[ASSIGNMENT 3]
[STP304X_01 KIỂM THỬ TỰ ĐỘNG]

Họ và tên : Phạm Hải Thanh
MSV : FX20171

TP HCM.2023

MỤC LỤC

KIỂM TRA PHẢN HỒI API - CÁCH 1

Giải thích: Ở cách làm này, chúng ta sẽ sử dụng file trong folder **Properties** để tiến hành lưu thông tin URL, user, password, lưu token.

Trình tự làm như sau:

- Đối với yêu cầu đăng nhập vào tài khoản:

B1. Lấy các thông tin cần thiết như baseUrl loginPath, accountLogin, passwordLogin từ file config.properties

B2. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra

B3. Lấy dữ liệu token (nếu có) ghi vào file token.properties

- Đối với tạo thông tin công việc mới cho tài khoản:

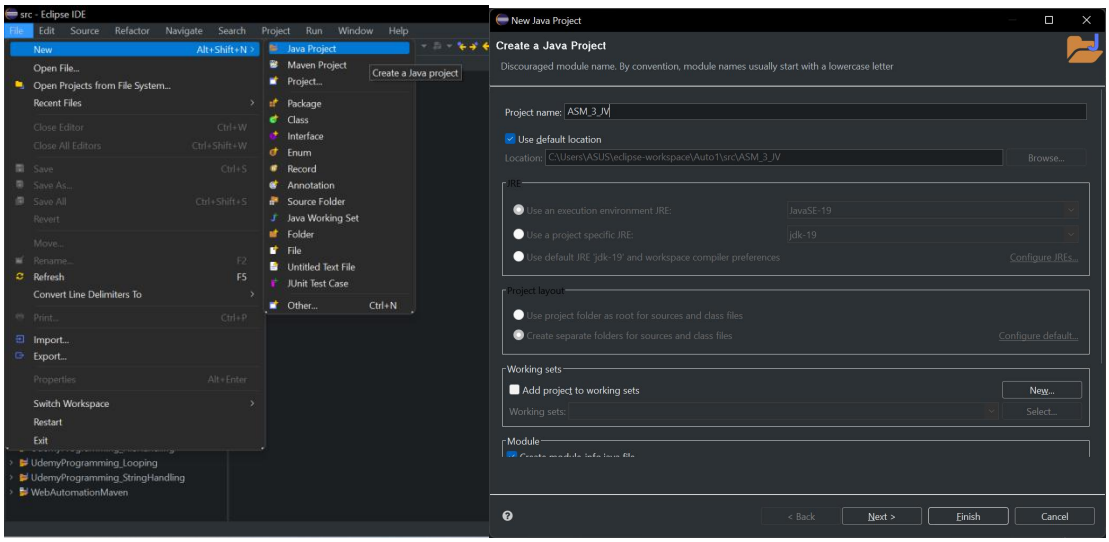
B1. Lấy dữ liệu token (nếu có) từ file token.properties ghi được ở bên trên để sử dụng làm header cho phần này

B2. Tạo thông tin công việc mới

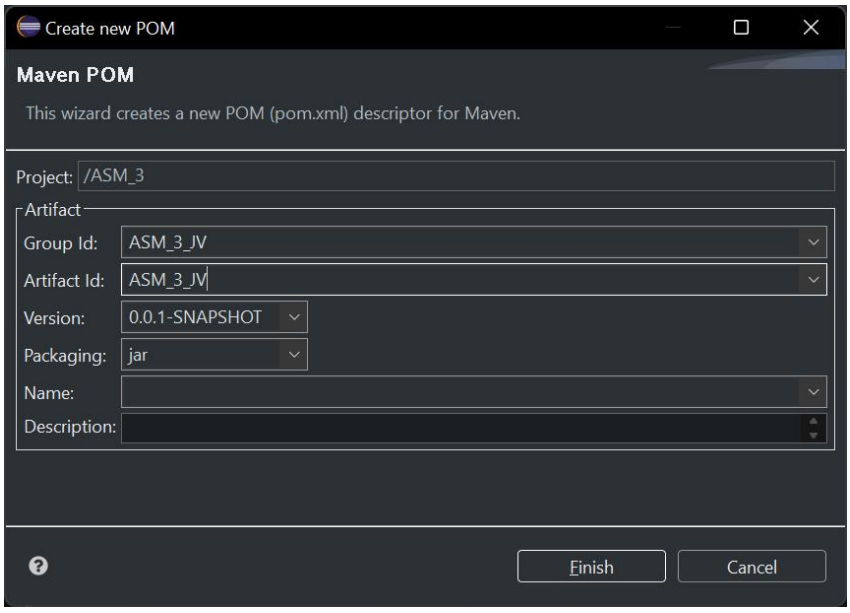
B3. Lấy dữ liệu phản hồi và so sánh với các yêu cầu đề ra

PHẦN 1: TẠO PROJECT, PACKAGE CHO ASM3

- B1: Chọn dạng dự án: Java Project



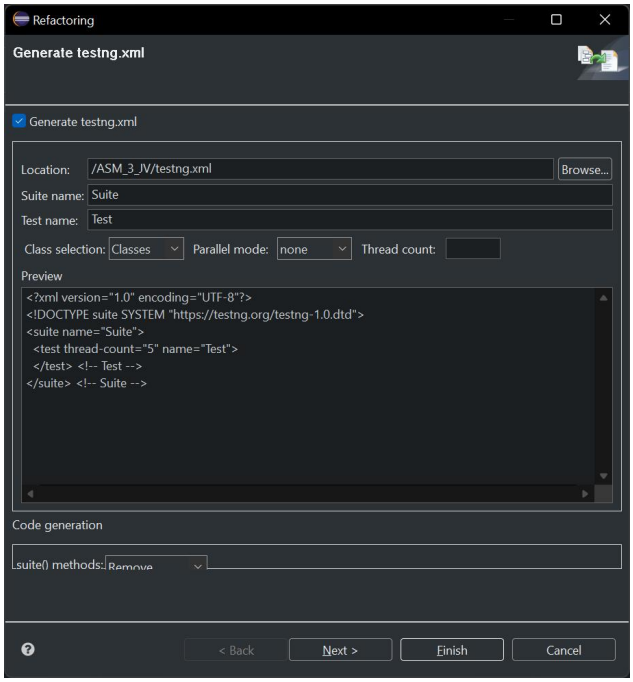
- B2: Convert sang Maven



- B3: Truy cập https://mvnrepository.com/ để cài đặt các Dependencies vào pom.xml của project để sử dụng

1. Cài poi và poi-ooxml
<dependency> <groupId>org.apache.poi</groupId> <artifactId>poi</artifactId> <version>5.2.3</version> </dependency>
<dependency> <groupId>org.apache.poi</groupId> <artifactId>poi-ooxml</artifactId> <version>5.2.3</version> </dependency>
2. Cài testng
<dependency> <groupId>org.testng</groupId> <artifactId>testng</artifactId> <version>7.4.0</version> <scope>test</scope> </dependency>
3. Cài rest-assured và dependencies hỗ trợ đọc json
<dependency> <groupId>io.rest-assured</groupId> <artifactId>rest-assured</artifactId> <version>4.4.0</version> <scope>test</scope> </dependency>
<dependency> <groupId>com.google.inject</groupId> <artifactId>guice</artifactId> <version>3.0</version> </dependency>
<dependency> <groupId>org.hamcrest</groupId> <artifactId>hamcrest</artifactId> <version>2.2</version> </dependency>
<dependency> <groupId>com.fasterxml.jackson.core</groupId> <artifactId>jackson-databind</artifactId> <version>2.13.4.2</version> </dependency>
<dependency> <groupId>org.apache.logging.log4j</groupId> <artifactId>log4j-core</artifactId> <version>2.20.0</version> </dependency>

- B4: Convert sang TestNG



PHẦN 2: TẠO CÁC THƯ MỤC VÀ VIẾT CÁC CÀI ĐẶT BỔ TRỢ

Tạo các thư mục chứa các file ,drive bổ trợ

Tên folder	File con	Giải thích
Properties	Config.properties	Chứa các thông tin như đường link web cần test, tên user, password để đăng nhập
	Token.properties	Chứa giá trị token sau khi login, dùng cho phần CreateWork ở sau.

Xác định và thêm thông tin vào Config.properties và Elements.properties
- Configs.properties chứa các thông tin URL page , username, password

#Maintaining Application Configuration Data

baseUrl=http://13.228.39.1:5000
loginPath=/api/users/login
createWorkPath=/api/work-user/createWork-user
updateWorkPath=
DetailWorkPath=
accountLogin=testerFunix
passwordLogin=Abc13579

- Token.properties chứa giá trị token sau khi login, dùng cho phần CreateWork ở sau. File này để trống và sẽ được ghi tự động khi login thành công, lấy giá trị token và ghi thành công

PHẦN 3: TẠO CÁC PACKAGES VÀ CLASSES TRONG PROJECT

Tạo 2 folder packages trong file ASM_3_JV: là src.test và src.main

Tên packages	Mục đích	Các class con
src.test		
com.automation.testcase	chứa các class test sử dụng TestNG	TC_API_Login.java
		TC_API_CreateWork.java
src.main		
com.automation.untils	chứa class hỗ trợ đọc properties file ...	PropertiesFileUtils.java

3.1 Tạo package com.automation.utils

Mục đích: chứa các class hỗ trợ đọc properties file …

3.1.1 Class PropertiesFileUtils.java

- Class PropertiesFileUtils.java giúp đọc các giá trị từ file Configs.proterties và Token.properties trong folder Properties.

```
package com.api.auto.utils;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Properties;

public class PropertiesFileUtils {

//Tạo đường dẫn đến properties files trong folder Properties
   public static String CONFIG_PATH = "./Properties/Config.properties";

//Tạo đường dẫn đến token properties files trong folder Properties
   public static String TOKEN_PATH = "./Properties/Token.properties";

// Đọc giá trị đã lưu từ file Config.properties
   public static String getProperty(String key) {
            Properties properties= new Properties();
            String value = null;
            FileInputStream fileInput = null;

//Tạo exception nếu giá trị cần đọc rỗng
      try {
            fileInput = new FileInputStream(CONFIG_PATH);
           properties.load(fileInput);
           value = properties.getProperty(key);

          if (value != null) {              // nếu không có key, trả về một chuỗi rỗng
           value = value.trim();            // loại bỏ khoảng trắng đầu, cuối chuỗi
          }
         return value;
         }catch (Exception ex) {
         System.out.println("Xảy ra lỗi khi đọc giá trị của  " + key);
         ex.printStackTrace();
         } finally {    //luôn nhảy vào đây dù có xảy ra exception hay không.
```

```
                    if (fileInput!= null) {
                    try {
                        fileInput.close(); //thực hiện đóng luồng đọc
                    } catch (IOException e) {
                         e.printStackTrace();
                              }
                    }
    }
    return value;
  }

// Lưu token vào file token.properties với key là “token”
   public static void saveToken(String key, String value) {
      Properties properties= new Properties();
      FileOutputStream fileOutput = null;

//Tạo exception nếu giá trị cần lưu rỗng
      try {
         fileOutput = new FileOutputStream(TOKEN_PATH);
         properties.setProperty(key, value);
         properties.store(fileOutput, value);
         System.out.println("Set new value in file Token.properties success.");
      } catch (IOException ex) {
         ex.printStackTrace();
      } finally {    //luôn nhảy vào đây dù có xảy ra exception hay không.
         if (fileOutput!= null) {
            try {
               fileOutput.close();        //thực hiện đóng luồng đọc
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
      }
   }

// Lấy ra token đã lưu, và đọc giá trị token
     public static String getToken(String key) {
           Properties properties= new Properties();
            String value = null;
           FileInputStream fileInput = null;
//Tạo exception nếu giá trị cần đọc rỗng
           try {
                fileInput = new FileInputStream(TOKEN_PATH);
                properties.load(fileInput);
                 value = properties.getProperty(key);
                 if (value != null) {        // nếu không có key, trả về một chuỗi rỗng
                 value = value.trim();   // loại bỏ khoảng trắng đầu, cuối chuỗi
                 }
                 return value;
           } catch (Exception ex) {
              System.out.println("Xảy ra lỗi khi đọc giá trị của token " + key);
             ex.printStackTrace();
           } finally {
                    if (fileInput != null) {
                    try {
                        fileInput.close();
                    }catch (IOException e) {
```

```
                        e.printStackTrace();
                        }
                    }
            }
            return value;
        }
}
```

3.2 Tạo package com.automation.testcase

3.2.1 Chứa class TC_API_Login.java

- Chứa class TC_API_Login.java là nơi để thực hiện việc test API login. Chúng ta sẽ sử dụng dữ liệu đã cho là user, password để đăng nhập và kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi

```
package com.api.auto.testcase;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.util.HashMap;
import java.util.Map;

import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import com.api.auto.utils.PropertiesFileUtils;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;

public class TC_API_Login {
// Gọi các biến sử dụng trong testcase
        private String account;
        private String password;
        private Response response;
        private ResponseBody responseBody;
        private JsonPath jsonBody;
        private String token ;

        @BeforeClass
        public void init() {
// Đọc đường dẫn URL và account/password đã lưu từ file Config.properties
                String baseUrl = PropertiesFileUtils.getProperty("baseUrl");
                String loginPath = PropertiesFileUtils.getProperty("loginPath");;
                account = PropertiesFileUtils.getProperty("accountLogin");
                password = PropertiesFileUtils.getProperty("passwordLogin");

                RestAssured.baseURI = baseUrl;

// Tạo và đọc body dưới dạng JSON
                Map<String, Object> user = new HashMap<String, Object>();
                user.put("account", account);
                user.put("password", password);
```

```
    RequestSpecification request = RestAssured.given()
                    .contentType(ContentType.JSON)
                    .body(user);

                    response = request.post(loginPath);
                    responseBody = response.body();
                    jsonBody = responseBody.jsonPath();

                    System.out.println(" " + responseBody.asPrettyString());
          }

//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
          @Test(priority = 0)
          public void TC01_Validate200Ok() {
// Kiểm chứng status code có là 200 hay không
                    assertEquals(response.getStatusCode(), 200);
                    System.out.println("\nStatus code is: " + response.getStatusCode()+"\n");
          }

          @Test(priority = 1)
          public void TC02_ValidateMessage() {
// Kiểm chứng response body có chứa trường message hay không
                    assertTrue(responseBody.asString().contains("message"), "Message field check
Failed!");
          }

          @Test(priority = 2)
          public void TC03_VerifyOnMatchMessage() {
// Kiểm chứng trường message có hiển thị nội dung "Đăng nhập thành công" hay không
          assertEquals("Đăng nhập thành công",
responseBody.jsonPath().getString("message"),"Message check Failed!");
          }

          @Test(priority = 3)
          public void TC04_ValidateToken() {
// Kiểm chứng response body có chứa trường token hay không
                    token = jsonBody.getString("token");
    assertTrue(responseBody.asString().contains("token"), "Token field check Failed!");

// Lưu lại token đã chạy vào Token.properties để sử dụng ở phần CreateWork
    PropertiesFileUtils.saveToken("token", token);
    System.out.println("\nYour token is :" +token+"\n");
  }

          @Test(priority = 4)
          public void TC05_ValidateUser() {
// Kiểm chứng response chứa thông tin user hay không
    assertTrue(responseBody.asString().contains("user"), "User field check Failed!");
          }

          @Test(priority = 5)
          public void TC06_ValidateUserType() {
// Kiểm chứng response body có chứa thông tin trường type hay không
    assertTrue(responseBody.asString().contains("type"), "Type field check Failed!");
          }
```

```
        @Test(priority = 6)
   public void TC07_VerifyOnMatchType() {
// Kiểm chứng trường type nội dung có phải là “UNGVIEN” hay không
                assertEquals(jsonBody.getString("user.type"),"UNGVIEN", "Type check Failed!");
        }

        @Test(priority = 7)
        public void TC08_ValidateAccount() {
// Kiểm chứng response body có chứa thông tin trường account hay không
     assertTrue(responseBody.asString().contains("account"), "account field check Failed!");
        }

        @Test(priority = 8)
        public void TC09_VerifyOnMatchAccount() {
// Kiểm chứng trường account có khớp với account đăng nhập đã cho hay không
                assertEquals(jsonBody.getString("user.account"), account,"Account check Failed!");
        }

        @Test(priority = 9)
        public void TC10_ValidatePassword()  {
 // Kiểm chứng response chứa thông tin trường password hay không
     assertTrue(responseBody.asString().contains("password"), "Type field check Failed!");
        }

        @Test(priority = 10)
        public void TC11_VerifyOnMatchPassword() {
// Kiểm chứng trường password có khớp với password đăng nhập đã cho hay không
    assertEquals(jsonBody.getString("user.password"), password,"Password check Failed!");
        }
}
```

3.2.2 Chứa class TC_API_CreateWork.java

- Class TC_API_CreateWork.java là nơi để thực hiện việc tạo công việc. Chúng ta sẽ sử dụng dữ liệu token đã lấy được từ lúc login thành công để làm header và tạo dữ liệu. Sau khi tạo dữ liệu, nhận được phản hồi thì sẽ kiểm tra phản hồi như status code, các trường dữ liệu trong phản hồi

```
package com.api.auto.testcase;

import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertTrue;

import java.util.HashMap;
import java.util.Map;

import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

import com.api.auto.utils.PropertiesFileUtils;

import io.restassured.RestAssured;
import io.restassured.http.ContentType;
import io.restassured.path.json.JsonPath;
import io.restassured.response.Response;
import io.restassured.response.ResponseBody;
import io.restassured.specification.RequestSpecification;
```

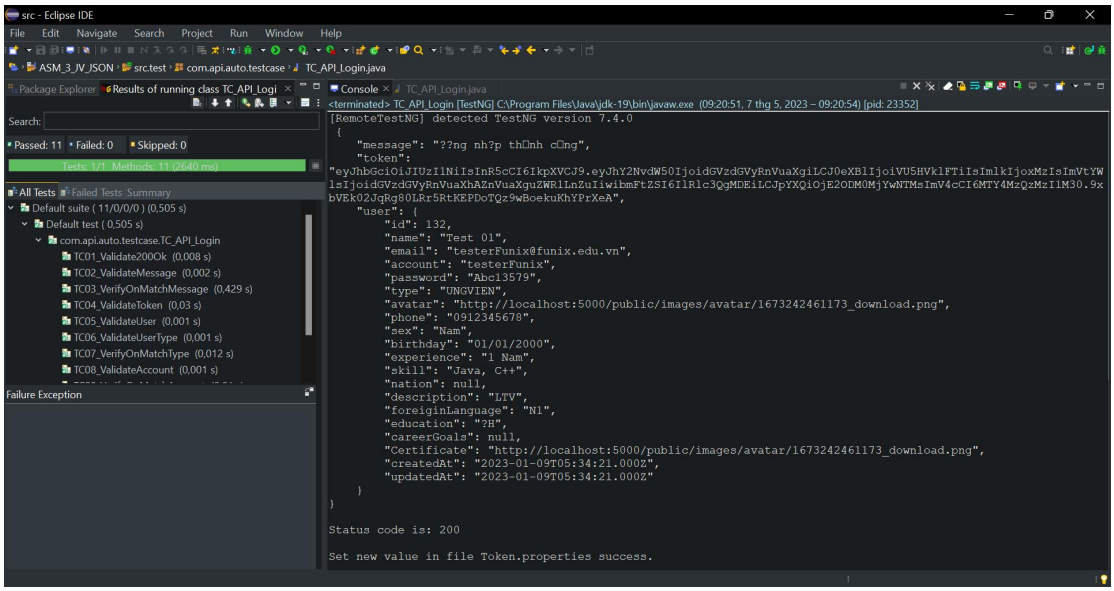
```
public class TC_API_CreateWork {

	private String token;
	private Response response;
	private ResponseBody responseBody;
	private JsonPath jsonBody;

//Tự tạo data sẽ dùng
	private String myWork = "Giang vien";
	private String myExperience = "3";
	private String myEducation = "Thac si";
	@BeforeClass
	public void init() {
// Đọc đường dẫn URL đã lưu từ file Config.properties, token từ file Token.properties file

		String baseUrl =PropertiesFileUtils.getProperty("baseUrl");
		String createWorkPath =PropertiesFileUtils.getProperty("createWorkPath");
		String token = PropertiesFileUtils.getToken("token");

		RestAssured.baseURI = baseUrl;
// Tạo và đọc body dưới dạng JSON
		Map<String, Object> inforCreated= new HashMap<String, Object>();
		inforCreated.put("nameWork",myWork);
		inforCreated.put("experience",myExperience);
		inforCreated.put("education",myEducation);

		RequestSpecification request = RestAssured.given()
				.contentType(ContentType.JSON)
				.header("token", token)
				.body(inforCreated);

		response = request.post(createWorkPath);
		responseBody = response.body();
		jsonBody = responseBody.jsonPath();

		System.out.println(" " + responseBody.asPrettyString());
	}

//THỰC HIỆN TESTCASE, CÀI ĐẶT THỨ TỰ ƯU TIÊN TEST NẾU CẦN
	@Test(priority = 0)
	public void TC01_Validate201Created() {
// Kiểm chứng status code có là 201 hay không
		assertEquals(response.getStatusCode(), 201);
		System.out.println("\nStatus code is: " + response.getStatusCode()+"\n");
	}

	@Test(priority = 1)
	public void TC02_ContainWorkId() {
// Kiểm chứng response body có chứa trường message hay không
		assertTrue(responseBody.asString().contains("id"), "Id field check Failed!");
	}

	@Test(priority = 2)
	public void TC03_ContainNameOfWork() {
// Kiểm chứng response body có chứa trường nameWork hay không
		assertTrue(responseBody.asString().contains( "nameWork"), "nameWork field check
Failed!");
```

```
	}

	@Test(priority = 3)
	public void TC04_ValidateNameOfWorkMatched() {
// Kiểm chứng trường nameWork có khớp với thông tin myWork đã tạo hay không
		assertEquals(jsonBody.getString("nameWork"), myWork,"nameWork check Failed!");
	}

	@Test(priority = 4)
	public void TC05_ContainExperienceOfWork() {
// Kiểm chứng response body có chứa trường experience hay không
		assertTrue(responseBody.asString().contains( "experience"), "experience field check Failed!");
	}

	@Test(priority = 5)
	public void TC06_ValidateExperienceMatched() {
// Kiểm chứng trường experience có khớp với thông tin myExperience đã tạo hay không
		assertEquals(jsonBody.getString("experience"), myExperience,"experience check Failed!");
	}

	@Test(priority = 6)
	public void TC07_ContainEducationOfUser() {
// Kiểm chứng response body có chứa trường education hay không
		assertTrue(responseBody.asString().contains( "education"), "education field check Failed!");
	}

	@Test(priority = 7)
	public void TC08_ValidateEducationMatched() {
// Kiểm chứng trường education có khớp với thông tin myEducation đã tạo hay không
		assertEquals(jsonBody.getString("education"), myEducation,"education check Failed!");
	}
}
```

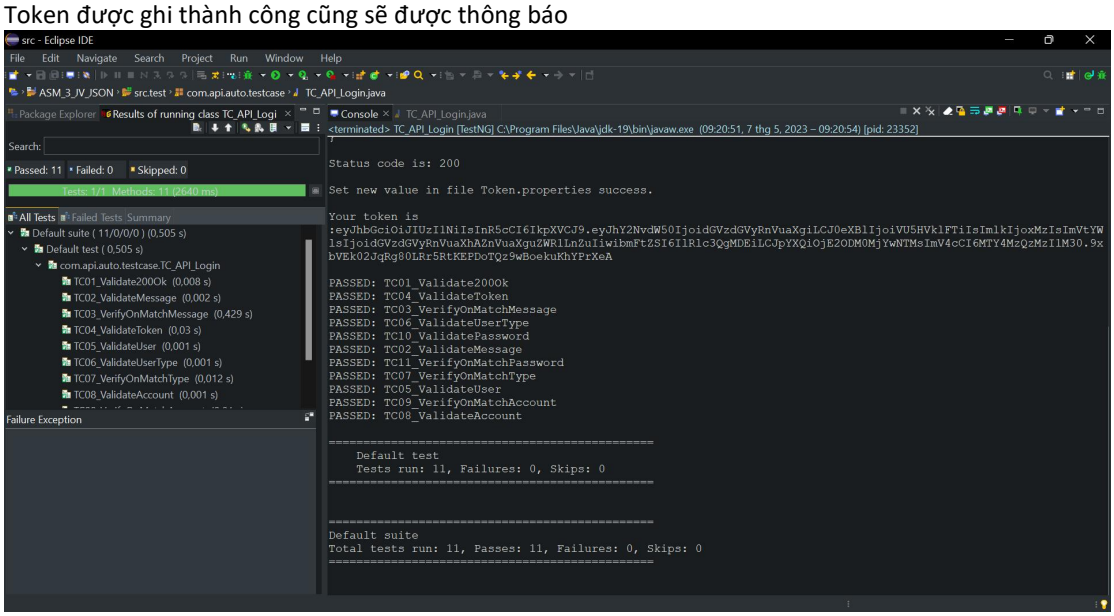
PHẦN 4: KẾT QUẢ CHẠY CHƯƠNG TRÌNH

4.1 Trường hợp TC_API_Login.java

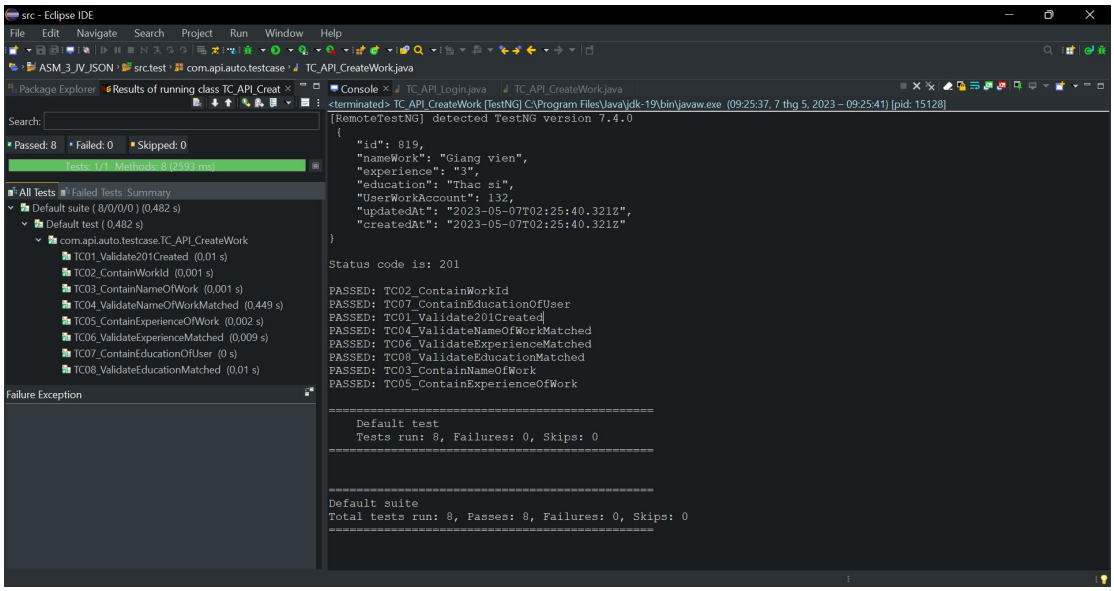
Khi chạy chương trình sẽ in ra các thông tin hiển thị của phản hồi sau khi login thành công như bên dưới.



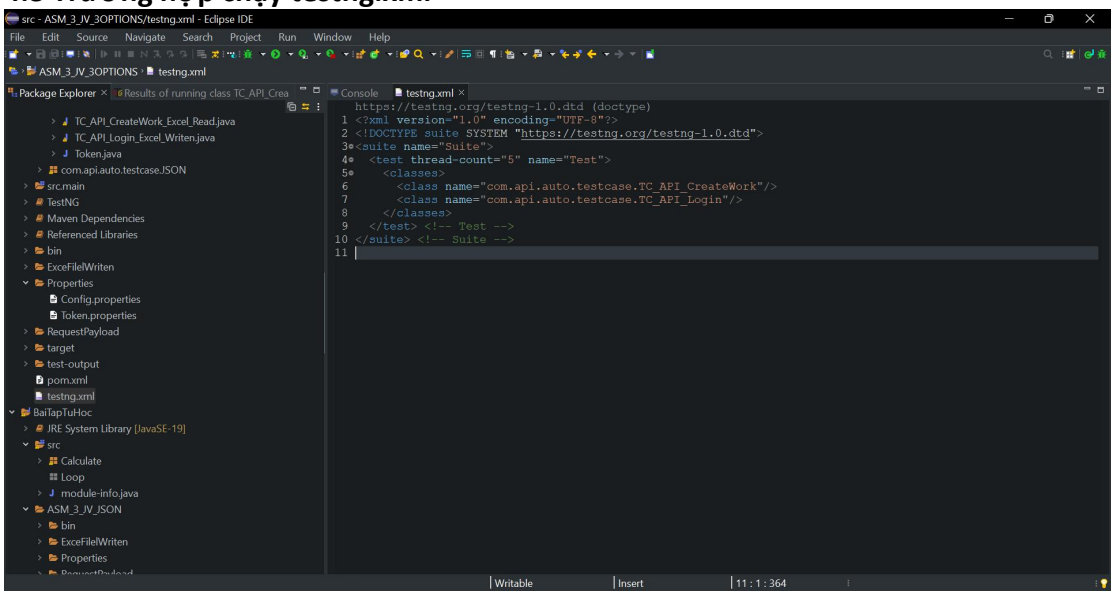
Token được ghi thành công cũng sẽ được thông báo

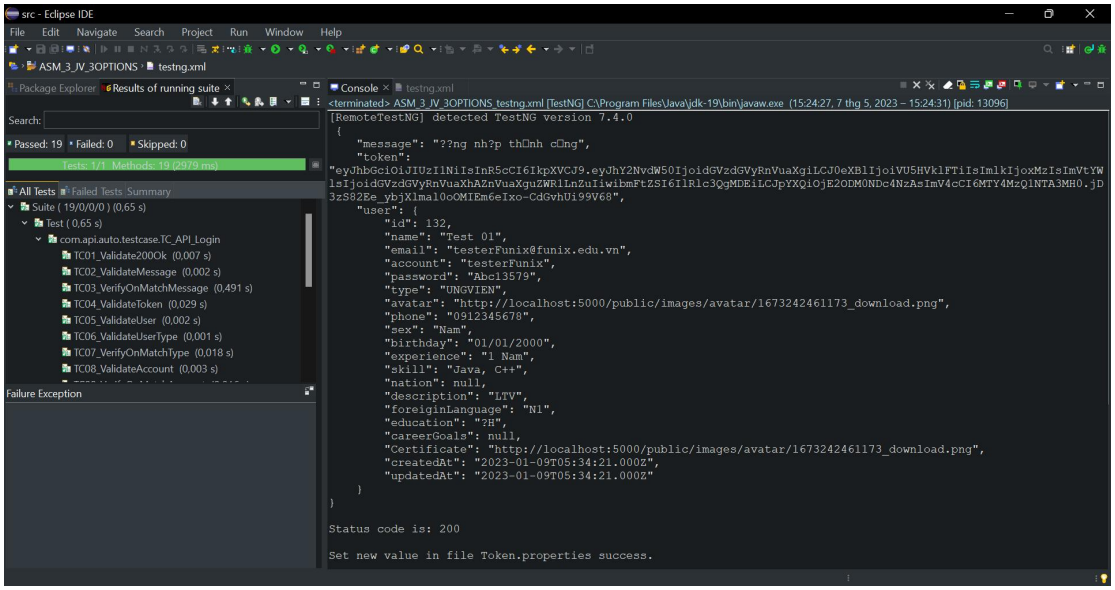


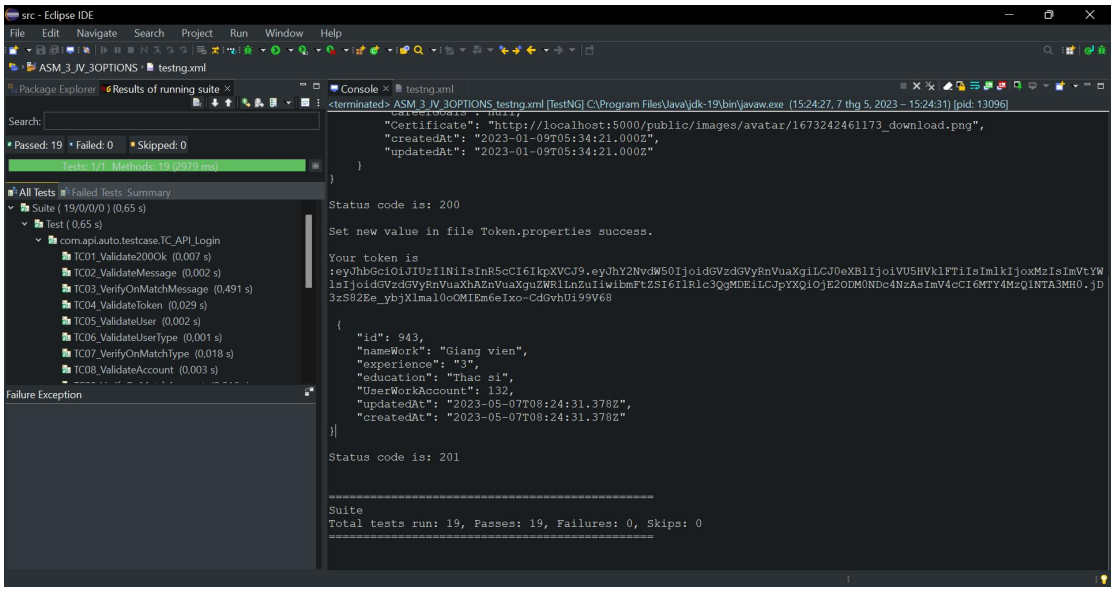
4.2 Trường hợp TC_API_CreateWork.java



4.3 Trường hợp chạy testng.xml




src - Eclipse IDE
File Edit Navigate Search Project Run Window Help
ASM_3_JV_3OPTIONS testng.xml
Package Explorer Results of running suite
Search:
Passed: 19 Failed: 0 Skipped: 0
Tests: 1/1 Methods: 19 (2979 ms)
All Tests Failed Tests Summary
Suite (19/0/0/0) (0,65 s)
Test (0,65 s)
com.api.auto.testcase.TC_API_Login
TC01_Validate200Ok (0,007 s)
TC02_ValidateMessage (0,002 s)
TC03_VerifyOnMatchMessage (0,491 s)
TC04_ValidateToken (0,029 s)
TC05_ValidateUser (0,002 s)
TC06_ValidateUserType (0,001 s)
TC07_VerifyOnMatchType (0,018 s)
TC08_ValidateAccount (0,003 s)
Failure Exception
Console testng.xml
<terminated> ASM_3_JV_3OPTIONS_testng.xml [TestNG] C:\Program Files\Java\jdk-19\bin\javaw.exe (15:24:27, 7 thg 5, 2023 – 15:24:31) [pid: 13096]
[RemoteTestNG] detected TestNG version 7.4.0
{
"message": "?ng nh?p th□nh c□ng",
"token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoidGVzdGVyRnVuaXgiLCJ0eXBlIjoiVU5HVklFTiIsImlkIjoxMzIsImVtYWlsIjoidGVzdGVyRnVuaXhAZnVuaXguZWR1LnZuIiwibmFtZSI6IlRlc3QgMDEiLCJpYXQiOjE2ODM0NDc4NzAsImV4cCI6MTY4MzQ1NTA3MH0.jD3zS82Ee_ybjXlmal0oOMIEm6eIxo-CdGvhUi99V68",
"user": {
"id": 132,
"name": "Test 01",
"email": "testerFunix@funix.edu.vn",
"account": "testerFunix",
"password": "Abc13579",
"type": "UNGVIEN",
"avatar": "http://localhost:5000/public/images/avatar/1673242461173_download.png",
"phone": "0912345678",
"sex": "Nam",
"birthday": "01/01/2000",
"experience": "1 Nam",
"skill": "Java, C++",
"nation": null,
"description": "LTV",
"foreiginLanguage": "N1",
"education": "?H",
"careerGoals": null,
"Certificate": "http://localhost:5000/public/images/avatar/1673242461173_download.png",
"createdAt": "2023-01-09T05:34:21.000Z",
"updatedAt": "2023-01-09T05:34:21.000Z"
}
}
Status code is: 200
Set new value in file Token.properties success.


src - Eclipse IDE
File Edit Navigate Search Project Run Window Help
ASM_3_JV_3OPTIONS testng.xml
Package Explorer Results of running suite
Search:
Passed: 19 Failed: 0 Skipped: 0
Tests: 1/1 Methods: 19 (2979 ms)
All Tests Failed Tests Summary
Suite (19/0/0/0) (0,65 s)
Test (0,65 s)
com.api.auto.testcase.TC_API_Login
TC01_Validate200Ok (0,007 s)
TC02_ValidateMessage (0,002 s)
TC03_VerifyOnMatchMessage (0,491 s)
TC04_ValidateToken (0,029 s)
TC05_ValidateUser (0,002 s)
TC06_ValidateUserType (0,001 s)
TC07_VerifyOnMatchType (0,018 s)
TC08_ValidateAccount (0,003 s)
Failure Exception
Console testng.xml
<terminated> ASM_3_JV_3OPTIONS_testng.xml [TestNG] C:\Program Files\Java\jdk-19\bin\javaw.exe (15:24:27, 7 thg 5, 2023 – 15:24:31) [pid: 13096]
"Certificate": "http://localhost:5000/public/images/avatar/1673242461173_download.png",
"createdAt": "2023-01-09T05:34:21.000Z",
"updatedAt": "2023-01-09T05:34:21.000Z"
}
}
Status code is: 200
Set new value in file Token.properties success.
Your token is
:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJhY2NvdW50IjoidGVzdGVyRnVuaXgiLCJ0eXBlIjoiVU5HVklFTiIsImlkIjoxMzIsImVtYWlsIjoidGVzdGVyRnVuaXhAZnVuaXguZWR1LnZuIiwibmFtZSI6IlRlc3QgMDEiLCJpYXQiOjE2ODM0NDc4NzAsImV4cCI6MTY4MzQ1NTA3MH0.jD3zS82Ee_ybjXlmal0oOMIEm6eIxo-CdGvhUi99V68
{
"id": 943,
"nameWork": "Giang vien",
"experience": "3",
"education": "Thac si",
"UserWorkAccount": 132,
"updatedAt": "2023-05-07T08:24:31.378Z",
"createdAt": "2023-05-07T08:24:31.378Z"
}
Status code is: 201
Suite
Total tests run: 19, Passes: 19, Failures: 0, Skips: 0