

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОКОГНИТИВНЫХ ТЕХНОЛОГИЙ

Автоматизация тестирования ПО

Расчёто-пояснительная записка
курсового проекта по дисциплине
«Веб-разработка»
студент группы 221-361
Дубровских Никита Евгеньевич.

Преподаватели:
асс. В.С. Можначёв

Москва, 2024 г.

Введение

Цель курсового проекта:

- изучение существующих принципов и подходов к автоматизации тестирования;
- приобретение навыков и изучение методов автоматизированного тестирования ПО в процессе выполнения заданий курсовой работы;
- получение навыков использования на начальном профессиональном уровне популярных средств автоматизации тестирования, формирования и выполнения автоматизированных тестов.

Задачи курсового проекта

- изучение материалов лекционных и практических занятий из курса LMS «Введение в автоматизированное тестирование»;
- выполнение заданий по тестированию фронтенда — тестирование "LambdaTest Sample App", тестирование страницы расписания на сайте Мосполитеха, тестирование Яндекс.Маркета;
- выполнение индивидуального задания (фронтенд);
- выполнение заданий по тестированию бэкенда.

В проекте были использованы следующие технологии и инструменты:

- Java 21;
- Junit 5.10.2;
- Selenium WebDriver 4.20.0;
- Rest-Assured 5.4.0;
- Allure 2.26.0;
- Maven 3.9.7;
- Apache Maven Wrapper.

Структура проекта

Было принято решение поделить курсовой проект на 2 Maven проекта (Рисунок 1), разделяющие задания по тестированию фронтенда и бэкенда. Таким образом для каждого вида тестирования есть свой pom.xml файл, содержащий соответствующий набор библиотек (Рис. 1).

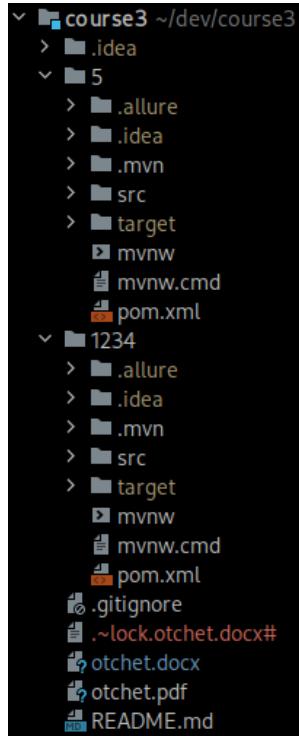


Рис. 1. Структура папок проекта

Для того что бы убедиться, что проект компилируется без ошибок у всех пользователей, был использован инструмент «Apache Maven Wrapper». Так, пользователям и разработчикам не нужно устанавливать версию Maven, используемую в проекте, в свою операционную систему. Вместо этого пользователь может использовать исполняемый файл внутри проекта.

Для тестирование фронтенда был использован один из наиболее полезных и используемых архитектурных решений в автоматизации — паттерн «PageObject». Таким образом, каждое задание имеет пакет «pages», содержащий Java классы с реализацией «PageObject» для каждой страницы (Рис. 2), и описывающих конкретную страницу на тестируемом сайте.

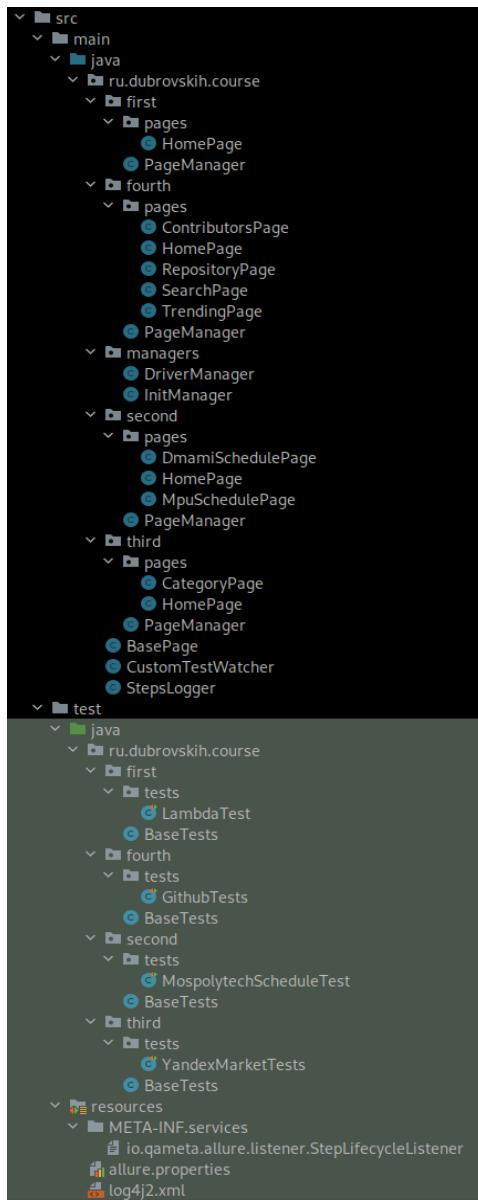


Рис. 2. Файловая структура проекта

Вспомогательные классы находятся в пакете «managers». Они реализованы через паттерн проектирования «Singleton», для того что бы гарантировать единственность экземпляра класса, что позволяет например гарантировать, что во всем проекте используется один WebDriver, что в любом месте, где он использован, он будет проинициализирован:

```

public class DriverManager {
    private static DriverManager instance;
    private WebDriver driver;

    private DriverManager() {
    }

    public static DriverManager getInstance() {
        if (instance == null) {
            instance = new DriverManager();
        }

        return instance;
    }

    public WebDriver getDriver() {
        if (driver == null) {
            driver = new FirefoxDriver();
        }

        return driver;
    }

    void quit() {
        if (driver != null) {
            driver.quit();
            driver = null;
        }
    }
}

```

В классе InitManager происходит первоначальная инициализация WebDriver, а также его первоначальная настройка — максимизация окна браузера Selenium:

```

public class InitManager {

    private static final DriverManager driverManager = DriverManager.getInstance();

    public static void init() {
        driverManager.getDriver().manage().window().maximize();
    }

    public static void quit() {
        driverManager.quit();
    }
}

```

}

Для каждого тестируемого сайта есть свой класс «PageManager», который также реализован через паттерн «Singleton», и отвечает за получение другими классами единственных, проинициализированных, экземпляров страниц. Так например класс «PageManager» для тестирования расписания сайта Московского Политеха содержит поля, являющиеся экземплярами классов тестируемых страниц, а также методы для их получения:

```
public class PageManager {  
    private static PageManager pageManager;  
    private HomePage homePage;  
    private MpuSchedulePage mpuSchedulePage;  
    private DmamiSchedulePage dmamiSchedulePage;  
  
    private PageManager() {  
    }  
  
    public static PageManager getInstance() {  
        if (pageManager == null) {  
            pageManager = new PageManager();  
        }  
  
        return pageManager;  
    }  
  
    public HomePage getHomePage() {  
        if (homePage == null) {  
            homePage = new HomePage();  
        }  
  
        return homePage;  
    }  
  
    public MpuSchedulePage getMpuSchedulePage() {  
        if (mpuSchedulePage == null) {  
            mpuSchedulePage = new MpuSchedulePage();  
        }  
  
        return mpuSchedulePage;  
    }  
  
    public DmamiSchedulePage getDmamiSchedulePage() {  
        if (dmamiSchedulePage == null) {  
            dmamiSchedulePage = new DmamiSchedulePage();  
        }  
    }  
}
```

```

    }

    return dmamiSchedulePage;
}
}
}
```

Вспомогательными классами также являются «BasePage», «CustomTestWatcher», «StepsLogger». От класса «BasePage» наследуются все классы реализующие паттерн «PageObject». Он содержит поле «driverManager», хранящее экземпляр WebDriver, устанавливает время, отведенное для поиска элемента на странице, а также содержит вспомогательные методы ожидания появления элемента на странице:

```

public class BasePage {
    protected final DriverManager driverManager = DriverManager.getInstance();
    protected WebDriverWait wait;
    protected JavascriptExecutor js = (JavascriptExecutor) driverManager.getDriver();

    public BasePage() {
        resetDriverWait();
        PageFactory.initElements(driverManager.getDriver(), this);
    }

    protected void resetDriverWait() {
        wait = new WebDriverWait(driverManager.getDriver(), Duration.ofSeconds(5),
Duration.ofSeconds(1));
    }

    protected WebElement waitUntilElementIsVisible(WebElement element) {
        return wait.until(ExpectedConditions.visibilityOf(element));
    }

    protected List<WebElement> waitUntilElementsIsVisible(List<WebElement> elements) {
        return wait.until(ExpectedConditions.visibilityOfAllElements(elements));
    }

    protected WebElement scrollToElementJs(WebElement element) {
        js.executeScript("arguments[0].scrollIntoView(true);", element);
        return element;
    }

    protected WebElement waitUntilElementToBeClickable(WebElement element) {
        return wait.until(ExpectedConditions.elementToBeClickable(element));
    }
}
```

Класс «CustomTestWatcher» отвечает за сохранение скриншота проваленного теста в папку проекта и в Allure-отчёт:

```
public class CustomTestWatcher implements TestWatcher {

    private final DriverManager driverManager = DriverManager.getInstance();

    @Override
    public void testAborted(ExtensionContext context, Throwable cause) {
        takeAndSaveScreenshot(context);
    }

    @Override
    public void testFailed(ExtensionContext context, Throwable cause) {
        takeAndSaveScreenshot(context);
    }

    private void takeAndSaveScreenshot(ExtensionContext context) {
        String fileName = generateScreenshotFileName(context);
        byte[] screenshot = takeScreenshot();
        saveScreenshot(fileName, screenshot);
        Allure.getLifecycle().addAttachment("failure screenshot", "image/png", "png", screenshot);
    }

    private byte[] takeScreenshot() {
        WebDriver driver = driverManager.getDriver();
        return ((TakesScreenshot) driver).getScreenshotAs(OutputType.BYTES);
    }

    private void saveScreenshot(String fileName, byte[] screenshot) {
        Path screenshotsPath;
        try {
            screenshotsPath = createScreenshotsDirectory();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        Path screenshotPath = screenshotsPath.resolve(fileName);
        File file = new File(screenshotPath.toString());

        file.getParentFile().mkdirs();
        try {
            file.createNewFile();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```

}

try (FileOutputStream outputStream = new FileOutputStream(screenshotPath.toFile())) {
    outputStream.write(screenshot);
} catch (IOException e) {
    throw new RuntimeException(e);
}
}

private Path createScreenshotsDirectory() throws IOException {
    String strClassesPath =
CustomTestWatcher.class.getProtectionDomain().getCodeSource().getLocation().getPath();
    Path classesPath = Paths.get(strClassesPath);
    Path targetPath = Paths.get(classesPath.getParent().toString());
    return targetPath.resolve("failure-screenshots" + File.separator);
}

private String generateScreenshotFileName(ExtensionContext context) {
    Date date = Calendar.getInstance().getTime();
    DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd_hh:mm:ss");
    String strDate = dateFormat.format(date);

    String fileName = strDate + " " + context.getDisplayName().replace("/", "|") + ".png";

    return fileName.replace(" ", "_");
}
}
}

```

Класс «StepsLogger» отвечает за логирование текущего шага тестирования в консоль:

```

public class StepsLogger implements StepLifecycleListener {

    private static final Logger logger = LogManager.getLogger();

    private void logStep(StepResult stepResult) {
        if (Objects.equals(stepResult.getName(), "step")) {
            return;
        }

        ThreadContext.put("name", stepResult.getName());
        ThreadContext.put("stage", stepResult.getStage().toString());

        Status status = stepResult.getStatus();
        ThreadContext.put("status", Objects.isNull(status) ? "null" : status.toString());

        logger.info("{} {} {}", stepResult.getName(), stepResult.getStage(), status);
    }
}

```

```

    ThreadContext.clearAll();
}

@Override
public void afterStepUpdate(StepResult result) {
    logStep(result);
}

@Override
public void afterStepStop(StepResult result) {
    logStep(result);
}
}

```

В папке test/resources содержатся файлы настройки для «Allure» и «Log4j2»:

```

<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="DEBUG">
    <Appenders>
        <Console name="LogToConsole" target="SYSTEM_OUT">
            <PatternLayout
                pattern="%d{HH:mm:ss.SSS} [%style{ %level }{ blue }] - %-75X{name}
                %style{ %-10X{stage} }{yellow} %style{ %X{status} }{green} %n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="info">
            <AppenderRef ref="LogToConsole"/>
        </Root>
    </Loggers>
</Configuration>

```

От класса «BaseTests» наследуются все тесты. Он отвечает за инициализацию и выход WebDriver перед каждым тестом, а также содержит соответствующий сайту «PageManager».

```

public class BaseTests {

    protected PageManager pageManager = PageManager.getInstance();

    @BeforeAll
    public static void beforeAll() {
        InitManager.init();
    }
}

```

```
@AfterAll
public static void afterAll() {
    InitManager.quit();
}
```

Тестирование списка дел "LambdaTest Sample App"

Для данной задачи использован следующий тест-кейс описанный в таблице 1:

Таблица 1. LambdaTest Sample App тест-кейс

| Шаг | Ожидаемый результат |
|---|--|
| Перейти по ссылке: https://lambdatest.github.io/sample-todo-app/ | На странице присутствует заголовок “LambdaTest Sample App”. |
| 2. Проверить, что присутствует текст: “5 of 5 remaining” | |
| 3. Проверить, что первый элемент списка не зачеркнут | Применён класс “done-false”, CSS-стили проверять не надо. |
| 4. Поставить галочку у первого элемента | Элемент списка становится зачеркнутым (применяется класс “done-true”), отображаемое число оставшихся элементов уменьшается на 1. |
| 5. Повторить шаги 3, 4 для остальных элементов списка | |
| 6. Добавить новый элемент списка | Новый элемент списка не зачеркнут, отображаемое общее число и число оставшихся элементов увеличиваются на 1. |
| 7. Нажать на новый элемент списка | Элемент списка становится зачеркнутым, отображаемое число оставшихся элементов уменьшается на 1. |

Класс теста:

```
public class LambdaTest extends BaseTests {
```

```
@Test
@ExtendWith(CustomTestWatcher.class)
```

```
@DisplayName("https://lambdatest.github.io sample todo app basic tests")
public void basicTests() {

    HomePage homePage = pageManager.getHomePage()
        .open()
        .verifyRemainingTasksTextPresence();

    for (int i = 0; i < 5; i++) {
        homePage.verifyTodoState(i, false)
            .clickTodo(i);
    }

    homePage.addTodo()
        .clickTodo(5);
}
}
```

Класс «HomePage», используемый в тесте:

```
public class HomePage extends BasePage {

    @FindAll({
        @FindBy(tagName = "h1"),
        @FindBy(tagName = "h2"),
        @FindBy(tagName = "h3"),
        @FindBy(tagName = "h4"),
        @FindBy(tagName = "h5"),
        @FindBy(tagName = "h6")
    })
    private List<WebElement> headers;

    @FindBy(xpath = "//span[@class='ng-binding']")
    private WebElement remainingTodosSpan;

    @FindBy(xpath = "//ul/li")
    private List<WebElement> todos;

    @FindBy(id = "addbutton")
    private WebElement addButton;

    public HomePage open() {
        Allure.step("open link https://lambdatest.github.io/sample-todo-app/", step -> {
            driverManager.getDriver().get("https://lambdatest.github.io/sample-todo-app/");
            verifyHeaderPresence("LambdaTest Sample App");
        });
        return this;
    }
}
```

```

}

public HomePage verifyHeaderPresence(String text) {
    Allure.step(String.format("verify '%s' header presence", text), step -> {
        waitUntilElementsIsVisible(headers);
        WebElement actualHeader = headers
            .stream()
            .filter(header -> header.getText().equals(text))
            .findFirst()
            .orElse(null);
        Assertions.assertNotNull(actualHeader, String.format("header '%s' is not founded",
text));
        Assertions.assertTrue(actualHeader.isDisplayed(),
String.format("header '%s' is not displayed", text));
        Assertions.assertEquals(text, actualHeader.getText(),
String.format("header text is not equal to '%s'", text));
    });
    return this;
}

@Step("verify '5 of 5 remaining' text presence")
public HomePage verifyRemainingTasksTextPresence() {
    String expectedText = "5 of 5 remaining";
    waitUntilElementIsVisible(remainingTodosSpan);
    Assertions.assertTrue(remainingTodosSpan.isDisplayed(), "remaining tasks text is not
displayed");
    Assertions.assertEquals(expectedText, remainingTodosSpan.getText(),
String.format("remaining tasks text is not equal to '%s'", expectedText));
    return this;
}

public HomePage verifyTodoState(int index, boolean state) {

    Allure.step(String.format("verify that todo number %s is %s", index + 1, state ? "done" :
"not done"), step -> {
        WebElement todo = todos.get(index);
        WebElement todoSpan = todo.findElement(By.tagName("span"));
        String expectedClass = String.format("done-%s", state);

        Allure.step(String.format("verify that done-%s css class applied", state), substep -> {
            Assertions.assertEquals(expectedClass, todoSpan.getAttribute("class"),
String.format("css class 'done-%s' is not applied", state));
        });
    });
}

```

```

    return this;
}

private boolean getTodoState(WebElement todo) {
    WebElement todoInput = todo.findElement(By.tagName("input"));
    return todoInput.isSelected();
}

public HomePage clickTodo(int index) {
    int remainingTodosAmount = getRemainingTodosAmount();
    WebElement todo = todos.get(index);
    boolean prevTodoState = getTodoState(todo);
    WebElement todoInput = todo.findElement(By.tagName("input"));
    todoInput.click();

    Allure.step(String.format("mark the todo number %s as %s", index + 1, prevTodoState ?
    "not done" : "done"),
        step -> {
            Allure.step(String.format("verify that remaining tasks amount %s by 1",
                prevTodoState ? "increased" : "decreased"), substep -> {
                Assertions.assertEquals(remainingTodosAmount + (prevTodoState ? 1 : -1),
                    getRemainingTodosAmount(),
                    "remaining tasks amount is not " + (prevTodoState ? "increased" :
                "decreased")
                    + " by 1");
            });
            verifyTodoState(index, !prevTodoState);
        });

    return this;
}

private int getRemainingTodosAmount() {
    String remainingTodosText = remainingTodosSpan.getText();
    String[] parts = remainingTodosText.split(" ");
    return Integer.parseInt(parts[0]);
}

private int getTotalTodosAmount() {
    String remainingTodosText = remainingTodosSpan.getText();
    String[] parts = remainingTodosText.split(" ");
    return Integer.parseInt(parts[2]);
}

@Step("add new todo")

```

```

public HomePage addTodo() {
    int totalTodosAmount = getTotalTodosAmount();
    int remainingTodosAmount = getRemainingTodosAmount();

    addButton.click();

    verifyTodoState(todos.size() - 1, false);

    Allure.step("verify that total and remaining todos amount is increased by 1", step -> {
        Assertions.assertEquals(totalTodosAmount + 1, getTotalTodosAmount(),
            "total todos amount is not increased by 1");
        Assertions.assertEquals(remainingTodosAmount + 1, getRemainingTodosAmount(),
            "remaining tasks amount is not increased by 1");
    });

    return this;
}
}

```

Тестируемый сайт изображен на рисунке 3:



Рис. 3. Страница LambdaTest Todo App

Тестирование страницы расписания на сайте Мосполитеха

Для данной задачи использован следующий тест-кейс описанный в таблице 2:

Таблица 2. Мосполитех тест-кейс

| Шаг | Ожидаемый результат |
|--|---------------------|
| Перейти по ссылке: https://mospolytech.ru/ | |

| | |
|--|--|
| 2. Нажать на кнопку Расписания | |
| 3. В разделе “Расписания занятий” нажать “Смотрите на сайте” | Открывается страница поиска расписания в новой вкладке |
| 4. Ввести номер группы в поле поиска | В результатах поиска отображается только искомая группа |
| 5. Нажать на найденную группу в результатах поиска | Открывается страница расписания выбранной группы, текущий день недели выделен цветом (необходимо написать метод для определения текущего дня недели) |

Класс «MospolytechScheduleTest» с реализацией тест-кейса:

```
public class MospolytechScheduleTest extends BaseTests {

    @Test
    @ExtendWith(CustomTestWatcher.class)
    @DisplayName("https://mospolytech.ru schedule test")
    public void basicTests() {
        pageManager.getHomePage().open()
            .openSideMenuSection(HomePage.LeftNavigationMenuSection.SCHEDULE)
            .clickSeeOnTheSiteButton()
            .fillGroupSearchField("221-361")
            .clickOnTheFoundedGroup();
    }
}
```

1 гаш — Перейти по ссылке: <https://mospolytech.ru/>:

```
public HomePage open() {
    Allure.step("open link https://mospolytech.ru", step -> {
        driverManager.getDriver().get("https://mospolytech.ru");
    });
    return this;
}
```

{

Вид сайта главной страницы представлен на рисунке 4:

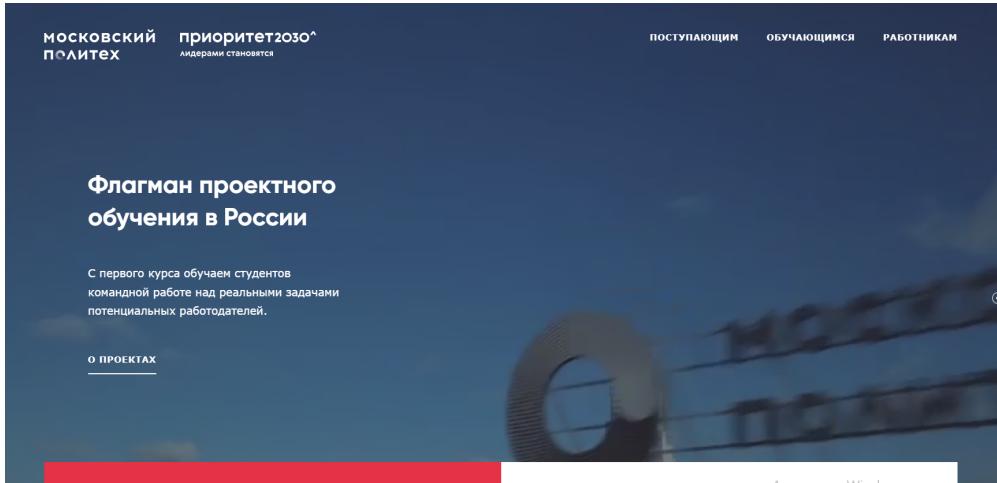


Рис. 4. Страница сайта Политеха

2 шаг - Нажать на кнопку Расписания:

```
public MpuSchedulePage openSideMenuSection(LeftNavigationMenuItem section) {  
    Allure.step(String.format("open left navigation menu item '%s'", section.getName()), step -> {  
        waitUntilElementsIsVisible(leftNavigationMenu);  
        WebElement menuItem = getLeftNavigationMenuItem(section);  
        Assertions.assertNotNull(menuItem, String.format("menu item '%s' is not found", section.getName()));  
        waitUntilElementToBeClickable(menuItem);  
        menuItem.click();  
    });  
    return PageManager.getInstance().getMpuSchedulePage();  
}
```

Результат клика по кнопке изображен на рисунке 5.

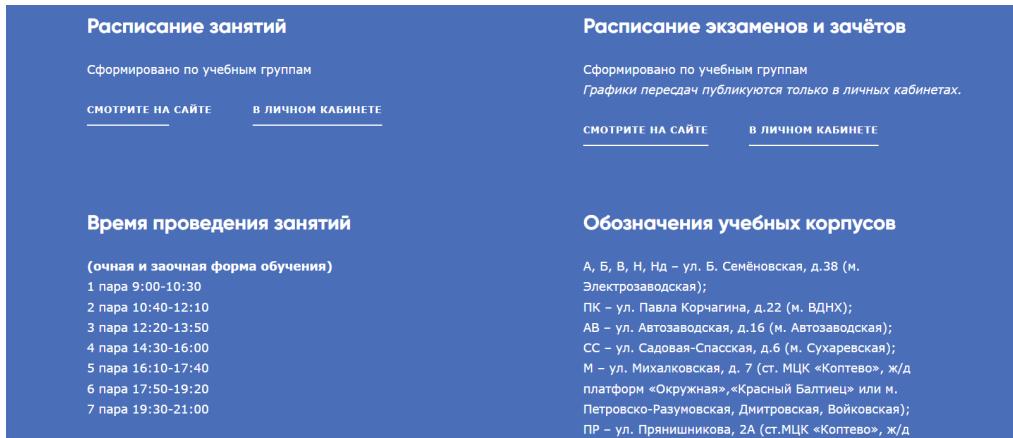


Рис. 5. Страница сайта Политеха (расписание)

Шаг 3: в разделе “Расписания занятий” нажать “Смотрите на сайте”:

```
public DmamiSchedulePage clickSeeOnTheSiteButton() {
    Allure.step("click 'see on the site' button", step -> {
        waitUntilElementIsVisible(seeOnTheSiteButton);
        scrollToElementJs(seeOnTheSiteButton);
        WebDriver driver = driverManager.getDriver();
        String currentHandle = driver.getWindowHandle();
        switchToNewTab(seeOnTheSiteButton);
        Allure.step("verify that page is opened in a new tab", subStep -> {
            Assertions.assertNotEquals(currentHandle, driver.getWindowHandle(), "new tab is not opened");
        });
    });
    return PageManager.getInstance().getDmamiSchedulePage();
}
```

Результат перехода на новую страницу можно увидеть на рисунке 6.

МОСКОВСКИЙ ПОЛИТЕХ

Для вывода расписания укажите в строке поиска номер учебной группы: **23A-321** цифровая кафедра / 2 квалификация: **23A-321**

Уважаемые студенты!
В расписании возможны изменения. Просим отслеживать расписание каждый день после 19:00.

Расписание занятий

Расписание зачетов по дисциплине "Проектная деятельность" доступно по ссылке

ВНИМАНИЕ! Занятия по дисциплинам "Общая физическая подготовка" и "Элективные курсы по физической культуре и спорту" реализуются по графику кафедры, размещенного по ссылке.

| Понедельник | Вторник | Среда | Четверг | Пятница | Суббота |
|---|---|---|--|--|---------|
| 12:20-13:50 ав2411 Методология научных исследований (Практика) Паршина Светлана Алексеевна, Меропушкин Вячеслав Георгиевич 01 Апр - 29 Апр | 12:20-13:50 ав2411 Методология научных исследований (Практика) Паршина Светлана Алексеевна, Меропушкин Вячеслав Георгиевич 01 Апр - 14 Апр | 10:40-12:10 ПК610 История и философия науки (Практика) Савченко Наталья Рафаиловна 06 Май - 02 Июн | 16:10-17:40 ПК535 Иностранный язык (Практика) Цапленко Любовь Петровна 01 Апр - 04 Май | 9:00-10:30 ав5205 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 15 Апр - 25 Апр | |
| 14:30-16:00 ав2411 Методология научных исследований (Практика) Паршина Светлана Алексеевна, Меропушкин Вячеслав Георгиевич 01 Апр - 19 Апр | 14:30-16:00 ав2411 Методология научных исследований (Практика) Паршина Светлана Алексеевна, Меропушкин Вячеслав Георгиевич 01 Апр - 14 Апр | 12:20-13:50 ПК610 История и философия науки (Практика) Савченко Наталья Рафаиловна 06 Май - 02 Июн | 17:50-19:20 ПК535 Иностранный язык (Практика) Цапленко Любовь Петровна 13 Май - 10 Июн | 10:40-12:10 ав5205 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 27 Апр - 05 Май | |
| 16:10-17:40 ав2411 Педагогика и психология высшей школы (Практика) Изразимова Галина Евгеньевна 01 Апр - 05 Май | 16:10-17:40 ав2411 Педагогика и психология высшей школы (Практика) Изразимова Галина Евгеньевна 10 Июн | 19:30-21:00 ПК535 Иностранный язык (Практика) Цапленко Любовь Петровна 01 Апр - 04 Май | 10:40-12:10 ав5205 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 15 Апр - 25 Апр | ав5206 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 27 Апр - 05 Май | |
| | | | ПК535 Иностранный язык (Практика) Цапленко Любовь Петровна 13 Май - 10 Июн | ав5206 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 15 Апр - 25 Апр | |
| | | | | ав5206 Педагогика и психология высшей школы (Практика) Плужникова Наталья Николаевна 27 Апр - 05 Май | |

Рис. 6. Страница сайта Политеха (расписание групп)

Шаг 4: ввести номер группы в поле поиска:

```
public DmamiSchedulePage fillGroupSearchField(String group) {
    waitUntilElementToBeClickable(groupSearchField);
    Allure.step(String.format("fill group search field with group '%s'", group), step -> {
        groupSearchField.click();
        groupSearchField.clear();
        groupSearchField.sendKeys(group);

        Allure.step(String.format("verify that the only founded group is '%s'", group), subStep -> {
            > {
                verifyFoundedGroup(group);
            });
        });
    return this;
}
```

Шаг 5: нажать на найденную группу в результатах поиска.

```
public DmamiSchedulePage clickOnTheFoundedGroup() {
    Allure.step("click on the founded group", step -> {
        waitUntilElementsIsVisible(foundedGroups);
        WebElement foundedGroup = foundedGroups.getFirst();
        waitUntilElementToBeClickable(foundedGroup);
        foundedGroup.click();

        Allure.step("verify that schedule is displayed", subStep -> {
```

```

waitUntilElementIsVisible(scheduleDiv);
Assertions.assertTrue(scheduleDiv.isDisplayed(), "schedule is not displayed");
});

Allure.step("verify that current day is colored", subStep -> {
    WebElement currentDayDiv = getCurrentDayDiv();

    Assertions.assertNotNull(currentDayDiv, "current day div is null");

    WebElement parentDiv = currentDayDiv.findElement(By.xpath(.."));

    String backgroundColor = parentDiv.getCssValue("background-color");

    Assertions.assertNotEquals(backgroundColor, "", "current day doesnt have
background color");
});

return this;
}

```

Тестирование Яндекс.Маркета

Для данной задачи использован следующий тест-кейс описанный в таблице 3:

Таблица 3. Яндекс.Маркет тест-кейс

| Шаг | Ожидаемый результат |
|---|---|
| 1. Перейти по ссылке: https://market.yandex.ru | Открывается главная страница Яндекс.Маркета |
| 2. В меню “Каталог” выбрать категорию: Ноутбуки и компьютеры -> Ноутбуки и планшеты -> Планшеты | Открывается страница “Планшеты” |
| 3. В меню фильтров выбрать производителя: Samsung | |
| 4. Установить сортировку: подешевле | |
| 5. Вывести в лог первые 5 найденных товаров (название и цену) | |
| 6. Запомнить вторую позицию из списка | |

| | |
|---|--|
| товаров (название и цену) | |
| 7. Ввести в строке поиска запомненный товар (наименование) и нажать «Поиск» | В результатах поиска 1 строкой указан искомый товар по наименованию и стоимости. |

Класс “YandexMarketTests” с реализацией тест-кейса:

```
public class YandexMarketTests extends BaseTests {
```

```
    @Test
    @ExtendWith(CustomTestWatcher.class)
    @DisplayName("https://market.yandex.ru basic test")
    public void basicTests() {

        pageManager.getHomePage()
            .open()
            .clickCatalogButton()
            .hoverLeftCatalogItem("Ноутбуки и компьютеры")
            .findCategory("Ноутбуки и планшеты")
            .clickSubcategory("Планшеты")
            .applyFilter("Производитель", "Samsung")
            .applySorting("подешевле")
            .logProducts(5)
            .saveProduct(2)
            .searchSavedProduct();
    }
}
```

Шаг 1: Перейти по ссылке: <https://market.yandex.ru>. Результат на рисунке 7.

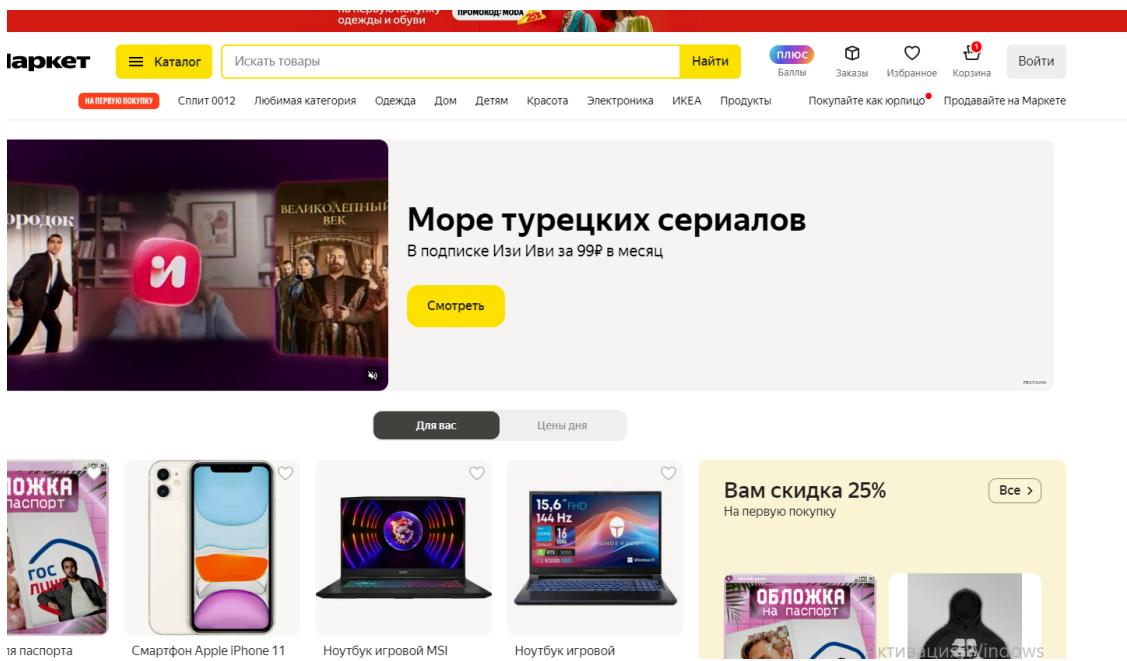


Рис. 7. Главная страница YandexMarket

Листинг-кода:

```
@Step("open link https://market.yandex.ru/")
public HomePage open() {
    driverManager.getDriver().get("https://market.yandex.ru");
    verifyYandexMarket MainPageOpened();
    return this;
}

@Step("verify Yandex Market main page opened")
private void verifyYandexMarket MainPageOpened() {
    String title = "Интернет-магазин Яндекс Маркет — покупки с быстрой доставкой";
    WebDriver driver = driverManager.getDriver();

    // to solve captcha
    wait = new WebDriverWait(driver, Duration.ofSeconds(60), Duration.ofSeconds(1));
    wait.until(ExpectedConditions.titleIs(title));

    Assertions.assertEquals(title, driver.getTitle(), "Yandex Market main page is not opened");
    resetDriverWait();
}
```

Шаг 2: В меню “Каталог” выбрать категорию: Ноутбуки и компьютеры -> Ноутбуки и планшеты -> Планшеты

```
public HomePage clickCatalogButton() {
```

```

waitUntilElementIsVisible(catalogButton);
waitUntilElementToBeClickable(catalogButton);
catalogButton.click();

return this;
}

public HomePage hoverLeftCatalogItem(String category) {
    waitUntilElementsIsVisible(leftCatalogLinks);
    WebElement categoryElement = null;
    for (WebElement element : leftCatalogLinks) {
        if (element.findElement(By.tagName("span")).getText().equals(category)) {
            categoryElement = element;
            break;
        }
    }
    Assertions.assertNotNull(categoryElement, String.format("category '%s' is not found", category));

    Actions actions = new Actions(driverManager.getDriver());
    actions.moveToElement(categoryElement).perform();

    return this;
}

public HomePage findCategory(String category) {
    waitUntilElementsIsVisible(navigationTreeCategories);
    for (WebElement element : navigationTreeCategories) {
        WebElement heading =
element.findElement(By.xpath("./div[@role='heading']"));
        String headingText = heading.findElement(By.tagName("a")).getText();
        if (headingText.equals(category)) {
            selectedCategory = element;
            break;
        }
    }
    return this;
}

@Step("open category {subcategory}")
public CategoryPage clickSubcategory(String subcategory) {

```

```

for (WebElement item : selectedCategory.findElements(By.xpath("./li//a")))
{
    if (item.getText().equals(subcategory))
    {
        waitUntilElementIsVisible(item);
        item.click();
        break;
    }
}

verifySubcategoryOpened(subcategory);
return PageManager.getInstance().getCategoryPage();
}

```

Результат представлен на рисунке 8.

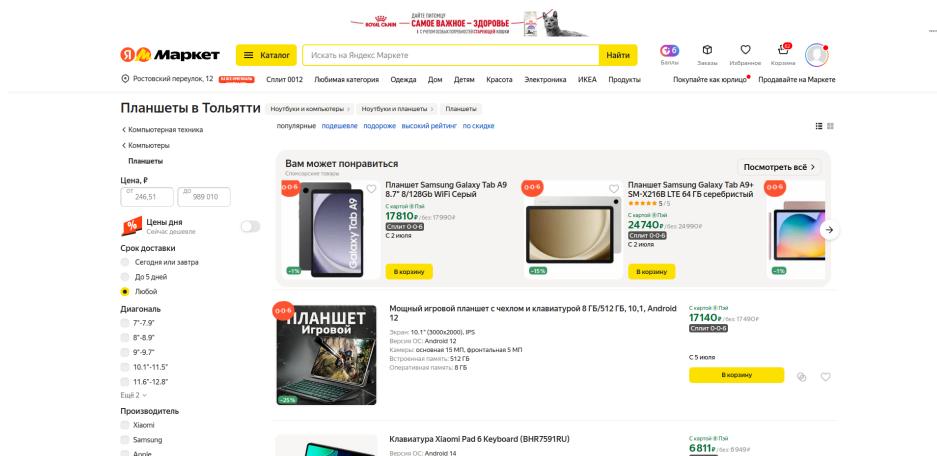


Рисунок 8. Страница планшетов YandexMarket

3. В меню фильтров выбрать производителя: Samsung

```

@Step("apply filter {key} to {value}")
public CategoryPage applyFilter(String key, String value) {
    waitUntilElementsIsVisible(filters);
}

```

```

for (WebElement filter : filters)
{
    String filterName;

    try {
        filterName = filter.findElement(By.xpath("./fieldset//legend//h4")).getText();
    } catch (NoSuchElementException e) {
        continue;
    }

    if (!filterName.equals(key)) {
        continue;
    }
}

```

```

}

List<WebElement> filterValues =
filter.findElements(By.xpath(".///fieldset//label"));

for (WebElement filterValue : filterValues) {
    String filterValueText =
filterValue.findElement(By.xpath(".///span//span/following-sibling::span")).getText();
    if (!filterValueText.equals(value)) {
        continue;
    }

    waitUntilElementToBeClickable(filterValue);
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    filterValue.click();
    waitForProductsUpdate();
    return this;
}
}

return this;
}

```

4. Установить сортировку: подешевле

```

@Step("apply sorting {type}")
public CategoryPage applySorting(String type) {
    waitUntilElementsIsVisible(sortingButtons);

    for (WebElement sortingButton : sortingButtons) {
        if (!sortingButton.getText().equals(type)) {
            continue;
        }

        waitUntilElementToBeClickable(sortingButton);
        sortingButton.click();
        waitForProductsUpdate();
        return this;
    }
}

```

```

    return this;
}

```

5. Вывести в лог первые 5 найденных товаров (название и цену)

```

@Step("log first {amount} products")
public CategoryPage logProducts(int amount) {
    waitUntilElementsIsVisible(products);
    for (int i = 0; i < amount; i++) {
        WebElement product = products.get(i);
        String productName = getProductName(product);
        String productPrice = getProductPrice(product);
        System.out.println(productName + " - " + productPrice + " руб.");
    }
    return this;
}

```

6. Запомнить вторую позицию из списка товаров (название и цену)

```

@Step("remember product number {number}")
public CategoryPage saveProduct(int number) {
    product = products.get(number - 1);
    return this;
}

```

7. Ввести в строке поиска запомненный товар (наименование) и нажать «Поиск»

```

@Step("enter in the search field remembered product")
public CategoryPage searchSavedProduct() {

    String prevProductName = getProductName(product);
    String prevProductPrice = getProductPrice(product);

    waitUntilElementIsVisible(search);
    waitUntilElementToBeClickable(search);
    search.click();
    search.clear();
    search.sendKeys(getProductName(product));

    waitUntilElementIsVisible(searchButton);
}

```

```

waitUntilElementToBeClickable(searchButton);

searchButton.click();

waitForElementsVisible(products);

WebElement foundedProduct = products.getFirst();

```

Allure.step("verify that in the search results first founded product is what was searched", step -> {
 Assertions.assertEquals(getProductName(foundedProduct),
 prevProductName);
 Assertions.assertEquals(getProductPrice(foundedProduct), prevProductPrice);
});

 return this;
}

Результат шага изображен на рисунке 9:

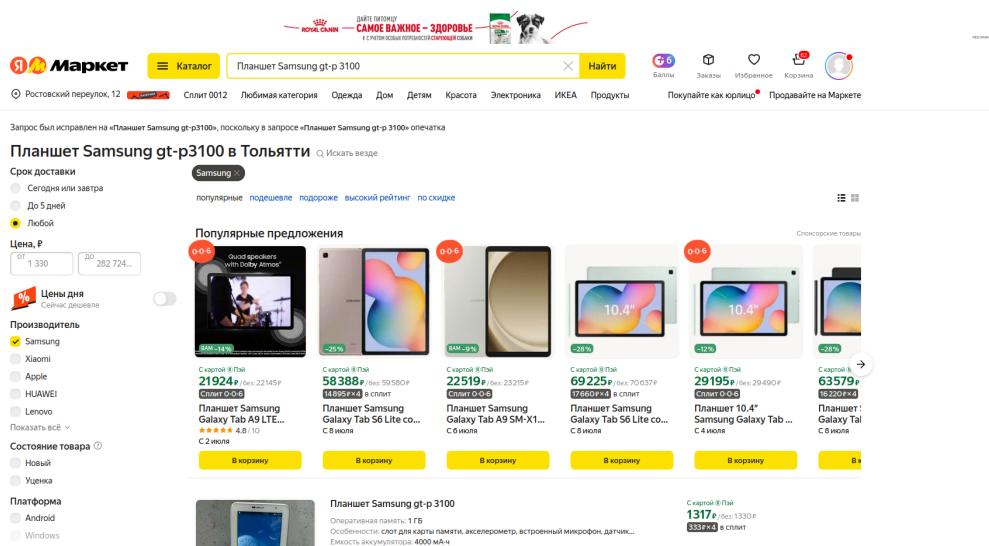


Рисунок 9. Поиск планшета

Тестирование GitHub

Для данной задачи использован следующий тест-кейс описанный в таблице 4.

Таблица 4. GitHub тест-кейс TU01

| Идентификатор | Описание | Шаг | Входные данные | Ожидаемый результат |
|---------------|--|---|----------------------|--|
| TU01 | Проверка поля поиска на главной странице | 1. Открыть github.com 2. Ввести название репозитория в поле поиска 3. Нажать на первый найденный репозиторий | 2. rust-lang/rust | 1. Открыта домашняя страница Github 2. Первый найденный результат совпадает с запросом 3. Открыта страница репозитория |

Код тест-кейса:

```

@Test
@DisplayName("https://github.com search test")
@ExtendWith(CustomTestWatcher.class)
public void searchTest() {

    final String input = "rust-lang/rust";

    pageManager.getHomePage()
        .open().search(input)
        .verifyFirstFoundedRepositoryName(input)
        .clickOnTheRepository(input)
        .verifyTitleContains(input);
}
  
```

1. Открыть github.com, результат представлен на рисунке 10.

```

@Step("open link https://github.com")
public HomePage open() {
    driverManager.getDriver().get("https://github.com");

    verifyOpened();

    return this;
}
  
```

{}

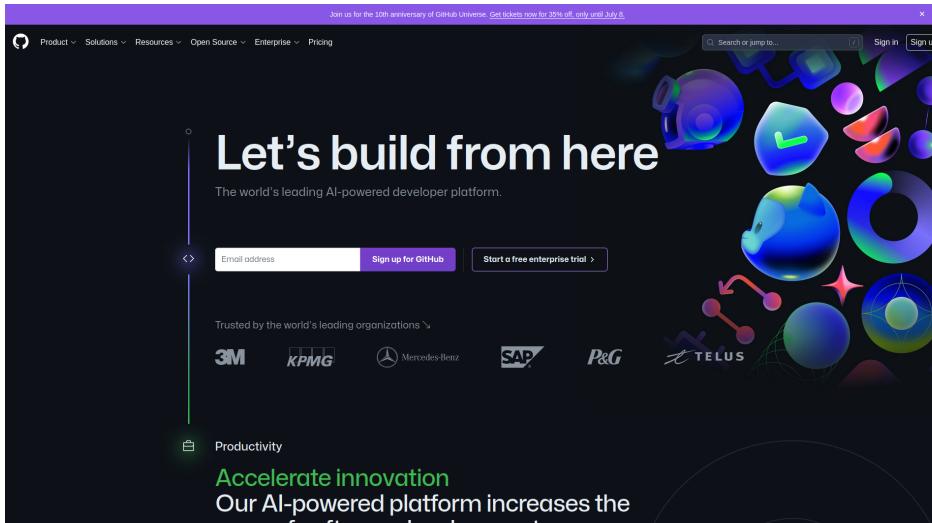


Рисунок 10. Главная страница Github

2. Ввести название репозитория в поле поиска “rust-lang/rust”

@Step("click on the search input")

```
public SearchPage search(String input) {
    waitUntilElementIsVisible(searchInput);
    waitUntilElementToBeClickable(searchInput);
    searchInput.click();

    waitUntilElementIsVisible(subSearchInput);

    subSearchInput.sendKeys(input);

    subSearchInput.sendKeys(Keys.RETURN);

    return PageManager.getInstance().getSearchPage();
}
```

3. Нажать на первый найденный репозиторий, результат представлен на рисунке 11.

@Step("click on the repository with name '{name}'")

```
public RepositoryPage clickOnTheRepository(String name) {
```

```
waitUntilElementsIsVisible(repositories);
```

```
WebElement foundedRepository = null;
```

```
for (WebElement repository : repositories) {
    String repositoryName = getRepositoryName(repository);

    if (repositoryName.equals(name)) {
        foundedRepository = repository;
        break;
    }
}
```

```
Assertions.assertNotNull(foundedRepository, String.format("repository '%s' not
founded", name));
```

```
WebElement repositoryLink =
foundedRepository.findElement(By.xpath(repositoryLinkXPath));

waitForElementToBeClickable(repositoryLink);

repositoryLink.click();

return PageManager.getInstance().getRepositoryPage();
}
```

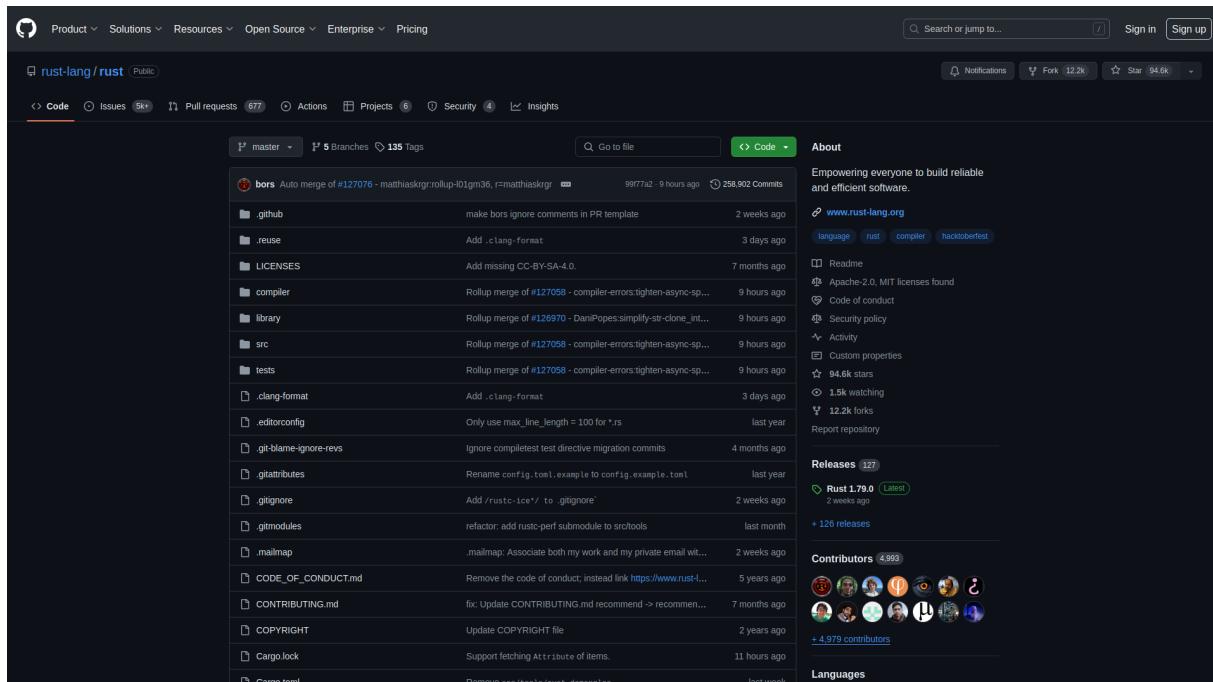


Рисунок 11. Страница репозитория на сайте Github

Кейс TU02 представлен в таблице 5.

Таблица 5. Github тест-кейс TU02

| Идентификатор | Описание | Шаг | Входные данные | Ожидаемый результат |
|---------------|---------------------------------|---|----------------|---|
| TU02 | Проверка раздела “Contributors” | 1. Открыть github.com/rust-lang/rust 2. Нажать на кнопку “Contributors” 3. Вывести в log 5 пользователей, внесших наибольший вклад в развитие проекта | | 1. Открыта страница репозитория rust-lang/rust 2. Открыт раздел “Contributors” |

Исходный код тест-кейса:

```

@Test
@DisplayName("https://github.com contributors test")
@ExtendWith(CustomTestWatcher.class)
public void contributorsTest() {
    pageManager.getRepositoryPage().open("rust-lang/rust")
        .clickContributorsButton().verifyContributorsPageOpened()
        .logContributors(5);
}
  
```

1. Открыть github.com/rust-lang/rust

```

@Step("open repository page https://github.com/{repositoryName}")
public RepositoryPage open(String repositoryName) {
    driverManager.getDriver().get("https://github.com/" + repositoryName);

    verifyTitleContains(repositoryName);

    return this;
}
  
```

2. Нажать на кнопку “Contributors”

```
public ContributorsPage clickContributorsButton() {
    waitUntilElementIsVisible(contributorsButton);
    scrollToElementJs(contributorsButton);
    waitUntilElementToBeClickable(contributorsButton);

    contributorsButton.click();

    return PageManager.getInstance().getContributorsPage();
}
```

Результат шага представлен на рисунке 12.

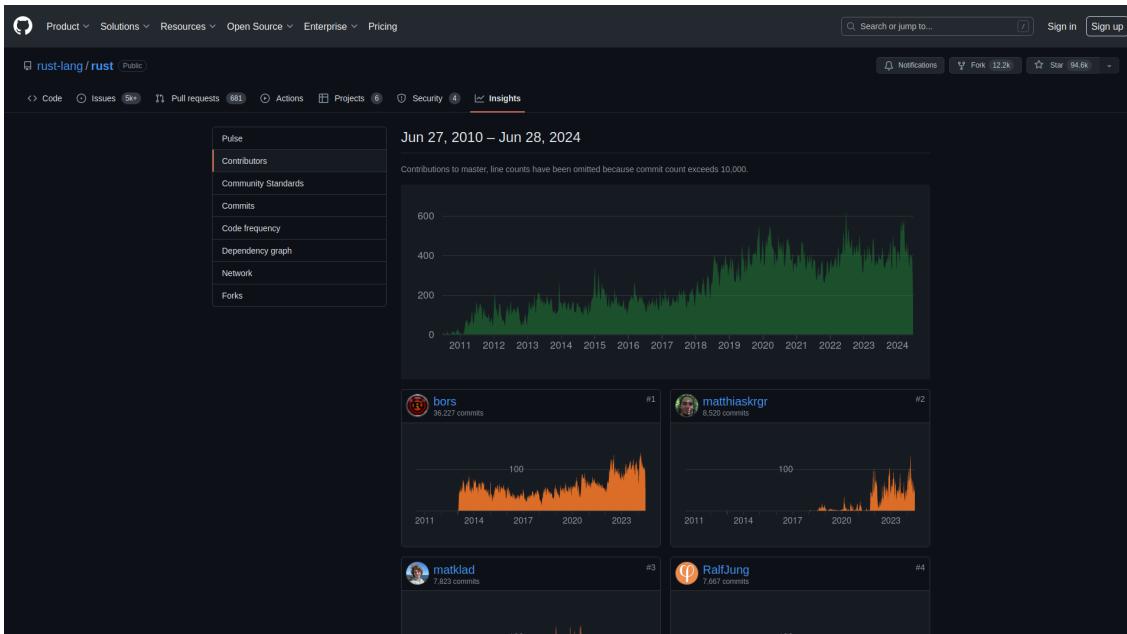


Рисунок 12. Страница “Contributors” репозитория “rust-lange/rust”

3. Вывести в log 5 пользователей внесших наибольший вклад в развитие проекта

```
public ContributorsPage logContributors(int amount) {
    waitUntilElementsIsVisible(contributors);

    System.out.println();
    for (int i = 0; i < amount; i++) {
        WebElement contributor = contributors.get(i);

        String contributorUsername =
contributor.findElement(By.xpath(contributorUsernameXPath)).getText();
```

```

String commits =
contributor.findElement(By.xpath(contributorCommitsXPath)).getText();

System.out.printf("%s. %s - %s%n", i + 1, contributorUsername, commits);

}

System.out.println();

return this;
}

```

Кейс TU3 описан в таблице 6:

Таблица 6. GitHub тест-кейс TU03

| Идентификатор | Описание | Шаги | Входные данные | Ожидаемые результаты |
|---------------|--|--|----------------|--|
| TU03 | Проверка фильтров трендовых репозиториев | 1. Открыть github.com 2. Навести курсор на пункт меню «Open Source» 3. Нажать на пункт подменю «Trending» 4. Нажать на фильтр “Language” 5. Ввести в поиск название языка программирования 6. Нажать “Enter” 7. Вывести в log 5 первых названий репозиториев | 5. Rust | 1. Открыта домашняя страница Github 2. Появилось подменю 3. Открыта страница “Trending” 4. Появилось подменю с выбором языка программирования 6. Применился фильтр по языку программирования |

Исходный код тест-кейса:

```
@Test  
    @DisplayName("https://github.com trending repositories filters test")  
    @ExtendWith(CustomTestWatcher.class)  
    public void trendingRepositoriesFiltersTest() {  
        pageManager.getHomePage().open().hoverHeaderMenuItem("Open  
Source").clickOpenSourceRepositoriesButton("Trending").verifyTrendingPageOpened().ap  
plyFilterByProgrammingLanguage("Rust")  
        .logRepositories(5);  
    }
```

1. Открыть github.com

```
@Step("open link https://github.com")  
public HomePage open() {  
    driverManager.getDriver().get("https://github.com");  
  
    verifyOpened();  
  
    return this;  
}
```

2. Навести курсор на пункт меню «Open Source»

```
@Step("hover header menu item '{name}'")  
public HomePage hoverHeaderMenuItem(String name) {  
  
    waitUntilElementsIsVisible(headerMenuItems);  
  
    for (WebElement menuItem : headerMenuItems) {
```

```

String menuItemName = menuItem.findElement(By.tagName("button")).getText();
if (menuItemName.equals(name)) {
    Actions actions = new Actions(driverManager.getDriver());
    actions.moveToElement(menuItem).perform();
    break;
}

}

return this;
}

```

3. Нажать на пункт подменю «Trending»

```

@Step("click button '{name}' in the Open Source submenu")
public TrendingPage clickOpenSourceRepositoriesButton(String name) {

    waitUntilElementsIsVisible(openSourceRepositoriesButtons);

    WebElement foundedButton = null;
    for (WebElement button : openSourceRepositoriesButtons) {
        String buttonName = button.findElement(By.tagName("a")).getText();
        if (buttonName.equals(name)) {
            foundedButton = button;
            break;
        }
    }

    Assertions.assertNotNull(foundedButton, String.format("can't find button with name '%s'", name));
}

```

```
waitForElementToBeClickable(foundedButton);
foundedButton.click();

return PageManager.getInstance().getTrendingPage();
}
```

4. Нажать на фильтр “Language”, 5. Ввести в поиск название языка программирования «Rust», 6. Нажать «Enter»:

```
@Step("apply filter by programming language to '{language}'")
public TrendingPage applyFilterByProgrammingLanguage(String language) {
```

```
waitForElementIsVisible(languageFilterButton);
waitForElementToBeClickable(languageFilterButton);
languageFilterButton.click();
```

```
WebElement languageFilterSubmenu =
languageFilterButton.findElement(By.tagName("details-menu"));
```

```
Allure.step("verify language filter submenu opened", () -> {
    waitForElementIsVisible(languageFilterSubmenu);
    Assertions.assertTrue(languageFilterSubmenu.isDisplayed());
});
```

```
WebElement languageFilterSearchInput =
languageFilterSubmenu.findElement(By.xpath("./filter-input/input"));
```

```
Allure.step(String.format("enter in the search field programming language name: %s",
language), () -> {
```

```
waitUntilElementIsVisible(languageFilterSearchInput);
waitUntilElementToBeClickable(languageFilterSearchInput);
languageFilterSearchInput.click();
languageFilterSearchInput.sendKeys(language);
});
```

```
Allure.step("press enter", () -> {
    wait.until(d -> {
        WebElement displayedLanguage =
languagesMenuItems.stream().filter(WebElement::isDisplayed).findFirst().orElse(null);

        if (displayedLanguage == null) {
            return false;
        }

        return
displayedLanguage.findElement(By.tagName("span")).getText().equals(language);
    });
}

languageFilterSearchInput.sendKeys(Keys.RETURN);

verifyFilterApplied(language);
});

return this;
})
```

7. Вывести в log 5 первых названий репозиториев

```
@Step("log first {amount} repositories")  
public TrendingPage logRepositories(int amount) {  
  
    waitUntilElementsIsVisible(repositories);  
  
    System.out.println();  
    for (int i = 0; i < amount; i++) {  
        WebElement repository = repositories.get(i);  
  
        String repositoryName = repository.findElement(By.xpath("./h2/a")).getText();  
  
        System.out.printf("%s. %s%n", i + 1, repositoryName);  
    }  
    System.out.println();  
  
    return this;  
}
```

Тестирование бэкенда reqres

Тест-кейс 1: Отправить GET-запрос на адрес <https://reqres.in/api/users?page=2>, проверить, что код ответа равен 200. В ответе проверить значение полей “page”, “page_count”, “total”, “total_pages”. Проверить, что внутри массива “data” значения полей “id” не равны null. Проверить, что в массиве есть элементы со следующими значениями полей: first_name=”Tobias”, last_name=”Funke”. Функция с кодом представлена на рисунке 13.

```

@Test
void listUsersTest() {

    ListUsersResponse listUsersResponse = checkStatusCodeGet("/users?page=2", 200)
        .assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath("json_schemas/ListUsers.json"))
        .extract().as(ListUsersResponse.class);

    assertThat(listUsersResponse).isNotNull()
        .extracting("page", "perPage", "total", "totalPages")
        .containsExactly(2, 6, 12, 2);

    List<User> users = listUsersResponse.getData();

    assertThat(users).allMatch(user -> user.getId() != null);

    assertThat(users).anyMatch(user -> user.getFirstName().equals("Tobias") && user.getLastName().equals("Funke"));
}

```

Рисунок 13. Тест-кейс 1 для бэкенда

Тест-кейс 2. Отправить GET-запрос на адрес <https://reqres.in/api/users/2>, проверить, что код ответа равен 200. Проверить, что внутри поля “data” значения полей равны следующим:

- "id": 2,
- "email": "janet.weaver@reqres.in",
- "first_name": "Janet",
- "last_name": "Weaver",
- "avatar": "https://reqres.in/img/faces/2-image.jpg"

Функция с кодом представлена на рисунке 14

```

@Test
void singleUserTest() {
    SingleUserResponse singleUserResponse = checkStatusCodeGet("/users/2", 200)
        .assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath("json_schemas/SingleUser.json"))
        .extract().as(SingleUserResponse.class);

    User actualUser = singleUserResponse.getData();

    User expectedUser = new User(2, "janet.weaver@reqres.in", "Janet", "Weaver", "https://reqres.in/img/faces/2-image.jpg");

    assertThat(actualUser).isEqualTo(expectedUser);
}

```

Рисунок 14. Тест-кейс 2 для бэкенда

Тест-кейс 3. Отправить GET-запрос на адрес <https://reqres.in/api/users/22>, проверить, что код ответа 404. Проверить, что тело ответа пустое (содержит только пустые фигурные скобки {}). Функция с кодом представлена на рисунке 15.

```

@Test
void singleResourceNotFoundTest() {
    checkStatusCodeGet("/unknown/23", 404)
        .assertThat().body(JsonSchemaValidator.matchesJsonSchemaInClasspath("json_schemas/Empty.json"));
}

```

Рисунок 15 Тест-кейс 3 для бэкенда

Остальные тест-кейсы составлены по аналогии.

Тестирование WEB (фронтенд) на главной странице. Листинг кода приведен на рисунке 16.

```

public class ReqResTests extends BaseTest {

    @Test
    @ExtendWith(CustomTestWatcher.class)
    @DisplayName("https://reqres.in basic test")
    public void basicTest() {
        HomePage homePage = pageManager.getHomePage().open();

        for (HomePage.RequestButton button : HomePage.RequestButton.values()) {
            homePage.clickRequestButton(button.getName(), button.getMethod());
        }
    }
}

```

Рисунок 16. Тест-кейс для тестирования фронтенда сайта reqres

Теория тестирования ПО

Тестирование программного обеспечения (ПО) — это процесс, направленный на оценку качества и обеспечение корректности работы приложения. Основные цели тестирования ПО включают:

- Обнаружение дефектов: идентификация и устранение ошибок, которые могут возникнуть в процессе разработки ПО;
- Повышение качества: обеспечение соответствия ПО требованиям и ожиданиям пользователей;
- Подтверждение соответствия требованиям. Проверка соответствия ПО техническим и функциональным спецификациям;
- Минимизация рисков, связанных с выпуском продукта на рынок;
- Улучшение пользовательского опыта. Обеспечение удобства и интуитивности использования ПО.

Виды тестирования ПО

Существует множество видов тестирования ПО, которые можно классифицировать по различным критериям. Основные виды тестирования включают:

- Функциональное тестирование. Проверка того, что ПО выполняет свои функции согласно спецификациям;
- Нефункциональное тестирование. Оценка характеристик ПО, таких как производительность, безопасность, удобство использования;
- Автоматизированное тестирование: Использование инструментов и скриптов для автоматизации процесса тестирования;
- Мануальное тестирование. Ручное выполнение тестов для проверки функциональности ПО;
- Регрессионное тестирование. Проверка того, что изменения в коде не нарушили существующую функциональность;
- Юнит-тестирование. Тестирование отдельных компонентов или модулей ПО;
- Интеграционное тестирование: Проверка взаимодействия между различными компонентами или модулями ПО;
- Системное тестирование. Тестирование всей системы в целом для проверки соответствия ПО требованиям;
- Приемочное тестирование. Проверка соответствия ПО требованиям пользователя перед его выпуском.

Обоснование целесообразности и преимуществ применения автоматизированного тестирования. Автоматизированное тестирование играет ключевую роль в современном процессе разработки ПО, обеспечивая следующие преимущества:

- Автоматизированные тесты выполняются быстрее, чем ручные, что позволяет сократить время тестирования;
- Автоматизированные тесты выполняются одинаково при каждом запуске, что исключает человеческий фактор;
- Возможность проведения большего числа тестов, что увеличивает покрытие кода и снижает вероятность дефектов;
- Автоматизированные тесты могут быть интегрированы в процесс непрерывной интеграции (CI), что позволяет обнаруживать дефекты на ранних этапах разработки;
- Автоматизация снижает затраты на тестирование в долгосрочной перспективе, так как требует меньше ручного труда;
- Автоматизированные тесты легко повторять, что делает их идеальными для регрессионного тестирования;
- Методы и средства проведения автоматизированного тестирования программного обеспечения.

Для автоматизированного тестирования используются различные методы и инструменты, которые могут быть классифицированы следующим образом:

Методы автоматизированного тестирования:

- Data-Driven Testing (DDT): использование различных наборов данных для выполнения одних и тех же тестов;

- Keyword-Driven Testing (KDT): использование ключевых слов, представляющих действия тестов, что позволяет легко создавать и поддерживать тесты;
- Behavior-Driven Development (BDD): тестирование на основе сценариев, описывающих поведение системы с точки зрения пользователя.

Инструменты автоматизированного тестирования:

- Selenium: инструмент для автоматизации веб-приложений, поддерживающий различные языки программирования;
- JUnit 4,5 Фреймворк для автоматизированного тестирования Java-приложений.
- RestAssured: Инструмент для тестирования REST API;
- Allure: Инструмент для создания отчетов о результатах тестирования;
- Применение автоматизированного тестирования позволяет значительно улучшить качество ПО, ускорить процесс разработки и сократить затраты на тестирование, что делает его неотъемлемой частью современного процесса разработки программного обеспечения.

Заключение

В ходе выполнения курсовой работы были достигнуты цели и задачи:

- Анализ и реализация тестов для веб-приложения;
- Были разработаны и проведены автоматизированные тесты.

Написаны тесты для "LambdaTest Sample App", YandexMarket, Kinopoisk MosPolytech, API-тест. Реализация тестов была выполнена с применением паттерна Page Object Model (POM), что обеспечило удобство сопровождения и расширяемость тестов. Данный подход помог отделить логику тестирования от логики взаимодействия с элементами веб-страницы, что способствовало улучшению читаемости и повторного использования кода. Применение инструментов автоматизации тестирования: в работе были использованы современные инструменты автоматизации тестирования, такие как Selenium для взаимодействия с веб-приложением и JUnit4, 5 для организации и выполнения тестов. Также был интегрирован Allure Framework для генерации отчетов о тестировании, что позволило наглядно представлять результаты тестов и выявленные дефекты.

Теоретическое обоснование автоматизированного тестирования: были изучены и описаны теоретические аспекты тестирования программного обеспечения, включая цели и виды тестирования.

Также по результатам тестирования был произведен Allure-отчет использовавшийся в проекте (рисунок 40). В приложение А вынесен код листинга ром файла с подключением библиотек.

Литература

1. **Page Object Model** // MTS Link [Электронный ресурс]. – Режим доступа:
<https://my.mts-link.ru/56082751/1150299225/record-new/918949682> (дата обращения: 17.06.2024).
2. **PageObject** // Google Drive [Электронный ресурс]. – Режим доступа:
<https://drive.google.com/file/d/1JMWussOTyULS9PxGxaUpX9Z8tItFV1CF/view?usp=sharing> (дата обращения: 17.06.2024).
3. **Selenium** // Habr [Электронный ресурс]. – Режим доступа:
<https://habr.com/ru/articles/152653/> (дата обращения: 17.06.2024).
4. **TestNG** // HowToDoInJava [Электронный ресурс]. – Режим доступа:
<https://howtodoinjava.com/testing/how-to-execute-testng-tests-with-maven-build/> (дата обращения: 17.06.2024).
5. **Allure-framework** // Habr [Электронный ресурс]. – Режим доступа:
<https://habr.com/ru/companies/sberbank/articles/358836/> (дата обращения: 17.06.2024).