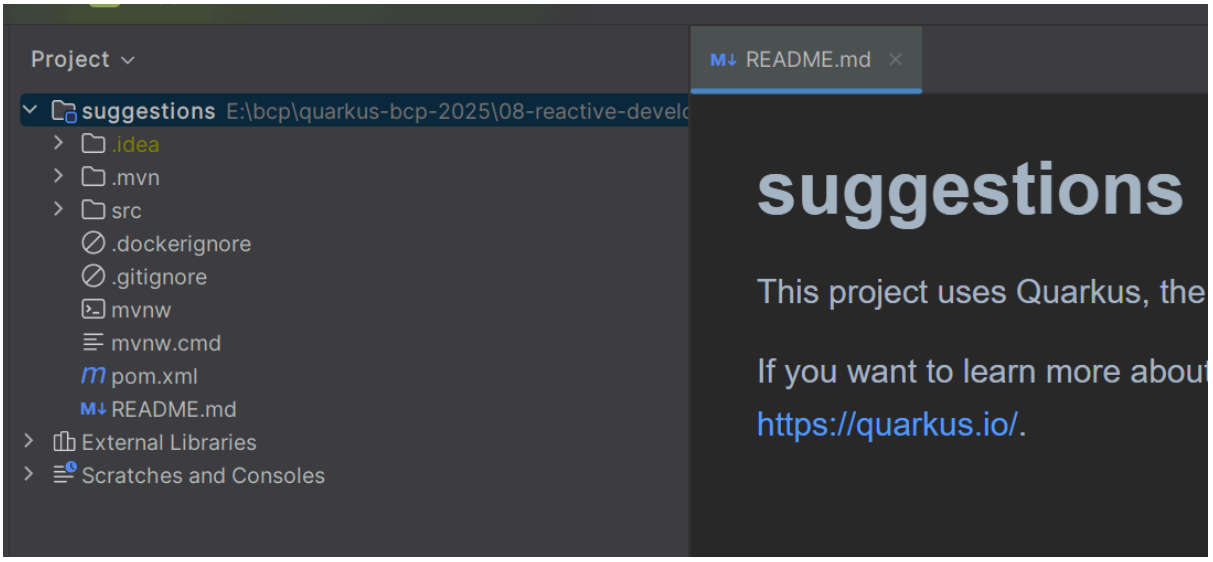
LAB 11: QUARKUS REACTIVE DEVELOP

Autor: José Díaz

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

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



Instructions

You are going to create a reactive API that implements an e-commerce purchase suggestions system. Store these suggestions in a reactive database to increase performance and reduce latency.

The project skeleton contains the tests to verify the applications' behavior.

1. Set up the project for reactive development.

    1.1. In a terminal window, navigate to the project directory.

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

    1.2. Add the `hibernate-reactive-panache` and `reactive-pg-client` extensions to the project.

```
[student@workstation reactive-develop]$ mvn quarkus:add-extension \
 -Dextension="hibernate-reactive-panache,reactive-pg-client"
...output omitted...
[INFO] [SUCCESS] ... Extension io.quarkus:quarkus-hibernate-reactive-panache has
 been installed
[INFO] [SUCCESS] ... Extension io.quarkus:quarkus-reactive-pg-client has been
 installed
...output omitted...
```

    1.3. Open the project with your editor, for example, by using VSCodium.

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

2. Review the code to use.

    2.1. Open the entity `Suggestion` to check that it is a regular JPA Entity and has a constructor to initialize the values.

    2.2. Open the `SuggestionResourceTest` class and review the existing tests for the API methods to build.

3. Start the Continuous Testing mode.

    3.1. Open a new terminal. In VSCodium you can do this by clicking Terminal > New Terminal.

    3.2. Start Quarkus in the Continuous Testing mode and verify that the tests are failing.

```
[student@workstation reactive-develop]$ mvn quarkus:test
...output omitted...
3 tests failed (0 passing, 0 skipped)...
```

4. Add the create suggestion endpoint.

    4.1. Add the create suggestion endpoint in the `SuggestionResource` class. This method must use the `PanacheEntity.persist()` method wrapped in a call to `Panache.withTransaction`.

```
@POST
public Uni<Suggestion> create( Suggestion newSuggestion ) {
    return Panache.withTransaction( newSuggestion::persist );
}
```

    4.2. Verify that after saving the file the new method makes its test pass.

```
...output omitted...
2 tests failed (1 passing, 0 skipped)
```

5. Add an endpoint to retrieve a suggestion by ID.

    5.1. Add the get method in the `SuggestionResource` class, by using the `PanacheEntity.findById()` method. Use the parameter to find the entity in the database.

```
@GET
@Path( "/{id}" )
public Uni<Suggestion> get( Long id ) {
    return Suggestion.findById( id );
}
```

    5.2. Save and verify that only one test is failing.

```
...output omitted...
1 tests failed (2 passing, 0 skipped)
```

6. Add a method to list all suggestions.

    6.1. Add a list method to the `SuggestionResource` class by using the GET HTTP method in the root path. Use the `PanacheEntity.streamAll()` method to return the values as they are available.

```
@GET
public Multi<Suggestion> list() {
    return Suggestion.streamAll();
}
```

    6.2. After saving the file, verify that all tests pass.

```
...output omitted...
All 3 tests are passing (0 skipped)
```

    6.3. Stop the Continuous Testing mode in the terminal by pressing `q`.

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-develop
```

This concludes the section.

enjoy!

Jose