PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 3
6ta práctica (tipo b)
(Segundo Semestre 2025)

Indicaciones Generales:

- Duración 1h 50 minutos (Inicio: 8:10 a.m. - Fin: 10:00 a.m.) (Subida y verificación de archivos en PAIDEIA: 10:00 a.m. a 10:05 a.m.) (Salida del Laboratorio: 10:05 a.m. a 10:10 a.m.)
- Se les recuerda que, de acuerdo con el reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otro estudiante o cometer plagio para el desarrollo de esta práctica.
- Para el desarrollo de toda la práctica debe utilizar el sistema operativo Windows con JDK 24 y el .NET Framework 4.8.1.
- Para el desarrollo de las preguntas debe usar NetBeans y Microsoft Visual Studio .NET.
- Está permitido el uso de Internet (únicamente para consultar páginas oficiales de Oracle y Microsoft). No obstante, está prohibida toda forma de comunicación con otros estudiantes o terceros.
- PUEDE UTILIZAR MATERIAL DE CONSULTA. Antes de comenzar el laboratorio, descargue todos los proyectos, apuntes, diapositivas que utilizará.
- Se considerará en la calificación el uso de buenas prácticas de programación (aquellas vistas en clase).

Puntaje total: 20 puntos

Cuestionario

- La finalidad principal de este laboratorio es la de reforzar los conceptos contenidos en las clases 06, 07, 08, 09, 10 y 11: Conexión a Bases de Datos.
- Al finalizar la práctica, comprima la carpeta dada en las indicaciones iniciales empleando el programa Zip que viene por defecto en el Windows, no se aceptarán los trabajos compactados con otros programas como RAR, WinRAR, 7zip o similares.

Descripción del caso

La autoridad de transporte metropolitana desea implementar un sistema piloto de control vial llamado **TransitSoft** que consta de dos partes. La primera parte corresponde a un conjunto de cámaras inteligentes instaladas en tres avenidas principales de la ciudad. Dichas cámaras realizan capturas automáticas del tránsito vehicular, registrando información relevante como:

- Placa del vehículo.
- Velocidad detectada.
- Fecha y hora exacta de la captura.
- Información de la Cámara que registró la captura

El objetivo principal es identificar aquellos vehículos que circulen a una velocidad mayor al límite permitido en las avenidas, que en este caso es de 50 km/h. Una vez detectada la infracción, los datos deben ser transferidos al sistema central de gestión vehicular, ubicado en las oficinas de la autoridad de tránsito, para

su procesamiento y registro formal.

En otras palabras, se requiere que el sistema piloto no solo actúe como un mecanismo de monitoreo, sino también como una herramienta que alimente automáticamente la base de datos de infracciones para posteriores sanciones y mejor gestión del control vial.

En resumen, la autoridad de transporte requiere un sistema que permita:

1. Capturar automáticamente la información del tránsito vehicular en las avenidas seleccionadas, asegurando precisión en la lectura de placas y en la medición de la velocidad.
2. Comparar en tiempo real la velocidad registrada frente al límite establecido (50 km/h) y determinar si existe una infracción.
3. Generar un registro estructurado de cada infracción que incluya: placa del vehículo, velocidad detectada, límite permitido, fecha, hora y ubicación de la cámara, así como información del vehículo que cometió la infracción y de su propietario.
4. Transmitir los registros de infracción hacia el sistema central de gestión vehicular de la autoridad.
5. Registrar la infracción cometida en el sistema de gestión vial centralizado.

Arquitectura de la solución

El sistema está conformado por dos componentes principales, desarrollados en Java y .NET, que trabajan de manera integrada para garantizar la correcta extracción, transformación y carga (ETL) de las infracciones de tránsito.

- **TransitSoft Control Vial (Java)**: este módulo accede a la base de datos MySQL de TransitSoft, donde se almacenan las capturas generadas por las cámaras de tránsito. Los datos registrados en la base de datos incluyen no solo la placa, velocidad, fecha y hora, sino también datos relacionados con la cámara de seguridad que realizó la captura. Además, la base de datos contiene información de vehículos y propietario, lo que permite contextualizar las infracciones. El sistema transforma estos datos mediante el patrón DTO (Data Transfer Object) y envía las infracciones detectadas a una Cola de Procesamiento de Infracciones, que asegura un flujo asincrónico, confiable y desacoplado.
- **TransitSoft Gestión Vial (.NET)**: este componente consume la información de la cola y registra las infracciones en una tabla **desnormalizada** con los datos de capturas, cámaras, vehículos y propietarios, en en la base de datos SQL Server de TransitSoft. Allí quedan disponibles para los procesos de gestión administrativa, tales como la emisión de sanciones.

La arquitectura descrita se muestra en la Figura 1.

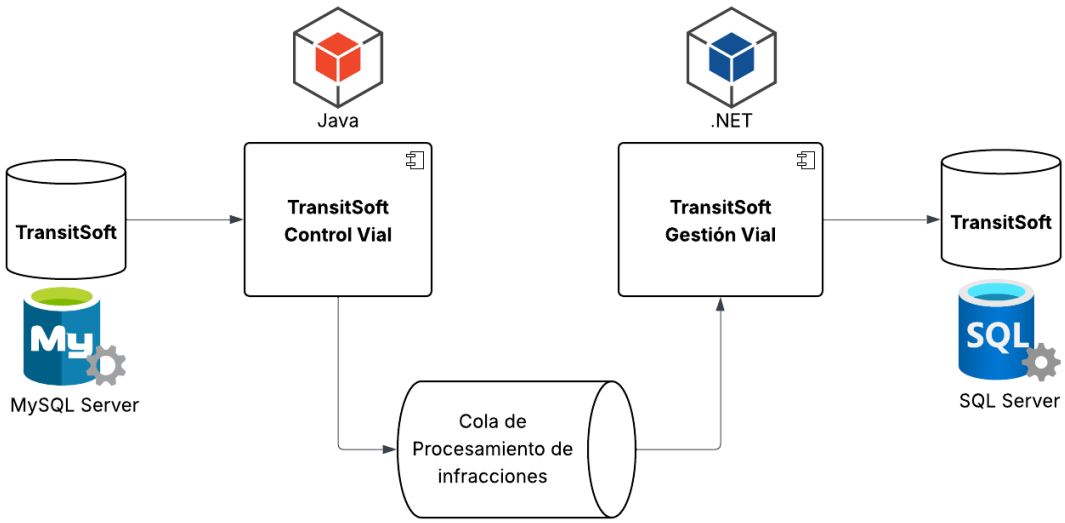


Figura 1 - Arquitectura de la Solución

Base de datos TransitSoft

A continuación, se muestra el modelo de base de datos del sistema de control y gestión vial transitsoft.

- Se registran los vehículos (placa, marca, modelo, año) y sus propietarios, vinculados mediante la tabla intermedia vehiculo_propietario.
- Las cámaras de seguridad (modelo, serie, ubicación) generan capturas con información de placa, velocidad, fecha y estado.
- A partir de estas capturas se generan las infracciones, donde se consolida la información del vehículo, propietario y cámara, junto con los datos de velocidad, exceso, ubicación, monto y fechas de captura/registro.

El diagrama ER se muestra en la Figura 2.

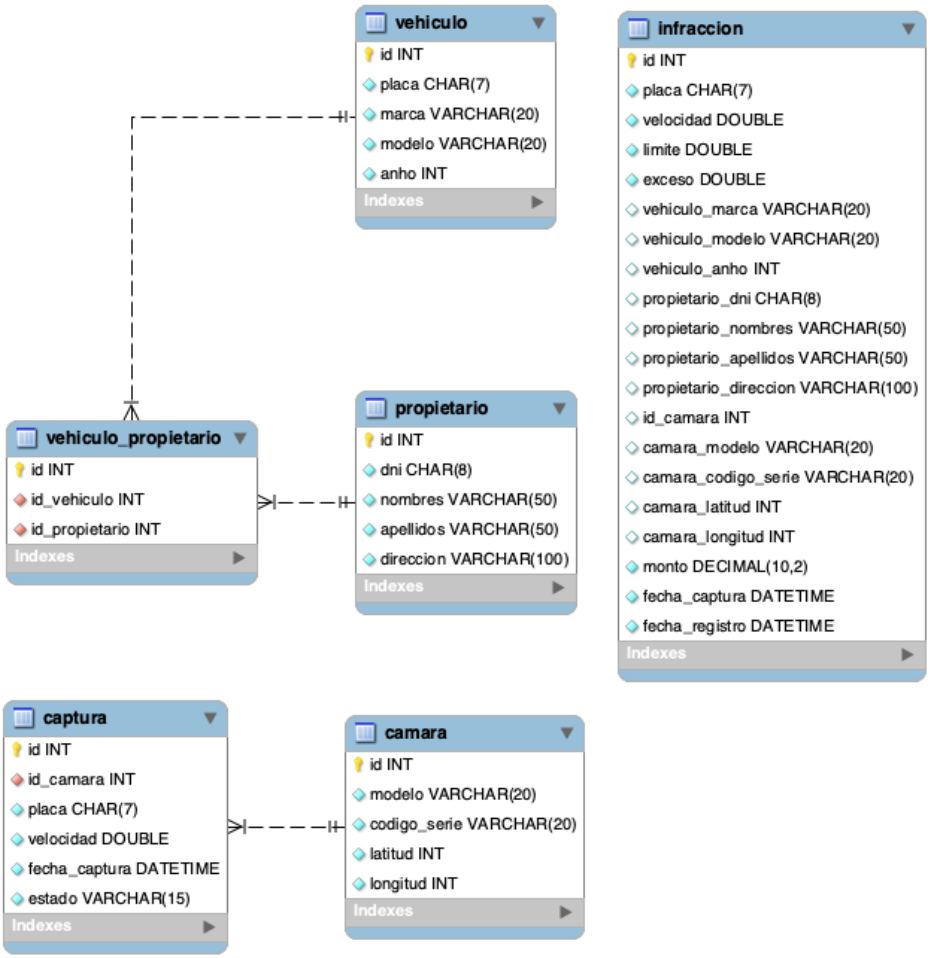
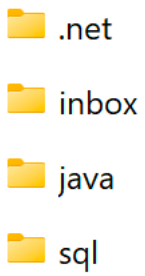


Figura 2 - Diagrama ER de TransitSoft

- Se proporcionará un script SQL para la creación de la base de datos en MySQL y en SQL Server, garantizando compatibilidad y consistencia del modelo en ambos motores de bases de datos.
- Se proporcionará un script SQL para la inicialización (creación de registros) de la base de datos en MySQL.
- Se proporcionará un script SQL con los procedimientos almacenados necesarios para el laboratorio. Usted puede crear procedimientos o sentencias SQL adicionales según crea conveniente.

Estructura de la Solución

Se le otorgará un archivo .zip llamado **Lab06-Estudiantes.zip** con la siguiente estructura.



1. java: contiene el proyecto de solución Java **(Maven)** para el componente Monitoreo Vehicular TransitSoft
    a. TransitSoftDBManager
    b. TransitSoftModelo
    c. TransitSoftPersistencia
    d. TransitSoftNegocio
    e. TransitSoftVigilanciaVial
2. .net: contiene el proyecto de solución .NET para el componente Gestión Vehicular TransitSoft
    a. TransitSoftDBManager
    b. TransitSoftModelo
    c. TransitSoftPersistencia
    d. TransitSoftNegocio
    e. TransitSoftVGestionVial
3. sql: contiene los scripts SQL para la creación de tablas, creación de registros y procedimientos almacenados.
4. inbox: Directorio usado como **Cola de Procesamiento de Infracciones**

Pregunta 01 (14 puntos):

Abra el proyecto Java (Maven) ubicado en \java\TransitSoft, este proyecto cuenta con los módulos necesarios para el correcto desarrollo de la pregunta. Al finalizar la pregunta el programa debe listar capturas donde se excede el límite de velocidad, generar infracciones, actualizarlas como PROCESADO y serializar las infracciones en formato JSON en la cola de procesamiento de infracciones.

Parte 01: Configuración de la base de datos y gestor de conexiones– 4 puntos

1. Cree un esquema llamado transitsoft.
2. Ejecute los scripts ubicados en \sql\MySQL en las bases de datos TransitSoft de MySQL.
3. En el módulo **TransitSoftDBManager** implementar un DBManager que le permita conectarse a una base de datos MySQL.
4. Configure en Maven la dependencia del Driver MySQL.
5. Debe seguir estrictamente los conceptos y patrones revisados en clase.

*Recomendación: Ubique las clases en el paquete **pe.edu.pucp.transitsoft.db***

Parte 02: Persistencia (DAO) – 6 puntos

1. Implementación del patrón DAO para la entidad *Captura*
    a. Implementar un método que permita **listar todas las capturas** disponibles en la base de datos.
    b. Implementar un método que permita **actualizar el estado** de una captura, considerando los valores posibles: *REGISTRADO* y *PROCESADO*.

   c. Se provee una definición inicial de CapturaDAO y su implementación. **Se solicita que el código implementado debe ser reutilizable y refactorizado.**

2. Utilizar las **entidades de dominio** provistas en el módulo **TransitSoftModelo**. La entidad de dominio *Captura* ya incluye referencias a los objetos de tipo **Camara** y **Vehiculo**. Asimismo, la entidad *Vehiculo* incluye referencia al objeto **Propietario**.
3. En el archivo \sql\MySQL\procedimientos.sql se encuentran los procedimientos almacenados **listarCapturas** y **modificarEstadoCaptura**.
   - Aunque se permite el uso directo de sentencias SQL, se **recomienda utilizar estos procedimientos almacenados**.
   - El procedimiento **listarCapturas** ya retorna la información completa de las capturas, incluyendo los datos de la **cámara**, el **vehículo** y su **propietario**.

*Recomendación: Ubique las interfaces en el paquete **pe.edu.pucp.transitsoft.dao** y las implementaciones en **pe.edu.pucp.transitsoft.daoimpl.***

Parte 03: Negocio (BO) y App – 4 puntos

El módulo **TransitSoftVigilanciaVial** contiene el método **main**, el cual ya hace uso de las interfaces y clases de negocio **CapturaBO** e **InfraccionBO**, junto con sus respectivas implementaciones. Estas se encargan de obtener las capturas que superan el límite de velocidad y de generar las infracciones correspondientes.

Adicionalmente, se incluye el método **serializarInfracciones**, responsable de serializar las infracciones creadas en la cola de procesamiento utilizando el formato JSON, lo que facilita su integración con otros componentes del sistema.

Con esta información se solicita:

1. Crear el archivo **db.properties** con sus datos de conexión a su motor de base de datos dentro del proyecto en:
   \java\TransitSoft\TransitSoftVigilanciaVial\src\main\resources
2. En el archivo app.properties, ubicado en
   \java\TransitSoft\TransitSoftVigilanciaVial\src\main\resources

   coloque la dirección del directorio **inbox.** Ejemplo.

   inbox.path=E:\\evaluaciones\\2025-2\\Lab06\\inbox

   Nota: recuerde que en el archivo de propiedades el carácter \ debe duplicarse (\\).

3. Analice el código provisto en InfraccionBOImpl, en particular analice el uso de **CapturaToInfraccionMapper**.

   Este mapper implementa el patrón DTO (Data Transfer Object) y se encarga de transformar la entidad de dominio **Captura** en el objeto de transferencia de datos **Infraccion.**

4. Implemente el método **obtenerCapturasConExcesoDeVelocidad** en **CapturaBOImpl**.
   - El método debe devolver únicamente las capturas cuyas velocidades excedieron el límite de velocidad de 50 KM/h.
   - Para ello, debe utilizar el **DAO de Captura**.
5. Implemente el método **actualizar** en **CapturaBOImpl**.

- Este método debe actualizar el estado de una captura a **PROCESADO**, invocando al **DAO de Captura**.

6. En el método **main,** invoque al método **actualizar** de **CapturaBO** por cada captura que supere el límite de velocidad.

Pregunta 02 (6 puntos):

Abra el proyecto .NET ubicado en \.net\TransitSoft, este proyecto cuenta con los módulos necesarios para el correcto desarrollo de la pregunta. Al finalizar la pregunta el programa debe registrar las infracciones disponibles en la **cola de procesamiento de infracciones** en la base de datos TransitSoft en Microsoft SQL Server e imprimir las infracciones procesadas en pantalla.

Parte 01: Configuración de la base de datos – 1 punto

1. Ejecutar los scripts ubicados en \sql\MSSQL sobre la base de datos **TransitSoft** en Microsoft SQL Server.

2. Analizar la implementación existente del módulo **TransitSoftDBManager**, en particular la clase DBFactoryProvider, identificando su rol dentro del patrón de gestión de conexiones.

Parte 02: Registrar y Leer Infracciones – 4 puntos

1. Implementación del patrón DAO para la entidad *Infraccion*
   a. Implementar un método que permita **crear** la infracción en la base de datos.

```
protected override DbCommand ComandoCrear(DbConnection conn, Infraccion modelo) {
    DbCommand cmd = conn.CreateCommand();
    cmd.CommandText = "insertarInfraccion";
    cmd.CommandType = CommandType.StoredProcedure;

    // Parámetros de la captura
    this.AgregarParametroEntrada(cmd, "@p_placa", DbType.AnsiStringFixedLength,
modelo.Placa);
    this.AgregarParametroEntrada(cmd, "@p_velocidad", DbType.Double, modelo.Velocidad);
    this.AgregarParametroEntrada(cmd, "@p_limite", DbType.Double, modelo.Limite);
    this.AgregarParametroEntrada(cmd, "@p_exceso", DbType.Double, modelo.Exceso);

    // Parámetros del vehículo
    this.AgregarParametroEntrada(cmd, "@p_vehiculo_marca", DbType.String,
modelo.MarcaVehiculo);
    this.AgregarParametroEntrada(cmd, "@p_vehiculo_modelo", DbType.String,
modelo.ModeloVehiculo);
    this.AgregarParametroEntrada(cmd, "@p_vehiculo_anho", DbType.Int32,
modelo.AnhoVehiculo);

    // Parámetros del propietario
    this.AgregarParametroEntrada(cmd, "@p_propietario_dni",
DbType.AnsiStringFixedLength, modelo.DniPropietario);
    this.AgregarParametroEntrada(cmd, "@p_propietario_nombres", DbType.String,
modelo.NombresPropietario);
    this.AgregarParametroEntrada(cmd, "@p_propietario_apellidos", DbType.String,
modelo.ApellidosPropietario);
    this.AgregarParametroEntrada(cmd, "@p_propietario_direccion", DbType.String,
modelo.DireccionPropietario);

    // Parámetros de la cámara
    this.AgregarParametroEntrada(cmd, "@p_camara_modelo", DbType.String,
modelo.ModeloCamara);
    this.AgregarParametroEntrada(cmd, "@p_camara_codigo_serie", DbType.String,
modelo.CodigoSerieCamara);
    this.AgregarParametroEntrada(cmd, "@p_camara_latitud", DbType.Int32,
modelo.Latitud);
    this.AgregarParametroEntrada(cmd, "@p_camara_longitud", DbType.Int32,
modelo.Longitud);
```

```
        // Parámetros de la infracción
        this.AgregarParametroEntrada(cmd, "@p_monto", DbType.Decimal, modelo.Monto);
        this.AgregarParametroEntrada(cmd, "@p_fecha_captura", DbType.DateTime,
    modelo.FechaCaptura);
        this.AgregarParametroEntrada(cmd, "@p_fecha_registro", DbType.DateTime,
    modelo.FechaRegistro);

        // Parámetro de salida
        this.AgregarParametroSalida(cmd, "@p_id", DbType.Int32);

        return cmd;
    }
```

b. Implementar un método que permita leer todas las infracciones en la base de datos.

```
    protected override DbCommand ComandoLeerTodos(DbConnection conn) {
        DbCommand cmd = conn.CreateCommand();
        cmd.CommandText = "listarInfracciones";
        cmd.CommandType = CommandType.StoredProcedure;
        return cmd;
    }
```

c. Se provee una definición inicial de InfraccionDAO y su implementación. **Se solicita que el código implementado debe ser reutilizable y refactorizado.**

2. Utilizar las **entidades de dominio** provistas en el módulo **TransitSoftModelo**. Analice la entidad de dominio *Infracción* la cual se encuentra completamente definida.

Parte 03: Procesar Infracciones – 1 puntos

1. En el archivo App.config ubicado en el proyecto TransitSoftGestionVial ingrese los datos de conexión de su base de datos Microsoft SQL Server (**db.host, db.usuario, db.password**)
2. En el archivo App.config ubicado en el proyecto TransitSoftGestionVial ingrese la ruta del directorio de la cola de procesamiento de infracciones (**inbox.path**). Ejemplo.

```
<appSettings>
      <add key="inbox.path" value="E:\evaluaciones\2025-2\Lab04\inbox" />
      <!-- Configuracion para SQL Server -->
      <add key="db.provider" value="System.Data.SqlClient" />
      <add key="db.host" value="[aquí su host]" />
      <add key="db.puerto" value="1433" />
      <add key="db.esquema" value="transitsoft" />
      <add key="db.usuario" value="admin" />
      <add key="db.password" value="[aquí su contraseña]" />
</appSettings>
```

Nota: recuerde que en el archivo de configuración el carácter \ en inbox.path, no se duplica.

Profesores del curso: Freddy Paz, Andrés Melgar, Eric Huiza

Pando, 24 de septiembre de 2025