INGENIERÍA INFORMÁTICA

Administración y diseño de base de datos

Sistema de gestión de logística y transporte internacional

Informe realizado por:

Alba Pérez Rodríguez
alu0101513768@ull.edu.es

Eric Bermúdez Hernández
alu0101517476@ull.edu.es

Índice.

1. Introducción.

El presente proyecto tiene como finalidad el diseño, desarrollo e implementación de una base de datos destinada a una empresa dedicada a la logística y el transporte internacional, cuyo volumen de actividad ha experimentado un crecimiento notable en los últimos años. La organización gestiona operaciones de exportación e importación mediante transporte marítimo, aéreo y terrestre, lo que implica la coordinación de múltiples actores, documentos, rutas, transportistas y procesos simultáneos.

En la situación actual, la empresa presenta importantes limitaciones operativas derivadas de la dispersión de la información en múltiples hojas de cálculo, herramientas aisladas y sistemas no integrados. Esto genera problemas habituales como la duplicidad y pérdida de información, dificultad para coordinar envíos, rutas y transportistas, errores de trazabilidad y seguimiento, gestión aduanera deficiente o incompleta, facturación poco eficiente e imposibilidad de obtener informes fiables y en tiempo real.

Ante este contexto, se plantea la necesidad de desarrollar una solución tecnológica unificada, basada en una base de datos robusta y normalizada, capaz de centralizar todos los datos operativos y administrativos que forman parte del ciclo logístico internacional. Este proyecto constituye la base estructural para futuros desarrollos, incluyendo APIs REST, interfaces web y sistemas TMS/WMS avanzados.

2. Objetivos y requisitos.

El proyecto tiene como objetivo principal diseñar una base de datos relacional que permita gestionar de forma integrada todas las operaciones logísticas de la empresa, asegurando que la información sea consistente, accesible y segura. La solución debe centralizar los datos de clientes, productos, envíos, contenedores, transportistas, rutas, almacenes, aduanas, incidencias, trazabilidad y procesos de facturación, permitiendo que cada registro pueda ser consultado, actualizado y rastreado en tiempo real.

Los objetivos específicos incluyen:

- Facilitar la gestión completa de los envíos y su trazabilidad, asociando cada producto a contenedores, transportistas y rutas correspondientes.

- Permitir la administración de transportistas mediante un modelo de supertipo y subtipos (aéreo, marítimo y terrestre) que reflejen las distintas modalidades de transporte.

- Registrar y controlar eventos de seguimiento, incidencias operativas y documentación aduanera, integrando esta información con la gestión de pagos y facturación.

- Optimizar la operación interna mediante el control de almacenes y movimientos de inventario, vinculando estos procesos con los envíos y rutas planificadas.

- Garantizar integridad y consistencia de los datos, evitando duplicidades y registros incompletos, y permitiendo consultas fiables para la toma de decisiones estratégicas.

En cuanto a los requisitos del sistema, se destacan los siguientes:

- La base de datos debe implementarse en PostgreSQL, con un diseño normalizado que asegure integridad referencial, uso adecuado de claves primarias y foráneas, y relaciones correctas entre todas las entidades.

- Se debe desarrollar una API RESTful con Flask que permita realizar operaciones CRUD (crear, consultar, actualizar y eliminar) sobre todas las entidades, ofreciendo respuestas en formato JSON y gestionando adecuadamente errores y excepciones.

- El proyecto debe ejecutarse en un entorno virtual de Python (venv) para aislar dependencias y facilitar el mantenimiento.

- La estructura inicial de la base de datos se debe generar mediante un script (init_db.py) que cree las tablas y cargue datos de prueba, asegurando el correcto funcionamiento de las relaciones y restricciones.

- El sistema debe ser escalable, de manera que pueda soportar futuras ampliaciones, como integración con sistemas TMS/WMS, seguimiento GPS o automatización documental.

3. Contexto de la base de datos.

La base de datos propuesta busca reflejar de manera precisa los procesos de logística internacional de la empresa, permitiendo centralizar información crítica de clientes, envíos, transportistas, rutas, almacenes y facturación. El diseño prioriza la integridad, la trazabilidad y la coherencia de los datos, asegurando que cada operación pueda ser registrada y consultada de manera confiable. Se han definido entidades fuertes, débiles y jerarquías de herencia, así como relaciones de distintos tipos (1:N, N:M, ternarias) que reflejan la complejidad de la operación logística.

3.1. Entidades.

País: Representa los países donde opera la empresa. Cada país tiene un nombre obligatorio y un código ISO de dos caracteres único. Esta entidad es clave para ubicar terminales, almacenes y planificar rutas internacionales. Se asegura la integridad mediante la restricción de unicidad en el código ISO y la obligatoriedad del nombre.

Terminal: Representa aeropuertos, puertos o estaciones terrestres. Está vinculada a un país y tiene un tipo específico (Aéreo, Marítimo, Terrestre), lo que permite clasificar correctamente las rutas y transportistas. Cada terminal es única y sirve como nodo de origen o destino de rutas, así como punto de registro para eventos de tracking.

Almacén: Espacios físicos de almacenamiento. Cada almacén está asociado a un país. Permite registrar movimientos de mercancías (entrada y salida), controlar inventarios y garantizar la trazabilidad de los productos antes y después del transporte.

Cliente: Persona o empresa que solicita envíos. Sus atributos incluyen nombre, dirección, correo electrónico y teléfono. El correo electrónico es único para evitar duplicidad. Cada cliente puede solicitar múltiples envíos y recibir varias facturas, permitiendo trazabilidad administrativa y financiera.

Producto: Representa los bienes que se transportan. Cada producto tiene nombre obligatorio, descripción opcional, peso y volumen, que no pueden ser negativos. Esta información permite calcular capacidades de contenedores, vehículos y rutas, asegurando la correcta planificación de la carga.

Contenedor: Representa contenedores físicos para transporte de mercancías. Cada contenedor tiene tipo, capacidad y tara, que no pueden ser negativas. Los contenedores se asignan a uno o varios envíos, y se asegura que la capacidad no se sobrepase.

Transportista (Supertipo): Representa a la empresa responsable del transporte y puede ser de tipo Aéreo, Marítimo o Terrestre. Se utiliza herencia (IS_A) para diferenciar los subtipos. La entidad principal permite centralizar información común como el nombre y tipo de transportista.

Transportista Aéreo, Marítimo y Terrestre (Subtipos): Especializaciones del transportista, que incluyen información propia de cada tipo. Los subtipos aseguran que los transportistas sean coherentes con su tipo de transporte. Por ejemplo, un transportista aéreo solo puede operar aeronaves. Triggers en la base de datos garantizan que no haya inconsistencias.

Vehículo, Buque y Aeronave: Representan los activos físicos de cada transportista según su subtipo. Cada activo tiene matrícula única y capacidad no negativa, lo que permite planificar rutas y asegurar que la carga no exceda la capacidad disponible.

Ruta: Define trayectos entre terminales de origen y destino, con un modo de transporte (Aéreo, Marítimo, Terrestre). Se asegura que origen y destino sean distintos y que las rutas estén alineadas con los tipos de transportista. Cada ruta puede tener múltiples tarifas asociadas.

Tarifa: Registra precios de transporte por ruta y tipo de carga, con fechas de vigencia. Permite llevar un historial de tarifas y calcular el costo de cada envío. La entidad depende de la ruta para contextualizar los precios.

Envío: Documento principal del sistema que representa la solicitud de transporte de un cliente. Contiene fechas de salida y llegada, estado (Pendiente, En tránsito, Entregado, Retrasado) y se vincula a líneas de envío, contenedores, asignaciones, incidencias, aduanas, movimientos de almacén y facturas.

Línea de Envío (Entidad Débil): Detalla los productos que contiene cada envío. No puede existir sin el envío padre. Cada línea registra cantidad de productos, asegurando que no sea negativa. Permite calcular volúmenes y pesos totales de un envío.

Asignación (Relación Ternaria): Representa la asignación de un envío a una ruta y un transportista. Esta relación ternaria asegura que un envío solo sea transportado por un transportista adecuado al modo de la ruta. Incluye fecha programada y estado, y es clave para la planificación logística.

TrackingEvent: Evento de trazabilidad de un envío en una terminal. Permite registrar movimientos, incidencias o hitos de transporte. Es una entidad débil, ya que depende de envío y terminal, y asegura la trazabilidad completa de cada operación.

Incidencia: Representa problemas ocurridos durante un envío, como retrasos o daños. Se vincula a un tipo de incidencia y a un envío específico. Su gravedad permite priorizar su gestión. Es una entidad débil porque no tiene sentido sin el envío asociado.

Tipo de Incidencia: Catálogo de posibles problemas logísticos, con un atributo de gravedad (Leve, Media, Grave). Esta entidad independiente permite clasificar las incidencias y generar estadísticas de gestión.

Aduana: Registra el paso de un envío por controles aduaneros, incluyendo fechas de inicio y fin y estado del trámite. Permite garantizar cumplimiento normativo y control de tiempos en transporte internacional.

Factura: Documento financiero asociado a un cliente y un envío. Contiene fecha de emisión y total a pagar, que no puede ser negativo. Se relaciona con pagos, asegurando trazabilidad contable.

Pago: Registra el pago parcial o total de una factura. Incluye fecha, cantidad y método de pago. La cantidad no puede ser negativa y está vinculada a la factura correspondiente.

Movimiento de Almacén: Registra entrada o salida de un envío en un almacén, con fecha y tipo (Entrada, Salida). Es una entidad débil dependiente de envío y almacén, que permite controlar inventarios y movimientos físicos de mercancías.

3.2. Relaciones.

País → Terminal / Almacén: Relación 1:N. Cada país puede tener varias terminales y almacenes, lo que facilita la planificación logística y la ubicación de infraestructuras.

Terminal → Ruta (Origen y Destino): Relación 1:N. Cada ruta tiene un terminal de origen y uno de destino. Esta distinción asegura que las rutas sean coherentes y evita duplicidades.

Ruta → Tarifa: Relación 1:N. Cada ruta puede tener múltiples tarifas por tipo de carga y vigencia, permitiendo calcular precios históricos y actuales.

Cliente → Envío: Relación 1:N. Un cliente puede generar múltiples envíos. Cada envío pertenece a un único cliente, asegurando trazabilidad.

Envío → Línea de Envío: Relación 1:N, entidad débil. Cada envío puede contener varias líneas de productos, que no existen sin el envío padre.

Producto → Línea de Envío: Relación 1:N. Un producto puede formar parte de muchas líneas de envío, facilitando control de inventario y planificación de carga.

Envío → Contenedor: Relación N:M. Un envío puede ocupar varios contenedores, y un contenedor puede contener varios envíos. Se modela en SQL mediante tabla intermedia.

Transportista → Subtipos (Aéreo, Marítimo, Terrestre): Jerarquía IS_A. Cada transportista tiene un subtipo exclusivo y se relaciona con sus activos (Vehículo, Buque, Aeronave) 1:N.

Envio – Ruta – Transportista (Asignación): Relación ternaria N:M:P. Permite asignar un envío a un transportista y una ruta específica, garantizando coherencia semántica.

Envío → TrackingEvent / Terminal → TrackingEvent: Relación 1:N. Permite registrar eventos de trazabilidad asociados a un envío y a una terminal.

Envío → Incidencia / Tipo de Incidencia → Incidencia: Relación 1:N. Cada incidencia se vincula a un envío y a su tipo, permitiendo clasificar y priorizar problemas.

Envío → Aduana: Relación 1:N. Permite registrar todos los pasos aduaneros de un envío.

Envío → Movimiento de Almacén / Almacén → Movimiento de Almacén: Relación 1:N. Permite controlar entradas y salidas de mercancía, garantizando trazabilidad física.

Cliente → Factura / Envío → Factura / Factura → Pago: Relaciones 1:N. Permite vincular la facturación de un envío al cliente y registrar los pagos realizados.

3.3. Consideraciones adicionales.

Validación de atributos: Pesos, volúmenes, capacidades, cantidades y totales no pueden ser negativos.

Fechas coherentes: La fecha de llegada de un envío no puede ser anterior a la de salida, y las fechas de vigencia de tarifas deben ser correctas.

Integridad referencial: Claves primarias y foráneas aseguran consistencia. Entidades débiles utilizan ON DELETE CASCADE.

Control de herencia: Triggers garantizan que transportistas se asignen solo a subtipos correspondientes y que el tipo coincida con la ruta.

Estados controlados: Campos con valores permitidos (estado, gravedad, tipo, modo, tipo_movimiento) evitan errores de captura.

Trazabilidad: Cada envío se vincula a cliente, productos, contenedores, rutas, transportistas, aduanas, incidencias y pagos.

Escalabilidad: Posibilidad de agregar nuevos tipos de transporte, terminales, almacenes, productos o clientes sin afectar integridad.

Historial de operaciones: Se registran tarifas históricas, movimientos de almacén y eventos de tracking para análisis y auditorías.

Control de inventarios: Línea de envío, contenedores y movimientos de almacén permiten asegurar que la carga no exceda capacidades físicas.

Coherencia semántica: Se asegura que un transportista asignado a una ruta tenga el tipo de transporte adecuado, evitando inconsistencias.

Gestión de duplicidad: Campos únicos como correo electrónico del cliente, matrícula de vehículos, buques y aeronaves evitan registros duplicados.

Manejo de dependencias: Entidades débiles como línea de envío, incidencias, tracking y movimientos de almacén dependen de su entidad padre, garantizando consistencia.

Seguridad de datos: La separación de entidades por tipos y jerarquías permite asignar permisos granulares según la operación y el usuario.

4. Scripts sql.

En este apartado se presenta la implementación en SQL de la base de datos Logística_Internacional. Se incluyen todas las instrucciones necesarias para crear la base de datos, definir su esquema y construir las tablas con sus relaciones, restricciones, entidades débiles y fuertes, así como las especializaciones de transportistas y las relaciones complejas como la relación ternaria de asignaciones. Se incorporan también triggers para garantizar la integridad semántica y la coherencia de los datos, siguiendo las reglas del modelo ER previamente definido.

4.1. Creación de la base de datos.

En este subapartado se describe la creación del esquema de la base de datos Logística_Internacional. Se incluyen instrucciones para eliminar cualquier base de datos previa con el mismo nombre, crear la nueva base de datos y establecer el esquema público donde se definirán todas las tablas y relaciones. Este paso garantiza que el entorno esté limpio y preparado para la implementación del modelo relacional.

```SQL
-- Creación de la base de datos
DROP DATABASE IF EXISTS Logistica_Internacional;
CREATE DATABASE Logistica_Internacional;

-- Conexión al esquema público
```

```
\c Logistica_Internacional
DROP SCHEMA public CASCADE;
CREATE SCHEMA public;
```

4.2. Creación de las tablas.

Aquí se implementa el modelo lógico de la base de datos definido en el modelo entidad–relación. Se crean las tablas maestras, que representan entidades independientes como país, cliente, producto o contenedor, así como los transportistas con su estructura de herencia. También se definen las tablas de infraestructura y rutas, operaciones principales de envío, seguimiento, incidencias, facturación y almacenes. Cada tabla incluye claves primarias, referencias a otras tablas y restricciones básicas para garantizar la integridad estructural de los datos.

```SQL
-- 1. Tablas Maestras (Independientes)
CREATE TABLE pais (
    id_pais SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    iso CHAR(2) UNIQUE
);

CREATE TABLE cliente (
    id_cliente SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    direccion VARCHAR(200),
    email VARCHAR(120) UNIQUE,
    telefono VARCHAR(30)
);

CREATE TABLE tipo_incidencia (
    id_tipo_incidencia SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    gravedad VARCHAR(20) CHECK (gravedad IN ('Leve', 'Media', 'Grave'))
);

CREATE TABLE contenedor (
    id_contenedor SERIAL PRIMARY KEY,
    tipo VARCHAR(50),
    capacidad NUMERIC(10,2),
    tara NUMERIC(10,2)
);

CREATE TABLE producto (
```

```
    id_producto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    peso NUMERIC(10,2),
    volumen NUMERIC(10,2)
);

-- 2. Herencia (IS_A) - Supertipo
CREATE TABLE transportista (
    id_transportista SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    tipo VARCHAR(20) CHECK (tipo IN ('Aereo','Maritimo','Terrestre'))
);

-- 3. Herencia (IS_A) - Subtipos
CREATE TABLE transportista_aereo (
    id_transportista INT PRIMARY KEY REFERENCES
transportista(id_transportista) ON DELETE CASCADE,
    codigo_iata VARCHAR(10)
);

CREATE TABLE transportista_maritimo (
    id_transportista INT PRIMARY KEY REFERENCES
transportista(id_transportista) ON DELETE CASCADE,
    codigo_imo VARCHAR(20)
);

CREATE TABLE transportista_terrestre (
    id_transportista INT PRIMARY KEY REFERENCES
transportista(id_transportista) ON DELETE CASCADE,
    matricula_base VARCHAR(20)
);

-- 4. Infraestructura y Rutas
CREATE TABLE terminal (
    id_terminal SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    tipo VARCHAR(20) CHECK (tipo IN ('Aereo','Maritimo','Terrestre')),
    id_pais INT REFERENCES pais(id_pais)
);

CREATE TABLE ruta (
    id_ruta SERIAL PRIMARY KEY,
    origen_terminal INT REFERENCES terminal(id_terminal),
    destino_terminal INT REFERENCES terminal(id_terminal),
    distancia NUMERIC(10,2),
    -- CORRECCION: Eliminado 'Aereo' duplicado del check original
    modo VARCHAR(20) CHECK (modo IN ('Aereo','Maritimo','Terrestre')),
```

```
    -- RESTRICCIÓN: Origen != Destino
    CONSTRAINT chk_ruta_distinta CHECK (origen_terminal <> destino_terminal)
);

-- CORRECCIÓN: Tabla Tarifa ahora vinculada a Ruta (antes estaba aislada)
CREATE TABLE tarifa (
    id_tarifa SERIAL PRIMARY KEY,
    id_ruta INT REFERENCES ruta(id_ruta) ON DELETE CASCADE,
    tipo_carga VARCHAR(50),
    precio_base NUMERIC(10,2),
    vigencia_inicio DATE,
    vigencia_fin DATE
);

-- 5. Operaciones Principales (Envios)
CREATE TABLE envio (
    id_envio SERIAL PRIMARY KEY,
    id_cliente INT NOT NULL REFERENCES cliente(id_cliente),
    fecha_salida DATE,
    fecha_llegada DATE,
    estado VARCHAR(30) CHECK (estado IN ('Pendiente','En
tránsito','Entregado','Retrasado'))
);

-- ENTIDAD DÉBIL: linea_envio
CREATE TABLE linea_envio (
    id_linea_envio SERIAL PRIMARY KEY,
    id_envio INT NOT NULL REFERENCES envio(id_envio) ON DELETE CASCADE,
    id_producto INT NOT NULL REFERENCES producto(id_producto),
    cantidad INT NOT NULL CHECK (cantidad > 0)
);

CREATE TABLE envio_contenedor (
    id_envio INT REFERENCES envio(id_envio) ON DELETE CASCADE,
    id_contenedor INT REFERENCES contenedor(id_contenedor),
    PRIMARY KEY (id_envio, id_contenedor)
);

-- RELACIÓN TERNARIA: Asignación
CREATE TABLE asignacion (
    id_asignacion SERIAL PRIMARY KEY,
    id_envio INT NOT NULL REFERENCES envio(id_envio) ON DELETE CASCADE,
    id_ruta INT NOT NULL REFERENCES ruta(id_ruta),
    id_transportista INT NOT NULL REFERENCES
transportista(id_transportista),
    fecha_programada DATE,
    estado VARCHAR(30)
);
```

```sql
-- 6. Seguimiento e Incidencias
CREATE TABLE trackingevent (
    id_evento SERIAL PRIMARY KEY,
    id_envio INT REFERENCES envio(id_envio),
    id_terminal INT REFERENCES terminal(id_terminal),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    descripcion TEXT
);

CREATE TABLE incidencia (
    id_incidencia SERIAL PRIMARY KEY,
    id_envio INT REFERENCES envio(id_envio),
    id_tipo_incidencia INT REFERENCES tipo_incidencia(id_tipo_incidencia),
    descripcion TEXT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE aduana (
    id_aduana SERIAL PRIMARY KEY,
    id_envio INT REFERENCES envio(id_envio),
    fecha_inicio DATE,
    fecha_fin DATE,
    estado VARCHAR(20)
);

-- 7. Facturación y Almacén
CREATE TABLE factura (
    id_factura SERIAL PRIMARY KEY,
    id_cliente INT REFERENCES cliente(id_cliente),
    id_envio INT REFERENCES envio(id_envio), -- Vinculada al envio
    fecha_emision DATE,
    total NUMERIC(10,2)
);

CREATE TABLE pago (
    id_pago SERIAL PRIMARY KEY,
    id_factura INT REFERENCES factura(id_factura) ON DELETE CASCADE,
    fecha DATE,
    cantidad NUMERIC(10,2),
    metodo VARCHAR(30)
);

CREATE TABLE almacen (
    id_almacen SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    id_pais INT REFERENCES pais(id_pais)
);
```

```sql
CREATE TABLE movimiento_almacen (
    id_movimiento SERIAL PRIMARY KEY,
    id_envio INT REFERENCES envio(id_envio),
    id_almacen INT REFERENCES almacen(id_almacen),
    fecha TIMESTAMP,
    tipo VARCHAR(20) CHECK (tipo IN ('Entrada','Salida'))
);

CREATE TABLE vehiculo (
    id_vehiculo SERIAL PRIMARY KEY,
    id_transportista INT REFERENCES
transportista_terrestre(id_transportista),
    matricula VARCHAR(20),
    capacidad NUMERIC(10,2)
);

CREATE TABLE buque (
    id_buque SERIAL PRIMARY KEY,
    id_transportista INT REFERENCES
transportista_maritimo(id_transportista),
    nombre VARCHAR(100),
    capacidad NUMERIC(10,2)
);

CREATE TABLE aeronave (
    id_aeronave SERIAL PRIMARY KEY,
    id_transportista INT REFERENCES transportista_aereo(id_transportista),
    matricula VARCHAR(20),
    capacidad NUMERIC(10,2)
);
```

4.3. Triggers.

En este subapartado se presentan los triggers implementados en la base de datos Logística_Internacional. Los triggers son mecanismos que permiten ejecutar automáticamente validaciones o acciones antes o después de realizar operaciones de inserción, actualización o borrado. Su propósito es reforzar la integridad semántica, garantizar la coherencia de los datos y asegurar que las reglas de negocio definidas en el modelo entidad–relación se cumplan en todo momento, más allá de las restricciones de esquema.

```sql
SQL
-- ------------------------------------------------------------
```

```sql
-- TRIGGER 1: Implementación de Exclusividad en Herencia (IS_A)
-- Garantiza que un transportista 'Aereo' no se inserte en tabla 'Maritimo'
-- ------------------------------------------------------------
CREATE OR REPLACE FUNCTION check_transportista_type() RETURNS TRIGGER AS $$
DECLARE
    tipo_real VARCHAR;
BEGIN
    SELECT tipo INTO tipo_real FROM transportista WHERE id_transportista =
NEW.id_transportista;

    IF (TG_TABLE_NAME = 'transportista_aereo' AND tipo_real <> 'Aereo') THEN
        RAISE EXCEPTION 'Error: ID corresponde a otro tipo de transporte, no
Aereo.';
    ELSIF (TG_TABLE_NAME = 'transportista_maritimo' AND tipo_real <>
'Maritimo') THEN
        RAISE EXCEPTION 'Error: ID corresponde a otro tipo de transporte, no
Maritimo.';
    ELSIF (TG_TABLE_NAME = 'transportista_terrestre' AND tipo_real <>
'Terrestre') THEN
        RAISE EXCEPTION 'Error: ID corresponde a otro tipo de transporte, no
Terrestre.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_validate_aereo BEFORE INSERT OR UPDATE ON
transportista_aereo
FOR EACH ROW EXECUTE FUNCTION check_transportista_type();

CREATE TRIGGER trg_validate_maritimo BEFORE INSERT OR UPDATE ON
transportista_maritimo
FOR EACH ROW EXECUTE FUNCTION check_transportista_type();

CREATE TRIGGER trg_validate_terrestre BEFORE INSERT OR UPDATE ON
transportista_terrestre
FOR EACH ROW EXECUTE FUNCTION check_transportista_type();
-- ------------------------------------------------------------
-- TRIGGER 2: Validación Semántica en Relación Ternaria
-- Garantiza que si la ruta es Aerea, el transportista sea Aereo
-- ------------------------------------------------------------
CREATE OR REPLACE FUNCTION check_asignacion_match() RETURNS TRIGGER AS $$
DECLARE
    modo_ruta VARCHAR;
    tipo_transp VARCHAR;
BEGIN
    SELECT modo INTO modo_ruta FROM ruta WHERE id_ruta = NEW.id_ruta;
```

```
    SELECT tipo INTO tipo_transp FROM transportista WHERE id_transportista =
NEW.id_transportista;

    IF (modo_ruta <> tipo_transp) THEN
        RAISE EXCEPTION 'Incoherencia semántica: No se puede asignar un
transportista % a una ruta %', tipo_transp, modo_ruta;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_check_asignacion BEFORE INSERT OR UPDATE ON asignacion
FOR EACH ROW EXECUTE FUNCTION check_asignacion_match();
```

4.4. Restricciones y reglas de integridad.

Con el objetivo de garantizar la coherencia, calidad y fiabilidad de la información almacenada en la base de datos Logística_Internacional, se han definido distintas restricciones de integridad a nivel de esquema. Estas restricciones permiten evitar la introducción de datos inválidos, asegurar la consistencia entre entidades relacionadas y hacer cumplir las reglas de negocio establecidas en el modelo entidad–relación.

```SQL
-- ============================================================
-- 5.1. Restricciones sobre valores numéricos
-- (Evitan valores negativos o incoherentes)
-- ============================================================

-- Producto
ALTER TABLE producto
ADD CONSTRAINT chk_producto_peso
CHECK (peso >= 0);

ALTER TABLE producto
ADD CONSTRAINT chk_producto_volumen
CHECK (volumen >= 0);

-- Contenedor
ALTER TABLE contenedor
ADD CONSTRAINT chk_contenedor_capacidad
CHECK (capacidad > 0);
```

```sql
ALTER TABLE contenedor
ADD CONSTRAINT chk_contenedor_tara
CHECK (tara >= 0);
-- Ruta
ALTER TABLE ruta
ADD CONSTRAINT chk_ruta_distancia
CHECK (distancia > 0);

-- Tarifa
ALTER TABLE tarifa
ADD CONSTRAINT chk_tarifa_precio
CHECK (precio_base >= 0);

-- Vehículo
ALTER TABLE vehiculo
ADD CONSTRAINT chk_vehiculo_capacidad
CHECK (capacidad > 0);

-- Buque
ALTER TABLE buque
ADD CONSTRAINT chk_buque_capacidad
CHECK (capacidad > 0);

-- Aeronave
ALTER TABLE aeronave
ADD CONSTRAINT chk_aeronave_capacidad
CHECK (capacidad > 0);

-- Factura
ALTER TABLE factura
ADD CONSTRAINT chk_factura_total
CHECK (total >= 0);

-- Pago
ALTER TABLE pago
ADD CONSTRAINT chk_pago_cantidad
CHECK (cantidad > 0);

-- ============================================================
-- 5.2. Restricciones en entidades débiles
-- (Dependencia existencial y valores válidos)
-- ============================================================

-- Línea de Envío
ALTER TABLE linea_envio
ADD CONSTRAINT chk_linea_envio_cantidad
CHECK (cantidad > 0);
```

```sql
-- ============================================================
-- 5.3. Restricciones temporales (coherencia de fechas)
-- ============================================================

-- Envío: la fecha de llegada no puede ser anterior a la salida
ALTER TABLE envio
ADD CONSTRAINT chk_fechas_envio
CHECK (
    fecha_llegada IS NULL
    OR fecha_llegada >= fecha_salida
);
```


```sql
-- ============================================================
-- 5.3. Restricciones temporales (coherencia de fechas)
-- ============================================================

-- Envío: la fecha de llegada no puede ser anterior a la salida
ALTER TABLE envio
ADD CONSTRAINT chk_fechas_envio
CHECK (
    fecha_llegada IS NULL
    OR fecha_llegada >= fecha_salida
);
```


```sql
-- Tarifa: vigencia correcta
ALTER TABLE tarifa
ADD CONSTRAINT chk_tarifa_fechas
CHECK (
    vigencia_fin IS NULL
    OR vigencia_fin >= vigencia_inicio
);

-- Aduana: fechas de inicio y fin coherentes
ALTER TABLE aduana
ADD CONSTRAINT chk_aduana_fechas
CHECK (
    fecha_fin IS NULL
    OR fecha_fin >= fecha_inicio
);

-- ============================================================
-- 5.4. Restricciones de dominios cerrados
-- (Valores permitidos según reglas de negocio)
-- ============================================================

-- Envío
ALTER TABLE envio
ADD CONSTRAINT chk_envio_estado
```

```
CHECK (estado IN ('Pendiente','En tránsito','Entregado','Retrasado'));

-- Tipo de Incidencia
ALTER TABLE tipo_incidencia
ADD CONSTRAINT chk_gravedad_incidencia
CHECK (gravedad IN ('Leve','Media','Grave'));

-- Terminal
ALTER TABLE terminal
ADD CONSTRAINT chk_terminal_tipo
CHECK (tipo IN ('Aereo','Maritimo','Terrestre'));

-- Ruta
ALTER TABLE ruta
ADD CONSTRAINT chk_ruta_modo
CHECK (modo IN ('Aereo','Maritimo','Terrestre'));

-- Movimiento de Almacén
ALTER TABLE movimiento_almacen
ADD CONSTRAINT chk_movimiento_tipo
CHECK (tipo IN ('Entrada','Salida'));


-- Transportista
ALTER TABLE transportista
ADD CONSTRAINT chk_transportista_tipo
CHECK (tipo IN ('Aereo','Maritimo','Terrestre'));

-- ============================================================
-- 5.5. Restricciones estructurales adicionales
-- ============================================================

-- Ruta: el terminal de origen y destino deben ser distintos
ALTER TABLE ruta
ADD CONSTRAINT chk_origen_destino_distintos
CHECK (origen_terminal <> destino_terminal);
```

4.5. Datos de la base de datos.

En este subapartado se presentan los datos iniciales que se insertan en la base de datos Logística_Internacional para probar su funcionamiento y asegurar la correcta implementación de las relaciones, restricciones y reglas de negocio. Estos datos incluyen información sobre países, clientes, productos, transportistas, terminales, rutas, envíos, incidencias, almacenes y activos logísticos como vehículos, buques y aeronaves. La carga

inicial permite realizar consultas de verificación, pruebas de integridad y simulaciones de operaciones reales.

Tabla país:

```sql
INSERT INTO pais (nombre, iso) VALUES
('España', 'ES'),
('Francia', 'FR'),
('Alemania', 'DE'),
('Estados Unidos', 'US'),
('China', 'CN');
```

	id_pais	nombre	iso
1	1	España	ES
2	2	Francia	FR
3	3	Alemania	DE
4	4	Estados Unidos	US
5	5	China	CN

Tabla cliente:

```sql
INSERT INTO cliente (nombre, direccion, email, telefono) VALUES
('TransGlobal S.A.', 'Calle Mayor 10, Madrid, España',
'info@transglobal.es', '+34 912345678'),
('LogiFrance', '10 Rue de la Paix, París, Francia', 'contact@logifrance.fr',
'+33 145678910'),
('Berlin Cargo', 'Alexanderplatz 5, Berlín, Alemania',
'ventas@berlincargo.de', '+49 301234567'),
('US Freight Inc.', '500 Market Street, Nueva York, USA',
'contact@usfreight.com', '+1 2125551234');
```

	id_cliente	nombre	direccion	email	telefono
1	1	TransGlobal S.A.	Calle Mayor 10, Madrid, España	info@transglobal.es	+34 912345678
2	2	LogiFrance	10 Rue de la Paix, París, Francia	contact@logifrance.fr	+33 145678910
3	3	Berlin Cargo	Alexanderplatz 5, Berlín, Alemania	ventas@berlincargo.de	+49 301234567
4	4	US Freight Inc.	500 Market Street, Nueva York, USA	contact@usfreight.com	+1 2125551234

Tabla tipo de incidencia:

```sql
INSERT INTO tipo_incidencia (nombre, gravedad) VALUES
('Retraso por clima', 'Media'),
('Documento aduanero incompleto', 'Grave'),
('Daño en contenedor', 'Grave'),
('Cambio de ruta', 'Leve');
```

	id_tipo_incidencia	nombre	gravedad
1	1	Retraso por clima	Media
2	2	Documento aduanero incompleto	Grave
3	3	Daño en contenedor	Grave
4	4	Cambio de ruta	Leve

Tabla contenedores:

```sql
INSERT INTO contenedor (tipo, capacidad, tara) VALUES
('Contenedor 20 pies', 28.0, 2.3),
('Contenedor 40 pies', 58.0, 3.5),
('Contenedor Refrigerado', 25.0, 3.0);
```

	id_contenedor	tipo	capacidad	tara
1	1	Contenedor 20 pies	28	2,3
2	2	Contenedor 40 pies	58	3,5
3	3	Contenedor Refrigerado	25	3

Tabla productos:

```sql
INSERT INTO producto (nombre, descripcion, peso, volumen) VALUES
('Ordenador portátil', 'Portátil empresarial 15"', 2.5, 0.004),
('Teléfono móvil', 'Smartphone gama media', 0.3, 0.001),
('Monitor', 'Monitor LED 24 pulgadas', 5.2, 0.06),
('Impresora', 'Impresora láser', 8.0, 0.12),
('Mesa oficina', 'Mesa de madera', 20.0, 0.8),
('Silla oficina', 'Silla ergonómica', 10.0, 0.6);
```

	id_producto	nombre	descripcion	peso	volumen
1	1	Ordenador portátil	Portátil empresarial 15"	2,5	0
2	2	Teléfono móvil	Smartphone gama media	0,3	0
3	3	Monitor	Monitor LED 24 pulgadas	5,2	0,06
4	4	Impresora	Impresora láser	8	0,12
5	5	Mesa oficina	Mesa de madera	20	0,8
6	6	Silla oficina	Silla ergonómica	10	0,6

Tabla Transportista y subtipos:

```sql
INSERT INTO transportista (nombre, tipo) VALUES
('AirGlobal', 'Aereo'),
('SkyLogistics', 'Aereo'),
('Oceanic Lines', 'Maritimo'),
('BlueSea Cargo', 'Maritimo'),
('RoadExpress', 'Terrestre'),
('EuroTruck', 'Terrestre');

INSERT INTO transportista_aereo (id_transportista, codigo_iata) VALUES
(1, 'AGL'),
(2, 'SKY');

INSERT INTO transportista_maritimo (id_transportista, codigo_imo) VALUES
(3, 'IMO1234567'),
(4, 'IMO7654321');

INSERT INTO transportista_terrestre (id_transportista, matricula_base)
VALUES
(5, 'RE-BASE'),
(6, 'ET-BASE');
```

	id_transportista	nombre	tipo
1	1	AirGlobal	Aereo
2	2	SkyLogistics	Aereo
3	3	Oceanic Lines	Maritimo
4	4	BlueSea Cargo	Maritimo
5	5	RoadExpress	Terrestre
6	6	EuroTruck	Terrestre

	id_transportista	codigo_iata
1	1	AGL
2	2	SKY

	id_transportista	codigo_imo
1	3	IMO1234567
2	4	IMO7654321

	id_transportista	matricula_base
1	5	RE-BASE
2	6	ET-BASE

Tabla Terminal:

```sql
INSERT INTO terminal (nombre, tipo, id_pais) VALUES
('Aeropuerto Madrid-Barajas', 'Aereo', 1),
('Aeropuerto París-CDG', 'Aereo', 2),
('Aeropuerto Berlín-Brandenburgo', 'Aereo', 3),
('Puerto de Hamburgo', 'Maritimo', 3),
('Puerto de Shanghái', 'Maritimo', 5),
('Puerto de Los Ángeles', 'Maritimo', 4),
('Terminal Madrid Norte', 'Terrestre', 1),
('Terminal París Este', 'Terrestre', 2),
('Terminal Berlín Oeste', 'Terrestre', 3),
('Terminal Nueva York', 'Terrestre', 4);
```

	id_terminal	nombre	tipo	id_pais
1	1	Aeropuerto Madrid-Barajas	Aereo	1
2	2	Aeropuerto París-CDG	Aereo	2
3	3	Aeropuerto Berlín-Brandenburgo	Aereo	3
4	4	Puerto de Hamburgo	Maritimo	3
5	5	Puerto de Shanghái	Maritimo	5
6	6	Puerto de Los Ángeles	Maritimo	4
7	7	Terminal Madrid Norte	Terrestre	1
8	8	Terminal París Este	Terrestre	2
9	9	Terminal Berlín Oeste	Terrestre	3
10	10	Terminal Nueva York	Terrestre	4

Tabla ruta:

```sql
INSERT INTO ruta (origen_terminal, destino_terminal, distancia, modo) VALUES
(1, 2, 1050, 'Aereo'),
(2, 3, 880, 'Aereo'),
(4, 5, 20000, 'Maritimo'),
(5, 6, 11000, 'Maritimo'),
(7, 8, 1200, 'Terrestre'),
(8, 9, 1000, 'Terrestre'),
(9, 10, 6500, 'Terrestre');
```

	id_ruta	origen_terminal	destino_terminal	distancia	modo
1	1	1	2	1.050	Aereo
2	2	2	3	880	Aereo
3	3	4	5	20.000	Maritimo
4	4	5	6	11.000	Maritimo
5	5	7	8	1.200	Terrestre
6	6	8	9	1.000	Terrestre
7	7	9	10	6.500	Terrestre

Tabla tarifa:

```sql
INSERT INTO tarifa (id_ruta, tipo_carga, precio_base, vigencia_inicio,
vigencia_fin) VALUES
(1, 'Electrónica', 3000, '2024-01-01', '2024-12-31'),
(2, 'Oficina', 2500, '2024-01-01', '2024-12-31'),
(3, 'Industrial', 8000, '2024-01-01', '2024-12-31'),
(4, 'Comercial', 7500, '2024-01-01', '2024-12-31'),
(5, 'Muebles', 1500, '2024-01-01', '2024-12-31');
```

	id_tarifa	id_ruta	tipo_carga	precio_base	vigencia_inicio	vigencia_fin
1	1	1	Electrónica	3.000	2024-01-01	2024-12-31
2	2	2	Oficina	2.500	2024-01-01	2024-12-31
3	3	3	Industrial	8.000	2024-01-01	2024-12-31
4	4	4	Comercial	7.500	2024-01-01	2024-12-31
5	5	5	Muebles	1.500	2024-01-01	2024-12-31
6	6	1	Electrónica	3.000	2024-01-01	2024-12-31
7	7	2	Oficina	2.500	2024-01-01	2024-12-31
8	8	3	Industrial	8.000	2024-01-01	2024-12-31
9	9	4	Comercial	7.500	2024-01-01	2024-12-31
10	10	5	Muebles	1.500	2024-01-01	2024-12-31

Tabla envio:

```sql
INSERT INTO envio (id_cliente, fecha_salida, fecha_llegada, estado) VALUES
(1, '2024-03-01', '2024-03-03', 'Entregado'),
(2, '2024-03-05', '2024-03-07', 'En tránsito'),
(3, '2024-03-10', '2024-03-20', 'Pendiente'),
(4, '2024-03-12', '2024-03-25', 'Retrasado');
```

	id_envio	id_cliente	fecha_salida	fecha_llegada	estado
1	1	1	2024-03-01	2024-03-03	Entregado
2	2	2	2024-03-05	2024-03-07	En tránsito
3	3	3	2024-03-10	2024-03-20	Pendiente
4	4	4	2024-03-12	2024-03-25	Retrasado

Tabla linea_envio:

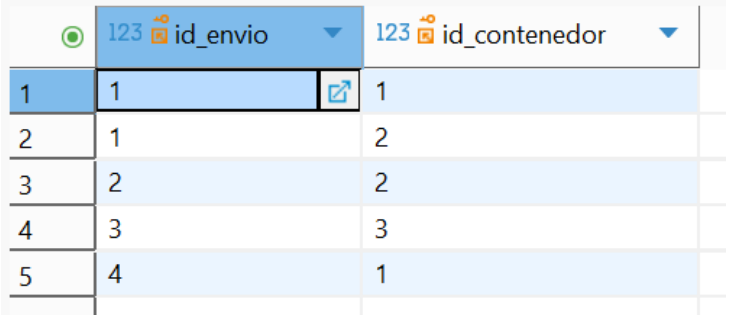
```sql
INSERT INTO linea_envio (id_envio, id_producto, cantidad) VALUES
(1, 1, 10),
(1, 2, 20),
(2, 3, 15),
(2, 4, 5),
(3, 5, 8),
(4, 6, 12);
```

	id_linea_envio	id_envio	id_producto	cantidad
1	1	1	1	10
2	2	1	2	20
3	3	2	3	15
4	4	2	4	5
5	5	3	5	8
6	6	4	6	12

Tabla envio_contenedor:

```sql
INSERT INTO envio_contenedor (id_envio, id_contenedor) VALUES
(1, 1),
```

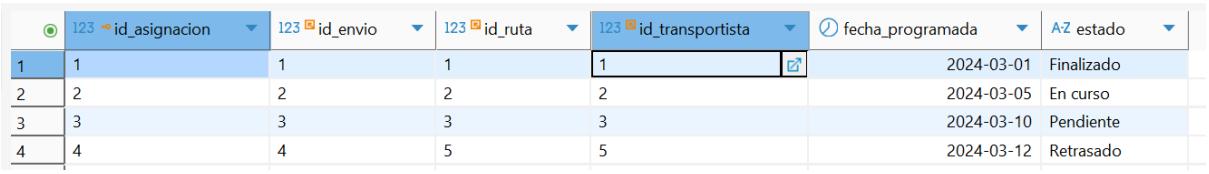
```
(1, 2),
(2, 2),
(3, 3),
(4, 1);
```



	id_envio	id_contenedor
1	1	1
2	1	2
3	2	2
4	3	3
5	4	1

Tabla asignación:

```SQL
INSERT INTO asignacion (id_envio, id_ruta, id_transportista,
fecha_programada, estado) VALUES
(1, 1, 1, '2024-03-01', 'Finalizado'),
(2, 2, 2, '2024-03-05', 'En curso'),
(3, 3, 3, '2024-03-10', 'Pendiente'),
(4, 5, 5, '2024-03-12', 'Retrasado');
```



	id_asignacion	id_envio	id_ruta	id_transportista	fecha_programada	estado
1	1	1	1	1	2024-03-01	Finalizado
2	2	2	2	2	2024-03-05	En curso
3	3	3	3	3	2024-03-10	Pendiente
4	4	4	5	5	2024-03-12	Retrasado

Tabla trackingevent:

```SQL
INSERT INTO trackingevent (id_envio, id_terminal, descripcion) VALUES
(1, 1, 'Salida del aeropuerto de Madrid'),
(1, 2, 'Llegada al aeropuerto de París'),
(2, 2, 'Salida de París'),
(3, 4, 'Carga en puerto de Hamburgo');
```

	id_evento	id_envio	id_terminal	fecha	descripcion
1	1	1	1	2025-12-13 15:30:06.305	Salida del aeropuerto de Madrid
2	2	1	2	2025-12-13 15:30:06.305	Llegada al aeropuerto de París
3	3	2	2	2025-12-13 15:30:06.305	Salida de París
4	4	3	4	2025-12-13 15:30:06.305	Carga en puerto de Hamburgo

Tabla incidencia:

```SQL
INSERT INTO incidencia (id_envio, id_tipo_incidencia, descripcion) VALUES
(4, 1, 'Retraso por condiciones meteorológicas'),
(3, 2, 'Documentación incompleta en aduana');
```

	id_incidencia	id_envio	id_tipo_incidencia	descripcion	fecha
1	1	4	1	Retraso por condiciones meteorológicas	2025-12-13 15:30:21.880
2	2	3	2	Documentación incompleta en aduana	2025-12-13 15:30:21.880

Tabla aduana:

```SQL
INSERT INTO aduana (id_envio, fecha_inicio, fecha_fin, estado) VALUES
(1, '2024-03-01', '2024-03-02', 'Aprobado'),
(3, '2024-03-11', NULL, 'En revisión');
```

	id_aduana	id_envio	fecha_inicio	fecha_fin	estado
1	7	1	2024-03-01	2024-03-02	Aprobado
2	8	3	2024-03-11	[NULL]	En revisión

Tabla almacen:

```SQL
INSERT INTO almacen (nombre, id_pais) VALUES
('Almacén Madrid', 1),
('Almacén París', 2),
('Almacén Berlín', 3),
('Almacén Shanghái', 5);
```

	id_almacen	nombre	id_pais
1	1	Almacén Madrid	1
2	2	Almacén París	2
3	3	Almacén Berlín	3
4	4	Almacén Shanghái	5

Tabla movimiento_almacen:

```sql
INSERT INTO movimiento_almacen (id_envio, id_almacen, fecha, tipo) VALUES
(1, 1, '2024-03-01 08:00', 'Entrada'),
(1, 1, '2024-03-03 15:00', 'Salida'),
(2, 2, '2024-03-05 09:00', 'Entrada'),
(3, 4, '2024-03-11 10:30', 'Entrada');
```

	id_movimiento	id_envio	id_almacen	fecha	tipo
1	1	1	1	2024-03-01 08:00:00.000	Entrada
2	2	1	1	2024-03-03 15:00:00.000	Salida
3	3	2	2	2024-03-05 09:00:00.000	Entrada
4	4	3	4	2024-03-11 10:30:00.000	Entrada

Tabla factura:

```sql
INSERT INTO factura (id_cliente, id_envio, fecha_emision, total) VALUES
(1, 1, '2024-03-03', 12000),
(2, 2, '2024-03-07', 9000),
(3, 3, '2024-03-20', 18000);
```

	id_factura	id_cliente	id_envio	fecha_emision	total
1	1	1	1	2024-03-03	12.000
2	2	2	2	2024-03-07	9.000
3	3	3	3	2024-03-20	18.000

Tabla pago:

```sql
INSERT INTO pago (id_factura, fecha, cantidad, metodo) VALUES
```

```
(1, '2024-03-03', 12000, 'Transferencia'),
(2, '2024-03-08', 9000, 'Tarjeta'),
(3, '2024-03-21', 10000, 'Transferencia');
```

	id_pago	id_factura	fecha	cantidad	metodo
1	1	1	2024-03-03	12.000	Transferencia
2	2	2	2024-03-08	9.000	Tarjeta
3	3	3	2024-03-21	10.000	Transferencia

Tabla Vehículos / Buques / Aeronaves:

```
SQL
INSERT INTO vehiculo (id_transportista, matricula, capacidad) VALUES
(5, 'RE-1234', 18000),
(6, 'ET-5678', 22000);

INSERT INTO buque (id_transportista, nombre, capacidad) VALUES
(3, 'Atlantic Carrier', 60000),
(4, 'Pacific Trader', 55000);

INSERT INTO aeronave (id_transportista, matricula, capacidad) VALUES
(1, 'AG-001', 2500),
(2, 'SK-002', 3000);
```

	id_buque	id_transportista	nombre	capacidad
1	3	3	Atlantic Carrier	60.000
2	4	4	Pacific Trader	55.000

	id_vehiculo	id_transportista	matricula	capacidad
1	1	5	RE-1234	18.000
2	2	6	ET-5678	22.000

	id_aeronave	id_transportista	matricula	capacidad
1	3	1	AG-001	2.500
2	4	2	SK-002	3.000
3	5	1	AG-001	2.500
4	6	2	SK-002	3.000
5	7	1	AG-001	2.500
6	8	2	SK-002	3.000

5. Ejemplos funcionamiento de la base de datos.

Este apartado presenta consultas ilustrativas diseñadas para verificar el correcto funcionamiento de la base de datos. Se incluyen consultas de verificación de relaciones, pruebas de integridad mediante inserciones erróneas, validaciones de reglas de negocio y consultas de explotación real. Estas consultas permiten comprobar la coherencia, consistencia y aplicabilidad de los datos según las reglas definidas en el modelo entidad–relación.

CONSULTAS DE VERIFICACIÓN DE RELACIONES.

Estas consultas permiten validar que las relaciones entre entidades se han implementado correctamente. Por ejemplo, se puede comprobar que cada envío está asociado al cliente correcto, que los productos están vinculados a sus envíos y que las rutas tienen origen y destino válidos.

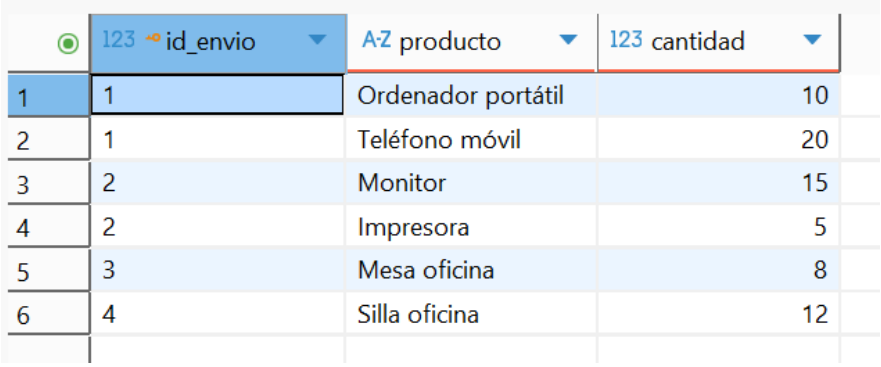
- **Comprobamos los envíos con sus respectivos clientes.**

```sql
SELECT c.nombre AS cliente, e.id_envio, e.estado, e.fecha_salida
FROM envio e
JOIN cliente c ON e.id_cliente = c.id_cliente;
```

	cliente	id_envio	estado	fecha_salida
1	TransGlobal S.A.	1	Entregado	2024-03-01
2	LogiFrance	2	En tránsito	2024-03-05
3	Berlin Cargo	3	Pendiente	2024-03-10
4	US Freight Inc.	4	Retrasado	2024-03-12

- **Comprobamos los productos asociados a cada envío.**

```sql
SELECT e.id_envio, p.nombre AS producto, l.cantidad
FROM envio e
JOIN linea_envio l ON e.id_envio = l.id_envio
JOIN producto p ON l.id_producto = p.id_producto;
```



	id_envio	producto	cantidad
1	1	Ordenador portátil	10
2	1	Teléfono móvil	20
3	2	Monitor	15
4	2	Impresora	5
5	3	Mesa oficina	8
6	4	Silla oficina	12

- **Comprobamos las rutas con origen y destino:**

```sql
SELECT
  r.id_ruta,
  t1.nombre AS origen,
  t2.nombre AS destino,
  r.modo
FROM ruta r
JOIN terminal t1 ON r.origen_terminal = t1.id_terminal
JOIN terminal t2 ON r.destino_terminal = t2.id_terminal;
```

	id_ruta	origen	destino	modo
1	1	Aeropuerto Madrid-Barajas	Aeropuerto París-CDG	Aereo
2	2	Aeropuerto París-CDG	Aeropuerto Berlín-Brandenburgo	Aereo
3	3	Puerto de Hamburgo	Puerto de Shanghái	Maritimo
4	4	Puerto de Shanghái	Puerto de Los Ángeles	Maritimo
5	5	Terminal Madrid Norte	Terminal París Este	Terrestre
6	6	Terminal París Este	Terminal Berlín Oeste	Terrestre
7	7	Terminal Berlín Oeste	Terminal Nueva York	Terrestre

- **Comprobamos los pagos asociados a cada factura.**

```sql

SELECT
```

```
  f.id_factura,
  p.fecha,
  p.cantidad,
  p.metodo
FROM factura f
JOIN pago p ON f.id_factura = p.id_factura;
```

	id_factura	fecha	cantidad	metodo
1	1	2024-03-03	12.000	Transferencia
2	2	2024-03-08	9.000	Tarjeta
3	3	2024-03-21	10.000	Transferencia

- **Comprobamos las incidencias junto con su tipo y gravedad.**

```
SQL
SELECT
  e.id_envio,
  ti.nombre AS tipo_incidencia,
  ti.gravedad,
  i.descripcion
FROM incidencia i
JOIN envio e ON i.id_envio = e.id_envio
JOIN tipo_incidencia ti ON i.id_tipo_incidencia = ti.id_tipo_incidencia;
```

	id_envio	tipo_incidencia	gravedad	descripcion
1	4	Retraso por clima	Media	Retraso por condiciones meteorológicas
2	3	Documento aduanero incompleto	Grave	Documentación incompleta en aduana

- **Comprobamos los movimientos de almacén por envío.**

```
SQL
SELECT
  e.id_envio,
  a.nombre AS almacen,
  m.tipo,
  m.fecha
FROM movimiento_almacen m
JOIN envio e ON m.id_envio = e.id_envio
```

```
JOIN almacen a ON m.id_almacen = a.id_almacen;
```

id_envio	almacen	tipo	fecha
1	Almacén Madrid	Entrada	2024-03-01 08:00:00.000
1	Almacén Madrid	Salida	2024-03-03 15:00:00.000
2	Almacén París	Entrada	2024-03-05 09:00:00.000
3	Almacén Shanghái	Entrada	2024-03-11 10:30:00.000

- **Comprobamos qué transportista realiza cada envío y por qué ruta**

```
SQL
SELECT
    e.id_envio,
    tr.nombre AS transportista,
    r.id_ruta,
    r.modo,
    a.fecha_programada
FROM asignacion a
JOIN envio e ON a.id_envio = e.id_envio
JOIN transportista tr ON a.id_transportista = tr.id_transportista
JOIN ruta r ON a.id_ruta = r.id_ruta;
```

	id_envio	transportista	id_ruta	modo	fecha_programada
1	1	AirGlobal	1	Aereo	2024-03-01
2	2	SkyLogistics	2	Aereo	2024-03-05
3	3	Oceanic Lines	3	Maritimo	2024-03-10
4	4	RoadExpress	5	Terrestre	2024-03-12

PRUEBAS DE INTEGRIDAD (INSERT ERRÓNEOS)

Este subapartado demuestra cómo la base de datos rechaza datos inválidos que violan restricciones o reglas de negocio. Las consultas de inserción errónea permiten verificar la efectividad de los CHECKs y triggers definidos.

- **Insertar una ruta con mismo origen y destino.**

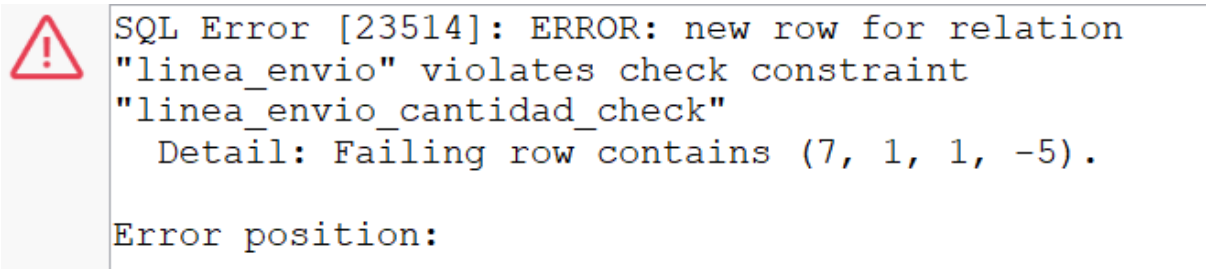
```
SQL
INSERT INTO ruta (origen_terminal, destino_terminal, distancia, modo)
VALUES (1, 1, 500, 'Aereo');
```

```
SQL Error [23514]: ERROR: new row for relation
"ruta" violates check constraint
"chk_ruta_distinta"
  Detail: Failing row contains (8, 1, 1, 500.00,
Aereo).

Error position:
```

- **Insertar una línea de envío con cantidad negativa**

```sql
INSERT INTO linea_envio (id_envio, id_producto, cantidad)
VALUES (1, 1, -5);
```

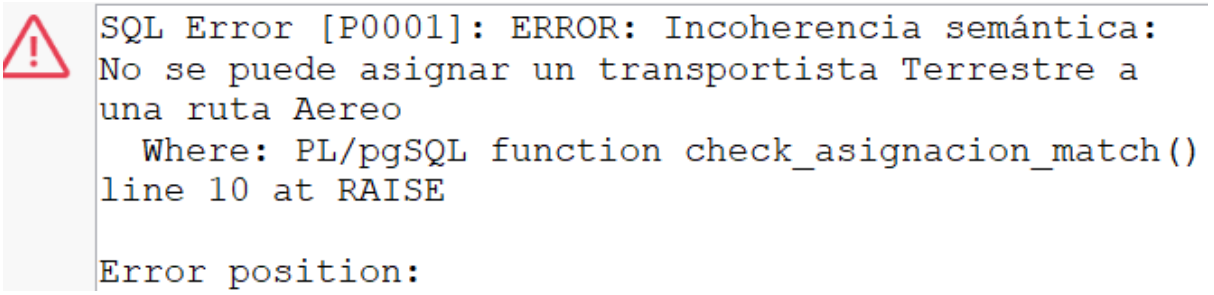


```
SQL Error [23514]: ERROR: new row for relation
"linea_envio" violates check constraint
"linea_envio_cantidad_check"
  Detail: Failing row contains (7, 1, 1, -5).

Error position:
```

- **Asignar transportista terrestre a ruta aérea**

```sql
INSERT INTO asignacion (id_envio, id_ruta, id_transportista,
fecha_programada)
VALUES (1, 1, 5, '2024-04-01');
```



```
SQL Error [P0001]: ERROR: Incoherencia semántica:
No se puede asignar un transportista Terrestre a
una ruta Aereo
  Where: PL/pgSQL function check_asignacion_match()
line 10 at RAISE

Error position:
```

- **Insertar un envío con fecha de llegada anterior a la salida.**

```sql
INSERT INTO envio (id_cliente, fecha_salida, fecha_llegada, estado)
VALUES (1, '2024-06-10', '2024-06-01', 'Pendiente');
```

```
SQL Error [23514]: ERROR: new row for relation
"envio" violates check constraint
"chk_fechas_envio"
  Detail: Failing row contains (6, 1, 2024-06-10,
2024-06-01, Pendiente).

Error position:
```

- **Insertar una tarifa con fechas incorrectas**

```sql
INSERT INTO tarifa (id_ruta, tipo_carga, precio_base, vigencia_inicio,
vigencia_fin)
VALUES (1, 'General', 500, '2024-06-10', '2024-05-01');
```

```
SQL Error [23514]: ERROR: new row for relation
"tarifa" violates check constraint
"chk_tarifa_fechas"
  Detail: Failing row contains (11, 1, General,
500.00, 2024-06-10, 2024-05-01).
```

- **Insertar un movimiento de almacén con tipo inválido**

```sql
INSERT INTO movimiento_almacen (id_envio, id_almacen, fecha, tipo)
VALUES (1, 1, CURRENT_TIMESTAMP, 'Transferencia');
```

```
SQL Error [23514]: ERROR: new row for relation
"movimiento_almacen" violates check constraint
"chk_movimiento_tipo"
  Detail: Failing row contains (5, 1, 1, 2025-12-13
16:37:42.46022, Transferencia).

Error position:
```

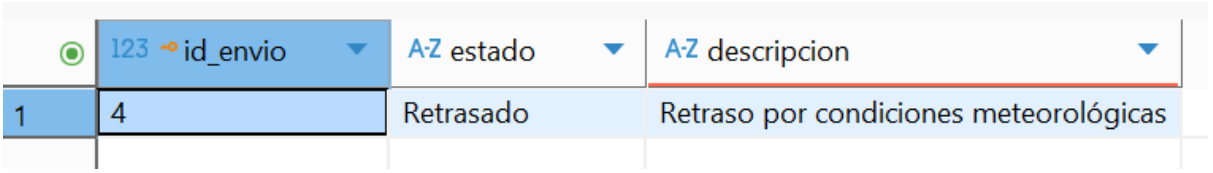
CONSULTAS DE REGLAS DE NEGOCIO.

Este subapartado presenta consultas diseñadas para verificar el cumplimiento de las reglas de negocio definidas en la base de datos. Permiten identificar situaciones específicas que afectan a la operativa logística, como envíos retrasados con incidencias, envíos que aún no

han pasado aduana, envíos con incidencias graves y envíos sin factura generada. Estas consultas no solo validan la coherencia de los datos, sino que también ayudan a detectar excepciones y a garantizar que las operaciones sigan las políticas establecidas por la empresa.

- **Envíos retrasados con incidencias.**

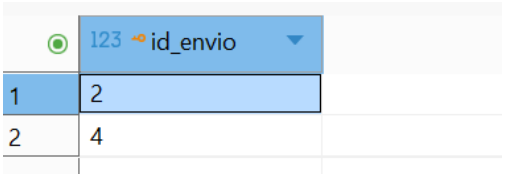
```sql
SELECT e.id_envio, e.estado, i.descripcion
FROM envio e
JOIN incidencia i ON e.id_envio = i.id_envio
WHERE e.estado = 'Retrasado';
```



	id_envio	estado	descripcion
1	4	Retrasado	Retraso por condiciones meteorológicas

- **Envíos que aún no han pasado aduana**

```sql
SELECT e.id_envio
FROM envio e
LEFT JOIN aduana a ON e.id_envio = a.id_envio
WHERE a.id_aduana IS NULL;
```



	id_envio
1	2
2	4

- **Envíos con incidencias graves**

```sql
SELECT
    e.id_envio,
    ti.nombre,
    ti.gravedad
FROM incidencia i
JOIN tipo_incidencia ti ON i.id_tipo_incidencia = ti.id_tipo_incidencia
JOIN envio e ON i.id_envio = e.id_envio
WHERE ti.gravedad = 'Grave';
```

	id_envio	nombre	gravedad
1	3	Documento aduanero incompleto	Grave

- **Envíos sin factura generada**

```sql
SELECT e.id_envio
FROM envio e
LEFT JOIN factura f ON e.id_envio = f.id_envio
WHERE f.id_factura IS NULL;
```

	id_envio
1	4

CONSULTAS DE EXPLOTACIÓN REAL.

Este subapartado incluye consultas orientadas a la obtención de información estratégica y operativa que puede ser utilizada para la toma de decisiones en la gestión logística. Permiten calcular métricas como la facturación total por cliente, el historial de trazabilidad de un envío, la capacidad total utilizada por tipo de transporte, el número de envíos por país de origen y el total facturado por envío. Estas consultas muestran cómo los datos almacenados en la base de datos pueden transformarse en información útil para análisis, planificación y optimización de los procesos logísticos.

- **Facturación total por cliente.**

```sql
SELECT c.nombre, SUM(f.total) AS facturacion_total
FROM cliente c
JOIN factura f ON c.id_cliente = f.id_cliente
GROUP BY c.nombre;
```

	nombre	facturacion_total
1	LogiFrance	9.000
2	Berlin Cargo	18.000
3	TransGlobal S.A.	12.000

- **Historial de trazabilidad de un envío**

```sql
SELECT e.id_envio, t.nombre AS terminal, te.fecha, te.descripcion
FROM trackingevent te
JOIN envio e ON te.id_envio = e.id_envio
JOIN terminal t ON te.id_terminal = t.id_terminal
WHERE e.id_envio = 1
ORDER BY te.fecha;
```

	id_envio	terminal	fecha	descripcion
1	1	Aeropuerto Madrid-Barajas	2025-12-13 15:30:06.305	Salida del aeropuerto de Madrid
2	1	Aeropuerto París-CDG	2025-12-13 15:30:06.305	Llegada al aeropuerto de París

- **Número de asignaciones por tipo de transporte**

```sql
SELECT r.modo, COUNT(a.id_asignacion) AS total_asignaciones
FROM asignacion a
JOIN ruta r ON a.id_ruta = r.id_ruta
GROUP BY r.modo;
```

	modo	total_asignaciones
1	Terrestre	1
2	Aereo	2
3	Maritimo	1

- **Número de envíos por país de origen**

```sql
SELECT
    p.nombre AS pais,
    COUNT(e.id_envio) AS total_envios
FROM envio e
JOIN asignacion a ON e.id_envio = a.id_envio
JOIN ruta r ON a.id_ruta = r.id_ruta
JOIN terminal t ON r.origen_terminal = t.id_terminal
JOIN pais p ON t.id_pais = p.id_pais
GROUP BY p.nombre;
```

	pais	total_envios
1	España	2
2	Alemania	1
3	Francia	1

- **Total facturado por envío**

```sql
SELECT
    e.id_envio,
    SUM(f.total) AS total_facturado
FROM envio e
JOIN factura f ON e.id_envio = f.id_envio
GROUP BY e.id_envio;
```

	id_envio	total_facturado
1	2	9.000
2	3	18.000
3	1	12.000

6. Uso de la API

La implementación de la API se ha hecho utilizando el siguiente stack tecnológico:

- Lenguaje: Python 3.12.
- Framework Web: Flask
- Conector de Base de Datos: psycopg2-binary (para la comunicación eficiente entre Python y PostgreSQL).
- Entorno de Ejecución: Servidor Linux (Ubuntu) con gestión de entornos virtuales (venv) para el aislamiento de dependencias.
- Herramienta de Pruebas: Thunder Client (extensión de VS Code).

6.1 Arquitectura y Configuración

El proyecto se ha estructurado en torno a un controlador principal (app.py) que gestiona las rutas HTTP y la lógica de negocio. La aplicación se ha configurado para escuchar en la dirección 0.0.0.0 y el puerto 8080, garantizando su accesibilidad tanto desde la máquina local (localhost) como desde clientes externos en la red.
Para hacer que la API funcione correctamente, hemos hecho lo siguiente:

- **Creación de un entorno virtual:**
  Esto lo hemos hecho con el fin de instalar los paquetes. Esto se puede hacer ejecutando los siguientes comandos:

```
None
python3 -m venv venv; . venv/bin/activate
```

Después ejecutamos el comando *python3 app.py* para lanzar la aplicación, una vez lanzada la aplicación, podemos conectarnos a la misma desde un buscador al escribir *http://10.6.130.17:8080*. Este Script de python tiene el siguiente código:

```python
Python
import psycopg2
from flask import Flask, jsonify, request

app = Flask(__name__)

# --- 1. CONFIGURACIÓN DE LA BASE DE DATOS ---
def get_db_connection():
    try:
        conn = psycopg2.connect(
            host='localhost',
            database='Logistica_Internacional',
            user='postgres',
            password='Eric1234'
        )
        return conn
    except Exception as e:
        print(f"Error conectando a la BD: {e}")
        return None

# --- RUTA DE BIENVENIDA ---
@app.route('/', methods=['GET'])
def home():
    return jsonify({'mensaje': 'API de Logística Internacional Operativa',
'estado': 'online'})

# --- RUTA 1: OBTENER CLIENTES (GET) ---
@app.route('/api/clientes', methods=['GET'])
def obtener_clientes():
    conn = get_db_connection()
    if conn is None:
        return jsonify({'error': 'Error de conexión a BD'}), 500

    cur = conn.cursor()
    try:
```

```
        cur.execute('SELECT id_cliente, nombre, email, telefono FROM
cliente;')
        filas = cur.fetchall()

        lista_clientes = []
        for fila in filas:
            lista_clientes.append({
                'id': fila[0],
                'nombre': fila[1],
                'email': fila[2],
                'telefono': fila[3]
            })
        return jsonify({'clientes': lista_clientes}), 200
    except Exception as e:
        return jsonify({'error': str(e)}), 400
    finally:
        cur.close()
        conn.close()

# --- RUTA 2: OBTENER ENVÍOS DE UN CLIENTE (GET) ---
@app.route('/api/envios', methods=['GET'])
def obtener_envios():
    id_cliente = request.args.get('id_cliente')

    if not id_cliente:
        return jsonify({'error': 'Falta el parámetro id_cliente'}), 400

    conn = get_db_connection()
    cur = conn.cursor()
    try:
        query = """
            SELECT id_envio, fecha_salida, estado, fecha_llegada
            FROM envio
            WHERE id_cliente = %s;
        """
        cur.execute(query, (id_cliente,))
        filas = cur.fetchall()

        lista_envios = []
        for fila in filas:
            lista_envios.append({
                'tracking_id': fila[0],
                'salida': str(fila[1]),
                'estado': fila[2],
                'llegada_estimada': str(fila[3]) if fila[3] else "Pendiente"
            })

        return jsonify({'envios': lista_envios}), 200
```

```
    except Exception as e:
        return jsonify({'error': str(e)}), 400
    finally:
        cur.close()
        conn.close()

# --- RUTA 3: CREAR UN NUEVO ENVÍO (POST) ---
@app.route('/api/envios/nuevo', methods=['POST'])
def crear_envio():
    datos = request.get_json()

    # Validamos que lleguen los datos
    if not datos or 'id_cliente' not in datos or 'fecha_salida' not in
datos:
        return jsonify({'error': 'Faltan datos obligatorios (id_cliente,
fecha_salida)'}), 400

    conn = get_db_connection()
    cur = conn.cursor()
    try:
        # Insertamos y devolvemos el ID generado
        query = """
            INSERT INTO envio (id_cliente, fecha_salida, estado)
            VALUES (%s, %s, 'Pendiente')
            RETURNING id_envio;
        """
        cur.execute(query, (datos['id_cliente'], datos['fecha_salida']))
        nuevo_id = cur.fetchone()[0]
        conn.commit() # Guardar cambios

        return jsonify({'mensaje': 'Envío creado con éxito', 'id_generado':
nuevo_id}), 201
    except Exception as e:
        conn.rollback()
        return jsonify({'error': str(e)}), 400
    finally:
        cur.close()
        conn.close()

# --- RUTA 4: ELIMINAR ENVÍO (DELETE) ---
@app.route('/api/envios/eliminar', methods=['DELETE'])
def eliminar_envio():
    id_envio = request.args.get('id_envio')

    if not id_envio:
        return jsonify({'error': 'Falta el parámetro id_envio'}), 400

    conn = get_db_connection()
```

```
    cur = conn.cursor()
    try:
        # Verificamos si existe primero
        cur.execute('SELECT id_envio FROM envio WHERE id_envio = %s',
(id_envio,))
        if not cur.fetchone():
            return jsonify({'error': 'El envío no existe'}), 404

        # Borramos
        cur.execute('DELETE FROM envio WHERE id_envio = %s', (id_envio,))
        conn.commit()
        return jsonify({'mensaje': f'Envío {id_envio} eliminado
correctamente'}), 200
    except Exception as e:
        conn.rollback()
        return jsonify({'error': str(e)}), 400
    finally:
        cur.close()
        conn.close()

# --- ARRANQUE DEL SERVIDOR ---
if __name__ == '__main__':
    print("--> API LISTA EN PUERTO 8080")
    app.run(host='0.0.0.0', port=8080, debug=True)
```

6.2 Definición de Endpoints (Puntos de Acceso)
La API expone recursos clave del negocio mediante verbos HTTP estándar, devolviendo siempre las respuestas en formato JSON. A continuación, se detallan los endpoints implementados:

- Consultar Clientes (GET)
  Este endpoint permite obtener el listado completo de la cartera de clientes.
  - Ruta: /api/clientes
  - Lógica: Ejecuta una consulta SELECT sobre la tabla cliente, transformando las tuplas resultantes en una lista de diccionarios JSON que incluye ID, nombre, email y teléfono.

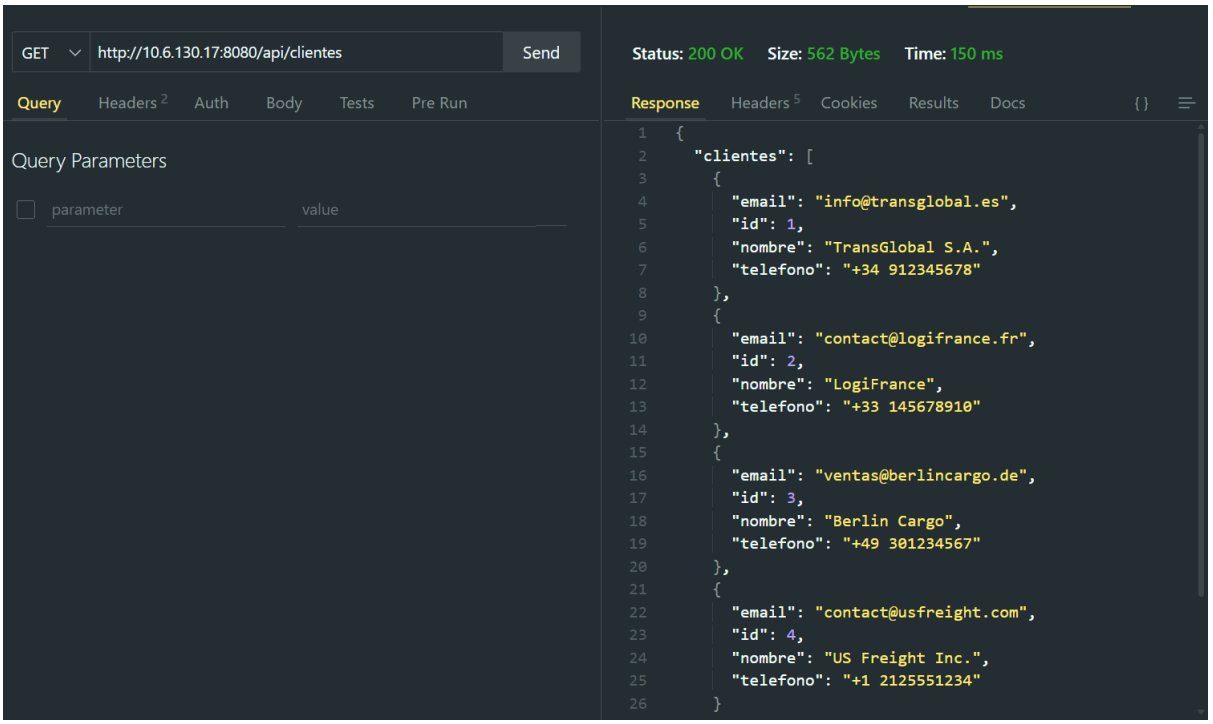
- Gestión de Envíos (GET con filtros)
  Permite a un usuario consultar el estado de sus envíos específicos.
  - Ruta: /api/envios
  - Parámetros: Acepta el parámetro id_cliente mediante query string (ej: ?id_cliente=1).
  - Lógica: Filtra la tabla envio por la clave foránea del cliente, devolviendo una lista con el ID de seguimiento (tracking_id), fecha de salida, estado actual y fecha estimada de llegada.

- Creación de Envíos (POST)
  Permite registrar un nuevo envío en el sistema.
  - Ruta: /api/envios/nuevo
  - Cuerpo (Body): Requiere un objeto JSON con id_cliente y fecha_salida.
  - Lógica:
    - Valida la recepción de los datos obligatorios.
    - Realiza una inserción (INSERT) en la tabla envio asignando el estado inicial "Pendiente" por defecto.
    - Confirma la transacción (commit) y devuelve el ID del nuevo envío generado por la base de datos (RETURNING id_envio).

6.3 Pruebas de Funcionalidad y Validación

Para verificar el correcto funcionamiento de la API, se han realizado pruebas exhaustivas utilizando Thunder Client. A continuación, se exponen las siguientes pruebas realizadas:

- **Caso de Uso 1: Obtención del listado de clientes**
  - Objetivo: Verificar la conectividad y la recuperación de datos maestros.
  - Petición: GET http://10.6.130.17:8080/api/clientes
  - Resultado: El servidor responde con código 200 OK y un array JSON conteniendo los clientes registrados.

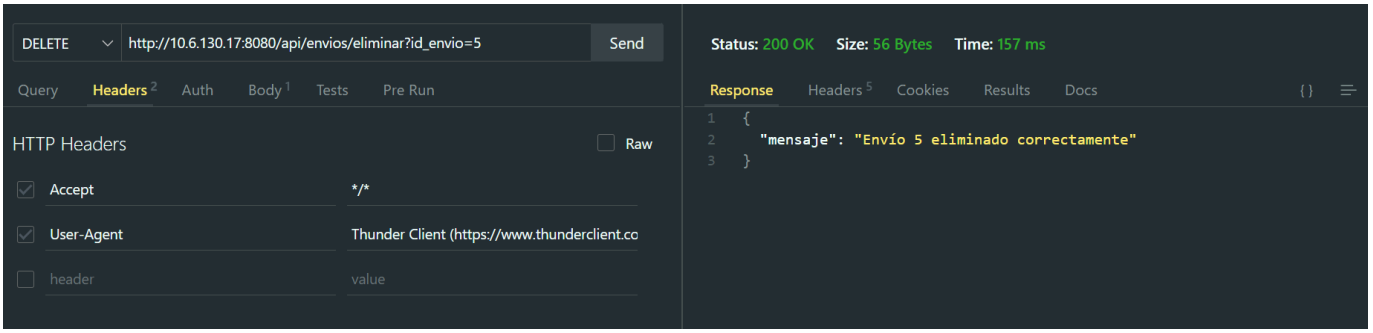


- **Caso de Uso 2: Creación de un nuevo envío**
  - Petición: POST http://10.6.130.17:8080/api/envios/nuevo
  - Body (JSON):

```
JSON
{
  "id_cliente": 1,
  "fecha_salida": "2024-01-15"
}
```

- Resultado: El servidor responde con código 201 Created (o 200 OK), confirmando la inserción y devolviendo el ID generado.

- **Caso de Uso 3: Eliminación de un envío**
  - Objetivo: Comprobar la eliminación de registros.
  - Petición: DELETE http://10.6.130.17:8080/api/envios/eliminar?id_envio=15
  - Resultado: El servidor responde con código 200 OK y un mensaje de confirmación indicando que el recurso ha sido eliminado correctamente.



7. Presupuesto.

El presente apartado detalla el presupuesto necesario para el diseño, desarrollo, implementación y puesta en marcha del sistema de gestión logística y de transporte internacional propuesto. El objetivo de este presupuesto es reflejar de manera transparente tanto los costes técnicos, recursos humanos, software, infraestructura, equipos físicos y servicios externos implicados en el proyecto. Al tratarse de una empresa de logística internacional que está iniciando su actividad, se consideran soluciones escalables pero económicamente contenidas. El presupuesto se divide en cinco bloques principales:

- Consultoría y análisis preliminar

- Desarrollo del sistema y base de datos
- Infraestructura y software
- Hardware y recursos materiales
- Mantenimiento anual

A continuación se detalla cada uno de estos elementos.

7.1. Consultoría y análisis preliminar.

Antes de iniciar cualquier desarrollo técnico es necesario realizar un análisis exhaustivo de los procesos operativos que la empresa llevará a cabo. Este análisis lo realizan profesionales especializados en logística, tecnología, aduanas y diseño de sistemas. Esta fase permite comprender los flujos de trabajo, las necesidades reales del negocio, las reglas de operación internacional y los requisitos que deberá cumplir la base de datos. Si esta fase no se realiza de manera adecuada, el sistema podría construirse con errores estructurales, funcionalidades incompletas o módulos que no responden a la actividad logística de la empresa.

Consultoría logística.
Esta actividad la ejecuta un consultor logístico con experiencia en transporte internacional multimodal. Su función es analizar el funcionamiento de los envíos, los modos de transporte utilizados (aéreo, marítimo y terrestre), las conexiones entre transportistas, la estructura de rutas y la trazabilidad operativa. Este análisis es necesario para definir qué entidades deben existir en la base de datos (como Envío, Contenedor, Transportista o Ruta), qué información debe registrar el sistema y cómo se relacionan entre sí. Coste: 1.800 €

Consultoría de almacén (WMS básico).
Este análisis lo realiza un consultor especializado en operaciones de almacén. Estudia el flujo físico de mercancías, desde su entrada y registro hasta la salida, pasando por movimientos internos, inventarios y reubicaciones. Gracias a este trabajo se definen correctamente las entidades relacionadas con Almacén y Movimiento de Almacén, así como las reglas que aseguran una operativa eficiente. Esto es esencial para empresas que manejan mercancía en tránsito o que realizan consolidación/desconsolidación. Coste: 1.000€

Consultoría de aduanas.
Realizada por un experto en normativa aduanera y comercio exterior. El consultor analiza los documentos requeridos, los códigos HS, los INCOTERMS, los procesos de despacho y las responsabilidades de cada parte en la operación internacional. Este análisis garantiza que el sistema pueda gestionar operaciones aduaneras reales y registrar toda la información necesaria (documentación, estados de despacho, aranceles, pruebas de entrega, registros de inspección, etc.). Coste: 1.800 €

Arquitectura del sistema.

Este trabajo lo lleva a cabo un arquitecto de software. Define la estructura general del sistema, los módulos que lo componen, los roles y permisos de usuarios, la seguridad, los flujos de datos y la arquitectura de la base de datos. Esta etapa es esencial porque establece los cimientos técnicos del desarrollo y asegura que todos los módulos encajen correctamente: TMS, WMS, aduanas, facturación y trazabilidad. Coste: 1.750 €

Análisis de interfaz (UX/UI).
Realizado por un diseñador UX/UI. Su objetivo es asegurar que la interfaz sea intuitiva y adecuada para usuarios internos y clientes externos. Analiza las tareas que realiza cada tipo de usuario y propone pantallas preliminares del TMS, WMS, módulo de aduanas y portal de clientes. Esto reduce errores operativos, facilita la adopción del sistema y permite una navegación clara. Coste: 1.200 €

Total consultoría y análisis preliminar: 7.550 €

7.2. Desarrollo del sistema y base de datos.

En este apartado se detallan todas las actividades relacionadas con la creación del software que gestionará las operaciones logísticas. Esta fase involucra a desarrolladores backend, frontend, especialistas en bases de datos y personal de QA (Quality Assurance). Incluye la programación de módulos, la implementación de la lógica de negocio, el diseño de interfaces, la creación de la base de datos relacional y las integraciones con proveedores externos. También se contemplan pruebas de funcionalidad y validación con usuarios.

Desarrollo del TMS (Transportation Management System)
Este módulo es desarrollado por un equipo compuesto por dos programadores backend, un programador frontend y un analista funcional. El TMS es el núcleo del sistema y cubre la gestión completa de un envío internacional: clientes, productos, contenedores, transportistas, rutas, tarifas, incidencias, trazabilidad y facturación básica. Cada funcionalidad se implementa siguiendo las reglas definidas durante la consultoría. El TMS permite que la empresa controle de manera centralizada el ciclo de vida de cada envío y asegura la integridad de los datos durante todo el proceso. Coste total TMS: 26.800 €

Desarrollo del WMS (Warehouse Management System)
El WMS es desarrollado por un programador backend, un diseñador de interfaz y un técnico especializado en operaciones de almacén. Su objetivo es registrar entradas, salidas, inventarios y movimientos internos, además de permitir una coordinación correcta entre almacén y transporte. Este módulo es imprescindible para asegurar la trazabilidad física de la carga, evitar pérdidas y mantener un inventario actualizado. Coste total WMS: 6.900 €

Módulo de aduanas.
Este módulo es desarrollado por un programador backend y un analista con conocimiento en comercio exterior. Permite gestionar la documentación aduanera, registrar códigos arancelarios, asociar documentos a envíos y controlar su estado de despacho. Es clave para empresas que realizan exportaciones e importaciones, ya que asegura cumplimiento normativo y reduce riesgos legales. Coste total aduanas: 3.850 €

Portal para clientes

Un programador frontend y un diseñador UI trabajan en este portal. Su función es ofrecer a los clientes acceso a información en tiempo real sobre sus envíos, documentos, incidencias y facturación. Este portal aumenta la transparencia y mejora la relación con el cliente, reduciendo consultas manuales y llamadas internas. Coste total portal: 3.900 €

Integraciones necesarias
Las integraciones son realizadas por un desarrollador especializado en APIs. Incluyen conexión con proveedores de GPS para tracking, generación de ficheros contables y carga automática de tarifas de transportistas. Estas integraciones permiten automatizar procesos internos y obtener información en tiempo real. Coste total integraciones: 3.050 €

Desarrollo de la base de datos
Esta tarea la realiza un administrador de bases de datos (DBA) y un analista de datos. Incluye el diseño lógico y físico de todas las entidades del sistema, sus relaciones, índices, triggers, vistas y todos los scripts necesarios para crear y mantener la estructura. También se consideran procesos de normalización, control de integridad y carga inicial de datos. La base de datos es el pilar central del sistema, y una mala definición comprometería todo el proyecto. Coste total base de datos: 13.100 €

Coste total desarrollo del sistema + BD: 57.600 €

7.3. Infraestructura y software.

Para garantizar que el sistema funcione de forma estable, segura y escalable, es necesario contar con una infraestructura en la nube adecuada a las necesidades de una empresa de logística internacional que está en fase inicial de crecimiento. Este apartado incluye los recursos tecnológicos imprescindibles para alojar el sistema, almacenar información, proteger los datos, asegurar la disponibilidad del servicio y optimizar el rendimiento. La configuración ha sido diseñada por un técnico de sistemas en la nube, que selecciona tecnologías fiables, con soporte profesional y con capacidad de ampliarse conforme aumente el volumen de envíos, usuarios y datos.

A continuación se detalla cada componente, junto con las tecnologías concretas necesarias, la función que cumplen y el motivo de su inclusión en el presupuesto.

Servidores en la nube – 1.320 € / año
Para el alojamiento del sistema se propone utilizar servicios cloud tipo AWS, Azure o Google Cloud, evitando infraestructura física local y garantizando disponibilidad 24/7. Se utilizan dos servidores principales:

Servidor de aplicación
Tecnología necesaria:

- Instancia virtual (por ejemplo, AWS EC2 t3.medium o equivalente).

- Sistema operativo Linux (Ubuntu Server 22.04 LTS).

- Servidor web Nginx para servir el frontend.
- Servidor de aplicaciones Node.js o Python FastAPI.

Función: aloja el backend, el frontend y las APIs del sistema.
Justificación: soporta el tráfico de usuarios, conexiones de transportistas, portal de clientes y procesos internos.

Servidor de base de datos
Tecnología necesaria:

- Motor PostgreSQL gestionado (AWS RDS, Azure PostgreSQL, Google Cloud SQL).
- Replicación automática y backups diarios.
- Cifrado de datos en reposo (AES-256).

Función: almacena la base de datos completa del sistema.
Justificación: PostgreSQL es robusto, altamente seguro, relacional y compatible con operaciones logísticas complejas.
Coste aproximado total servidores: 1.320 € / año

Almacenamiento en la nube – 576 € / año

Este almacenamiento es esencial para mantener copias de seguridad, documentos, facturas, certificados aduaneros, BL, AWB, registros de seguimiento, etc.
Tecnologías necesarias:

- AWS S3, Azure Blob Storage o Google Cloud Storage.
- Política de retención de 90 días para backups.
- Versionado de objetos para evitar pérdida de información.
- Cifrado gestionado por el proveedor (AES-256).

Función: guardar archivos externos relacionados con la operación logística y todas las copias de seguridad automáticas del sistema.
Justificación: se requiere almacenar documentos sensibles (aduanas, facturas, tarifas, listas de carga), por lo que se necesita un almacenamiento seguro con políticas de durabilidad del 99.999999999%.
Coste total almacenamiento: 576 € / año

Seguridad de la infraestructura – 500 € / año

Para garantizar un entorno seguro, el sistema incorpora los siguientes elementos tecnológicos:
Firewall administrado
Tecnología:

- AWS Security Groups / Azure Firewall / Google Cloud Firewall.
  Función: filtrar tráfico entrante y saliente, bloquear accesos no autorizados y limitar el acceso a la base de datos solo a servidores internos.

Certificados SSL (HTTPS)
Tecnología:
- Certificado TLS 1.2+ emitido por Cloudflare o Let's Encrypt.
  Función: cifrar todas las comunicaciones entre usuarios y servidor.

CDN básico
Tecnología:
- Cloudflare CDN / AWS CloudFront.
  Función: acelerar la carga de la web y proteger contra ataques DDoS.

Monitorización de infraestructura
Tecnología:
- AWS CloudWatch / Azure Monitor / Google Operations Suite.
  Función: vigilar rendimiento, caídas, tráfico irregular o intentos de ataque.

Justificación: en logística internacional se manipulan documentos confidenciales, datos de clientes y procesos críticos, por lo que es imprescindible garantizar disponibilidad, protección y cifrado completo del entorno.
Coste total seguridad: 500 € / año

Coste anual total infraestructura: 2.396 €
(1.320 € servidores + 576 € almacenamiento + 500 € seguridad)

7.4. Hardware y recursos materiales.

Este apartado recoge todos los recursos físicos necesarios para poner en funcionamiento el sistema en la empresa. Aunque la infraestructura principal se aloja en la nube, es imprescindible contar con equipamiento adecuado para el personal de administración, los operarios de almacén, los conductores, las áreas de facturación/aduanas y el departamento de TI. El conjunto de recursos está dimensionado para una empresa logística pequeña en fase inicial, con previsión de crecimiento moderado.

7.4.1 Equipos informáticos para personal administrativo – 4.200 €

Los equipos administrativos se utilizan para gestionar los envíos, facturación, documentación aduanera, creación de rutas, asignación de contenedores, y generación de reportes. Se requieren equipos fiables, con procesadores modernos y discos SSD para un rendimiento estable.
Equipos:
- 6 ordenadores de oficina (gestores logísticos, administración, atención al cliente)
  Características técnicas:
  - CPU Intel i5 / Ryzen 5

- 16 GB RAM
- SSD 512 GB
- Pantalla 22”-24”
- Windows 11 Pro o Ubuntu 22.04
  Coste unitario: ~700 €
  Total: 4.200 €

Se utilizarán para ejecutar el software de gestión, acceder al sistema en la nube, procesar documentación de clientes y aduanas, facturar servicios logísticos y trabajar con hojas de cálculo internas. Serán utilizados por personal administrativo, agentes de aduanas internos, responsables de operaciones y personal financiero.

6.4.2 Equipos para almacén y operativa logística – 3.400 €
Incluye dispositivos que permiten al personal del almacén registrar movimientos, controlar inventarios, actualizar trazabilidad y operar con el WMS integrado.
a) Terminales portátiles (PDA) para operarios

- 2 PDA industriales
  Características técnicas:
  - Android empresarial
  - Lector de códigos 1D/2D
  - Conectividad WiFi + 4G
  - Carcasa resistente IP65
    Coste unitario: ~750 €
    Total: 1.500 €

Para la lectura de códigos de contenedores, pallets, productos y registro de movimientos de almacén.
b) Impresoras de etiquetas

- 1 impresora térmica industrial para etiquetas logísticas
  Coste: ~600 €

Para la impresión de etiquetas con códigos de barras para contenedores, pallets, cajas y mercancía.
c) Routers y repetidores WiFi para almacén

- 1 router profesional + 2 repetidores WiFi industriales
  Coste total: ~500 €

Garantizar cobertura completa en zonas de carga/descarga para los terminales portátiles y tablets.
d) Tablets para supervisores de almacén

- 1 tablet resistente para supervisión de operaciones
  Coste: ~800 €

Revisión de inventarios, seguimiento de envíos, registro de incidencias en tiempo real.
Total equipos almacén: 3.400 €

6.4.3 Dispositivos para transportistas y personal en ruta – 1.900 €
Aunque muchos transportistas usan sus propios teléfonos, la empresa necesita equipar a algunos conductores para asegurar conectividad y trazabilidad.
a) Smartphones empresariales reforzados

- 3 dispositivos resistentes (rugged phones)
  Coste unitario: ~350 €
  Total: 1.050 €

Uso:

- Registrar eventos de tracking (TrackingEvent).
- Recibir asignaciones de ruta.
- Adjuntar fotos de cargas/incidencias.

b) Soportes, cargadores y accesorios para vehículos

- Accesorios varios para 3 vehículos
  Coste total: ~300 €*

Utilizados para soporte y seguridad de los dispositivos en camiones propios o arrendados.
c) GPS externos opcionales

- 2 receptores GPS dedicados
  Coste unitario: ~275 €
  Total: 550 €

Usados para mejorar la precisión en zonas de baja cobertura y sincronizar con el sistema de rutas.
Total transportistas: 1.900 €

6.4.4 Equipos de red y seguridad internos – 1.200 €
Aunque la infraestructura está en la nube, la empresa necesita una red interna segura.
a) Firewall físico básico

- 1 firewall UTM pequeño para oficina
  Coste: ~500 €
  Tecnologías: Fortinet FortiGate 40F, Sophos XG86, Cisco Meraki Go.

b) Switch Gigabit administrable

- 1 switch de 16 puertos
  Coste: ~250 €

c) Sistema de alimentación ininterrumpida (SAI)

- 1 SAI de 1500 VA para proteger servidores internos ligeros y router
  Coste: ~450 €

Usado para protección eléctrica, redundancia local mínima y seguridad en comunicaciones.
Total red y seguridad: 1.200 €

6.4.5 Impresoras y escáneres para documentación logística – 1.050 €
La documentación logística requiere impresión frecuente de:

- BL (Bill of Lading)
- AWB (Air Waybill)
- Documentos aduaneros (DUA)
- Facturas
- Packing lists
- Reportes internos

a) Impresora multifunción para oficina

- 1 impresora multifunción láser
  Coste: ~450 €

b) Escáner documental de alta velocidad

- 1 escáner profesional
  Coste: ~600 €

Total impresión y digitalización: 1.050 €

6.4.6 Mobiliario y cableado de red – 800 €
Incluye:

- Sillas ergonómicas para operarios administrativos
- Mesas para equipos
- Armario rack pequeño para switch, firewall y router
- Cableado Ethernet categoría 6
- Soportes para PDA y tablets

Total: 800 €

Coste total de Hardware y Recursos Materiales: 12.550 €

7.5. Mantenimiento anual.

Este bloque recoge los costes necesarios para garantizar el correcto funcionamiento del sistema una vez puesto en marcha. Incluye soporte técnico, actualizaciones, mantenimiento de servidores y monitorización general del entorno. Estas tareas son realizadas de forma combinada por un proveedor externo de software y un responsable interno de TI, lo que permite asegurar la continuidad operativa sin necesidad de contratar un departamento completo de informática.

7.5.1 Soporte técnico y resolución de incidencias – 3.000 €
Incluye la atención de problemas de uso del sistema, errores en pantallas, incidencias con usuarios, fallos en el TMS/WMS o en las integraciones.
El proveedor externo resuelve incidencias técnicas y el responsable interno atiende dudas operativas básicas. Es necesario porque la actividad logística es continua y cualquier fallo puede afectar a los envíos.

7.5.2 Actualizaciones funcionales menores – 1.800 €
Comprende pequeñas mejoras del sistema: ajustes de pantallas, ampliación de campos, optimización de procesos y mejoras en informes. Son desarrolladas por el equipo externo y permiten adaptar el sistema a las necesidades cambiantes de la empresa.

7.5.3 Actualizaciones legales y normativas – 1.200 €
Incluye la actualización de INCOTERMS, códigos arancelarios (HS Codes), requisitos documentales de aduanas y cambios normativos. Las realiza el consultor aduanero y el desarrollador. Es fundamental para operar correctamente en comercio internacional.

7.5.4 Mantenimiento preventivo de servidores – 1.500 €
Abarca revisiones periódicas de la infraestructura en la nube: actualizaciones de seguridad, comprobación de certificados SSL, limpieza de logs y verificación de copias de seguridad. Lo realiza un técnico de sistemas del proveedor externo.

7.5.5 Monitorización del sistema – 1.500 €
Consiste en supervisar el rendimiento del sistema, tiempos de respuesta, carga de la base de datos y estado del servidor. Permite detectar problemas antes de que afecten a la operativa.

Coste total mantenimiento anual: 9.000 €

Resumen del presupuesto.

Bloque / Concepto	Descripción	Responsable	Coste (€)

7.1 Consultoría y análisis preliminar			**7.550**
Consultoría logística	Estudio de flujos de envíos, modos de transporte, rutas y trazabilidad	Consultor logístico	1.800
Consultoría de almacén (WMS)	Análisis de entradas, salidas, inventarios y movimientos internos	Consultor de almacén	1.000
Consultoría de aduanas	Normativa aduanera, INCOTERMS, códigos HS, documentación	Consultor de aduanas	1.800
Arquitectura del sistema	Estructura del software, roles, permisos y base de datos	Arquitecto de software	1.750
Análisis de interfaz (UX/UI)	Diseño preliminar de pantallas TMS, WMS, aduanas y portal	Diseñador UX/UI	1.200
7.2 Desarrollo del sistema y base de datos			**57.600**
Desarrollo TMS	Gestión de clientes, envíos, contenedores, transportistas, rutas, tarifas, incidencias y facturación	Backend, frontend, analista funcional	26.800
Desarrollo WMS	Control de entradas/salidas, inventario y movimientos internos	Backend, diseñador UI, técnico de almacén	6.900
Módulo de aduanas	Gestión de documentación, códigos arancelarios y despacho	Backend, analista de comercio exterior	3.850

Portal para clientes	Acceso a envíos, incidencias, documentación y facturación	Frontend, diseñador UI	3.900
Integraciones necesarias	GPS tracking, exportación contable y carga de tarifas	Desarrollador API	3.050
Desarrollo base de datos	Diseño lógico/físico, normalización, relaciones, triggers, vistas y carga inicial	DBA y analista de datos	13.100
7.3 Infraestructura y software			**2.396**
Servidores en la nube	Backend, frontend y base de datos PostgreSQL con backups y cifrado	Técnico cloud	1.320
Almacenamiento en la nube	Copias de seguridad, documentos y archivos de operación logística	Técnico cloud	576
Seguridad	Firewall, certificados SSL, CDN y monitorización	Técnico cloud	500
7.4 Hardware y recursos materiales			**12.550**
Equipos administrativos	Ordenadores para gestión, facturación y documentación	Personal administrativo	4.200
Equipos de almacén	PDA, impresora de etiquetas, tablets, routers	Operarios de almacén	3.400
Equipos para transportistas	Smartphones, GPS, accesorios	Conductores/transportistas	1.900

Red y seguridad interna	Firewall físico, switch y SAI	Responsable TI	1.200
Impresión y escaneo	Impresora multifunción y escáner	Oficina / administración	1.050
Mobiliario y cableado	Mesas, sillas, armario rack y soportes	Oficina y almacén	800
7.5 Mantenimiento anual			**9.000**
Soporte técnico e incidencias	Resolución de problemas del sistema	Proveedor externo + responsable interno	3.000
Actualizaciones funcionales menores	Mejoras y ajustes en TMS, WMS, portal y base de datos	Proveedor externo	1.800
Actualizaciones legales y normativas	INCOTERMS, HS Codes, documentación aduanera	Consultor aduanas + desarrollador	1.200
Mantenimiento preventivo servidores	Parches de seguridad, certificados, backups	Técnico cloud	1.500
Monitorización del sistema	Rendimiento de servidores, base de datos y APIs	Técnico cloud + DBA	1.500
TOTAL PROYECTO	**Suma de todos los bloques anteriores**	**-**	**72.000**
Mantenimiento anual	**Costo recurrente para asegurar operatividad**	**-**	**9.000**