Manual Técnico

Proyecto: Sistema de Transporte de Pasajeros y Encomiendas **Autor:** Jeferson Stiv González Barrios **Fecha:** Octubre 2025 **Versión:** 1.0

1. Introducción

Este manual técnico proporciona una descripción detallada de la arquitectura de la base de datos, los componentes de software y las directrices para la instalación, configuración y mantenimiento del **Sistema de Transporte de Pasajeros y Encomiendas**. Está dirigido a desarrolladores, administradores de sistemas y personal técnico encargado de la gestión del sistema.

1.1. Alcance del Sistema

El sistema está diseñado para gestionar de forma integral las operaciones de una empresa de transporte, cubriendo las siguientes áreas funcionales:

- **Gestión de Flota:** Administración de buses y tipos de buses.
- **Gestión de Personal:** Registro y control de pilotos.
- **Gestión de Rutas:** Definición de rutas, destinos y escalas.
- **Planificación de Viajes:** Programación de viajes asignando recursos.
- **Reservaciones y Ventas:** Proceso de reservación y venta de boletos en línea.
- **Gestión de Encomiendas:** Registro y seguimiento de paquetes.
- **Procesamiento de Pagos:** Integración de diferentes métodos de pago y su verificación.
- **Gestión de Clientes:** Administración de la información de los clientes.
- **Seguridad:** Control de acceso basado en roles y permisos.
- **Auditoría y Reportes:** Generación de bitácoras y reportes gerenciales.

1.2. Arquitectura Tecnológica

- **Base de Datos:** Oracle Database.
- **Lenguaje de Base de Datos:** SQL, PL/SQL.
- **Modelado de Datos:** PowerDesigner (compatible con el DDL proporcionado).
- **Desarrollo de Aplicación (sugerido):**
  - **Backend:** Java (Spring Boot), Python (Django/Flask) o Node.js (Express).
  - **Frontend:** HTML, CSS, JavaScript (React, Angular o Vue.js).
  - **Móvil:** React Native, Flutter o nativo (Android/iOS).
- **Control de Versiones:** Git.

---

2. Estructura de la Base de Datos

El diseño de la base de datos es el núcleo del sistema. Se ha desarrollado siguiendo un modelo relacional normalizado para garantizar la integridad, consistencia y eficiencia de los datos.

2.1. Diagrama Entidad-Relación (E-R)

El siguiente diagrama muestra las principales entidades del sistema y cómo se relacionan entre sí. Fue generado utilizando Mermaid y es compatible con herramientas de modelado como PowerDesigner.

Diagrama Entidad-Relación

Diagrama Entidad-Relación

2.2. Descripción de Módulos de la Base de Datos

El esquema se organiza en los siguientes módulos lógicos:

- **Módulo de Catálogos:** Contiene tablas maestras con datos que rara vez cambian, como `DEPARTAMENTO`, `MUNICIPIO`, `TIPO_BUS` y `TIPO_PAGO`.
- **Módulo de Flota y Personal:** Administra los recursos físicos y humanos con las tablas `BUS` y `PILOTO`.
- **Módulo de Operaciones de Transporte:** Define la logística de los viajes a través de las tablas `RUTA`, `ESCALA` y `VIAJE`.
- **Módulo de Clientes:** Centraliza la información de los usuarios del servicio en la tabla `CLIENTE`.
- **Módulo Transaccional:** Gestiona las operaciones de negocio principales: `RESERVACION`, `BOLETO`, `ENCOMIENDA` y `PAGO`.
- **Módulo de Seguridad:** Controla el acceso y la auditoría con las tablas `USUARIO`, `ROL`, `PERMISO`, `ROL_PERMISO` y `BITACORA`.
- **Módulo de Configuración:** Almacena parámetros globales en la tabla `PARAMETRO` para flexibilizar el comportamiento del sistema.

---

3. Instalación y Configuración

Para implementar la base de datos en un entorno Oracle, siga los siguientes pasos en orden. Se recomienda utilizar un cliente de base de datos como SQL*Plus, Oracle SQL Developer o DBeaver.

3.1. Prerrequisitos

- Acceso a una instancia de Oracle Database (11g o superior).
- Un usuario de base de datos con permisos para crear tablas, secuencias, triggers, vistas, procedimientos y roles.

3.2. Orden de Ejecución de Scripts

Los scripts SQL deben ejecutarse en el siguiente orden para respetar las dependencias entre objetos:

1. **`01_crear_tablas.sql`**: Crea todas las tablas, secuencias y constraints (PK, FK, CHECK, UNIQUE).
2. **`02_datos_iniciales.sql`**: Inserta los datos maestros en las tablas de catálogos (departamentos, roles, permisos, parámetros, etc.) y crea el usuario administrador inicial.
3. **`03_triggers.sql`**: Crea los triggers para auditoría, validaciones de negocio y lógica automática (cálculo de tarifas, actualización de disponibilidad, etc.).
4. **`04_funciones.sql`**: Crea las funciones PL/SQL utilizadas para cálculos y validaciones reutilizables.
5. **`05_procedimientos.sql`**: Crea los procedimientos almacenados que encapsulan la lógica de negocio principal (crear reservación, registrar pago, etc.).
6. **`06_vistas.sql`**: Crea las vistas para simplificar las consultas complejas y para la generación de reportes.
7. **`07_paquetes.sql`**: Crea los paquetes PL/SQL que agrupan procedimientos y funciones relacionados por módulo.
8. **`08_seguridad_roles.sql`**: Crea los roles de base de datos y asigna los permisos necesarios para cada uno. También incluye la creación de usuarios de ejemplo.

3.3. Verificación

Después de ejecutar todos los scripts, verifique que no se hayan producido errores. Realice una consulta simple a una de las vistas para confirmar que la instalación fue exitosa:

```
SELECT * FROM v_viajes_disponibles;
```

---

4. Componentes de la Base de Datos

A continuación se describe cada tipo de objeto de la base de datos y su propósito en el sistema.

4.1. Tablas

Las tablas son la estructura fundamental de almacenamiento. Para una descripción detallada de cada tabla y sus columnas, consulte el **Diccionario de Datos**.

4.2. Secuencias

Se utiliza una secuencia por cada tabla para generar claves primarias autoincrementales, asegurando la unicidad de los identificadores. Ejemplo: `seq_cliente` para la tabla `CLIENTE`.

4.3. Triggers

Los triggers se utilizan para automatizar procesos y aplicar reglas de negocio a nivel de base de datos. Los más importantes son:

- **Triggers de Auditoría (`trg_audit_*`)**: Registran cambios (INSERT, UPDATE, DELETE) en tablas críticas como `CLIENTE`, `RESERVACION`, `PAGO` y `VIAJE` en la tabla `BITACORA`.
- **`trg_actualizar_lugares_viaje`**: Disminuye o aumenta el contador de `lugares_disponibles` en la tabla `VIAJE` cuando se emite o cancela un `BOLETO`.
- **`trg_calcular_tarifa_encom`**: Calcula automáticamente la tarifa de una encomienda basándose en su peso y los parámetros del sistema.
- **`trg_confirmar_reserva_pago`**: Cambia el estado de una `RESERVACION` a 'CONFIRMADA' cuando se actualiza un `PAGO` a 'VERIFICADO'.
- **`trg_encriptar_password`**: Encripta automáticamente la contraseña de un `USUARIO` usando SHA-512 antes de guardarla.
- **`trg_bloquear_usuario`**: Bloquea a un `USUARIO` después de un número configurable de intentos de inicio de sesión fallidos.

4.4. Funciones

Las funciones encapsulan cálculos y lógica que pueden ser reutilizados en consultas y otros bloques PL/SQL.

- **`fn_calcular_tarifa_encomienda`**: Retorna la tarifa correcta para una encomienda según su peso.
- **`fn_verificar_disponibilidad`**: Comprueba si hay suficientes lugares en un viaje para una cantidad de pasajeros.
- **`fn_validar_credenciales`**: Valida el usuario y la contraseña, gestionando el bloqueo de cuentas y el registro de intentos fallidos.
- **`fn_porcentaje_ocupacion`**: Calcula el porcentaje de ocupación de un viaje.

4.5. Procedimientos Almacenados

Los procedimientos almacenados contienen la lógica de negocio principal para las operaciones complejas.

- **`sp_crear_reservacion`**: Orquesta el proceso de creación de una nueva reservación, validando la disponibilidad.
- **`sp_registrar_pago`**: Maneja el registro de un pago y confirma la reservación si no requiere verificación.
- **`sp_verificar_pago`**: Proceso para que un operador apruebe o rechace un pago por depósito.
- **`sp_emitir_boleto`**: Genera un boleto para un pasajero si la reservación está confirmada.
- **`sp_expirar_reservaciones`**: Procedimiento de mantenimiento que se puede ejecutar periódicamente para cambiar el estado de las reservaciones no pagadas a 'EXPIRADA'.

4.6. Vistas

Las vistas proporcionan una capa de abstracción sobre las tablas, simplificando las consultas para la aplicación y los reportes.

- **`v_viajes_disponibles`**: Muestra toda la información relevante de los viajes programados y disponibles para la venta.
- **`v_reservaciones_completas`**: Une la información de reservaciones, clientes y viajes para una consulta completa.
- **`v_pagos_pendientes`**: Facilita a los operadores la visualización de los pagos que requieren verificación manual.
- **Vistas de Reportes (`v_ingresos_por_viaje`, `v_top_clientes`, `v_rutas_mas_utilizadas`, etc.)**: Pre-calculan y agregan datos para la generación de estadísticas y reportes gerenciales.

4.7. Paquetes

Los paquetes agrupan procedimientos y funciones relacionados por módulo, mejorando la organización y el mantenimiento del código PL/SQL.

- **`pkg_reservaciones`**: Contiene toda la lógica de negocio para gestionar reservaciones (crear, confirmar, cancelar).
- **`pkg_pagos`**: Encapsula los procedimientos para registrar y verificar pagos.
- **`pkg_reportes`**: Proporciona una API para generar los principales reportes del sistema a través de cursores de referencia (REF CURSOR).

4.8. Roles y Seguridad

La seguridad se gestiona a dos niveles: roles de aplicación (tabla `ROL`) y roles de base de datos (`rol_db_*`). Los roles de base de datos definen los permisos DML (SELECT, INSERT, UPDATE, DELETE) sobre los objetos de la base de datos, proporcionando una capa de seguridad fundamental. El script `08_seguridad_roles.sql` detalla la configuración de estos roles y la creación de usuarios de ejemplo.
