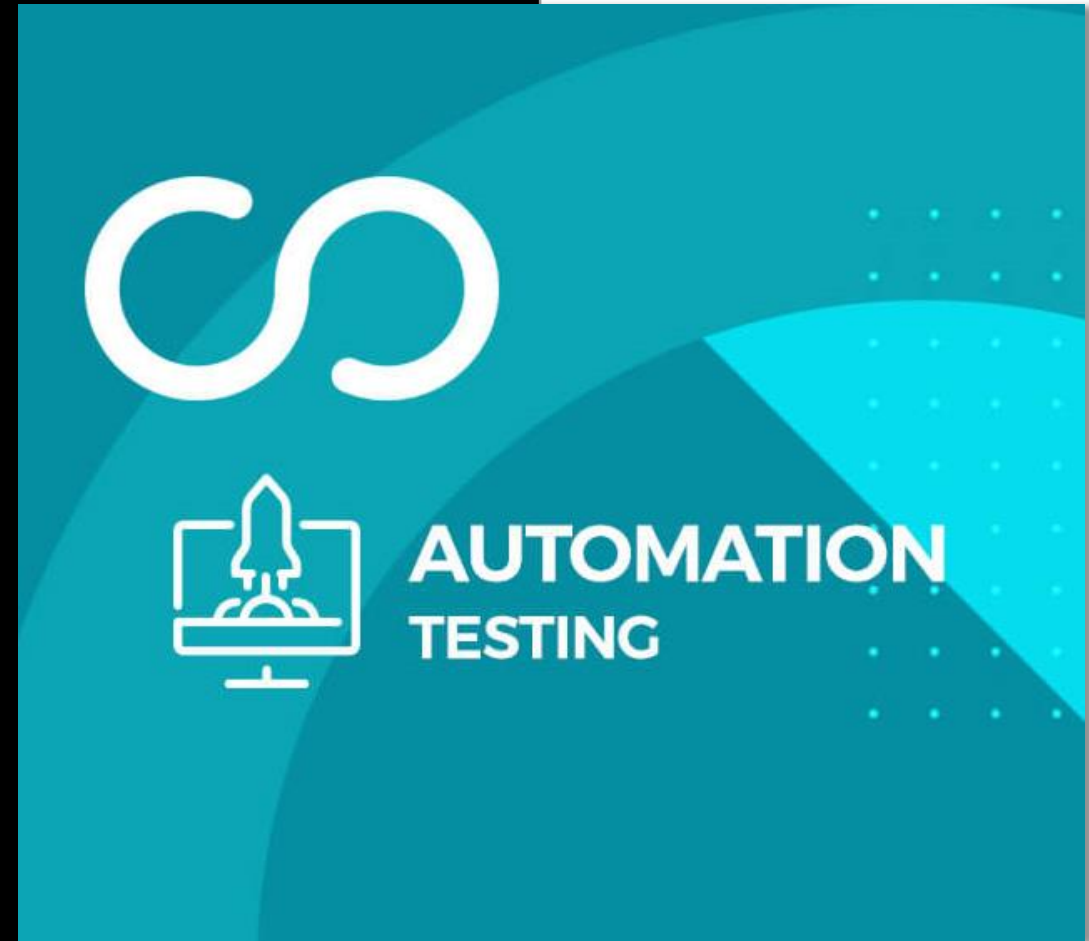


Библиотека «kbv-testdriver» на базе Selenium Webdriver для автоматизации тестирования UI и API веб-сайтов

Выполнил студент группы 883871
БУЛЫШКИН Кирилл Викторович

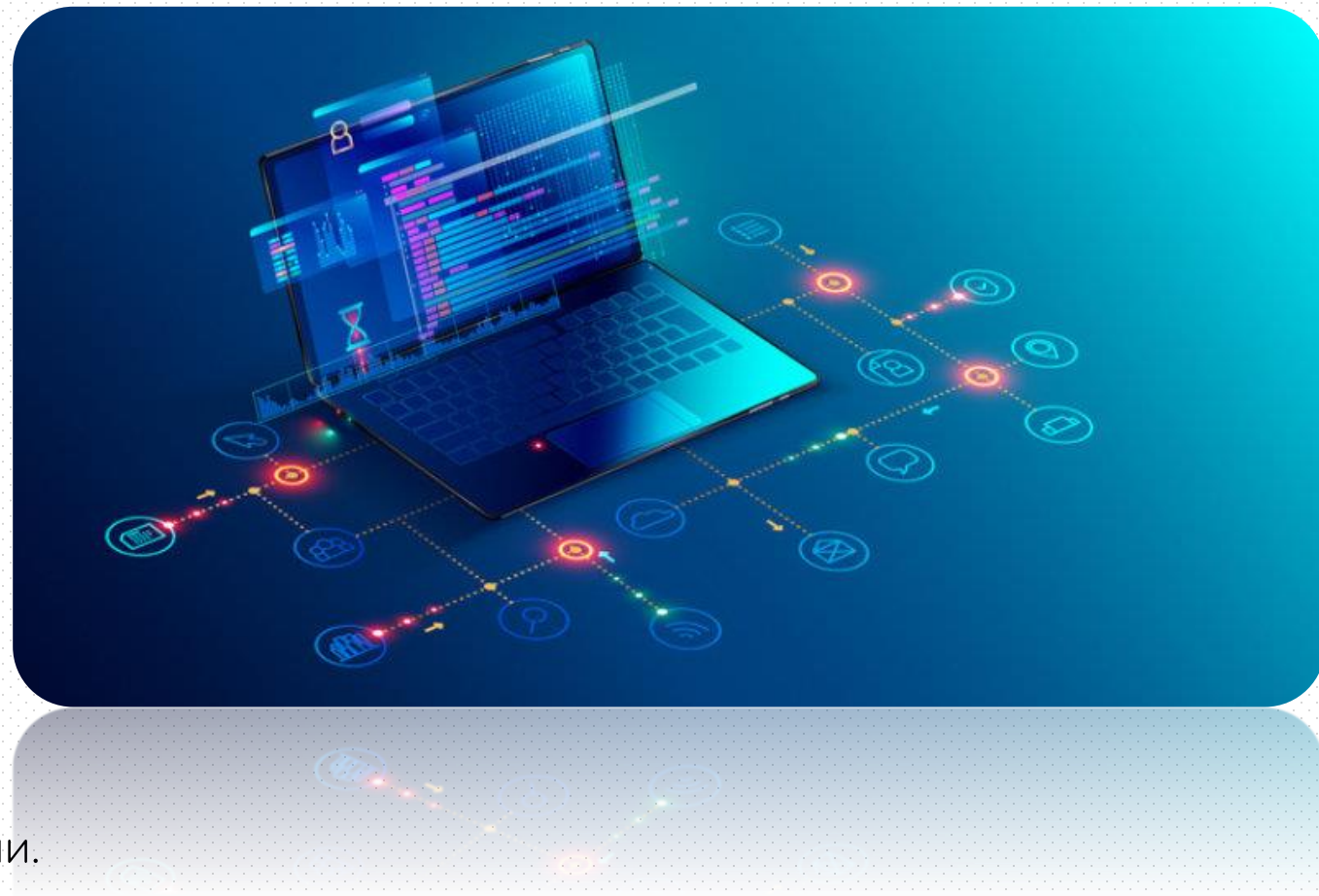
Научный руководитель - проректор по учебной работе,
канд.тех.наук, доцент кафедры ПИКС
ШНЕЙДЕРОВ Евгений Николаевич



Актуальность выбранной темы

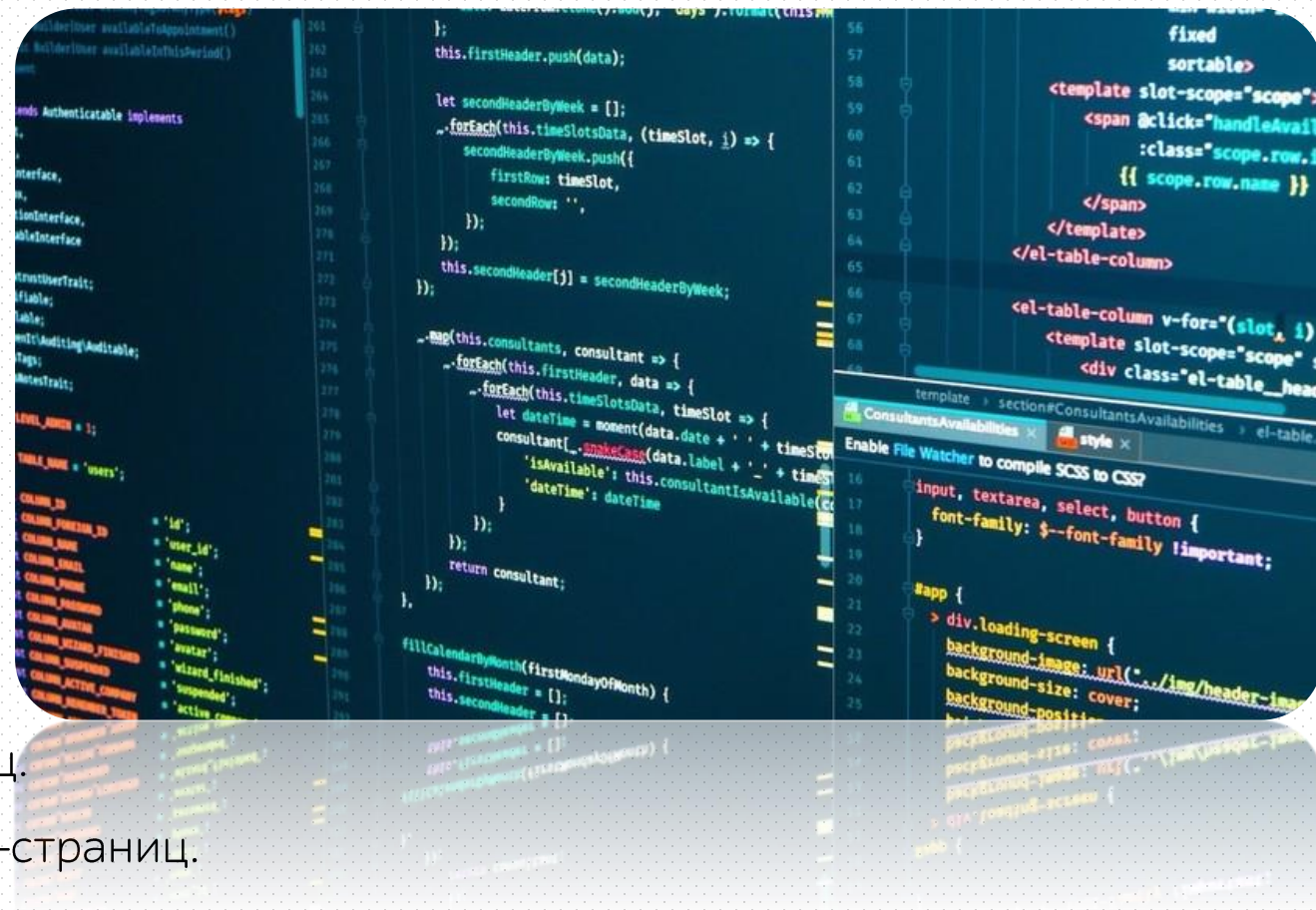
На текущий момент времени автоматизация тестирования является одним из наиболее бурно развивающихся направлений в IT-индустрии.

Качество разрабатываемых продуктов является крайне важным моментом, по причине высокой конкурентности рынка.



Функциональность разрабатываемой библиотеки

- Работа с графическими элементами веб-страниц.
- Проверка корректности функционирования веб-страниц.
- Возможность взаимодействия с программируемыми интерфейсами приложений.
- Возможность репрезентации итогов проведенных тестов помощью отчетов по результатам тестирования.
- Наличие подсистемы мониторинга работы библиотеки.



Технологии и инструменты, применяемые при разработке библиотеки



Visual Studio Code



JavaScript

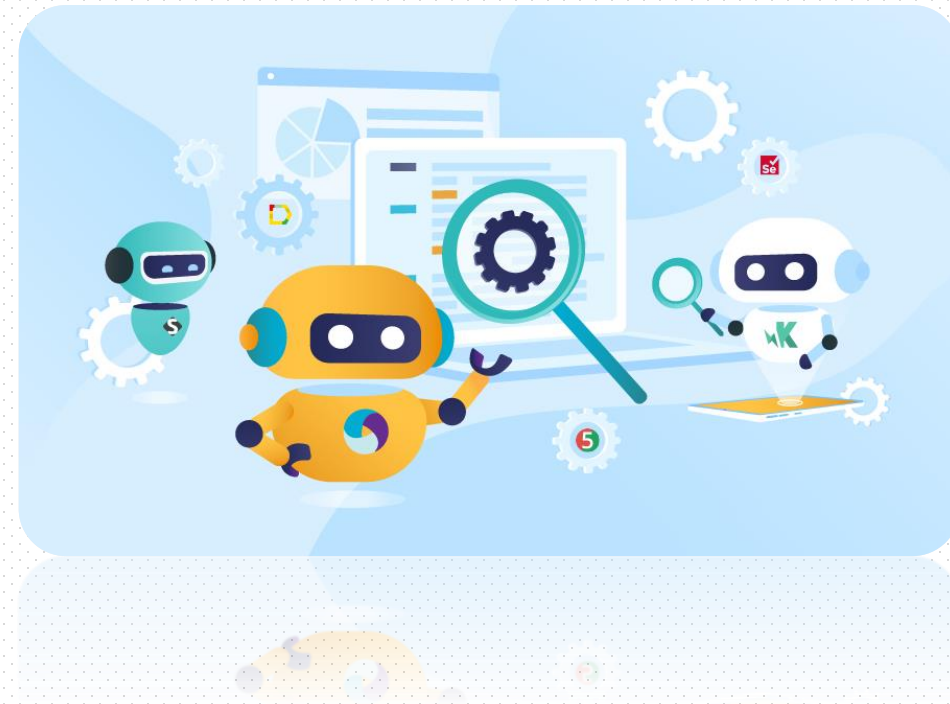


Selenium



Алгоритм взаимодействия пользователя с разрабатываемой библиотекой

- Предполагаемый пользователь (тестировщик), будет взаимодействовать с разрабатываемой библиотекой из консоли используемой среды разработки.
- Для запуска тестов достаточно будет ввести команду `npm test`, после чего будет произведено открытие браузера и выполнение тестовых сценариев.
- Результаты выполнения тестов будут выведены в консоль.
- Также пользователь имеет возможность просмотра отчета по результатам выполнения тестов.



Управление библиотекой

```
project > JS test.js > it('VK sign in and operations with post') callback
49
50 it('Slider', async () => {
51   await Browser.navigate(`${testData.host}${path.slider}`);
52   await Browser.windowMaximize();
53   const sliderPage = new SliderPage();
54   expect (await sliderPage.isDisplayed()).to.eql(testData.pageIsDisplaying);
55   const randomSliderPosition = RandomGenerators.getRandomValueFromArray(sliderValues);
56   await sliderPage.dragAndDropSlider(randomSliderPosition);
57   expect (await sliderPage.getSliderValue()).to.eql(sliderMap.get(randomSliderPosition));
58 });
59
60 hoversTestUserNumbers.forEach(function(userNumber) {
61
62   it(`Hovers (Test User Number ${userNumber})`, async () => {
63     await Browser.navigate(`${testData.host}${path.hovers}`);
64     await Browser.windowMaximize();
65     const hoversPage = new HoversPage();
66     expect (await hoversPage.isDisplayed()).to.eql(testData.pageIsDisplaying);
67     await hoversPage.moveCursorToHover(userNumber);
68     expect (await hoversPage.getHoverText(userNumber)).to.eql(testData.hoverText(userNumber));
69     expect (await hoversPage.hoverLinkIsDisplayed(userNumber)).to.eql(testData.elementIsDisplaying);
70     const hoverLinkHref = await hoversPage.getHoverLinkHref(userNumber);
71     await hoversPage.hoverLinkClick(userNumber);
72     expect (await Browser.getCurrentUrl()).to.eql(hoverLinkHref);
73     await Browser.backToPreviousPage();
74   });
75
76 });
```

Пример автоматизированных тестов,
созданных с помощью библиотеки
«kbv-testdriver»

kbv-testdriver 28.7s 0 7 6 1

22.1s 7 6 1

"before each" hook in "(root)"

```
await Browser.initBrowser(configs.browserName);
```

✓ Basic Authorization	1.9s	🔄
✓ Alerts	2.9s	🔄
✓ Slider	2.7s	🔄
✓ Hovers (Test User Number 1)	2.3s	🔄
✓ Hovers (Test User Number 3)	2.7s	🔄
✓ IFrame	3s	🔄
✗ VK sign in and operations with post	6.4s	🔄

NoSuchElementException: no such element: Unable to locate element: {"method":"xpath","selector":"//h1[@class='page_name']"}
(Session info: chrome=96.0.4664.110)

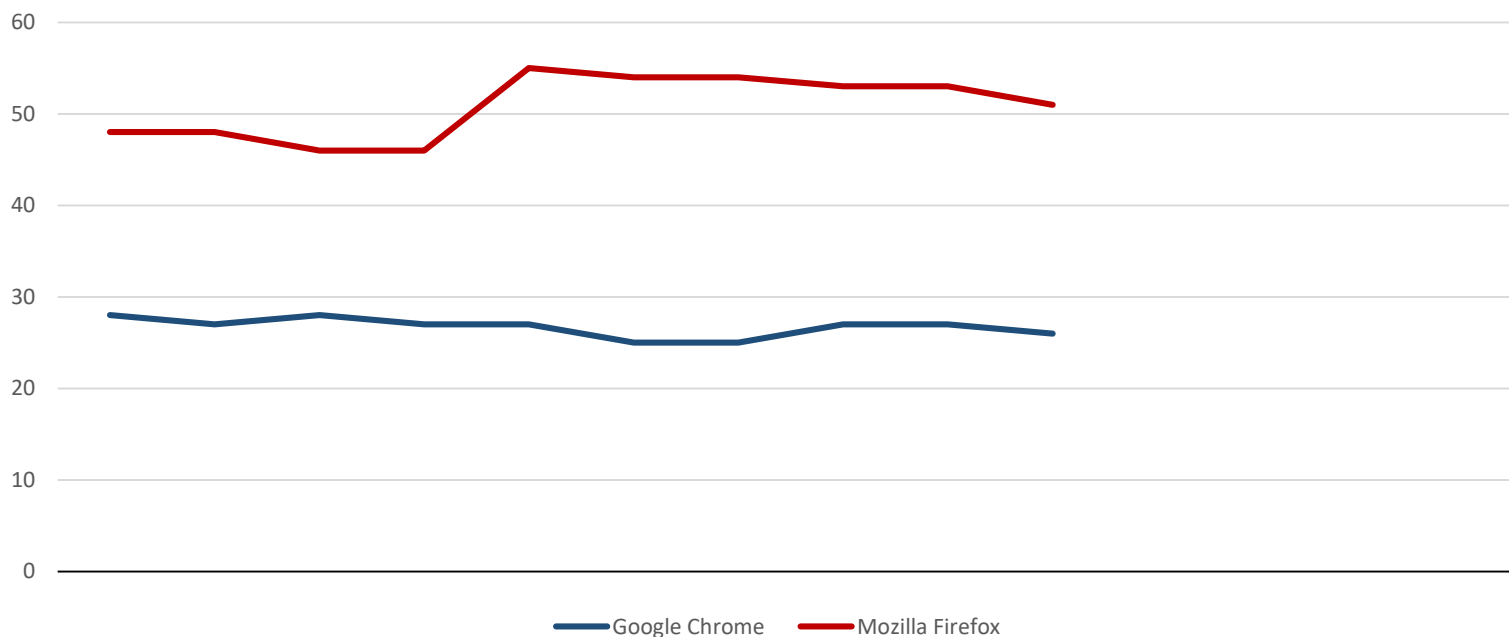
"after each" hook in "(root)"

Пример отчета по
результатам
выполнения тестов

Оценка временных показателей работы библиотеки

Итерация, номер	1	2	3	4	5	6	7	8	9	10
Google Chrome, с	28	27	28	27	27	25	25	27	27	26
Mozilla Firefox, с	48	48	46	46	55	54	54	53	53	51

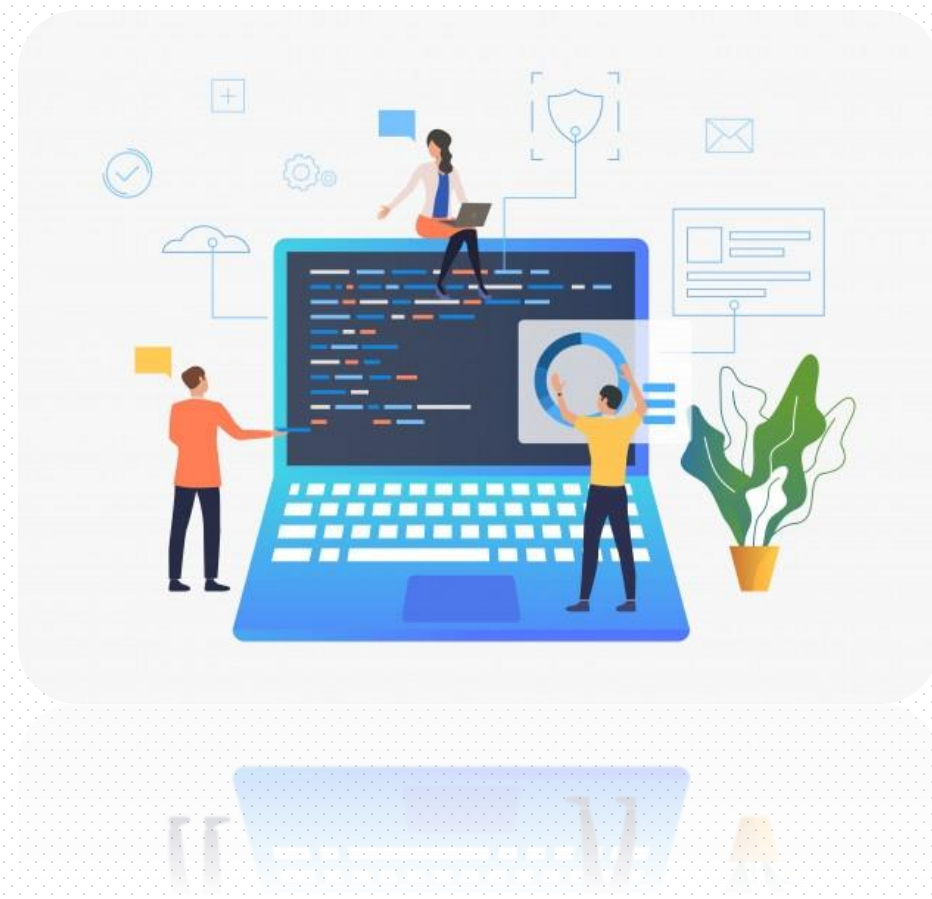
Временные показатели
тестовых прогонов
веб-сервиса
«[http://the-
internet.herokuapp.com](http://the-internet.herokuapp.com)»



График, отображающий
временные показатели
при выполнении тестовых
сценариев

Заключение

- В ходе выполнения дипломного проекта была спроектирована и разработана библиотека «kbv-testdriver» на базе Selenium Webdriver для автоматизации тестирования UI и API веб-сайтов.
- Разработанная библиотека полностью соответствует техническим и экономическим требованиям.
- Разработанная библиотека полностью готова к использованию и реализации автоматизации тестирования веб-сайтов.





Спасибо за внимание!

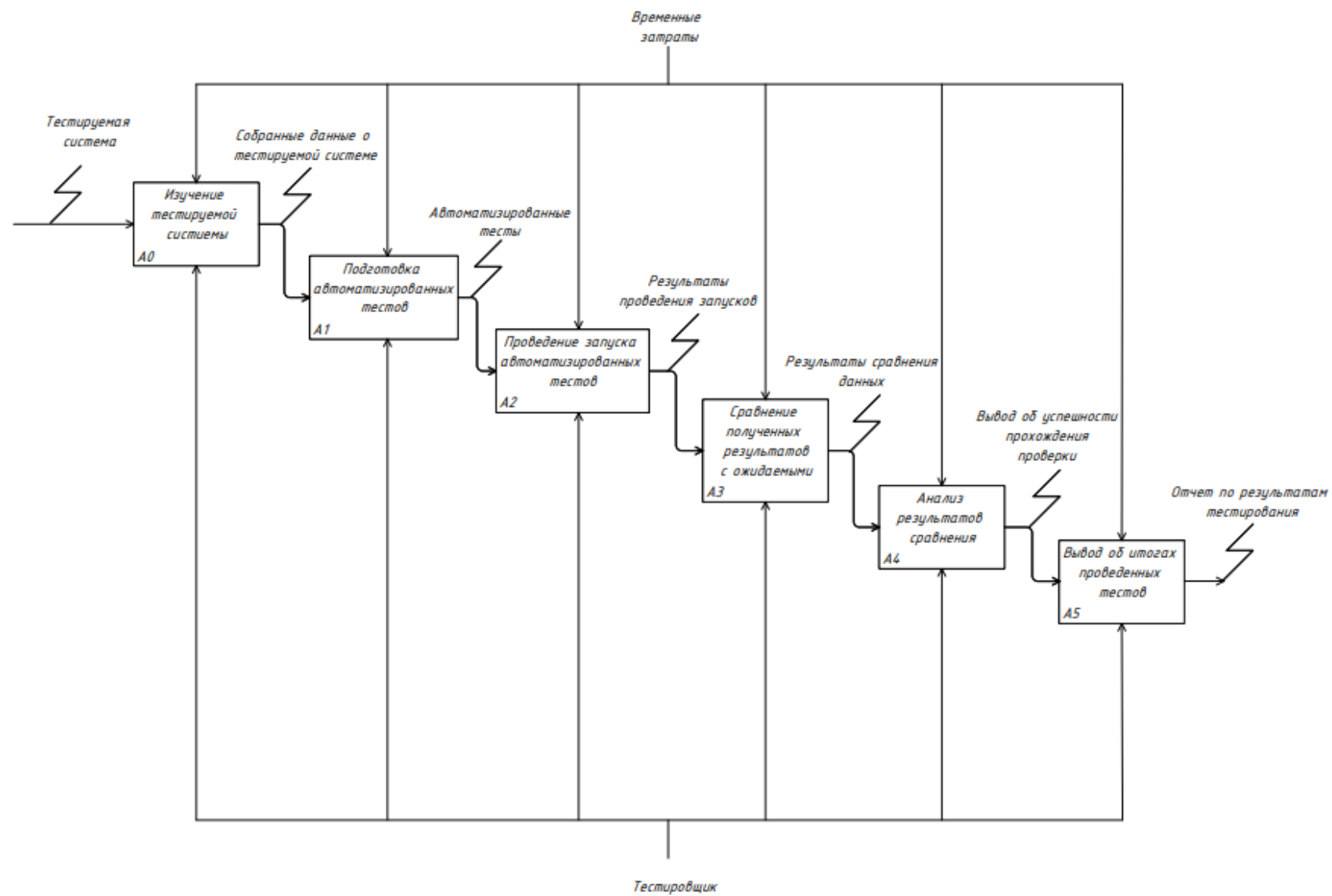
Видеозапись работы библиотеки

The screenshot displays the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure for 'KBV-TESTDRIVER' with folders 'configs', 'locators', 'pages', 'requests', and 'testData'. The 'test.js' file is selected in the 'testData' folder. The main editor shows the content of 'test.js', which includes various Jest assertions and Cypress commands for testing a web application. The terminal at the bottom shows the output of running the tests, including messages about Chrome browser initialization, window maximization, and successful authorization.

```
project > JS test.js > ...
104 await Logger.infoLog('Generated test JS file!');
105 const postId = await VkApiUtils.createPost(randomText);
106 await Logger.infoLog(`Post ID of the created post is ${postId}`);
107 const wallPage = new WallPage();
108 const author = await wallPage.getAuthorText();
109 await wallPage.waitingPostWithText(postId, randomText);
110 expect(await wallPage.getPostText(postId)).to.eql(randomText);
111 expect(await wallPage.getPostAuthor(postId)).to.eql(author);
112 const uploadUrl = await VkApiUtils.getWallUploadServer(postId);
113 const image = await fs.readFile(vkProjectTestData.filePath);
114 const form = new FormData();
115 form.append(vkProjectTestData.formDataKey, image, vkProjectTestData.formDataValue);
116 let {photo, server, hash} = await VkApiUtils.uploadPhotoToUrl(uploadUrl, form);
117 const photoId = await VkApiUtils.saveWallPhoto(photo, server, hash);
118 const randomTextEdited = RandomGenerators.randomStr(vkProjectTestData.randomStringLength);
119 await VkApiUtils.editPost(postId, randomTextEdited, photoId);
120 await wallPage.waitingPostWithText(postId, randomTextEdited);
121 const uploadedImageUrl = await VkApiUtils.getPhotoUrl(photoId);
122 const downloadUtils = new DownloadUtils();
123 await downloadUtils.downloadImageByUrl(uploadedImageUrl, vkProjectTestData.pathToUploadedImage);
```

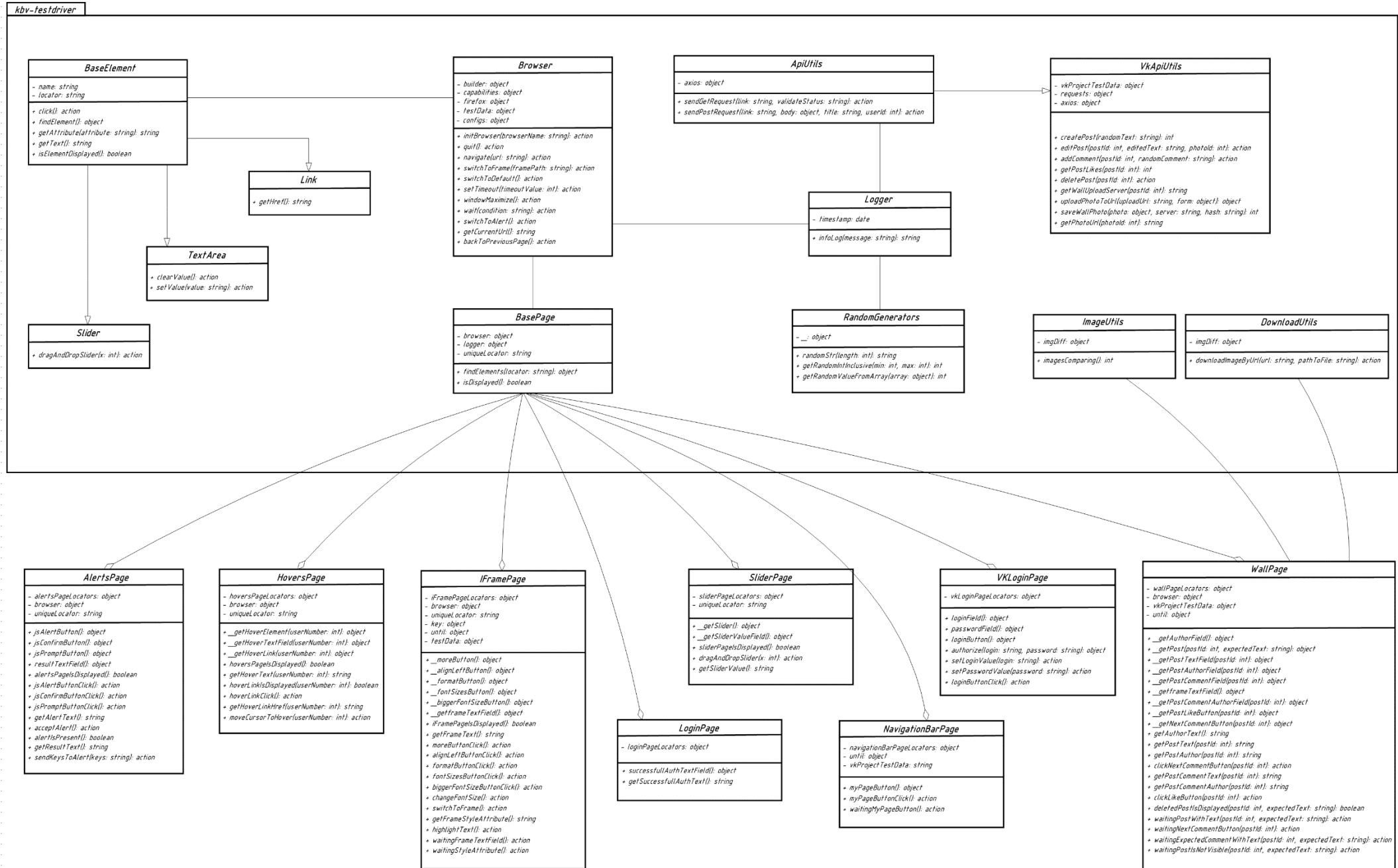
TERMINAL

```
INFO: Chrome browser initialization - (Mon Jan 10 2022 00:36:26 GMT+0300 (Москва, стандартное время))
DevTools listening on ws://127.0.0.1:59237/devtools/browser/12fd1e57-080f-445e-a699-1fe57f185def
INFO: Navigating to URL http://admin:admin@the-internet.herokuapp.com/basic_auth - (Mon Jan 10 2022 00:36:27 GMT+0300 (Москва, стандартное время))
INFO: Maximizing of window - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
INFO: Getting text of successfull authorization - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
INFO: Getting successfullAuthTextField - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
INFO: Getting text of element successfullAuthTextField - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
INFO: Finding element By(xpath, //div[@class='example']/p) - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
✓ Basic Authorization (2402ms)
INFO: Closing browser - (Mon Jan 10 2022 00:36:29 GMT+0300 (Москва, стандартное время))
INFO: Chrome browser initialization - (Mon Jan 10 2022 00:36:30 GMT+0300 (Москва, стандартное время))
Завершить выполнение пакетного файла [Y(да)/N(нет)]? y
PS C:\Users\kiril\Desktop\диплом\program\kbv-testdriver>
```

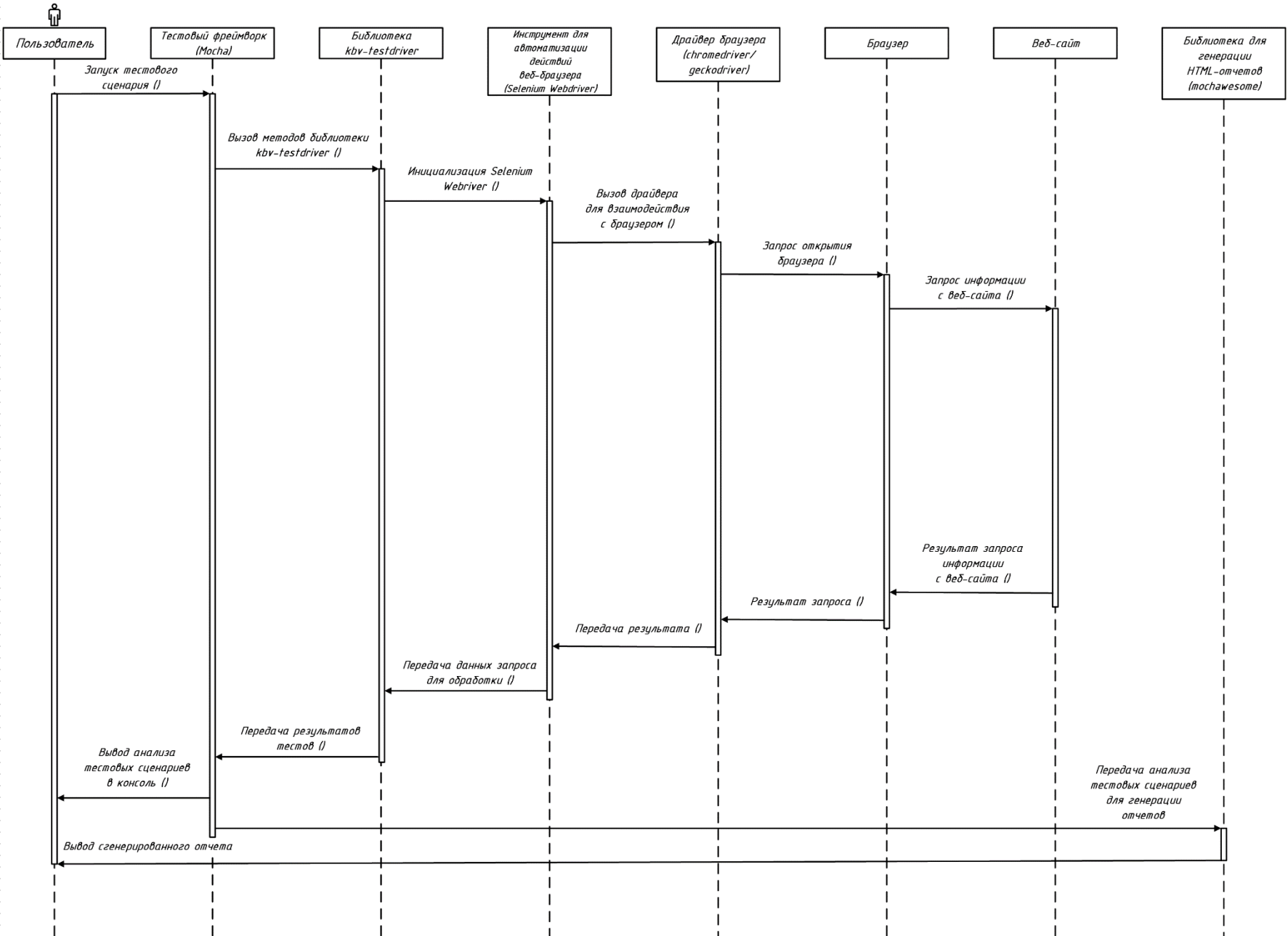


ГЦИР.88387-01 01 01									
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом
Имя	Фамилия	И. О.	Дата	Время	Место	Страна	Город	Улица	Дом

UML диаграмма классов



UML диаграмма последовательности (отработки тестового сценария)



Графический интерфейс отчета о результатах выполнения тестов

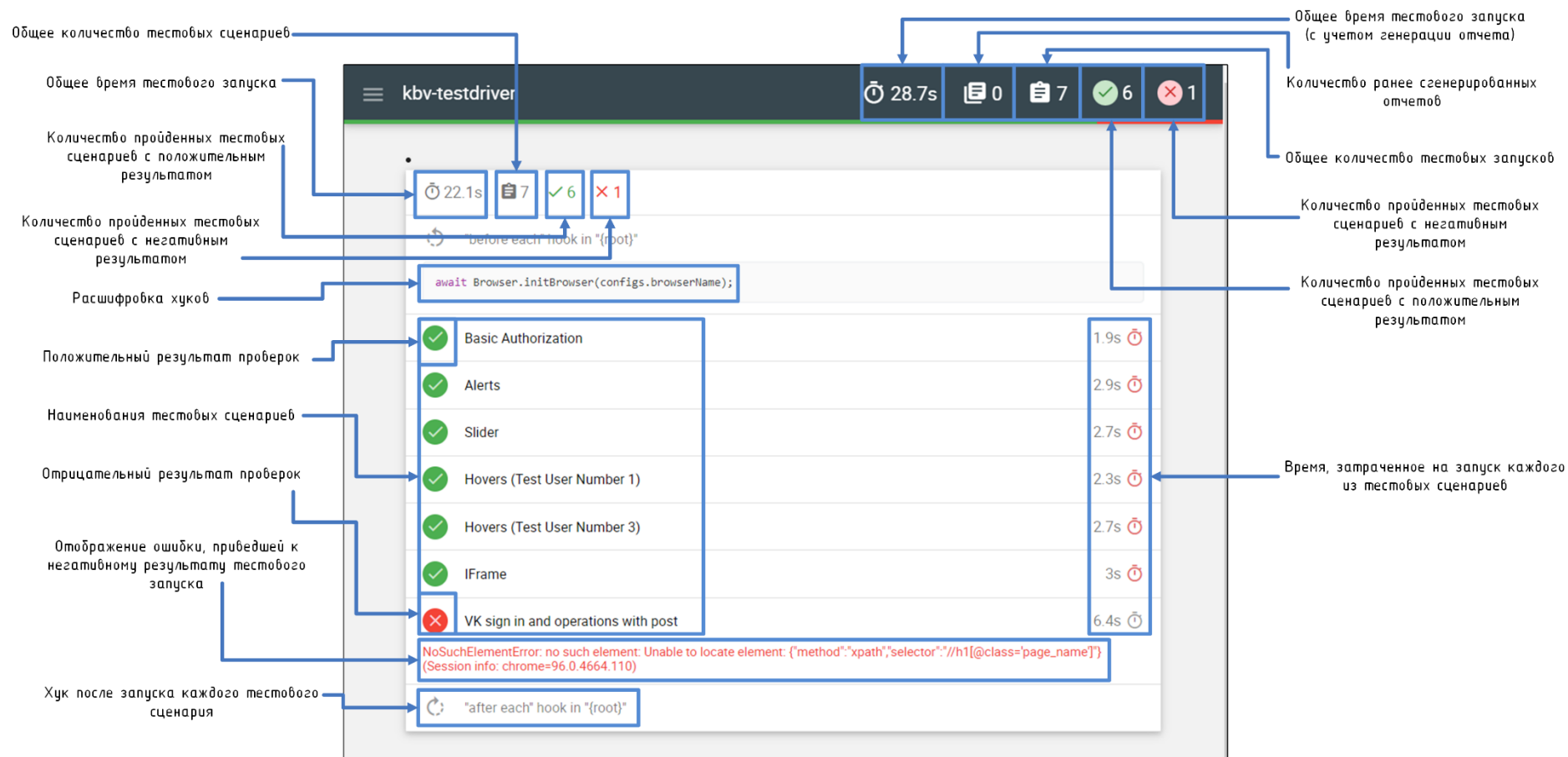


Рисунок 1 – Общий вид отчета

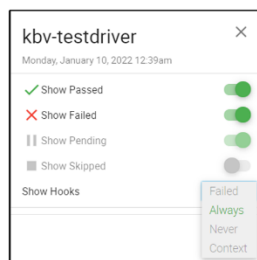


Рисунок 2 – Панель конфигурации отчета по результатам выполнения тестов



Рисунок 3 – Расшифровка тестового сценария с положительным результатом



Рисунок 4 – Расшифровка тестового сценария с негативным результатом

Результаты оценки количественных показателей программного средства

Временные показатели запуска автоматизированных тестов веб-сервиса http://the-internet.herokuapp.com										
Итерация, номер	1	2	3	4	5	6	7	8	9	10
Google Chrome, с	28	27	28	27	27	25	25	27	27	26
Mozilla Firefox, с	48	48	46	46	55	54	54	53	53	51

Выводы:

- все 20 запущенных подряд тестовых прогонов были успешно выполнены, что свидетельствует о высокой степени стабильности разработанных нами автоматизированных тестов и надежности при работе с библиотекой kbv-testdriver;
- разбегка во времени при запуске тестовых сценариев будет меньше при использовании браузера Google Chrome;
- тестовые запуски совершаются в среднем на 30 процентов быстрее в браузере Google Chrome.

Зарегистрированные показатели потребления оперативной памяти в ходе запуска автоматизированных тестов веб-сервиса http://the-internet.herokuapp.com	
Вид нагрузки на систему	Объем свободной оперативной памяти, МБ
Минимальный расход оперативной памяти	74
Пиковый расход оперативной памяти	8

Выводы:

- если произвести расчеты относительно изначально доступной оперативной памяти (рис.3), потребление оперативной памяти, создаваемое тестовыми запусками при использовании библиотеки kbv-testdriver составляет 286 МБ;
- оперативная память расходуется умеренно, никаких дополнительных мероприятий по оптимизации не требуется.

Меню	Избранное	Поле	Значение
АIDA64 v5.97.4600		Физическая память	
Компьютер		Всего	16212 МБ
Системная плата		Занято	11418 МБ
ЦП		Свободно	4794 МБ
CPUID		Загрузка	70 %
Системная плата		Виртуальная память	
Память		Всего	32596 МБ
SPD		Занято	17755 МБ
Чипсет		Свободно	14841 МБ
BIOS		Загрузка	54 %
ACPI		Файл подкачки	
Операционная система		Файл подкачки	E:\pagefile.sys
Сервер		Текущий размер	16384 МБ
Отображение		Текущая/пиковая загрузка	764 МБ / 1744 МБ
Мультимедиа		Загрузка	5 %
Хранение данных		Physical Address Extension (PAE)	
Сеть		Поддерживается ОС	Да
DirectX			
Устройства			
Устройства Windows			
Физические устройства			
Устройства PCI			

Рисунок 1 – Аппаратные характеристики оперативной памяти в AIDA64

Процессы						
Образ	ИД п...	Ошибк...	Заверш...	Рабочи...	Общий...	Частны...
Memory Compression	2656	0	3 728	1 053 3...	0	1 053 3...
chrome.exe	31696	0	719 520	663 792	96 440	567 352
chrome.exe	5048	0	540 904	544 076	74 684	469 392
chrome.exe	17412	0	377 156	380 600	72 004	308 596
Postman.exe	9576	0	665 296	333 376	126 048	207 328
MsMpEng.exe	4608	0	668 076	277 700	75 636	202 064
chrome.exe	18244	0	305 556	342 608	149 080	193 528
chrome.exe	27772	0	684 540	416 548	232 384	184 164
chrome.exe	16584	0	215 656	233 040	67 456	165 584

Физическая пам...	
Используется:	10928 МБ
Доступно:	5200 МБ

Зарезервировано	173
Используется	10928 мегабайт
Изменено	83 мегабайт
Ожидается	4741 мегабайт
Свободно	453 мегабайт

Доступно	5200 мегабайт
Кэшировано	4824 мегабайт
Всего	16211
Установлено	16384

Рисунок 2 – Исходные показатели потребления оперативной памяти в Мониторе ресурсов