Autor: Jose Adrián Aglio Gascón
Fecha: 24/05/2024

Colección de videojuegos

Índice

1. 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 tanto introducir como eliminar datos del videojuego. Podremos introducir datos como el género al que pertenece, el título del videojuego, el año de lanzamiento, el año en el que lo hemos obtenido, la plataforma en la que esta disponible, y el estado en que tenemos el videojuego(digital o físico). También podrás ver una lista con tus videojuegos introducidos, y podrás crear o eliminar un usuario, independiente a los demás, teniendo cada uno su propia colección.
En este documento iremos detallando como se ha ido realizando la aplicación y explicando cada una de sus partes, y al final tendrás un enlace a GitHub para que puedas utilizar esta aplicación.

2. Herramientas y Métodos

Aqui se detalla el software utilizado para realizar el proyecto con sus funciones, y despues pasaremos a ver son las perspectivas estáticas y dinámicas,

- MongoDB para poder conectar la base de datos con 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.

3. Perspectiva Estática

La perspectiva estática en bases de datos y diseño de sistemas es fundamental para entender y documentar la estructura fija y las relaciones dentro del sistema. En bases de datos, se enfoca en la organización de tablas, columnas y claves, mientras que en el diseño de sistemas, se representa mediante diagramas de clases, componentes y estructuras de datos. Esta perspectiva es crucial para el diseño, desarrollo, mantenimiento y documentación del sistema, asegurando que todos los elementos estén bien definidos y organizados.

3.1. E/R(Entidad-Relación)

El modelo Entidad-Relación es una herramienta fundamental en el diseño de bases de datos que permite organizar y estructurar la información de manera lógica y comprensible.

En la **Figura 1.1** tenemos el diagrama Entidad-Relación para el proyecto Colección de Videojuegos

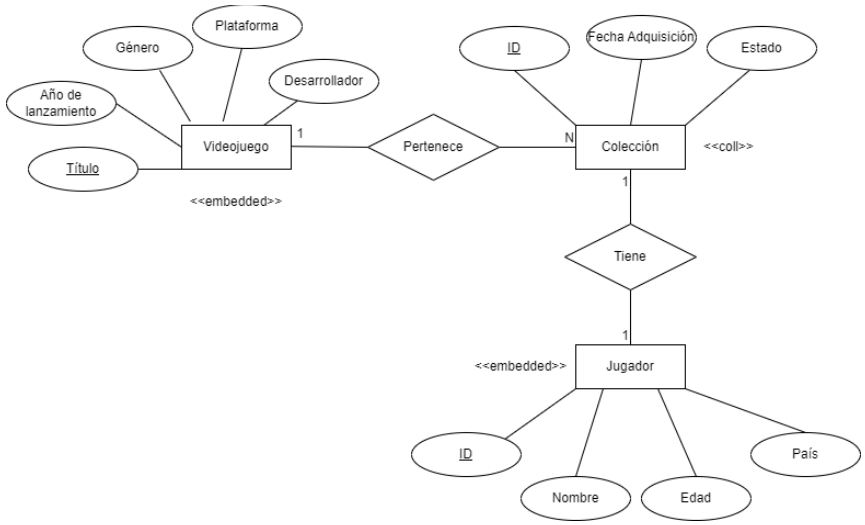

Fig 1.1

3.2. Paso a tablas

En este apartado explicaremos los patrones utilizados en el diagrama anterior.

Primero tenemos el “Patron de documento incorporado”, con la relacion 1 a N (que significa 1 a muchos), donde tenemos que el atributo Videojuego es de tipo embedded, ya que en la base de datos este atributo tendrá los datos de Coleccion.

Por último tenemos también el mismo patrón pero con una relacion 1 a 1, ya que un jugador solo podrá tener una colección, y el atributo Colección tendrá datos del atributo Jugador

3.3. DDL

DDL, o Data Definition Language, es un conjunto de comandos utilizados para definir y modificar la estructura de las bases de datos.
Por ejemplo tenemos estas lineas:

self.collection = self.db[‘jugadores’]

Esta linea crea la colección “jugadores”.

self.videojuegos = self.db[‘videojuegos’]

Esta linea crea la colección “videojuegos”.

3.4. DML

DML, o Data Manipulation Language, es un conjunto de comandos utilizados para manipular los datos almacenados en una base de datos.
Por ejemplo tenemos estas lineas:

self.collection.insert_one({“ID”: siguiente_id, “Nombre”: nombre, “Edad”: edad, “Pais”: pais})

Esta linea inserta un nuevo documento en la colección “jugadores”.

self.videojuegos.insert_one({“Titulo”: titulo, “Año_Lanzamiento”: año_lanzamiento, “Genero”: genero, “Desarrollador”: desarrollador, “Plataforma”: plataforma, “Jugador_ID”: self.jugador_id})

Esta linea inserta un nuevo documento en la colección “videojuegos”.

self.colecciones.insert_one({“Fecha_Adquisicion”: fecha_adquisicion, “Estado”: estado, “Jugador_ID”: self.jugador_id})

Esta linea inserta un nuevo documento en la colección “colección”.

3.5. 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.
Por ejemplo tenemos estas lineas:

self.collection.find_one({“Nombre”:nombre})

max_id_jugador = self.collection.find_one(sort=[(“ID”, -1])

videojuegos = self.db[‘videojuegos’].find({“Jugador_ID”: self.jugador_id})

Estas lineas realizan consultas a la base de datos para recuperar documentos que cumplan con ciertos criterios de búsqueda.

3.6. 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.
Por ejemplo tenemos estas lineas:

import tkinter as tk

import tkinter.simpledialog as simpledialog

from tkinter import ttk, messagebox

from pymongo import MongoClient, errors

class RegistroJugador(tk.Toplevel)

class iniciarCuenta(tk.Toplevel)

class MenuDatos(tk.Toplevel)

Estas lineas definen las clases y las importaciones de módulos necesarios para el funcionamiento del programa.

4. Perspectiva Dinámica

La "perspectiva dinámica" en una base de datos se refiere a la capacidad de la base de datos para manejar y representar datos en movimiento o cambiantes, permitiendo la ejecución de consultas y análisis en tiempo real sobre datos que están siendo constantemente actualizados.

4.1. Sketch

Un sketch es un boceto rápido y preliminar utilizado para capturar y comunicar ideas de manera visual.

En la **Figura 1.3** tenemos un sketch con la idea principal del programa y sus funciones

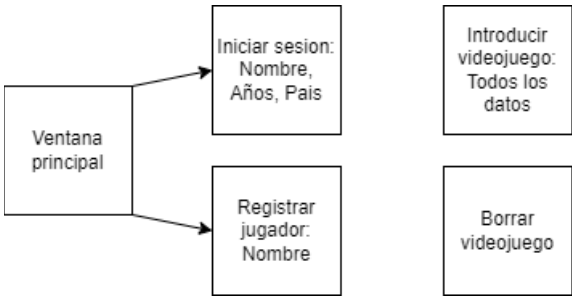

Fig 1.3

4.2. Casos de Uso

Los casos de uso son una herramienta esencial en el desarrollo de software para capturar y comunicar los requisitos funcionales del sistema. Describen cómo los actores interactúan con el sistema para alcanzar objetivos específicos, y proporcionan una base sólida para el diseño, desarrollo, pruebas y documentación del sistema.

En la **Figura 1.4** tenemos el diagrama de los casos de uso con todas las funciones del programa, explicadas a continuación.

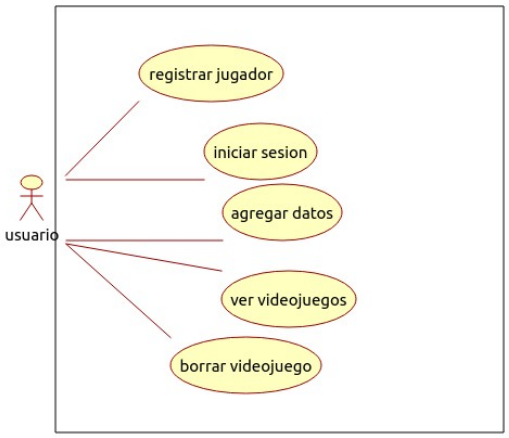

Fig 1.4

Aqui se explican los usos que va a tener la aplicación:

- 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.

5. Conclusiones

5.1. 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.

5.2. 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.

6. Bibliografía y Webgrafía

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

ChatGPT. (s. f.). https://chat.openai.com/

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

Umbrello Project - Welcome to Umbrello - The UML Modeller. (s. f.). https://uml.sourceforge.io/

Repositorio de GitHub

https://github.com/JoseAdrianAG/MicroProyectoBD2