

Caso de uso especificacione

UC-01 Registrar ingreso de vehículo (genera OT)

Actores: Recepcionista, Guardia de Acceso (colabora)

Precondiciones: Usuario autenticado; vehículo llega al taller.

Flujo principal:

Recepcionista ingresa patente y datos del chofer.

Sistema valida que no exista ingreso activo para esa patente (RF-04).

Sistema crea OT con estado “Ingresado” y hora de ingreso (RF-02).

Opcional: adjunta fotos/checklist del ingreso (RF-03).

Sistema registra auditoría (RF-23).

Flujos alternos:

2a. Si hay ingreso activo → muestra error y sugiere búsqueda (RF-04).

Postcondiciones: OT creada, datos guardados y trazados.

Referencias: RF-01..RF-04, RF-23; RNF-05/07/12.

UC-02 Asignar OT a mecánico

Actor: Jefe de Taller

Pre: OT en estado “Ingresado” o “Diagnóstico”.

Flujo: selecciona OT → asigna mecánico → sistema registra evento y deja traza (RF-05, RF-07, RF-23).

Post: OT con responsable asignado; histórico actualizado.

Refs: RF-05/07/23; RNF-12.

UC-03 Actualizar estado de OT

Actor: Mecánico

Pre: OT asignada al mecánico.

Flujo: cambia estado (Diagnóstico/En reparación/En pausa/Finalizada) → valida transición → registra en historial (RF-06/RF-07).

Alternos: transición inválida → mensaje y no persiste.

Post: historial consistente; si “Finalizada”, notifica (ver UC-09).

Refs: RF-06/07/20/23; RNF-05/12.

UC-04 Registrar pausa de OT

Actor: Mecánico

Pre: OT en ejecución.

Flujo: selecciona motivo (catálogo) y comienza pausa → al reanudar, sistema calcula tiempo y deja traza (RF-06/07).

Refs: RF-06/07; RNF-12.

UC-05 Gestionar repuestos (consultar stock y entregar)

Actores: Asistente de Repuestos, API Inventario

Pre: OT activa.

Flujo: busca pieza → consulta API Inventario (RF-11) → si hay stock, registra entrega y actualiza inventario (RF-12) → traza (RF-23).

Alternos: caída API → registra incidencia (RF-13) y permite solicitud diferida.

Post: movimiento de repuesto asociado a OT.

Refs: RF-10..RF-14, RF-23; RNF-26..29.

UC-06 Cerrar OT

Actor: Jefe de Taller

Pre: OT “Finalizada” por mecánico.

Flujo: valida checklist y documentos → cierra OT (RF-08) → registra en historial (RF-07) → dispara notificación (UC-09).

Refs: RF-07/08/20/23.

UC-07 Registrar entrada/salida de vehículo

Actor: Guardia de Acceso

Pre: Vehículo en portería.

Flujo: registra entrada o salida con timestamp y evidencia (fotos) (RF-03) → traza (RF-23).

Refs: RF-03/23.

UC-08 Generar reportes y exportar

Actores: Jefe de Taller, Supervisor Zonal

Pre: Datos existentes.

Flujo: filtra por fechas/estado/mecánico → el sistema genera KPI y permite exportar PDF/XLS (RF-15..RF-19).

Rendimiento: p95 ≤ 5 s (RNF-07).

Refs: RF-15..RF-19; RNF-07/18.

UC-09 Enviar notificaciones por eventos

Actores: Sistema (timer/evento), API Notificaciones

Disparadores: creación/ cierre de OT, pausas prolongadas, vehículo listo.

Flujo: genera payload → envía vía API Notificaciones → registra log/estado de entrega (RF-20, RF-23).

Alternos: reintentos ante error (RNF-28); latencia p95 ≤ 3–5 s (RNF-29).

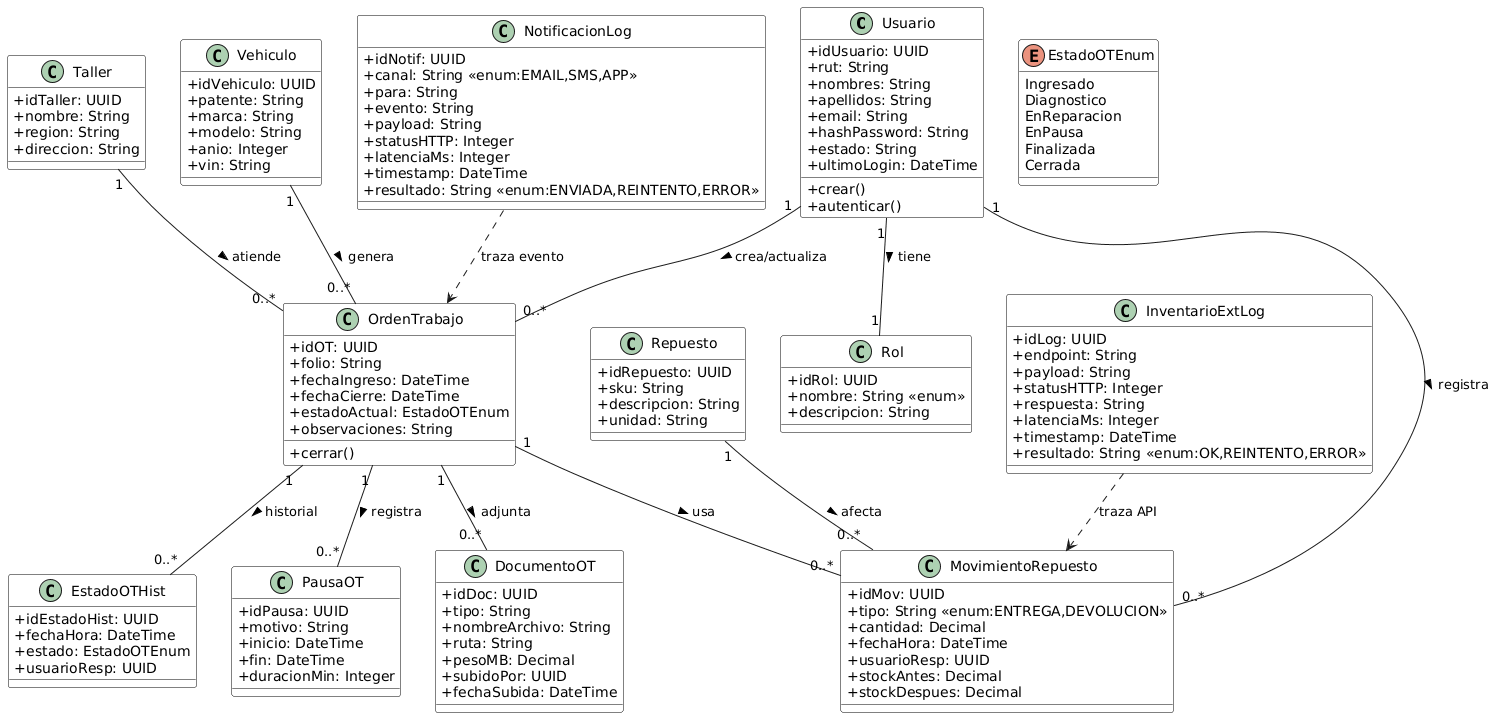
Refs: RF-20/23; RNF-26..29.

UC-10 Autenticación y gestión de usuarios/roles

Actores: Usuario, Administrador

Flujo: login con credenciales → asignación de permisos por rol → auditoría (RF-21..RF-25).

Refs: RF-21..RF-25; RNF-09..12/21..22.



### Diccionario de diagrama de clases

El sistema está diseñado bajo una estructura de base de datos relacional en PostgreSQL, organizada en diversas tablas que representan los componentes principales del proceso de gestión de ingreso de vehículos al taller de PepsiCo. A continuación, se describen las entidades, sus atributos y las relaciones más relevantes.

#### **Tabla: usuario**

Esta tabla almacena la información de los usuarios que interactúan con el sistema.  
 Cada registro se identifica mediante el atributo id\_usuario (UUID) como clave primaria.  
 Contiene campos como rut (varchar(12)), que es único y almacena el RUT chileno del usuario, nombres y apellidos (varchar(80)), además del email (varchar(120)) que también es único y se utiliza para el acceso al sistema.  
 El campo hash\_password (varchar(255)) guarda la contraseña cifrada mediante algoritmos seguros (bcrypt o argon2).  
 El atributo estado (varchar(15)) indica si el usuario se encuentra “ACTIVO” o “BLOQUEADO”.  
 También se incluye ultimo\_login (timestamp) para registrar la fecha y hora del último acceso, y id\_rol (UUID) como clave foránea que vincula al usuario con su respectivo rol en la tabla rol.

#### **Tabla: rol**

La tabla rol define los perfiles y permisos que puede tener un usuario dentro del sistema.  
 El campo id\_rol (UUID) actúa como clave primaria.  
 El atributo nombre (varchar(30)) identifica el rol, pudiendo ser ADMIN, JEFE\_TALLER, MECANICO, RECEPCIONISTA, SUPERVISOR, GUARDIA o REPUESTOS.  
 Finalmente, descripcion (varchar(120)) permite registrar una breve explicación del propósito o alcance del rol.

#### **Tabla: taller**

Esta tabla representa los diferentes talleres registrados en la plataforma.  
 Cada taller tiene un identificador único id\_taller (UUID) como clave primaria.  
 Se definen atributos como nombre (varchar(60)) que es único, region (varchar(30)) que indica la localización geográfica, y direccion (varchar(120)), campo opcional que almacena la dirección física del taller.

#### **Tabla: vehiculo**

Contiene los datos de los vehículos que ingresan a los talleres.  
 Su identificador principal es id\_vehiculo (UUID).  
 Entre sus atributos destacan patente (varchar(10)), que es único y valida el formato de placa chilena; marca y modelo (varchar(40)), que describen las características del vehículo; anio (smallint) para el año de fabricación; y vin (varchar(20)), que es único y almacena el número de serie del fabricante (opcional).

#### **Tabla: orden\_trabajo**

Registra todas las órdenes de trabajo generadas en el sistema.  
 El atributo id\_ot (UUID) es la clave primaria.  
 Incluye folio (varchar(20)), que es único y sirve como identificador legible por el usuario.  
 Cada orden se asocia a un vehículo y un taller mediante los campos id\_vehiculo y id\_taller (FK).  
 También contiene fecha\_ingreso (timestamp) para el registro de entrada, fecha\_cierre (timestamp) para el cierre del proceso, y estado\_actual (varchar(15)), que refleja el estado actual de la reparación según el catálogo de estados definidos.  
 El campo observaciones (varchar(500)) permite agregar comentarios relevantes, y id\_usuario\_crea (UUID) indica qué usuario registró la orden.

#### **Tabla: stado\_ot\_hist**

Esta tabla almacena el historial de cambios de estado de cada orden de trabajo, permitiendo la trazabilidad completa del proceso.  
 Su clave primaria es id\_estado\_hist (UUID).  
 Los atributos id\_ot (UUID) y usuario\_resp (UUID) funcionan como claves foráneas hacia las tablas orden\_trabajo y usuario.  
 El campo fecha\_hora (timestamp) registra el momento exacto del cambio de estado, y el atributo estado (varchar(15)) almacena el valor del estado asignado, correspondiente al dominio EstadoOTEnum.

**Tabla: pausa\_ot**

Registra las pausas realizadas durante una reparación.  
 El identificador principal es id\_pausa (UUID).  
 Incluye id\_ot (UUID) como clave foránea que enlaza con la orden de trabajo, motivo (varchar(60)) basado en un catálogo predefinido de causas, inicio (timestamp) y fin (timestamp) que indican el rango temporal de la pausa, y duracion\_min (int) que puede ser calculado automáticamente a partir de ambos valores.

#### **Tabla: documento\_ot**

Guarda la documentación asociada a cada orden de trabajo, como fotografías, informes o listas de chequeo.  
 El campo id\_doc (UUID) es la clave primaria, y id\_ot (UUID) actúa como clave foránea hacia la tabla orden\_trabajo.  
 El atributo tipo (varchar(20)) identifica la naturaleza del documento (FOTO, CHECKLIST o INFORME).  
 Se incluyen también nombre\_archivo (varchar(120)), ruta (varchar(255)) para el almacenamiento físico o URL, y peso\_mb (numeric(6,2)) que representa el tamaño del archivo.  
 El campo subido\_por (UUID) indica el usuario que realizó la carga, y fecha\_subida (timestamp) almacena el momento exacto de la subida.

#### **Tabla: repuesto**

Almacena los datos de los repuestos registrados en el sistema.  
 El atributo id\_repuesto (UUID) es la clave primaria.  
 Incluye sku (varchar(30)), único para cada repuesto, descripcion (varchar(120)) para su detalle y unidad (varchar(10)) para indicar la unidad de medida utilizada (por ejemplo, unidad, litro o set).

#### **Tabla: movimiento\_repuesto**

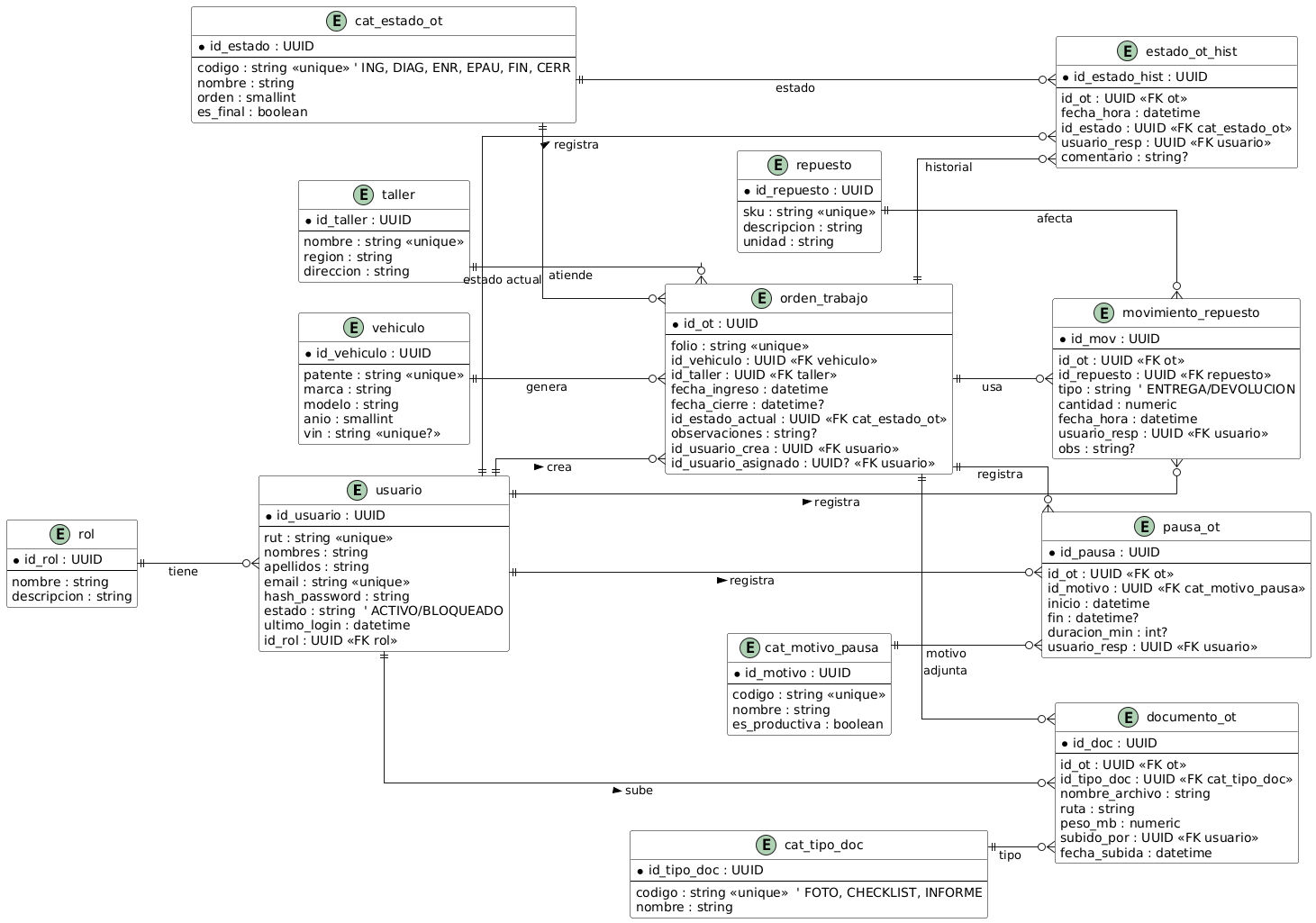
Registra los movimientos de entrada o salida de repuestos en relación con las órdenes de trabajo.  
 El campo id\_mov (UUID) es la clave primaria.  
 Se incluyen las claves foráneas id\_ot (UUID) hacia la orden de trabajo y id\_repuesto (UUID) hacia la tabla de repuestos.  
 El atributo tipo (varchar(12)) especifica si el movimiento corresponde a una ENTREGA o DEVOLUCIÓN, y cantidad (numeric(10,2)) representa la cantidad involucrada.  
 Los campos fecha\_hora (timestamp), usuario\_resp (UUID), stock\_antes y stock\_despues permiten registrar la trazabilidad del movimiento y el control del inventario.

#### **Tabla: inventario\_ext\_log**

Esta tabla lleva el registro de las interacciones con la API de Inventario externa.  
 El campo id\_log (UUID) actúa como clave primaria.  
 Incluye endpoint (varchar(120)) y payload (text) que almacenan el destino y los datos enviados, además de status\_http (int), respuesta (text) y latencia\_ms (int) que permiten analizar el rendimiento de la integración.  
 El atributo timestamp (timestamp) marca el momento de la llamada y resultado (varchar(10)) indica el estado final (OK, REINTENTO o ERROR).

**Tabla: notificacion\_log**

Registra las notificaciones emitidas por el sistema a través de la API de Notificaciones.  
 El campo id\_notif (UUID) es la clave primaria.  
 El atributo canal (varchar(10)) identifica el medio utilizado (EMAIL, SMS o APP).  
 El campo para (varchar(120)) indica el destinatario, mientras que evento (varchar(40)) describe el tipo de evento que originó la notificación (por ejemplo, cierre de OT o pausa prolongada).  
 Se almacenan también payload (text), status\_http (int), latencia\_ms (int), timestamp (timestamp) y resultado (varchar(12)), que reflejan la trazabilidad del envío y su resultado final (ENVIADA, REINTENTO o ERROR).



El modelo de base de datos propuesto para la *Plataforma de Gestión Digital de Ingreso de Vehículos al Taller* de PepsiCo Chile se estructura bajo una arquitectura relacional normalizada en Tercera Forma Normal (3FN), garantizando la eliminación de redundancias y la integridad de los datos. A continuación, se describen las entidades principales y sus relaciones de forma narrativa.

El usuario representa a toda persona que interactúa con el sistema (recepcionista, mecánico, jefe de taller, asistente de repuestos, supervisor, coordinador o administrador). Cada registro contiene su identificador, rut, nombres, apellidos, correo electrónico, contraseña cifrada, estado y último inicio de sesión. El usuario se vincula con la entidad rol, que define el perfil y los permisos asociados a cada tipo de actor dentro de la plataforma, permitiendo controlar las operaciones que puede realizar.

La entidad taller almacena la información de cada centro de mantenimiento donde se gestionan las órdenes de trabajo. Incluye datos como nombre, región y dirección física. Por su parte, la entidad vehículo conserva la información básica de los camiones o automóviles de la flota: patente, marca, modelo, año y número VIN. Un vehículo puede generar múltiples órdenes de trabajo a lo largo del tiempo, lo que se representa mediante una relación uno a muchos entre *vehículo* y *orden\_trabajo*.

La orden de trabajo (OT) es la entidad central del modelo y representa el proceso de ingreso, diagnóstico, reparación y cierre de un vehículo. Cada OT registra su folio único, las fechas de ingreso y cierre, el estado actual, observaciones, el usuario que la creó y el mecánico asignado. Se relaciona con las entidades *vehículo*, *taller* y *usuario* para mantener trazabilidad completa sobre quién la creó, dónde se atiende y a qué vehículo corresponde.

El historial de cambios de estado de cada orden se gestiona mediante la entidad estado\_ot\_hist, donde se almacena cada transición junto con la fecha, el usuario responsable y el tipo de estado, referenciado desde el catálogo cat\_estado\_ot, que define las fases válidas del proceso (por ejemplo: Ingresado, Diagnóstico, En reparación, En pausa, Finalizada, Cerrada). Este diseño permite mantener una auditoría detallada de la evolución de cada OT.

La entidad pausa\_ot registra las interrupciones del proceso productivo, especificando el motivo, las marcas de tiempo de inicio y fin, la duración total y el usuario responsable. Los motivos se administran en el catálogo cat\_motivo\_pausa, donde se clasifican las causas y se indica si son o no productivas, favoreciendo el análisis de eficiencia operativa.

Los documentos digitales asociados a las órdenes, tales como fotografías, checklists o informes técnicos, se almacenan en la entidad documento\_ot. Esta contiene el tipo de documento, la ruta de almacenamiento, el tamaño del archivo, el usuario que lo subió y la fecha de registro. Los tipos de documentos válidos se definen en el catálogo cat\_tipo\_doc, asegurando consistencia en la clasificación de la evidencia visual o documental.

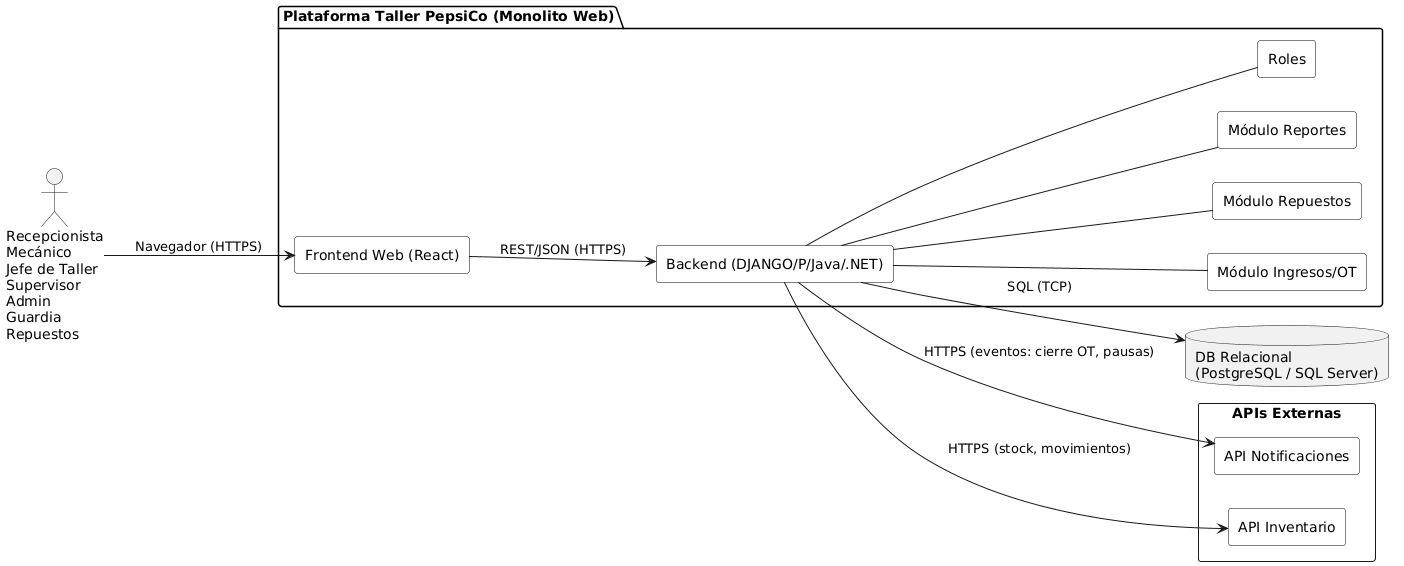
El manejo de insumos se modela a través de las entidades repuesto y movimiento\_repuesto. La primera guarda el detalle de cada ítem disponible en bodega, incluyendo su código SKU, descripción y unidad de medida. La segunda registra los movimientos asociados a una OT —ya sea entrega o devolución— indicando la cantidad, la fecha y el responsable del movimiento. Este enfoque permite mantener la trazabilidad del uso de repuestos y su impacto en cada reparación.

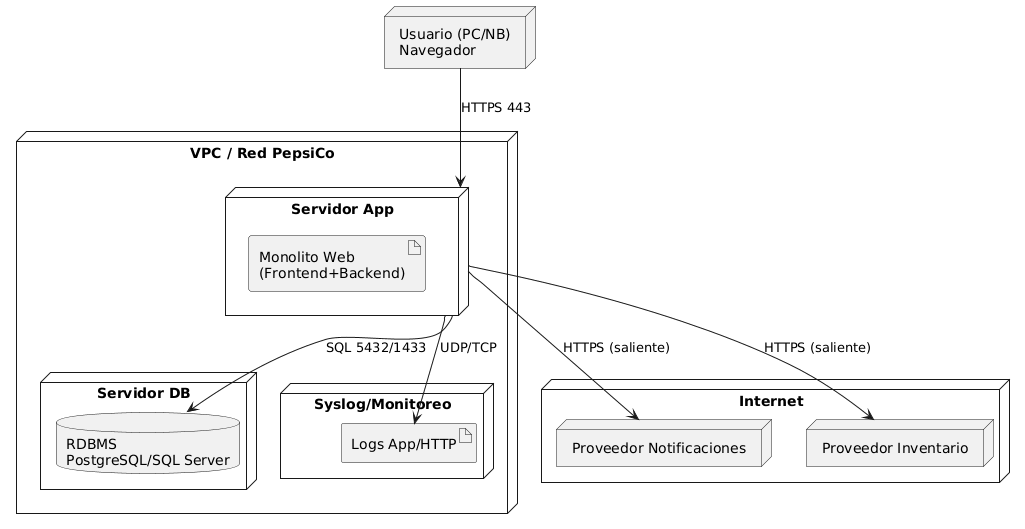
Finalmente, todos los catálogos del sistema (cat\_estado\_ot, cat\_motivo\_pausa y cat\_tipo\_doc) funcionan como tablas maestras que evitan la repetición de valores y estandarizan la información utilizada por las tablas transaccionales. El modelo en su conjunto garantiza integridad referencial mediante claves foráneas y elimina dependencias transitivas, cumpliendo con las normas de normalización establecidas para bases de datos relacionales.

DIAGRAMAS UML

La solución adopta una arquitectura web monolítica modular desplegada en un servidor de aplicaciones y una base de datos relacional. El frontend web (SPA) consume endpoints del backend vía HTTPS. El backend implementa módulos de negocio (*Ingresos/OT, Repuestos, Reportes, Autenticación/Roles*) y se integra con dos APIs externas: Notificaciones (para envío de alertas de eventos como “vehículo listo” o “OT cerrada”) e Inventario (para consultar/afectar stock de repuestos). La DB persiste el dominio (OT, vehiculo, etc.) y la evidencia transaccional (historial de estados, pausas, documentos, movimientos). El monitoreo se realiza vía logs de aplicación y métricas básicas (latencia, errores).

Flujo típico: un usuario opera desde navegador → frontend llama backend → backend persiste/consulta en DB y, si corresponde, invoca APIs (p. ej., consultar stock o enviar una notificación). Los errores de integración se manejan con reintentos y registro de auditoría. La seguridad se basa en autenticación (credenciales) y autorización por rol, todo bajo HTTPS.





# 4) “Diccionario” de Integración – APIs y tablas de soporte

## 4.1 API de Notificaciones (externa)

Propósito: enviar alertas de eventos (pausas prolongadas, OT finalizada/cerrada).  
 Endpoint ejemplo: POST /notifications/send  
 Request (JSON):

* event (string): PAUSA\_PROLONGADA | OT\_FINALIZADA | OT\_CERRADA
* target (string): email/teléfono/usuario
* payload (obj): { id\_ot, folio, patente, estado, fecha\_evento }  
   Response:
* status (string): SENT | QUEUED | ERROR
* trace\_id (string)
* latency\_ms (number)

Tabla sugerida de log (interna, opcional): notif\_log

* id\_log (uuid, PK)
* evento (varchar(30), N)
* destinatario (varchar(120), N)
* payload (json/text, N)
* status (varchar(12), N): ENVIADA/REINTENTO/ERROR
* http\_status (int, S)
* latencia\_ms (int, S)
* timestamp (timestamp, N)

## 4.2 API de Inventario (externa)

Propósitos:

* Consulta de stock por sku
* Registro de movimiento (descuento/devolución)

Endpoints ejemplo:

* GET /inventory/stock/{sku} → { sku, stock\_disponible }
* POST /inventory/movements → { sku, tipo: 'DESCUENTO'|'AJUSTE'|'DEVOLUCION', cantidad, referencia: id\_ot }

Tabla sugerida de log (interna, opcional): inventario\_log

* id\_log (uuid, PK)
* endpoint (varchar(100), N)
* request (json/text, N)
* response (json/text, S)
* http\_status (int, S)
* resultado (varchar(10), N): OK/REINTENTO/ERROR
* latencia\_ms (int, S)
* timestamp (timestamp, N)

DIAGRAMAS LOGICO

# 1) Explicación – Modelo Lógico de Datos (ER lógico)

El modelo lógico organiza la información del sistema en tablas normalizadas que garantizan integridad y trazabilidad del proceso completo de la Orden de Trabajo (OT). Las entidades son: vehiculo, taller, usuario/rol y orden\_trabajo.

Desde la OT se desprenden los registros transaccionales que dan evidencia del ciclo de vida: estado\_ot\_hist (historial de estados), pausa\_ot (pausas con motivo), documento\_ot (evidencia digital) y movimiento\_repuesto (consumo/devoluciones de ítems). Para asegurar consistencia, se emplean catálogos (cat\_estado\_ot, cat\_motivo\_pausa, cat\_tipo\_doc) que estandarizan valores y facilitan auditoría.

dominante es 1:N: un *vehículo* genera muchas OTs, un *taller* atiende muchas OTs, y una OT tiene múltiples estados, pausas, documentos y movimientos de repuesto. Las claves foráneas materializan estas dependencias y evitan huérfanos; las columnas únicas (patente, folio, sku) impiden duplicados. El modelo favorece reportabilidad: consultas por fechas, estado, mecánico, taller o vehículo se optimizan con índices propuestos.

# **diccionario de Datos – Modelo Lógico (tablas principales)**

Tipos pensados para PostgreSQL; en SQL Server cambia uuid por uniqueidentifier. “N” = NO nulo, “S” = SÍ nulo.

### 2.1 Maestros y seguridad

rol

* id\_rol (uuid, N, PK): identificador del rol.
* nombre (varchar(30), N): ADMIN, JEFE\_TALLER, etc.
* descripcion (varchar(120), S): detalle.

usuario

* id\_usuario (uuid, N, PK)
* rut (varchar(12), N, ÚNICO)
* nombres (varchar(80), N)
* apellidos (varchar(80), N)
* email (varchar(120), N, ÚNICO)
* hash\_password (varchar(255), N)
* estado (varchar(15), N): ACTIVO/BLOQUEADO
* ultimo\_login (timestamp, S)
* id\_rol (uuid, N, FK→rol)

taller

* id\_taller (uuid, N, PK)
* nombre (varchar(60), N, ÚNICO)
* region (varchar(30), N)
* direccion (varchar(120), S)

vehiculo

* id\_vehiculo (uuid, N, PK)
* patente (varchar(10), N, ÚNICO)
* marca (varchar(40), S)
* modelo (varchar(40), S)
* anio (smallint, S)
* vin (varchar(20), S, ÚNICO)

### 2.2 Catálogos

cat\_estado\_ot

* id\_estado (uuid, N, PK)
* codigo (varchar(10), N, ÚNICO): ING, DIAG, ENR, EPAU, FIN, CERR
* nombre (varchar(40), N)
* orden (smallint, N)
* es\_final (boolean, N)

cat\_motivo\_pausa

* id\_motivo (uuid, N, PK)
* codigo (varchar(20), N, ÚNICO)
* nombre (varchar(60), N)
* es\_productiva (boolean, N)

cat\_tipo\_doc

* id\_tipo\_doc (uuid, N, PK)
* codigo (varchar(20), N, ÚNICO): FOTO, CHECKLIST, INFORME
* nombre (varchar(60), N)

### 2.3 Transaccionales

orden\_trabajo

* id\_ot (uuid, N, PK)
* folio (varchar(20), N, ÚNICO)
* id\_vehiculo (uuid, N, FK→vehiculo)
* id\_taller (uuid, N, FK→taller)
* fecha\_ingreso (timestamp, N)
* fecha\_cierre (timestamp, S)
* id\_estado\_actual (uuid, N, FK→cat\_estado\_ot)
* observaciones (varchar(500), S)
* id\_usuario\_crea (uuid, N, FK→usuario)
* id\_usuario\_asignado (uuid, S, FK→usuario)

estado\_ot\_hist

* id\_estado\_hist (uuid, N, PK)
* id\_ot (uuid, N, FK→orden\_trabajo)
* fecha\_hora (timestamp, N)
* id\_estado (uuid, N, FK→cat\_estado\_ot)
* usuario\_resp (uuid, N, FK→usuario)
* comentario (varchar(300), S)

pausa\_ot

* id\_pausa (uuid, N, PK)
* id\_ot (uuid, N, FK→orden\_trabajo)
* id\_motivo (uuid, N, FK→cat\_motivo\_pausa)
* inicio (timestamp, N)
* fin (timestamp, S)
* duracion\_min (int, S)
* usuario\_resp (uuid, N, FK→usuario)

documento\_ot

* id\_doc (uuid, N, PK)
* id\_ot (uuid, N, FK→orden\_trabajo)
* id\_tipo\_doc (uuid, N, FK→cat\_tipo\_doc)
* nombre\_archivo (varchar(120), N)
* ruta (varchar(255), N)
* peso\_mb (numeric(6,2), N)
* subido\_por (uuid, N, FK→usuario)
* fecha\_subida (timestamp, N)

repuesto

* id\_repuesto (uuid, N, PK)
* sku (varchar(30), N, ÚNICO)
* descripcion (varchar(120), N)
* unidad (varchar(10), N)

movimiento\_repuesto

* id\_mov (uuid, N, PK)
* id\_ot (uuid, N, FK→orden\_trabajo)
* id\_repuesto (uuid, N, FK→repuesto)
* tipo (varchar(12), N, CHECK ∈ {ENTREGA, DEVOLUCION})
* cantidad (numeric(10,2), N, >0)
* fecha\_hora (timestamp, N)
* usuario\_resp (uuid, N, FK→usuario)
* obs (varchar(200), S)