# Автоматическое тестирование UI и API

Выполнил Хилько А.А.

### Задание

#### Тестирование UI

- Наличие элементов интерфейса надписи, заголовков и надписей.
- Тестирование работоспособности формы входа при вводе корректных и некорректных данных
- Тестирование поиска билетов при некорректных данных, с разным числом пассажиров

#### Тестирование АРІ

- Тестирование ответов на запросы
- ▶ Ответы с корректными данными
- Ответы с некорректными данными
- Наличие в ответе корректных значений

### Технические требования

Проект Maven (pom.xml)

**TestNG** 

Selenium

Driver Singleton, PageObject

Allure

Surefire

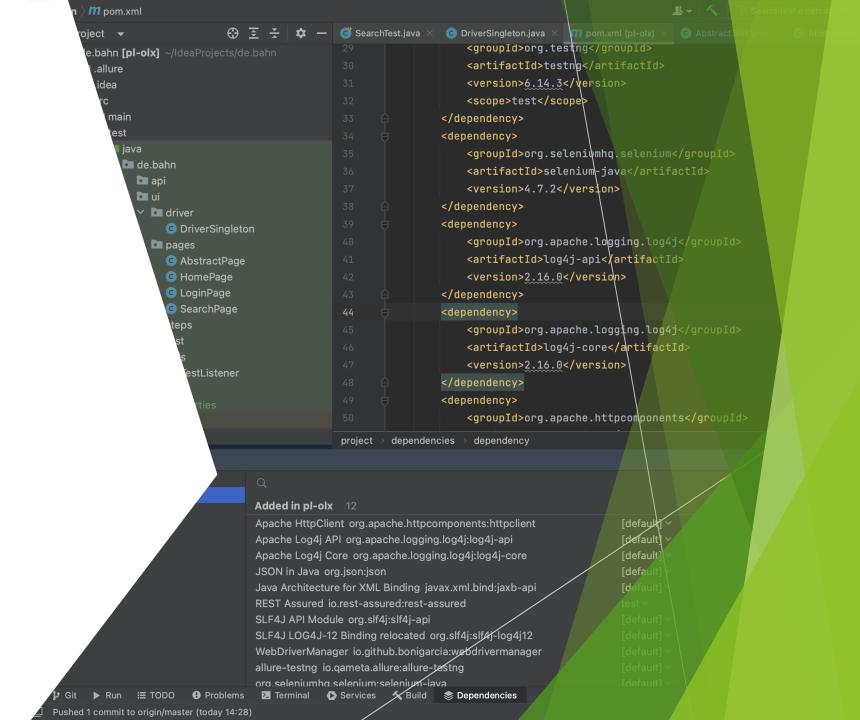
RestAssured

# Архитектура ТАF

- Selenium (Webdriver Singleton)
- Page (PageObject)
- Page Factory
- Steps
- Test

# Библиотеки и инструменты

- Selenium
- TestNG
- RestAssured
- Allure
- Surefire



de.bahn api **API TEST G** ApiTest **D**ui O DriverSingleton ✓ □ pages AbstractPage C HomePage PageObject Company of the com SearchPage ✓ steps Compare the com Steps SearchStep ✓ test AbstractTest **©** LoginTest **Tests** SearchTest ✓ utils TestListener allure.properties .gitignore

# Реализация проекта

- Webdriver Singleton
- PageObject
- Steps
- Test

```
age de.bahn.ui.driver;
 brt org.openga.selenium.WebDriver;
ort org.openga.selenium.chrome.ChromeDriver;
sages 😃 aleksei
blic class DriverSingleton {
 6 usages
 private static WebDriver driver;
 public static WebDriver getDriver() {
     if (driver == null) {
         driver = new ChromeDriver();
         driver.manage().window().maximize();
     return driver;
 public static void closeDriver() {
     driver.quit();
     driver = null;
```

# Webdriver Singleton

Используется паттерн
Singleton. Задача этого
шаблона ограничить количество
экземпляров класса в пределах
приложения.

```
jimport org.openqa.seleni∪m.WebElement;
5 usages 			 ≜ aleksei
public class LoginPage extends AbstractPage {
    @FindBy(xpath = "//input[@id='username']")
    private WebElement inputUsername;
    3 usages
    @FindBy(xpath = "//input[@id='password']")
    private WebElement inputPassword;
    @FindBy(xpath = "//button[@type='submit']")
    private WebElement buttonSubmit;
    @FindBy(xpath = "//div[@class='login-page__mess
    private WebElement messageError;
    public LoginPage fillLoginForm(String username
        fillForm(username, password);
        clickSubmit();
        return this;
    public Boolean existsInputUsernaem() {
        return inputUsername.isDisplayed();
    public Boolean existsInputPassword() { r
```

# Page object

Используется паттерн PageObject один из наиболее полезных и используемых архитектурных решений в автоматизации.

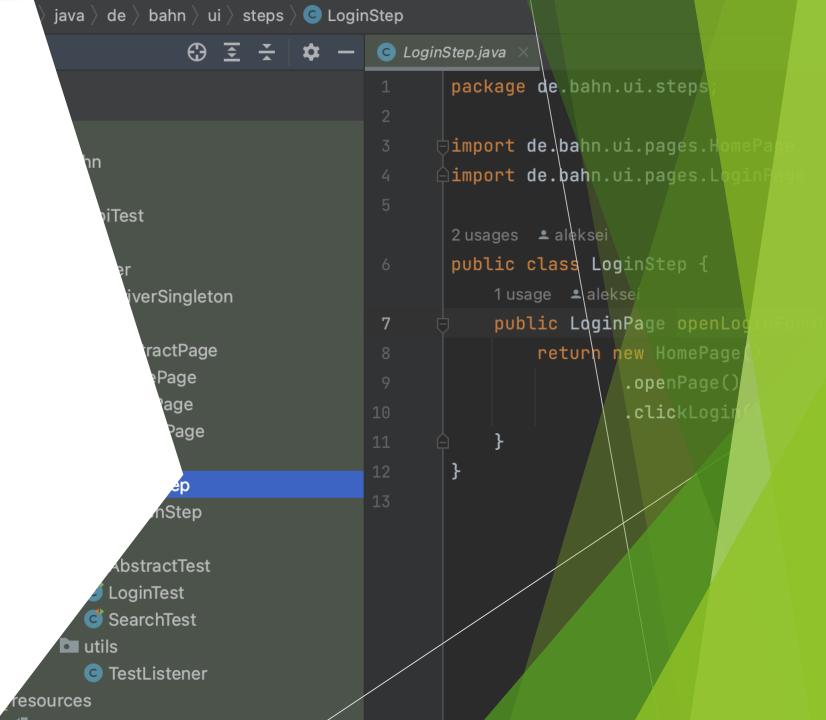
Данный шаблон проектирования помогает инкапсулировать работу с отдельными элементами страницы, что позволяет уменьшить количество кода и упростить его поддержку.

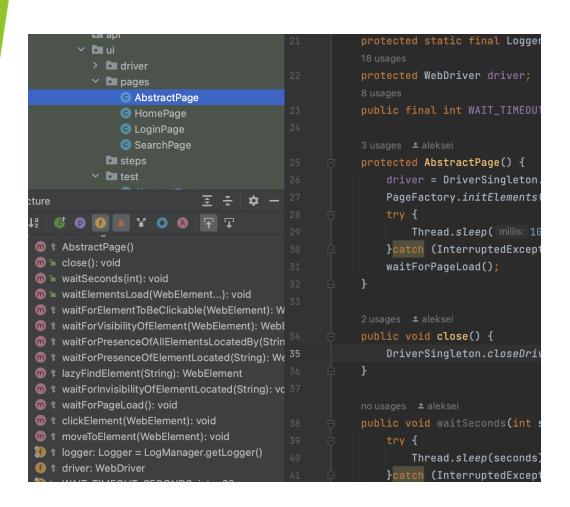
#### Основные преимущества:

- Разделение кода тестов и описания страниц
- Объединение всех действий по работе с веб-странией в одном месте

### Steps

В классых типа Stesp происходит объединение шагов (действий над элементами), реализуемых в коде в один метод, чтобы упростить реализацию логики тестов.





# BasePage

Класс AbstractPage описывающий базовые действия с элементами интерфейса веб-страницы

### **Page Factory**

Page Factory - это встроенная концепция фреймворка Page Object Model в Selenium Web Driver. Она хорошо оптимизирована и используется для инициализации объектов страницы или для создания объекта страницы в целом.

Page Factory инициализирует элементы класса страницы, не используя FindElement(s) вместо этого применяется аннотация <u>@FindBy</u>, служащая для поиска веб-элемент

```
public class SearchPage extends AbstractPage
   no usages
   private final String BASE_URL = "https://www.bahn.de/";
   4 usages
   @FindBy(xpath = "//input[@id='locS0']")
   private WebElement inputFrom;
   3 usages
   @FindBy(xpath = "//input[@id='locZ0']")
   private WebElement inputTo;
   3 usages
    @FindBy(xpath = "//select[@id='traveller_Nr']")
    private WebElement selectTravellers;
     indBy(xpath = "//input[@id='searchConnectionButton']")
       vate WebElement buttonSubmit;
        dBy(xpath = "//a[@class='flex-reset link']")
       vate WebElement linkSearch;
    @FindBy(xpath = "//div[@id='errormsg_S']")
   private WebElement errorMessgeFrom;
   @FindBy(xpath = "//div[@id='errormsg_Z']
   private WebElement errorMessgeTo;
   no usages
   @FindBy(xpath = "//div[@id='conTrave]
```

```
@Test
                       public void stationNameDataTest() {
                           String actual = "Zwingenberg (Bergst
                                    .accept(ContentType.JSON)
                                   .get( s: "/station-data/v2/
                                   .statusCode( i: 200)
                                   .body( s: "result[0].name
                       @Test
                       public void railwayStationsTest()
                           String actual = "Lietuva (Lita
                                    accept(ContentType.J.

✓ Default Suite

                                        2 min 34 sec

✓ ✓ bahn

✓ ✓ LoginTest

✓ existsInputPasswordTest

✓ existsInputUsernaemTest

         ✓ loginEmptyLoginAndPassw 1sec 611 m

✓ loginEmptyLoginTest

         ✓ loginEmptyPasswordTest

✓ loginNotCorrectLoginAndF 1sec 368

✓ SearchTest

✓ existsInputFromTest

✓ existsInputToTest

✓ searchCorrectFromAndToT 5 sec 8<sup>o</sup>

✓ searchErrorMessageFromAndTc 5

✓ searchErrorMessageFromTi1sec

✓ searchErrorMessageToTest

✓ searchTwoTravellersTest 2 sec
```

# Процесс тестирования

- 1. Узнаем ожидаемый результат.
- 2. Узнаем фактический результат.
- 3. Сравниваем ожидаемый и фактический результаты.

Источником ожидаемого результата является спецификация - детальное описание того, как должно работать ПО. Любой дефект представляет собой отклонение от спецификации. Важно обнаружить эти дефекты до того, как их найдут конечные пользователи.

# Результат UI тестирования

UI-тестирование помогает проверить большую часть действий пользователя, взаимодействие сервисов и компонентов.

Сегодня тестирование является неотъемлемой частью процесса производства программных продуктов.

- **Default Suite** 2 min bahn LoginTest existsInputPasswordTest ✓ existsInputUsernaemTest loginEmptyLoginAndPassw ✓ loginEmptyLoginTest ✓ loginEmptyPasswordTest ✓ loginNotCorrectLoginAndF1sec
  - ✓ SearchTest
    - existsInputFromTest
    - existsInputToTest
    - searchCorrectFromAndToT
    - ✓ searchErrorMessageFromAn
    - ✓ searchErrorMessageFromTe
    - ✓ searchErrorMessageToTest

```
no usages 🚨 aleksei
@Test
public void stationNameDataTest() {
    String actual = "Zwingenberg (Bergstr
    given() RequestSpecification
             .accept(ContentType.JSON)
             .get( s: "/station-data/v2/st
             .then() ValidatableResponse
             .assertThat()
             .statusCode( i: 200)
             .body( s: "result[0].name"
no usages 🚨 aleksei
@Test
public void railwayStationsTest() {
    String actual = "Lietuva (Litaus
    given() RequestSpecification
             .accept(ContentType.JSQ
             .get( s: "/api.railway-
             .then() ValidatableRespons
              assertThat()
```

# API тестирование

АРІ (программный интерфейс приложения) - это набор вызовов, при помощи которых приложение общается со своими частями.

**Тестирование** помогает убедиться, что программа выполняет поставленную перед ней цель и сможет корректно взаимодействовать с другими программами.

#### Реализация API тестов

Для тестирования используется библиотека REST Assured.

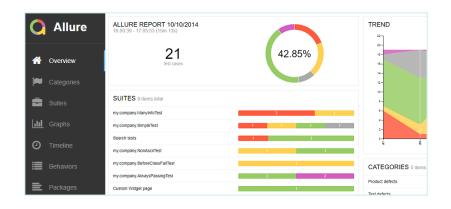
Посылаются запросы и проверяется корректность ответа.

#### **Default Suite**

- 🖍 bahn
  - ApiTest
    - connectingTimesTest

5 se

- ✓ railwayStationsTest
- ✓ stationIdDataTest
- stationNameDataTest
  - timetablesEvaTest
  - timetablesStationTest



#### Allure Framework

Allure Framework - популярный инструмент построения отчётов автотестов, упрощающий их анализ. Это гибкий и легкий инструмент, который позволяет получить не только краткую информацию о ходе выполнения тестов, но и предоставляет всем участникам производственного процесса максимум полезной информации из повседневного выполнения автоматизированных тестов.

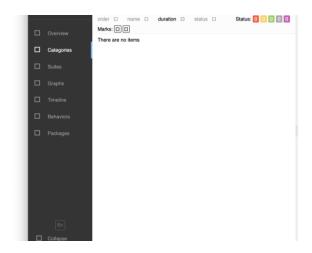
### Страница «Overview»



Overview является главной страницей Allure отчета.

Она имеет блочную структуру. Рассмотрим блоки, присутствующие на главной странице по умолчанию:

- 1. Блок ALLURE REPORT. Включает в себя дату и время прохождения теста, общее количество прогнанных кейс ов.
- а также диаграмму с указанием процента и количества успешных, упавших и сломавшихся в процессе выполн ения тестов.
- 2. Блок TREND. Показывает тренд прохождения тестов от сборки к сборке.
- 3. Блок SUITES. Показывает распределение результатов тестов по тестовым наборам. В данном случае тесты были распределены по пакетам.
- 4. Блок ENVIRONMENT. Показывает тестовое окружение, на котором запускались тесты. Данная информация п опадает в отчет из специального файла, расположенного в проекте.
- 5. Блок CATEGORIES. Показывает распределение неуспешно прошедших тестов по видам дефектов.
- 6. Блок FEATURES BY STORIES. Показывает распределение тестов по функционалу, который они проверяют.
- 7. Блок EXECUTORS. Показывает исполнителя текущей сборки. Если выполнение производилось на инструмент е CI (например, на Jenkins), то будет предоставлена информация о джобе и номере сборки.

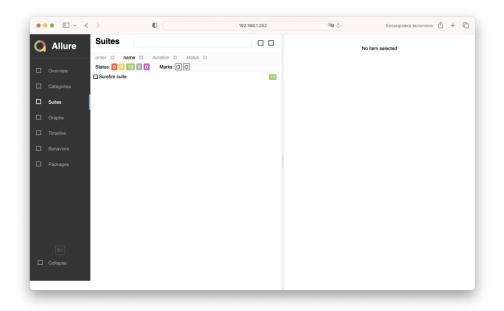


# Страница «Categories»

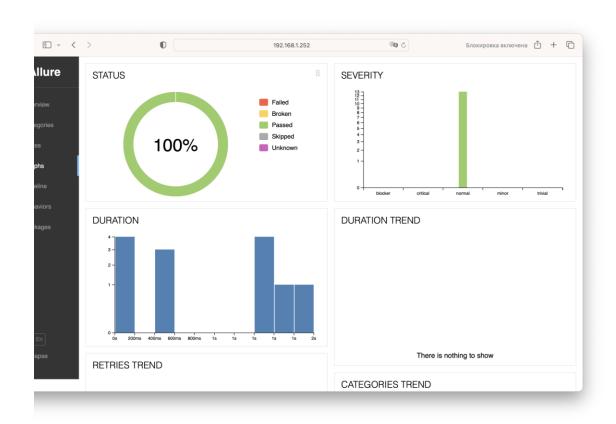
Данная страница предоставляет информацио о распределении дефектов по их видам.

# Страница «Suites»

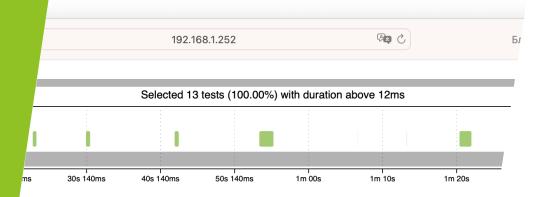
На данной странице представляется стандартное распределение выполнявшихся тестов по тестовым наборам или классам, в которых находятся тестовые методы.



# Страница «Graphs»



На этой странице можно получить информацию о тестовом прогоне в графическом виде: статус прогона, распределение тестов по их критичности, длительности прохождения, перезапусках, категориях дефектах и так далее.



# Страница «Timeline»

В случае запуска тестов в параллельном режиме:

Данная страница визуализирует временные рамки прохождения каждого теста.

llure

view

egories

tes

neline

al a ma

skayes

En

lapse

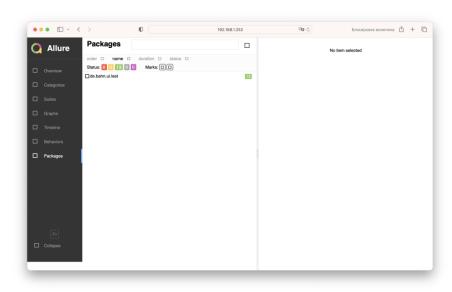
**Behaviors** duration □ status order 0 13 0 0 Marks: □ □ Status: 0 existsInputFromTest #8 existsInputPasswordTest #2 existsInputToTest #9 existsInputUsernaemTest #10 loginEmptyLoginAndPasswordTest #11 loginEmptyLoginTest #12 loginEmptyPasswordTest #13 loginNotCorrectLoginAndPasswordTest #3 searchCorrectFromAndToTest #4 searchErrorMessageFromAndToTest #5 searchErrorMessageFromTest #6 searchErrorMessageToTest searchTwoTravellersTest

12ms
28ms
25ms
1s 609ms
1s 544ms СТРАНИЦа
1s 596ms СТРАНИЦа
1s 528ms
4518 Chaviors>>

На данной странице тесты сгруппированы по проверяемому функционалу (Epic, Feature, Story).

# Страница «Packages»

На этой странице тесты сгруппированы по пакетам, в которых лежат тестовые классы.



#### All suites

#### suite

le name]

bs

er output d methods ological view

nethods, 19 passed ed methods (hide) connectingTimesTest existsInputFromTest existsInputPasswordTest

existsInputToTest

existsInputUsernaemTest

IoginEmptyLoginAndPasswordTest

loginEmptyLoginTest

■ loginEmptyPasswordTest

■ loginNotCorrectLoginAndPasswordTest

railwayStationsTest

searchCorrectFromAndToTest

■ searchErrorMessageFromAndToTest

■ searchErrorMessageFromTest

searchErrorMessageToTest

searchTwoTravellersTest

■ stationIdDataTest

■ timetablesEvaTest

■ timetablesStationTest

#### ☑ de.bahn.ui

existsInputFromTe

existsInputToTest

searchCorrectFro

searchErrorMessa

searchErrorMess

searchErrorMess

searchTwoTrave

#### ☑ de.bahr

existsInputPa

existsInputU

loginEmptyLo

loginEmptyI

loginEmpty

loginNotCo

#### ☑ de.b

connecti

railways

station

station

timeta

# Surefire

Предназначен для запуска тестов и генерации отчетов по результатам их выполнения.

#### Test results

1 suite

imes for Surefire suite

Total	running	time:	14	seconds
-------	---------	-------	----	---------

Number	Method	Class	Time (ms)
0	searchTwoTravellersTest	de.bahn.ui.test.SearchTest	1,828
1	searchCorrectFromAndToTest	de.bahn.ui.test.SearchTest	1,667
2	loginEmptyLoginAndPasswordTest	de.bahn.ui.test.LoginTest	1,504
3	connectingTimesTest	de.bahn.api.ApiTest	1,423
4	loginEmptyLoginTest	de.bahn.ui.test.LoginTest	1,341
5	login Not Correct Login And Password Test	de.bahn.ui.test.LoginTest	1,339
6	loginEmptyPasswordTest	de.bahn.ui.test.LoginTest	1,329
7	searchErrorMessageFromTest	de.bahn.ui.test.SearchTest	873
8	search Error Message From And To Test	de.bahn.ui.test.SearchTest	645
9	searchErrorMessageToTest	de.bahn.ui.test.SearchTest	587
10	railwayStationsTest	de.bahn.api.ApiTest	569
11	timetablesEvaTest	de.bahn.api.ApiTest	490
12	timetablesStationTest	de.bahn.api.ApiTest	455
13	stationIdDataTest	de.bahn.api.ApiTest	386
14	stationNameDataTest	de.bahn.api.ApiTest	385

# **Times**

Отображает отчет о времени прохождения тестов.

#### Default Suite ✓ ✓ bahn ✓ ✓ LoginTest ✓ existsInputPasswordTest ✓ existsInputUsernaemTest ✓ loginEmptyLoginAndPassw ✓ loginEmptyLoginTest ✓ loginEmptyPasswordTest ✓ loginNotCorrectLoginAnd ✓ ✓ SearchTest existsInputFromTest existsInputToTest searchCorrectFromAnd ✓ searchErrorMessageFr ✓ searchErrorMessageFi searchErrorMessageTell import de.bahn.ui.pages.HomePage import de.bahn.ui.pages.LoginPag public class LoginStep { public LoginPage openLogi .openPage() rchPage inStep rchStep tractTest

#### Заключение

В результате проделанной работы был покрытый тест кейсами UI и API интерфейс.

Были осуществлены прогоны тестов и созданы отчеты по результатам прогонов.

Качественное тестирование помогает своевременно выявлять и исправлять ошибки, тем самым уменьшая риски и затраты на разработку программного обеспечения. При автоматизации тестирования скорость и качество тестирования может повыситься, что приведет к еще большему снижению издержек и повышению качества.