Object Oriented Architectures and Secure Development

06-01 Vert.x Futures and promises

This lab continues with the student administration application.

From now on, the courses are available on an external api.

The goal of this lab is to consume a Web API through Vert.X with Promises and Futures.

All the new code must be unit tested as well.

Before you begin

Publish all your solutions in your (private) group on gitlab: `https://git.ti.howest.be/TI/2022-2023/s3/object-oriented-architectures-and-secure-development/students`

Create a new project/repository for each exercise with a different theme, reuse the existing project if the exercise has the same theme as a previous exercise.

0 Add vertx to your project

- Add the following dependencies:
  - implementation "io.vertx:vertx-core:4.3.4"
  - implementation "io.vertx:vertx-web-client:4.3.4"
- Add the following **test** dependencies
  - testImplementation "io.vertx:vertx-junit5:$vertxVersion"
  - testImplementation "io.vertx:vertx-web-client:$vertxVersion"
  - testImplementation 'com.squareup.okhttp3:mockwebserver:4.10.0'

1 Create an async repository.

- Create a new Interface for this async repository.
- Make sure the function getCourses() returns **Future**
- Use the **Vert.X WebClient** to consume
  - https://courses-ooasd.herokuapp.com/

2 Let the service layer use this new async repository along with the current repositories

3 The controller uses the new async getProducts function to load the available courses.

4 Saving or modifying courses isn't necessary anymore. (clean up the code)

5 Unit test all the new code

- Remember, unit tests must be fast. Using the real Web API with courses isn't an option.
  - Spin a mockWebServer to mock the external Web API.
- The service layer shouldn't use the real AsyncRepository.
  - Spin a mockWebServer or mock the AsyncRepository