.

5 Validación y resultados

La validación combina revisión estructural, pruebas automatizadas, comandos de calidad y comprobación funcional. El criterio principal es que cada requisito pueda rastrearse desde una regla de negocio hasta su implementación y evidencia.

5.1 Plan de validación

Tabla 5-1. Plan de validación de la arquitectura de datos.

Fuente: elaboración propia.

Ámbito	Criterio	Evidencia
OLTP	Relaciones, restricciones y estados consistentes	Migraciones y pruebas de cuentas, flota y viajes
Multi-tenancy	Sin acceso cruzado entre organizaciones	Pruebas de permisos y filtros analíticos
OLAP	Granularidad y claves dimensionales correctas	Modelos DWH y consultas de indicadores
ETL	Carga completa/incremental trazable e idempotente	populate_dwh, DWHSyncLog y pruebas
Calidad	Hechos sin claves críticas inválidas	Comando check_dwh_quality
Indicadores	Métricas derivadas del hecho correcto	Pruebas de KPIs y consultas agregadas
Continuidad	Estrategia con objetivos RPO/RTO y protocolo de verificación	Runbook previsto, WAL/PITR y diseño de pruebas de restauración

5.2 Resultados del modelo transaccional

El modelo representa organizaciones de estructura variable, perfiles, ubicaciones reutilizables y los procesos principales de flota y viajes. La relación recursiva de OrgUnit permite organigramas de distinta profundidad. La separación TravelRequest/Trip conserva demanda no ejecutada y la vinculación temporal conductor-vehículo representa disponibilidad operacional real.

Las reglas de seguridad resuelven el tenant desde el perfil y su unidad organizacional. Las consultas sin organización válida fallan de forma cerrada. Esto reduce el riesgo de exposición cruzada y mantiene el modelo compatible con múltiples clientes.

5.3 Resultados del modelo analítico y ETL

Las cuatro tablas de hechos permiten responder preguntas diferentes sin mezclar granularidades. El ETL sincroniza los registros de OrgUnit hacia DimUnidadOrganizacional, preservando identificadores, organización y etiquetas para filtrar indicadores por estructura interna. Las dimensiones de estados, motivos, tipos y usuarios agregan contexto estable.

La existencia de DWHSyncLog y del comando check_dwh_quality permite detectar ejecuciones fallidas y revisar integridad analítica. Las cargas por organización y las marcas de actualización acotan el procesamiento y apoyan el aislamiento multi-tenant.

5.4 Resultados de indicadores analíticos

La capa analítica permite construir y validar consultas para métricas de demanda, operación, flota, conductores, territorio y unidades organizacionales. Las pruebas verifican que cada indicador utilice la tabla de hechos correspondiente, respete el periodo solicitado y permanezca acotado a la organización del usuario. Los resultados obtenidos

corresponden a validación funcional del modelo, no a conclusiones estadísticas definitivas del pilotaje.

La validación actual utiliza datos sintéticos y de demostración con más de una organización para probar comportamiento e independencia. Estos datos son adecuados para validación funcional, pero no reemplazan una evaluación estadística con datos reales del pilotaje. Las Figuras 5-1 y 5-2 presentan ejemplos de esta comprobación: la primera muestra indicadores operacionales obtenidos desde la capa analítica y la segunda segmenta la demanda según unidades organizacionales.

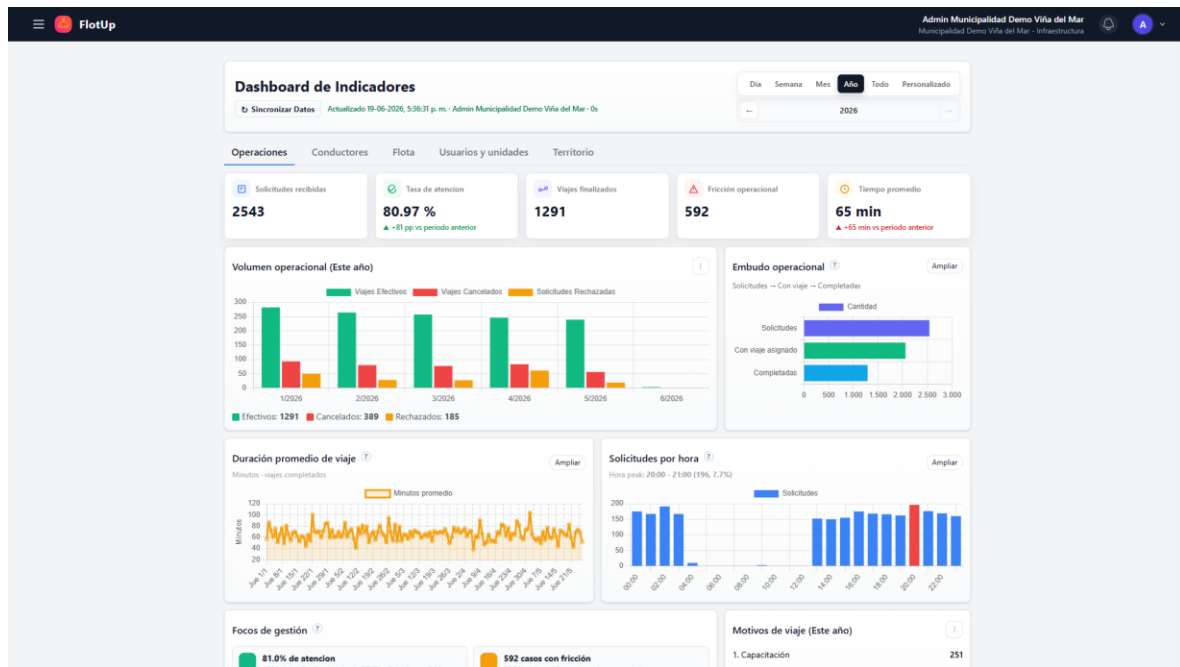


Figura 5-1. Dashboard de indicadores operacionales con datos de demostración.
Fuente: elaboración propia a partir de FlotUp.

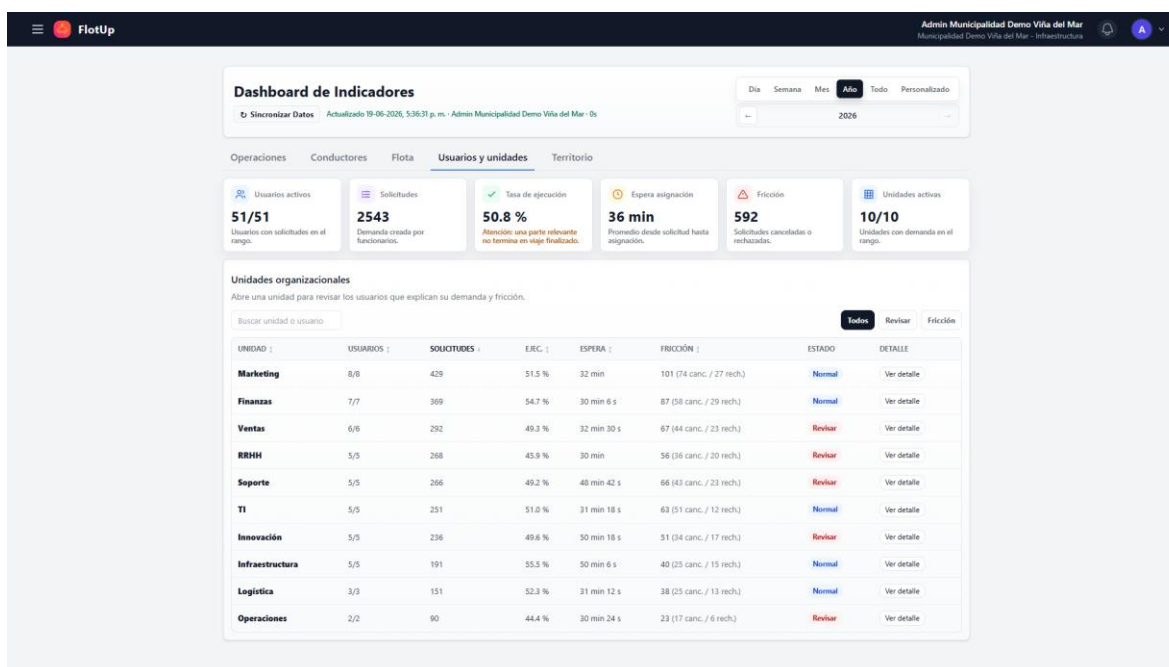


Figura 5-2. Análisis de demanda por unidades organizacionales con datos de demostración. Fuente: elaboración propia a partir de FlotUp.

5.5 Definición y estado de validación de la continuidad

La estrategia de respaldo queda definida a nivel de arquitectura para la base operacional, el DWH y los archivos de medios. La selección de pgBackRest y el archivado WAL proporcionará un mecanismo técnico para recuperación a un punto en el tiempo cuando su configuración sea desplegada y verificada mediante restauraciones. El runbook previsto deberá establecer responsables, retención, objetivos y comprobaciones posteriores a una recuperación.

La recuperabilidad debe evidenciarse mediante restauraciones integrales y verificaciones aisladas registradas periódicamente. Mientras esas pruebas no se mantengan como evidencia repetible, los RPO/RTO deben entenderse como objetivos de diseño y operación. La restauración selectiva de un tenant permanece como trabajo de profundización debido a dependencias relacionales y al alcance de los respaldos físicos.

5.6 Limitaciones de la validación

La principal limitación es la ausencia de una serie productiva extensa del cliente inicial. Por esta razón no se presentan tasas reales de crecimiento, desempeño longitudinal ni conclusiones sobre comportamiento de demanda. La tesis valida la arquitectura y su capacidad de producir estas mediciones; los resultados operacionales deberán completarse durante el pilotaje.

Asimismo, la frecuencia final del ETL en producción y los umbrales de alerta deben ajustarse con datos de carga reales. La política de respaldo define objetivos verificables, pero su efectividad debe sostenerse mediante pruebas periódicas y el registro de sus resultados.

6 Conclusiones y trabajo futuro

6.1 Conclusiones principales

Los modelos de datos y el proceso ETL implementados permiten que FlotUp registre y relacione la información necesaria para operar como SaaS de gestión de flota. El modelo OLTP representa el dominio sin imponer una estructura organizacional única y mantiene la organización como límite de seguridad. La protección y recuperabilidad de estos datos dependerá del despliegue de la estrategia de respaldo y de su verificación periódica.

La separación OLTP/OLAP resultó adecuada para distinguir operación y análisis. Las tablas de hechos separan demanda, ejecución, eventos y estado de flota, mientras el ETL conserva trazabilidad mediante registros de sincronización. La validación de indicadores muestra que el modelo puede responder preguntas de gestión sin sobrecargar la base operacional, aunque las conclusiones operacionales deberán consolidarse con datos reales del pilotaje.

La continuidad se integra como parte de la arquitectura. La política de respaldo, el uso previsto de archivado WAL, la retención diferenciada y los objetivos RPO/RTO entregan un marco concreto para proteger información operacional, analítica y documental, siempre que se mantengan pruebas periódicas de restauración.

6.2 Aprendizajes

El primer aprendizaje es que el modelo conceptual debe conservar el significado del negocio y expresarlo de forma consistente en las capas operacional y analítica. La unidad organizacional es el concepto estable; su estructura jerárquica y su proyección al DWH deben documentarse con claridad.

El segundo aprendizaje es que el aislamiento multi-tenant no puede depender de la interfaz. Debe estar presente en relaciones, consultas, servicios, WebSockets, ETL e indicadores. El tercero es que un respaldo solo es útil cuando existe un procedimiento de restauración probado y objetivos de recuperación explícitos.

6.3 Trabajo futuro

Durante el pilotaje se deberá recopilar una línea base de volumen, crecimiento, completitud, latencia del ETL y comportamiento de KPIs. Esta evidencia permitirá ajustar índices, frecuencia de sincronización, capacidad y retención con datos reales.

También se propone consolidar la recuperación selectiva por organización, automatizar el registro de pruebas de restauración y fortalecer las alertas de respaldo. En analítica, será conveniente incorporar historial de cambios en las dimensiones que lo requieran y mantener migraciones controladas entre la base operacional y el DWH.

Finalmente, la arquitectura puede incorporar telemetría, historial de posiciones y análisis espacial más avanzado. Estas extensiones deben preservar las mismas reglas de trazabilidad, minimización de datos, aislamiento y calidad definidas en este trabajo.

7 Referencias

- [1] PostgreSQL Global Development Group. PostgreSQL 16 Documentation: data definition, indexes and transaction processing. <https://www.postgresql.org/docs/16/>
- [2] Kimball, R. y Ross, M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. 3.^a ed. Wiley, 2013.
- [3] PostgreSQL Global Development Group. PostgreSQL 16 Documentation: Continuous Archiving and Point-in-Time Recovery (PITR). <https://www.postgresql.org/docs/16/continuous-archiving.html>
- [4] PostGIS Project. PostGIS Manual. <https://postgis.net/documentation/>
- [5] Django Software Foundation. Django 5.2 documentation: models, migrations and GeoDjango. <https://docs.djangoproject.com/en/5.2/>
- [6] Celery Project. Celery Documentation: Tasks and periodic tasks. <https://docs.celeryq.dev/en/stable/>
- [7] Redis Ltd. Redis Documentation. <https://redis.io/docs/latest/>
- [8] pgBackRest Project. pgBackRest User Guide. <https://pgbackrest.org/user-guide.html>
- [9] OWASP Foundation. OWASP Application Security Verification Standard. <https://owasp.org/www-project-application-security-verification-standard/>
- [10] International Organization for Standardization. ISO/IEC 25012:2008, Software engineering - Software product Quality Requirements and Evaluation - Data quality model. <https://www.iso.org/standard/35736.html>

Anexo A. Diccionario resumido y metadatos del modelo OLTP

El presente anexo organiza jerárquicamente los metadatos del modelo OLTP. Cada aplicación agrupa sus entidades y cada entidad contiene sus atributos con el tipo físico y tamaño o precisión definido en Django/PostgreSQL. Los campos heredados de TimeStampModel se documentan una sola vez; las claves foráneas utilizan el sufijo _id y corresponden a BIGINT.

Tabla A-1. Diccionario resumido de datos del modelo OLTP. Fuente: elaboración propia.

Entidad	Atributos (tipo / tamaño)
Aplicación: base común (accounts, fleet y trips)	
TimeStampModel	id: BIGINT created_at: TIMESTAMPTZ updated_at: TIMESTAMPTZ created_by_id: BIGINT updated_by_id: BIGINT is_active: BOOLEAN
Aplicación: accounts	
Organization	id: BIGINT label: VARCHAR(250) logo: VARCHAR(100) country_code: VARCHAR(2) org_code: VARCHAR(3) public_prefix: VARCHAR(5) assignment_driver_mode: INTEGER scheduled_assignment_policy: INTEGER scheduled_assignment_lead_minutes: INTEGER shared_trip_acceptance_window_minutes: INTEGER
OrgUnitType	id: BIGINT public_id: UUID organization_id: BIGINT label: VARCHAR(250) depth_hint: INTEGER
OrgUnit	id: BIGINT public_id: UUID organization_id: BIGINT parent_id: BIGINT type_id: BIGINT label: VARCHAR(250)

Entidad	Atributos (tipo / tamaño)
Location	id: BIGINT public_id: UUID label: VARCHAR(250) address_text: VARCHAR(500) lat: NUMERIC(9,6) lng: NUMERIC(9,6) place_id: VARCHAR(255)
OrgUnitLocation	id: BIGINT public_id: UUID org_unit_id: BIGINT location_id: BIGINT label: VARCHAR(250) is_active: BOOLEAN
ProfileLocation	id: BIGINT profile_id: BIGINT location_id: BIGINT label: VARCHAR(250) created_at: TIMESTAMPTZ
Profile	id: BIGINT public_id: UUID user_id: BIGINT photo: VARCHAR(100) rut: VARCHAR(8) phone: VARCHAR(9) role: INTEGER org_unit_id: BIGINT
DriverProfile	id: BIGINT profile_id: BIGINT expiration_license_date: DATE
Aplicación: django.contrib.auth	
User (Django)	id: BIGINT username: VARCHAR(150) first_name: VARCHAR(150) last_name: VARCHAR(150) email: VARCHAR(254) password: VARCHAR(128) is_active: BOOLEAN
Aplicación: fleet	
VehicleType	id: BIGINT public_id: UUID label: VARCHAR(50)

Entidad	Atributos (tipo / tamaño)
Vehicle	id: BIGINT public_id: UUID qr_identifier: UUID plate_number: VARCHAR(6) code: VARCHAR(30) organization_id: BIGINT type_id: BIGINT brand: VARCHAR(50) model: VARCHAR(50) year: INTEGER passenger_capacity: INTEGER number_of_passengers: INTEGER state: INTEGER last_kilometer: INTEGER last_location: GEOMETRY(POINT,4326) last_location_date: TIMESTAMPTZ soap_date_expiration: DATE technical_review_date_expiration: DATE insurance_date_expiration: DATE
VehicleBinding	id: BIGINT public_id: UUID driver_id: BIGINT vehicle_id: BIGINT start_date: TIMESTAMPTZ end_date: TIMESTAMPTZ
FleetBlock	id: BIGINT public_id: UUID organization_id: BIGINT title: VARCHAR(120) description: TEXT start: TIMESTAMPTZ end: TIMESTAMPTZ is_all_day: BOOLEAN
Aplicación: trips	
TripReason	id: BIGINT label: VARCHAR(150)

Entidad	Atributos (tipo / tamaño)
TravelRequest	id: BIGINT public_id: UUID folio: VARCHAR(20) folio_normalized: VARCHAR(20) folio_date: DATE daily_sequence: INTEGER organization_id: BIGINT traveler_id: BIGINT status: INTEGER origin_coordinates: GEOMETRY(POINT,4326) origin_address: VARCHAR(150) reason_id: BIGINT personalized_reason: TEXT num_passengers: INTEGER schedule_type: INTEGER expected_departure_at: TIMESTAMPTZ expected_arrival_at: TIMESTAMPTZ
Stop	id: BIGINT travel_request_id: BIGINT address: VARCHAR(150) coordinates: GEOMETRY(POINT,4326) order: INTEGER
Trip	id: BIGINT public_id: UUID travel_request_id: BIGINT driver_id: BIGINT vehicle_id: BIGINT travel_status: INTEGER date_time_departure_real: TIMESTAMPTZ date_time_arrival_real: TIMESTAMPTZ pin: VARCHAR(4)
TripEvent	id: BIGINT trip_id: BIGINT event_type: VARCHAR(40) occurred_at: TIMESTAMPTZ actor_id: BIGINT idempotency_key: VARCHAR(120) metadata: JSONB

Anexo B. Diccionario resumido y metadatos del Data Warehouse

El presente anexo organiza jerárquicamente los metadatos del modelo analítico. La aplicación analytics agrupa dimensiones, hechos y auditoría; cada objeto contiene sus atributos y tipos. DimUnidadOrganizacional representa dimensionalmente a OrgUnit y mantiene una estructura independiente de niveles organizacionales fijos.

Tabla B-1. Diccionario resumido de datos del Data Warehouse. Fuente: elaboración propia.

Entidad	Atributos (tipo / tamaño)
Aplicación: analytics	
DimTiempo	tiempo_id: INTEGER fecha: DATE anio: INTEGER mes: INTEGER nombre_mes: VARCHAR(20) dia: INTEGER dia_semana: INTEGER nombre_dia_semana: VARCHAR(20) trimestre: INTEGER
DimOrganizacion	organizacion_id: INTEGER label: VARCHAR(250)
DimUnidadOrganizacional	unidad_organizacional_id: INTEGER label: VARCHAR(250) organizacion: VARCHAR(250) organizacion_id: INTEGER
DimVehiculo	vehiculo_id: INTEGER patente: VARCHAR(10) marca: VARCHAR(50) modelo: VARCHAR(50) anio: INTEGER tipo_vehiculo: VARCHAR(50) organizacion_id: INTEGER
DimConductor	conductor_id: INTEGER nombre_completo: VARCHAR(255) rut: VARCHAR(12)
DimUsuario	usuario_id: INTEGER username: VARCHAR(150) nombre_completo: VARCHAR(255) unidad_organizacional_nombre: VARCHAR(250) organizacion: VARCHAR(250)
DimMotivo	motivo_id: INTEGER descripcion: VARCHAR(255)
DimTipoSolicitud	tipo_id: INTEGER nombre: VARCHAR(150)

Entidad	Atributos (tipo / tamaño)
DimEstadoSolicitud	estado_id: INTEGER label: VARCHAR(100)
DimEstadoViaje	estado_id: INTEGER label: VARCHAR(100)
DimEventoViaje	evento_id: VARCHAR(40) nombre: VARCHAR(100) orden: INTEGER
FactSolicitudes	id: INTEGER solicitud_id_origen: INTEGER tiempo_id: INTEGER unidad_organizacional_id: INTEGER motivo_id: INTEGER tipo_id: INTEGER usuario_id: INTEGER estado: INTEGER cantidad_pasajeros: INTEGER tiempo_espera_asignacion_min: INTEGER created_at_origen: TIMESTAMPTZ tiene_viaje: BOOLEAN fue_aprobada: BOOLEAN fue_rechazada: BOOLEAN fue_cancelada: BOOLEAN
FactViajes	id: INTEGER viaje_id_origen: INTEGER tiempo_id: INTEGER vehiculo_id: INTEGER conductor_id: INTEGER unidad_organizacional_id: INTEGER motivo_id: INTEGER tipo_id: INTEGER usuario_id: INTEGER estado: INTEGER duracion_minutos: INTEGER distancia_km: DOUBLE PRECISION cantidad_pasajeros: INTEGER tiempo_espera_solicitud_min: INTEGER tiempo_asignacion_min: INTEGER numero_incidentes: INTEGER desvio_horario_min: DOUBLE PRECISION

Entidad	Atributos (tipo / tamaño)
FactEventosViaje	id: INTEGER evento_id_origen: INTEGER viaje_id_origen: INTEGER solicitud_id_origen: INTEGER tiempo_id: INTEGER tipo_evento_id: VARCHAR(40) vehiculo_id: INTEGER conductor_id: INTEGER unidad_organizacional_id: INTEGER occurred_at: TIMESTAMPTZ actor_id_origen: INTEGER metadata: JSONB
FactEstadoFlotaDiario	id: INTEGER tiempo_id: INTEGER vehiculo_id: INTEGER horas_en_uso: DOUBLE PRECISION kilometros_recorridos: DOUBLE PRECISION documentacion_vencida: BOOLEAN
DWHSyncLog	id: BIGINT started_at: TIMESTAMPTZ finished_at: TIMESTAMPTZ status: VARCHAR(10) triggered_by_id: BIGINT organization_id: BIGINT error_message: TEXT

Anexo C. Matriz de trazabilidad

Necesidad	OLTP	OLAP / consumo	Validación
Aislar clientes	Organization y filtros	DimOrganizacion y consultas acotadas	Pruebas multi-tenant
Representar organigramas	OrgUnitType y OrgUnit	DimUnidadOrganizacional con semántica de OrgUnit	Pruebas de jerarquía y ETL
Medir demanda	TravelRequest	FactSolicitudes	KPIs de solicitudes
Medir ejecución	Trip	FactViajes	Cumplimiento y duración
Reconstruir el ciclo	TripEvent	FactEventosViaje	Eventos e hitos
Observar flota	Vehicle y estados	FactEstadoFlotaDiario	Utilización diaria
Obtener indicadores	Datos operacionales	Hechos y dimensiones	Pruebas de KPIs
Recuperar información	PostgreSQL y medios	DWH reconstruible	PITR y pruebas aisladas de restauración