Proyecto Entrega Final

[Primer Entregable.]

[Segundo Entregable.]

Coderhouse SQL

Alumno: Jun

Criterios de la entrega:

Curso SQL: Proyecto Final
Datos para todos

Crearás tu propia base de datos, en la cual se implementará el modelo relacional para representar procesos basados en un modelo de negocio propio, con dataset público o ficticio. Implementarás los procesos técnicos que requiere el mantenimiento de una base de datos.

Objetivo 1 Crear una base de datos relacional, basada en un modelo de negocio.
Objetivo 2 Desarrollar objetos que permitan el mantenimiento de la base de datos.
Objetivo 3 Implementar consultas SQL que permitan la generación de informes.

User story/brief:

- El proyecto debe contener una estructura que permita la generación de reportes de diversas temáticas, todas relacionadas con la temática principal.
- También debe contener la documentación necesaria para que cualquier usuario, técnico o no, pueda entender en que está basada la base de datos y cuáles son los componentes de esta.

Piezas sugeridas

Te recomendamos incluir:
• Tablas • Vistas • Store procedure • Trigger • Funciones

Requisitos base

Los requisitos base serán parte de los criterios de evaluación para aprobar

el proyecto.
La base de datos debe contener al menos: ● 15 tablas, entre las cuales debe haber al menos 1 tabla de hechos, 2 tablas transaccionales. ✔ ● 5 vistas. ✔ ● 2 stored procedure. ✔ ● 2 trigger. ✔ ● 2 funciones ✔ El documento debe contener: ● Introducción, ✔ ● Objetivo, ✔ ● Situación problemática, ✔ ● Modelo de negocio, ✔ ● Diagrama de entidad relación ✔ ● Listado de tablas con descripción de estructura (columna, descripción, tipo de datos, tipo de clave), ✔ ● Scripts de creación de cada objeto de la base de datos, ✔ ● Scripts de inserción de datos, ✔ ● Informes generados en base a la información de la base, ✔ ● Herramientas y tecnologías usadas, ✔ ● Futuras líneas. ✔

Requisitos Extra

Los requisitos extra *pro-coders* no se incluyen en los criterios de evaluación.
● Implementar tablones para la generación de reportes.

Dont's

No es necesario ni recomendado.
-

Introducción

El objetivo principal de este proyecto es la creación de una base de datos relacional que gestione un sistema de videojuegos en línea. A través de esta base de datos, se busca organizar y optimizar diversas funciones relacionadas con la experiencia del jugador, incluyendo el registro de cuentas, la gestión de transacciones financieras, la adquisición de artículos virtuales, el seguimiento de logros, y la atención al cliente.

El proyecto se estructura en varias secciones clave que incluyen:

1. **Objetivos**: Detalla las metas que se persiguen, como la implementación de una base de datos eficaz y el desarrollo de consultas SQL para la generación de informes.
2. **Situación problemática**: Aborda los desafíos actuales en la gestión de datos en videojuegos en línea, como la falta de un sistema centralizado.
3. **Modelo de negocio**: Describe cómo se monetiza la experiencia del jugador a través de la venta de artículos virtuales y la gestión de transacciones.
4. **Diagrama de entidad relación**: Presenta la estructura de la base de datos y cómo se relacionan las diferentes entidades.
5. **Listado de tablas**: Proporciona una descripción detallada de cada tabla en la base de datos, incluyendo columnas, tipos de datos y claves.
6. **Scripts de creación e inserción de objetos**: Incluye los scripts necesarios para crear y poblar la base de datos, garantizando un correcto funcionamiento para las pruebas y demostraciones.
7. **Informes generados**: Describe los informes que se pueden obtener a partir de la base de datos, utilizando las vistas y funciones creadas.
8. **Herramientas y tecnologías**: Enumera las herramientas utilizadas, como MySQL y MySQL Workbench, y el lenguaje SQL para el desarrollo de scripts.
9. **Futuras líneas**: Sugiere posibles mejoras y expansiones del sistema en el futuro.

Objetivo

El objetivo principal es desarrollar una base de datos eficiente que soporte las funcionalidades del sistema de videojuego en línea. Esto incluye la gestión de usuarios, artículos virtuales, logros, transacciones financieras y atención al cliente. También se busca asegurar la escalabilidad y el fácil acceso a los datos para futuros análisis.

Situación problemática

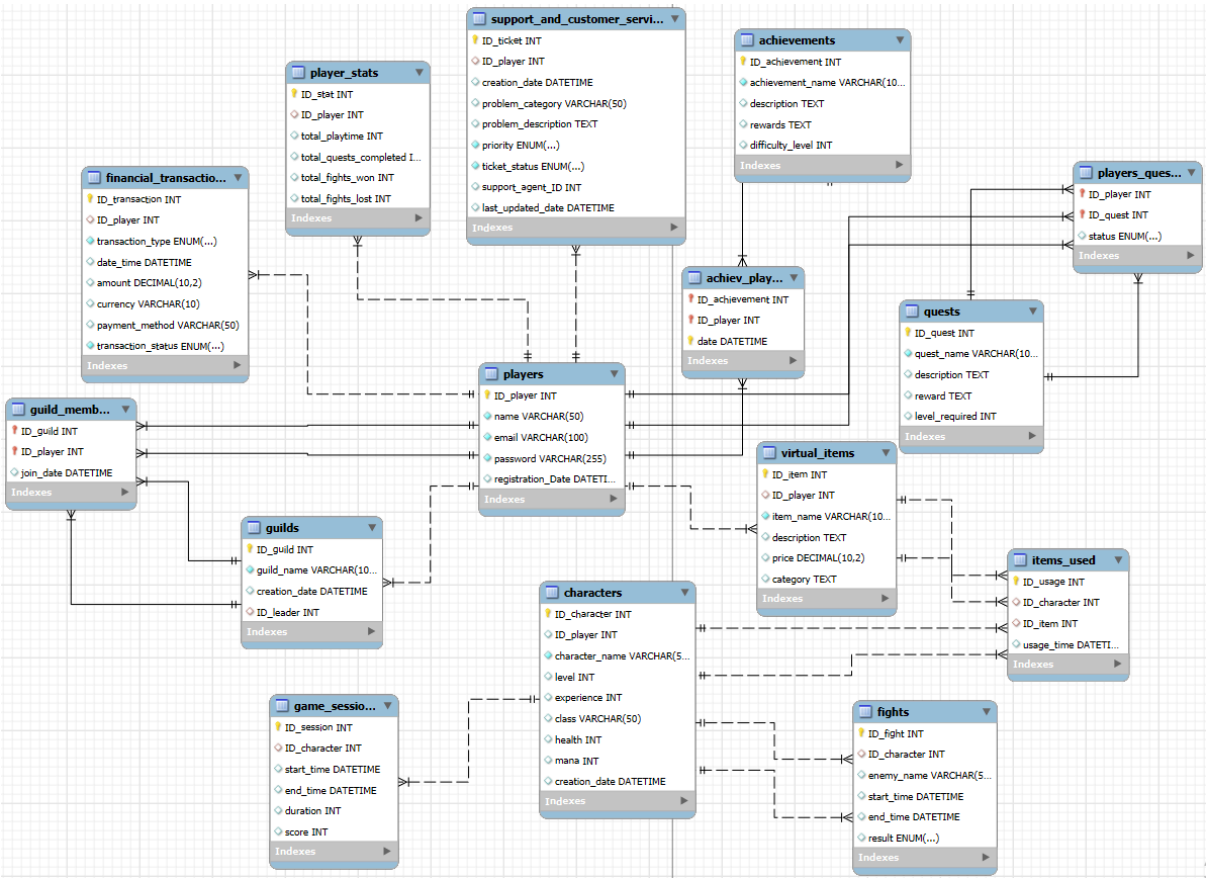
En la actualidad, los videojuegos en línea enfrentan problemas de gestión de datos, como la falta de un sistema para manejar transacciones, artículos virtuales y logros. Esto puede llevar a errores, pérdida de información y dificultades en la atención al cliente. Este proyecto aborda estas problemáticas mediante la implementación de una base de datos relacional que organiza y relaciona adecuadamente la información.

Modelo de negocio

El modelo de negocio se centra en la venta de artículos virtuales exclusivos y la gestión de transacciones dentro del juego. Los jugadores pueden comprar artículos, ganar logros y recibir recompensas, lo que fomenta la interacción y el compromiso con el juego. La atención al cliente es fundamental, ofreciendo soporte a los jugadores para resolver problemas y mejorar la experiencia del usuario.

Diagrama de entidad relación

https://github.com/Cavo2/Entregable-Final/blob/main/Diagram.png



Listado de tablas con descripción de estructura

1. Players: La tabla Players almacena la información básica de cada jugador que se registra en el juego. Contiene un ID único (ID_player) que se genera automáticamente, el nombre del jugador, su dirección de correo electrónico (que debe ser única para cada cuenta) y una contraseña encriptada. También se registra la fecha de registro del jugador, lo que permite a los administradores llevar un seguimiento de cuándo se unió cada jugador y, potencialmente, evaluar la actividad de los jugadores a lo largo del tiempo.

Players	**Tipo de Dato**
ID_player	INT
name	VARCHAR(50)
email	VARCHAR(100)
password	VARCHAR(255)
registration_Date	DATETIME

2. Characters: Este registra información detallada sobre los personajes que los jugadores crean. Se almacenan datos como el nombre del personaje, su nivel, experiencia acumulada, clase, salud, y maná, junto con la fecha de creación del personaje.

Characters	**Tipo de Dato**
ID_character	INT
ID_player	INT
character_name	VARCHAR(50)
level	INT
experience	INT
class	VARCHAR(50)
health	INT

mana	INT
creation_date	DATETIME

3. Player_Stats: Esta tabla almacena estadísticas de rendimiento de los jugadores, permitiendo un análisis de su actividad en el juego. Incluye el tiempo total de juego, el número de misiones completadas, y los resultados de los combates, tanto victorias como derrotas. Cada entrada está vinculada a un jugador específico mediante el ID_player.

Player_Stats	**Tipo de Dato**
ID_stat	INT
ID_player	INT
total_playtime	INT
total_quests_completed	INT
total_fights_won	INT
total_fights_lost	INT

4. Support_and_Customer_Service: Cada ticket tiene un ID único y contiene información sobre el jugador que lo generó, la fecha de creación, la categoría del problema y una descripción detallada del mismo. También incluye la prioridad del ticket (baja, media o alta) y su estado actual (abierto, en progreso, resuelto o cerrado). La tabla registra el ID del agente de soporte asignado y la última fecha de actualización, lo que facilita un seguimiento eficaz de los problemas reportados.

Support_and_Customer_Service	**Tipo de Dato**
ID_ticket	INT
ID_player	INT
creation_date	DATETIME
problem_category	VARCHAR(50)
problem_description	TEXT

priority	ENUM('Low', 'Medium', 'High')
ticket_status	ENUM('Open', 'In Progress', 'Resolved', 'Closed')
support_agent_ID	INT
last_updated_date	DATETIME

5. Financial_Transactions: La tabla Financial_Transactions documenta todas las transacciones financieras que realizan los jugadores dentro del juego. Cada transacción tiene un ID único y se clasifica según el tipo (depósito o compra). También se registran detalles como la fecha y hora de la transacción, el monto, la moneda utilizada, el método de pago, y se mantiene un registro del estado de cada transacción (pendiente, completada o fallida).

Financial_Transactions	**Tipo de Dato**
ID_transaction	INT
ID_player	INT
transaction_type	ENUM('Deposit', 'Purchase')
date_time	DATETIME
amount	DECIMAL(10, 2)
currency	VARCHAR(10)
payment_method	VARCHAR(50)
transaction_status	ENUM('Pending', 'Completed', 'Failed')

6. Achievements: La tabla Achievements alberga información sobre los logros que los jugadores pueden desbloquear. Cada logro tiene un ID único, un nombre descriptivo y una descripción que explica en qué consiste. Además, se registran las recompensas asociadas y el nivel de dificultad, lo que permite a los jugadores elegir logros que desean perseguir.

Achievements	**Tipo de Dato**
ID_achievement	INT

achievement_name	VARCHAR(100)
description	TEXT
rewards	TEXT
difficulty_level	INT

7. Achiev_Players: La tabla Achiev_Players vincula los logros obtenidos por los jugadores con sus respectivas cuentas. Cada entrada contiene el ID del logro, el ID del jugador y la fecha en que se desbloqueó el logro.

Achiev_Players	**Tipo de Dato**
ID_achievement	INT
ID_player	INT
date	DATETIME

8. Quests: La tabla Quests almacena información sobre las misiones disponibles en el juego. Cada misión tiene un ID único, un nombre descriptivo, una descripción detallada del objetivo de la misión, las recompensas que ofrece y el nivel requerido para poder participar.

Quests	**Tipo de Dato**
ID_quest	INT
quest_name	VARCHAR(100)
description	TEXT
reward	TEXT
level_required	INT

9. Players_Quests: La tabla Players_Quests relaciona a los jugadores con las misiones que han comenzado. Incluye el ID del jugador, el ID de la misión y el estado actual de la misión (no comenzada, en progreso o completada).

Players_Quests	Tipo de Dato
ID_player	INT
ID_quest	INT
status	ENUM('Not Started', 'In Progress', 'Completed')

10. Guilds: La tabla Guilds se utiliza para gestionar los gremios dentro del juego que los jugadores forman. Cada gremio tiene un ID único, un nombre y una fecha de creación. También incluye un campo para el ID del líder del gremio, que se relaciona con la tabla Players.

Guilds	Tipo de Dato
ID_guild	INT
guild_name	VARCHAR(100)
creation_date	DATETIME
ID_leader	INT

11. Guild_Members: La tabla Guild_Members vincula a los miembros de cada gremio. Contiene el ID del gremio y el ID del jugador, así como la fecha en que el jugador se unió al gremio

Guild_Members	Tipo de Dato
ID_guild	INT
ID_player	INT
join_date	DATETIME

12. Virtual_Items: La tabla Virtual_Items almacena información sobre los ítems virtuales que los jugadores pueden adquirir o usar en el juego. Cada ítem tiene un ID único, un nombre descriptivo, una descripción, un precio y una categoría. Esto es fundamental para la gestión de los inventarios y las transacciones dentro del juego.

Virtual_Items	Tipo de Dato

ID_item	INT
ID_player	INT
item_name	VARCHAR(100)
description	TEXT
price	DECIMAL(10, 2)
category	TEXT

13. Items_Used: La tabla Items_Used documenta los ítems que los personajes han utilizado durante el juego. Cada uso tiene un ID único, un ID del personaje, un ID del ítem y la fecha y hora en que se utilizó.

Items_Used	**Tipo de Dato**
ID_usage	INT
ID_character	INT
ID_item	INT
usage_time	DATETIME

14. Fights: La tabla Fights registra los combates en los que participan los personajes. Incluye un ID único para cada combate, el ID del personaje que luchó, el nombre del enemigo, el tiempo de inicio y final del combate, y el resultado (victoria, derrota o empate).

Fights	**Tipo de Dato**
ID_fight	INT
ID_character	INT
enemy_name	VARCHAR(50)
start_time	DATETIME
end_time	DATETIME
result	ENUM('Win', 'Lose', 'Draw')

15. Game_Sessions: La tabla Game_Sessions documenta las sesiones de juego de cada personaje. Cada sesión tiene un ID único, el ID del personaje, y se registra el tiempo de inicio y final, así como la duración total en minutos. Esta información permite a los desarrolladores analizar la actividad de los jugadores y realizar mejoras en la experiencia de juego.

Game_Sessions	**Tipo de Dato**
ID_session	INT
ID_character	INT
start_time	DATETIME
end_time	DATETIME
duration	INT
score	INT

16. Audit_Support_Service: registra las acciones realizadas sobre los tickets en el sistema de soporte y atención al cliente. Cada registro tiene un ID único y está vinculado a un ticket específico. Se documentan las acciones de creación, actualización y cierre, incluyendo el estado anterior y el nuevo estado del ticket, así como la fecha y hora de cada acción. Esta tabla es crucial para mantener un historial de cambios, lo que permite la trazabilidad y la transparencia en la gestión de tickets.

Audit_Support_Service	**Tipo de Dato**
Audit_ID	INT
ID_ticket	INT
Action	ENUM('Create', 'Update', 'Close')
Change_Date	DATETIME
Previous_Status	ENUM('Open', 'In Progress', 'Resolved', 'Closed')
New_Status	ENUM('Open', 'In Progress', 'Resolved', 'Closed')

Scripts de creación de cada objeto de la base de datos

https://github.com/Cavo2/Entregable-Final/blob/main/3.Objects.sql

Los objetos de la base de datos están diseñados para gestionar y analizar funcionalidades dentro del sistema del juego, la mayoría de estas se han desarrollado centrándose en la interacción de los jugadores con elementos virtuales y transacciones financieras. A continuación se describen brevemente cada una de ellas.

Vista 1: Player_Items_View

- Objetivo: Esta vista permite visualizar los tres ítems virtuales más caros del juego, mostrando sus detalles como nombre, descripción, precio y categoría.
- Tablas Manipuladas:
    - Players: Para obtener información sobre los jugadores (ID y nombre).
    - Virtual_Items: Para obtener detalles sobre los ítems que poseen los jugadores.
- Detalles: Sobre el campo ID_Player de las tablas Players y Virtual_Items se realiza un left join para determinar los ítems que ha comprado el jugador.
- Datos: Selecciona el nombre del ítem, su descripción, precio y categoría, así como el nombre del jugador y su ID.

Vista 2: Item_Economic_Activity_View

- Objetivo: Esta vista presenta un resumen de la actividad económica de los ítems virtuales, incluyendo el número de transacciones y el total de ingresos generados por cada ítem.
- Tablas Manipuladas:
    - Virtual_Items: Para obtener los nombres de los ítems.
    - Financial_Transactions: Para contar el número de transacciones y sumar los montos generados por cada ítem.
- Detalles: Se realiza un left join de los campos ID_Item y ID_Player para obtener una lista relacionada entre los ítems virtuales y los jugadores.
- Datos: Muestra el nombre del ítem, la cantidad de transacciones y el total de ingresos, ordenando por ingresos totales.

Vista 3: Player_Financial_Summary

- Objetivo: Esta vista proporciona un resumen financiero por jugador, mostrando el total depositado, el total gastado y el número de transacciones realizadas.
- Tablas Manipuladas:
    - Players: Para obtener información sobre los jugadores (ID y nombre).
    - Financial_Transactions: Para sumar las transacciones de depósito y compra, y contar el número total de transacciones.
- Detalles: Se utiliza un left join entre las tablas Players y Financial_Transactions, agrupando los resultados por el ID del jugador.
- Datos: Muestra el ID del jugador, el nombre del jugador, el total depositado, el total gastado y el conteo de transacciones.

Vista 4: Player_Virtual_Items

- Objetivo: Esta vista proporciona información sobre los jugadores y sus artículos virtuales, incluyendo el total de artículos que poseen y su valor total.
- Tablas Manipuladas:
    - Players: Para obtener datos sobre los jugadores (ID, nombre, email y fecha de registro).
    - Virtual_Items: Para contar los ítems y calcular su valor total.
- Detalles: Se realiza un left join entre Players y Virtual_Items, agrupando los resultados por el ID del jugador.
- Datos: Muestra el ID del jugador, el nombre del jugador, el email, la fecha de registro, el total de ítems, una lista de ítems y su valor total.

Vista 5: Financial_Transaction_Log

- Objetivo: Esta vista presenta un log de las transacciones financieras, mostrando detalles de cada transacción realizada por los jugadores.
- Tablas Manipuladas:
    - Financial_Transactions: Para obtener información sobre las transacciones (ID, tipo, monto, moneda, método de pago y estado).

    - Players: Para asociar cada transacción con el nombre del jugador correspondiente.
- Detalles: Se realiza un join entre las tablas Financial_Transactions y Players, ordenando los resultados por la fecha de la transacción en orden descendente.
- Datos: Muestra el ID de la transacción, el nombre del jugador, el tipo de transacción, la fecha, el monto, la moneda, el método de pago, el estado de la transacción y un mensaje que indica el resultado de la transacción.

Función 1: Get_Total_Spending

- Objetivo: Esta función calcula el total gastado por un jugador en todas sus transacciones.
- Tablas Manipuladas:
    - Financial_Transactions: Para sumar los montos de todas las transacciones del jugador especificado.
- Detalles: Se realiza la suma con la función SUM(), habiendo filtrado anteriormente por el ID del jugador para obtener todas las transacciones.
- Datos: DECIMAL. Retorna un valor decimal que representa el total gastado por el jugador.

Función 2: Get_Average_Spending_Status

- Objetivo: Esta función calcula el gasto promedio de un jugador y lo compara con el gasto promedio de todos los jugadores, retornando un mensaje de estado.
- Tablas Manipuladas:
    - Financial_Transactions: Para calcular el gasto promedio del jugador y del total de jugadores.
- Detalles: Se calculan los gastos promedios generales, y del jugador seleccionado, para a continuación, realizar una comparativa IF de estos valores.
- Datos: VARCHAR. Retorna un mensaje que indica si el jugador gasta más, menos o igual que el promedio general.

Procedimiento 1: Update_Support_Ticket_Status

- Objetivo: Este procedimiento permite actualizar el estado de un ticket de soporte y registrar la última fecha de actualización.
- Tablas Manipuladas:
    - Support_and_Customer_Service: Para actualizar el estado del ticket específico.
- Detalles: Se realiza un SET después de haber realizado un filtrado con WHERE.
- Datos: Se actualizan el campo Ticket_Status, y el campo Last_Updated_Date.

Procedimiento 2: Register_Transaction

- Objetivo: Este procedimiento registra una nueva transacción en el sistema. Antes de registrar esta nueva transacción, se valida la existencia del jugador y que el monto de la transacción no se trate de un valor negativo.
- Tablas Manipuladas:
    - Players: Para verificar si el jugador existe en el sistema.
    - Financial_Transactions: Para insertar la nueva transacción.
- Detalles: Se realiza un count(*) para verificar si el jugador existe. En caso de existir, procede con la creación de la transacción.
- Datos: Se insertan los siguientes datos en la transacción.
  ID_Player, Transaction_Type, Date_Time, Amount, Currency, Payment_Method, Transaction_Status.
- Nota: El parámetro “p_Payment_Method” tiene que ser los siguientes valores debido al último trigger ("Credit Card", "PayPal" y "Bank Transfer")

Trigger 1: Validate_Transaction_Amount

- Objetivo: Este trigger valida que el monto de una transacción no sea negativo antes de insertar el registro en la tabla de transacciones.
- Tablas Manipuladas:
    - Financial_Transactions: Se activa antes de la inserción de un nuevo registro.

- Detalles: Con la sentencia IF se hace la verificación de que el monto no sea negativo.
- Datos: VARCHAR. Devuelve una cadena de texto en caso de activación del trigger.

Trigger 2: Validate_Payment_Method

- Objetivo: Verifica si el método de pago proporcionado en la nueva transacción está en una lista de métodos permitidos (en este caso, "Credit Card", "PayPal" y "Bank Transfer").
- Tablas Manipuladas:
    - Financial_Transactions: Se activa antes de la inserción de un nuevo registro en esta tabla, y se verifica el método de pago.
- Detalles: Se utiliza un WHERE para verificar que el método de pago es uno de los permitidos. En caso de activación del trigger, se lanza un error.
- Datos: VARCHAR. Devuelve una cadena de texto en caso de activación del trigger.

Trigger 3: Audit_Support_Service

Objetivo: Registrar automáticamente la creación de un nuevo ticket en la tabla de auditoría Audit_Support_Service.

Tablas Manipuladas: Audit_Support_Service

- Support_and_Customer_Service: Se activa tras la inserción de un nuevo ticket.
- Audit_Support_Service: Se inserta un nuevo registro en esta tabla.

Detalles: Este trigger se ejecuta después de la inserción de un nuevo ticket. Captura el ID del ticket, la acción ('Create'), la fecha y hora de creación, el estado anterior (NULL) y el nuevo estado del ticket.

Datos: **ID_ticket**: INT,

Action: ENUM ('Create'),

Change_Date: DATETIME (hora de creación),

Previous_Status: ENUM (estado previo),

New_Status: ENUM (estado del nuevo ticket).

Scripts de inserción de datos

El siguiente script se ocupa de insertar registros en las tablas previamente definidas. Es fundamental haber seguido el orden de ejecución de los scripts adecuadamente para llevar a cabo las pruebas y demostraciones correspondientes.

https://github.com/Cavo2/Entregable-Final/blob/main/2.Population.sql

Informes generados en base a la información de la base

Los informes se pueden generar a través de las vistas y funciones definidas, como:

- **Player_Items_View**: Muestra los artículos de los jugadores.
- **Item_Economic_Activity_View**: Proporciona un resumen de la actividad económica de los artículos.
- **Funciones**: Get_Total_Spending y Get_Average_Spending_Status permiten obtener información sobre el gasto de los jugadores.

Herramientas y tecnologías usadas

- **SGBD**: Se utiliza MySQL como sistema de gestión de bases de datos para la creación y gestión de la base de datos.
- **Herramientas de desarrollo**: MySQL Workbench se emplea como herramienta principal para la administración visual de la base de datos. Esta herramienta proporciona una interfaz gráfica que facilita el diseño, desarrollo y mantenimiento de bases de datos.
- **Lenguaje de programación**: Se utiliza SQL como el lenguaje principal para la creación de scripts y procedimientos.

Futuras líneas

En futuras líneas, se pueden considerar las siguientes mejoras:

- Implementar un sistema de vistas más elaborado para monitorear el comportamiento de los jugadores y obtener datos relevantes.
- Expandir el sistema para incluir categorías de artículos, así como la posibilidad de que los jugadores creen sus propios artículos virtuales.