Colección de videojuegos

Autor: Jose Adrián Aglio Gascón

Fecha: 19/04/2024

Indice

2.Introducción

En este proyecto se va a realizar una aplicación donde podremos llevar una gestión de una colección de videojuegos para llevar un control sobre ellos, pudiendo introducir tanto datos del videojuego, como datos sobre la compañía, el género al que pertenece y la plataforma en la que esta disponible.

3.Herramientas y Métodos

Para realizar este proyecto he utilizado:

-SQLite para crear la base de datos con las tablas necesarias para la aplicación.

-ChatGPT para escribir el código con el lenguaje de programación Python para conectar la base de datos.

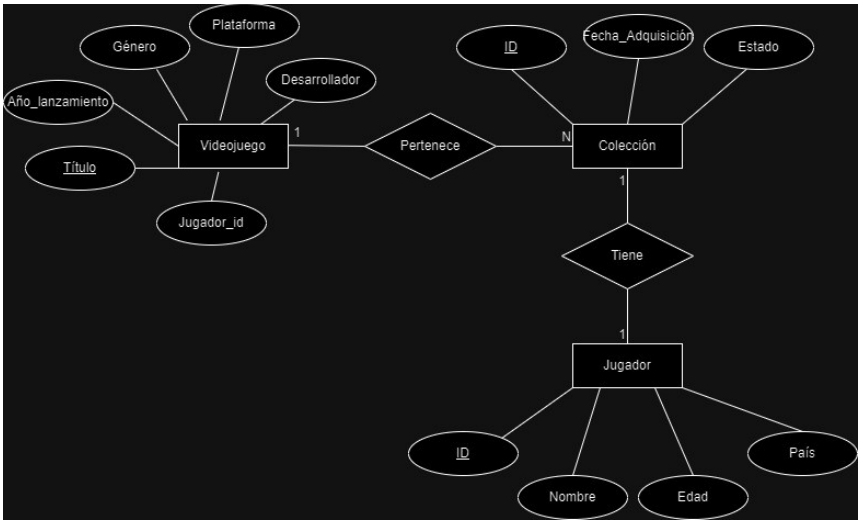
-VisualStudio Code para conectar la base de datos y poder modificarla con el código de Python.

-Draw.io para crear el modelo entidad-relación de la base de datos.

-Umbrello para crear el caso de uso y el diagrama de clases para la aplicación.

4.Perspectiva Estática

-E/R(Entidad-Relación)



-Paso a tablas

VIDEOJUEGO: Título+Año_lanzamiento+Género+Plataforma+Desarrollador+Jugador_id
Clau Aliena: Jugador_id → Jugador(ID)
COLECCIÓN: ID+Fecha_Adquisición+Estado
JUGADOR: ID+Nombre+Edad+País
Pertenece: Título_v+ID_col
Clau Aliena: Título_v → Videojuego(Título)
ID_col → Colección(ID)
Tiene: ID_col+ID_jug
Clau Aliena: ID_jug → Jugador(ID)

-DDL

DDL, o Data Definition Language, es un conjunto de comandos utilizados para definir y modificar la estructura de las bases de datos.
Aqui creamos la tabla Jugador si no existe con los atributos ID, Nombre, Edad, Pais.

```
self.c.execute('''CREATE TABLE IF NOT EXISTS Jugador (
                    ID INTEGER PRIMARY KEY,
                    Nombre TEXT,
                    Edad INTEGER,
                    Pais TEXT
                    )''')
```

Aqui creamos la tabla Videojuego si no existe con los atributos Titulo, Año_Lanzamiento, Genero, Desarrollador, Plataforma y Jugador_ID.

```
self.c.execute('''CREATE TABLE IF NOT EXISTS Videojuego (
                    Titulo TEXT,
                    Año_Lanzamiento INTEGER,
                    Genero TEXT,
                    Desarrollador TEXT,
                    Plataforma TEXT,
                    Jugador_ID INTEGER,
                    FOREIGN KEY (Jugador_ID) REFERENCES Jugador(ID)
                    )''')
```

Aqui creamos la tabla Coleccion si no existe con los atributos ID, Fecha_Adquisicion y Estado.

```
self.c.execute('''CREATE TABLE IF NOT EXISTS Coleccion (
                    ID INTEGER PRIMARY KEY AUTOINCREMENT,
                    Fecha_Adquisicion TEXT,
                    Estado TEXT
                    )''')
```

-DML

DML, o Data Manipulation Language, es un conjunto de comandos utilizados para manipular los datos almacenados en una base de datos.
Aqui insertamos datos en las tablas de Jugador, Videojuego y Coleccion.

```
self.c.execute("INSERT INTO Jugador (ID, Nombre, Edad, Pais) VALUES (?, ?, ?, ?)", (jugador_id, nombre, edad, pais))
```

```
self.c.execute("INSERT INTO Videojuego (Titulo, Año_Lanzamiento, Genero, Desarrollador, Plataforma, Jugador_ID) VALUES (?, ?, ?, ?, ?, ?)",
               (titulo, año_lanzamiento, genero, desarrollador, plataforma, self.jugador_id))
```

```
self.c.execute("INSERT INTO Coleccion (Fecha_Adquisicion, Estado) VALUES (?, ?)", (fecha_adquisicion, estado))
```

-DQL

DQL, o Data Query Language, se utiliza específicamente para realizar consultas o consultas de datos en una base de datos. Aqui utilizamos el Select para elegir una tabla y poder modificar un atributo.

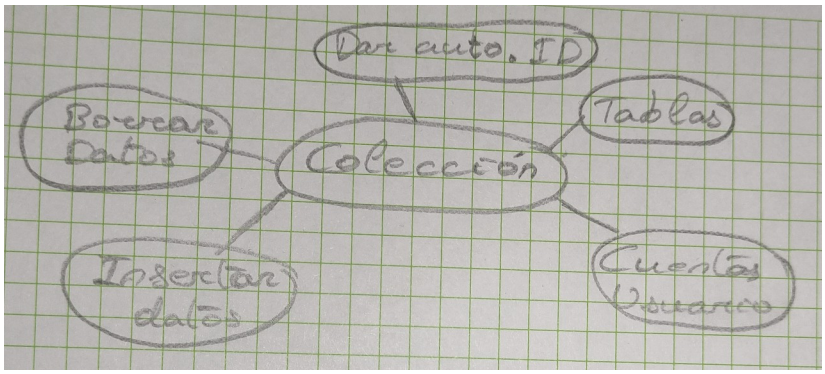
```
self.c.execute("SELECT MAX(ID) FROM Jugador")
self.c.execute("SELECT * FROM Jugador WHERE ID=?", (jugador_id,))
self.c.execute("SELECT * FROM Jugador WHERE ID=? AND Nombre=?", (jugador_id, nombre))
self.c.execute("SELECT * FROM Videojuego WHERE Jugador_ID=?", (self.jugador_id,))
```

-DCL

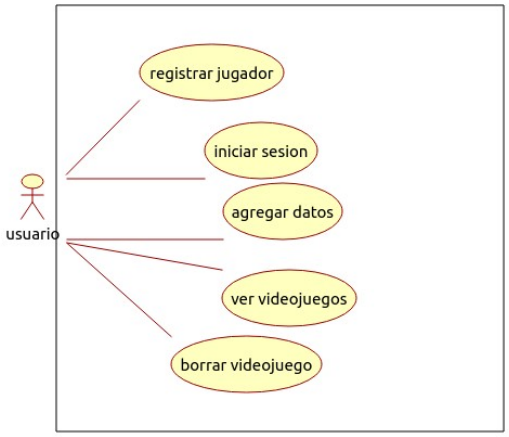
DCL, o Data Control Language es un lenguaje proporcionado por el sistema de gestión de base de datos que incluye una serie de comandos SQL que permiten al administrador controlar el acceso a los datos contenidos en la base de datos.
Aqui damos permisos al usuario admin de la aplicación sobre la tabla Jugador.

```
# Otorgar permisos al usuario en la tabla Jugador
self.c.execute("GRANT SELECT, INSERT, UPDATE, DELETE ON Jugador TO <admin>")
```

5.Perspectiva Dinámica
-Skecth



-Casos de Uso



-Registrar jugador:
Se utiliza para registrar un jugador proporcionandole una ID única y introduciendo su nombre, edad y país.

-Iniciar sesión:
Se utiliza para que el usuario entre con su cuenta particular donde tendrá su colección guardada en la base de datos.

-Agregar datos:
Tiene la función de agregar los datos que quiera el usuario a la base de datos.

-Ver videojuegos:
Se utiliza para que el usuario pueda ver que datos y que videojuegos tiene en su colección.

-Borrar videojuegos:
Se utiliza para borrar los videojuegos, junto con los otros datos, de la colección del usuario.

6.Conclusiones

-Resumen de los resultados obtenidos

Tenemos una aplicación con la que podemos registrarnos y introducir datos de los videojuegos que queramos junto con su fecha de adquisición y estado para nuestra colección, y también podemos ver la lista de videojuegos de nuestra colección que ya hayamos agregado iniciando sesión en nuestra cuenta.

-Reflexiones sobre el proceso y posibles mejoras futuras

Algunas de las posibles mejoras que se podrían implementar serian que a la hora de agregar videojuegos nos muestre un desplegable con algunos títulos de videojuegos, también añadir alguna foto a la colección, y poder hablar con otros usuarios para comparar colecciones.

7.Bibliografía y Webgrafía

Lo que utilizado es ChatGPT de OpenAI para escribir el código y corregir algunos errores de la aplicación, y draw.io para crear el modelo entidad-relacion.

ChatGPT. (s. f.). https://chat.openai.com/
draw.io - free flowchart maker and diagrams online. (s. f.). https://app.diagrams.net/

Repositorio de GitHub

https://github.com/JoseAdrianAG/MicroProyectoBD.git