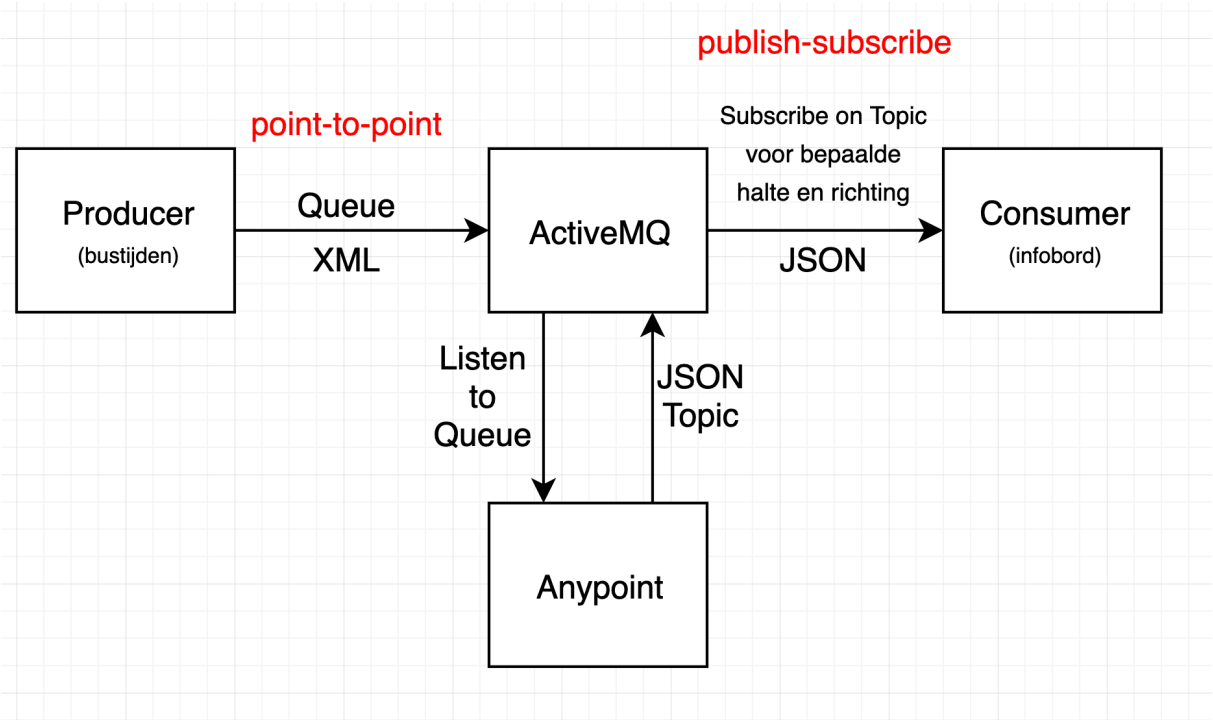
Opdracht 2 - Queue patterns (5pt.)



Patterns:
Message Translator: Van XML naar JSON
Splitter: ETAs into ETA messages

Producer stuurt allemaal berichten in XML formaat naar een queue op ActiveMQ. De flow in Anypoint heeft een listener die luistert naar deze queue, zodra en er een bericht op komt gaat dit door zijn flow. Er wordt data gesplitst en geselecteerd voor in een JSON object, deze JSON wordt dan op een topic gezet. De consumer (of het infobord) kijkt of het bericht voor hem bedoeld is door naar de User Properties te kijken van het bericht, als dit overeenkomt met zijn halte en richting pakt hij het bericht op van de topic en wordt de informatie weergegeven op zijn infobord.

Opdracht 3 - Queue patterns (5pt.)
Waar een bericht bezorgd moet worden kan point to point worden gebruikt. Bankautomaten moeten bij een transactie deze door kunnen geven aan de bank, ook al ligt er bij de bank een systeem eruit. Deze transacties kunnen dan op een queue via het point to point pattern.

Publish Subscribe is handig wanneer een bericht alleen realtime relevant is – deze heeft geen persistance nodig. Een voorbeeld zou een streamingsdienst zijn zoals Twitch, waar gebruikers zich dan subscriben voor de frames van een bepaalde stream. Deze frames hoeven alleen maar real time te worden afgeleverd, gebruikers hebben niks aan oude frames. Als een subscriber dus even offline gaat kan de stream daarna hervat worden met de nieuwe informatie.

Opdracht 4 - Queues en loadbalancing (5pt.)

Evenredig uitdelen van requests op webservers is een manier om loadbalancing te realiseren. Round robin doet dit, en is default bij Queues als er meerdere subscribers zijn. Als webservers dus achter een Queue hangen en alle requests komen binnen op die Queue, worden die dus automatisch round robin verdeeld over de webservers. Dit noemt de website Message Dispatcher pattern.

Een specifiek probleem zou kunnen zijn een veelbezochte website met meerdere webservers om requests af te kunnen handelen.

Deze manier van verdelen, Round Robin, werkt alleen als de taken die worden uitgedeeld evenveel werk zijn.

Vragen bij opdracht 2

Opdracht 2 (5pt).

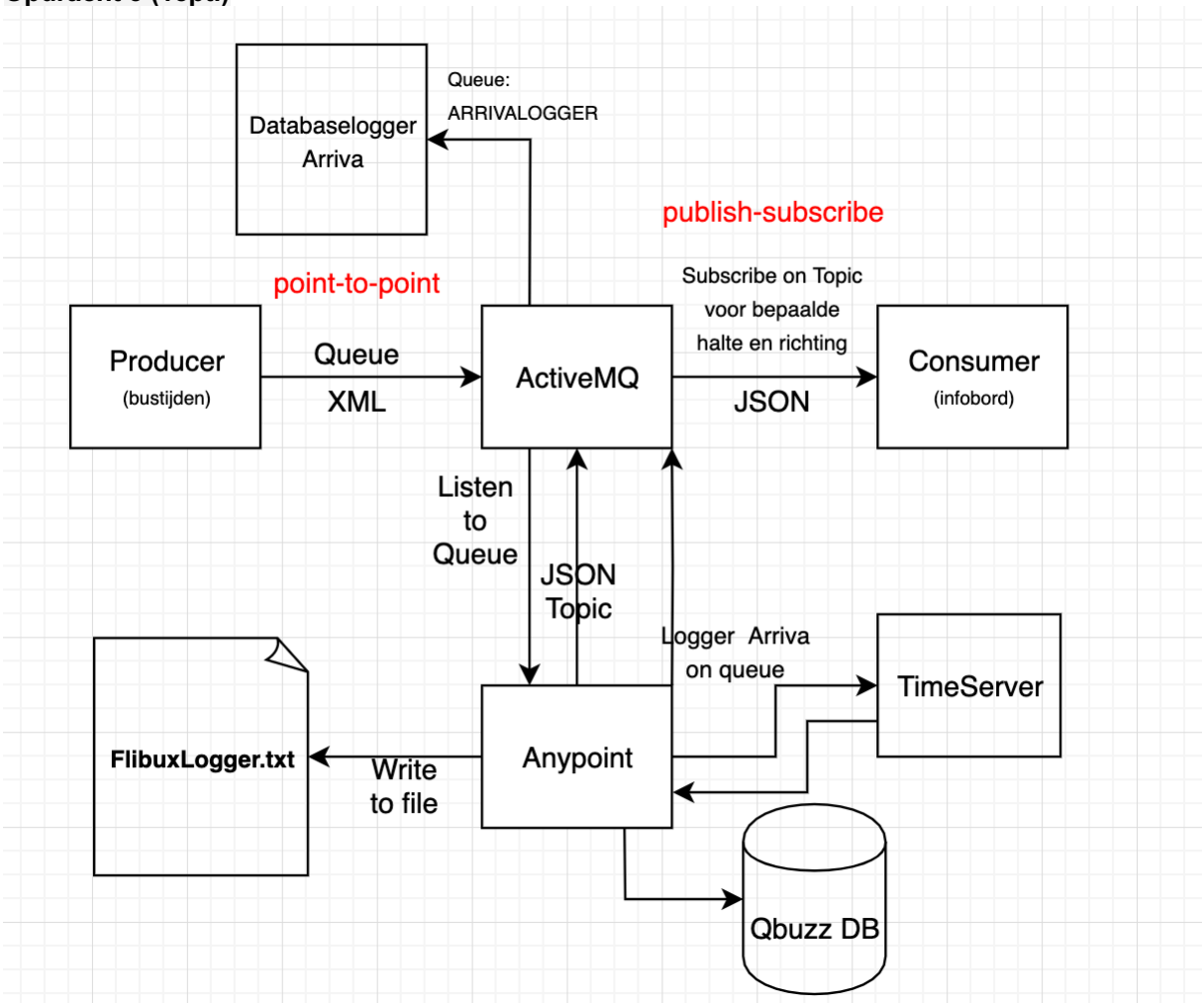
Listener

Consume

Opdracht 3 (10pt).

Vragen bij opdracht 3

Opdracht 3 (15pt.)



Dit schema is een uitbreiding van het schema uit opdracht 1. De producer maakt nog steeds de bustijden aan. Er is een timeserver aan toegevoegd. Waar Anypoint een request naar toestuurt om de huidige tijd te krijgen sinds dat de timeserver is aangezet. Deze kan vanuit de bussimulator applicatie worden opgehaald met een get request.

Naast dat Anypoint informatie haalt uit de berichten om de infoborden te updaten, worden er ook logs aangemaakt.

Voor Arriva wordt hiervoor een json-object aangemaakt en op een queue, ARRIVALOGGER, gezet. Deze haalt de Arrivalogger dan weer van de queue af in batches.

Voor Flixbus worden de logs naar een bestand geschreven.

Voor Qbuzz worden de logs opgeslagen in de database.

Om deze verschillende manier van logs te waarborgen wordt het pattern **Splitter** toegepast. Waarbij de gegevens worden gesplit en op de manier hoe de verschillende bedrijven dit aangeleverd willen krijgen verwerkt.

Voor Qbuzz is het pattern **Shared Database** toegepast zodat Anypoint kan schrijven en qbuzz hierbij kan voor de logs.