**UNIVERSIDAD REGIONAL AUTÓNOMA DE LOS ANDES “UNIANDES”**



**FACULTAD DE SISTEMAS MERCANTILES**

**CARRERA DE SOFTWARE ADMINISTRACIÓN DE BASE DE DATOS**

**TEMA:** Sistema de Control de Buses

**AUTORES:** Diego Joel Cabrera Jácome

Juan Carlos Cevallos Franklin Barros

**TUTOR:** Ing. Rita Diaz

**AMBATO –**

**ECUADOR**

**2025–2026**

# Sistema de Control de Buses: Desarrollo e Implementación

**Resumen**

Este informe describe el desarrollo y la implementación de un sistema de control de acceso para el transporte público, utilizando tecnologías modernas como Visual Studio Community 2022 y MySQL. El sistema tiene como propósito optimizar la gestión de usuarios, conductores, rutas y buses, a la vez que mejora la seguridad y la eficiencia operativa mediante el uso de tarjetas de identificación para el registro de acceso de los pasajeros. En el presente informe se detallan los objetivos del proyecto, el marco teórico que sustenta su desarrollo, el diseño de la base de datos, la implementación del sistema, y las conclusiones obtenidas al finalizar la investigación.

# Introducción

La gestión eficiente de los servicios de transporte público es un componente clave para mejorar la calidad del servicio y satisfacer las expectativas de los usuarios. En muchos casos, el control manual de accesos y la gestión de rutas es susceptible a errores y falta de transparencia. Por lo tanto, la automatización del proceso de registro y control de accesos mediante un sistema informatizado puede aumentar la eficiencia operativa, mejorar la seguridad y garantizar una mayor satisfacción de los usuarios.

Este proyecto busca crear un sistema automatizado para gestionar el acceso a los buses utilizando tecnologías modernas, asegurando su correcta implementación en el entorno de transporte público.

# Objetivos

**Los objetivos fundamentales del proyecto son los siguientes:**

1. Diseñar e implementar una base de datos que contenga la información necesaria sobre usuarios, buses, conductores y rutas.
2. Desarrollar una aplicación en Visual Studio Community 2022 para permitir la interacción del usuario con la base de datos.
3. Facilitar el registro de acceso de los usuarios mediante el uso de tarjetas de identificación para cada pasajero.
4. Generar reportes y estadísticas que permitan monitorear el uso del sistema y la gestión de los recursos.

# Marco Teórico

El sistema propuesto se apoya en las siguientes tecnologías y principios:

* + Bases de datos relacionales: La gestión de grandes volúmenes de datos es esencial para mantener la coherencia y la integridad del sistema. En este caso, MySQL, un sistema de gestión de bases de datos relacional, se utiliza para organizar la información de manera estructurada.
  + Desarrollo con Visual Studio Community 2022: Este entorno de desarrollo integrado (IDE) facilita la creación de aplicaciones con lenguajes como C#, permitiendo una interfaz de usuario fluida y herramientas para la integración con bases de datos.
  + Principios de diseño de bases de datos: La normalización de la base de datos es crucial para minimizar redundancias y garantizar la integridad de los datos.

Además, se han adoptado prácticas de diseño modular y escalable para que el sistema sea fácil de mantener y expandir.

# Diseño del Sistema

El diseño del sistema se ha centrado en tres componentes principales: la base de datos, la aplicación de usuario y la integración de ambos.

# Estructura de la Base de Datos

**La base de datos está compuesta por las siguientes tablas:**

* + Usuarios: Información personal de los usuarios del sistema.
  + Tarjetas: Relación con los usuarios, para almacenar los datos de las tarjetas de identificación y su saldo.
  + Buses: Registra los datos relacionados con los buses disponibles en el sistema.
  + Conductores: Información de los conductores que operan los buses.
  + Rutas: Información de las rutas asignadas a los buses.
  + Registros de acceso: Registra los accesos de los usuarios a los buses utilizando sus tarjetas.

# Diagrama Entidad-Relación

El diseño de la base de datos se puede representar gráficamente mediante un diagrama entidad-relación que muestra las interacciones entre las tablas. En los anexos se incluye un diagrama que detalla las relaciones clave entre las diferentes entidades del sistema.

# Interfaz de Usuario

La interfaz de la aplicación permite realizar diversas funciones mediante botones intuitivos y un diseño claro. Algunas de las operaciones disponibles incluyen:

* + **Registro y gestión de usuarios.**
  + **Consulta de rutas y buses disponibles.**
  + **Registro de acceso de los usuarios mediante lectura de tarjetas. Implementación**

El desarrollo del sistema se basó en el uso de las siguientes tecnologías:

* C# para el desarrollo de la aplicación de escritorio.
* MySQL como el sistema de gestión de bases de datos.
* Entity Framework para la interacción con la base de datos a través de un enfoque de programación orientada a objetos.

# Las funcionalidades implementadas en la aplicación incluyen:

* CRUD (Crear, Leer, Actualizar, Eliminar) para las entidades principales (usuarios, buses, conductores, etc.).
* Gestión de saldo de tarjetas: Los usuarios pueden recargar su tarjeta y el sistema realiza un seguimiento del saldo.
* Generación de reportes: El sistema puede generar informes sobre el uso de las tarjetas y los buses.

# Base de Datos Lógica

La base de datos lógica tiene las siguientes tablas y relaciones clave:

# Usuarios Atributos:

* + ID\_Usuario (PK): Identificador único.
  + Nombre: Nombre del usuario.
  + Apellido: Apellido del usuario.
  + Email: Correo electrónico del usuario.
  + Telefono: Teléfono de contacto.

# Tarjetas

**Atributos:**

* + ID\_Tarjeta (PK): Identificador único de la tarjeta.
  + ID\_Usuario (FK): Relación con la tabla Usuarios.
  + Saldo: Saldo disponible en la tarjeta.

# Buses

**Atributos:**

* + ID\_Bus (PK): Identificador único del bus.
  + Placa: Número de placa del bus.
  + Capacidad: Capacidad máxima del bus.
  + Modelo: Modelo del bus.

# Conductores Atributos:

* + ID\_Conductor (PK): Identificador único del conductor.
  + Nombre: Nombre del conductor.
  + Apellido: Apellido del conductor.
  + Licencia: Número de licencia del conductor.

# Rutas

**Atributos:**

* + ID\_Ruta (PK): Identificador único de la ruta.
  + Nombre\_Ruta: Nombre descriptivo de la ruta.
  + Origen: Punto de inicio de la ruta.
  + Destino: Punto final de la ruta.

# Registros\_Acceso Atributos:

* + ID\_Registro (PK): Identificador único del registro.
  + ID\_Tarjeta (FK): Relación con la tabla Tarjetas.
  + ID\_Bus (FK): Relación con la tabla Buses.
  + Fecha\_Hora: Fecha y hora del acceso.

# Relaciones Clave

1. Usuarios - Tarjetas: Relación 1:N. Un usuario puede tener varias tarjetas, pero cada tarjeta pertenece a un único usuario.
2. Buses - Conductores: Relación 1:1. Cada bus tiene asignado un único conductor en un momento dado.
3. Rutas - Buses: Relación 1:N. Una ruta puede ser asignada a múltiples buses, pero cada bus opera en una única ruta en un momento dado.
4. Registros de acceso - Tarjetas y Buses: Relación N:M. Un registro de acceso vincula una tarjeta con un bus en un momento determinado.

# Conclusiones

El desarrollo del sistema de control de acceso a buses ha demostrado ser una solución efectiva para mejorar la eficiencia operativa del transporte público. La implementación de la base de datos en MySQL y la creación de una interfaz de usuario interactiva en Visual Studio Community 2022 proporciona una herramienta robusta para la gestión de recursos en el sistema de transporte. La automatización de procesos de acceso y la capacidad para generar informes detallados ofrecen una ventaja significativa para la administración de los buses, aumentando la transparencia, la seguridad y la satisfacción de los usuarios.

# Bibliografía

* + Date, C. J. (2019). *An Introduction to Database Systems*. Pearson.
  + Connolly, T., & Begg, C. (2020). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson.
  + Microsoft. (2022). *Visual Studio Community Documentation*. Retrieved from <https://visualstudio.microsoft.com/>
  + Oracle. (2022). *MySQL Documentation*. Retrieved from <https://dev.mysql.com/doc/>

# Anexos

**Base de datos:**

CREATE DATABASE IF NOT EXISTS acceso\_bus;

USE acceso\_bus;

-- usuario grupo7 con contraseña grupo7

CREATE USER IF NOT EXISTS 'grupo7'@'localhost' IDENTIFIED BY 'grupo7';

GRANT ALL PRIVILEGES ON acceso\_bus.\* TO 'grupo7'@'localhost' WITH GRANT OPTION;

FLUSH PRIVILEGES;

-- Tabla de Usuarios

CREATE TABLE IF NOT EXISTS Usuarios (

UsuarioID INT AUTO\_INCREMENT PRIMARY KEY,

Nombre VARCHAR(100) NOT NULL,

TipoUsuario ENUM('Pasajero', 'Conductor') NOT NULL

);

-- Tabla de Buses

CREATE TABLE IF NOT EXISTS Buses (

BusID INT AUTO\_INCREMENT PRIMARY KEY,

Placa VARCHAR(20) NOT NULL UNIQUE

);

-- Tabla de Rutas

CREATE TABLE IF NOT EXISTS Rutas (

RutaID INT AUTO\_INCREMENT PRIMARY KEY,

Origen VARCHAR(100) NOT NULL,

Destino VARCHAR(100) NOT NULL

);

-- Tabla de Horarios

CREATE TABLE IF NOT EXISTS Horarios (

HorarioID INT AUTO\_INCREMENT PRIMARY KEY,

RutaID INT NOT NULL,

BusID INT NOT NULL,

HoraSalida DATETIME NOT NULL,

FOREIGN KEY (RutaID) REFERENCES Rutas(RutaID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (BusID) REFERENCES Buses(BusID) ON DELETE CASCADE ON UPDATE CASCADE

);

-- Tabla de Accesos

CREATE TABLE IF NOT EXISTS Accesos (

AccesoID INT AUTO\_INCREMENT PRIMARY KEY,

UsuarioID INT NOT NULL,

HorarioID INT NOT NULL,

FechaHoraAcceso DATETIME NOT NULL,

FOREIGN KEY (UsuarioID) REFERENCES Usuarios(UsuarioID) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (HorarioID) REFERENCES Horarios(HorarioID) ON DELETE CASCADE ON UPDATE CASCADE

);

-- Trigger para validar que Origen y Destino no sean iguales

DELIMITER $$

CREATE TRIGGER before\_insert\_rutas

BEFORE INSERT ON Rutas

FOR EACH ROW

BEGIN

IF NEW.Origen = NEW.Destino THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: El origen y el destino no pueden ser iguales.';

END IF;

END $$

DELIMITER ;

-- Trigger para validar que la hora de salida de un horario sea en el futuro

DELIMITER $$

CREATE TRIGGER before\_insert\_horarios

BEFORE INSERT ON Horarios

FOR EACH ROW

BEGIN

IF NEW.HoraSalida <= NOW() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La hora de salida debe ser en el futuro.';

END IF;

END $$

DELIMITER ;

-- Trigger para validar que la fecha de acceso no sea en el pasado

DELIMITER $$

CREATE TRIGGER before\_insert\_accesos

BEFORE INSERT ON Accesos

FOR EACH ROW

BEGIN

IF NEW.FechaHoraAcceso < NOW() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Error: La fecha de acceso no puede ser en el pasado.';

END IF;

END $$

DELIMITER ;

**Restricciones de tipo ENUM** para limitar valores a ciertos tipos en TipoUsuario.

**Restricciones de unicidad** para garantizar que la columna Placa en la tabla Buses no tenga valores duplicados.

**Triggers para validaciones:**

Verificar que Origen y Destino en Rutas no sean iguales.

Verificar que HoraSalida en Horarios sea una fecha futura.

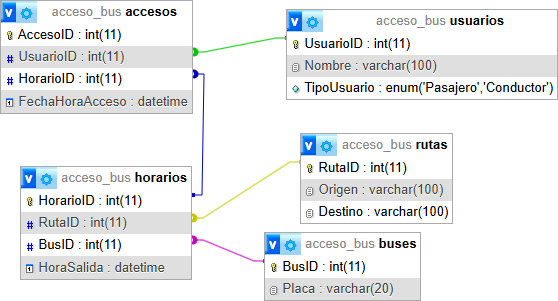
Verificar que FechaHoraAcceso en Accesos no sea una fecha en el pasado.

**Restricciones de clave foránea con** ON DELETE CASCADE y ON UPDATE CASCADE para mantener la integridad referencial.

El **usuario** y la **contraseña** que configuramos en el código SQL son:

* **Usuario:** grupo7
* **Contraseña:** grupo7

**Scrip base de datos:**

****

* + Diagrama de la Ventana Principal

