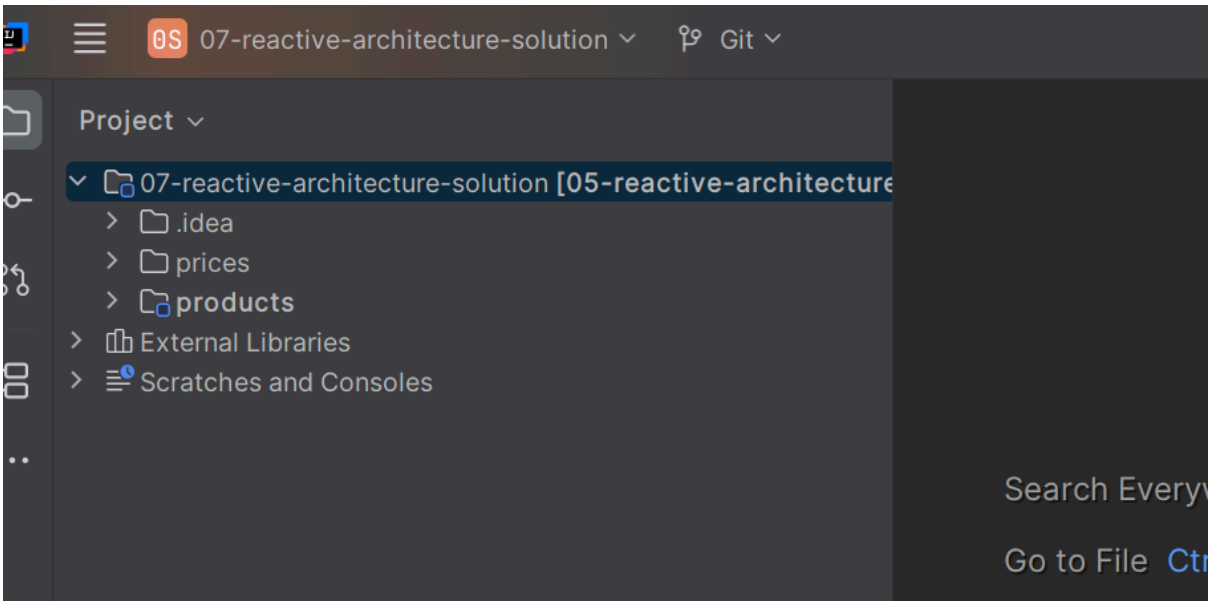
LAB 10: QUARKUS REACTIVE ARCHITECTURE

Autor: José Díaz

Github Repo: https://github.com/joedayz/quarkus-bcp-2025.git

1. Abre el proyecto **reactive-architecture-start**.



Instructions

Consider a `products` service that exposes a REST API. Each request to the `products` API uses a worker thread. If the `products` service receives more requests than what this thread can handle, then the application starts queuing requests.

> **Note**
> For the demonstration purposes of this exercise, the `products` service is configured to use only one worker thread and only one event-loop thread.

The `GET /product/{id}/priceHistory` endpoint of the `products` service depends on the `prices` service to gather product price historical data. Gathering historical data is a costly process, so the `prices` service takes about two seconds to serve a request. Therefore, each request to `/product/{id}/priceHistory` takes at least two seconds and keeps the worker thread blocked, waiting for the response from the `prices` service.

Use RESTEasy Reactive and REST Client Reactive in the `products` service to mitigate this problem.

The `lab start` script runs the `prices` service for you. You do not need to modify this service.

1. Run the `products` service and verify that the two threads do not have enough capacity to handle the slow I/O.

    1.1. Navigate to the `~/D0378/reactive-architecture` directory and open the project with an editor of your choice.

```
[student@workstation ~]$ cd ~/DO378/reactive-architecture
[student@workstation reactive-architecture]$ codium .
```

1.2. Start the `products` service in development mode.

```
[student@workstation reactive-architecture]$ mvn quarkus:dev
...output omitted...
INFO  [io.quarkus] ... Listening on: http://localhost:8080
...output omitted...
```

1.3. Open a terminal window and send a request to `http://localhost:8080/product/1/priceHistory`. Verify that the request takes about two seconds to finish.

```
[student@workstation ~]$ time curl http://localhost:8080/products/1/priceHistory
{"prices":[...output omitted...],"product_id":1}
real 0m2.247s
user 0m0.003s
sys 0m0.005s
```

1.4. Run the `benchmark.sh` script. This script sends 10 requests to `http://localhost:8080/products/1/priceHistory` in one second but takes more than 20 seconds to receive all the responses.

```
[student@workstation ~]$ time ~/DO378/reactive-architecture/benchmark.sh
...output omitted...
Sending request...

real 0m20.187s
user 0m0.050s
sys 0m0.036s
```

1.5. Inspect the application logs. Verify that the `executor-thread-0` worker thread serves the requests one after another, taking about two seconds for each request.

```
INFO  [io.qua.htt.access-log] (executor-thread-0) ... 26/Jan/2023:14:08:28 ...
INFO  [io.qua.htt.access-log] (executor-thread-0) ... 26/Jan/2023:14:08:30 ...
INFO  [io.qua.htt.access-log] (executor-thread-0) ... 26/Jan/2023:14:08:32 ...
```

2. Add the RESTEasy and Rest Client Reactive extensions.

2.1. Replace the `quarkus-resteasy`, `quarkus-resteasy-jackson`, `quarkus-rest-client`, and `quarkus-rest-client-jackson` extensions with their reactive counterparts.

```
[student@workstation reactive-architecture]$ mvn quarkus:remove-extensions \
 -Dextensions="resteasy,resteasy-jackson,rest-client,rest-client-jackson"
 ...output omitted...
... Extension io.quarkus:quarkus-resteasy has been uninstalled
... Extension io.quarkus:quarkus-resteasy-jackson has been uninstalled
```

```
... Extension io.quarkus:quarkus-rest-client has been uninstalled
... Extension io.quarkus:quarkus-rest-client-jackson has been uninstalled
...output omitted...

[student@workstation reactive-architecture]$ mvn quarkus:add-extensions \
 -Dextensions="resteasy-reactive,resteasy-reactive-jackson, \
 rest-client-reactive,rest-client-reactive-jackson"
 ...output omitted...
... Extension io.quarkus:quarkus-resteasy-reactive has been installed
... Extension io.quarkus:quarkus-resteasy-reactive-jackson has been installed
... Extension io.quarkus:quarkus-rest-client-reactive has been installed
... Extension io.quarkus:quarkus-rest-client-reactive-jackson has been installed
...output omitted...
```

3. Force the `/product/{id}/priceHistory` endpoint to be a non-blocking operation.

3.1. In the `ProductsResource.java` file, use the `@NonBlocking` annotation in the `/product/{productId}/priceHistory` endpoint.

```
@GET
@NonBlocking
@Path( "/{productId}/priceHistory" )
public ProductPriceHistory getProductPriceHistory(
    @PathParam( "productId" ) final Long productId ) {
    return pricesService.getProductPriceHistory( productId );
}
```

3.2. Rerun the curl request. The request returns an error because you are trying to execute the request to the `prices` service, which is blocking, inside a non-blocking operation.

```
[student@workstation ~]$ curl http://localhost:8080/products/1/priceHistory | jq
{"details":"Error
 id ..., org.jboss.resteasy.reactive.common.core.BlockingNotAllowedException: ...}
```

4. Modify the `PricesService` rest client to be non-blocking.

4.1. Modify the `PricesService#getProductPriceHistory` method to return a Uni stream.

```
@GET
@Path( "/history/{productId}" )
Uni<ProductPriceHistory> getProductPriceHistory(
    @PathParam( "productId" ) final Long productId
);
```

4.2. Return to the `ProductsResource.java` file, and modify the `/product/{productId}/priceHistory` endpoint method to return a Uni Stream.

```
@GET
@NonBlocking
@Path( "/{productId}/priceHistory" )
public Uni<ProductPriceHistory> getProductPriceHistory(
    @PathParam( "productId" ) final Long productId ) {
    return pricesService.getProductPriceHistory( productId );
}
```

4.3. Stop the application and rerun `mvn quarkus:dev`.

> **Note**
> The development mode might not apply the thread configuration for Reactive REST if you do not restart the application.

4.4. Rerun the request and verify that the asynchronous endpoint returns a valid response.

```
[student@workstation ~]$ curl http://localhost:8080/products/1/priceHistory | jq
{"prices":...output omitted...],"product_id":1}
```

4.5. Inspect the application logs. Verify that the `vert.x-eventloop-thread-0` event-loop thread serves the request.

```
INFO  [io.qua.htt.access-log] (vert.x-eventloop-thread-0) ...
```

4.6. Run the benchmark script and verify that the application serves the requests in less time.

```
[student@workstation ~]$ time ~/D0378/reactive-architecture/benchmark.sh
...output omitted...
Sending request...

real 0m3.119s
user 0m0.042s
sys 0m0.042s
```

The non-blocking response time to process 10 requests is about 6 times faster than using a blocking strategy.

4.7. Inspect the application logs to verify that the `vert.x-eventloop-thread-0` thread serves all the requests.

```
...output omitted...
INFO [io...access-log] (vert.x-eventloop-thread-0) ... 26/Jan/2023:14:15:19 ...
INFO [io...access-log] (vert.x-eventloop-thread-0) ... 26/Jan/2023:14:15:19 ...
INFO [io...access-log] (vert.x-eventloop-thread-0) ... 26/Jan/2023:14:15:20 ...
INFO [io...access-log] (vert.x-eventloop-thread-0) ... 26/Jan/2023:14:15:20 ...
...output omitted...
```

5. Verify that RESTEasy Reactive treats methods that return a stream as non-blocking operations.

5.1. Remove the `@NonBlocking` annotation from the `/product/{productId}/priceHistory` endpoint.

```
@GET
@Path( "/{productId}/priceHistory" )
public Uni<ProductPriceHistory> getProductPriceHistory(
    @PathParam( "productId" ) final Long productId ) {
    return pricesService.getProductPriceHistory( productId );
}
```

5.2. Rerun the request and verify that the asynchronous endpoint returns a valid response.

```
[student@workstation ~]$ curl http://localhost:8080/products/1/priceHistory | jq
{"prices":...output omitted...],"product_id":1}
```

5.3. Inspect the application logs. Verify that an event-loop thread serves the request.

```
INFO  [io.qua.htt.access-log] (vert.x-eventloop-thread-0) ...
```

6. Block the event loop.

6.1. Send a request to the `/products/blocking` endpoint. This operation, although executed as asynchronous, blocks the event loop for 30 seconds.

```
[student@workstation ~]$ curl http://localhost:8080/products/blocking; echo
```

You might see a `io.vertx.core.VertxException: Thread blocked` error in the application logs.

6.2. While you are waiting for the blocking endpoint to respond, open a new terminal and run the benchmark script. Verify that the benchmark is slower now, because the event-loop thread is blocked.

```
[student@workstation ~]$ time ~/DO378/reactive-architecture/benchmark.sh
...output omitted...
Sending request...

real 0m23.145s
user 0m0.042s
sys 0m0.042s
```

Note
To see the effects of blocking the event-loop thread, make sure that you run the benchmark while the thread is blocked. You can send more requests to the blocking endpoint if you need to block the thread again.

7. Annotate the `/products/blocking` endpoint as `@Blocking` and rerun the benchmark.

7.1. Add the `@Blocking` annotation to the `/products/blocking` endpoint.

```
@GET
@Blocking
@Path( "/blocking" )
public Uni<String> blocking() {
    ...implementation omitted...
}
```

7.2. Send a request to the `/blocking` endpoint. This operation is now offloaded to a worker thread.

```
[student@workstation ~]$ curl http://localhost:8080/products/blocking; echo
```

7.3. While you are waiting for the blocking endpoint to respond, run the benchmark script. Verify that the benchmark is faster now.

```
[student@workstation ~]$ time ~/D0378/reactive-architecture/benchmark.sh
...output omitted...
Sending request...

real 0m3.164
user 0m0.037s
sys 0m0.043s
```

7.4. Inspect the application logs. Verify that the blocking endpoint runs on a worker thread.

```
INFO  [io.qua.htt.access-log] (vert.x-eventloop-thread-0) ... "GET /products/1/
priceHistory HTTP/1.1" 200 6741
INFO  [io.qua.htt.access-log] (vert.x-eventloop-thread-0) ... "GET /products/1/
priceHistory HTTP/1.1" 200 6741
INFO  [io.qua.htt.access-log] (executor-thread-0) ... "GET /products/blocking
 HTTP/1.1" 200 25
```

8. Press q to stop the `products` application.

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish reactive-architecture
```

This concludes the section.

Solución:

enjoy!

Jose