МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БГТУ им. В.Г.ШУХОВА»

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

КУРСОВАЯ РАБОТА

по дисциплине:

«Тестирование программных систем»

тема: интеграционное тестирование Веб-приложения

Автор работы _____Прокопенко Николай Николаевич, ПВ-41

(подпись)

Руководитель проекта_____ Поляков Владимир Константинович

(подпись)

Оценка _____

Белгород
2020 г.

Оглавление

Введение

С развитием веб-технологий появилась необходимость в интеграционном тестировании веб-приложений. Одним из средств является Selenium WebDriver, инструмент для выполнения данной задачи.

Постановка задачи

Цель – интеграционное тестирование веб-приложения.

Теоретические сведения

Интеграционное тестирование веб-приложения включает в себя эмуляцию пользовательских историй, то есть симуляция поведения пользователя с помощью программных средств

Глава 1. Интеграционное тестирование

Интеграционное тестирование - одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию.

Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования.

Целью интеграционного тестирования является проверка соответствия проектируемых единиц функциональным, приёмным и требованиям надежности. Тестирование этих проектируемых единиц — объединения, множества или группы модулей — выполняется через их интерфейс, с использованием тестирования «чёрного ящика».

Глава 2. Selenium WebDriver

Selenium WebDriver - это инструмент для автоматизации действий веб-браузера. В большинстве случаев используется для тестирования веб-приложений, но этим не ограничивается. В частности, он может быть использован для решения рутинных задач администрирования сайта или регулярного получения данных из различных источников (сайтов).

Selenium WebDriver — это в первую очередь набор библиотек для различных языков программирования. Эти библиотеки используются для отправки HTTP запросов драйверу (отсюда и название WebDriver), с помощью протокола JsonWireProtocol, в которых указано действие, которое должен совершить браузер в рамках текущей сессии. Примерами таких команд могут быть команды нахождения элементов по локатору, переход по ссылкам, парсинг текста страницы/элемента, нажатие кнопок или переход по ссылкам на странице веб-сайта. Существуют как официальные привязки библиотеки к популярным языкам программирования, так и любительские. К примеру, библиотека для поддержки языка PHP не является официальной и разрабатывается Facebook.

Проектом Selenium и сообществом поддерживается работа с браузерами Microsoft Internet Explorer, Google Chrome, Mozilla Suite и Mozilla Firefox под управлением операционных систем Microsoft Windows, Linux и Apple Macintosh.

Глава 3. Процесс создания тестов

Для выполнения работы был выбран сайт Кинопоиск (https://www.kinopoisk.ru/). Он содержит необходимые списки сущностей, сами сущности и достаточное количество кнопок для проверки и изучения интеграционного тестирование веб-приложения.

Для тестов была выбрана библиотека tap (JavaScript), для интеграционного тестирования – Selenium WebDriver. Каждый тест открывает браузер, выполняет необходимые для данного теста действия, закрывает браузер и проверяет полученные данные или завершает тест в случае, если работа с браузером была выполнена успешно.

Заключение

В ходе выполнения курсовой работы были получены навыки в интеграционном тестировании, закреплены навыки тестирования с помощью библиотеки tap (JavaScript), получены навыки использования Selenium WebDriver.

Список использованной литературы

1. **Wikipedia. Selenium** [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Selenium;
2. **Wikipedia. Интеграционное тестирование** [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Интеграционное_тестирование;
3. **Selenium** [Электронный ресурс] – Режим доступа: https://www.selenium.dev/;
4. **Tap** [Электронный ресурс] – Режим доступа: https://node-tap.org/;

Приложение

tests.js

```
const t = require('tap');

const { Builder, By } = require('selenium-webdriver');
let fs = require('fs');

t.test('Selenium', (t) => {
  t.test('Open browser', async (t) => {
    let driver = await new Builder().forBrowser('chrome').build();
    await driver.get('https://www.kinopoisk.ru/');

    await driver.manage().window().maximize();
    const image64 = await driver.takeScreenshot();
    fs.writeFileSync('./images/openBrowser.png', image64, 'base64');

    await driver.quit();

    t.ok(driver, 'The browser should open');

    t.end();
  });

  t.test('Open browser', async (t) => {
    let driver = await new Builder().forBrowser('chrome').build();
    await driver.get('https://www.kinopoisk.ru/');

    await driver.manage().window().maximize();
    const buttons = await driver.findElements(
```

```
    By.className('_2Qgi-A90mMUOTbOm_3H84K')
  );
  await buttons[15].click();

  const image64 = await driver.takeScreenshot();
  fs.writeFileSync('./images/openPage.png', image64, 'base64');

  const url = await driver.getCurrentUrl();

  await driver.quit();

  t.same(
    url,
    'https://www.kinopoisk.ru/lists/top250/',
    'URL should be https://www.kinopoisk.ru/lists/top250/'
  );

  t.end();
});

t.test('Get first entity', async (t) => {
  let driver = await new Builder().forBrowser('chrome').build();
  await driver.get('https://www.kinopoisk.ru/lists/top250/');
  const entityText = await driver
    .findElement(By.className('selection-film-item-meta__name'))
    .getText();

  await driver.quit();

  t.notSame(entityText, null, "Name of entity shouldn't be null");

  t.end();
});

t.test('Pagination', async (t) => {
  let driver = await new Builder().forBrowser('chrome').build();
  await driver.get('https://www.kinopoisk.ru/lists/top250/');
  const firstPageEntity = await driver
    .findElement(By.className('selection-film-item-meta__name'))
```

```
    .getText();

  await (
    await driver.findElements(By.className('paginator__page-number'))
  )[1].click();

  const secondPageEntity = await driver
    .findElement(By.className('selection-film-item-meta__name'))
    .getText();

  await driver.quit();

  t.notSame(firstPageEntity, secondPageEntity, "Names shouldn't be equals");

  t.end();
});

t.test('Open entity', async (t) => {
  let driver = await new Builder().forBrowser('chrome').build();
  await driver.get('https://www.kinopoisk.ru/lists/top250/');

  const entityTextInList = await driver
    .findElement(By.className('selection-film-item-meta__name'))
    .getText();

  await (
    await driver.findElement(By.className('selection-film-item-meta__link'))
  ).click();

  await driver.manage().window().maximize();
  const image64 = await driver.takeScreenshot();
  fs.writeFileSync('./images/openEntity.png', image64, 'base64');

  const entityTextOutList = await driver
    .findElement(By.className('styles_title__2l0HH'))
    .getText();

  await driver.quit();
```

```
    t.same(entityTextInList, entityTextOutList, 'Names should be equals');

    t.end();
  });

  t.end();
});
```

Скриншоты

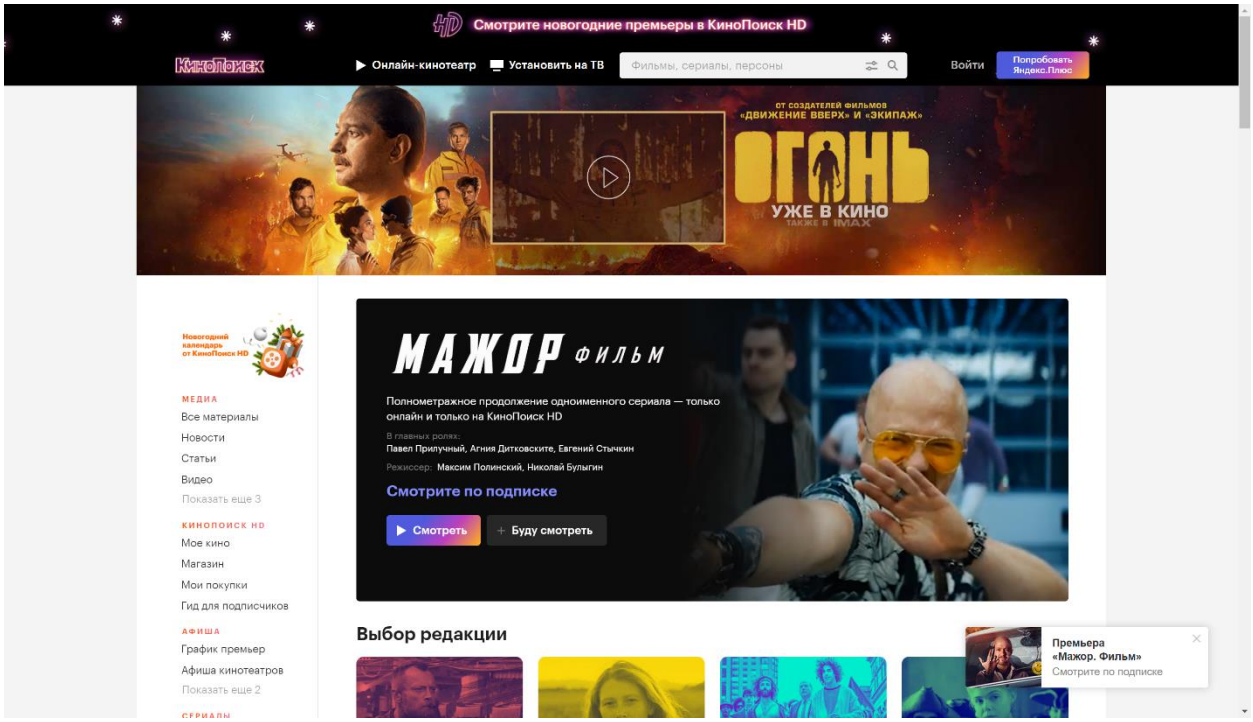

Рис.1. Открытый браузер

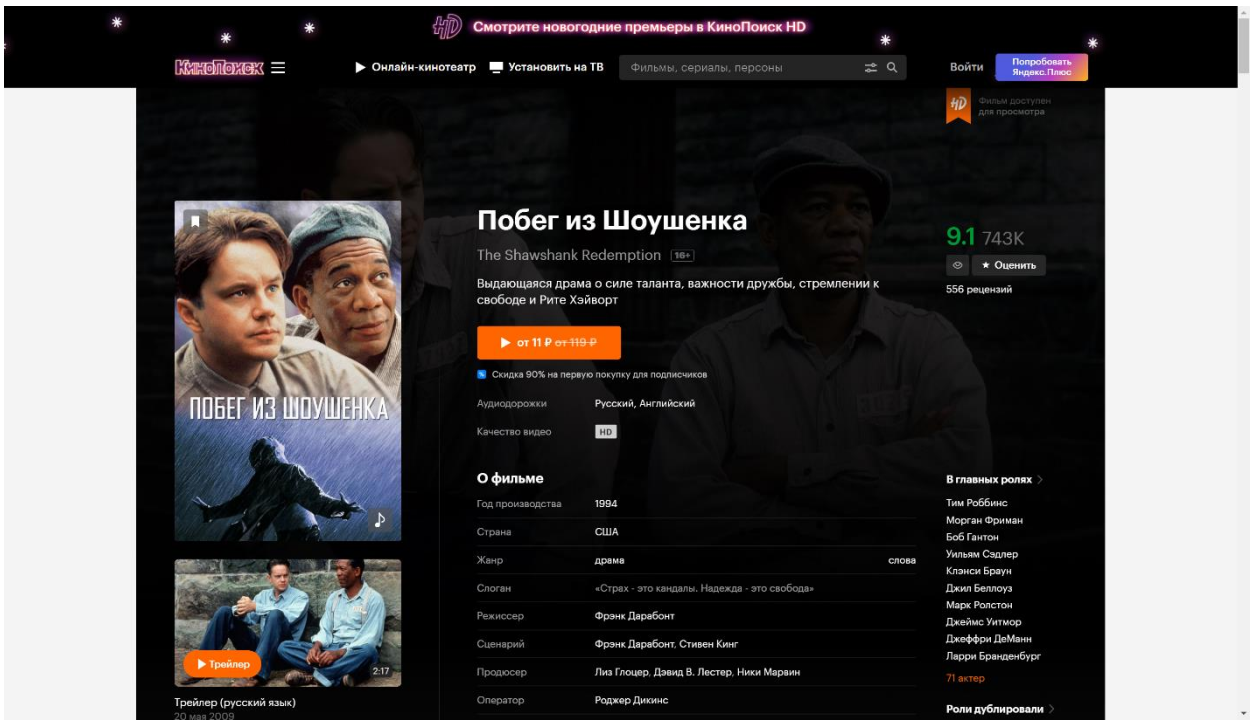

Рис. 2. Открытая сущность

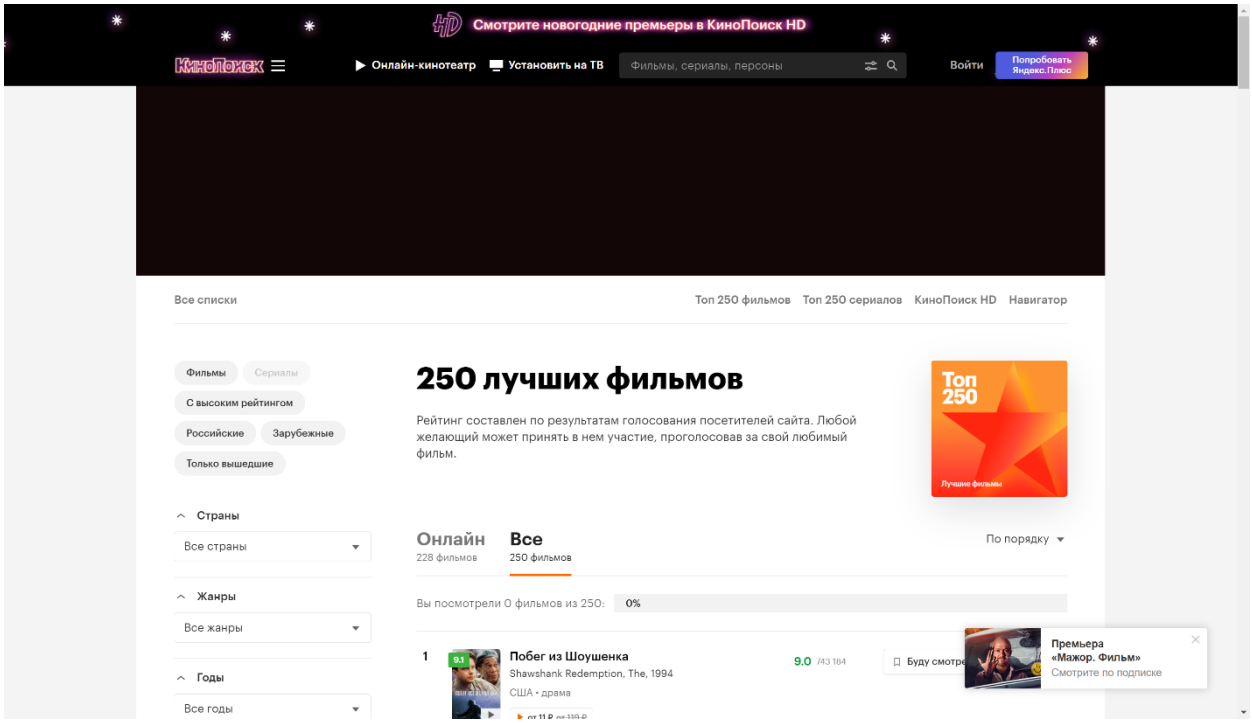

Рис. 3. Открытая страница (переход с главной)