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 API

- 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

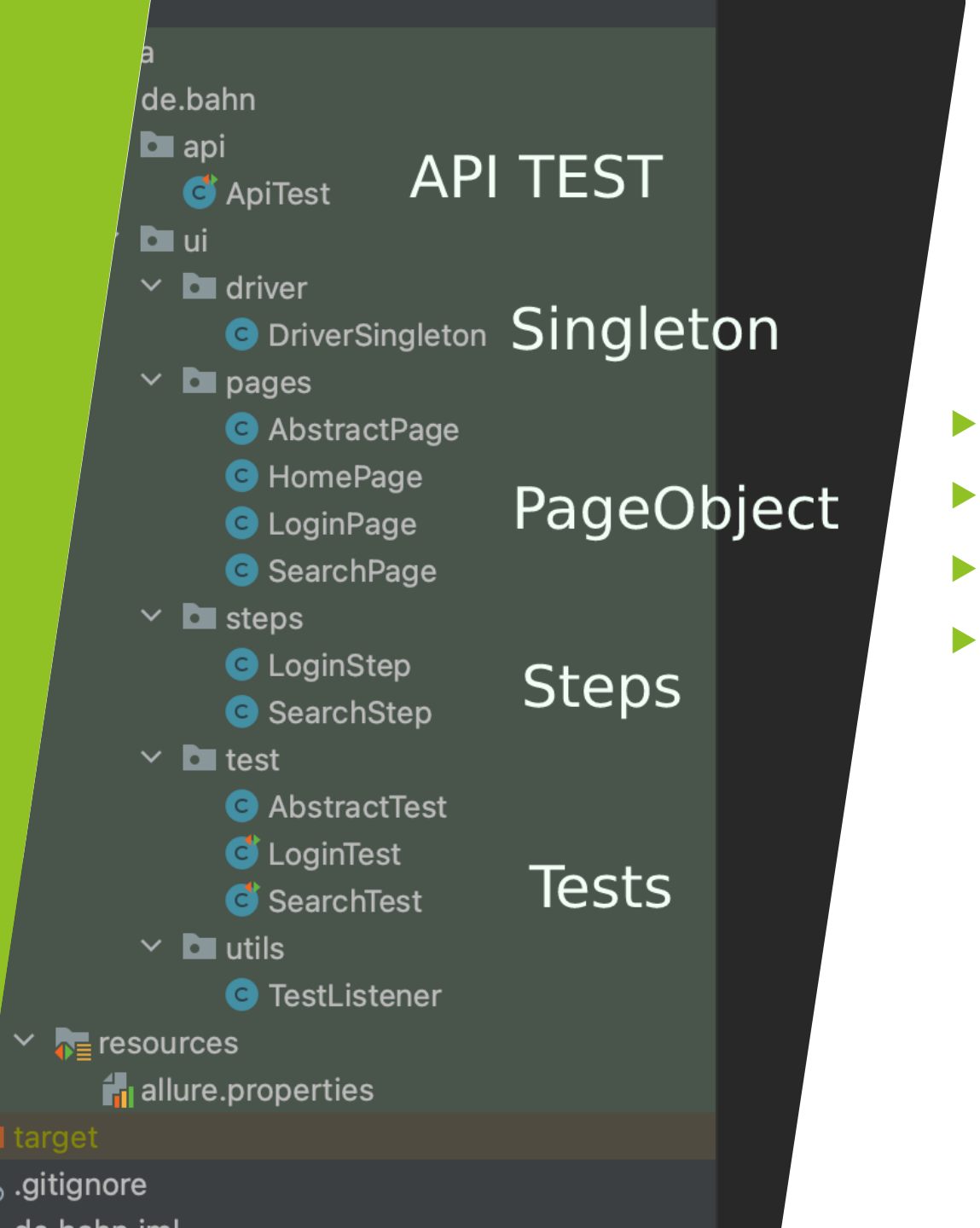
RestAssured

Архитектура TAF

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

Библиотеки и инструменты

- Selenium
- TestNG
- RestAssured
- Allure
- Surefire



Реализация проекта

- Webdriver Singleton
- PageObject
- Steps
- Test

```
age de.bahn.ui.driver;

ort org.openqa.selenium.WebDriver;
ort org.openqa.selenium.chrome.ChromeDriver;

sages  aleksei
blic class DriverSingleton {
    6 usages
    private static WebDriver driver;

    2 usages  aleksei
    public static WebDriver getDriver() {
        if (driver == null) {
            driver = new ChromeDriver();
            driver.manage().window().maximize();
        }
        return driver;
    }

    1 usage  aleksei
    public static void closeDriver() {
        driver.quit();
        driver = null;
    }
}
```

Webdriver Singleton

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

```java
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

5 usages  aleksei
public class LoginPage extends AbstractPage {
    3 usages
    @FindBy(xpath = "//input[@id='username']")
    private WebElement inputUsername;
    3 usages
    @FindBy(xpath = "//input[@id='password']")
    private WebElement inputPassword;
    1 usage
    @FindBy(xpath = "//button[@type='submit']")
    private WebElement buttonSubmit;
    1 usage
    @FindBy(xpath = "//div[@class='login-page__mess
    private WebElement messageError;

    4 usages  aleksei
    public LoginPage fillLoginForm(String username
        fillForm(username, password);
        clickSubmit();
        return this;
    }

    1 usage  aleksei
    public Boolean existsInputUsernaem() {
        return inputUsername.isDisplayed();
    }

    1 usage  aleksei
    public Boolean existsInputPassword() { re
```

Page object

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

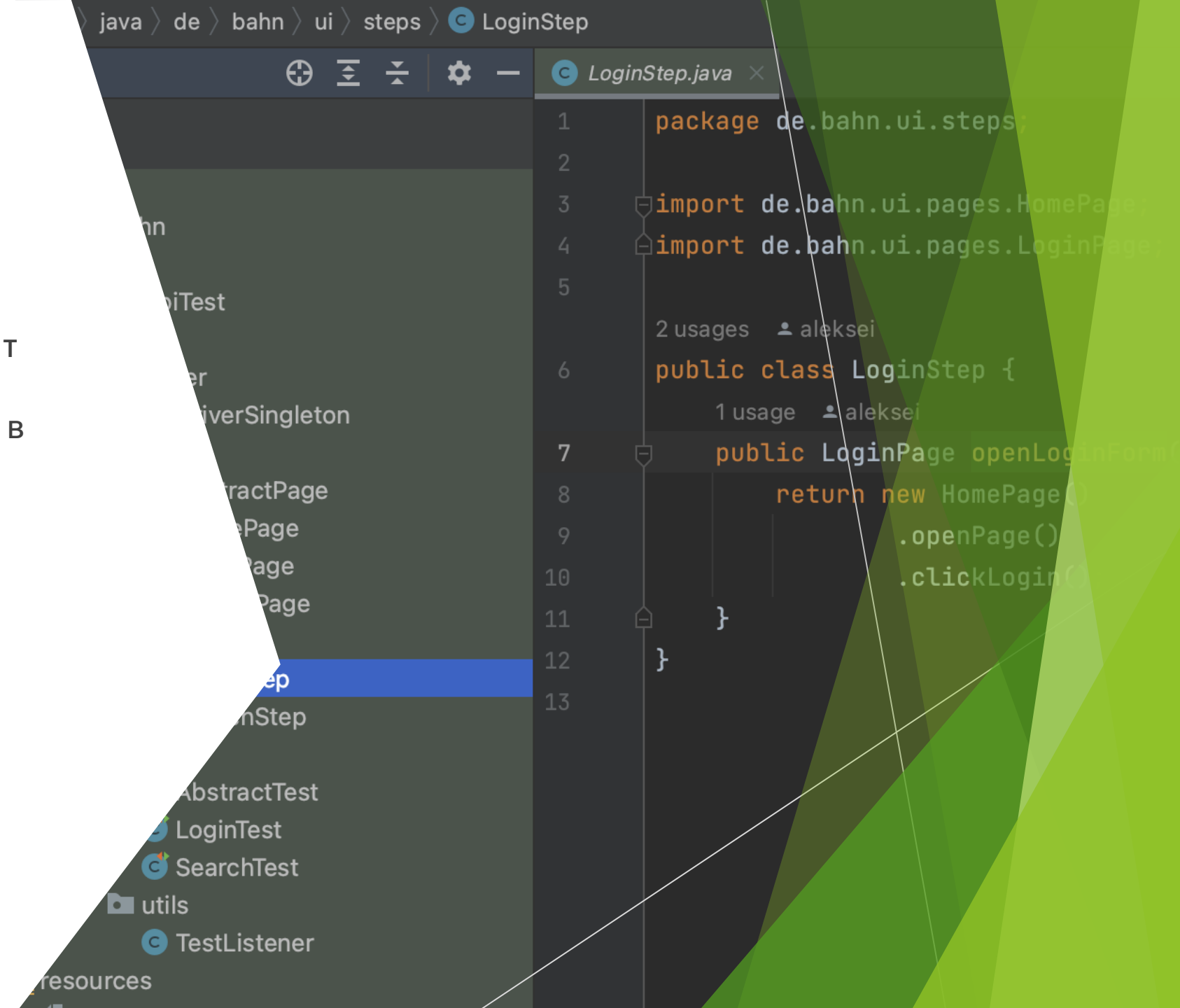
Данный шаблон проектирования помогает инкапсулировать работу с отдельными элементами страницы, что позволяет уменьшить количество кода и упростить его поддержку.

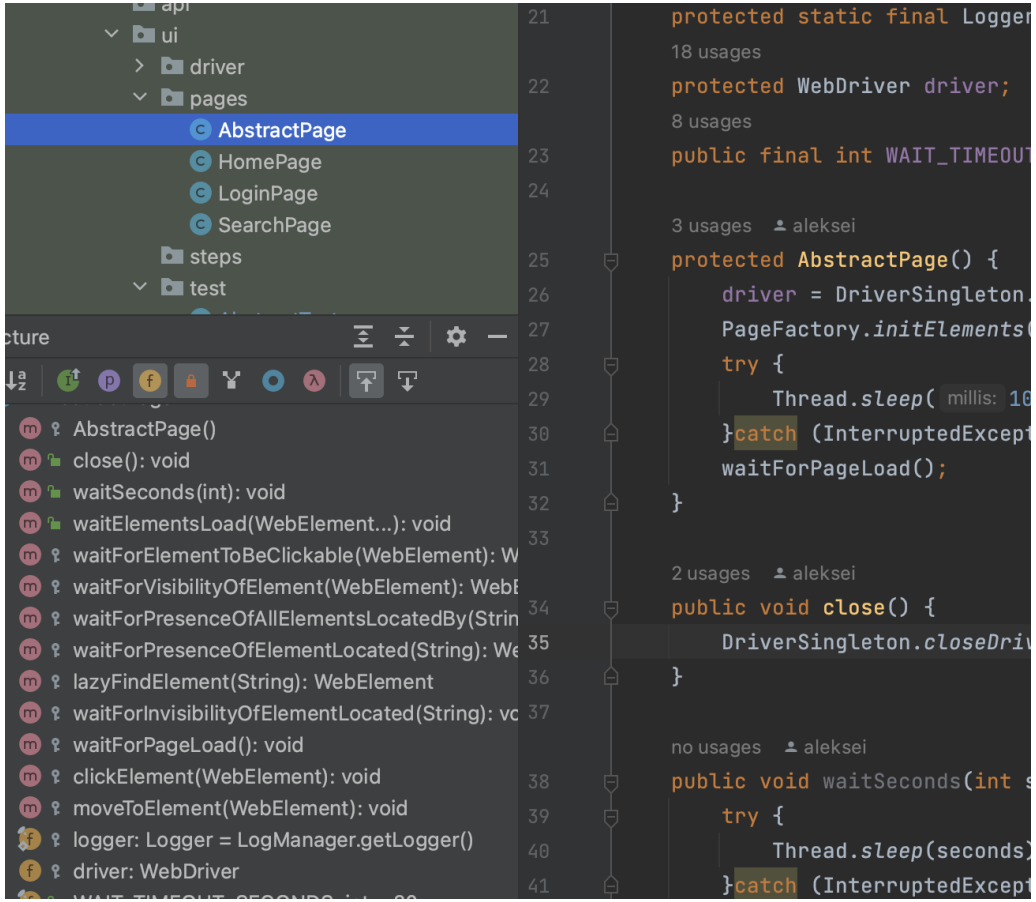
Основные преимущества:

- Разделение кода тестов и описания страниц
- Объединение всех действий по работе с веб-страницей 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.





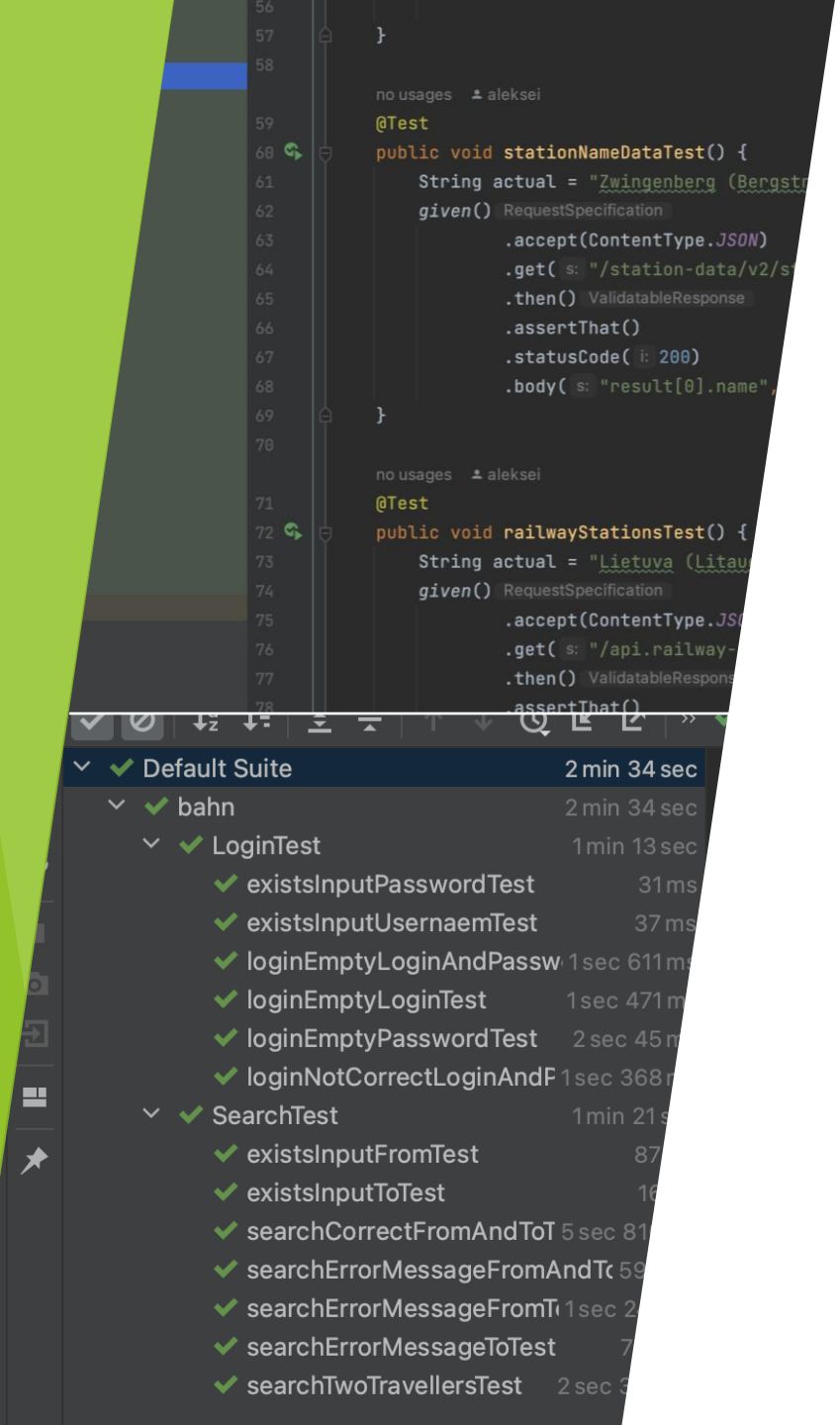
BasePage

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 @FindBy, служащая для поиска веб-элемент



Процесс тестирования

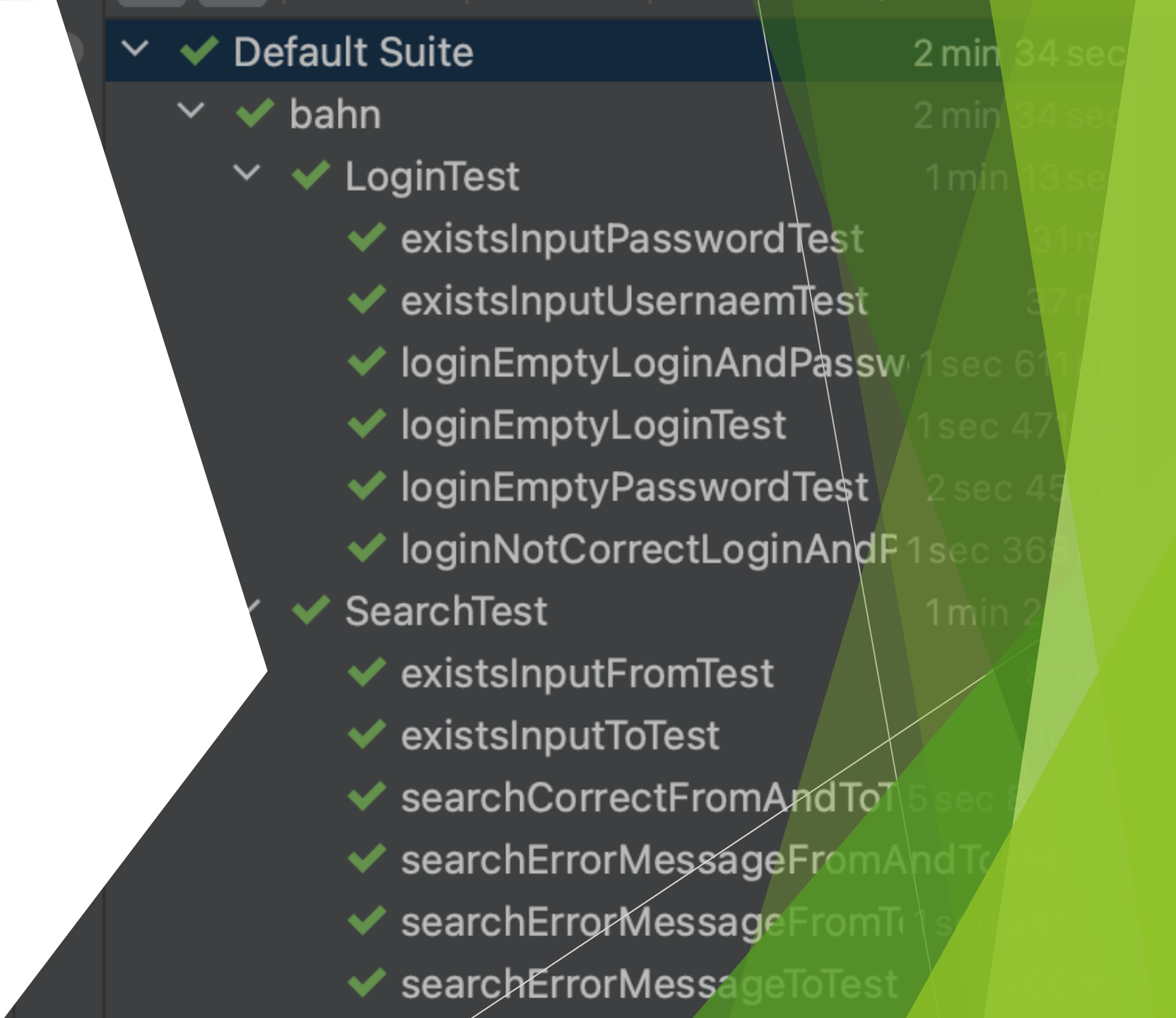
1. Узнаем ожидаемый результат.
2. Узнаем фактический результат.
3. Сравниваем ожидаемый и фактический результаты.

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.



API тестирование

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

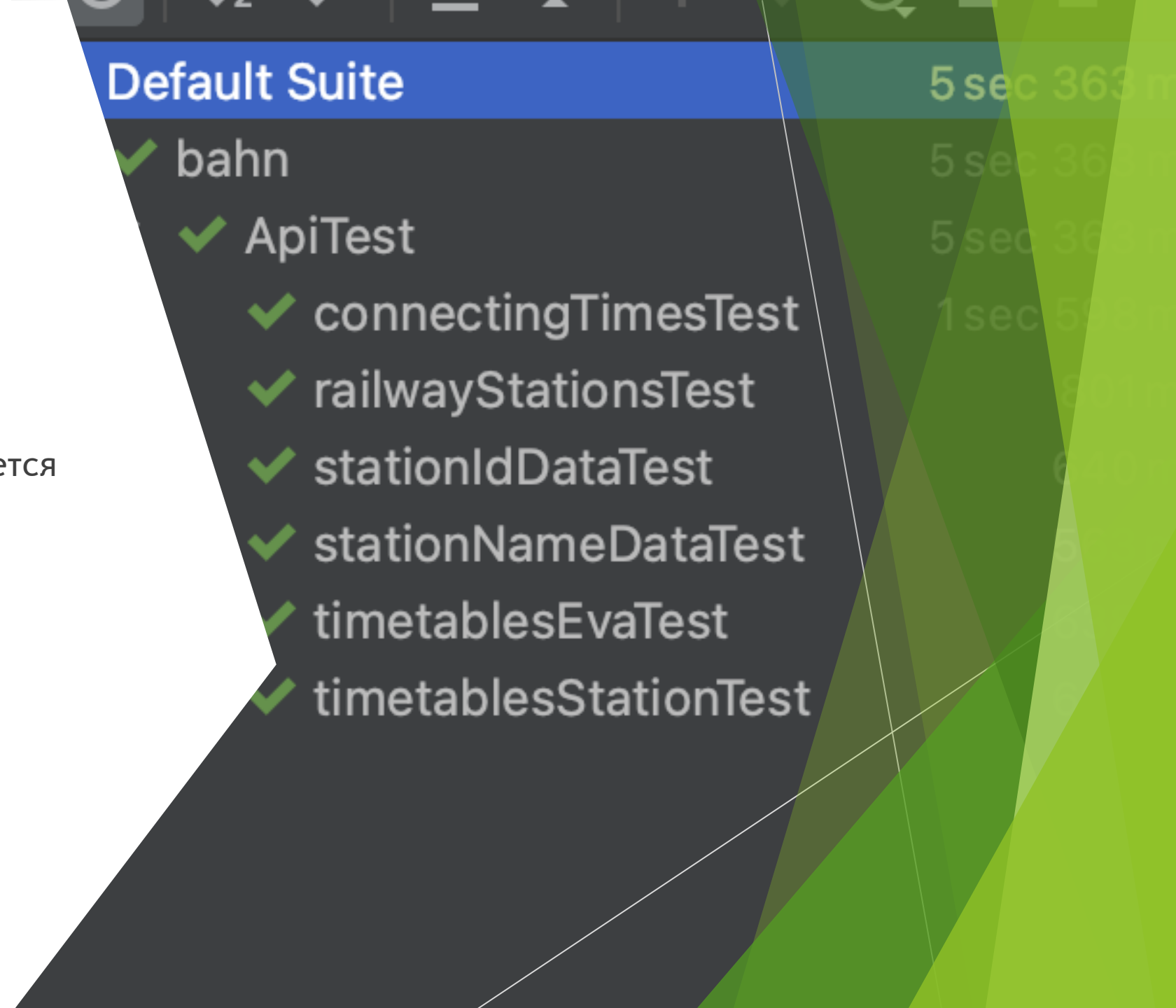
Тестирование помогает убедиться, что программа выполняет поставленную перед ней цель и сможет корректно взаимодействовать с другими программами.

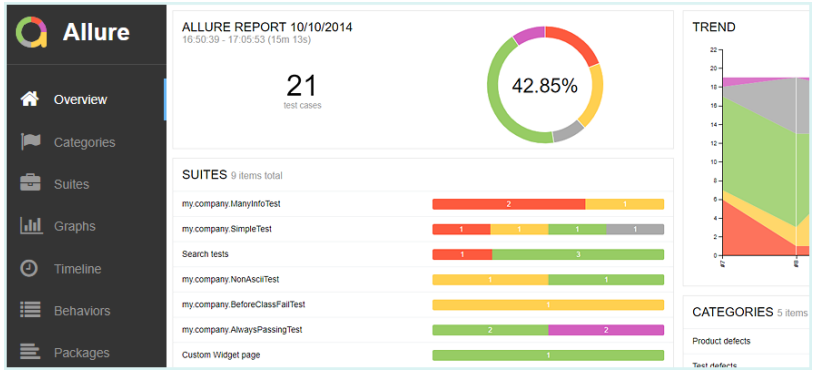
Реализация API тестов

Для тестирования используется

библиотека REST Assured.

Посылаются запросы и проверяется корректность ответа.

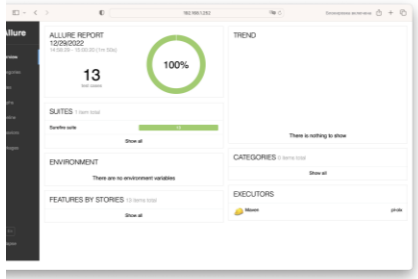




Allure Framework

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 кейсов,
а также диаграмму с указанием процента и количества успешных, упавших и сломавшихся в процессе выполнения тестов.

2. Блок TREND. Показывает тренд прохождения тестов от сборки к сборке.

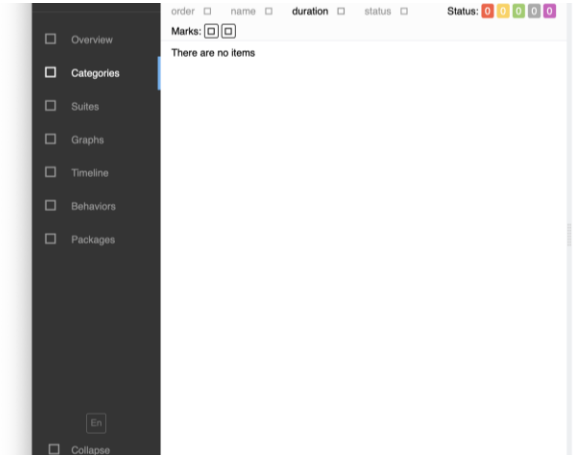
3. Блок SUITES. Показывает распределение результатов тестов по тестовым наборам.
В данном случае тесты были распределены по пакетам.

4. Блок ENVIRONMENT. Показывает тестовое окружение, на котором запускались тесты. Данная информация попадает 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 инструменте CI (например, на Jenkins), то будет предоставлена информация о джобе и номере сборки.

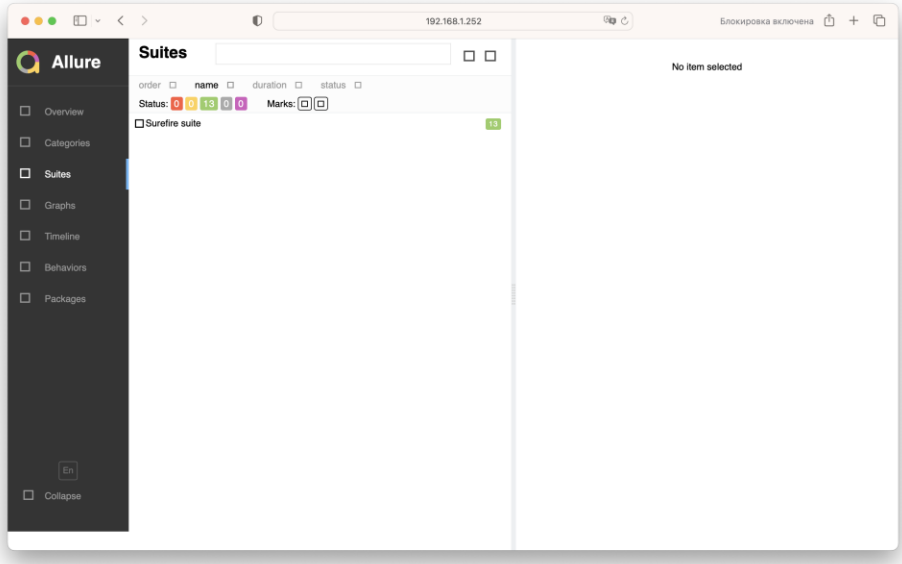


Страница «Categories»

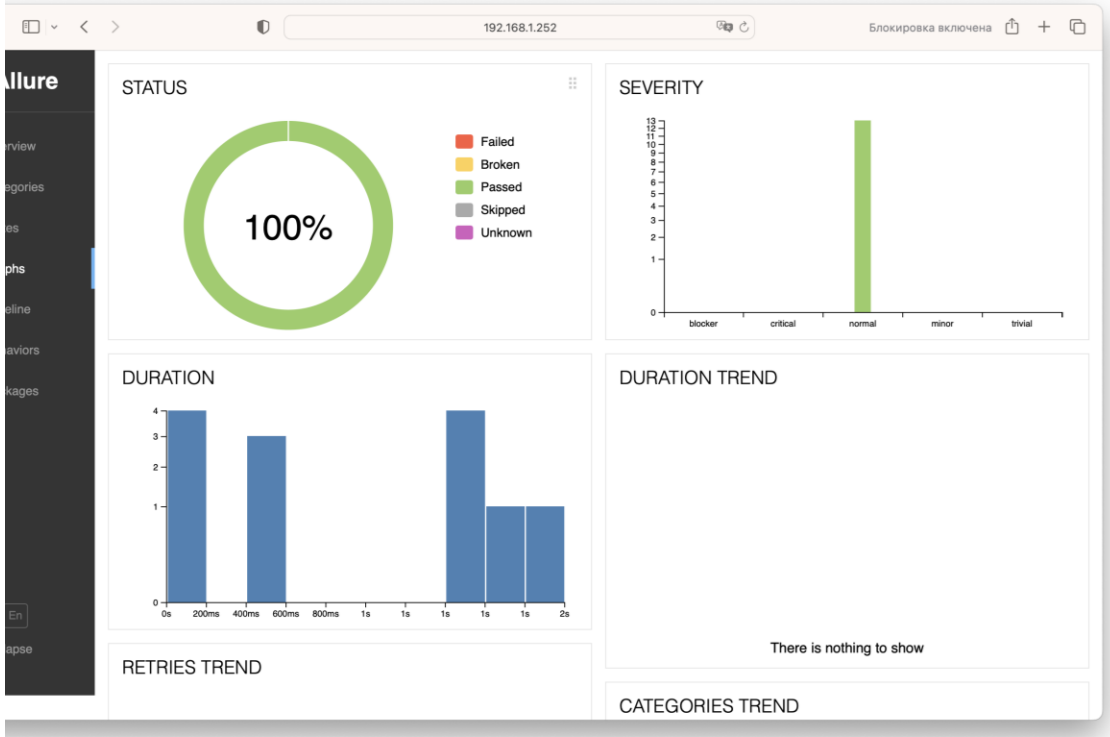
Данная страница предоставляет информацио о распределении дефектов по их видам.

Страница «Suites»

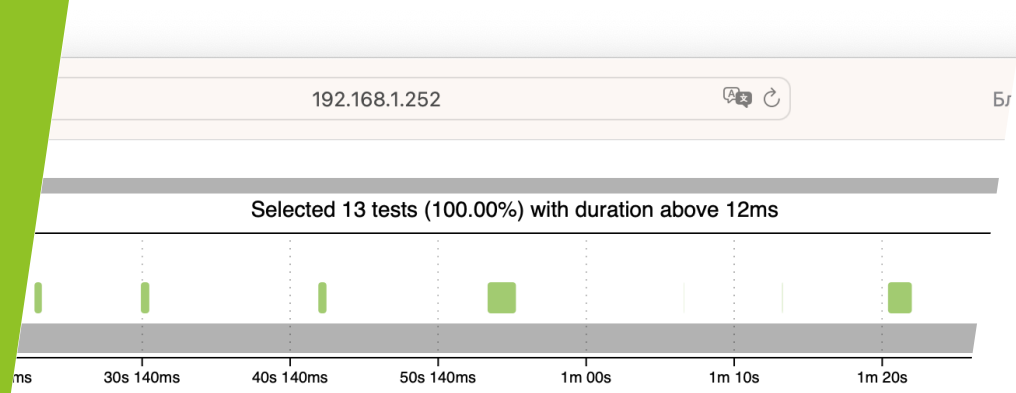
На данной странице представляется стандартное распределение выполнявшихся тестов по тестовым наборам или классам, в которых находятся тестовые методы.



Страница «Graphs»



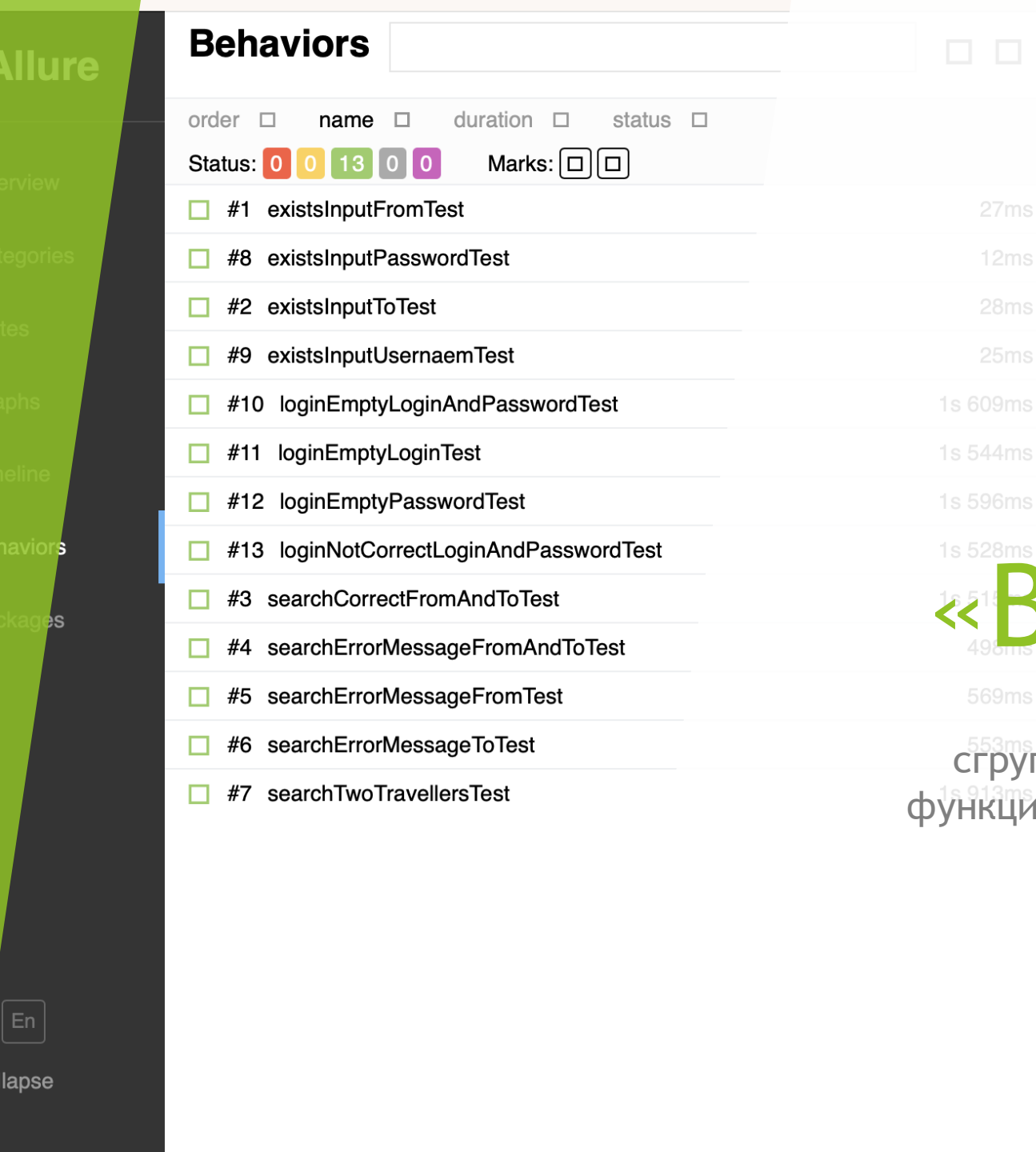
На этой странице можно получить информацию о тестовом прогоне в графическом виде: статус прогона, распределение тестов по их критичности, длительности прохождения, перезапусках, категориях дефектах и так далее.



Страница «Timeline»

В случае запуска тестов в параллельном режиме:

Данная страница визуализирует временные рамки прохождения каждого теста.

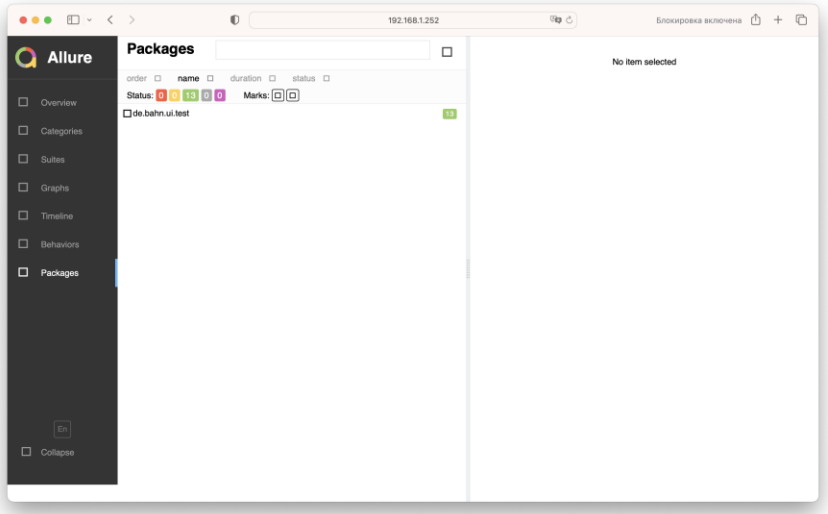


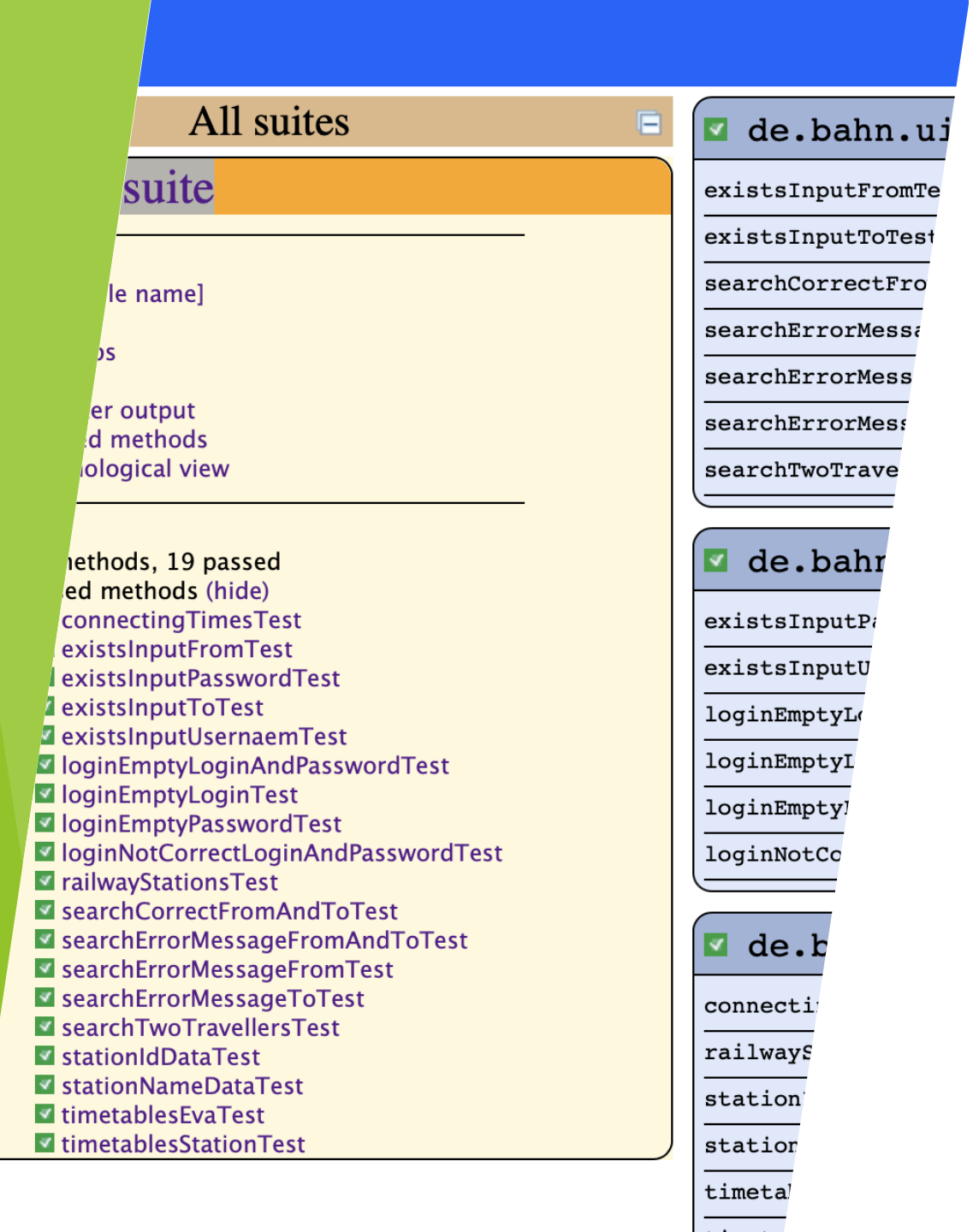
Страница «Behaviors»

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

Страница «Packages»

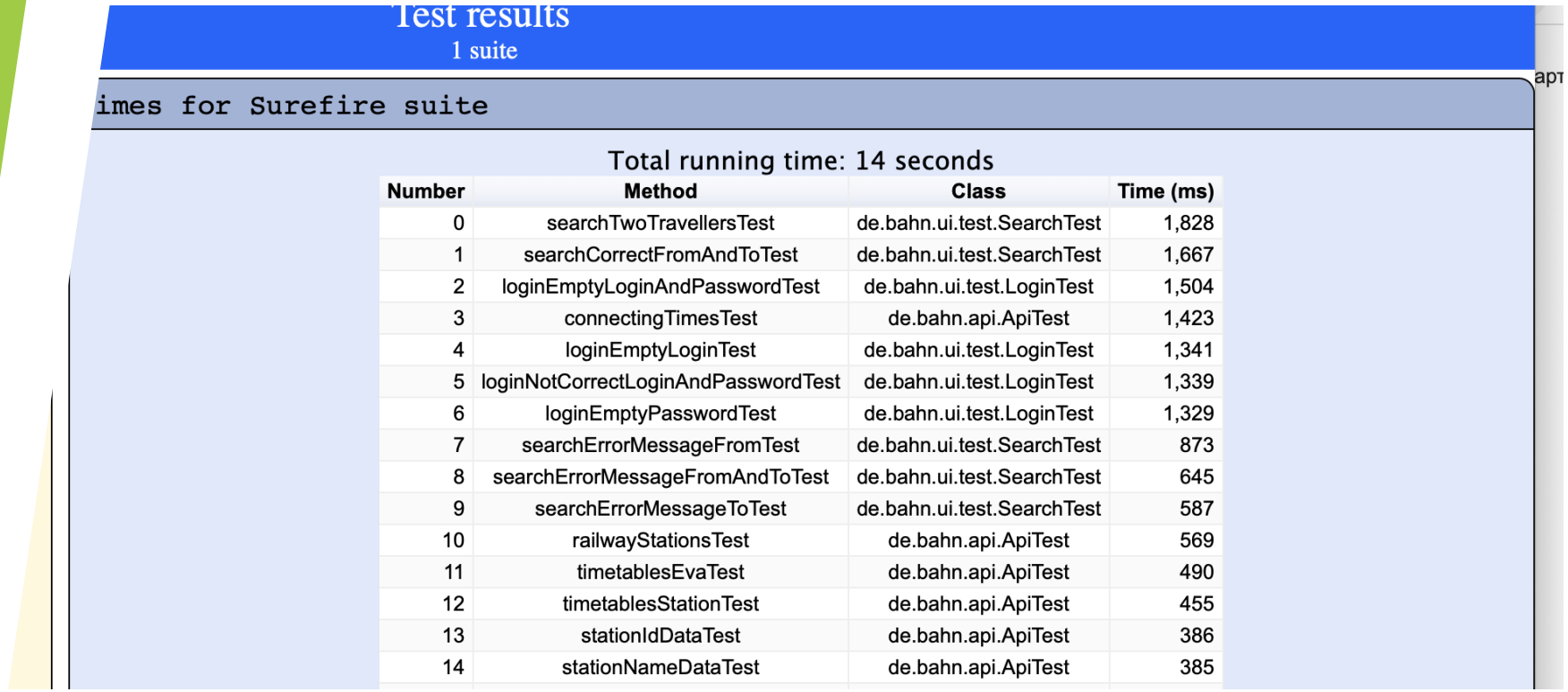
На этой странице тесты сгруппированы по пакетам, в которых лежат тестовые классы.





Surefire

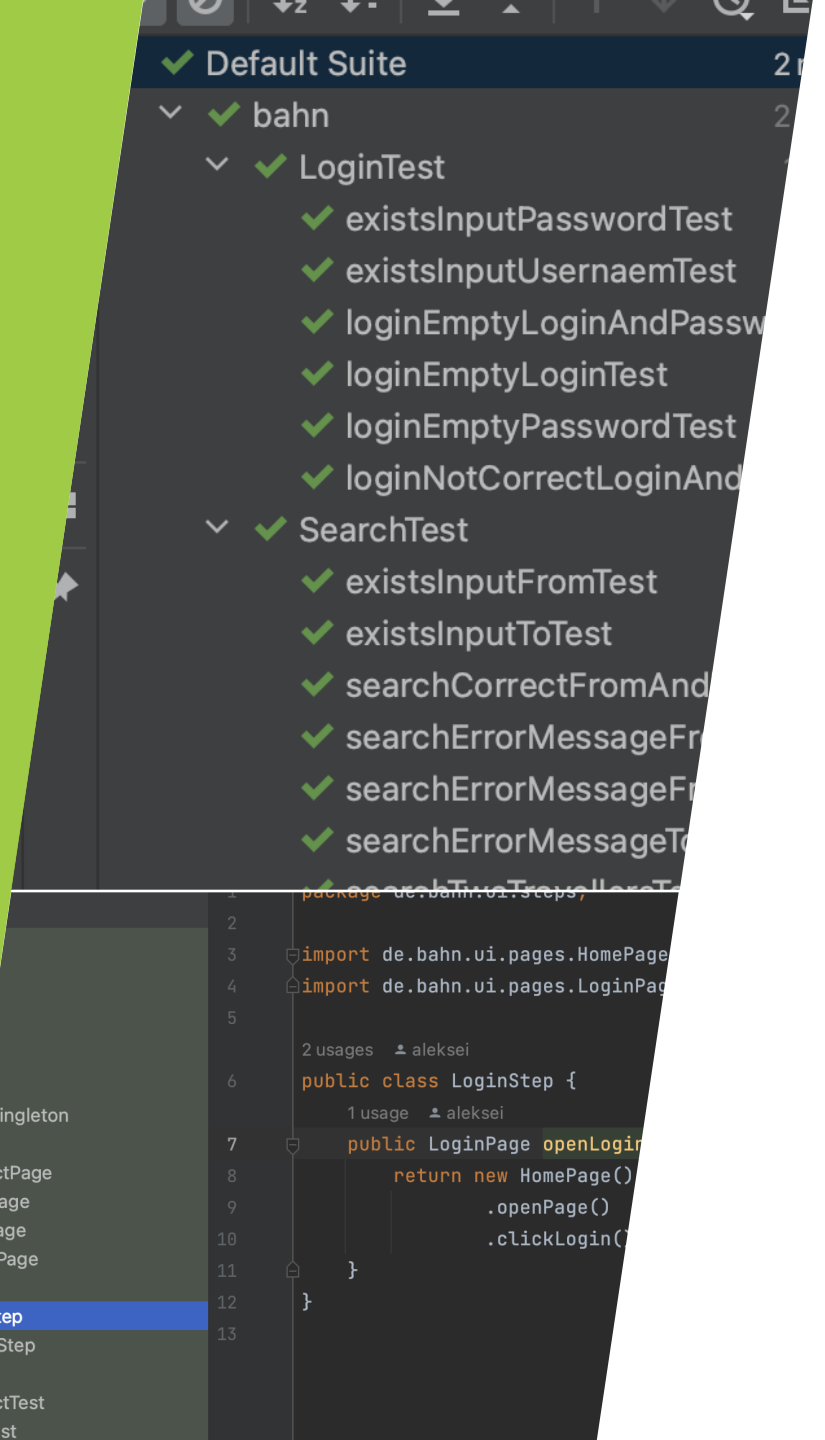
Предназначен для запуска тестов и генерации отчетов по результатам их выполнения.


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	loginNotCorrectLoginAndPasswordTest	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	searchErrorMessageFromAndToTest	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.



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.