Fedex Aggregation API

The Fedex Aggregation API (or fedex-aggregation-api) was developed with the use of Maven, Spring Boot, WebFlux 3.3.0 and Docker.

Provided the infrastructure and the backend-services running in the background, I have initially developed a WebClient Service to communicate with the singular Pricing, Track and Shipments APIs.

Secondly, I have decided of using a Blocking Deque for storing the Pricing, Track and Shipments APIs abstraction.

Although the backend-services Pricing, Track and Shipments APIs provide different responses datatypes, the overall logic and components of the application can be generalised to a super class.
So, for simplicity, here and in the graph below you will see API, APIBlockingDeque<API> and APIService (and not, for example, PricingAPIBlockingDeque<PricingAPI>).

Also, while I do provide an interaction graph below, I have not drawn aggregation APIs with mixed requests (for example pricing and shipments together) as eventually their singular logic is described in this generalised design.
Eventually, for these cases, the AggregationController is responsible of grouping the responses datatypes into a Mono object and answer it to the original request.

The logic juice is within the QueueHandler class and how it deals with requests which have not reached the 5 parameters cap yet.
When a new request comes in, the QueueHandler loops every item of the APIBlockingDeque and counts the parameters.

If the initial request plus the APIBlockingDeque APIs parameters do not reach exactly 5 params or the count goes above 5, the new request is put in the queue with status DISABLED.
The request in this case will wait either for 5 seconds to pass by or for the status to be set to ACTIVE, and then it is performed to the backend-service.

If the initial request plus the APIBlockingDeque APIs parameters do reach exactly 5 params, while the initial request is directly performed to the backend-service, all the APIs which helped in reaching the cap are set to status ACTIVE, activating their threads and allowing them to perform their apis to the backend-services.