HW1: Mid-term assignment report

David Cobileac [102409], v2024-04-11

1 Introduction

1.1 Overview of the work

This report presents the midterm individual project required for TQS, covering both the software product features and the adopted quality assurance strategy.
BusTicket is a simple web app that offers its users a way to reserve tickets for bus trips.

1.2 Current limitations

Currently the api has only the basic essential functionality for the app to work and the cache lacks an api. Limited api testing, more error checking tests should have been made.

2 Product specification

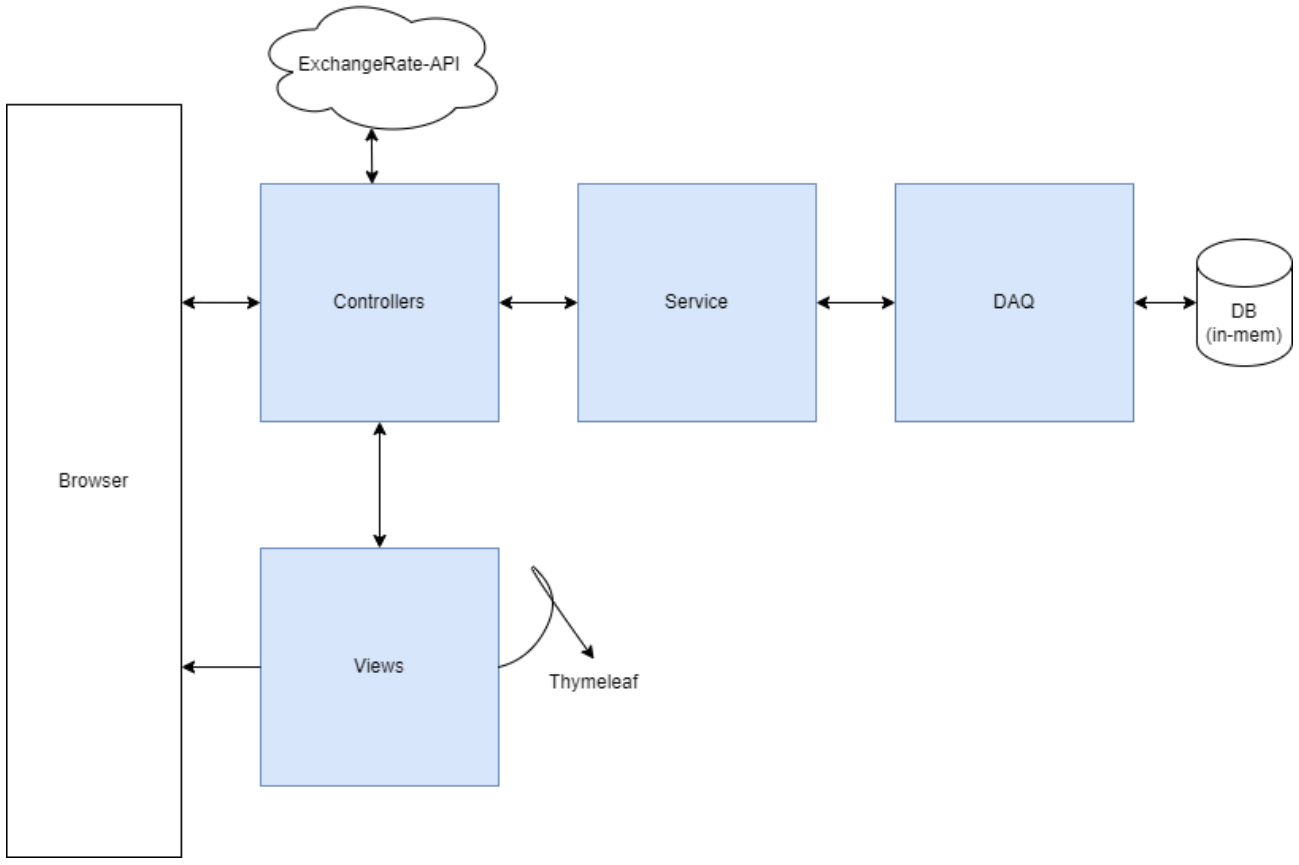
2.1 Functional scope and supported interactions

There is really only one actor for this application, whoever wants to purchase a bus ticket between two cities. The main usage scenario is as follows:

1. User opens the app, selects the departure city, the destination city, the date of the trip and the preferred currency.
2. Then a page is prompted with a list of the available trips for that date.
3. User clicks/selects the desired trip.
4. A page is prompted with the trip information as well as a form for reservation that the user has to fill.
5. Upon submitting the form a confirmation message is displayed.

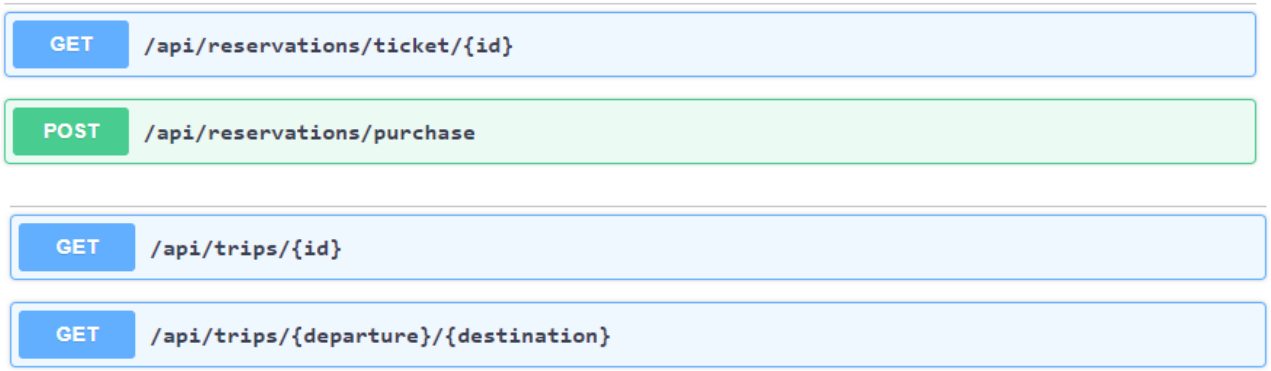
2.2 System architecture

App was developed in spring boot with thymeleaf as the template engine for serving html at the view layer



2.3 API for developers

Currently there is only an api for trips and tickets. An api for the cache system should have been implemented.



3 Quality assurance

3.1 Overall strategy for testing

Starting the development of the app I firstly designed the data model (Trip, Ticket) and made a very simple front and backend to understand the concept of the project (without communicating with the exchange rate api), then I moved on integrating the external api and writing the cache. After having a somewhat solid application I started writing tests, testing the cache, controllers and repositories. I left the web view testing with Selenium for last as well as SonarCloud for static code analysis.

3.2 Unit and integration testing

I wrote these unit tests for testing the exchange rate cache, checks on the cache validity, obligatory cache hits and misses.

```java
@Test
void whenAskedForValidExchangeRate_thenReturnRate() {
    double rate = currencyService.GetExchangeRateFromDollar("EUR");
    assertTrue(rate > 0);
}

@Test
void whenAskedForNotCachedExchangeRate_thenDontReturnRateFromCache() {
    double rate = currencyService.GetExchangeRateFromDollar("EUR");
    Mockito.verify(currencyService, Mockito.never()).GetCodeFromCache("EUR");
    assertTrue(rate > 0);
}

@Test
void whenAskedForExpiredCachedExchangeRate_thenDontReturnRateFromCache() throws Exception {
    double rate = currencyService.GetExchangeRateFromDollar("EUR");
    rate = currencyService.GetExchangeRateFromDollar("EUR"); // should fetch from cache
    Mockito.verify(currencyService, Mockito.times(1)).GetCodeFromCache("EUR");

    Thread.sleep(6000);

    rate = currencyService.GetExchangeRateFromDollar("EUR");

    Mockito.verify(currencyService, Mockito.times(1)).GetCodeFromCache("EUR");
    assertTrue(rate > 0);
}

@Test
void whenAskedForCachedExchangeRate_thenReturnRateFromCache() throws Exception {
    double rate = currencyService.GetExchangeRateFromDollar("EUR");
    rate = currencyService.GetExchangeRateFromDollar("EUR"); // should fetch from cache
    Mockito.verify(currencyService, Mockito.times(1)).GetCodeFromCache("EUR");
    assertTrue(rate > 0);
}
```

Also tested the services, controllers and repositories

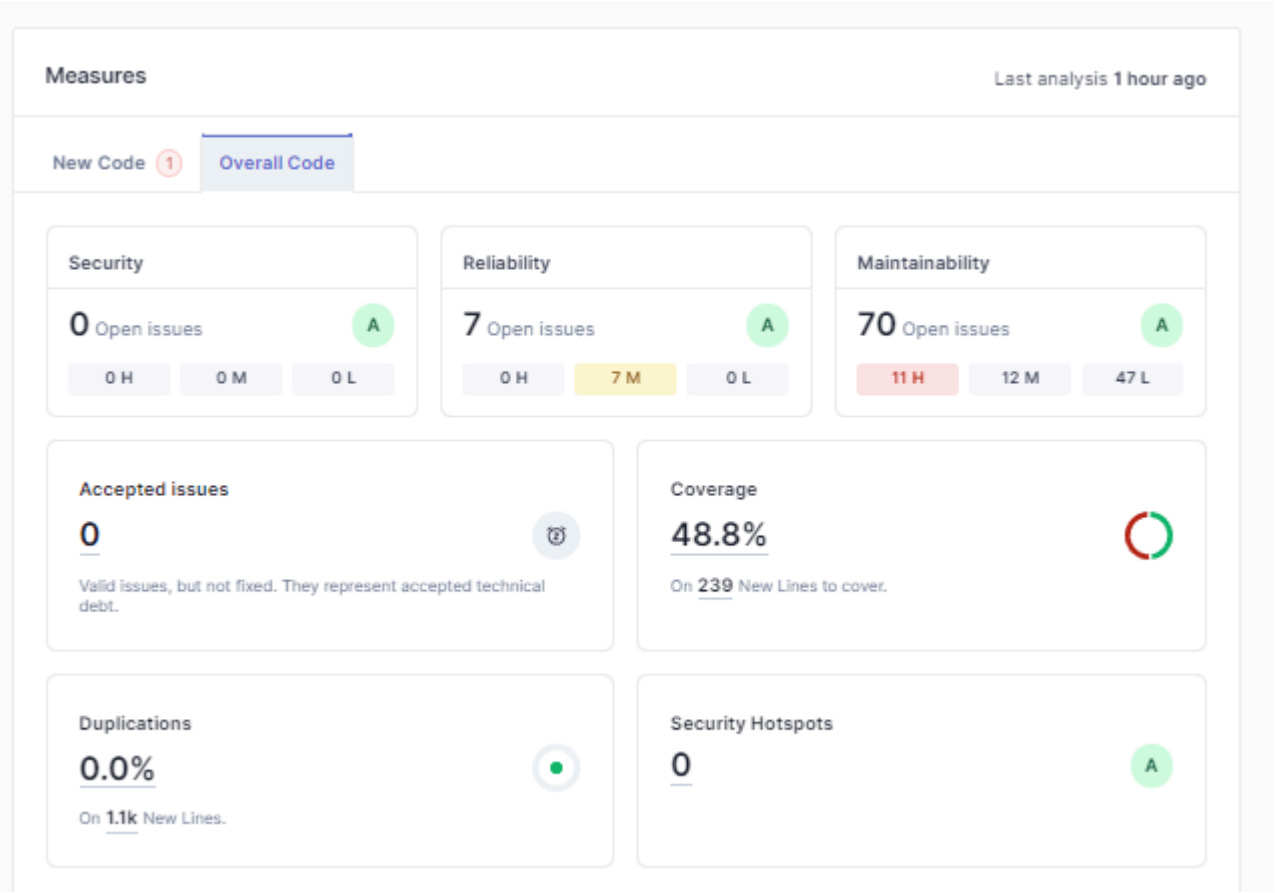
3.3 Functional testing

For functional testing I used selenium, selenium plugin to record and export the test.

```java
@Test
public void test(ChromeDriver driver) throws Exception {
driver.get("http://localhost:8080/");
driver.manage().window().setSize(new Dimension(1796, 1080));
driver.findElement(By.name("fromPort")).click();
{
  WebElement dropdown = driver.findElement(By.name("fromPort"));
  dropdown.findElement(By.xpath("//option[. = 'Chicago']")).click();
}
driver.findElement(By.cssSelector(".mt-5")).click();
//driver.findElement(By.name("toPort")).click();
{
  WebElement selectElement = driver.findElement(By.xpath("//select[@name='toPort']"));
  Select select = new Select(selectElement);

  select.selectByVisibleText("Miami");
}
driver.findElement(By.cssSelector(".mt-5")).click();

WebElement date = driver.findElement(By.name("tripDate"));
//date.clear();
js = (JavascriptExecutor) driver;
js.executeScript("document.getElementById('date').value = '2024-04-15'");
//date.sendKeys("15-04-2024");

driver.findElement(By.name("currency")).click();
{
  WebElement dropdown = driver.findElement(By.name("currency"));
  dropdown.findElement(By.xpath("//option[. = 'EUR']")).click();
}
driver.findElement(By.cssSelector(".btn")).click();
driver.findElement(By.linkText("Provide Information")).click();
driver.findElement(By.id("name")).click();
driver.findElement(By.id("name")).sendKeys("asdf");
driver.findElement(By.id("address")).click();
driver.findElement(By.id("address")).sendKeys("asdf");
driver.findElement(By.id("city")).click();
driver.findElement(By.id("city")).sendKeys("asdf");
driver.findElement(By.id("state")).click();
driver.findElement(By.id("state")).sendKeys("asdf");
driver.findElement(By.id("zipCode")).click();
driver.findElement(By.id("zipCode")).sendKeys("asdf");
driver.findElement(By.id("cardType")).click();
driver.findElement(By.id("cardType")).sendKeys("asdf");
driver.findElement(By.id("cardNumber")).click();
driver.findElement(By.id("cardNumber")).sendKeys("asdf");
driver.findElement(By.id("month")).click();
driver.findElement(By.id("month")).sendKeys("asdf");
driver.findElement(By.id("year")).click();
driver.findElement(By.id("year")).sendKeys("asdf");
driver.findElement(By.id("nameOnCard")).click();
driver.findElement(By.id("nameOnCard")).sendKeys("asdf");
driver.findElement(By.cssSelector(".btn")).click();
assertThat(driver.findElement(By.cssSelector(".mt-5")).getText()).contains("Thank you for trusting us!");
}
```

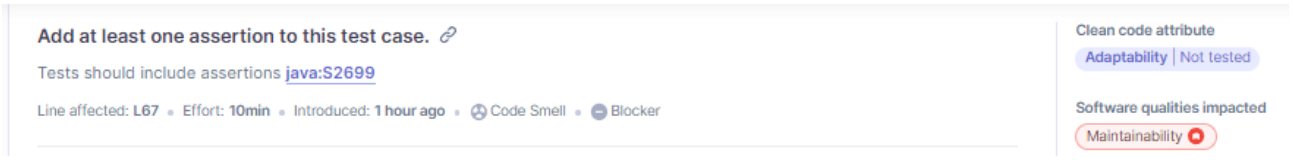
3.4 Code quality analysis

SonarQube was used for the static code analysis, here are the results:



Coverage could be improved.

There was actually one issue flagged that was pretty useful, I didn't have an assert on the selenium test, I forgot to test for the confirmation page.



3.5 Continuous integration pipeline [optional]

4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/cobileacd/TQS_102409
Video demo	https://github.com/cobileacd/TQS_102409 (under HW1)

Reference materials

Exchange rate api used for the project: https://www.exchangerate-api.com/docs/free