LAB 13: QUARKUS REACTIVE REVIEW

Autor: José Díaz

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

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

1. Open the application located in the `~/D0378/reactive-review` directory with an editor, such as VSCodium or vim.

1.1. Navigate to the `~/D0378/reactive-review` directory.

```
[student@workstation ~]$ cd ~/D0378/reactive-review
```

1.2. Open the project with an editor, such as VSCodium or vim.

```
[student@workstation reactive-review]$ codium .
```

2. Add the required dependencies to create a reactive endpoint that stores data in a PostgreSQL database, and sends events to Apache Kafka.

3. Configure the application to use four channels.

- An incoming channel that consumes `SpeakerWasCreated` events from the `new-speakers-in` channel. Use the `speaker-was-created` Kafka topic to consume events. Set the `offset.reset` property of the incoming channel to `earliest`, and deserialize the incoming messages with the `com.redhat.training.serde.SpeakerWasCreatedDeserializer` class.

- An outgoing channel that publishes `SpeakerWasCreated` events to the `new-speakers-out` channel. Use the `speaker-was-created` Kafka topic to publish events.
- An outgoing channel that publishes `EmployeeSignedUp` events to the `employees-out` channel . Use the `employees-signed-up` Kafka topic to publish events.
- An outgoing channel that publishes `UpstreamMemberSignedUp` events to the `upstream-members-out` channel. Use the `upstream-members-signed-up` Kafka topic to publish events.

4. Create a reactive POST endpoint with the following requisites:
   - Receives a `Speaker` object as payload.
   - Stores the payload in the database by using a transaction.
   - Sends a `SpeakerWasCreated` event to the `new-speakers-out` channel.
   - Returns a `201` HTTP response that includes in the `location` response header the URI of the inserted element. The URI must follow the `/speakers/{id}` pattern.
5. Create an event processor that consumes and filters `SpeakerWasCreated` events.
   - If the affiliation is `RED_HAT`, then send a `EmployeeSignedUp` event to the `employees-out` channel.
   - If the affiliation is `GNOME_FOUNDATION`, then send a `UpstreamMemberSignedUp` event to the `upstream-members-out` channel.
   - Always acknowledge the message.
6. Execute the tests to validate the code changes. Optionally, you can use the Swagger UI to manually validate the changes before executing the tests.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation reactive-review]$ lab grade reactive-review
```

Finish

enjoy!

Jose