Práctica 2: Arquitectura de software – Parte 2

Objetivo: Representar la arquitectura de software de la Práctica 1 mediante un diagrama de clases UML, también se deben crear las historias de usuario y escalar esta arquitectura para añadir las nuevas funcionalidades en el diagrama UML y en el código de C#.

Descripción: Para unificar las diferentes arquitecturas desarrolladas por los alumnos en la Práctica 1, **se va a partir de la solución proporcionada por el profesor**. También será necesario añadir nuevas clases en el diagrama UML para cumplir con los nuevos requisitos. Posteriormente implementar la nueva arquitectura al código de C# con un criterio crítico sobre la implementación.

Requisitos (Parte 1):

1. Los policías pertenecen a una Comisaría:
    - Esta tiene una lista de coches de policía y puede registrar coches de policía mediante su matrícula.
    - Además, la comisaría tiene una alerta que puede activarse desde cualquier coche de policía cuando se detecta un vehículo por encima de la velocidad legal. El coche de policía que envíe la alarma proporcionará la matrícula del coche infractor.
    - Cuando la alerta se activa, notifica la matrícula del vehículo infractor a todos los coches de policías **que estén patrullando** para poder ir en su búsqueda.
2. El coche de policía va a poder perseguir un vehículo en concreto:
    - Deberá tener un atributo que indique si está persiguiendo un vehículo o no.
    - Podrá perseguir un vehículo cuando detecte a ese vehículo por encima de la velocidad legal o le notifiquen desde la comisaría de policía.
3. Va a existir una ciudad con una Comisaría de Policía, también se pueden registrar nuevas licencias de taxi y retirar licencias de taxi existentes.

Requisitos (Parte 2):

1. **Crear un repositorio en GitHub**, incluyendo el código de la práctica 1 y a partir de ahí, ir subiendo cada cambio al repositorio mediante ***commits*** como se ha visto en teoría. Incluir la **URL del repositorio en el documento** donde se incluya la arquitectura UML.
2. Tras desarrollar la arquitectura pedida, los alumnos a muy buen criterio han observado que el coche de policía y el radar no son un todo y las partes pueden vivir independientemente. Pero el código del profesor, que poco tiene de modular, no es capaz de plasmar esta relación.
    - Hacer que el coche de policía pueda tener o no un radar de velocidad, para esto se debe **gestionar los casos en los que el valor del radar sea NULL** en la clase del coche de policía.
3. Incluir en la arquitectura UML e implementar un patinete, este patinete es un vehículo, aunque en este caso el patinete **no tiene matrícula**. Prestar especial atención para no incumplir *Liskov Substitution Principle* de SOLID.
4. Incluir en la arquitectura UML e implementar la escritura de mensajes en las nuevas funcionalidades mediante la interfaz.

5. Añadir a la secuencia existente las nuevas casuísticas, cada caso debe representarse mediante un mensaje impreso por consola con la misma estructura seguida en la práctica 1:
    - La creación de la ciudad.
    - La creación de la comisaría de policía.
    - El registro de varios taxis en la ciudad.
    - El registro de varios coches de policía (algunos con radar y otro sin radar).
    - Intento de utilizar el radar en un coche de policía que no tiene radar.
    - Alerta a la comisaría de comenzar a perseguir un vehículo con cierta matrícula. Este vehículo debe haber sido pillado con el radar por encima de la velocidad legal.
    - Aviso de que los demás policías comienzan a perseguir ese vehículo (para esto debe haber varios coches de policía anteriormente patrullando).
    - Quitar la licencia a alguno de los taxis que haya sobrepasado la velocidad legal.
6. **Explicar brevemente** si se están aplicando los principios SOLID dentro del código y el porqué de esa implementación, **hacer las modificaciones necesarias en el código y en la arquitectura para que se cumplan los principios**. Redactar en el mismo documento donde se incluya la arquitectura UML.
7. Ahora queremos que el policía pueda tener diferentes aparatos de medida, que pueden ser un medidor de velocidad (Radar) o un medidos de alcohol (Alcoholímetro). **Al coche de policía solo se le puede asignar un único medidor,** y en el coche de policía únicamente va a haber un método para activar el aparato de medida. Con la arquitectura actual, ¿Qué principio SOLID incumpliríamos y cómo lo solucionarías? Redactar en el mismo documento donde se incluya la arquitectura UML, no será necesario implementarlo en el código.

Se anima a los alumnos a comparar la arquitectura proporcionada por el profesor con la desarrollada y discutir las diferentes soluciones de implementación.

Herramientas:

draw.io para hacer los diagramas UML online (https://app.diagrams.net/)