پیاده‌سازی پروژه با زبان برنامه نویسی java

برای پیاده‌سازی وبسایت خواسته شده با زبان java از فریمورک Spring استفاده شده است. وبسایت به صورت REST API پیاده‌سازی شده و endpoint های آن بدین گونه است:

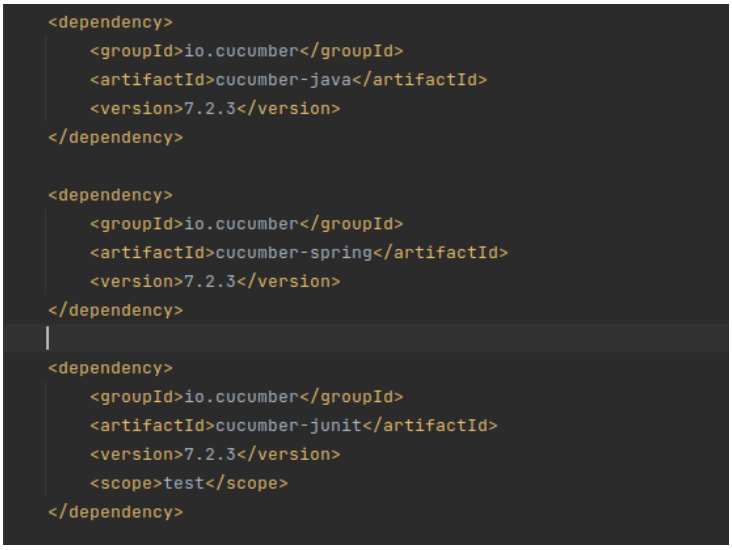
Method	Endpoint	description
POST	/api/auth/login	User can login to system
POST	/api/auth/signup	New user can register to system
POST	/api/auth/logout	User can logout from sysetm
GET	/api/user/profile	A logged in user can see its info
GET	/api/user/orders	A logged in user can see the list of his orders
GET	/order/{id}	A logged in user can see details of an order(products, …)
POST	/api/user/profile/edit	A logged in User can post its new profile information
GET	/api/user/address	A logged in user can get a list of its submitted addresses
POST	/api/user/address/add	A logged in user can submit a new address
PUT	/api/user/address/{id}	A logged in user can edit a previously submitted address
GET	/api/products	Any user can see the list of available products
GET	/api/product/{id}	Any user can see the details of a product

پروژه ی پیاده‌سازی شده در آدرس گیتهاب زیر قرار دارد:

https://github.com/fatemehkarimi/spring-ecommerce-store.git

ابزار Cucumber برای زبان برنامه نویسی java

ابزار cucumber را می توان برای Gradle و Maven برای پروژه های ی java راه اندازی نمود. در این پروژه ابزار Maven به کار گرفته شده است. برای نصب cucumber در Maven باید dependency های زیر به فایلی به نام pom.xml اضافه شود.



برای آسان‌تر شدن فرایند آزمون نرم افزار، از ابزارهای دیگر آزمون مانند rest-assured ، Junit و Mockito نیز بهره گرفته شده است.

پیاده‌سازی unit test ها

پروژه ی پیاده‌سازی شده چهار controller به نام های ProductController، UserAccountController، UserAddressController و UserAuthController دارد. برای پیاده‌سازی unit testها از ابزار Mockito بهره می بریم. به کمک این ابزار می‌توان درخواست هایی که به وسیله Daoها به دیتابیس داده می‌شود را mock کرد و آن چیزی که مدنظر است را برگرداند. مزیت این کار این است که چون با unit test ها می‌خواهیم از کارایی درست controller ها مطمئن شویم، اگر خطایی در فاز گرفتن اطلاعات از دیتابیس رخ دهد، نمی‌توانیم بفهمیم که آیا این خطا در controller بوده یا دیتابیس. Mock کردن کمک می‌کند تا از درخواست به دیتابیس جلوگیری کنیم و اگر خطایی رخ داد مطمئن هستیم که در controller رخ داده است.

ساختار کلاس‌های unit test به این صورت است که یکم کلاسی که در پایان نام خود واژه ی Test را دارد را می سازیم. پس از آن با افزودن عبارت های زیر مشخص می‌کنیم که این کلاس با چه ابزاری می بایست اجرا شود و کلاسی که تست می‌کند چه کلاسی ست:

```
@RunWith(SpringRunner.class)
@WebMvcTest({ControllerClass.class})
```

پس از آن، dao ها و service هایی که در کلاس controller تعریف شده‌اند را با افزودن عبارت MockBean پیش از تعریف آن‌ها در کلاس تست mock می کنیم. مانند مثال زیر:

```
@MockBean
ProductDao productDao;

@MockBean
UserService userService;
```

اکنون توابع تست را پیاده‌سازی می کنیم. می بایست حتماً این توابع را با @Test مشخص کنیم.

پیاده‌سازی integration test ها

برای پیاده‌سازی integration test ها از ابزار cucumber بهره می بریم. روش کار به این صورت است که سناریوهای تست را با زبان Gherkin پیاده‌سازی می کنیم و در پوشه ی /src/test/resources/ در فایل‌هایی با پسوند feature. قرار می دهیم. یک نمونه از سناریو نوشته شده به شکل زیر است:

```
Feature: User Login To Account
    a registered user must be able to login with email and password

    Scenario Outline: User Can Login To Account
        Given i send a request to URL "/api/auth/login" with email = "<email>" and password = "<password>"
        Then the result is "<result>"

        Examples:
            | email                          | password  | result        |
            | fatemehkarimi.1998@yahoo.com   | admin     | authenticated |
            | nsk.es@yahoo.com               | admin     | authenticated |
            | leily.nourbakhsh1999@gmail.com | admin     | authenticated |
            | person@example.com             | test123   | access denied |
            | firstname.lastname@gmail.com   | passs12   | access denied |


    Scenario Outline:  A logged in user must be able to log out
        Given i am already logged in with email = "<email>" and password = "<password>"
        When i send a request to URL "/api/auth/logout"
        Then i must not be able to see my profile information at URL "/api/user/profile"

        Examples:
            | email                          | password  |
            | fatemehkarimi.1998@yahoo.com   | admin     |
```

سناریوی اول مربوط به تست ورود کاربر به سیستم است. کاربری که قبلاً در وبسایت ثبت نام کرده باید بتواند با فرستادن username و password خود به منابع authorize شده سیستم دسترسی داشته باشد. در این سناریو برای چند کاربر login را می آزماییم. اطلاعات برخی از کاربران می بایست منجر به نتیجه ی authenticated یا access denied برای برخی دیگر شود. برای پیاده‌سازی گام های گفته شده در سناریو، آن‌ها را در یک کلاس Java تعریف می کنیم. در بالای هر تابع توسط annotation ها بیان شده که برای کدام گام از سناریو نوشته شده اند.

پس از پیاده‌سازی تست ها، آن‌ها را با ابزار Cucumber اجرا می کنیم. نتیجه ی اجرای تست ها در آدرس زیر آمده است.

https://reports.cucumber.io/report-collections/1dd6b00c-2cbe-4991-8a1d-9ad839e26178

```java
public class UserLoginSteps {
    private final static String BASE_URI = "http://localhost";
    @LocalServerPort
    private int port;

    ObjectMapper objectMapper = new ObjectMapper();

    private Response response;

    @Before
    public void setup() {
        RestAssured.baseURI = BASE_URI;
        RestAssured.port = port;
    }

    @Given("i send a request to URL {string} with email = {string} and password = {string}")
    public void i_send_a_request_to_url_with_email_and_password(
            String endpoint, String user_email, String user_password) throws Exception
    {
        Object loginData = new Object() {
            public String email = user_email;
            public String password = user_password;
        };

        String json = objectMapper.writeValueAsString(loginData);
        this.response = given().contentType(ContentType.JSON)
                .body(json).post(endpoint);
    }

    @Then("the result is {string}")
    public void the_result_is(String result) {
        if(result.equals("authenticated")) {
            this.response.then().assertThat().statusCode(200);
            String message = this.response.then().extract().asString();
            assertThat(message).isEqualTo("user is now logged in");
        }
        else {
            this.response.then().assertThat().statusCode(401);
            String message = this.response.jsonPath().get("message");
            assertThat(message).isEqualTo("Bad credentials");
        }
    }
}
```