Informe Proyecto Segundo Bimestre

API de gestión de estudiantes (CRUD)

Materia

BASE DE DATOS

Estudiantes

Reyes Josselyn

Paladines Justin

Docente

Yadira Franco

Fecha de entrega

2026/01/25

Introducción

El presente proyecto bimestral de la asignatura Base de Datos tiene como objetivo el diseño, implementación y validación de una base de datos relacional orientada a un sistema de venta de videojuegos.
El sistema simula un entorno real de comercio, donde se gestionan clientes, videojuegos, ventas, métodos de pago y usuarios del sistema, garantizando la integridad, seguridad y consistencia de la información almacenada.

La base de datos constituye el núcleo central del sistema, ya que permite el almacenamiento estructurado de los datos, la ejecución eficiente de consultas, la aplicación de reglas de negocio y el control de accesos. Un diseño adecuado facilita el mantenimiento, la escalabilidad y la recuperación ante fallos, aspectos fundamentales en sistemas reales de venta.

Objetivo general

Diseñar, implementar y administrar una base de datos relacional para un sistema de venta de videojuegos, aplicando principios de modelado, normalización, seguridad y optimización, utilizando un sistema gestor estándar.

Objetivos Específicos

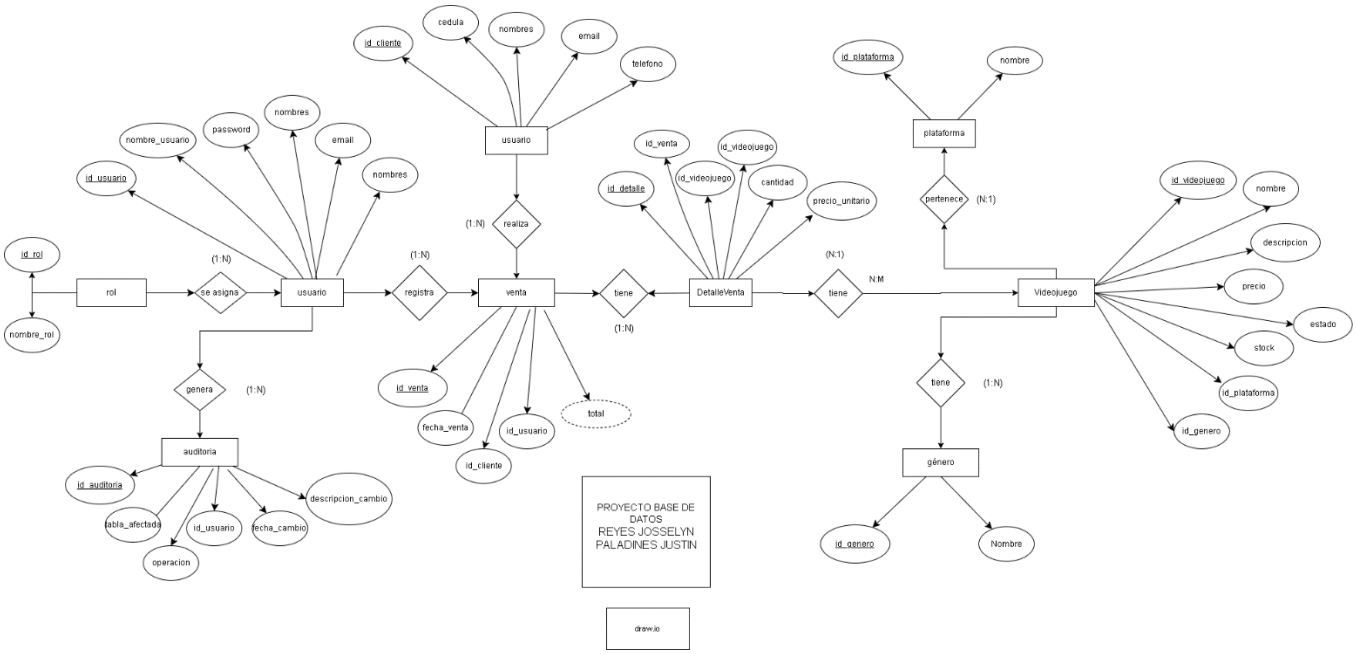
- Analizar los requerimientos del sistema de venta de videojuegos para identificar entidades, atributos y relaciones.
- Diseñar el modelo entidad–relación y transformarlo en un esquema lógico normalizado hasta tercera forma normal (3FN).
- Implementar la base de datos en un SGBD, definiendo tablas, claves primarias, foráneas y restricciones.
- Ejecutar consultas SQL que permitan gestionar la información del sistema.
- Integrar la base de datos con un backend y frontend funcional.
- Validar el correcto funcionamiento del sistema mediante pruebas técnicas y funcionales.

Modelo Conceptual

El modelo conceptual fue diseñado utilizando un **diagrama entidad–relación (E-R)**, en el cual se identifican las principales entidades del sistema y sus interacciones.

Las entidades principales son:

- **Usuario**: representa a los usuarios del sistema que registran ventas y generan auditorías.
- **Rol**: define los roles asignados a los usuarios del sistema.
- **Cliente** (modelado como usuario cliente): representa a los clientes que realizan compras.
- **Venta**: almacena la información general de cada transacción.
- **DetalleVenta**: representa los videojuegos incluidos en cada venta.
- **Videojuego**: almacena la información de los productos.
- **Plataforma**: representa la plataforma a la que pertenece cada videojuego.
- **Género**: clasifica los videojuegos según su tipo.
- **Auditoría**: registra las acciones realizadas por los usuarios del sistema.



Modelo Lógico

El modelo lógico se obtuvo a partir del modelo conceptual, transformando cada entidad en una tabla relacional.

Cada tabla cuenta con una clave primaria y, cuando corresponde, claves foráneas para representar las relaciones entre entidades.

El modelo se encuentra normalizado hasta la Tercera Forma Normal (3FN), evitando redundancias y garantizando la integridad de los datos.

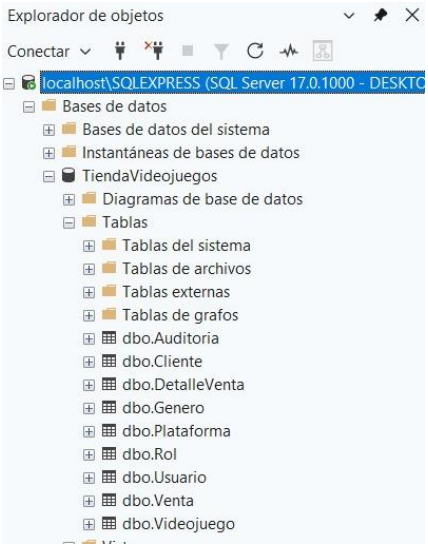
Entre los atributos más relevantes se encuentran:

- Identificadores únicos (id_usuario, id_cliente, id_venta, id_videojuego, etc.).
- Atributos descriptivos como nombres, correos, precios y descripciones.
- Atributos de control como fechas, estados y cantidades.

Implementación del SGBD

La base de datos fue implementada en un Sistema de Gestión de Base de Datos relacional, donde se realizaron las siguientes tareas:

- Creación de la base de datos.
- Definición de tablas según el modelo lógico.
- Implementación de claves primarias y claves foráneas.
- Aplicación de restricciones de integridad (NOT NULL, UNIQUE).
- Inserción de datos de prueba para validar el funcionamiento del sistema.



```sql
-- Guarda datos del cliente
CREATE TABLE Cliente (
  id_cliente INT IDENTITY(1,1) PRIMARY KEY,
  cedula VARCHAR(13) NOT NULL UNIQUE,
  nombres VARCHAR(100) NOT NULL,
  email VARCHAR(100) NOT NULL UNIQUE,
  telefono VARCHAR(20) NOT NULL UNIQUE
);

-- Guarda usuarios que usan el sistema (cajero/admin)
CREATE TABLE Usuario (
  id_usuario INT IDENTITY(1,1) PRIMARY KEY,
  nombre_usuario VARCHAR(50) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,          -- Contraseña guardada de forma segura
  nombres VARCHAR(100) NOT NULL,                -- Nombre real de la persona
  id_rol INT FOREIGN KEY REFERENCES Rol(id_rol)
);
```

```sql
-- Guarda los videojuegos que vende la tienda
CREATE TABLE Videojuego (
  id_videojuego INT IDENTITY(1,1) PRIMARY KEY,
  nombre VARCHAR(200) NOT NULL,
  descripcion VARCHAR(MAX) NOT NULL,
  precio DECIMAL(10,2) NOT NULL CHECK (precio > 0), -- Precio obligatorio y debe ser mayor a 0
  stock INT NOT NULL CHECK (stock >= 0),            -- Cantidad en stock, no puede ser negativa
  estado VARCHAR(20) CHECK (estado IN ('en stock', 'agotado')), -- Solo acepta estas 2 opciones
  id_plataforma INT FOREIGN KEY REFERENCES Plataforma(id_plataforma),
  id_genero INT FOREIGN KEY REFERENCES Genero(id_genero)
);

-- Guarda la venta
CREATE TABLE Venta (
  id_venta INT IDENTITY(1,1) PRIMARY KEY,
  fecha_venta DATETIME DEFAULT GETDATE(),       -- Fecha y hora automática cuando se crea la venta
  id_cliente INT FOREIGN KEY REFERENCES Cliente(id_cliente),
  id_usuario INT FOREIGN KEY REFERENCES Usuario(id_usuario), -- Usuario/cajero que atendió
  total DECIMAL(12,2) NOT NULL CHECK (total >= 0) -- Total de la venta, no puede ser negativo
);
```

```sql
-- Guarda el detalle: qué juegos se vendieron en esa venta
CREATE TABLE DetalleVenta (
  id_detalle INT IDENTITY(1,1) PRIMARY KEY,
  id_venta INT FOREIGN KEY REFERENCES Venta(id_venta) ON DELETE CASCADE, -- A qué venta pertenece
  id_videojuego INT FOREIGN KEY REFERENCES Videojuego(id_videojuego),    -- Qué juego se vendió
  cantidad INT NOT NULL CHECK (cantidad > 0),    -- Cuántas unidades (mínimo 1)
  precio_unitario DECIMAL(10,2) NOT NULL CHECK (precio_unitario > 0) -- Precio por unidad
);

-- Guarda un historial de cambios
CREATE TABLE Auditoria (
  id_auditoria INT IDENTITY(1,1) PRIMARY KEY,
  tabla_afectada VARCHAR(50) NOT NULL,
  operacion VARCHAR(10) CHECK (operacion IN ('INSERT','UPDATE','DELETE')), -- Qué se hizo: crear/editar/eliminar
  id_usuario INT FOREIGN KEY REFERENCES Usuario(id_usuario), -- Quién de los usuarios lo hizo
  fecha_cambio DATETIME DEFAULT GETDATE(),      -- Fecha y hora automática del cambio/GETDATE() devuelve la fecha
  descripcion_cambio VARCHAR(MAX) NOT NULL      -- Un texto comentando el porque de la acción
);
```

Consultas

En la base de datos TiendaVideojuegos se diseñaron y ejecutaron consultas SQL para cubrir las operaciones principales del sistema. Se implementó un listado de videojuegos disponibles, mostrando su plataforma y género mediante JOIN entre las tablas Videojuego, Plataforma y Genero.

Asimismo, se realizaron consultas de ventas por cliente, utilizando el procedimiento almacenado SP_VentasPorCliente, y consultas con JOIN entre Venta, DetalleVenta y Videojuego para obtener el detalle completo de cada venta.

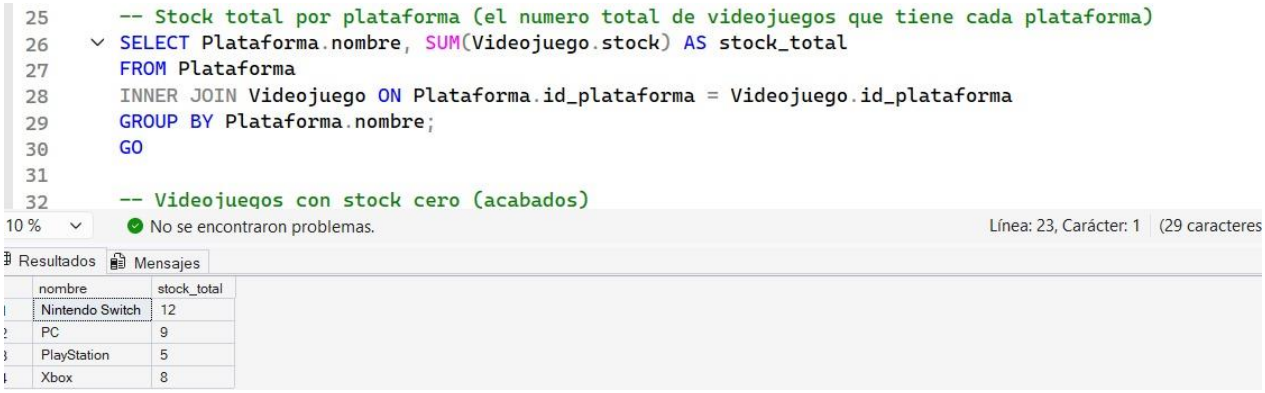
Se emplearon funciones de agregación como SUM y COUNT para calcular totales de ventas y cantidades de videojuegos, además de ejecutar operaciones CRUD (inserción, actualización y eliminación) de forma controlada, garantizando la integridad de los datos.

```
-- Cantidad de juegos por plataforma (verifica cuantos videojuegos DIFERENTES tiene cada plataforma)
SELECT Plataforma.nombre, COUNT(Videojuego.id_videojuego) AS cantidad_videojuegos
FROM Plataforma
INNER JOIN Videojuego ON Plataforma.id_plataforma = Videojuego.id_plataforma
GROUP BY Plataforma.nombre;
GO
```

110 % — No se encontraron problemas. — Línea: 13, Carácter: 1 — TABUL

Resultados | Mensajes

	nombre	cantidad_videojuegos
1	Nintendo Switch	1
2	PC	2
3	PlayStation	1
4	Xbox	1



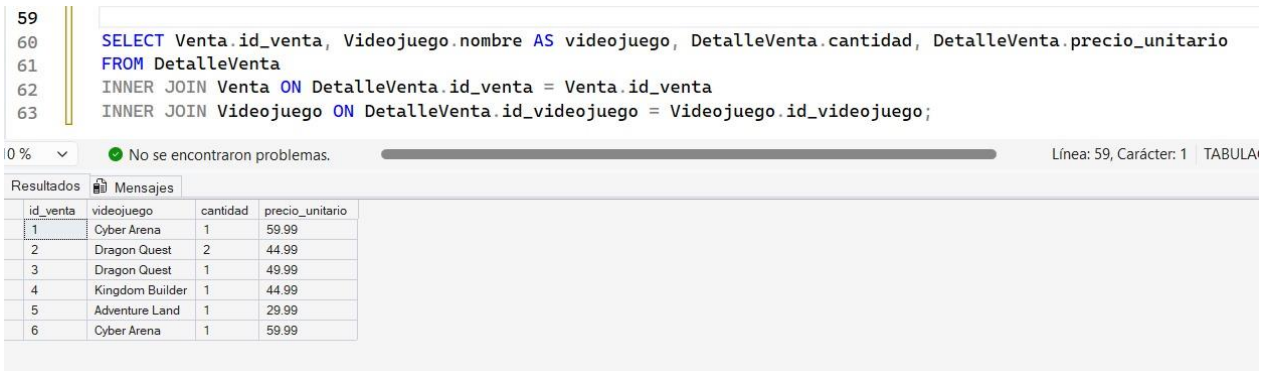
```
-- Stock total por plataforma (el numero total de videojuegos que tiene cada plataforma)
SELECT Plataforma.nombre, SUM(Videojuego.stock) AS stock_total
FROM Plataforma
INNER JOIN Videojuego ON Plataforma.id_plataforma = Videojuego.id_plataforma
GROUP BY Plataforma.nombre;
GO

-- Videojuegos con stock cero (acabados)
```

10 % — No se encontraron problemas. — Línea: 23, Carácter: 1 — (29 caracteres

Resultados | Mensajes

	nombre	stock_total
	Nintendo Switch	12
	PC	9
	PlayStation	5
	Xbox	8



```
SELECT Venta.id_venta, Videojuego.nombre AS videojuego, DetalleVenta.cantidad, DetalleVenta.precio_unitario
FROM DetalleVenta
INNER JOIN Venta ON DetalleVenta.id_venta = Venta.id_venta
INNER JOIN Videojuego ON DetalleVenta.id_videojuego = Videojuego.id_videojuego;
```

0 % — No se encontraron problemas. — Línea: 59, Carácter: 1 — TABULA

Resultados | Mensajes

	id_venta	videojuego	cantidad	precio_unitario
	1	Cyber Arena	1	59.99
	2	Dragon Quest	2	44.99
	3	Dragon Quest	1	49.99
	4	Kingdom Builder	1	44.99
	5	Adventure Land	1	29.99
	6	Cyber Arena	1	59.99

Backend, Frontend y Pruebas

Backend

El backend fue desarrollado para manejar la lógica del sistema y la comunicación con la base de datos.
Sus principales funciones incluyen:

- Gestión de usuarios y roles.
- Registro de ventas.
- Consulta de videojuegos y ventas.
- Ejecución de operaciones CRUD.
- Uso de consultas preparadas para prevenir inyección SQL.

Frontend

El frontend permite la interacción del usuario con el sistema de manera visual.
Desde la interfaz se pueden:

- Registrar ventas.
- Consultar videojuegos.
- Visualizar resultados de consultas.
- Verificar el funcionamiento del sistema en tiempo real.

Pruebas

Se realizaron pruebas para validar:

- Integridad referencial de la base de datos.
- Correcto funcionamiento de las operaciones CRUD.
- Funcionamiento de las consultas con JOIN.
- Generación de registros de auditoría.
- Cumplimiento de propiedades ACID en las transacciones.

Conclusión

El proyecto permitió comprender cómo se diseña y construye una base de datos aplicada a un caso real, en este caso un sistema de venta de videojuegos. A lo largo del desarrollo se trabajó desde la organización de la información hasta su implementación, lo que ayudó a entender la importancia de planificar correctamente antes de crear una base de datos.

El diseño del modelo permitió estructurar la información de forma ordenada y evitar datos repetidos, facilitando el manejo de clientes, ventas y videojuegos. Además, las consultas realizadas demostraron que la base de datos responde correctamente a las necesidades del sistema y permite obtener información clara y útil.

Finalmente, la conexión de la base de datos con el backend y el frontend permitió comprobar su funcionamiento en un entorno práctico, reforzando los conocimientos adquiridos durante la asignatura y mostrando cómo una base de datos es una parte fundamental en el desarrollo de sistemas reales.

Evidencias

GitHub:

josselynreyes-ux/ExamenFinal_ProyectoBasedeDatos