

Resolución del ejercicio 1

Desarrollar un **modelo de negocio de objetos** para un **videoclub** utilizando IA (ChatGPT) y Mermaid.
Implementar el **modelo y sus pruebas unitarias**,
aplicando iteraciones para mejorarlo.

Paso 1: Comprender el modelo de negocio del videoclub

Antes de comenzar sería bueno **analizar el modelo de negocio** del videoclub para identificar las principales **entidades y sus relaciones**.



Descripción breve del modelo de negocio: Un videoclub alquila películas a clientes.

Se necesita representar este negocio en términos de objetos de programación. Para comenzar, puedes **realizar un diagrama simple** (puede ser un boceto a mano o un diagrama inicial) **que ilustre las entidades y sus relaciones.**



Por ejemplo:

Entidades clave:

- a. **Película:** Representa los filmes disponibles para alquiler.
- b. **Cliente:** Persona que alquila películas.
- c. **Alquiler:** Registro de una transacción de alquiler entre un cliente y una película.

Relaciones:

- d. Un Cliente puede tener múltiples Alquileres.
- e. Una Película puede estar asociada a múltiples Alquileres.

Utilizar las siguientes preguntas como guía para la confección del modelo:

- **¿Cuáles son las entidades principales en un videoclub?**

Ejemplos: Película, Cliente, Alquiler.

- **¿Qué atributos debe tener cada entidad?**

- Película: Título, Género, Disponibilidad.
- Cliente: Nombre, Email, Fecha de registro.

- **¿Qué métodos o funciones deben tener las entidades?**

- Película: Alquilar(), Devolver().
- Cliente: Registrarse(), ConsultarAlquileres().

- **¿Cómo se relacionan las entidades entre sí?**

Ejemplo: Un Cliente puede alquilar múltiples Películas, y una Película puede ser alquilada por diferentes Clientes.

- **¿Qué información debe guardarse en cada relación?**

Ejemplo: Fecha de inicio y fin del alquiler.

- **¿Qué operaciones o acciones deben realizarse en el sistema?**

Ejemplo: Registrar un nuevo alquiler, devolver una película, generar una factura.

Paso 2: Generar el Modelo de Objetos con ChatGPT

Utilizar **ChatGPT** para generar el modelo de objetos **basado en el diseño inicial** del videoclub.

ChatGPT puede ayudar a definir las **clases, atributos, y métodos** necesarios para el sistema del videoclub.

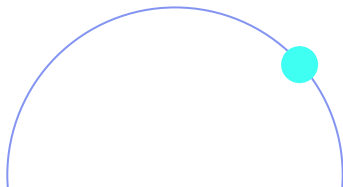
A continuación, se detalla el **proceso y el prompt recomendado**.

Prompt a utilizar

“Crear un modelo de clases para un videoclub que alquila películas.

Incluir las siguientes entidades:

- Clases
- Atributos
- Métodos
- Relaciones entre clases”.



Crear un modelo de clases para un videoclub que alquila películas.
Incluir las siguientes entidades:

- Clases
- Atributos
- Métodos
- Relaciones entre clases



Para modelar un sistema de videoclub que alquila películas, necesitamos definir las clases principales, sus atributos y métodos, así como las relaciones entre ellas. Aquí está un modelo básico:

1. Clase Película

- **Atributos:**
 - `titulo` (String): Título de la película.
 - `anio` (Integer): Año de lanzamiento.
 - `genero` (String): Género de la película (e.g., Acción, Comedia, Drama).

Contrasta tu modelo con el generado por ChatGPT

- ¿Qué **diferencias** hay?
- ¿Qué aspectos del modelo de ChatGPT **podrían mejorar** tu diseño?
- ¿Hay alguna **entidad en el modelo de ChatGPT que no esté en tu propuesta**?
- ¿Hay alguna **entidad en tu propuesta** que no esté en el modelo de ChatGPT?
- ¿Faltan **atributos importantes** en tu modelo que están presentes en el modelo de ChatGPT?
- ¿Existen **atributos en el modelo de ChatGPT** que no parecen necesarios?
- ¿Hay **métodos importantes** en el modelo de ChatGPT que no has incluido en tu propuesta?
- ¿Hay **métodos en tu propuesta** que ChatGPT no ha mencionado y podrían ser útiles?
- ¿Las **relaciones entre entidades** en tu modelo coinciden con las del modelo de ChatGPT?
- ¿Hay **relaciones adicionales** en el modelo de ChatGPT que no habías considerado?
- ¿El modelo de ChatGPT **ofrece más detalles o es más específico** en algunos aspectos?
- ¿Tu modelo es más **detallado en algún aspecto** en comparación con el modelo de ChatGPT?

Paso 3: Visualizar el modelo con Mermaid

Usar Mermaid para visualizar el modelo de objetos generado. Esto facilitará su comprensión y revisión.

Prompt utilizado:

“Generar el diagrama Mermaid del modelo propuesto”.



Pasos para visualizar el modelo:

- Generar el código Mermaid** del modelo de objetos obtenido de ChatGPT.
- Utilizar [Mermaid.live](https://mermaid.live) para ver el diagrama.
- Introducir el código Mermaid** en el editor de *Mermaid.live* para generar el diagrama.



ChatGPT:




Generar el diagrama Mermaid del modelo propuesto



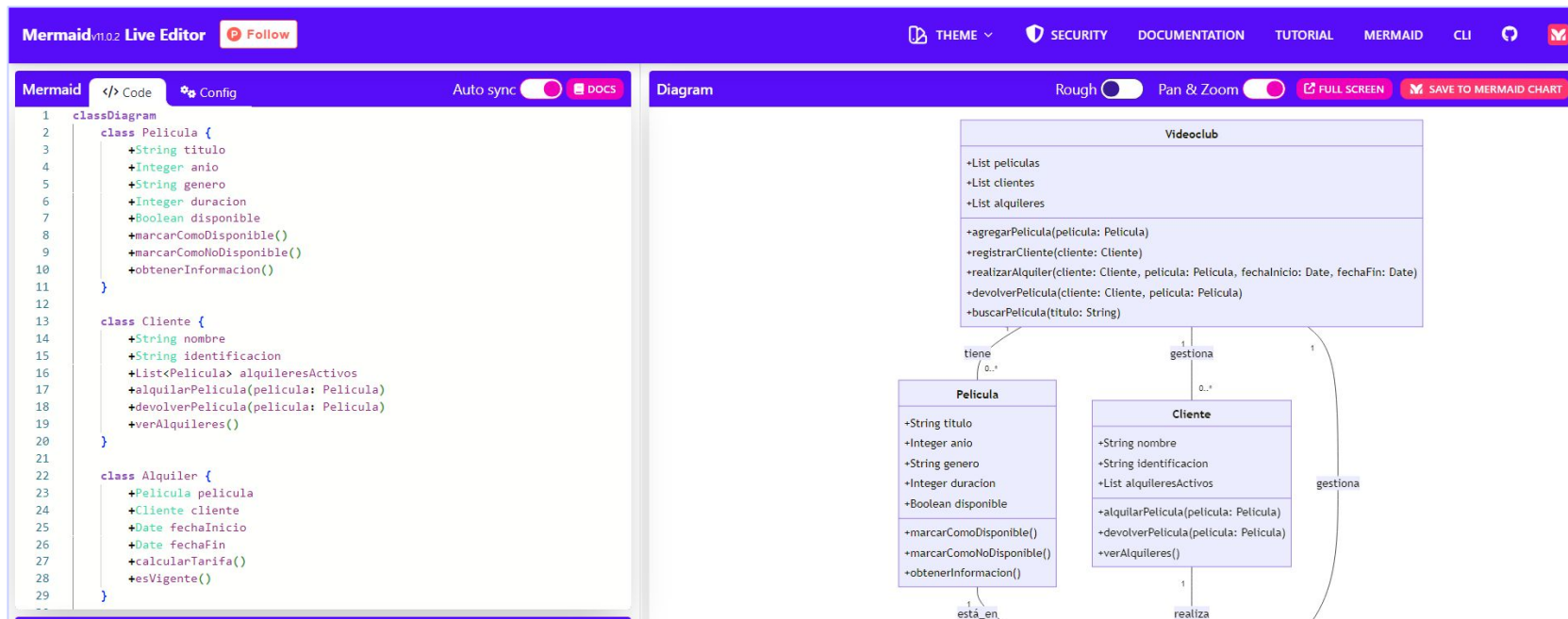
Aquí está el diagrama Mermaid para el modelo de clases del videoclub que alquila películas:

mermaid

 Copiar código

```
classDiagram
    class Pelicula {
        +String titulo
        +Integer anio
        +String genero
        +Integer duracion
        +Boolean disponible
        +marcarComoDisponible()
        +marcarComoNoDisponible()
        +obtenerInformacion()
    }
```


Diagrama Mermaid de ejemplo:



Paso 4: Implementar el modelo en código

Convertir el **modelo de objetos en código funcional** con el lenguaje de programación elegido.

Utilizar el siguiente *prompt* para **generar el código basado en el modelo de objetos**:

“A partir del modelo de objetos para un videoclub (clases, atributos y métodos), generar el código en [nombre del lenguaje de programación]. Incluir los siguientes elementos:

-Definición de las clases.

-Implementación de atributos y métodos.

-Relaciones entre las clases.”

Instrucciones:

- Generar el código** con el *prompt* proporcionado.
- Revisar el código** para asegurarse de que refleja correctamente el modelo de objetos.
- Implementar el código** en el entorno de desarrollo y realizar pruebas para asegurar su funcionalidad.



ChatGPT:



A partir del modelo de objetos para un videoclub (clases, atributos y métodos), generar el código en [Python]. Incluir los siguientes elementos:

Definición de las clases.


Implementación de atributos y métodos.

Relaciones entre las clases.



Aquí tienes el código en Python para el modelo de objetos del videoclub. El código incluye la definición de las clases, atributos, métodos y las relaciones entre ellas.

python

 Copiar código

```
from typing import List, Optional
from datetime import datetime

class Pelicula:
    def __init__(self, titulo: str, anio: int, genero: str, duracion: int):
        self.titulo = titulo
        self.anio = anio
        self.genero = genero
        self.duracion = duracion
        self.disponible = True
```

Paso 5: Escribir pruebas unitarias

Verificar que el código implementado funciona correctamente. Las pruebas unitarias son esenciales para asegurar que cada componente del código se comporta como se espera.

Prompt para generar pruebas unitarias con ChatGPT:

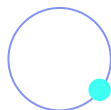
“Generar pruebas unitarias para el código en [Python]. El código contiene clases y métodos que deben ser evaluados. Por favor, contempla pruebas para:

- *Casos de prueba para entradas válidas e inválidas.*
- *Comprobación de resultados esperados.*
- *Manejo de excepciones”.*

Pasos para utilizar ChatGPT:

- Proporcionar el código:** Compartir el código de las clases y métodos que se necesitan probar.
- Usar el *prompt* adecuado:** Utilizar un *prompt* específico para pedir a ChatGPT que genere las pruebas unitarias.
- Revisar y ajustar:** Revisar las pruebas generadas y ajustarlas según sea necesario.
- Implementar las pruebas unitarias** en algún entorno de preferencia.

ChatGPT:



Generar pruebas unitarias para el código en [Python]. El código contiene clases y métodos que deben ser evaluados. Por favor, contempla pruebas para:

- Casos de prueba para entradas válidas e inválidas.
- Comprobación de resultados esperados.
- Manejo de excepciones.



Claro, aquí te dejo un conjunto de pruebas unitarias para el código en Python utilizando el módulo `unittest`. Estas pruebas cubren los métodos principales, casos de prueba para entradas válidas e inválidas, comprobación de resultados esperados, y manejo de excepciones.

python

 Copiar código

```
import unittest
from datetime import datetime, timedelta

# Suponiendo que el código del videoclub está en un archivo llamado videoclub.py
from videoclub import Pelicula, Cliente, Alquiler, Videoclub

class TestPelicula(unittest.TestCase):
```

Verificar las pruebas generadas por ChatGPT e iterar sobre la solución.



Guiar la escritura de pruebas:

- **¿Qué funcionalidades deben ser probadas?**

- ¿Qué métodos y atributos necesitan pruebas específicas?

- **¿Qué casos de prueba son necesarios?**

- ¿Qué entradas diferentes deben ser probadas?
- ¿Qué resultados esperados se deben verificar?

- **¿Cómo manejar casos de error?**

- ¿Qué sucede si los métodos reciben entradas inválidas?

- **¿Cómo verificar el comportamiento esperado?**

- ¿Cómo se puede comprobar que el código se comporta correctamente en diferentes escenarios?

Paso 6: Iterar y mejorar el modelo

La Inteligencia Artificial brinda mejores resultados cuando **se itera sobre una solución**. Esto puede requerir volver sobre los pasos anteriores.

Recuerda que la palabra final sobre el modelo la tienes tú y no la IA, y si algo no te gusta o quieres implementar debes decírselo.



Ejemplo de *prompt* para iterar:

“Revisar el modelo de objetos para un videoclub y su implementación en código. Por favor, proporciona recomendaciones para mejorar el diseño y la implementación. Considera aspectos como:

- *Consistencia y claridad del modelo.*
- *Uso de buenas prácticas de programación.*
- *Manejo de excepciones y casos especiales.*
- *Optimización de relaciones y atributos”.*

ChatGPT:



Revisar el modelo de objetos para un videoclub y su implementación en código. Por favor, proporciona recomendaciones para mejorar el diseño y la implementación. Considera aspectos como:

- Consistencia y claridad del modelo.
- Uso de buenas prácticas de programación.
- Manejo de excepciones y casos especiales.
- Optimización de relaciones y atributos.



Aquí tienes algunas recomendaciones para mejorar el diseño y la implementación del modelo de objetos para un videoclub:

1. Consistencia y Claridad del Modelo

- **Definición de Clases:** Asegúrate de que cada clase tenga una única responsabilidad bien definida. Por ejemplo, la clase `Videoclub` está haciendo mucho, como gestionar películas, clientes y alquileres. Considera descomponerla en varias clases o agregar un diseño de patrón de arquitectura como el patrón de repositorio o el patrón de servicio.

Experimentación adicional

Además del videoclub, prueba a **modelar otros sistemas que sean de tu interés**. Puedes aplicar la misma metodología y *prompts* para diseñar y refinar modelos de cualquier negocio o aplicación.



Más ideas de modelos a explorar:

- **Sistema de reservas:** Modela un sistema de reservas para hoteles, vuelos, o eventos.
- **Aplicación de gestión personal:** Desarrolla un modelo para una app que ayude a organizar tareas, calendarios, y notas.
- **E-commerce:** Diseña un sistema para una tienda en línea, incluyendo gestión de inventario, usuarios, y órdenes.
- **Red social:** Crea un modelo de una red social con usuarios, publicaciones, comentarios, y relaciones.

Preguntas guía:

- **¿Qué funcionalidades clave tiene el nuevo modelo?** ¿Cuáles son los componentes principales del sistema que deseas modelar?
- **¿Cómo se relacionan las entidades?** Define cómo interactúan y se conectan las entidades del nuevo modelo.
- **¿Qué nuevos desafíos aparecen?** Considera las particularidades y posibles complicaciones del sistema que quieres modelar.
- **¿Cómo podrías optimizar el modelo?** Aplica las buenas prácticas que aprendiste para mejorar tu nuevo diseño.

Consejo final

- **Itera y aprende:** Cada nuevo experimento es una oportunidad para mejorar tus habilidades en modelado, implementación y validación.
- **Solicita *feedback* a ChatGPT:** No dudes en usar ChatGPT para revisar tus nuevos modelos, obtener sugerencias de mejora, y explorar nuevas funcionalidades.



¡Terminaste el módulo!
Todo listo para rendir el examen