Санкт-Петербургский государственный университет

Асланов Элтон Хосрович

Выпускная квалификационная работа

Разработка механизма автоматизации типичных действий пользователей в web-приложении для управления корпоративным контентом

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2018 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Системное программирование*

Научный руководитель:
доцент кафедры системного программирования, к.ф.-м.н., Д. В. Луцив

Рецензент:
разработчик ПО ООО «ДоксВижн» О.Д. Ноздря

Санкт-Петербург
2022

Saint Petersburg State University

Elton Aslanov

Bachelor's Thesis

Development of a mechanism for automating typical user actions in a web application for enterprise content management

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2018 "Software and Administration of Information Systems"*

Profile: *Software Engineering*

Scientific supervisor:
docent, C.Sc. D.V. Luciv

Reviewer:
software developer Docsvision LLC O.D. Nozdrya

Saint Petersburg
2022

Оглавление

1. Введение

«Цифровой трансформации» [7] — это внедрение современных технологий в бизнес-процесс, направленного на создание определенного продукта или услуги, и фундаментальных идей к управлению организацией. Цифровая трансформация призвана:

- увеличить эффективность производства,
- повысить производительность и удобство работы каждого сотрудника,
- максимально устранить неэффективный труд человека,
- минимизировать длительности всех бизнес-процессов,
- осуществить поступление информации вовремя и в оптимальном объеме,

В результате повышается уровень удовлетворённости сотрудников и клиентов, а также увеличиваются продажи организации.

Docsvision [12] является одной из программ, которую можно внедрить в компанию для цифровой трансформации. Docsvision — полнофункциональная BPM[1]/ECM[2]-платформа для автоматизации бизнес-процессов, обработки и хранения документов.

Данная работа направлена на сокращение длительности бизнес- процессов и предоставление сотрудникам возможности заниматься исключительно интеллектуальными задачами. Для этого можно автоматизировать рутинные действия пользователя в программе. Популярным решением является использование инструментов RPA[3], которые имитируют действия пользователя в системе.

[1]BPM (Business Process Management) [5] — системный подход к управлению бизнес-процессами организации

[2]ECM (Enterprise content management) [2] — система управления корпоративным контентом, в том числе хранение, обработка и передача контента.

[3]Robotic process automation [4] — форма технологии автоматизации бизнес-процессов, основанная на ботах.

2. Постановка задачи

Целью работы является реализация и внедрение механизма автоматизации типичных действий пользователей в веб-клиент Docsvision. Для её достижения были поставлены следующие задачи:

- обзор существующих инструментов;
- формулировка требований к механизму;
- выработка архитектуры механизма на основе обзора, требований и особенностей архитектуры Docsvision;
- реализация механизма автоматизации в веб-клиенте Docsvision;
- проведение апробации;
- тестирование механизма автоматизации.

3. Обзор

В обзоре представлены популярные инструменты с открытым исходным кодом для сквозного[4] тестирования, которые можно использовать за основу для автоматизации действий пользователей в веб-браузере.

3.1. Puppeteer

Puppeteer [6] — библиотека Node, предоставляющая API[5] для управления Chrome или Chromium по протоколу DevTools. Несколько примеров использования Puppeteer: автоматизирование отправки форм и ввода с клавиатуры, создание скриншотов и pdf-файлов страниц, получение данных с веб-страницы и т.д.

3.2. Playwright

Playwright [13] — платформа для веб-тестирования и автоматизации, которая содержит большую часть функций Puppeteer. Платформа позволяет тестировать Chromium, Firefox и WebKit с помощью единого API. Playwright создан для того, чтобы сделать возможным кросс-браузерную автоматизацию веб-сайтов.

3.3. Protractor

Protractor [14] — это фреймворк для сквозного тестирования приложений на базе AngularJS. Protractor запускает тесты для веб-приложения, открытого в браузере, взаимодействуя с ним как пользователь. Инструмент запускается поверх Selenium WebDriver. Помимо основных функций Selenium WebDriver, Protractor предлагает свои локаторы[6] и методы для захвата UI компонентов приложения.

[4]Сквозное тестирование - метод тестирования программного обеспечения, основанный на проверке рабочего процесса программы от начала до конца. Метод направлен на воспроизведение пользовательских сценариев.

[5]API (Application Programming Interface) - набор функций (методов), которые предоставляет система для доступа к её функциональности.

[6]Локатор — уникальная строка, которая идентифицирует элемент UI на странице.

3.4. Selenium

Selenium — проект с открытым исходным кодом, содержащий в себе инструменты и библиотеки, предназначенные для автоматизации браузера. История создания Selenium начинается в 2004 году, когда Jason Huggins и другие разработчики «ThoughtWorks» создали Selenium как внутренний инструмент компании. На данный момент проект развился в несколько отдельных продуктов с открытым исходным кодом:

- Selenium WebDriver [11],
- Selenium IDE [15],
- Selenium Grid [10].

3.4.1. Selenium WebDriver

Selenium WebDriver — инструмент для автоматизации действий пользователя в веб-браузере позволяет другим программам взаимодействовать с браузером с помощью тестов. Инструмент умеет масштабировать и распределять сценарии в среде браузера. В 2007 году разработчик Simon Stewart презентовал свой продукт под названием «Selenium WebDriver» на конференции GTAC-2007 [16]. Selenium WebDriver — это набор библиотек для различных языков программирования. На рисунке 1 клиент общается с драйвером[7] браузера сообщениями JSON через HTTP с использованием устаревшего протокола JSON Wire Protocol, в которых указаны команды для браузера.

У каждого браузера свои команды управления, поэтому для каждого браузера требуется свой драйвер. На данный момент драйвера популярных браузеров реализуют стандарт W3C WebDriver [17].

[7] Драйвер браузера — веб-сервер, который предоставляет API внешним программам для управления поведением браузера.

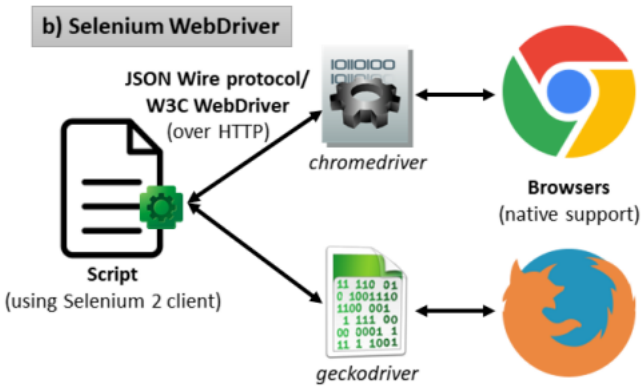

Рис. 1: Архитектура Selenium WebDriver [8].

3.4.2. Selenium IDE

Selenium IDE — инструмент для записи и воспроизведения взаимодействия пользователя с браузером. Shinya Kasatani из Японии заинтересовался Selenium и создал плагин Firefox под названием «Selenium IDE» в 2006 году. Инструмент предоставляет возможность записывать, редактировать и воспроизводить сценарии Selenium, тем самым Selenium IDE устраняет необходимость вручную выполнять повторяющиеся шаги. Инструмент может быть расширен с помощью использования плагинов. Поддерживает браузеры: Firefox, Google Chrome, Microsoft Edge.

3.5. Результат обзора

Для нас важно использовать кросс-браузерный инструмент с хорошей документацией. Это позволит использовать нашу программу в различных браузерах, повысит удобство и, как следствие, скорость процесса разработки.

Из сравнительной таблицы 2 видно, что Selenuim больше других подходит под эти требования. Проект до сих пор развивается, имеет большое сообщество и хорошую документацию. Поэтому за основу были выбраны инструменты Selenium, с помощью которых создаются многие RPA-системы. Однако никто не запрещает совмещать вышеперечисленные инструменты.

Инструмент	Puppeteer	Playwright	Selenium	Cypress
Разработчик	Google	Microsoft	ThoughtWorks	Cypress
Поддержка браузеров	Chrome-family browsers, Edge	Chrome-family browsers, Edge, Firefox, WebKit.	Chrome, Firefox, Edge, Internet Explorer, Opera, Safari.	Chrome-family browsers, Edge, Firefox
Открытый исходный код	+	+	+	Платные функции
Разработан	2012	2020	2004	2015
Запись действий	+	+	Selenium IDE	-

Рис. 2: Сравнительная таблица инструментов.

Использование Selenium IDE позволяет нам записывать действия пользователя и воспроизводить их. Однако Selenium использует локаторы страницы для записи действий, что не совсем надежно: тестирование таких рекордеров показало, что при воспроизведении уже записанных действий инструменты могут не найти нужных локаторов по разным причинам, и все дальнейшие команды прерываются. Поэтому при разработке механизма мы будем больше стараться взаимодействовать напрямую с системой, например, с помощью API Docsvision.

Также, если нам требуется взаимодействовать с другими системами, то «общение» по API происходит за пару кликов и гораздо меньшее время по сравнению с обычным способом, используя пользовательский интерфейс системы, даже если это записанный сценарий рекордера.

Например, нам требуется заниматься миграцией больших данных между системами или делать запросы в различные государственные сервисы для проверки информации.

4. Устройство платформы Docsvision

Платформа Docsvision состоит из:

- карточки — это объектная сущность платформы, например: «исходящий документ» или «задание», которая состоит из выбранных контролов, экземпляр карточки может быть в одном из состояний: создание, редактирование, просмотр и т.д;

- разметки — структура отображения карточки, создается для каждого состояния в КР (конструктор разметок), взаимно располагая контролы;

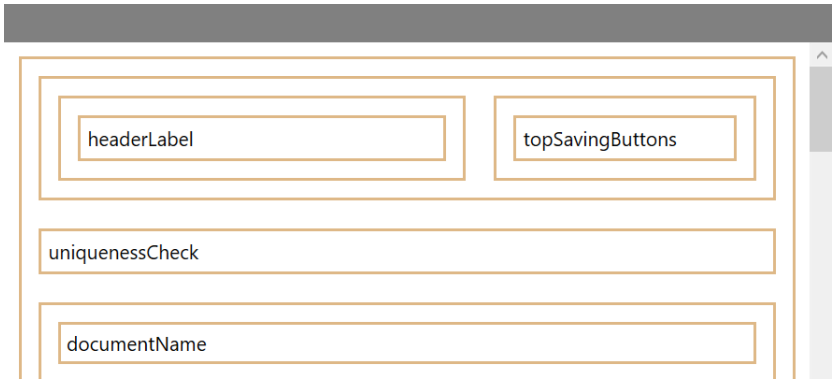

Рис. 3: Пример разметки в КР.

- контролов — элементы управления на базе ReactJS, которые содержат компонент React для отображения на документе, например: текстовые поля, чекбоксы, кнопки и другие более сложные контролы, взаимодействующие с сервером.

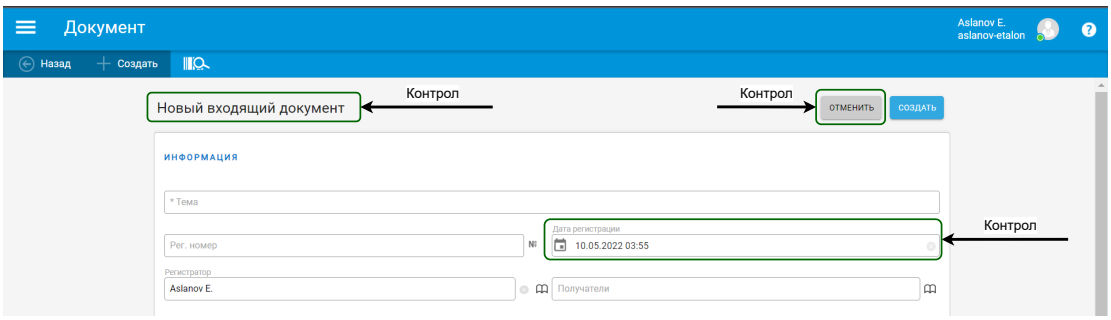

Рис. 4: Входящий документ DV.

5. Требования к механизму

Требуется разработать механизм в виде клиентского расширения веб-клиента, который будет автоматизировать типичные сценарии пользователей.

5.1. Нефункциональные требования

Перечислим нефункциональные требования:

- механизм в виде клиентского расширения,
- взаимодействие с API платформы,
- язык программирования JavaScript/TypeScript.

5.2. Функциональные требования

У пользователя должна быть возможность записывать повторяемые последовательности действий, которые он сможет воспроизводить в автоматическом режиме. Процесс взаимодействия пользователя с системой представлен на рисунках 5, 6.

Для автоматизации действий нам требуются такие возможности:

- запись действий пользователя в браузере;
- изменение действий (команд) записи;
- воспроизведение записи на другом документе;
- работа со снапшотами[8] :
  - создание/удаление снапшотов,
  - применение снапшота на разметке документа,
  - импорт снапшотов в хранилище,

[8]Снапшот — статическое состояние разметки в виде списка контролов и их состояний на данной разметке.

– экспорт снапшотов из хранилища.

На рис. 5 представлена диаграмма активности. Она демонстрирует возможное использование механизма.

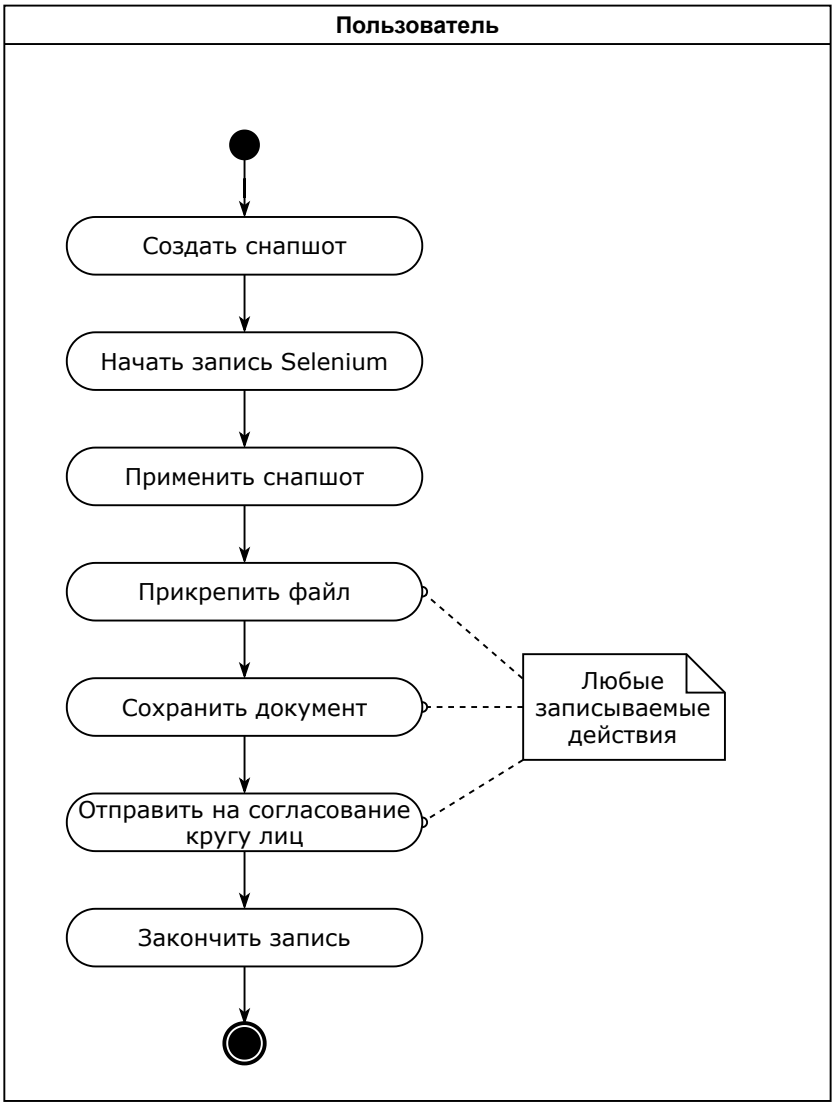

Рис. 5: Диаграмма активности UML.

На рис. 6 подробнее описан процесс работы со снапшотами в виде диаграммы последовательности.

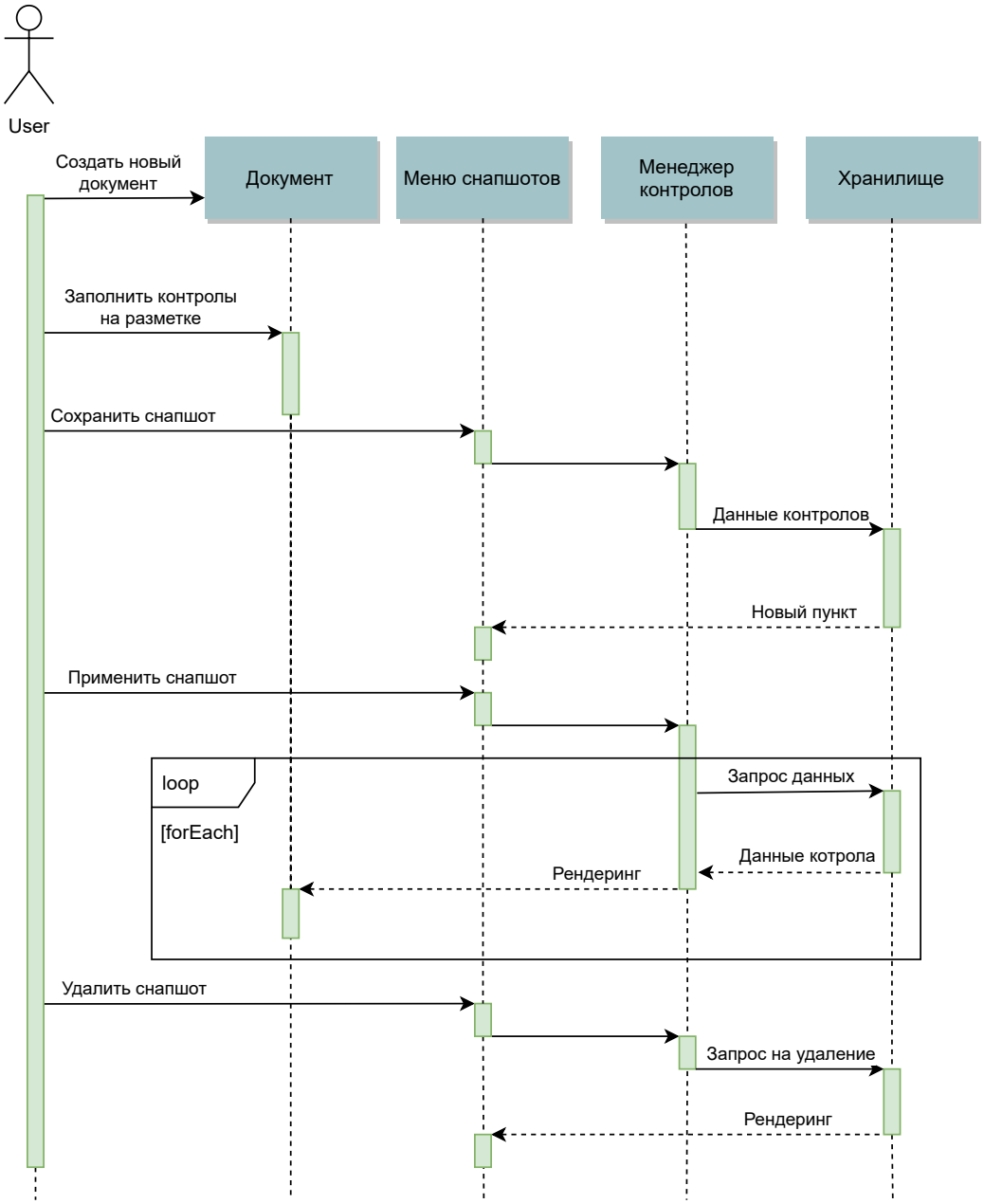

Рис. 6: Диаграмма последовательности UML.

6. Архитектура

Архитектура механизма представлена на рис. 7. На данный момент реализованы алгоритмы обработки основных типов контролов. В дальнейшем нам может потребоваться реализовать алгоритмы остальных контролов или новых контролов, которые добавятся позже другими разработчиками. Поэтому для улучшения расширяемости программного продукта был реализован шаблон проектирования «простая фабрика».

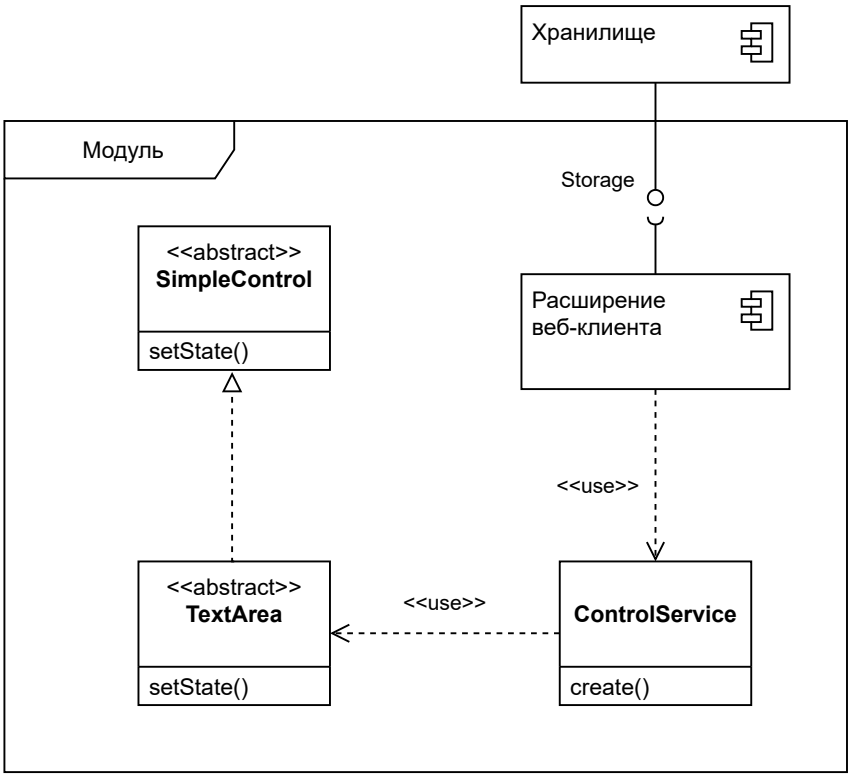

Рис. 7: Архитектура механизма.

7. Особенности реализации

Основным инструментом RPA является рекордер действий пользователей. При тестировании таких инструментов: Selenium IDE, Chrome Recorder, мы нашли несколько недостатков, связанных с ненадежностью использования рекордеров:

- ненахождение локаторов на странице,
- неожиданное поведение страницы,
- превышение тайм-аута выполнения команды.

Каждый из пунктов приводит к прерыванию воспроизведения записи (теста). Поэтому было решено использовать рекордер там, где это необходимо, в остальном использовать стандартные механизмы автоматизации, например, программный интерфейс приложения.

Для записи и воспроизведения действий пользователя на платформе мы решили создать механизм обработки контролов. Весь пользовательский интерфейс Docsvision представлен в виде разметок, «построенных» с помощью взаимного расположения контролов. Поэтому, когда пользователь взаимодействует с UI системы, он фактически взаимодействует с контролами.

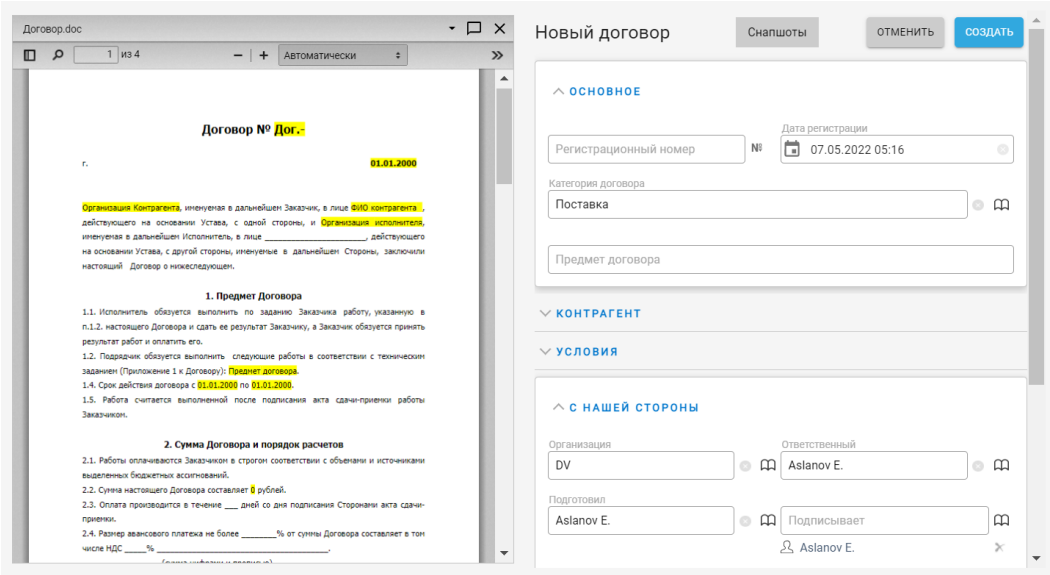

Рис. 8: Карточка договора DV.

Следовательно, мы можем запоминать изменения состояния контролов при активной записи действий. На данный момент мы умеем запоминать только статическое состояние разметки, поэтому мы назвали этот механизм «снапшотами». На рис. 8 изображена кнопка «Снапшоты», при нажатии которой откроется модальное окно со списком снапшотов.

Так как мы запоминаем только конечное состояние разметки, то нет необходимости активировать запись, мы просто заполняем разметку и сохраняем именованный снапшот в модальном окне представленном на рис. 9. Далее мы можем применить этот снапшот на любой другой разметке, но в этом есть смысл только если на разметке есть соответствующие контролы, поэтому мы сделали возможность фильтровать отображение снапшотов на 2 вида: все снапшоты и снапшоты, созданные на текущем типе разметки, например, снапшоты для создания договоров или редактирования заявок.

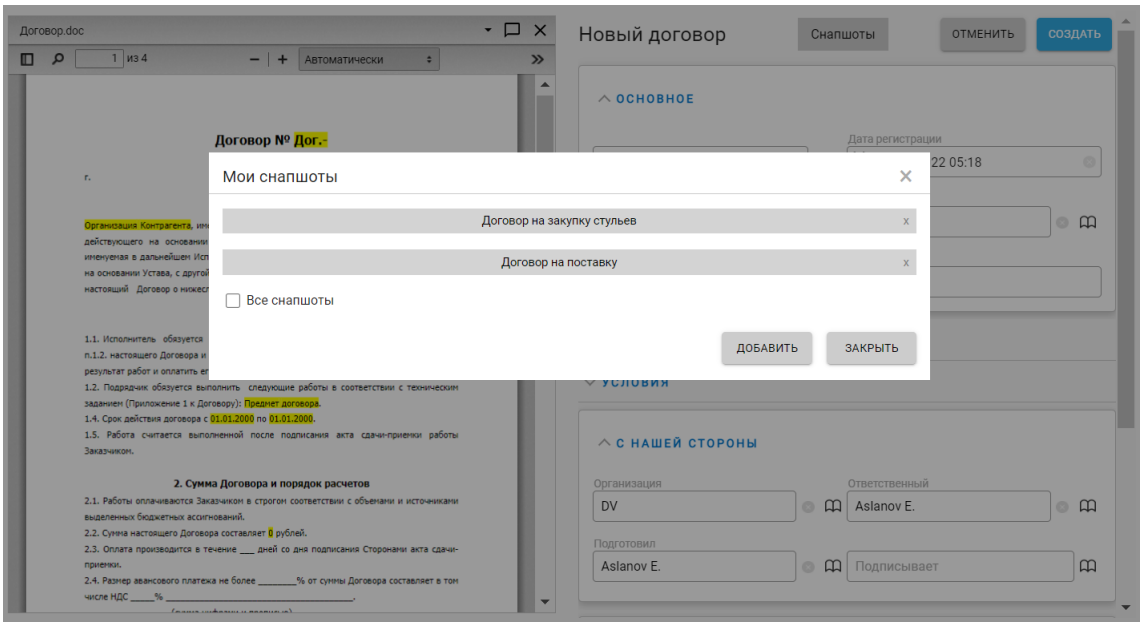

Рис. 9: Модальное окно снапшотов.

7.1. Дальнейшее представление механизма

Далее мы хотим реализовать возможность применения одного снапшота сразу к нескольким карточкам. Например, у нас IT-компания, в которой работает 100 разработчиков. У компании появилась необходимость поменять адрес базы данных. Вследствие этого, каждому сотруднику нужно пересоздать базу данных по этому адресу. Для этого начальник создает карточки заданий для каждого сотрудника с определенной датой окончания задачи. По окончании дедлайна часть сотрудников успешно справляется с задачей, а остальная часть не выполнила её. Начальнику нужно понять, почему задача не выполнена вовремя, для этого ему необходимо открыть все просроченные карточки с этой задачей и написать в поле комментария: "почему задача не выполнена?". Это занимает достаточно много времени для простой рутинной задачи. Поэтому мы можем применить снапшот для выбранных карточек, что запустит автоматический процесс их обработки.

Также мы хотим избавиться от использования сторонних рекордеров. Для этого нам нужно научиться запоминать не только статическое состояние контролов, но и запоминать различные переходы в виде нажатий на кнопки, так как кнопки тоже являются контролами, то это не должно быть сложной задачей, учитывая, что мы уже умеем обрабатывать состояния контролов.

8. Апробация

Целью апробации данной работы является оценка удобства использования механизма автоматизации повторяющихся действий пользователей. Для этого существуют различные средства, которые дают субъективное представление о простоте использования системы. Например, SUS (System Usability Scale) — шкала Лайкерта из 10 вопросов.

SUS создана John Brooke [3] в 1986 году и за многие годы использования зарекомендовала себя как надежная шкала для оценки удобства использования системы. В исследовании [9] различных юзабилити-метрик приводится, что в 90% случаев SUS имеет наименьшую разницу между результатами, полученными при малой и большой выборке участников.

При использовании SUS респондентам предлагается пройти опрос из 10 стандартных вопросов. Каждый вопрос оценивается по пятибальной шкале, которые варьируются от «категорически не согласен» до «полностью согласен». После происходит вычисление итогового балла для каждого респондента по методу SUS. Итоговые баллы умножаются на 2.5, чтобы они варьировались от 0 до 100. В таблице 1 представлены результаты опроса.

Респондент	Итоговый балл	Баллы SUS
1	30	75,0
2	31	77,5
3	26	65,0
4	31	77,5
5	25	62,5
6	30	75,0

Таблица 1: Результаты опроса пользователей.

В опросе участвовали группа аналитиков компании и один реальный пользователь Docsvision. Баллы SUS не являются процентами. В статье [1] приводится шкала нормализации баллов SUS для оценки продукта с помощью прилагательных, начиная от «худшее, что можно вообразить» и заканчивая «лучшее, что можно себе представить». Среднее значение баллов SUS ~ 72, что соответствует оценке «Хорошо».

9. Тестирование

9.1. Описание тестирования

Данная работа направлена на сокращение рутинных действий пользователя, поэтому целью тестирования нашего механизма является измерение длительности автоматического выполнения типичных действий на примере. Также мы хотим получить положительный ответ на вопрос: ”смогли ли мы добиться экономии времени при использовании механизма?”.

В качестве примера возьмем один из часто используемых процессов на платформе — заполнение договора с контрагентом. Допустим, у нас есть компания, которая занимается оптовой поставкой продуктов. Наш постоянный клиент — вымышленная компания ООО «СетьПродуктов», ежедневно отправляет нам заказ на 100 различных позиций в виде таблицы Excel. На рис. 10 представлена часть такой таблицы с требуемыми полями для заполнения договора.

Категория	Предмет договора	Контакт. Лицо	Сумма	Нач. договора	Кон. договора	Пролонгация
Поставка	Молоко "Милк" 1,5%	Иванов И. И.	100000	03.05.2022	03.06.2022	Нет
Поставка	Молоко "Милк" 2,5%	Иванов И. И.	98000	03.05.2022	03.06.2022	Нет
Поставка	Молоко "Милк" 3,2%	Иванов И. И.	120000	03.05.2022	03.06.2022	Нет
Поставка	Йогурт "Милк" 2,6%	Иванов И. И.	126000	03.05.2022	03.06.2022	Нет
Поставка	Йогурт "Милк" 3,2%	Иванов И. И.	136000	03.05.2022	03.06.2022	Нет
Поставка	Йогурт "Милк" 4,8%	Иванов И. И.	146000	03.05.2022	03.06.2022	Нет
Поставка	Филе индейки 1кг	Иванов И. И.	156000	03.05.2022	03.06.2022	Нет
Поставка	Фарш индейки 300г	Иванов И. И.	166000	03.05.2022	03.06.2022	Нет
Поставка	Фарш курин. 500 г	Иванов И. И.	176000	03.05.2022	03.06.2022	Нет

Рис. 10: Таблица поставок.

Теперь сотруднику нашей компании необходимо создать и заполнить 100 карточек договора значениями из данной таблицы и информацией со стороны нашей компании, например, о лицах, подписавших договор.

Для тестирования мы решили сократить количество договоров до 25, потому что тестирование создания и заполнения 100 договоров заняло бы много времени, а после результаты тестов можно умножить на 4. Тестирование проводилось 3-мя разными способами:

- ручное заполнение,
- заполнение с использованием рекордера,
- заполнение с использованием рекордера и снапшота.

Использование рекордера и снапшота подразумевает применение заранее подготовленного снапшота для заполнения одинаковых контролов (полей) среди всех договоров, например, значения поля "категория" на рис. 10. Выше на рис. 5 уже приводилась диаграмма активности для варианта совместного использования рекордера и снапшотов.

9.2. Результаты тестирования

В результате тестирования создания и заполнения 25 договоров 3-мя разными способами мы получили:

- ручной способ — 37,5 минут;
- использование рекордера — 10,4 минуты;
- использование рекордера и снапшота — 5 минут.

Конечно, результаты нельзя считать точными, так как измерение зависит от многих факторов: скорость интернета, скорость выполнения ручным способом теста отдельно взятого человека, характеристика компьютера и т.д. При этом мы старались поддерживать одинаковые условия для всех тестов, например, перед каждым тестом проверялось наличие постоянной скорости интернета.

Также нужно понимать, что 2-ой и 3-ий способ требуют предварительной настройки, а это дополнительные затраты времени, но при больших данных это незначительное время. А 1-ый способ не имеет линейную скорость роста времени, потому что человеку свойственно ошибаться и уставать.

Учитывая все вышесказанное мы можем с уверенностью сказать, что использование 3-его способа даёт наилучший результат для экономии времени пользователя.

10. Заключение

Таким образом, для реализации механизма выполнены следующие задачи:

- выполнен обзор существующих инструментов: основным инструментом был выбран рекордер Selenium IDE;
- сформулированы требования к механизму:
  - запись и воспроизведение действий пользователя в браузере,
  - создание/удаление снапшотов,
  - применение снапшота на разметке документа,
  - импорт снапшотов в хранилище,
  - экспорт снапшотов из хранилища;
- выработана архитектура механизма: для простого создания новых обработчиков контролов был реализован шаблон проектирования «простая фабрика»;
- реализованы основные требования, а также добавлена фильтрация отображения снапшотов в зависимости от текущей карточки;
- проведена апробация: результат апробации по шкале SUS соответствует оценке «Хорошо»;
- выполнено тестирование механизма: мы измерили время выполнения одного из популярных сценариев с использованием механизма и без - получили, что использование механизма может ускорить процесс минимум в 7,5 раз в данном случае.

В дальнейшем планируется реализовать групповое применение снапшотов для нескольких карточек и доработать механизм таким образом, чтобы отказаться от использования сторонних рекордеров.

Код проекта закрыт и принадлежит компании ООО «ДоксВижн».

Список литературы

[1] Bangor Aaron, Kortum Philip, Miller James. Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale // Journal of Usability studies. — 2009. — Vol. 4.

[2] ECM Wikipedia. — URL: https://en.wikipedia.org/wiki/Enterprise_content_management (online; accessed: 2021-12-22).

[3] John Brooke. SUS: A Retrospective // Journal of Usability Studies. — 2013. — 02. — P. 29–40. — URL: https://uxpajournal.org/sus-a-retrospective/ (online; accessed: 2022-04-24).

[4] Mary C. Lacity, Leslie Willcocks. What Knowledge Workers Stand to Gain from Automation. — 2015. — URL: https://hbr.org/2015/06/what-knowledge-workers-stand-to-gain-from-automation (online; accessed: 2022-04-21).

[5] Nathaniel Palmer. What is BPM? — 2018. — . — URL: https://bpm.com/what-is-bpm.

[6] Puppeteer GitHub. — URL: https://github.com/puppeteer/puppeteer (online; accessed: 2021-12-14).

[7] SAP insights. — URL: https://www.sap.com/cis/insights/what-is-digital-transformation.html (online; accessed: 2021-12-22).

[8] A Survey of the Selenium Ecosystem / Boni García, Micael Gallego, Francisco Gortázar, Mario Organero // Electronics. — 2020. — 06. — Vol. 9. — P. 1067. — URL: https://www.researchgate.net/publication/342581217_A_Survey_of_the_Selenium_Ecosystem (online; accessed: 2022-04-21).

[9] Tullis Thomas, Stetson Jacqueline. A Comparison of Questionnaires for Assessing Website Usability. — 2006. — 06.

[10] Документация Selenium Grid. — URL: https://www.selenium.dev/documentation/grid/ (online; accessed: 2022-04-24).

[11] Документация Selenium WebDriver. — URL: https://www.selenium.dev/documentation/webdriver/ (online; accessed: 2021-12-14).

[12] Домашняя страница Docsvision. — URL: https://docsvision.com/ (online; accessed: 2021-12-14).

[13] Домашняя страница Playwright. — URL: https://playwright.dev/ (online; accessed: 2021-12-14).

[14] Домашняя страница Protractor. — URL: https://www.protractortest.org/#/ (online; accessed: 2021-12-14).

[15] Домашняя страница Selenium IDE. — URL: https://www.selenium.dev/selenium-ide/ (online; accessed: 2022-04-24).

[16] Запись конференция GTAC 2007: Simon Stewart - Web Driver. — URL: https://www.youtube.com/watch?v=tGu1ud7hk5I (online; accessed: 2022-04-21).

[17] Стандарт W3C WebDriver. — URL: https://www.w3.org/TR/webdriver/ (online; accessed: 2022-04-21).