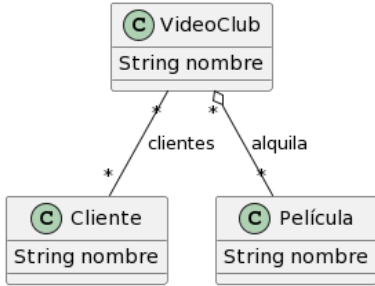


1. Introducción

ChatGPT es un bot con el que podemos conversar a través de la introducción de *prompts*, para los que dará una respuesta. Un prompt es una pequeña entrada de texto que se proporciona a ChatGPT para que genere una respuesta en forma de texto conversacional. El prompt puede ser cualquier tipo de entrada de texto, como una pregunta, una declaración o incluso una palabra o frase simple. Por supuesto, estas frases pueden incluir peticiones para realizar tareas simples (por ejemplo, escribir un resumen de un asunto determinado).

En nuestro caso, nos vamos a centrar en prompts que nos conduzcan a la generación de un diagrama de clases de UML como los que hemos visto en clase. Por ejemplo, para intentar generar el pequeño diagrama de clases UML sobre un videoclub como el que tenemos más abajo, podríamos escribir el siguiente prompt:

Prompt	Diagrama de clases UML	PlantUML
Me gustaría escribir un diagrama de clases en PlantUML en el que un Videoclub alquila muchas películas. El Videoclub tiene muchos clientes, y tanto el Videoclub, como las películas y los clientes deben tener nombre	 <pre> classDiagram class VideoClub { String nombre } class Cliente { String nombre } class Película { String nombre } VideoClub "*" -- "*" Cliente : clientes VideoClub "*" -- "*" Película : alquila </pre>	<pre> @startuml class VideoClub { String nombre } class Cliente { String nombre } class Película { String nombre } VideoClub "*" -- "*" Cliente : clientes VideoClub "*" o-- "*" Película : alquila @enduml </pre>

En el prompt de arriba, podemos observar que hemos añadido una notación específica llamada PlantUML, en lugar de mencionar simplemente UML. Esto es así por conveniencia, ya que ChatGPT no genera diagramas en formato de imagen, pero sí puede generar una representación textual del diagrama, que se puede traducir directamente a una imagen como la de la figura de arriba, si la insertamos en la web de Plant UML (<https://www.plantuml.com/plantuml/>).

Es importante destacar que el prompt que se proporciona puede tener un gran impacto en la respuesta generada por ChatGPT, ya que el modelo utiliza el contexto del prompt para generar una respuesta relevante y coherente. Por lo tanto, el prompt debe ser cuidadosamente diseñado para obtener la respuesta deseada del modelo.

Aun diseñando el prompt con cuidado, en muchas ocasiones ChatGPT no generará el resultado esperado. Para intentar mejorarlo, podemos seguir una conversación con el bot proporcionando múltiples prompts sucesivos en los que pidamos que se modifique uno u otro aspecto del resultado obtenido. Por ejemplo, si genera una clase *Película* que no contiene un atributo *nombre*, podemos indicarle que las películas deben tener un nombre.

2. Objetivo

El objetivo de esta actividad es determinar la utilidad de ChatGPT para generar diagramas de clases, así como realizar una reflexión crítica sobre sus capacidades, ventajas e inconvenientes en el contexto de un proyecto de desarrollo de software.

3. Procedimiento

Para llevar a cabo esta actividad, debemos completar los siguientes pasos:

1. Abre una cuenta en OpenAI/ChatGPT, siguiendo las instrucciones que se pueden encontrar en <https://chat.openai.com/>
2. Realiza los ejercicios 3.1 y 3.2 que se detallan a continuación. Para ello, habrá que:
 - a. Llevar a cabo las actividades descritas en los ejercicios.
 - b. Entregar los archivos que se detallan en la sección de “entrega” de cada uno de los ejercicios.
3. Finalmente, contestar las preguntas en el formulario que se encuentra en la siguiente URL: [URL eliminada]

3.1. Ejercicio: Generación a partir de descripción de dominio

En este ejercicio vamos a crear un diagrama de clases para un problema que formulamos a continuación, con la ayuda de ChatGPT.

Descripción del problema

Queremos desarrollar una aplicación para gestionar las obras que se representan en los teatros de una ciudad.

- Una obra se puede representar en varios teatros, y varias obras se pueden representar en un mismo teatro. Los teatros se encuentran en diversas localizaciones.
- Las obras pueden ser óperas o musicales, y pueden representarse por el día o por la noche.
- En cada obra, pueden participar uno o más intérpretes, que son identificados por su nombre.
- Una obra está escrita por uno o varios autores, que son identificados por su nombre.
- Cada obra se puede representar en una o más sesiones, que tienen una hora de comienzo y una hora de finalización.

Para resolver este ejercicio, los pasos a seguir son los siguientes:

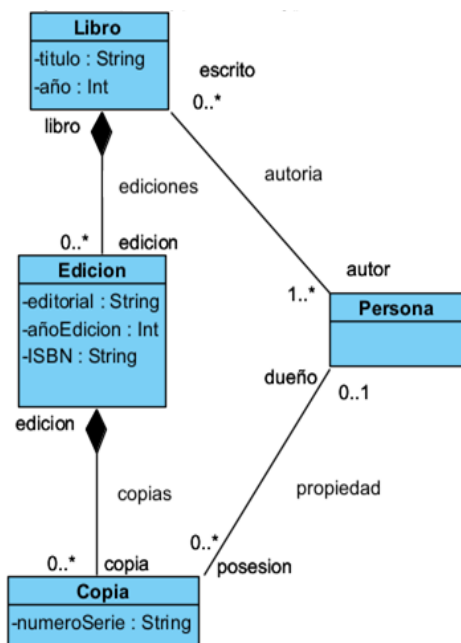
1. Generar un diagrama de clases para la solución a través de una conversación con ChatGPT (llamaremos a este diagrama **D31-ChatGPT**). Para ello habrá que:
 - a. Escribir un prompt inicial.
 - b. Si el resultado no es el esperado, escribir prompts sucesivos en los que se pida a ChatGPT modificar el diagrama generado para corregir los posibles errores/omisiones.
2. Si fuera necesario, modificar manualmente el resultado generado por ChatGPT para corregir los posibles problemas que presente el diagrama (llamaremos a este diagrama **D31-Final**). Para ello, se puede dibujar el diagrama creado por ChatGPT (**D31-ChatGPT**) en Visual Paradigm y, a partir de ahí, corregirlo.

Entrega: Se entregarán dos documentos pdf con los siguientes contenidos en el punto de entrega del CV:

- **D31-conversación.pdf:** en el que se incluirá una transcripción de la conversación con ChatGPT para la generación de **D31-ChatGPT** (basta con seleccionar toda la conversación, copiarla y pegarla en el documento). Debe incluir también una imagen de **D31-ChatGPT**.
- **D31-reflexión.pdf:** en el que se incluirá una descripción de:
 - Los problemas encontrados (si los hubiera) en **D31-ChatGPT**.
 - Cómo se han solucionado los problemas para llegar a **D31-Final**. La descripción deberá estar acompañada de una justificación razonada de los cambios realizados.
 - Debe incluir también una imagen de **D31-Final**.

3.2. Ejercicio: Generación a partir de diagrama de clases

En este segundo ejercicio, vamos a intentar generar un diagrama de clases similar al diagrama que proporcionamos a continuación.



Para resolver este ejercicio, los pasos a seguir son los siguientes:

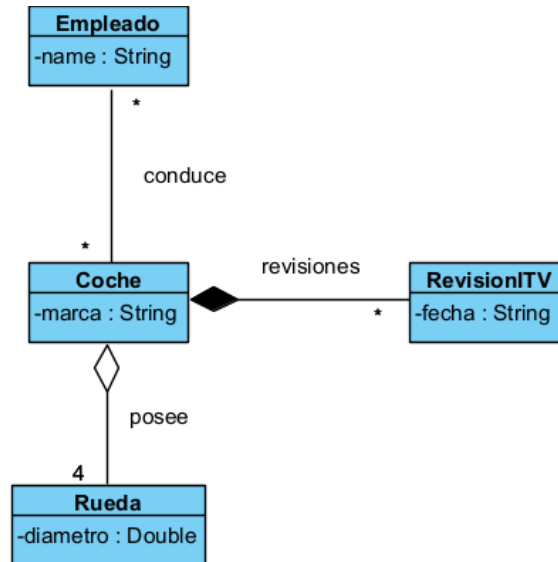
1. Intenta generar un diagrama de clases como el ofrecido a través de una conversación con ChatGPT. Para ello habrá que:
 - a. Escribir un prompt inicial.
 - b. Si el resultado no es el esperado, escribir prompts sucesivos en los que se pida a ChatGPT modificar el diagrama generado para corregir los posibles errores/omisiones con el objetivo de aproximar el diagrama al de la figura de arriba.
2. Si fuera necesario (por ejemplo, porque el diagrama obtenido diste mucho del diagrama esperado), se puede iniciar una nueva conversación desde cero para intentar obtener el resultado esperado, pudiendo comenzar con un prompt diferente.
3. Cuando se vea que no es posible mejorar más el diagrama obtenido, se considerará el diagrama final como **D32-ChatGPT**.

Entrega: Se entregarán dos documentos pdf con los siguientes contenidos en el punto de entrega del CV:

- **D32-conversación.pdf:** en el que se incluirá una transcripción de la conversación con ChatGPT para la generación de **D32-ChatGPT** (basta con seleccionar toda la conversación, copiarla y pegarla en el documento). Debe incluir también una imagen de **D32-ChatGPT**.
- **D32-reflexión.pdf:** en el que se incluirá una descripción de las dificultades que se observa para que ChatGPT pueda llegar a la solución esperada. Debe incluir también una imagen de **D32-ChatGPT**.

4. Ejemplo de conversación con ChatGPT

Imaginemos que queremos obtener el siguiente diagrama de clases UML:



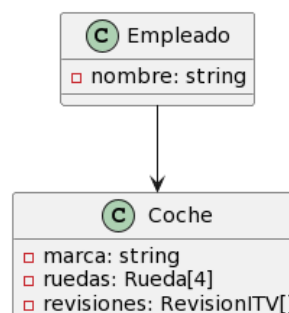
La siguiente podría ser una posible conversación para intentar obtener el diagrama (las imágenes no las ofrece ChatGPT directamente, sino que se obtienen introduciendo la solución en PlantUML de ChatGPT en <https://www.plantuml.com/plantuml/>):

Tenemos un sistema compuesto de coches, donde distintos empleados pueden conducir distintos coches. Cada coche tiene información sobre su marca, y cada empleado tiene un nombre. Cada coche se compone de 4 ruedas, y de cada rueda conocemos su diámetro. Por último, se tiene un registro de las revisiones de ITV que se realiza sobre cada coche, donde para cada revisión se registra su fecha. Necesito que realices un diagrama de clases UML usando la notación PlantUML de este sistema.

¡Claro! Aquí te presento el diagrama de clases UML usando la notación PlantUML:

```

@startuml
class Coche {
    -marca: string
    -ruedas: Rueda[4]
    -revisiones: RevisionITV[]
}
class Empleado {
    -nombre: string
}
class Rueda {
    -diametro: float
}
class RevisionITV {
}
  
```



Introducción a la Ingeniería del Software

Actividad de modelado de Diagrama de Clases UML con ChatGPT

```
-fecha: date
}

Empleado --> Coche
Coche --> Rueda
Coche --> RevisionITV

@enduml
```

En este diagrama, se puede observar que existen cuatro clases principales: Coche, Empleado, Rueda y RevisionITV. La clase Coche tiene información sobre la marca del coche, las ruedas que lo componen (como un array de objetos de tipo Rueda) y un registro de las revisiones de ITV realizadas (como un array de objetos de tipo RevisionITV). La clase Empleado simplemente tiene información sobre el nombre del empleado. La clase Rueda tiene información sobre el diámetro de la rueda. Y la clase RevisionITV tiene información sobre la fecha en que se realizó la revisión de ITV.

La relación entre estas clases se establece mediante flechas que indican la dirección de la relación. En este caso, un Empleado puede conducir varios Coches, por lo que hay una relación de asociación entre Empleado y Coche. Un Coche, a su vez, está compuesto por 4 objetos de tipo Rueda y tiene varias RevisionITV, por lo que hay una relación de composición entre Coche y Rueda, y una relación de agregación entre Coche y RevisionITV.

Espero que este diagrama sea útil para ti. Si necesitas más ayuda, ¡no dudes en preguntar!

Gracias. Los atributos "ruedas" y "revisiones" de la clase "Coche" no serían necesarios, porque esa información la tenemos en las asociaciones. Además, por favor, mejora el diagrama para que incluya multiplicidades en las relaciones.

¡Claro! Aquí te presento una versión actualizada del diagrama de clases UML usando la notación PlantUML, teniendo en cuenta tus sugerencias:

```
@startuml

class Coche {
    -marca: string
}

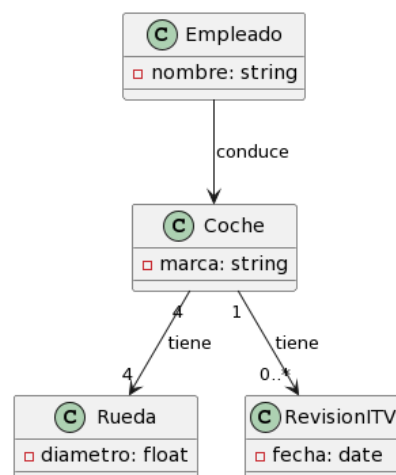
class Empleado {
    -nombre: string
}

class Rueda {
    -diametro: float
}

class RevisionITV {
    -fecha: date
}

Empleado --> Coche : conduce
Coche "4" --> "4" Rueda : tiene
Coche "1" --> "0..*" RevisionITV : tiene

@enduml
```



Introducción a la Ingeniería del Software

Actividad de modelado de Diagrama de Clases UML con ChatGPT

En este diagrama, se puede observar que se han eliminado los atributos "ruedas" y "revisiones" de la clase Coche, ya que la información está presente en las asociaciones. Además, se han agregado multiplicidades en las relaciones.

La relación entre las clases se establece mediante flechas que indican la dirección de la relación y un texto que especifica la multiplicidad de la relación. En este caso, un Empleado puede conducir varios Coches (relación 1:N), por lo que se ha agregado la multiplicidad "1:N" en la flecha que va de Empleado a Coche. Un Coche, a su vez, tiene 4 Ruedas (relación 4:4), por lo que se ha agregado la multiplicidad "4:4" en la flecha que va de Coche a Rueda. Y un Coche puede tener varias RevisionITV (relación 1:N), por lo que se ha agregado la multiplicidad "1:N" en la flecha que va de Coche a RevisionITV.

Espero que esta versión actualizada del diagrama sea de utilidad para ti. Si necesitas más ayuda, ¡no dudes en preguntar!

Así está mejor, gracias, pero todavía se puede mejorar. Ten en cuenta, para las multiplicidades, que una rueda va a estar únicamente en un coche. Además, me gustaría que añadieras asociaciones de agregación o composición donde sea necesario.

¡Claro! Aquí te presento la versión mejorada del diagrama de clases UML usando la notación PlantUML, teniendo en cuenta tus sugerencias:

```
@startuml
```

```
class Coche {
    -marca: string
    -ruedas: Rueda[4]
    -revisiones: RevisionITV[*]
}
```

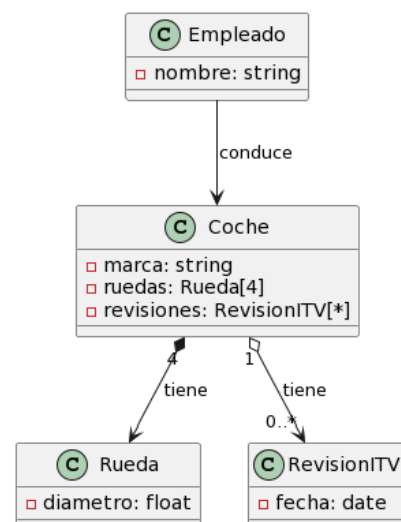
```
class Empleado {
    -nombre: string
}
```

```
class Rueda {
    -diametro: float
}
```

```
class RevisionITV {
    -fecha: date
}
```

```
Empleado --> Coche : conduce
Coche "4" *--> Rueda : tiene
Coche "1" o--> "0..*" RevisionITV : tiene
```

```
@enduml
```



En este diagrama, se han agregado dos nuevas de asociación: una de composición entre Coche que un Coche está compuesto por 4 Ruedas) y agregación entre Coche y RevisionITV (ya que un Coche puede tener varias RevisionITV). Además, se ha agregado la multiplicidad "1" en la flecha que va de Rueda a Coche, ya que una Rueda está presente en un solo Coche.

La relación entre las clases se establece mediante flechas que indican la dirección de la relación y un texto que especifica la multiplicidad de la relación. En este caso, un Empleado puede conducir

Introducción a la Ingeniería del Software

Actividad de modelado de Diagrama de Clases UML con ChatGPT

varios Coches (relación 1:N), por lo que se ha agregado la multiplicidad "1:N" en la flecha que va de Empleado a Coche. Un Coche, a su vez, está compuesto por 4 Ruedas (relación 1:4), por lo que se ha agregado el símbolo de composición "*-->" en la flecha que va de Coche a Rueda. Y un Coche puede tener varias RevisionITV (relación 1:N), por lo que se ha agregado el símbolo de agregación "o-->" en la flecha que va de Coche a RevisionITV.

Espero que esta versión mejorada del diagrama sea de utilidad para ti. Si necesitas más ayuda, ¡no dudes en preguntar!

Vemos que, en este punto, el diagrama no es semánticamente correcto. La cardinalidad entre Coche y Rueda sigue siendo errónea, y esa asociación debería ser de agregación, mientras que la asociación con RevisionITV debería ser de composición. Por tanto, seguiríamos indicando correcciones a ChatGPT hasta que veamos que es incapaz de mejorar el diagrama.