



AT Lab 2022CW16

Java project

30.09.2022

Mikhail Starilov

Mger Sarkisyan

Nastassia Stanchyk

Yauheniya Virinskaya

AGENDA

- 1 PROJECT SUMMARY
- 2 REQUIREMENTS SUMMARY
- 3 SCENARIOS WORKFLOW
- 4 TECH WORKFLOW
- 5 SPRINTS SUMMARY DONE
- 6 FUTURE PLANS



PROJECT SUMMARY

Our project is an automation testing framework for two websites:

automationpractice.com (Online shop)



TOOLS

- **Code** – Java 11, IntelliJ IDEA
- **Test** – Selenium WebDriver, JUnit 5, Cucumber
- **Build** – Maven
- **Deploy** - Jenkins
- **Monitor** – Report portal

open-meteo.com (Open-source weather API)

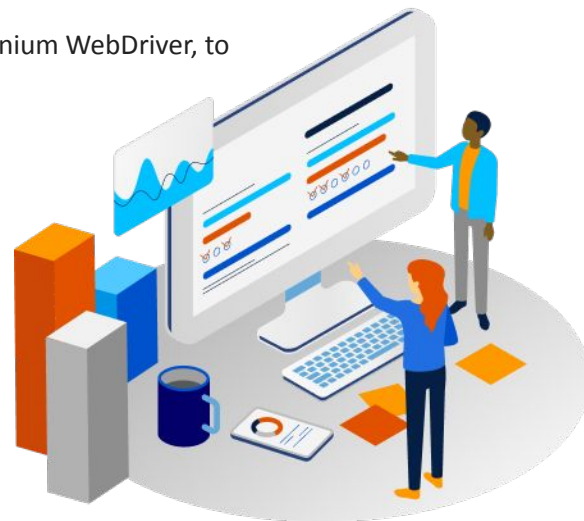


TOOLS

- **Code** – Java 11, IntelliJ IDEA
- **Test** – RestAssured, JUnit 5
- **Build** – Maven
- **Deploy** - Jenkins
- **Monitor** – Report portal

REQUIREMENTS SUMMARY

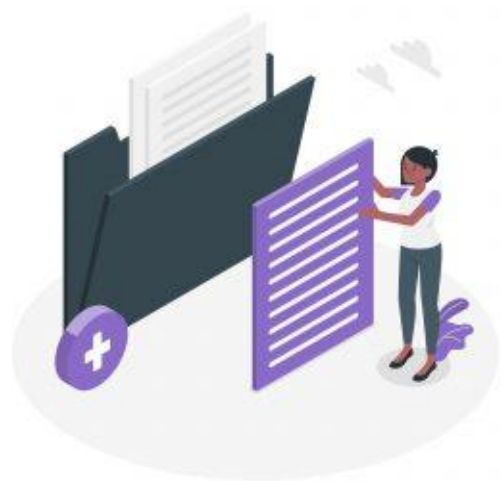
- Create **API** test case in JIRA for **OpenMeteo** (10-20 test cases, 1 test case has 2 descriptions – BDD and not BDD as separate Jira test tickets)
- Create **UI** test cases in JIRA for **YourLogo** app (10-20 test cases, 1 test case has 2 descriptions – BDD and not BDD as separate Jira test tickets)
- Create test **framework** with implemented UI test scenarios using Maven, JUnit, Cucumber, Selenium WebDriver, to run tests via Chrome (primary) and Firefox browsers
- Add implemented **API** test scenarios to the current test framework using RestAssured
- Create pipeline with tests in **CI** (Jenkins)
- **Report Portal** integration
- **Logger / Screenshots**
- **Tests Mapping** - Mark test classes with tickets' ID from JIRA



SCENARIOS WORKFLOW

We implemented about **20** test scenarios for **UI** and **15** for **API** testing:

- Every scenario has own **Task** in JIRA (BDD and not BDD)
- In test scenarios is implemented **user behavior** on the site
- Scenarios fully cover the **main functions** of the websites





Review and estimation:

- All scenarios have been **reviewed** by team and **approved**
- After Sprint Planning scenarios have been **estimated** during Poker Planning meetings

Maven

Two ways in maven-surefire-plugin to achieve parallel test execution

1. Forks (JVM processes) 
2. Threads 

Why forks?

1. Resource isolation
2. Easier to set up and control than Threads

```
mvn test -DforkCount=2
```

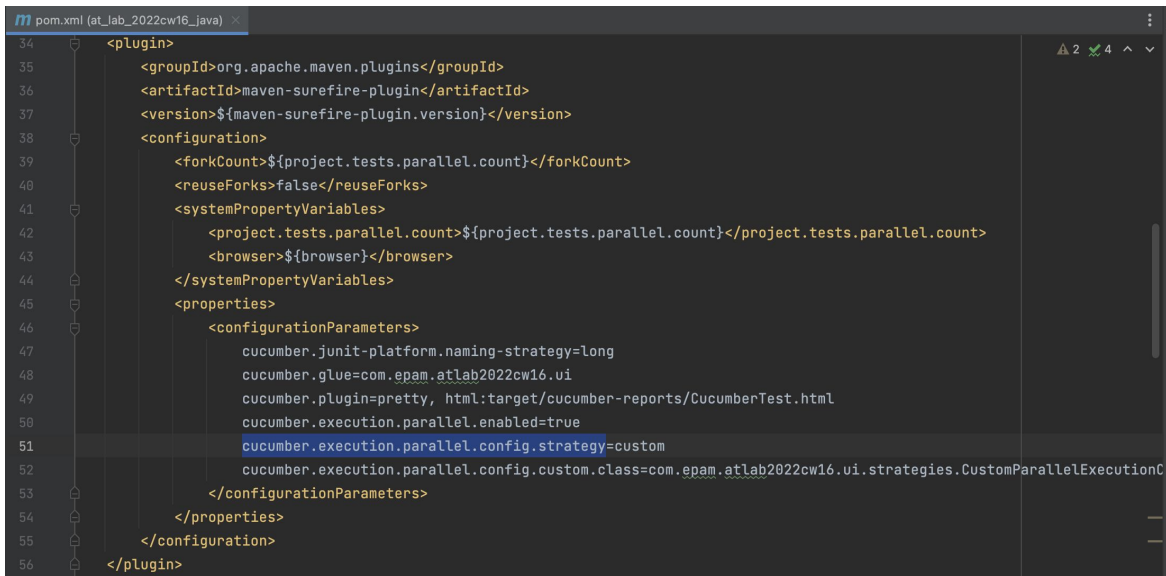
```
<properties>
  <junit-jupiter.version>5.9.0</junit-jupiter.version>
  <forkCount>1</forkCount>
  <java.version>11</java.version>
</properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>${maven-surefire-plugin.version}</version>
      <configuration>
        <forkCount>${forkCount}</forkCount>
        <reuseForks>false</reuseForks>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>${maven-compiler-plugin.version}</version>
    </plugin>
  </plugins>
</build>
```

BDD

- Running BDD test scenarios with maven
- Running BDD test scenarios in parallel



```
34 <plugin>
35   <groupId>org.apache.maven.plugins</groupId>
36   <artifactId>maven-surefire-plugin</artifactId>
37   <version>${maven-surefire-plugin.version}</version>
38   <configuration>
39     <forkCount>${project.tests.parallel.count}</forkCount>
40     <reuseForks>false</reuseForks>
41     <systemPropertyVariables>
42       <project.tests.parallel.count>${project.tests.parallel.count}</project.tests.parallel.count>
43       <browser>${browser}</browser>
44     </systemPropertyVariables>
45     <properties>
46       <configurationParameters>
47         cucumber.junit-platform.naming-strategy=long
48         cucumber.glue=com.epam.atlab2022cw16.ui
49         cucumber.plugin=pretty, html:target/cucumber-reports/CucumberTest.html
50         cucumber.execution.parallel.enabled=true
51         cucumber.execution.parallel.config.strategy=custom
52         cucumber.execution.parallel.config.custom.class=com.epam.atlab2022cw16.ui.strategies.CustomParallelExecutionC
53       </configurationParameters>
54     </properties>
55   </configuration>
56 </plugin>
```

Driver

```
class AbstractBuilder{  
    defaultConfig();  
    windowsSize();  
    driverVersion();  
    build();}
```



```
class ChromeBuilder{  
    @Override  
    build();  
    @Override  
    defaultConfig();}
```



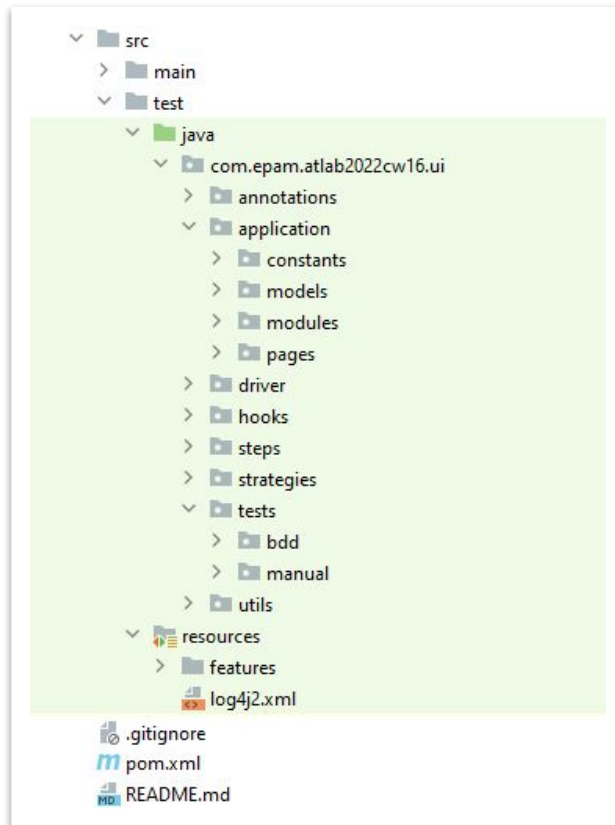
```
class FirefoxBuilder{  
    @Override  
    build();  
    @Override  
    defaultConfig();}
```

```
class WebDriverApp{  
    getChrome()  
    getFirefox()  
    systemPropertyBrowser(){  
        switch(EnvironmentUtils.getBrowserName())  
        {  
            case CHROME: {  
                return new ChromeBuilder();  
            }  
            case FIREFOX: {  
                return new FirefoxBuilder();  
            }  
        }  
    }  
    return null;}}
```


FRAMEWORK ARCHITECTURE

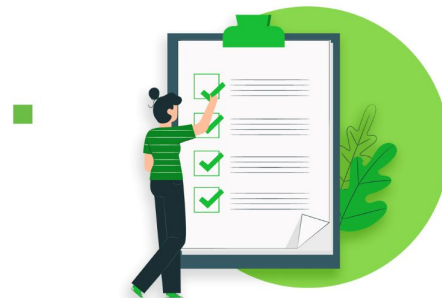
Framework layers :

- **Page Objects**
- **UI Tests**
- **Business Models**
- **Logging / Screenshot hooks**
- **BDD: Steps / Feature files**



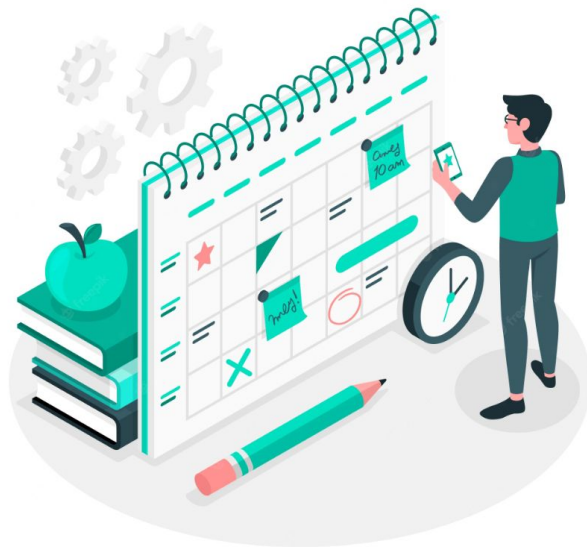
SPRINTS SUMMARY DONE

1. Created **API** test cases in JIRA for **OpenMeteo**
2. Created **UI** test cases in JIRA for **YourLogo**
3. All scenarios estimated during Poker planning meetings
4. Implemented UI test scenarios:
 - BDD – 7
 - Manual - 9
5. **Multibrowsing** support implemented



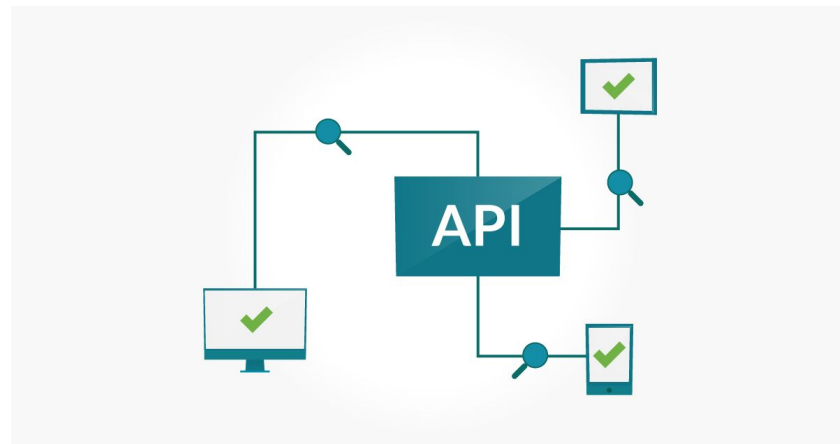
FUTURE PLANS

- Add implemented **API** test scenarios to the current test framework using RestAssured (BDD and not BDD part)
- Create pipeline with tests in **CI** (Jenkins),
- **Report Portal** integration
- Optional: Update test framework using **Gradle** configuration



Entry Criteria

- 32 test scenarios created
 - 16 BDD
 - 16 not BDD
- 2 test environments
 - httpbin.org
 - open-meteo.com
- Tools
 - RestAssured



API

Results

- 16 tests were implemented
 - 8 BDD
 - 8 not BDD
- Tests filter by @Tag annotation
 - @Tag("api")
 - @Tag("bdd")
 - -Dgroups=api & bdd

Implemented endpoints

- open-meteo.com
 - marine
 - elevation
 - geocoding
 - forecast (hourly, daily)
 - archive
- httpbin.org
 - cookies

CI TOOL (JENKINS)

SCOPE OF WORK:

1. Needed to run auto tests on Jenkins using the **jenkins pipeline**.
2. **Pipeline scripts** should be stored in project repository as **Jenkinsfiles**.
3. Should be created two **jobs** for API and UI tests
4. Should be configured **gathering of statistics on running tests**.

JENKINSFILE FOR API TESTS

```
pipeline {
  agent {
    label 'windows'
  }
  triggers {
    cron 'H H(8-20)/3 * * H(1-5)'
  }
  parameters {
    string(name: 'forkCount', defaultValue: '3')
    string(name: 'testTag', defaultValue: 'api')
    string(name: 'testSuite', defaultValue: 'com/epam/atlab2022cw16/ui/tests/bdd/RunCucumberTest')
  }
  stages {
    stage('Test') {
      steps {
        bat "mvn -Dproject.tests.parallel.count=${forkCount} -Dexclude=${testSuite} -Dgroups=${testTag} clean test"
      }
    }
  }
  post {
    always {
      junit '/target/surefire-reports/TEST-*.xml'
      archiveArtifacts artifacts: '/target/surefire-reports/TEST-*.xml'
    }
    failure {
      archiveArtifacts artifacts: '/target/**'
    }
  }
}
```

JENKINSFILE FOR UI TESTS

```
pipeline {
  agent {
    label 'windows'
  }
  triggers {
    cron 'H H(8-20)/3 * * H(1-5)'
  }
  parameters {
    choice (choices: ['chrome', 'firefox'], name: 'browser')
    string (defaultValue: '3', name: 'testCount')
    string (name: 'testTag', defaultValue: 'ui')
  }
  stages {
    stage('Test') {
      steps {
        bat "mvn -Dbrowser=${browser} -Dproject.tests.parallel.count=${testCount} -Dgroups=${testTag} clean test"
      }
    }
  }
  post {
    always {
      junit '/target/surefire-reports/TEST-*.xml'
      archiveArtifacts allowEmptyArchive: true, artifacts: '**/target/surefire-reports/TEST-*.xml', followSymlinks: false
      archiveArtifacts allowEmptyArchive: true, artifacts: '**/target/cucumber-reports/CucumberTest.html', followSymlinks:
false
    }
    failure {
      archiveArtifacts allowEmptyArchive: true, artifacts: '**/target/screenshots/*.png', followSymlinks: false
    }
  }
}
```


Report Portal

ReportPortal is a service, that provides increased capabilities to speed up results analysis and reporting through the use of built-in analytic features.

Why we need Report Portal:

1. Add all logs and screenshots to be able to manage and view all testing results
2. The results of all autotests on your project are accumulated in one place
3. The test cases are shown together with all related data like logs, screenshots
4. With ReportPortal, you can define the failure reasons of a test case and set a Defect type for it: Product bug, Auto Bug, System Issue, or custom type

Q&A