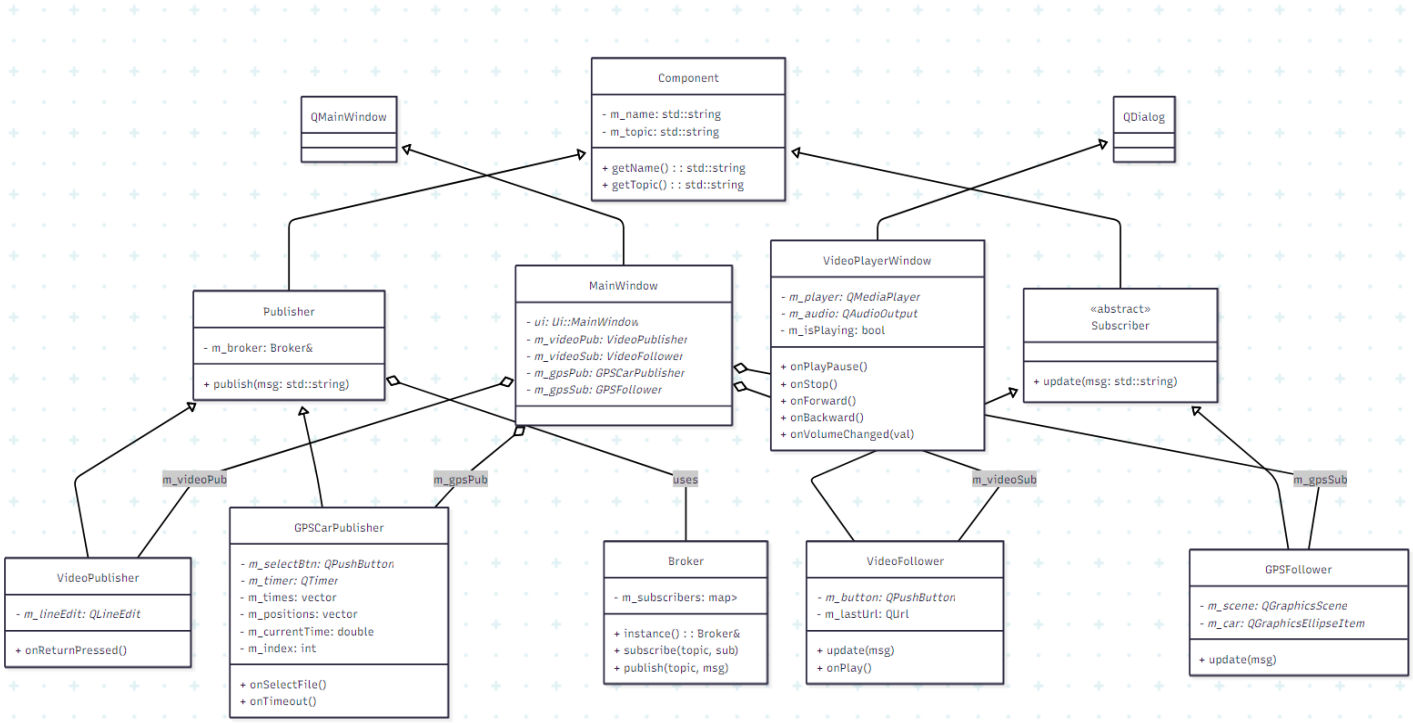
Documentación Tarea 3

- Diagrama UML:



- Dificultades:

1) Integración de múltiples módulos especializados en una misma arquitectura

Situación: En esta etapa, fue necesario integrar módulos con funcionalidades muy distintas como videopublisher, videofollower, gpscarpublisher y gpsfollower. La dificultad estuvo en hacer que todos estos actores trabajaran de forma coordinada bajo el mismo patrón Observer, sin que se generaran dependencias cruzadas o se rompiera el modularidad del sistema.

Aprendizaje: Tuvimos que profundizar en el uso del patrón Observer y entender cómo diseñar clases concretas (Publisher, Subscriber) que se comuniquen a través del Broker sin acoplamiento directo. También aprendimos a mantener una estructura flexible y reutilizable.

Solucion: Se extendieron las clases base para construir módulos especializados que mantuvieran su independencia. Gracias a esto, por ejemplo, VideoPublisher pudo emitir eventos que VideoFollower recibía sin conocerse directamente, todo mediante el Broker.

2) Conexión entre la interfaz gráfica Qt y la lógica del sistema

Situacion: Vincular la lógica del sistema con la interfaz gráfica Qt (mainwindow, videoplayerwindow) fue más complejo de lo esperado. A veces las señales no se conectaban correctamente, o los objetos no reaccionaban como debían al publicar o recibir mensajes.

Aprendizaje: Fue clave entender el sistema de señales y slots de Qt, y cómo integrarlo con clases C++ externas. Además, aprendimos a organizar la creación de objetos y la conexión entre vista y lógica para evitar errores de tiempo de ejecución.

Solucion: Se organizó la inicialización en mainwindow.cpp de forma clara y ordenada. Se encapsuló la lógica del Observer dentro de funciones bien definidas, y se conectaron correctamente a los eventos de la interfaz, separando la vista del backend.

3) Sincronización de datos entre módulos activos (GPS y Video)

Situacion: Los módulos como GpsCarPublisher y VideoPublisher generan información en tiempo real, lo que provocó desincronización entre lo que se publicaba y lo que los Subscribers mostraban. Esto causaba que, por ejemplo, un seguidor mostrara una posición errónea o un video desfasado.

Aprendizaje: Aprendimos que al trabajar con eventos en paralelo o datos que cambian constantemente, es esencial definir estructuras de datos claras y validar la información antes de reaccionar. También entendimos que cada módulo debe tener responsabilidad limitada y bien definida.

Solucion: Se establecieron formatos estándar de mensaje y validaciones en los Subscribers para actuar solo con información relevante. Además, se probaron los módulos por separado antes de integrarlos, logrando un sistema más robusto y sincronizado.